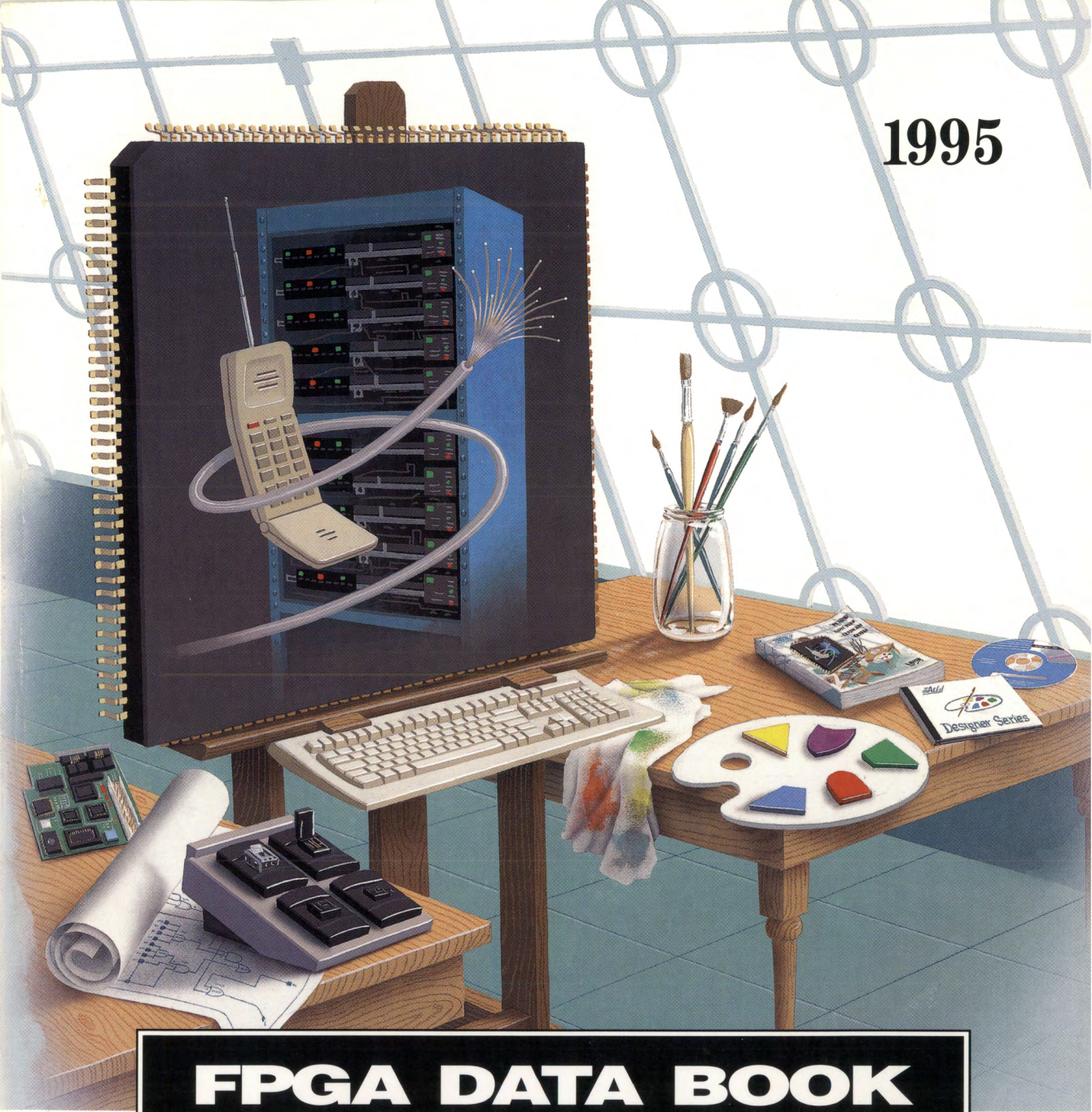
1995



FPGA DATA BOOK AND DESIGN GUIDE

Actel

Are you an artist?

We think so.

Like an artist, you start with a blank canvas. And on that canvas you craft a design that's unique, inspired and, hopefully, the most efficient answer to a problem. That answer, in turn, translates into a product that gives your company a competitive edge in the market.

We call that creative.

At Actel, we understand this process and what you must go through. And we're committed to support your creativity and goals in every way that we can.

First, we support you with proven FPGAs that offer a full range of choices in speed, capacity, pin outs, and packaging. So that you can get the performance you need at a price you can afford.

That's performance/price value from Actel, without painful trade-offs.

Second, we have the advanced design and development tools you need to achieve fast, flexible and predictable design, with guaranteed gate utilization.

Our tools let you capture your vision in silicon, exactly the way you want it.

Third, we provide you with knowledgeable and responsive technical support. Real engineers to answer your questions and help you complete your masterpiece on time, on budget and to your specifications.

You have the ideas and the vision.

We provide the brushes, the paint and the canvas.

Actel FPGA Data Book and Design Guide

1995

Contributors: Bruce Apkarian, Tara Anderson, Sam Beal, Larry Blessman, Sanjiv Desai, Tracy Fang, Gervais Fong, Steve Gurklys, Ken Hayes, Yousef Khalilollahi, Kamal Koraitem, Joel Landry, Ashena Massoumi, Jemelee Melendres, Warren Miller, Kirk Owyang, Hitesh Patel, Mike Sarpa, and Bruce Weyer.

Special thanks to many other contributors throughout Actel without whom this document could not have been completed.

Cover art was designed by Robert F. Tinney of Robert Tinney Graphics.

Order of Contents

◆ *Most commonly referenced documents*

Section 3: Application Notes—Design with Actel Devices (continued)

Section 4: Testing and Reliability

Section 5: PREP Data

Section 6: Macro Libraries

Section 7: Development Tools

◆ *Most commonly referenced documents*

Section 8: Synthesis

Section 9: Application Examples and Design Techniques

◆ Cross Reference By Vertical Segment

Section 10: Application Notes—Using Actel Tools

◆ *Most commonly referenced documents*

Section 10: Application Notes—Using Actel Tools (continued)

Section 11: Customer Case Histories

Section 12: Technical Support Services

◆ *Most commonly referenced documents*

How to Use This Data Book

The *Actel 1995 FPGA Data Book and Design Guide* contains a great deal of information. The following are suggested references in the Data Book, grouped by level of experience and design phase to help you find the information you need. Please refer to the order of contents for a complete listing of all documents in the Data Book. (The most commonly referenced documents in the order of contents are highlighted in blue for your convenience.)

Evaluating Actel

If you are not currently designing with Actel and are looking for a possible solution for your needs, the following documents will assist you with your evaluation.

- Introduction (pg ix)
- Component Selector Guide (pg 1-1)
- Actel PREP Benchmark Results (pg 5-17)
- ACT Family Reliability Report (pg 4-9)
- Application Examples and Design Technique Cross Reference by Vertical Segment (Section 9 order of contents)
- Switching Elements: The Key to FPGA Architecture (pg 3-1)
- Customer Case Histories (Section 11)
- Actel Offices (last three pages in the Data Book)

First-Time Users

If you are new to FPGAs, or particularly to Actel devices, we strongly recommend that you begin by reading the Data Book introduction (page ix). The introduction discusses the FPGA market and presents a short summary of the Actel architecture and the benefits it offers you. In addition, the following documents are particularly useful to get you started.

- Component Selector Guide (pg 1-1)
- Guidelines for Optimal FPGA Designs (pg 9-77)
- Application Note Cross-References (pg 3-1, 7-1, 9-1, 10-1)
- Technical Support Services (pg 12-1)

Current Users—What's New

Most of the documents in the 1995 Data Book have been updated. However, some of the new documents are especially noteworthy.

- 1200XL Family Data Sheet (pg 1-103)
- Speed Improvements of the 1200XL Family (pg 3-13)
- ACT 3 3.3 Volt Data Sheet (pg 1-223)
- System Level Timing Considerations of ACT 3 I/O Macros (pg 3-37)
- ACT 1 3.3 Volt Data Sheet (pg 1-35)
- ACT 1 Global Clock Network Enhancement (pg 3-77)
- Actel Mask Programmed Gate Array (MPGA) Data Sheet (pg 1-337)
- Design for Migration to Actel MPGAs (pg 3-59)
- Synthesis Application Notes (Section 8)
- Desinger Advantage System with Candence Concept/Verilog Design Kit (pg 7-9)
- New Tool Applications: Multichip Simulation for PROsim, ViewSim and QuickSim II (Section 10)
- Guidelines for Optimal FPGA Designs (pg 9-77)
- Implementing Multipliers with Actel FPGAs (pg 9-109)
- Designing High-Speed ATM Switch Fabrics by Using Actel FPGAs (pg 9-25)
- A 64 MHz RISC Coprocessor Using the A1460 and VHDL Entry (pg 9-59)
- A High Speed Graphics Processing Application Using FPGAs (pg 9-53)
- Using Actel FPGAs to Implement the 100/Mbit/s Ethernet Standard (pg 9-1)
- PCI to Generic Memory Bus Bridge (pg 9-37)
- Bulletin Board Application Files (Section 9 order of contents)

Before Starting a Design

Review the following documents before you begin a design. These documents allow you to take full advantage of Actel device features and software techniques to complete your design quickly and effectively.

- Component Selector Guide (pg 1-1)
- Guidelines for Optimal FPGA Designs (pg 9-77)
- Actel Device Data Sheets (Section 1)
- Application Notes—Design with Actel Devices (Section 3)

During a Design

The Data Book is an invaluable resource when you work on an Actel design. You may find the following documents of particular interest.

- Macro Libraries (Section 6)
- Actel Device Data Sheets (Section 1)
- Application Notes—Using Actel Tools (Section 10)
- Socket Recommendations for Actel FPGA Packages (pg 2-25)
- Technical Support Services (pg 12-1)

Expert Users

For experienced Actel users, the following documents offer useful, detailed information. These topics are intended for expert users with a good understanding of the Actel architecture.

- Using the Actel Debugger as a Functional Tester (pg 10-39)
- External Probe Circuit Control for Actel FPGAs (pg 10-27)

Introduction

The FPGA Market

Over the past five years, the market for field programmable gate arrays (FPGAs) has expanded rapidly. In fact, today it is the fastest growing segment of the $5.7 billion ASIC market. Many market research companies, including Dataquest Inc., predict that the FPGA market will grow to nearly one billion dollars by 1998—a compound annual growth rate of nearly 30 percent!

What's behind all this growth? Quite simply, with microprocessors and memory becoming more and more standardized, logic is quickly becoming the only area where an engineer can differentiate a product.

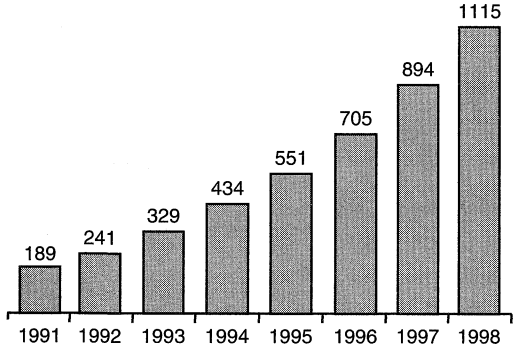


Figure 1 • FPGA Sales in Millions of Dollars

FPGAs Deliver What Other Logic Solutions Can't

Aside from product differentiation, a top priority for every engineer is shorter design cycles. Since FPGAs offer the design flexibility and capacity of gate arrays, with the convenience and speed of desktop programming, they satisfy this time-to-market demand. In fact, FPGAs deliver the fastest time-to-market solution of all logic integration methodologies.

Today's design engineers must also deal with the relentless demand to make products that are both cheaper and smaller. Here again, FPGAs are the solution. By integrating thousands of gates of random logic onto a single chip, FPGAs let engineers simultaneously reduce board space, component count and power requirements.

Actel Sales Growth

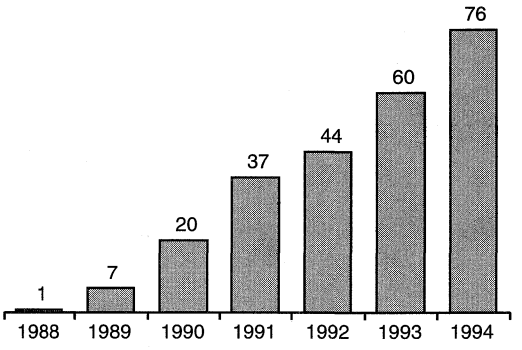


Figure 2 • Actel Sales in Millions of Dollars

Actel—A Leader in FPGAs and the Leader in Antifuse Technology

In 1988, Actel introduced the first antifuse-based FPGA. Since then, we've grown to become one of the world's leading suppliers of high-performance FPGAs. Through our leadership in antifuse technology and our unique, patented segmented routing architecture, we can offer you logic solutions that deliver significant advantages over competitive technologies.

High Performance at High Capacity

Actel antifuse FPGA technology is inherently fast, even at high capacities. Currently, Actel FPGAs can give you 250 MHz counter and datapath designs with 7.5 nanosecond clock-to-out speeds in 4,000 gate devices and 200 MHz designs in 10,000 gate devices. As a result, you can easily create very fast designs such as 250 MHz serial communications devices, 100 Mbit Ethernet receivers, and 67 MHz DRAM and DMA controllers.

Predictability

Actel FPGAs give you high performance—even after you've modified a fixed-pin design. Because of the small size of the Actel antifuse and the flexibility of our channeled architecture, you get high device utilization, design after design. With Actel FPGAs, you always have a high level of confidence that you can meet your design performance and capacity requirements.

Flexibility

The choices are all yours. With Actel FPGAs and design tools, you can shrink all or some of your system's logic onto a single, programmable device. You can also fix your I/O pins even before you've finished your design. In addition, Actel's automated design tools lets you make changes in your logic design quickly, without affecting other parts of the design. As a result, Actel FPGAs let you cut your overall system design time dramatically.

Cost Effectiveness

The Actel antifuse is significantly smaller than comparable SRAM and EPROM switching technologies. Consequently, Actel FPGA die sizes are dramatically smaller than SRAM and EPROM devices of comparable gate counts. Moreover, because of our extremely fine-grained architecture, Actel FPGAs offer much more efficient gate utilization than SRAM and EPROM devices. The bottom line is that Actel FPGAs give you more functionality in a much smaller space than competitive devices.

In addition, to make sure that you always have the most efficient and cost effective solution available for your application, we offer you four distinct families of FPGAs to choose from. These FPGA families provide a wide range of capacities and performance levels—up to 250 MHz and 10,000 gates today, and even more in the near future.

Finally, to assure you of a plentiful and reliable supply of Actel FPGAs to meet your production needs, we've established fabrication agreements with multiple foundries and manufacturing facilities.

Ease of Use

Actel Designer Series tools automate many of the repetitious tasks in logic design, which makes them exceptionally easy to use. You get high performance, predictability and flexibility through 100% automatic placement and routing—and automatic generation of counters, accumulators and other functional blocks. Designer Series tools also support standard schematic capture and high-level design through both Boolean and synthesis tools, as well as EDIF netlists. For maximum flexibility, our tools also run on Sun and HP 700 workstations and PCs, using industry-standard X-Window and Microsoft Windows user interfaces. So there's no need to change your equipment or the way you work.

Actel FPGAs

Currently, we offer four families of FPGAs that go from 1,200 to 10,000 gates in capacity—the ACT' 1, ACT 2, ACT 3 and 1200XL families. These FPGA families are based on our unique, patented antifuse and channeled FPGA architecture. Working together, these two technologies give you a unique combination of speed, capacity and cost effectiveness that's unmatched by competitive technologies and architectures.

The Actel Antifuse Advantage

The Actel antifuse is a nonvolatile, two-terminal element that delivers a low "on" resistance when programmed (see Figure 3). It provides the same wire-to-wire interconnect functions as "vias" do in mask-programmable gate arrays—and that transistor-based EPROM and RAM cells and metal fuses deliver in traditional programmable logic devices.

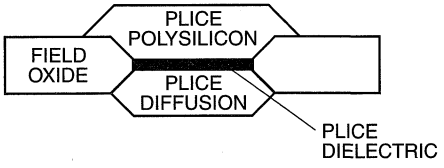


Figure 3 • The PLICE Antifuse Element

The Actel antifuse gives you major design advantages because of its size and electrical properties. The antifuse itself is small enough to fit within the width of a channeled routing track, which means it contributes virtually no die size overhead.

The Actel Patented Segmented Routing Architecture

Actel's segmented, channeled routing structure consists of various lengths of routing segments separated by antifuses, which are shown in the diagram as Horizontal Fuses. Some tracks have many short routing segments, while others give you direct interconnect across the chip. Because of the large number of routing tracks per channel, our automatic place and route tools have an enormous variety of segment lengths to choose from (see Figure 4).

The wide variety of interconnect segments minimizes the number of antifuses in any path. Short tracks connect closely placed modules—long tracks connect more distant modules. This highly flexible architecture ensures that no more than four antifuses are used in any path, with 90% of all interconnects needing just two antifuses.

The Highest, Most Predictable Performance

Actel offers the highest performance in the FPGA industry, including on-chip speeds up to 250 MHz, chip-to-chip speeds over 100 MHz, and 7.5 ns clock-to-out speeds. And all this performance is achieved with 100% automatic placement and routing.

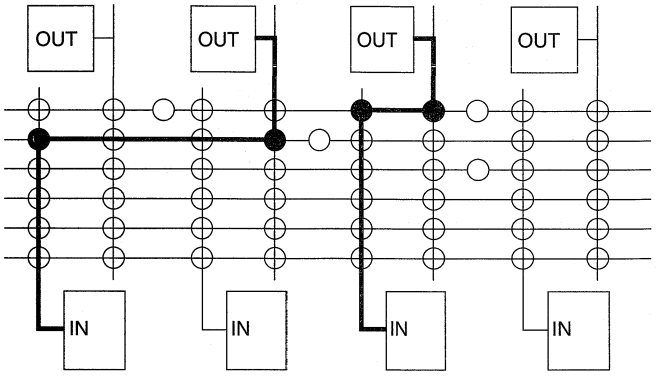


Figure 4 • ACT 1 Routing Example

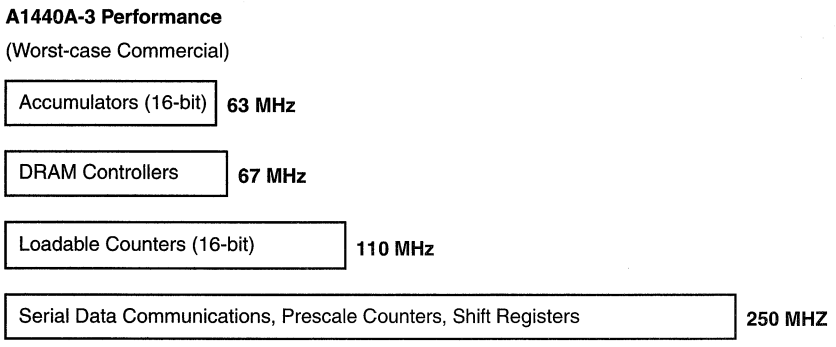


Figure 5 • A1440A-3 Performance (Worst-case Commercial)

The highly predictable performance of Actel FPGAs comes directly from our proprietary antifuse technology and segmented, channeled routing architecture. Our technology minimizes the total capacitance and resistance of each interconnect path because:

- The antifuse is inherently a low impedance device.
- 90% of all interconnect paths in an Actel FPGA are made with just two antifuses, which offers minimal resistance.
- Because of the varying lengths of routing tracks, you can always choose the shortest route to connect any logic modules.

The ACT 3 Family: High Performance, High Capacity Solution

In 1991, Actel introduced the largest FPGA in the industry, the 8000-gate A1280. Soon thereafter in 1993, we introduced the industry's first high performance, high capacity programmable solutions, the A1460A and A14100A. Now, in 1995, we're introducing ACT 3 speed grades, which offer clock rates two times faster than before. These new devices give you up to 10,000 gate-array gates, up to 228 user I/O, on-chip performance up to 250 MHz and clock-to-out speeds of 7.5 nanoseconds. And they're all fully supported by the latest automatic placement and routing tools.

	High Performance		High Capacity		
	On-Chip	Clock-Out	Gates	Flip-Flops	User I/Os
A1415A-3	250 MHz	7.5 ns	1,500	264	80
A1425A-3	250 MHz	7.5 ns	2,500	360	100
A1440A-3	250 MHz	8.5 ns	4,000	568	140
A1460A-3	200 MHz	9.0 ns	6,000	768	168
A14100A-3	200 MHz	9.5 ns	10,000	1,153	228

ACT 3 family devices fully illustrate the advantages of the antifuse technology. The low impedance of the antifuse gives you high performance designs, even at high gate capacities. No other programmable technology can offer you this unique combination of high speed and high capacity—and opens up so many new design possibilities in computers, graphics, high speed telecommunications and other high speed and high I/O applications.

With ACT 3 devices, you can create functional blocks, such as:

- 250 MHz shift registers and prescale counters
- 110 MHz loadable counters
- 63 MHz accumulators

This lets you design applications such as:

- 250 MHz serial communications
- 150 MHz graphic controllers
- 100 Mbit ethernet receivers
- 67 MHz DRAM and DMA controllers

The Actel Value Proposition

Today, Actel can give you the most cost competitive solutions in programmable logic. Our technology allows us to produce the industry's smallest die sizes for comparable capacity devices. And our strategic partnerships assure you of dependable supply of FPGAs, at industry leading costs:

- Very small die sizes achievable because of our PLICE antifuse technology
- Our competitive, multiple foundry fabrication agreements
- Leadership in process technology and architectural design

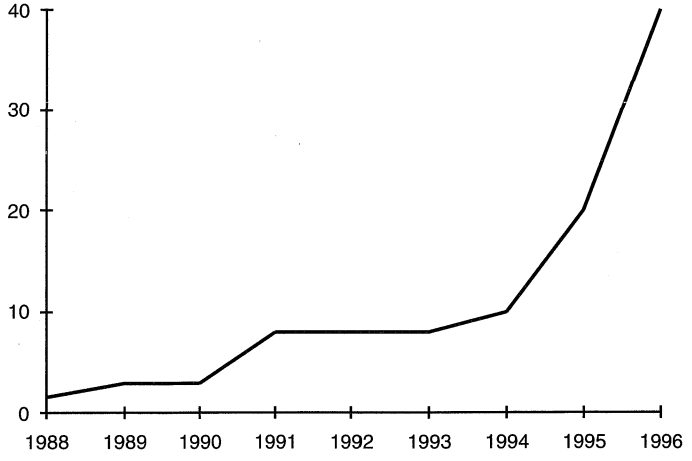


Figure 6 • Largest Gate Capacity Device, In Thousands of Gates

The 1200XL Family: High Performance at a Very Low Cost

The 1200XL family offers higher performance at half the price of our very popular ACT 2 family. By using 0.6 micron technology, along with clock distribution and I/O enhancements, the 1200XL family gives you from 2,500 to 8,000 gates, high speeds, and a wide variety of low cost package solutions.

1200XL devices are very popular for gate intensive designs such as bus interfacing and data switches. Within the 1200XL family, you'll find devices that deliver up to 135 MHz counter and datapath performance, and have up to 624 dedicated flip-flops and 140 user I/Os. And, as with all Actel devices, you can count on predictable performance and high gate utilization.

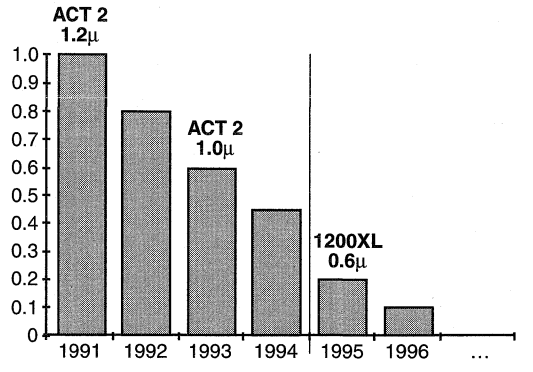


Figure 7 • Relative Price Progression

	High Performance		High Capacity		
	On-Chip	Clock-Out	Gates	Flip-Flops	User I/Os
A1225XL-1	135 MHz	10.0 ns	2,500	231	83
A1240XL-1	130 MHz	10.5 ns	4,000	348	104
A1280XL-1	115 MHz	11.0 ns	8,000	624	140

ACT 1 Family: Low Cost Logic Integration

If you're tired of having multiple logic devices on your board, integrate them with a single ACT 1 device. Our A1010B gives you up to 1200 gate array gates and lets you integrate up to 30 TTL or 12 PLD devices onto a single device costing less than $8.00, in high volumes. Its partner, the A1020B, gives you up to 2000 gates and can integrate up to 50 TTL or 20 PLD devices.

With the ACT 1 family, you have nonvolatile solutions for up to 30 MHz system speeds. As with all Actel devices, ACT 1 designs can be integrated using our ACTmap Boolean conversion tool, allowing immediate translation of PALASM, CUPL or ABEL files. You can also enter your design using standard CAE or EDIF formats.

	Capacity		Replaces These Packages		Performance
	Gates	User I/Os	TTL	20-pin PALs	On-Chip
A1010B-3	1,200	57	30	12	70 MHz
A1020B-3	2,000	69	50	20	70 MHz

Actel's High Capacity Proposition

In 1991 Actel introduced the 8,000 gate ACT 2 Family, which had twice the capacity of the competition. As a result, designs needing up to 200 TTL, or 80 PLDs, could be integrated into a single device. In 1994, we introduced very high performance up to 10,000 gates with our ACT 3 family. In 1995, Actel will introduce a new family of FPGAs that offer up to 40,000 gate array gates. As always, we continue to deliver predictable performance and gate utilization, while using 100% automatic place and route tools.

	High Performance		High Capacity		
	On-Chip	Clock-Out	Gates	Flip-Flops	User I/Os
A1225A-2	105 MHz	18.0 ns	2,500	231	83
A1240A-2	100 MHz	18.0 ns	4,000	348	104
A1280A-2	85 MHz	18.5 ns	8,000	624	140

The Actel Portable Solution

Actel FPGAs also give you excellent low power and small outline solutions ideal for portable computing. These devices include both our high performance ACT 3 3.3V family as well as our low cost ACT 1 3.3V family. Combined with very compact VQFP and TQFP packages, our 3.3V solution is perfect for most board and power constrained designs. With our 3.3V solutions, you get:

- Very low power consumption—the lowest active power consumption for programmable devices
- Very high logic integration—4000 gates in 100-pin VQFP packages 6000 gates in 176-pin TQFP packages

The ACT 3 3.3V Family: High Performance, Low Power Solution

Our 3.3V ACT 3 family gives you very high performance for portable and power constrained applications—and up to 4000 gates in 100-pin VQFP and 6000 gates in 176-pin TQFP packages. This high level of logic integration is perfect for creating designs such as 100 MHz serial communications and 33 MHz DRAM controllers, with greater than 60 MHz chip-to-chip speeds.

	High Performance		High Capacity		
	On-Chip	Clock-Out	Gates	Flip-Flops	User I/Os
A14V15A	100 MHz	13.0 ns	1,500	264	80
A14V25A	100 MHz	13.0 ns	2,500	360	100
A14V40A	100 MHz	14.4 ns	4,000	568	140
A14V60A	75 MHz	15.0 ns	6,000	768	168
A14V100A	75 MHz	15.7 n	10,000	1,153	228

ACT 1 3.3V Family: Low Cost, Low Power Solution

Actel's 3.3V ACT 1 family offers a very low cost, low power logic integration path for traditional PAL and TTL designers. These devices support clock rates to 50 MHz and are available in 80-pin VQFP as well as PLCC packages.

	Capacity		Replaces These Packages		Performance
	Gates	User I/Os	TTL	20-pin PALs	On-Chip
A10V10B-3	1,200	57	30	12	50 MHz
A10V20B-3	2,000	69	50	20	50 MHz

Actel's Designer and Designer Advantage Systems

The Actel Designer and Designer Advantage systems are high-productivity, computer-aided engineering environments for Actel FPGA devices. The Designer System can handle all Actel FPGA families up to 2500-gates, while the Designer Advantage System handles every Actel device, from the smallest to the largest.

Depending on whether you work on a PC or a workstation, you can use either the Designer or Designer Advantage systems outfitted with our new, easy-to-use Microsoft Windows (PC) or X Window (workstation) graphical user interfaces. A key advantage of either system is that they let you take designs from concept to silicon in hours, without costly non recurring engineering (NRE) expenses.

Each Designer Series system gives you placement and routing, timing verification, and programming interface software, as well as interfaces to many popular CAE and electronic design automation platforms such as Cadence, Mentor Graphics, OrCAD, Synopsys, Data I/O, Exemplar Logic, IST, and Viewlogic. In addition, both Designer Series systems support programming on Data I/O's Unisite and 3900 series programmers.

Another benefit is that both systems are very open. You can continue using your existing design environments to enter VHDL descriptions, capture schematics, enter PAL or Boolean equations, simulate, and perform timing analysis. That lets you design Actel FPGAs in your chosen environment.

Both the Designer and Designer Advantage systems include the following tools for speeding your design to market:

- **Automatic Place and Route.** For all devices and any design. 100% of the nets are routed automatically resulting in 95% to 100% logic module utilization for all Actel FPGA devices. In addition, the abundant routing resources of Actel FPGAs assure tight delay distributions and predictable design performance.
- **DirectTime™ Layout.** This tool allows you to specify overall operating frequency and duty cycles, as well as individual path delays. (Available as an option to Designer Series 3.0.)
- **DirectTime™ Analyzer.** This innovative addition to the Designer Series 3.0 toolset gives you interactive timing analysis and a graphic display of design path performance. It also provides you with a spreadsheet view of the timing results and lets you optimize your design further using the new DirectTime Layout Tool. (Available in Designer Series 3.0.)
- **ACTmap VHDL Synthesis.** This tool gives you a powerful, entry-level VHDL synthesis solution, optimized for designers who are familiar with PAL design methodologies. ACTmap VHDL offers both VHDL design language entry, as well as optimization of Actel, EDIF, and PALASM design files.
- **ACTgen Macro Builder.** This time-saving tool automatically generates counters, adders and other structural blocks, eliminating hours and days of tedious manual labor.

Design Creation

Designer Series systems let you create Actel FPGA designs several different ways. No matter if you use schematic capture, synthesis or PAL design methodologies to implement your logic, the Designer Series supports you.

Component Data

Section 1: Component Data

Component Selector Guide

5 Volt Component Selector Guide

Device	Pkg[1]	# Pins	Speed Option[2]	Temp.[3]	User I/O	Gates	Flip-Flops		Equiv. Pkgs.	
							Dedicated	Max.	TTLs	20-pin PALs
A1010B	PG	84	Std, –1	C, M, B	57	1,200	0	147	30	12
	PL	44	Std, –1, –2, –3	C, I	34	1,200	0	147	30	12
	PL	68	Std, –1, –2, –3	C, I	57	1,200	0	147	30	12
	PQ	100	Std, –1, –2, –3	C, I	57	1,200	0	147	30	12
	VQ	80	Std, –1, –2, –3	C	57	1,200	0	147	30	12
A1020B	CQ	84	Std, –1**	C, M, B, E	69	2,000	0	273	50	20
	PG	84	Std, –1	C, M, B	69	2,000	0	273	50	20
	PL	44	Std, –1, –2, –3	C, I	34	2,000	0	273	50	20
	PL	68	Std, –1, –2, –3	C, I	57	2,000	0	273	50	20
	PL	84	Std, –1, –2, –3	C, I	69	2,000	0	273	50	20
	PQ	100	Std, –1, –2, –3	C, I	69	2,000	0	273	50	20
	VQ	80	Std, –1, –2, –3	C	69	2,000	0	273	50	20
A1225A	PG	100	Std, –1, –2	C	83	2,500	231	382	63	25
	PL	84	Std, –1, –2	C, I	72	2,500	231	382	63	25
	PQ	100	Std, –1, –2	C, I	83	2,500	231	382	63	25
	VQ	100	Std, –1, –2	C	83	2,500	231	382	63	25
A1225XL*	PG	100	Std, –1	C	83	2,500	231	382	63	25
	PL	84	Std, –1	C, I	72	2,500	231	382	63	25
	PQ	100	Std, –1	C, I	83	2,500	231	382	63	25
	VQ	100	Std, –1	C	83	2,500	231	382	63	25
A1240A	PG	132	Std, –1, –2***	C, M, B	104	4,000	348	568	100	40
	PL	84	Std, –1, –2	C, I	72	4,000	348	568	100	40
	PQ	144	Std, –1, –2	C, I	104	4,000	348	568	100	40
	TQ	176	Std, –1, –2	C	104	4,000	348	568	100	40
A1240XL*	PG	132	Std, –1	C	104	4,000	348	568	100	40
	PL	84	Std, –1	C, I	72	4,000	348	568	100	40
	PQ	144	Std, –1	C, I	104	4,000	348	568	100	40
	TQ	176	Std, –1	C	104	4,000	348	568	100	40

* consult Actel for availability

** consult Actel for '–1' speed Extended Flow (E) product availability

*** offered for Commercial (C) devices only

5 Volt Component Selector Guide (Continued)

Device	Pkg[1]	# Pins	Speed Option[2]	Temp.[3]	User I/O	Gates	Flip-Flops		Equiv. Pkgs.	
							Dedicated	Max.	TTLs	20-pin PALs
A1280A	CQ	172	Std, –1**	C, M, B, E	140	8,000	624	998	200	80
	PG	176	Std, –1, –2***	C, M, B	140	8,000	624	998	200	80
	PL	84	Std, –1, –2	C, I	72	8,000	624	998	200	80
	PQ	160	Std, –1, –2	C, I	125	8,000	624	998	200	80
	TQ	176	Std, –1, –2	C	140	8,000	624	998	200	80
A1280XL*	PG	176	Std, –1	C	140	8,000	624	998	200	80
	PL	84	Std, –1	C, I	72	8,000	624	998	200	80
	PQ	160	Std, –1	C, I	125	8,000	624	998	200	80
	TQ	176	Std, –1	C	140	8,000	624	998	200	80
A1415A	PG	100	Std, –1, –2, –3	C	80	1,500	264	312	38	15
	PL	84	Std, –1, –2, –3	C, I	70	1,500	264	312	38	15
	PQ	100	Std, –1, –2, –3	C, I	80	1,500	264	312	38	15
	VQ	100	Std, –1, –2, –3	C	80	1,500	264	312	38	15
A1425A	CQ	132	Std, –1	C, M, B	100	2,500	360	435	63	25
	PG	133	Std, –1, –2***, –3***	C, M, B	100	2,500	360	435	63	25
	PL	84	Std, –1, –2, –3	C, I	70	2,500	360	435	63	25
	PQ	100	Std, –1, –2, –3	C, I	80	2,500	360	435	63	25
	PQ	160	Std, –1, –2, –3	C, I	100	2,500	360	435	63	25
	VQ	100	Std, –1, –2, –3	C	83	2,500	360	435	63	25
A1440A	PG	175	Std, –1, –2, –3	C	140	4,000	568	706	100	40
	PL	84	Std, –1, –2, –3	C, I	70	4,000	568	706	100	40
	PQ	160	Std, –1, –2, –3	C, I	131	4,000	568	706	100	40
	TQ	176	Std, –1, –2, –3	C	140	4,000	568	706	100	40
	VQ	100	Std, –1, –2, –3	C	83	4,000	568	706	100	40
A1460A	BG	225	Std, –1, –2, –3	C, I	168	6,000	768	976	150	60
	CQ	196	Std, –1	C, M, B	168	6,000	768	976	150	60
	PG	207	Std,–1,–2***,–3***	C, M, B	168	6,000	768	976	150	60
	PQ	160	Std, –1, –2, –3	C, I	131	4,000	568	706	100	40
	PQ	208	Std, –1, –2, –3	C, I	167	6,000	768	976	150	60
	TQ	176	Std, –1, –2, –3	C	151	6,000	768	976	150	60
A14100A	BG	313	Std, –1, –2, –3	C, I	228	10,000	1,153	1493	250	100
	CQ	256	Std, –1	C, M, B	228	10,000	1,153	1493	250	100
	PG	257	Std,–1,–2***,–3***	C, M, B	228	10,000	1,153	1,493	250	100
	RQ	208	Std, –1, –2, –3	C, I	175	10,000	1,153	1493	250	100

* consult Actel for availability

** consult Actel for '–1' speed Extended Flow (E) product availability

*** offered for Commercial (C) devices only

3.3 Volt Component Selector Guide

Device	Pkg	# Pins	Speed Option	Temp.	User I/O	Gates	Flip-Flops		Equiv. Pkgs.	
							Dedicated	Max.	TTLs	20-pin PALs
A10V10B	PL	68	Standard	C	57	1,200	0	147	30	12
	VQ	80	Standard	C	57	1,200	0	147	30	12
A10V20B	PL	68	Standard	C	57	2,000	0	273	50	20
	PL	84	Standard	C	69	2,000	0	273	50	20
	VQ	80	Standard	C	69	2,000	0	273	50	20
A14V15A	PL	84	Standard	C	70	1,500	264	312	38	15
	VQ	100	Standard	C	80	1,500	264	312	38	15
A14V25A	PL	84	Standard	C	70	2,500	360	435	63	25
	VQ	100	Standard	C	83	2,500	360	435	63	25
A14V40A	PL	84	Standard	C	70	4,000	568	706	100	40
	TQ	176	Standard	C	140	4,000	568	706	100	40
	VQ	100	Standard	C	83	4,000	568	706	100	40
A14V60A	PQ	208	Standard	C	167	6,000	768	976	150	60
	TQ	176	Standard	C	151	6,000	768	976	150	60
A14V100A	BG	313	Standard	C	228	10,000	1,153	1493	250	100
	RQ	208	Standard	C	175	10,000	1,153	1493	250	100

Note 1: *Package types:*
CQ = Ceramic Quad Flat Packs
PG = Ceramic Pin Grid Arrays
PL = Plastic J-Leaded Chip Carriers
PQ = Plastic Quad Flat Packs
TQ = Very Thin (1.0mm) Quad Flatpacks
VQ = Very Thin (1.0mm) Quad Flatpacks

Note 2: *Speed Options*
Std = Standard Speed
–1 = Approximately 15% faster than Standard
–2 = Approximately 25% faster than Standard
–3 = Approximately 35% faster than Standard

Note 3: *Temperature Range:*
C = Commercial Temperature (0 to +75 C)
I = Industrial (-40 to +85 C)
M = Military (-55 to +125 C)
B = MIL-STD-883C
E = Extended Flow

ACT™ 1
Field Programmable Gate Arrays

Features

- Up to 2000 Gate Array Gates (6000 PLD equivalent gates)
- Replaces up to 50 TTL Packages
- Replaces up to twenty 20-Pin PAL® Packages
- Design Library with over 250 Macro Functions
- Gate Array Architecture Allows Completely Automatic Place and Route
- Up to 547 Programmable Logic Modules
- Up to 273 Flip-Flops
- Flip-Flop Toggle Rates to 100 MHz
- Two In-Circuit Diagnostic Probe Pins Support Speed Analysis to 25 MHz
- Built-In High Speed Clock Distribution Network
- I/O Drive to 10 mA
- Nonvolatile, User Programmable
- Logic Fully Tested Prior to Shipment
- Fabricated in 1.0 micron CMOS technology

Description

The ACT™ 1 family of field programmable gate arrays (FPGAs) offers a variety of package, speed, and application combinations. Devices are implemented in silicon gate, 1-micron two-level metal CMOS, and they employ Actel's PLICE® antifuse technology. The unique architecture offers gate array flexibility, high performance, and instant turnaround through user programming. Device utilization is typically 95 percent of available logic modules.

ACT 1 devices also provide system designers with unique on-chip diagnostic probe capabilities, allowing convenient testing and debugging. Additional features include an on-chip clock driver with a hardwired distribution network. The network provides efficient clock distribution with minimum skew.

The user-definable I/Os are capable of driving at both TTL and CMOS drive levels. Available packages include plastic and ceramic J-leaded chip carriers, ceramic and plastic quad flatpacks, and ceramic pin grid array.

A security fuse may be programmed to disable all further programming and to protect the design from being copied or reverse engineered.

Product Family Profile

Device	A1010B	A1020B
Capacity		
Gate Array Equivalent Gates	1,200	2,000
PLD Equivalent Gates	3,000	6,000
TTL Equivalent Packages	30	50
20-Pin PAL Equivalent Packages	12	20
Logic Modules	295	547
Flip-Flops (maximum)	147	273
Routing Resources		
Horizontal Tracks/Channel	22	22
Vertical Tracks/Column	13	13
PLICE Antifuse Elements	112,000	186,000
User I/Os (maximum)	57	69
Packages:	44 PLCC	44 PLCC
	68 PLCC	68 PLCC
		84 PLCC
	100 PQFP	100 PQFP
	80 VQFP	80 VQFP
	84 CPGA	84 CQFP
		84 CPGA
Performance		
Flip-Flop Toggle Rate (maximum)	100 MHz	100 MHz
System Speed (maximum)	40 MHz	40 MHz

Note:

1. *See Product Plan on page 1-8 for package availability.*

The Designer and Designer Advantage™ Systems

The ACT 1 device family is supported by Actel's Designer and Designer Advantage Systems, allowing logic design implementation with minimum effort. The systems offer Microsoft® Windows™ and XWindows™ graphical user interfaces and integrate with the resident CAE system to provide a complete gate array design environment: schematic capture, simulation, fully automatic place and route, timing verification, and device programming. The systems also include the ACTmap™ optimization and synthesis tool and the ACTgen™ Macro Builder, a powerful macro function generator for counters, adders, and other structural blocks. The systems are available for 386/486/Pentium™ PC and for HP™ and Sun™ workstations and for running Viewlogic®, Mentor Graphics®, Cadence™, and OrCAD™.

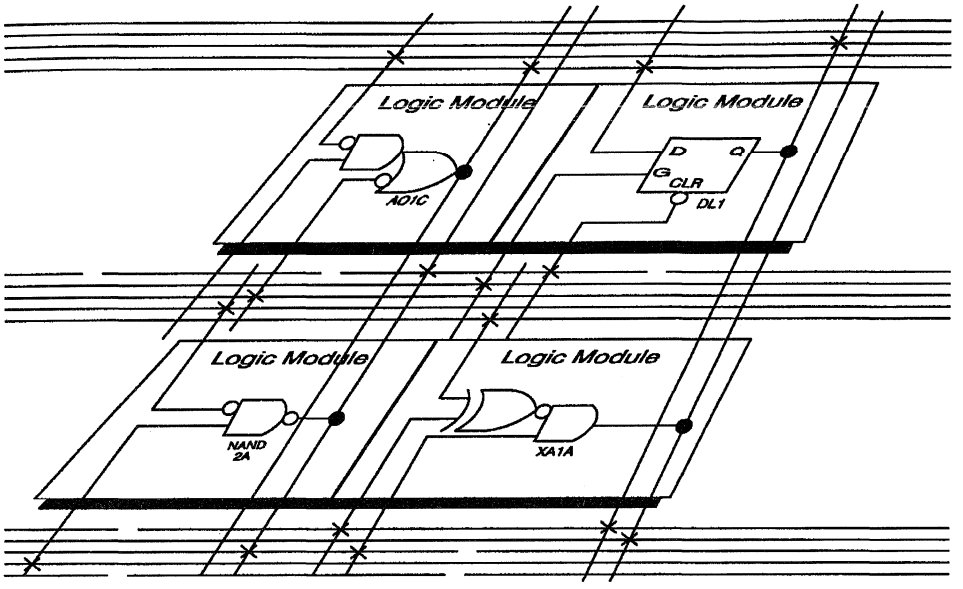


Figure 1 • Partial View of an ACT 1 Device

ACT 1 Device Structure

A partial view of an ACT 1 device (Figure 1) depicts four logic modules and distributed horizontal and vertical interconnect tracks. PLICE antifuses, located at intersections of the horizontal and vertical tracks, connect logic module inputs and outputs. During programming, these antifuses are addressed and programmed to make the connections required by the circuit application.

The ACT 1 Logic Module

The ACT 1 logic module is an 8-input, one-output logic circuit chosen for the wide range of functions it implements and for its efficient use of interconnect routing resources (Figure 2).

The logic module can implement the four basic logic functions (NAND, AND, OR, and NOR) in gates of two, three, or four inputs. Each function may have many versions, with different combinations of active-low inputs. The logic module can also implement a variety of D-latches, exclusivity functions, AND-ORs, and OR-ANDs. No dedicated hardwired latches or flip-flops are required in the array, since latches and flip-flops may be constructed from logic modules wherever needed in the application.

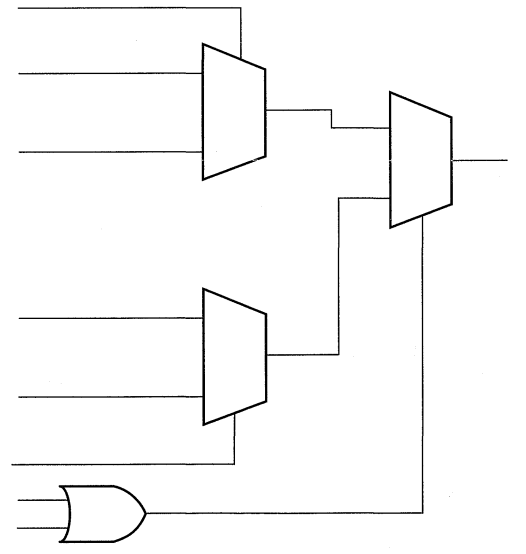


Figure 2 • ACT 1 Logic Module

I/O Buffers

Each I/O pin is available as an input, output, three-state, or bidirectional buffer. Input and output levels are compatible with standard TTL and CMOS specifications. Outputs sink or source 10 mA at TTL levels. See Electrical Specifications for additional I/O buffer specifications.

Device Organization

ACT 1 devices consist of a matrix of logic modules arranged in rows separated by wiring channels. This array is surrounded by a ring of peripheral circuits including I/O buffers, testability circuits, and diagnostic probe circuits providing real-time diagnostic capability. Between rows of logic modules are routing channels containing sets of segmented metal tracks with PLICE antifuses. Each channel has 22 signal tracks. Vertical routing is permitted via 13 vertical tracks per logic module column. The resulting network allows arbitrary and flexible interconnections between logic modules and I/O modules.

Probe Pin

ACT 1 devices have two independent diagnostic probe pins. These pins allow the user to observe any two internal signals by entering the appropriate net name in the diagnostic software. Signals may be viewed on a logic analyzer using Actel's Actionprobe® diagnostic tools. The probe pins can also be used as user-defined I/Os when debugging is finished.

ACT 1 Array Performance

Temperature and Voltage Effects

Worst-case delays for ACT 1 arrays are calculated in the same manner as for masked array products. A typical delay parameter is multiplied by a derating factor to account for temperature, voltage, and processing effects. However, in an ACT 1 array, temperature and voltage effects are less dramatic than with masked devices. The electrical characteristics of module interconnections on ACT 1 devices remain constant over voltage and temperature fluctuations.

As a result, the total derating factor from typical to worst-case for a standard speed ACT 1 array is only 1.19 to 1, compared to 2 to 1 for a masked gate array.

Logic Module Size

Logic module size also affects performance. A mask programmed gate array cell with four transistors usually implements only one logic level. In the more complex logic module (similar to the complexity of a gate array macro) of an ACT 1 array, implementation of multiple logic levels within a single module is possible. This eliminates interlevel wiring and associated RC delays. The effect is termed "net compression."

Ordering Information

A1010 B – 2 PL 84 C

- C
  - Application (Temperature Range)
    - C = Commercial (0 to +70°C)
    - I = Industrial (–40 to +85°C)
    - M = Military (–55 to +125°C)
    - B = MIL-STD-883
    - E = Extended Flow
- 84
  - Package Lead Count
- PL
  - Package Type
    - PL = Plastic J-Leaded Chip Carriers
    - PQ = Plastic Quad Flatpacks
    - CQ = Ceramic Quad Flatpack
    - PG = Ceramic Pin Grid Array
    - VQ = Very Thin Quad Flatpack
- 2
  - Speed Grade
    - Blank = Standard Speed
    - –1 = Approximately 15% faster than Standard
    - –2 = Approximately 25% faster than Standard
    - –3 = Approximately 35% faster than Standard
- B
  - Die Revision
    - B = 1.0 micron CMOS Process
- A1010
  - Part Number
    - A1010 = 1200 Gates
    - A1020 = 2000 Gates

Product Plan

	Speed Grade*				Application				
	Std	–1	–2	–3	C	I	M	B	E
A1010B Device									
44-pin Plastic Leaded Chip Carrier (PL)	✔	✔	✔	✔	✔	✔	—	—	—
68-pin Plastic Leaded Chip Carrier (PL)	✔	✔	✔	✔	✔	✔	—	—	—
100-pin Plastic Quad Flatpack (PQ)	✔	✔	✔	✔	✔	✔	—	—	—
80-pin Very Thin (1.0 mm) Quad Flatpack (VQ)	✔	✔	✔	✔	✔	—	—	—	—
84-pin Ceramic Pin Grid Array (PG)	✔	✔	—	—	✔	—	✔	✔	—
A1020B Device									
44-pin Plastic Leaded Chip Carrier (PL)	✔	✔	✔	✔	✔	✔	—	—	—
68-pin Plastic Leaded Chip Carrier (PL)	✔	✔	✔	✔	✔	✔	—	—	—
84-pin Plastic Leaded Chip Carrier (PL)	✔	✔	✔	✔	✔	✔	—	—	—
100-pin Plastic Quad Flatpack (PQ)	✔	✔	✔	✔	✔	✔	—	—	—
80-pin Very Thin (1.0 mm) Quad Flatpack (VQ)	✔	✔	✔	✔	✔	—	—	—	—
84-pin Ceramic Pin Grid Array (PG)	✔	✔	—	—	✔	—	✔	✔	—
84-pin Ceramic Quad Flatpack (CQ)	✔	✔	—	—	✔	—	✔	✔	✔

Applications: C = Commercial
I = Industrial
M = Military
B = MIL-STD-883
E = Extended Flow

Availability: 4 = Available
P = Planned
—= Not Planned

* Speed Grade: –1= Approx. 15% faster than Standard
–2= Approx. 25% faster than Standard
–3= Approx. 35% faster than Standard

Device Resources

			User I/Os				
Device	Logic Modules	Gates	44-pin	68-pin	80-pin	84-pin	100-pin
A1010B	295	1200	34	57	57	57	57
A1020B	547	2000	34	57	69	69	69

Pin Description

CLK **Clock (Input)**

TTL Clock input for global clock distribution network. The Clock input is buffered prior to clocking the logic modules. This pin can also be used as an I/O.

DCLK **Diagnostic Clock (Input)**

TTL Clock input for diagnostic probe and device programming. DCLK is active when the MODE pin is HIGH. This pin functions as an I/O when the MODE pin is LOW.

GND **Ground**

Input LOW supply voltage.

I/O **Input/Output (Input, Output)**

I/O pin functions as an input, output, three-state, or bidirectional buffer. Input and output levels are compatible with standard TTL and CMOS specifications. Unused I/O pins are automatically driven LOW by the ALS software.

MODE **Mode (Input)**

The MODE pin controls the use of multifunction pins (DCLK, PRA, PRB, SDI). When the MODE pin is HIGH, the special functions are active. When the MODE pin is LOW, the pins function as I/O.

NC **No Connection**

This pin is not connected to circuitry within the device.

PRA **Probe A (Output)**

The Probe A pin is used to output data from any user-defined design node within the device. This independent diagnostic pin is used in conjunction with the Probe B pin to allow real-time diagnostic output of any signal path within the device. The Probe A pin can be used as a user-defined I/O when debugging has been completed. The pin's probe capabilities can be permanently disabled to protect the programmed design's confidentiality. PRA is active when the MODE pin is HIGH. This pin functions as an I/O when the MODE pin is LOW.

PRB **Probe B (Output)**

The Probe B pin is used to output data from any user-defined design node within the device. This independent diagnostic pin is used in conjunction with the Probe A pin to allow real-time diagnostic output of any signal path within the device. The Probe B pin can be used as a user-defined I/O when debugging has been completed. The pin's probe capabilities can be permanently disabled to protect the programmed design's confidentiality. PRB is active when the MODE pin is HIGH. This pin functions as an I/O when the MODE pin is LOW.

SDI **Serial Data Input (Input)**

Serial data input for diagnostic probe and device programming. SDI is active when the MODE pin is HIGH. This pin functions as an I/O when the MODE pin is LOW.

V_{CC} **Supply Voltage**

Input HIGH supply voltage.

V_{PP} **Programming Voltage**

Input supply voltage used for device programming. This pin must be connected to V_{CC} during normal operation.

Absolute Maximum Ratings[1]

Free air temperature range

Symbol	Parameter	Limits	Units
V_{CC}	DC Supply Voltage[2]	–0.5 to +7.0	Volts
V_I	Input Voltage	–0.5 to V_{CC} +0.5	Volts
V_O	Output Voltage	–0.5 to V_{CC} +0.5	Volts
I_{IO}	I/O Sink/Source Current[3]	±20	mA
T_{STG}	Storage Temperature	–65 to +150	°C

Notes:

1. *Stresses beyond those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. Exposure to absolute maximum rated conditions for extended periods may affect device reliability. Device should not be operated outside the Recommended Operating Conditions.*
2. *$V_{PP} = V_{CC}$, except during device programming.*
3. *Device inputs are normally high impedance and draw extremely low current. However, when input voltage is greater than V_{CC} + 0.5 V or less than GND – 0.5 V, the internal protection diode will be forward biased and can draw excessive current.*

Recommended Operating Conditions

Parameter	Commercial	Industrial	Military	Units
Temperature Range[1]	0 to +70	–40 to +85	–55 to +125	°C
Power Supply Tolerance	±5	±10	±10	%V_{CC}

Note:

1. *Ambient temperature (T_A) used for commercial and industrial; case temperature (T_C) used for military.*

Electrical Specifications

Symbol	Parameter	Commercial		Industrial		Military		Units
		Min.	Max.	Min.	Max.	Min.	Max.	
V_{OH}[1]	(I_{OH} = –10 mA)[2]	2.4						V
	(I_{OH} = –6 mA)	3.84						V
	(I_{OH} = –4 mA)			3.7		3.7		V
V_{OL}[1]	(I_{OL} = 10 mA)[2]		0.5					V
	(I_{OL} = 6 mA)		0.33		0.40		0.40	V
V_{IL}		–0.3	0.8	–0.3	0.8	–0.3	0.8	V
V_{IH}		2.0	V_{CC} + 0.3	2.0	V_{CC} + 0.3	2.0	V_{CC} + 0.3	V
Input Transition Time t_R, t_F[2]			500		500		500	ns
C_{IO} I/O Capacitance[2, 3]			10		10		10	pF
Standby Current, I_{CC}[4] (typical = 1 mA)			3		10		20	mA
Leakage Current[5]		–10	10	–10	10	–10	10	µA

Notes:

1. *Only one output tested at a time. V_{CC} = min.*
2. *Not tested, for information only.*
3. *Includes worst-case 84-pin PLCC package capacitance. V_{OUT} = 0 V, f = 1 MHz.*
4. *Typical standby current = 1 mA. All outputs unloaded. All inputs = V_{CC} or GND.*
5. *V_O, V_{IN} = V_{CC} or GND.*

Package Thermal Characteristics

The device junction to case thermal characteristics is θjc, and the junction to ambient air characteristics is θja. The thermal characteristics for θja are shown with two different air flow rates. Maximum junction temperature is 150°C.

A sample calculation of the maximum power dissipation for an 84-pin plastic leaded chip carrier at commercial temperature is as follows:

$$\frac{\text{Max junction temp. (°C)} - \text{Max commercial temp. (°C)}}{\theta ja\ (°C/W)} = \frac{150°C - 70°C}{44°C/W} = 1.82\ W$$

Package Type	Pin Count	θja Still Air	θja 300 ft/min	Units
Plastic J-Leaded Chip Carrier	44	45	35	°C/W
	68	38	29	°C/W
	84	37	28	°C/W
Plastic Quad Flatpack	100	48	40	°C/W
Very Thin (1.0 mm) Quad Flatpack	80	43	35	°C/W
Ceramic Pin Grid Array	84	33	20	°C/W
Ceramic Quad Flatpack	84	40	30	°C/W

General Power Equation

$$P = [I_{CC}standby + I_{CC}active] * V_{CC} + I_{OL} * V_{OL} * N + I_{OH} * (V_{CC} - V_{OH}) * M$$

Where:

I_{CC}standby is the current flowing when no inputs or outputs are changing.

I_{CC}active is the current flowing due to CMOS switching.

I_{OL}, I_{OH} are TTL sink/source currents.

V_{OL}, V_{OH} are TTL level output voltages.

N equals the number of outputs driving TTL loads to V_{OL}.

M equals the number of outputs driving TTL loads to V_{OH}.

An accurate determination of N and M is problematical because their values depend on the family type, design details, and on the system I/O. The power can be divided into two components: static and active.

Static Power Component

Actel FPGAs have small static power components that result in lower power dissipation than PALs or PLDs. By integrating multiple PALs/PLDs into one FPGA, an even greater reduction in board-level power dissipation can be achieved.

The power due to standby current is typically a small component of the overall power. Standby power is calculated below for commercial, worst case conditions.

I_{CC}	V_{CC}	Power
3 mA	5.25 V	15.8 mW

The static power dissipated by TTL loads depends on the number of outputs driving high or low and the DC load current. Again, this value is typically small. For instance, a 32-bit bus sinking 4 mA at 0.33 V will generate 42 mW with all outputs driving low, and 140 mW with all outputs driving high. The actual dissipation will average somewhere between as I/Os switch states with time.

Active Power Component

Power dissipation in CMOS devices is usually dominated by the active (dynamic) power dissipation. This component is frequency dependent, a function of the logic and the

external I/O. Active power dissipation results from charging internal chip capacitances of the interconnect, unprogrammed antifuses, module inputs, and module outputs, plus external capacitance due to PC board traces and load device inputs. An additional component of the active power dissipation is the totem-pole current in CMOS transistor pairs. The net effect can be associated with an equivalent capacitance that can be combined with frequency and voltage to represent active power dissipation.

Equivalent Capacitance

The power dissipated by a CMOS circuit can be expressed by the Equation 1.

$$\text{Power (uW)} = C_{EQ} * V_{CC2} * F \quad (1)$$

Where:

C_{EQ} is the equivalent capacitance expressed in pF.

V_{CC} is the power supply in volts.

F is the switching frequency in MHz.

Equivalent capacitance is calculated by measuring I_{CC}active at a specified frequency and voltage for each circuit component of interest. Measurements have been made over a range of frequencies at a fixed value of V_{CC}. Equivalent capacitance is frequency independent so that the results may be used over a wide range of operating conditions. Equivalent capacitance values are shown below.

C_{EQ} Values for Actel FPGAs

Modules (C_{EQM})	3.7
Input Buffers (C_{EQI})	22.1
Output Buffers (C_{EQO})	31.2
Routed Array Clock Buffer Loads (C_{EQCR})	4.6

To calculate the active power dissipated from the complete design, the switching frequency of each part of the logic must be known. Equation 2 shows a piece-wise linear summation over all components.

$$\text{Power} = V_{CC}^2 * [(m * C_{EQM} * f_m)_{modules} + (n * C_{EQI} * f_n)_{inputs} + (p * (C_{EQO} + C_L) * f_p)_{outputs} + 0.5 * (q_1 * C_{EQCR} * f_{q1})_{routed\_Clk1} + (r_1 * f_{q1})_{routed\_Clk1}] \quad (2)$$

Where:

m = Number of logic modules switching at fm

n = Number of input buffers switching at fn

p = Number of output buffers switching at fp

q_1 = Number of clock loads on the first routed array clock (All families)

r_1 = Fixed capacitance due to first routed array clock (All families)

C_{EQM} = Equivalent capacitance of logic modules in pF

C_{EQI} = Equivalent capacitance of input buffers in pF

C_{EQO} = Equivalent capacitance of output buffers in pF

C_{EQCR} = Equivalent capacitance of routed array clock in pF

C_L = Output lead capacitance in pF

f_m = Average logic module switching rate in MHz

f_n = Average input buffer switching rate in MHz

f_p = Average output buffer switching rate in MHz

f_{q1} = Average first routed array clock rate in MHz (All families)

Fixed Capacitance Values for Actel FPGAs (pF)

Device Type	r_1 routed_Clk1
A1010B	41.4
A1020B	68.6

Determining Average Switching Frequency

To determine the switching frequency for a design, you must have a detailed understanding of the data input values to the circuit. The following guidelines are meant to represent worst-case scenarios so that they can be generally used to predict the upper limits of power dissipation. These guidelines are as follows:

Logic Modules (m)	90% of modules
Inputs switching (n)	#inputs/4
Outputs switching (p)	#outputs/4
First routed array clock loads (q_1)	40% of modules
Load capacitance (C_L)	35 pF
Average logic module switching rate (f_m)	F/10
Average input switching rate (f_n)	F/5
Average output switching rate (f_p)	F/10
Average first routed array clock rate (f_{q1})	F

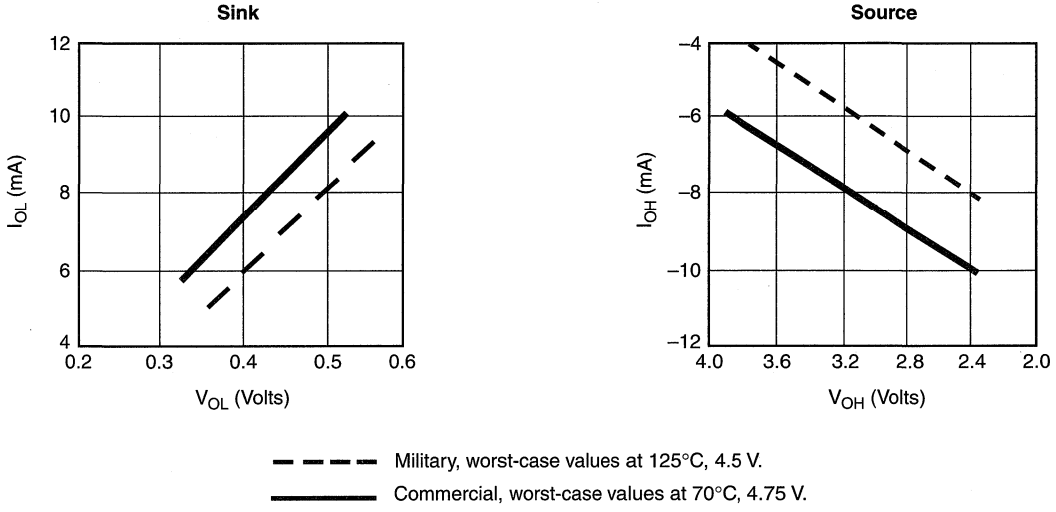
Functional Timing Tests

AC timing for logic module internal delays is determined after place and route. The ALS Timer utility displays actual timing parameters for circuit delays. ACT 1 devices are AC tested to a "binning" circuit specification.

The circuit consists of one input buffer + n logic modules + one output buffer (n = 16 for A1010B; n = 28 for A1020B). The logic modules are distributed along two sides of the device, as inverting or non-inverting buffers. The modules are connected through programmed antifuses with typical capacitive loading.

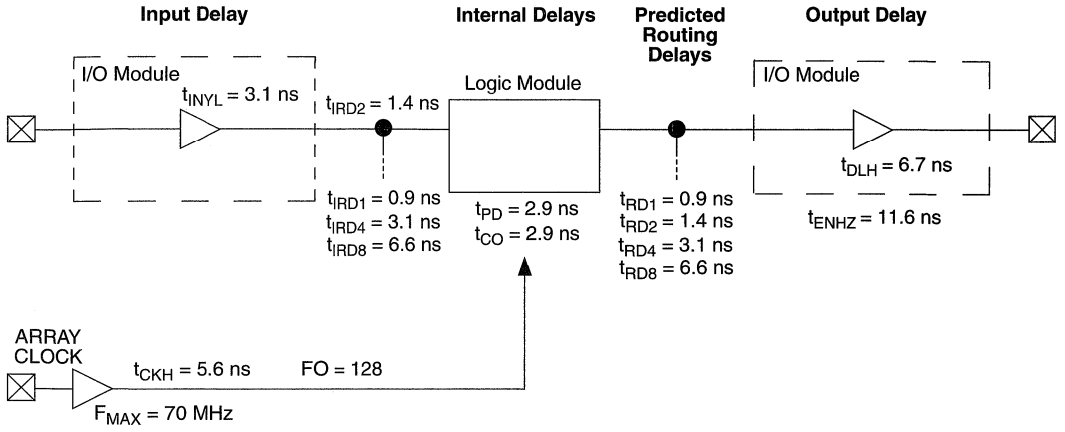
Propagation delay [$t_{PD} = (t_{PLH} + t_{PHL})/2$] is tested to the following AC test specifications.

Output Buffer Performance Derating



Note: *The above curves are based on characterizations of sample devices and are not completely tested on all devices.*

ACT 1 Timing Module*



** Values shown for ACT 1 '–3 speed' devices at worst-case commercial conditions.*

Predictable Performance: Tight Delay Distributions

Propagation delay between logic modules depends on the resistive and capacitive loading of the routing tracks, the interconnect elements, and the module inputs being driven. Propagation delay increases as the length of routing tracks, the number of interconnect elements, or the number of inputs increases.

From a design perspective, the propagation delay can be statistically correlated or modeled by the fanout (number of loads) driven by a module. Higher fanout usually requires some paths to have longer routing tracks.

The ACT 1 family delivers a very tight fanout delay distribution. This tight distribution is achieved in two ways: by decreasing the delay of the interconnect elements and by decreasing the number of interconnect elements per path.

Actel's patented PLICE antifuse offers a very low resistive/capacitive interconnect. The ACT 1 family's antifuses, fabricated in 1.0 micron lithography, offer nominal levels of 200 ohms resistance and 7.5 femtofarad (fF) capacitance per antifuse.

The ACT 1 fanout distribution is also tight due to the low number of antifuses required for each interconnect path. The ACT 1 family's proprietary architecture limits the number of antifuses per path to a maximum of four, with 90% of interconnects using two antifuses.

Logic Module + Routing Delay, by Fanout (ns) (Worst-Case Commercial Conditions)

Family	FO=1	FO=2	FO=3	FO=4	FO=8
'STD'	5.9	6.7	7.8	9.3	14.7
'–1' speed	5.0	5.7	6.6	7.9	12.5
'–2' speed	4.5	5.1	5.9	7.0	11.1
'–3' speed	3.8	4.3	5.0	6.0	9.5

Timing Characteristics

Timing characteristics for ACT 1 devices fall into three categories: family dependent, device dependent, and design dependent. The input and output buffer characteristics are common to all ACT 1 family members. Internal routing delays are device dependent. Design dependency means actual delays are not determined until after placement and routing of the user design is complete. Delay values may then be determined by using the ALS Timer utility or performing simulation with post-layout delays.

Critical Nets and Typical Nets

Propagation delays are expressed only for typical nets, which are used for initial design performance evaluation. Critical net delays can then be applied to the most time-critical paths. Critical nets are determined by net property assignment prior to placement and routing. Up to 6% of the nets in a design may be designated as critical, while 90% of the nets in a design are typical.

Long Tracks

Some nets in the design use long tracks. Long tracks are special routing resources that span multiple rows, columns, or modules. Long tracks employ three and sometimes four antifuse connections. This increases capacitance and resistance, resulting in longer net delays for macros connected to long tracks. Typically, up to 6% of nets in a fully utilized device require long tracks. Long tracks contribute approximately 5 ns to 10 ns delay. This additional delay is represented statistically in higher fanout (FO=8) routing delays in the datasheet specifications section.

Timing Derating

A best case timing derating factor of 0.45 is used to reflect best case processing. Note that this factor is relative to the "standard speed" timing parameters, and must be multiplied by the appropriate voltage and temperature derating factors for a given application.

Timing Derating Factor (Temperature and Voltage)

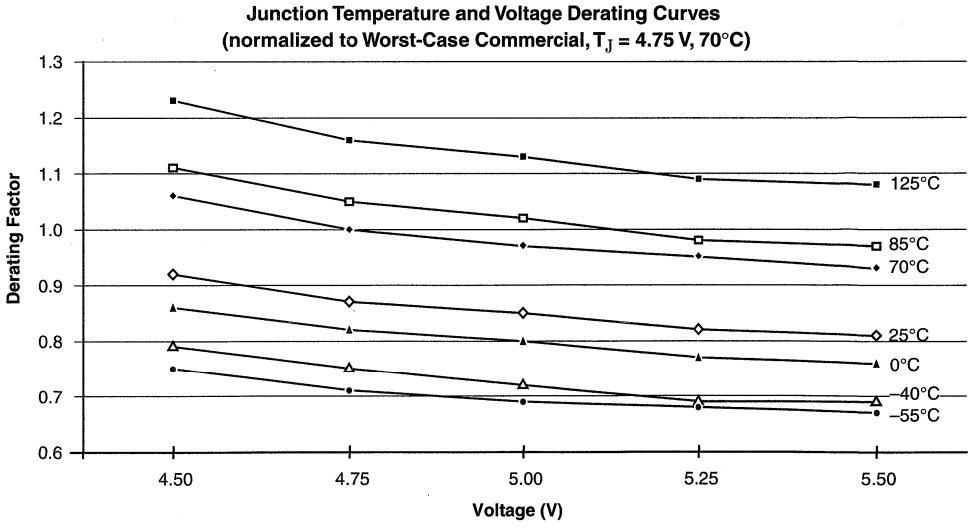
	Industrial		Military	
	Min.	Max.	Min.	Max.
(Commercial Minimum/Maximum Specification) x	0.69	1.11	0.67	1.23

Timing Derating Factor for Designs at Typical Temperature (T_J = 25°C) and Voltage (5.0 V)

(Commercial Maximum Specification) x	0.85

Temperature and Voltage Derating Factors (normalized to Worst-Case Commercial, T_J = 4.75 V, 70°C)

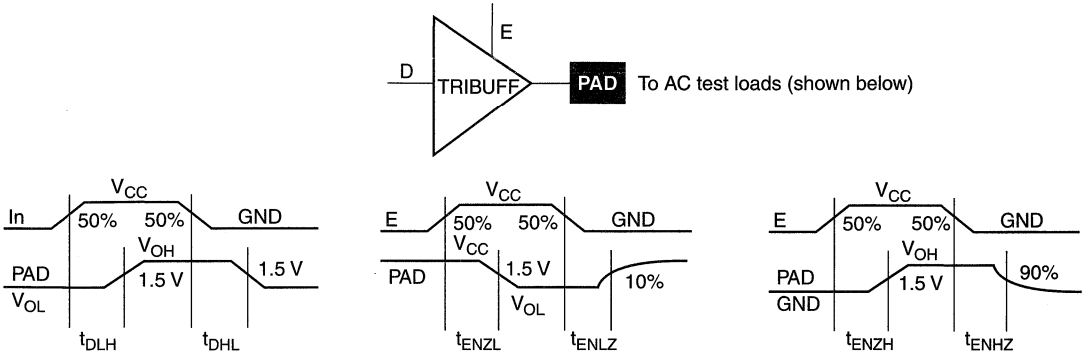
	–55	–40	0	25	70	85	125
4.50	0.75	0.79	0.86	0.92	1.06	1.11	1.23
4.75	0.71	0.75	0.82	0.87	1.00	1.05	1.16
5.00	0.69	0.72	0.80	0.85	0.97	1.02	1.13
5.25	0.68	0.69	0.77	0.82	0.95	0.98	1.09
5.50	0.67	0.69	0.76	0.81	0.93	0.97	1.08



Note: *This derating factor applies to all routing and propagation delays.*

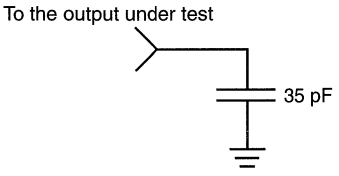
Parameter Measurement

Output Buffer Delays

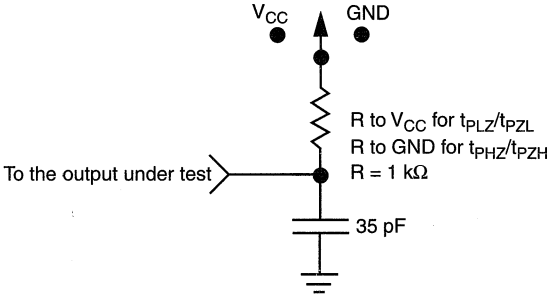


AC Test Loads

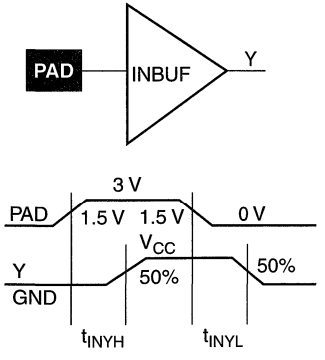
Load 1
(Used to measure propagation delay)



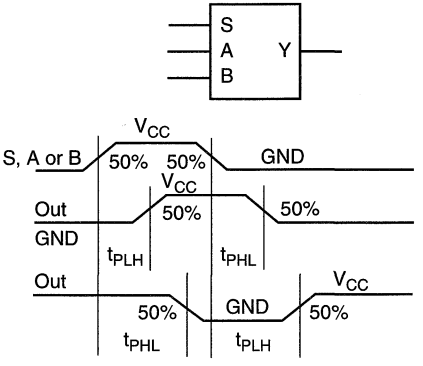
Load 2
(Used to measure rising/falling edges)



Input Buffer Delays

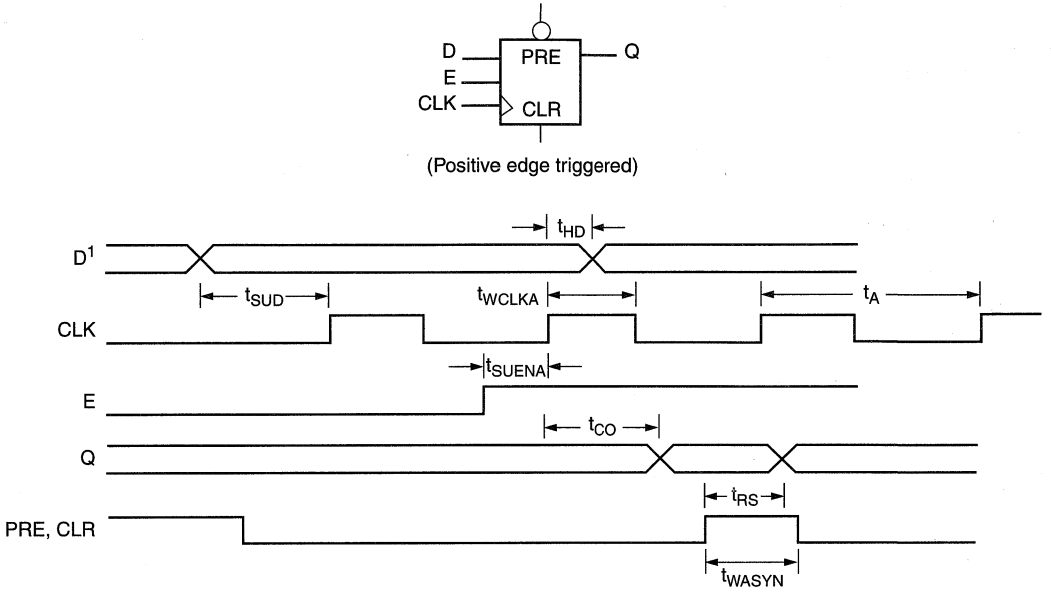


Module Delays



Sequential Timing Characteristics

Flip-Flops and Latches



Note: *D represents all data functions involving A, B, S for multiplexed flip-flops.*

ACT 1 Timing Characteristics

(Worst-Case Commercial Conditions, V_{CC} = 4.75 V, T_J = 70°C)

Logic Module Propagation Delays		'Std' Speed		'–1' Speed		'–2' Speed		'–3' Speed		
Parameter	**Description**	**Min.**	**Max.**	**Min.**	**Max.**	**Min.**	**Max.**	**Min.**	**Max.**	**Units**
t_{PD1}	Single Module		4.5		3.8		3.4		2.9	ns
t_{PD2}	Dual Module Macros		10.4		8.8		7.8		6.8	ns
t_{CO}	Sequential Clk to Q		4.5		3.8		3.4		2.9	ns
t_{GO}	Latch G to Q		4.5		3.8		3.4		2.9	ns
t_{RS}	Flip-Flop (Latch) Reset to Q		4.5		3.8		3.4		2.9	ns
Predicted Routing Delays[1]										
t_{RD1}	FO=1 Routing Delay		1.4		1.2		1.1		0.9	ns
t_{RD2}	FO=2 Routing Delay		2.2		1.9		1.7		1.4	ns
t_{RD3}	FO=3 Routing Delay		3.3		2.8		2.5		2.1	ns
t_{RD4}	FO=4 Routing Delay		4.8		4.1		3.6		3.1	ns
t_{RD8}	FO=8 Routing Delay		10.2		8.7		7.7		6.6	ns
Sequential Timing Characteristics[2]										
t_{SUD}	Flip-Flop (Latch) Data Input Setup	8.5		7.2		6.4		5.5		ns
t_{HD}[3]	Flip-Flop (Latch) Data Input Hold	0.0		0.0		0.0		0.0		ns
t_{SUENA}	Flip-Flop (Latch) Enable Setup	8.5		7.2		6.4		5.5		ns
t_{HENA}	Flip-Flop (Latch) Enable Hold	0.0		0.0		0.0		0.0		ns
t_{WCLKA}	Flip-Flop (Latch) Clock Active Pulse Width	10.5		9.0		8.0		6.8		ns
t_{WASYN}	Flip-Flop (Latch) Asynchronous Pulse Width	10.5		9.0		8.0		6.8		ns
t_A	Flip-Flop Clock Input Period	22.3		18.9		16.7		14.2		ns
f_{MAX}	Flip-Flop (Latch) Clock Frequency (FO = 128)		45		53		60		70	MHz

Notes:

1. *Routing delays are for typical designs across worst-case operating conditions. These parameters should be used for estimating device performance. Post-route timing analysis or simulation is required to determine actual worst-case performance. Post-route timing is based on actual routing delay measurements performed on the device prior to shipment.*
2. *Setup times assume fanout of 3. Further testing information can be obtained from the ALS Timer utility.*
3. *The Hold Time for the DFME1A macro may be greater than 0 ns. Use the Designer 3.0 or later Timer to check the Hold Time for this macro.*

ACT 1 Timing Characteristics (continued)

(Worst-Case Commercial Conditions

Input Module Propagation Delays			'Std' Speed		'–1' Speed		'–2' Speed		'–3' Speed		
Parameter	**Description**		**Min.**	**Max.**	**Min.**	**Max.**	**Min.**	**Max.**	**Min.**	**Max.**	**Units**
t_{INYH}	Pad to Y High			4.7		4.0		3.5		3.1	ns
t_{INYL}	Pad to Y Low			4.7		4.0		3.5		3.1	ns
Input Module Predicted Routing Delays[1]											
t_{IRD1}	FO=1 Routing Delay			1.4		1.2		1.1		0.9	ns
t_{IRD2}	FO=2 Routing Delay			2.2		1.9		1.7		1.4	ns
t_{IRD3}	FO=3 Routing Delay			3.3		2.8		2.5		2.1	ns
t_{IRD4}	FO=4 Routing Delay			4.8		4.1		3.6		3.1	ns
t_{IRD8}	FO=8 Routing Delay			10.2		8.7		7.7		6.6	ns
Global Clock Network											
t_{CKH}	Input Low to High	FO = 16		7.5		6.4		5.6		4.9	ns
		FO = 128		8.6		7.3		6.4		5.6	
t_{CKL}	Input High to Low	FO = 16		9.9		8.4		7.4		6.4	ns
		FO = 128		10.8		9.2		8.1		7.0	
t_{PWH}	Minimum Pulse Width High	FO = 16	10.0		8.5		7.5		6.5		ns
		FO = 128	10.5		9.0		8.0		6.8		
t_{PWL}	Minimum Pulse Width Low	FO = 16	10.0		8.5		7.5		6.5		ns
		FO = 128	10.5		9.0		8.0		6.8		
t_{CKSW}	Maximum Skew	FO = 16		1.8		1.5		1.3		1.2	ns
		FO = 128		2.8		2.4		2.1		1.8	
t_P	Minimum Period	FO = 16	20.9		17.6		15.4		13.2		ns
		FO = 128	22.3		18.9		16.7		14.2		
f_{MAX}	Maximum Frequency	FO = 16		48		57		65		75	MHz
		FO = 128		45		53		60		70	

Note:

1. These parameters should be used for estimating device performance. Optimization techniques may further reduce delays by 0 to 4 ns. Routing delays are for typical designs across worst-case operating conditions. Post-route timing analysis or simulation is required to determine actual worst-case performance. Post-route timing is based on actual routing delay measurements performed on the device prior to shipment.

ACT 1 Timing Characteristics (continued)

(Worst-Case Commercial Conditions)

Output Module Timing		'Std' Speed		'–1' Speed		'–2' Speed		'–3' Speed		
Parameter	**Description**	**Min.**	**Max.**	**Min.**	**Max.**	**Min.**	**Max.**	**Min.**	**Max.**	**Units**
TTL Output Module Timing[1]										
t_{DLH}	Data to Pad High		10.3		8.7		7.6		6.7	ns
t_{DHL}	Data to Pad Low		11.5		9.8		8.6		7.5	ns
t_{ENZH}	Enable Pad Z to High		10.2		8.6		7.5		6.6	ns
t_{ENZL}	Enable Pad Z to Low		12.2		10.4		9.1		7.9	ns
t_{ENHZ}	Enable Pad High to Z		15.4		13.1		11.6		10.0	ns
t_{ENLZ}	Enable Pad Low to Z		13.9		11.8		10.4		9.0	ns
d_{TLH}	Delta Low to High		0.09		0.08		0.07		0.06	ns/pF
d_{THL}	Delta High to Low		0.12		0.10		0.09		0.08	ns/pF
CMOS Output Module Timing[1]										
t_{DLH}	Data to Pad High		12.2		10.4		9.2		7.9	ns
t_{DHL}	Data to Pad Low		9.8		8.2		7.2		6.4	ns
t_{ENZH}	Enable Pad Z to High		9.2		7.9		6.9		6.0	ns
t_{ENZL}	Enable Pad Z to Low		12.7		10.7		9.4		8.3	ns
t_{ENHZ}	Enable Pad High to Z		15.4		13.1		11.6		10.0	ns
t_{ENLZ}	Enable Pad Low to Z		13.9		11.8		10.4		9.0	ns
d_{TLH}	Delta Low to High		0.15		0.13		0.11		0.10	ns/pF
d_{THL}	Delta High to Low		0.09		0.08		0.07		0.06	ns/pF

Note:

1. Delays based on 35 pF loading.

Macro Library

Hard Macros—Combinatorial

Function	Macro	Description	Modules C
Adder	FA1A	1-bit adder, carry in and carry out active low, A-input active low	2
	FA1B	1-bit adder, carry in and carry out active low	2
	FA2A	2-bit adder, carry in and carry out active low, A0 and A1 inputs active low	2
	HA1	Half-Adder	2
	HA1A	Half-Adder with active low A-input	2
	HA1B	Half-Adder with active low carry out and sum	2
	HA1C	Half-Adder with active low carry out	2
AND	AND2	2-input AND	1
	AND2A	2-input AND with active low A-input	1
	AND2B	2-input AND with active low inputs	1
	AND3	3-input AND	1
	AND3A	3-input AND with active low A-input	1
	AND3B	3-input AND with active low A- and B-inputs	1
	AND3C	3-input AND with active low inputs	1
	AND4	4-input AND	2
	AND4A	4-input AND with active low A-input	2
	AND4B	4-input AND with active low A- and B-inputs	1
	AND4C	4-input AND with active low A-, B-, and C-inputs	1
	AND4D	4-input AND with active low inputs	2
AND-OR	AO1	3-input AND-OR	1
	AO1A	3-input AND-OR with active low A-input	1
	AO1B	3-input AND-OR with active low C-input	1
	AO1C	3-input AND-OR with active low A- and C-inputs	1
	AO2	4-input AND-OR	1
	AO2A	4-input AND-OR with active low A-input	1
	AO3	4-input AND-OR	1
	AO4A	4-input AND-OR	1
	AO5A	4-input AND-OR	1
	AOI1	3-input AND-OR-INVERT	2
	AOI1A	3-input AND-OR-INVERT with active low A-input	1
	AOI1B	3-input AND-OR-INVERT with active low C-input	1
	AOI2A	4-input AND-OR-INVERT with active low A-input	1
	AOI2B	4-input AND-OR-INVERT with active low A- and C-inputs	1
	AOI3A	4-input AND-OR-INVERT with active low inputs	1
	AOI4	2-wide 4-input AND-OR-INVERT	2
AND-XOR	AX1	3-input AND-XOR with active low A-input	1
	AX1A	3-input AND-XOR-INVERT with active low A-input	1
	AX1B	3-input AND-XOR with active low A- and B-inputs	1
Buffer	BUF	Buffer with active high input and output	1
	BUFA	Buffer with active low input and output	1
Clock Net	GAND2	2-input AND Clock Net	1
	GMX4	4-to-1 Multiplexor Clock Net	1
	GNAND2	2-input NAND Clock Net	1
	GNOR2	2-input NOR Clock Net	1
	GOR2	2-input OR Clock Net	1
	GXOR2	2-input Exclusive OR Clock Net	1

Hard Macros—Combinatorial (Continued)

Function	Macro	Description	Modules C
Combinatorial	CM8A	Combinational Module	1
Inverter	INV	Inverter with active low output	1
	INVA	Inverter with active low input	1
Majority	MAJ3	3-input complex AND-OR	1
MUX	MX2	2-to-1 Multiplexor	1
	MX2A	2-to-1 Multiplexor with active low A-input	1
	MX2B	2-to-1 Multiplexor with active low B-input	1
MUX	MX2C	2-to-1 Multiplexor with active low output	1
	MX4	4-to-1 Multiplexor	1
	MXC1	Boolean	
	MXT	Boolean	
NAND	NAND2	2-input NAND	1
	NAND2A	2-input NAND with active low A-input	1
	NAND2B	2-input NAND with active low inputs	1
	NAND3	3-input NAND	
	NAND3A	3-input NAND with active low A-input	1
	NAND3B	3-input NAND with active low A- and B-inputs	1
	NAND3C	3-input NAND with active low inputs	1
	NAND4	4-input NAND	2
	NAND4A	4-input NAND with active low A-input	
	NAND4B	4-input NAND with active low A- and B-inputs	
	NAND4C	4-input NAND with active low A-, B-, and C-inputs	1
	NAND4D	4-input NAND with active low inputs	1
NOR	NOR2	2-input NOR	1
	NOR2A	2-input NOR with active low A-input	1
	NOR2B	2-input NOR with active low inputs	1
	NOR3	3-input NOR	1
	NOR3A	3-input NOR with active low A-input	1
	NOR3B	3-input NOR with active low A- and B-inputs	1
	NOR3C	3-input NOR with active low inputs	1
	NOR4	4-input NOR	2
	NOR4A	4-input NOR with active low A-input	1
	NOR4B	4-input NOR with active low A- and B-inputs	1
	NOR4C	4-input NOR with active low A-, B-, and C-inputs	2
	NOR4D	4-input NOR with active low inputs	2
OR	OR2	2-input OR	1
	OR2A	2-input OR with active low A-input	1
	OR2B	2-input OR with active low inputs	1
	OR3	3-input OR	1
	OR3A	3-input OR with active low A-input	1
	OR3B	3-input OR with active low A- and B-inputs	1
	OR3C	3-input OR with active low inputs	2
	OR4	4-input OR	1
	OR4A	4-input OR with active low A-input	1
	OR4B	4-input OR with active low A- and B-input	2
	OR4C	4-input OR with active low A-, B-, and C-inputs	2
	OR4D	4-input OR with active low inputs	2

Hard Macros—Combinatorial (Continued)

Function	Macro	Description	Modules C
OR-AND	OA1	3-input OR-AND	1
	OA1A	3-input OR-AND with active low A-input	1
	OA1B	3-input OR-AND with active low C-input	1
	OA1C	3-input OR-AND with active low A- and C-inputs	1
	OA2	2-wide 4-input OR-AND	1
	OA2A	2 wide 4-input OR-AND with active low A-input	1
	OA3	4-input OR-AND	1
	OA3A	4-input OR-AND with active low C-input	1
	OA3B	4-input OR-AND with active low A- and C-inputs	1
	OA4A	4-input OR-AND with active low C-input	1
	OA5	4-input complex OR-AND	1
	OAI1	3-input OR-AND-INVERT	1
	OAI2A	4-input OR-AND-INVERT with active low D-input	1
	OAI3	4-input OR-AND-INVERT	2
	OAI3A	4-input OR-AND-INVERT with active low C- and D-inputs	1
XNOR	XNOR	2-input XNOR	1
XNOR-AND	XA1A	3-input XNOR-AND	1
XNOR-OR	XO1A	3-input XNOR-OR	1
XOR	XOR	2-input XOR	1
XOR-AND	XA1	3-input XOR-AND	1
XOR-OR	XO1	3-input XOR-OR	1

Hard Macros—Sequential

Function	Macro	Description	Modules C
D-Type	DF1	D-Type Flip-Flop	2
	DF1A	D-Type Flip-Flop with active low output	2
	DF1B	D-Type Flip-Flop with active low clock	2
	DF1C	D-Type Flip-Flop with active low clock and output	2
	DFC1	D-Type Flip-Flop with active high Clear	2
	DFC1A	D-Type Flip-Flop with active high Clear and active low clock	2
	DFC1B	D-Type Flip-Flop with active low Clear	2
	DFC1C	D-Type Flip-Flop with Clear, Sequential	2
	DFC1D	D-Type Flip-Flop with active low Clear and clock	2
	DFC1E	D-Type Flip-Flop with Clear, Sequential	2
	DFC1F	D-Type Flip-Flop with Clear, Sequential	2
	DCF1G	D-Type Flip-Flop with Clear, Sequential	2
	DFE	D-Type Flip-Flop with active high Enable	2
	DFE1B	D-Type Flip-Flop with active low Enable	2
	DFE1C	D-Type Flip-Flop with active low Enable and clock	2
	DFE2D	D-Type Flip-Flop with Enable, Sequential	2
	DFE3A	D-Type Flip-Flop with Enable and active low Clear	2
	DFE3B	D-Type Flip-Flop with Enable and active low Clear and clock	2
	DFE3C	D-Type Flip-Flop with active low Enable and Clear	2
	DFE3D	D-Type Flip-Flop with active low Enable, Clear, and clock	2
	DFE4	D-Type Flip-Flop with Enable, Sequential	2
	DFE4A	D-Type Flip-Flop with Enable, Sequential	2
	DFE4B	D-Type Flip-Flop with Enable, Sequential	2
	DFE4C	D-Type Flip-Flop with Enable, Sequential	2
	DFEA	D-Type Flip-Flop with Enable and active low clock	2
	DFEB	D-Type Flip-Flop with Enable, Sequential	2
	DFEC	D-Type Flip-Flop with Enable, Sequential	2
	DFED	D-Type Flip-Flop with Enable, Sequential	2
	DFM	2-input D-Type Flip-Flop with Multiplexed Data	2
	DFM1B	2-input D-Type Flip-Flop with Multiplexed Data and active low output	2
	DFM1C	2-input D-Type Flip-Flop with Multiplexed Data and active low clock and output	2
	DFM3	2-input D-Type Flip-Flop with Multiplexed Data and Clear	2
	DFM3B	2-input D-Type Flip-Flop with Multiplexed Data and active low Clear and clock	2
	DFM3E	2-input D-Type Flip-Flop with Multiplexed Data, Clear, and active low clock	2
	DFM3F	D-Type Flip-Flop with Multiplexed Data, Enable and Sequential	2
	DFM3G	D-Type Flip-Flop with Multiplexed Data, Enable and Sequential	2
	DFM4	D-Type Flip-Flop with Multiplexed Data, Enable and Sequential	2
	DFM4A	D-Type Flip-Flop with Multiplexed Data, Enable and Sequential	2
	DFM4B	D-Type Flip-Flop with Multiplexed Data, Enable and Sequential	2
	DFM4C	2-input D-Type Flip-Flop with Multiplexed Data and active low Preset and output	2
	DFM4D	2-input D-Type Flip-Flop with Multiplexed Data and active low Preset, clock, and output	2
	DFM4E	D-Type Flip-Flop with Multiplexed Data, Enable and Sequential	2
	DFM5A	D-Type Flip-Flop with Multiplexed Data, Enable and Sequential	2

Hard Macros—Sequential (Continued)

Function	Macro	Description	Modules C
D-Type	DFM5B	D-Type Flip-Flop with Multiplexed Data, Enable and Sequential	2
	DFMA	2-input D-Type Flip-Flop with Multiplexed Data and active low Clock	2
	DFMB	2-input D-Type Flip-Flop with Multiplexed Data and active low Clear	2
	DFME1A	2-input D-Type Flip-Flop with Multiplexed Data and active low Enable	2
	DFP1	D-Type Flip-Flop with active high Preset	2
	DFP1A	D-Type Flip-Flop with active high Preset and active low clock	2
	DFP1B	D-Type Flip-Flop with active low Preset	2
	DFP1C	D-Type Flip-Flop with active high Preset and active low output	2
	DFP1D	D-Type Flip-Flop with active low Preset and clock	2
	DFP1E	D-Type Flip-Flop with active low Preset and output	2
	DFP1F	D-Type Flip-Flop with active high Preset and active low clock and output	2
	DFP1G	D-Type Flip-Flop with active low Preset, clock, and output	2
	DFPC	D-Type Flip-Flop with active high Preset, active low Clear, and active high clock	2
	DFPCA	D-Type Flip-Flop with active high Preset and active low Clear and clock	2
J-K Type	JKF	JK Flip-Flop with active low K-input	2
	JKF1B	JK Flip-Flop with active low clock and K-input	2
	JKFPC	JK Flip-Flop, Sequential	2
	JKF2A	JK Flip-Flop with active low Clear and K-input	2
	JKF2B	JK Flip-Flop with active low Clear, clock, and K-input	2
	JKF2C	JK Flip-Flop with active high Clear and active low K-input	2
	JKF2D	JK Flip-Flop with active high Clear and active low clock and K-input	2
	JKF3A	JK Flip-Flop, Sequential	2
	JKF3B	JK Flip-Flop, Sequential	2
	JKF3C	JK Flip-Flop, Sequential	2
	JKF3D	JK Flip-Flop, Sequential	2
	JKF4B	JK Flip-Flop, Sequential	2
Latch	DL1	Data Latch	1
	DL1A	Data Latch with active low output	1
	DL1B	Data Latch with active low clock	1
	DL1C	Data Latch with active low clock and output	1
	DL2A	Sequential, Data Latch	1
	DL2B	Sequential, Data Latch	1
	DL2C	Sequential, Data Latch	1
	DL2D	Sequential, Data Latch	1
	DLC	Data Latch with active low Clear	1
	DLC1	Data Latch with active high Clear	1
	DLC1A	Data Latch with active high Clear and active low clock	1
	DLC1F	Data Latch with active high Clear and active low output	1
	DLC1G	Data Latch with active high Clear and active low clock and output	1
	DLCA	Data Latch with active low Clock and Clear	1
	DLE	Data Latch with active high Enable	1
	DLE1D	Data Latch with active high Enable and clock and active low input and output	1
	DLE2A	Sequential, Data Latch with Enable	1
	DLE2B	Data Latch with active low Enable, Clear, and clock	1
	DLE2C	Data Latch with active low Enable and clock and active high clear	1

Hard Macros—Sequential (Continued)

Function	Macro	Description	Modules C
Latch	DLE3A	Sequential, Data Latch with Enable	1
	DLE3B	Data Latch with active low Enable and clock and active low Preset	1
	DLE3C	Data Latch with active low Enable Preset and clock	1
	DLEA	Data Latch with active low Enable and active high clock	1
	DLEB	Data Latch with active high Enable and active high clock	1
	DLEC	Data Latch with active low Enable and clock	1
	DLM	2-input Data Latch with Multiplexed Data	1
	DLM2A	Sequential, Data Latch with Multiplexed Data	1
	DLMA	2-input Data Latch with Multiplexed Data and active low clock	1
	DLME1A	2-input Data Latch with Multiplexed Data and Enable and active low clock	1
	DLP1	Data Latch with active high Preset and clock	1
	DLP1A	Data Latch with active high Preset and active low clock	1
	DLP1B	Data Latch with active low Preset and active high clock	1
	DLP1C	Data Latch with active low Preset and clock	1
	DLP1D	Data Latch with active low Preset and output and active high clock	1
	DLP1E	Data Latch with active low Preset, clock, and output	1

Input/Output Macros

Function	Macro	Description	I/O Modules
Buffer	INBUF	Input Buffer	1
	OUTBUF	Output buffer, High Slew	1
Bidirectional	BIBUF	Bidirectional Buffer, High Slew (with hidden buffer at Y pin)	1
Input	CLKBUF	Input for Dedicated Routed Clock Network	1
Output	TRIBUFF	Tristate output, High Slew	1

Soft Macros

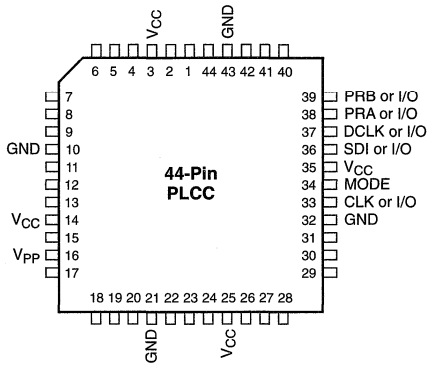
Function	Macro	Description	Maximum Logic Levels	Modules S	Modules C
Adder	FADD16	16-bit adder	5		79
	FADD8	8-bit adder	4		37
	FADD24	24-bit adder	6		120
	FADD32	32-bit adder	7		160
Comparator	ICMP4	4-bit Identity Comparator	2		5
	ICMP8	8-bit Identity Comparator	3		9
	MCMP16	16-bit Magnitude Comparator	5		93
	MCMPC2	2-bit Magnitude Comparator with Enable	3		9
	MCMPC4	4-bit Magnitude Comparator with Enable	4		18
	MCMPC8	8-bit Magnitude Comparator with Enable	6		36
Counter	CNT4A	4-bit binary counter with load and clear	4		18
	CNT4B	4-bit binary counter with load, clear, carry-in, carry-out	4		15
	UDCNT4A	4-bit up/down counter with load, carry-in, and carry-out	6		24
Decoder	DEC2X4	2-to-4 decoder	1		4
	DEC2X4A	2-to-4 decoder with active low outputs	1		4
	DEC3X8	3-to-8 decoder	1		8
	DEC3X8A	3-to-8 decoder with active low outputs	1		9
	DEC4X16A	4-to-16 decoder with active low outputs	2		20
	DECE2X4	2-to-4 decoder with enable	1		4
	DECE2X4A	2-to-4 decoder with enable and active low outputs	1		5
	DECE3X8	3-to-8 decoder with enable	2		11
	DECE3X8A	3-to-8 decoder with enable and active low outputs	2		11
Latch	DLC8A	octal latch with clear active low 8-bit Data Latch with active low Clear	1		8
	DLE8	octal latch with enable 8-bit Data Latch with active high Enable	1		8
	DLM8	octal latch with multiplexed data 8-bit Data Latch with Multiplexed Data	1		8
MUX	MX16	16-to-1 Multiplexor	2		5
	MX8	8-to-1 Multiplexor with active high output	2		3
	MX8A	8-to-1 Multiplexor with active low output	2		3
Multiplier	SMULT8	8-bit by 8-bit Multiplier			241
Registers	REGE8A	8-bit register, with enable, preset, and active low clear	1		20
	REGE8B	8-bit register, with enable, preset, and active low clear and negative edge clock	1		20
Shift Register	SREG4A	4-bit shift register with clear active low	1		8
	SREG8A	8-bit shift register with clear active low	1		18

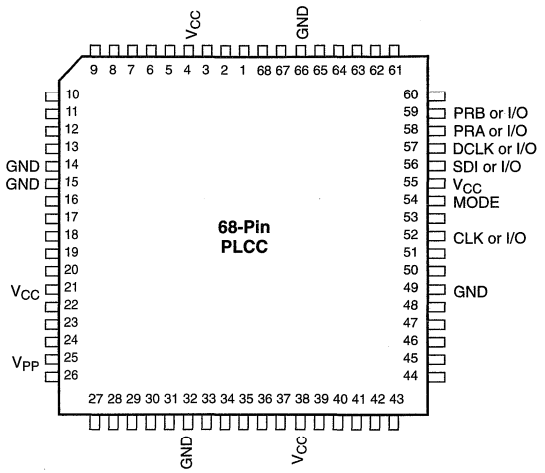
Soft Macros—TTL Equivalent

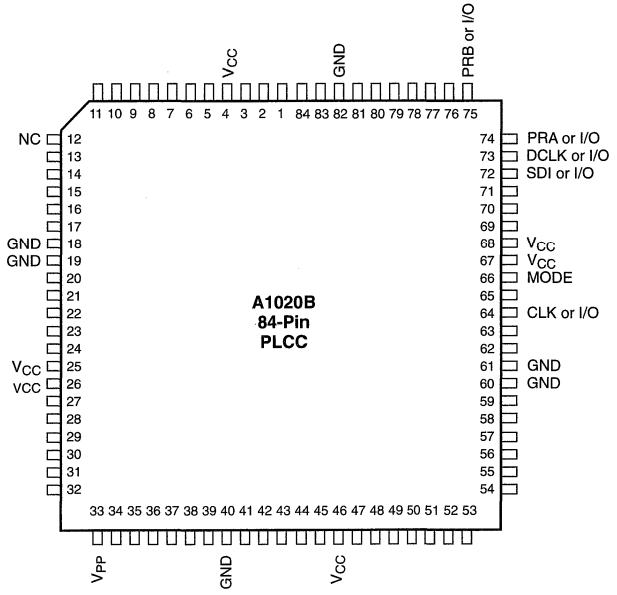
Function	Macro	Description	Maximum Logic Levels	Modules S	Modules C
	TA138	3-to-8 decoder with enable and active low outputs	2		12
	TA139	2-to-4 decoder with active low enable and outputs	1		4
	TA151	8-to-1 multiplexor with enable and both active low and active high output	3		5
	TA153	4-to-1 multiplexor with active low enable	2		2
	TA157	2-to-1 multiplexor with active low enable	1		1
	TA161	4-bit binary counter with active low clear and load	3		22
	TA164	8-bit serial in, parallel out shift register, active low clear	1		18
	TA169	4-bit Up/Down Counter	6		25
	TA181	ALU	4		37
	TA194	4-bit bidirectional universal shift register	1		14
	TA195	4-bit parallel-access shift register	1		11
	TA269	8-bit up/down binary counter	8		50
	TA273	octal register with clear	1		18
	TA280	9-bit odd/even parity generator and checker	4		9
	TA377	octal register with active low enable	1		16

Package Pin Assignments

(Top View)





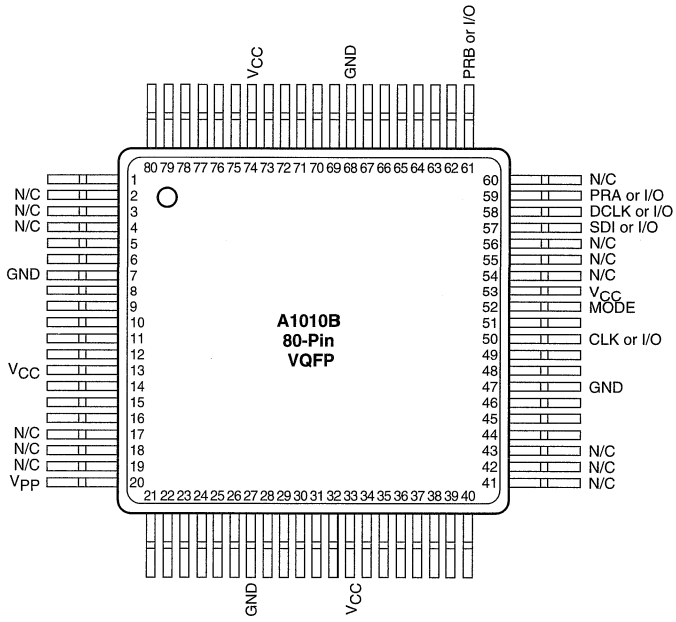


Notes:

1. *V_{PP} must be terminated to V_{CC}, except during device programming.*
2. *MODE must be terminated to circuit ground, except during device programming or debugging.*
3. *Unused I/O pins are designated as outputs by ALS and are driven low.*
4. *All unassigned pins are available for use as I/Os.*

Package Pin Assignments (continued)

(Top View)

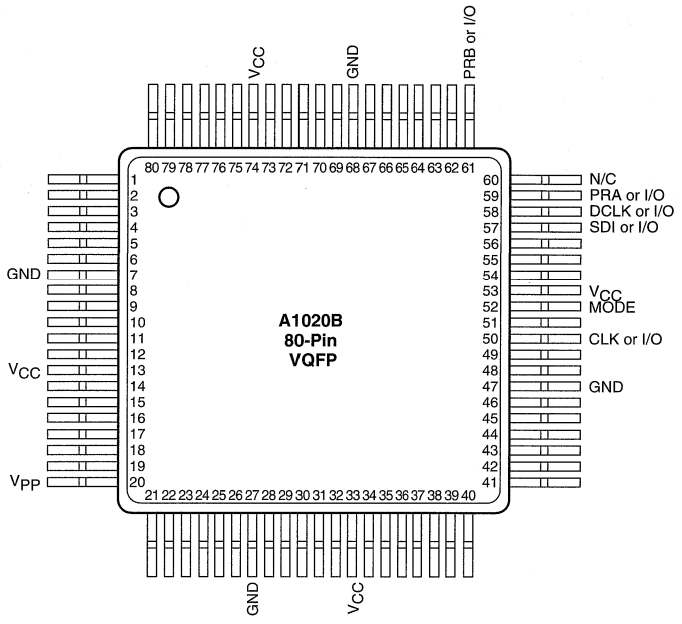


Notes:

1. *V_{PP} must be terminated to V_{CC}, except during device programming.*
2. *MODE must be terminated to circuit ground, except during device programming or debugging.*
3. *Unused I/O pins are designated as outputs by ALS and are driven low.*
4. *All unassigned pins are available for use as I/Os.*

Package Pin Assignments (continued)

(Top View)

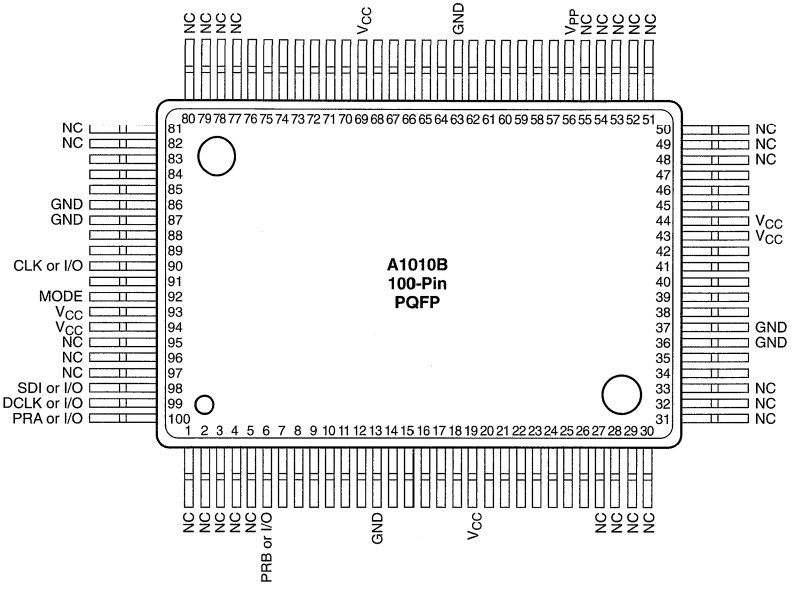


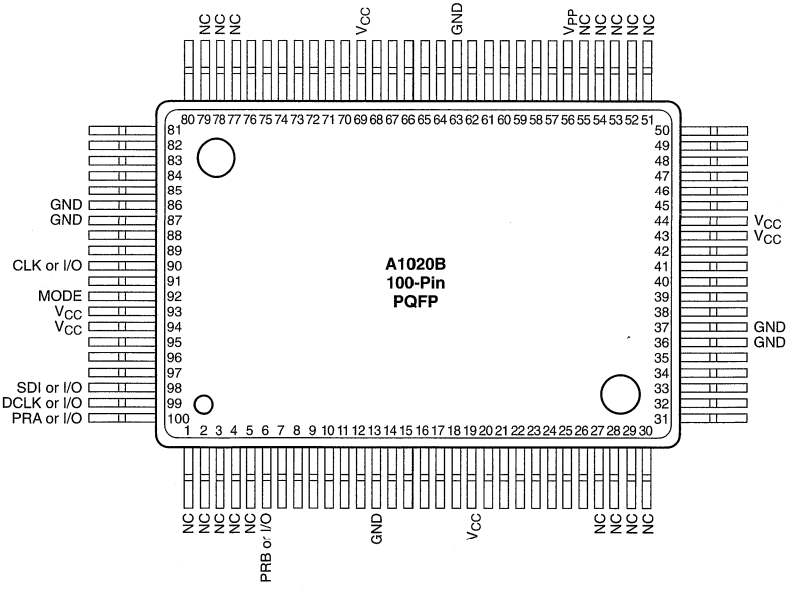
Notes:

1. *V_{PP} must be terminated to V_{CC}, except during device programming.*
2. *MODE must be terminated to circuit ground, except during device programming or debugging.*
3. *Unused I/O pins are designated as outputs by ALS and are driven low.*
4. *All unassigned pins are available for use as I/Os.*

Package Pin Assignments (continued)

(Top View)



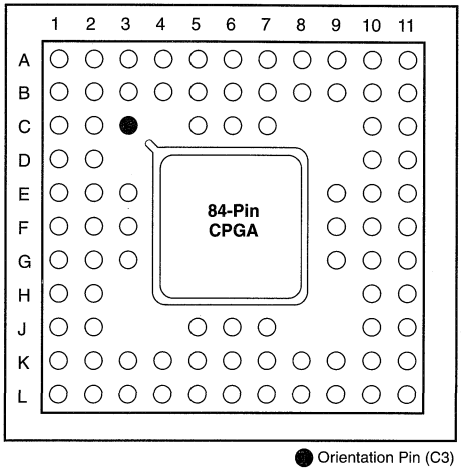


Notes:

1. *V_{PP} must be terminated to V_{CC}, except during device programming.*
2. *MODE must be terminated to circuit ground, except during device programming or debugging.*
3. *Unused I/O pins are designated as outputs by ALS and are driven low.*
4. *All unassigned pins are available for use as I/Os.*

Package Pin Assignments (continued)

(Top View)



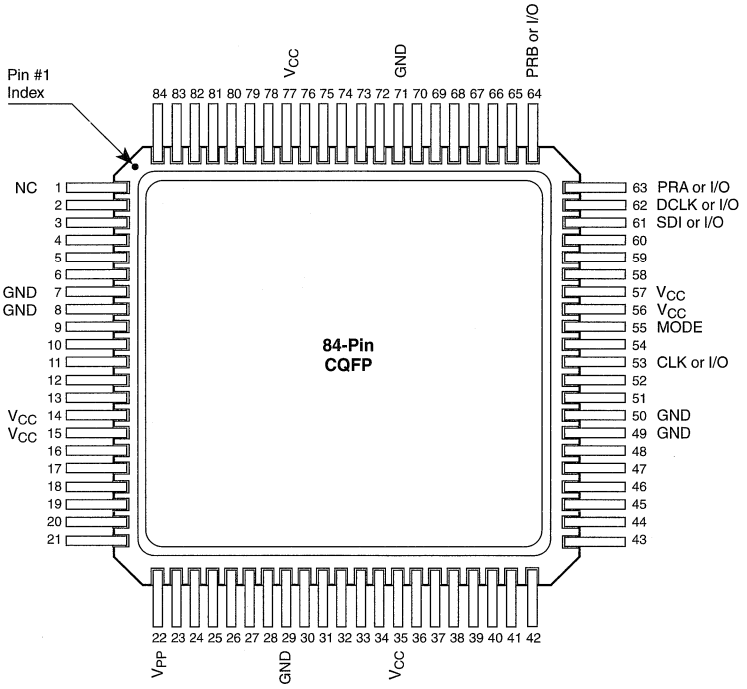
Signal	A1010B Devices	A1020B Devices
PRA	A11	A11
PRB	B10	B10
MODE	E11	E11
SDI	B11	B11
DCLK	C10	C10
V_{PP}	K2	K2
CLK or I/O	F9	F9
GND	B7, E2, E3, K5, F10, G10	B7, E2, E3, K5, F10,G10
V_{CC}	B5, F1, G2, K7, E9, E10	B5, F1, G2, K7, E9, E10
N/C (No Connection)	B1, B2, C1, C2, K1, J2, J10, K10, K11, C11, D10, D11, L1	B2

Notes:

1. *V_{PP} must be terminated to V_{CC}, except during device programming.*
2. *MODE must be terminated to circuit ground, except during device programming or debugging.*
3. *Unused I/O pins are designated as outputs by ALS and are driven low.*
4. *All unassigned pins are available for use as I/Os.*

Package Pin Assignments (continued)

(Top View)



Notes:

1. *V_{PP} must be terminated to V_{CC}, except during device programming.*
2. *MODE must be terminated to circuit ground, except during device programming or debugging.*
3. *Unused I/O pins are designated as outputs by ALS and are driven low.*
4. *All unassigned pins are available for use as I/Os.*

ACT™ 1 3.3 Volt Field Programmable Gate Arrays

Features

- 3.3 V functionality fully compliant with JEDEC specifications
- Up to 2000 Gate Array Gates (6000 PLD equivalent gates)
- Replaces up to 50 TTL Packages
- Replaces up to twenty 20-Pin PAL™ Packages
- Design Library with over 250 Functions
- Gate Array Architecture Allows Completely Automatic Place and Route
- Up to 547 Programmable Logic Modules
- Up to 273 Flip-Flops
- Flip-Flop Toggle Rates to 100 MHz
- Two In-Circuit Diagnostic Probe Pins Support Speed Analysis to 15 MHz
- Built-In High Speed Clock Distribution Network
- I/O Drive to 6 mA
- Nonvolatile, User Programmable
- Logic Fully Tested Prior to Shipment
- Fabricated in 1.0 micron CMOS technology

Description

The ACT™ 1 3.3 Volt family of field programmable gate arrays (FPGAs) offers a variety of package, speed, and application combinations. Devices are implemented in silicon gate, 1-micron two-level metal CMOS, and they employ Actel's PLICE™ antifuse technology. The unique architecture offers gate array flexibility, high performance, and instant turnaround through user programming. Device utilization is typically 95 percent of available logic modules.

ACT 1 3.3 Volt devices also provide system designers with unique on-chip diagnostic probe capabilities, allowing convenient testing and debugging. Additional features include an on-chip clock driver with a hardwired distribution network. The network provides efficient clock distribution with minimum skew.

The user-definable I/Os are capable of driving at both TTL and CMOS drive levels. Available packages include plastic and ceramic J-leaded chip carriers, ceramic and plastic quad flatpacks, and ceramic pin grid array.

A security fuse may be programmed to disable all further programming and to protect the design from being copied or reverse engineered.

Product Family Profile

Device	A10V10B	A10V20B
Capacity		
Gate Array Equivalent Gates	1200	2000
PLD Equivalent Gates	3000	6000
TTL Equivalent Packages	30	50
20-Pin PAL Equivalent Packages	12	20
Logic Modules	295	547
Flip-Flops (maximum)	147	273
Routing Resources		
Horizontal Tracks/Channel	22	22
Vertical Tracks/Column	13	13
PLICE Antifuse Elements	112,000	186,000
User I/Os (maximum)	57	69
Packages:	68 PLCC	68 PLCC
		84 PLCC
	80 VQFP	80 VQFP
Performance		
Flip-Flop Toggle Rate (maximum)	100 MHz	100 MHz
System Speed (maximum)	33MHz	33MHz

The Designer and Designer Advantage™ Systems

The ACT 1 3.3 Volt device family is supported by Actel's Designer and Designer Advantage Systems, allowing logic design implementation with minimum effort. The systems interface with the resident CAE system to provide a complete gate array design environment: schematic capture, simulation, fully automatic place and route, timing verification, and device programming. The systems also include the ACTmap™ optimization and sysnthesis tool, and the ACTgen™ Macro Builder, a powerful macro function generator for counters, adders, and other structural blocks. The systems are available for 386/486/Pentium™ PC and for HP™ and Sun™ workstations and for running Viewlogic®, Mentor Graphics®, Cadence™, and OrCAD™.

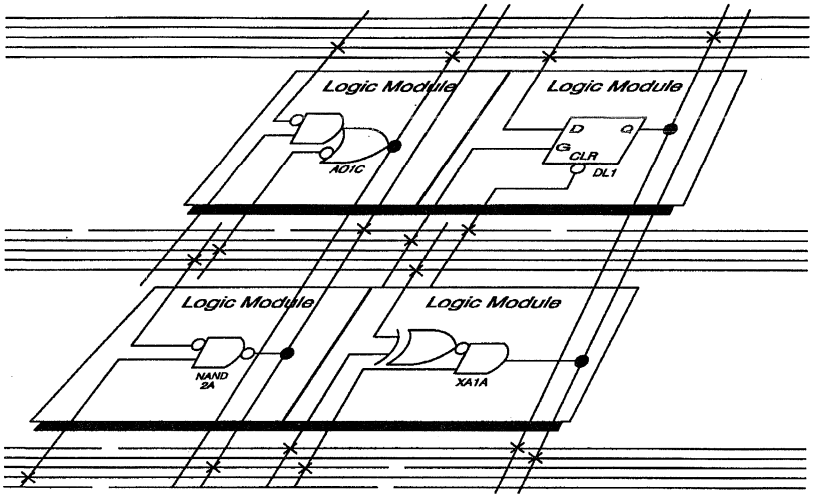


Figure 1 • Partial View of an ACT 1 Device

ACT 1 Device Structure

A partial view of an ACT 1 device (Figure 1) depicts four logic modules and distributed horizontal and vertical interconnect tracks. PLICE antifuses, located at intersections of the horizontal and vertical tracks, connect logic module inputs and outputs. During programming, these antifuses are addressed and programmed to make the connections required by the circuit application.

The ACT 1 Logic Module

The ACT 1 logic module is an 8-input, one-output logic circuit chosen for the wide range of functions it implements and for its efficient use of interconnect routing resources (Figure 2).

The logic module can implement the four basic logic functions (NAND, AND, OR, and NOR) in gates of two, three, or four inputs. Each function may have many versions, with different combinations of active-low inputs. The logic module can also implement a variety of D-latches, exclusivity functions, AND-ORs, and OR-ANDs. No dedicated hardwired latches or flip-flops are required in the array, since latches and flip-flops may be constructed from logic modules wherever needed in the application.

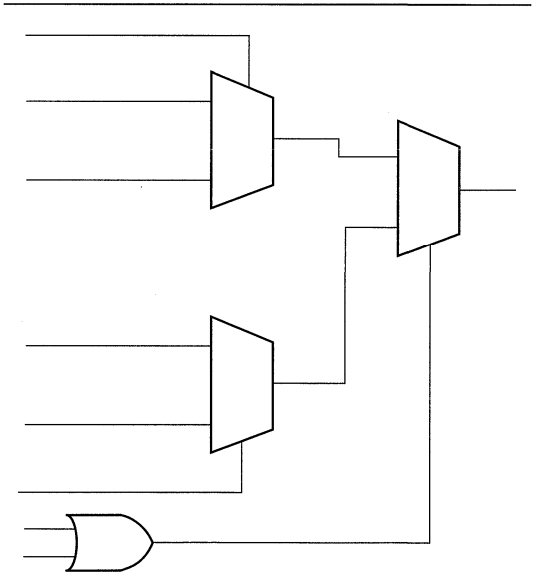


Figure 2 • ACT 1 Logic Module

I/O Buffers

Each I/O pin is available as an input, output, three-state, or bidirectional buffer. Input and output levels are compatible with standard TTL and CMOS specifications. Outputs sink or source 10 mA at TTL levels. See Electrical Specifications for additional I/O buffer specifications.

Device Organization

ACT 1 devices consist of a matrix of logic modules arranged in rows separated by wiring channels. This array is surrounded by a ring of peripheral circuits including I/O buffers, testability circuits, and diagnostic probe circuits providing real-time diagnostic capability. Between rows of logic modules are routing channels containing sets of segmented metal tracks with PLICE antifuses. Each channel has 22 signal tracks. Vertical routing is permitted via 13 vertical tracks per logic module column. The resulting network allows arbitrary and flexible interconnections between logic modules and I/O modules.

Probe Pin

ACT 1 devices have two independent diagnostic probe pins. These pins allow the user to observe any two internal signals by entering the appropriate net name in the diagnostic software. Signals may be viewed on a logic analyzer using Actel's Actionprobe® diagnostic tools. The probe pins can also be used as user-defined I/Os when debugging is finished.

ACT 1 Array Performance

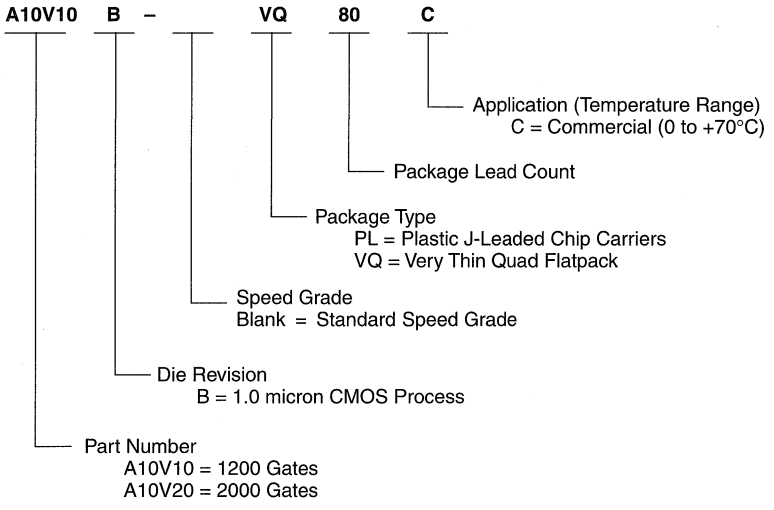
Temperature and Voltage Effects

Worst-case delays for ACT 1 arrays are calculated in the same manner as for masked array products. A typical delay parameter is multiplied by a derating factor to account for temperature, voltage, and processing effects. However, in an ACT 1 array, temperature and voltage effects are less dramatic than with masked devices. The electrical characteristics of module interconnections on ACT 1 devices remain constant over voltage and temperature fluctuations.

Logic Module Size

Logic module size also affects performance. A mask programmed gate array cell with four transistors usually implements only one logic level. In the more complex logic module (similar to the complexity of a gate array macro) of an ACT 1 array, implementation of multiple logic levels within a single module is possible. This eliminates interlevel wiring and associated RC delays. The effect is termed "net compression."

Ordering Information



Product Plan

	Speed Grade	Application
	Std	C
A10V10B Device		
68-pin Plastic Leaded Chip Carrier (PL)	✔	✔
80-pin Very Thin (1.0 mm) Quad Flatpack (VQ)	✔	✔
A10V20B Device		
68-pin Plastic Leaded Chip Carrier (PL)	✔	✔
84-pin Plastic Leaded Chip Carrier (PL)	✔	✔
80-pin Very Thin (1.0 mm) Quad Flatpack (VQ)	✔	✔

Applications: C = Commercial Availability: ✔ = Available, P = Planned, — = Not Planned

Device Resources

Device	Logic Modules	Gates	User I/Os		
			68-pin	80-pin	84-pin
A10V10B	295	1200	57	57	
A10V20B	547	2000	57	69	69

Pin Description

CLK Clock (Input)

TTL Clock input for global clock distribution network. The Clock input is buffered prior to clocking the logic modules. This pin can also be used as an I/O.

DCLK Diagnostic Clock (Input)

TTL Clock input for diagnostic probe and device programming. DCLK is active when the MODE pin is HIGH. This pin functions as an I/O when the MODE pin is LOW.

GND Ground

Input LOW supply voltage.

I/O Input/Output (Input, Output)

I/O pin functions as an input, output, three-state, or bidirectional buffer. Input and output levels are compatible with standard TTL and CMOS specifications. Unused I/O pins are automatically driven LOW by the ALS software.

MODE Mode (Input)

The MODE pin controls the use of multifunction pins (DCLK, PRA, PRB, SDI). When the MODE pin is HIGH, the special functions are active. When the MODE pin is LOW, the pins function as I/O.

NC No Connection

This pin is not connected to circuitry within the device.

PRA Probe A (Output)

The Probe A pin is used to output data from any user-defined design node within the device. This independent diagnostic pin is used in conjunction with the Probe B pin to allow real-time diagnostic output of any signal path within the device. The Probe A pin can be used as a user-defined I/O when debugging has been completed. The pin's probe capabilities can be permanently disabled to protect the programmed design's confidentiality. PRA is active when the MODE pin is HIGH. This pin functions as an I/O when the MODE pin is LOW.

PRB Probe B (Output)

The Probe B pin is used to output data from any user-defined design node within the device. This independent diagnostic pin is used in conjunction with the Probe A pin to allow real-time diagnostic output of any signal path within the device. The Probe B pin can be used as a user-defined I/O when debugging has been completed. The pin's probe capabilities can be permanently disabled to protect the programmed design's confidentiality. PRB is active when the MODE pin is HIGH. This pin functions as an I/O when the MODE pin is LOW.

SDI Serial Data Input (Input)

Serial data input for diagnostic probe and device programming. SDI is active when the MODE pin is HIGH. This pin functions as an I/O when the MODE pin is LOW.

V_{CC} Supply Voltage

Input HIGH supply voltage.

V_{PP} Programming Voltage

Input supply voltage used for device programming. This pin must be connected to V_{CC} during normal operation.

Absolute Maximum Ratings[1]

Free air temperature range

Symbol	Parameter	Limits	Units
V_{CC}	DC Supply Voltage[2]	–0.5 to +7.0	Volts
V_I	Input Voltage	–0.5 to V_{CC} +0.5	Volts
V_O	Output Voltage	–0.5 to V_{CC} +0.5	Volts
I_{IO}	I/O Sink/Source Current[3]	±20	mA
T_{STG}	Storage Temperature	–65 to +150	°C

Notes:

1. *Stresses beyond those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. Exposure to absolute maximum rated conditions for extended periods may affect device reliability. Device should not be operated outside the Recommended Operating Conditions.*
2. *$V_{PP} = V_{CC}$ except during device programming.*
3. *Device inputs are normally high impedance and draw extremely low current. However, when input voltage is greater than V_{CC} + 0.5 V or less than GND – 0.5 V, the internal protection diode will be forward biased and can draw excessive current.*

Recommended Operating Conditions

Parameter	Commercial	Units
Temperature Range[1]	0 to +70	°C
Power Supply Tolerance	3.0 to 3.6	V

Note:

1. *Ambient temperature (T_A) used for commercial.*

Electrical Specifications

Parameter		Commercial		Units
		Min.	Max.	
V_{OH}[1]	(I_{OH} = –4 mA)	2.15		V
	(I_{OH} = –3.2 mA)	2.4		V
V_{OL}[1]	(I_{OL} = 6 mA)		0.4	V
V_{IL}		–0.3	0.8	V
V_{IH}		2.0	V_{CC} + 0.3	V
Input Transition Time t_R, t_F[2]			500	ns
C_{IO} I/O Capacitance[2, 3]			10	pF
Standby Current, I_{CC}[4] (typical = 0.3 mA)			0.75	mA
Leakage Current[5]		–10	10	µA

Notes:

1. *Only one output tested at a time. V_{CC} = min.*
2. *Not tested, for information only.*
3. *Includes worst-case 84-pin PLCC package capacitance. V_{OUT} = 0 V, f = 1 MHz.*
4. *Typical standby current = 0.3 mA. All outputs unloaded. All inputs = V_{CC} or GND.*
5. *V_O, V_{IN} = V_{CC} or GND.*

Package Thermal Characteristics

The device junction to case thermal characteristics is θjc, and the junction to ambient air characteristics is θja. The thermal characteristics for θja are shown with two different air flow rates. Maximum junction temperature is 150°C.

A sample calculation of the maximum power dissipation for an 84-pin plastic leaded chip carrier at commercial temperature is as follows:

$$\frac{\text{Max junction temp. (°C)} - \text{Max commercial temp. (°C)}}{\theta ja\ (°C/W)} = \frac{150°C - 70°C}{44°C/W} = 1.82\ W$$

Package Type	Pin Count	θja Still Air	θja 300 ft/min	Units
Plastic J-Leaded Chip Carrier	68	38	29	°C/W
	84	37	28	°C/W
Very Thin (1.0 mm) Quad Flatpack	80	43	35	°C/W

General Power Equation

$$P = [I_{CC}\ \text{standby} + I_{CC}\ \text{active}] * V_{CC} + I_{OL} * V_{OL} * N + I_{OH} * (V_{CC} - V_{OH}) * M$$

Where:

I_{CC} standby is the current flowing when no inputs or outputs are changing.

I_{CC} active is the current flowing due to CMOS switching.

I_{OL}, I_{OH} are TTL sink/source currents.

V_{OL}, V_{OH} are TTL level output voltages.

N equals the number of outputs driving TTL loads to V_{OL}.

M equals the number of outputs driving TTL loads to V_{OH}.

An accurate determination of N and M is problematical because their values depend on the family type, design details, and on the system I/O. The power can be divided into two components: static and active.

Static Power Component

Actel FPGAs have small static power components that result in lower power dissipation than PALs or PLDs. By integrating multiple PALs/PLDs into one FPGA, an even greater reduction in board-level power dissipation can be achieved.

The power due to standby current is typically a small component of the overall power. Standby power is calculated below for commercial, worst case conditions.

I_{CC}	V_{CC}	Power
0.75 mA	3.6 V	2.70 mW (max)
0.30 mA	3.3 V	0.99 mW (typ)

Active Power Component

Power dissipation in CMOS devices is usually dominated by the active (dynamic) power dissipation. This component is frequency dependent, a function of the logic and the external I/O. Active power dissipation results from charging internal chip capacitances of the interconnect, unprogrammed antifuses, module inputs, and module outputs, plus external capacitance due to PC board traces and load device inputs. An additional component of the active power dissipation is the totem-pole current in CMOS transistor pairs. The net effect can be associated with an equivalent capacitance that can be combined with frequency and voltage to represent active power dissipation.

Equivalent Capacitance

The power dissipated by a CMOS circuit can be expressed by the Equation 1:

$$\text{Power (uW)} = C_{EQ} * V_{CC}^2 * F \quad (1)$$

Where:

C_{EQ} is the equivalent capacitance expressed in pF.

V_{CC} is the power supply in volts.

F is the switching frequency in MHz.

Equivalent capacitance is calculated by measuring ICC active at a specified frequency and voltage for each circuit component of interest. Measurements have been made over a range of frequencies at a fixed value of VCC. Equivalent capacitance is frequency independent so that the results may be used over a wide range of operating conditions.

C_{EQ} Values for Actel FPGAs

Modules (C_{EQM})	3.2
Input Buffers (C_{EQI})	10.9
Output Buffers (C_{EQO})	11.6
Routed Array Clock Buffer Loads (C_{EQCR})	4.1

To calculate the active power dissipated from the complete design, the switching frequency of each part of the logic must be known. Equation 2 shows a piece-wise linear summation over all components.

$$\text{Power} = V_{CC}^2 * [(m * C_{EQM} * f_m)_{modules} + (n * C_{EQI} * f_n)_{inputs} + (p * (C_{EQO} + C_L) * f_p)_{outputs} + 0.5 * (q1 * C_{EQCR} * f_{q1})_{routed\_Clk1} + (r1 * fq1)_{routed\_Clk1}] \quad (2)$$

Where:

m	=	Number of logic modules switching at fm
n	=	Number of input buffers switching at fn
p	=	Number of output buffers switching at fp
q_1	=	Number of clock loads on the first routed array clock (All families)
r_1	=	Fixed capacitance due to first routed array clock (All families)
C_{EQM}	=	Equivalent capacitance of logic modules in pF
C_{EQI}	=	Equivalent capacitance of input buffers in pF
C_{EQO}	=	Equivalent capacitance of output buffers in pF
C_{EQCR}	=	Equivalent capacitance of routed array clock in pF
C_L	=	Output lead capacitance in pF
f_m	=	Average logic module switching rate in MHz
f_n	=	Average input buffer switching rate in MHz
f_p	=	Average output buffer switching rate in MHz
f_{q1}	=	Average first routed array clock rate in MHz (All families)

Fixed Capacitance Values for Actel FPGAs (pF)

Device Type	r_1 routed_Clk1
A10V10B	40
A10V20B	65

Determining Average Switching Frequency

To determine the switching frequency for a design, you must have a detailed understanding of the data input values to the circuit. The following guidelines are meant to represent worst-case scenarios so that they can be generally used to predict the upper limits of power dissipation. These guidelines are as follows:

Logic Modules (m)	90% of modules
Inputs switching (n)	#inputs/4
Outputs switching (p)	#outputs/4
First routed array clock loads (q_1)	40% of modules
Load capacitance (C_L)	35 pF
Average logic module switching rate (f_m)	F/10
Average input switching rate (f_n)	F/5
Average output switching rate (f_p)	F/10
Average first routed array clock rate (f_{q1})	F

Functional Timing Tests

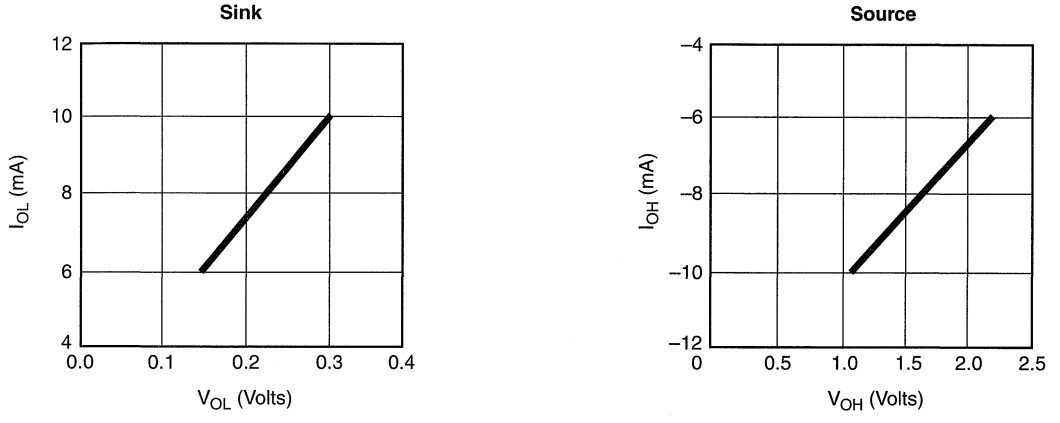
AC timing for logic module internal delays is determined after place and route. The ALS Timer utility displays actual timing parameters for circuit delays. ACT 1 devices are AC tested to a "binning" circuit specification.

The circuit consists of one input buffer + n logic modules + one output buffer (n = 16 for A10V10B; n = 28 for A10V20B).

The logic modules are distributed along two sides of the device, as inverting or non-inverting buffers. The modules are connected through programmed antifuses with typical capacitive loading.

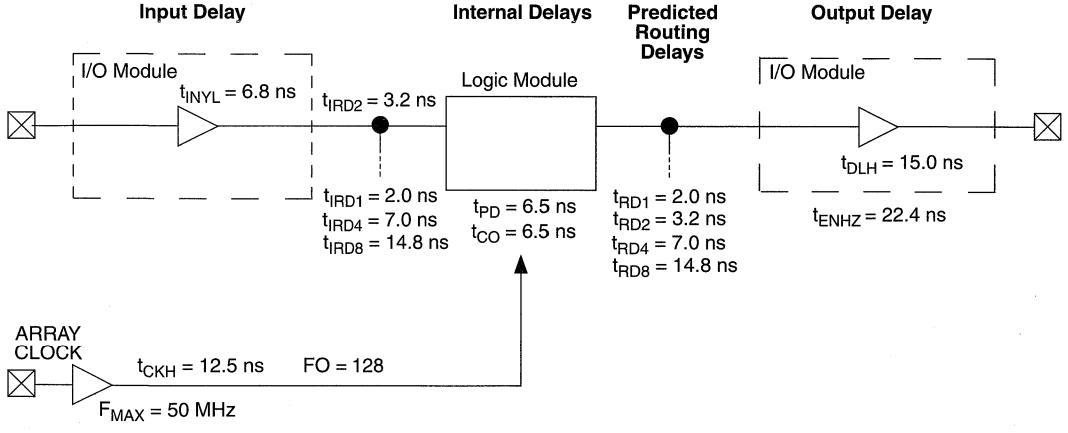
Propagation delay [$t_{PD} = (t_{PLH} + t_{PHL})/2$] is tested to the following AC test specifications.

Output Buffer Performance Derating



Note: *The above curves are based on characterizations of sample devices and are not completely tested on all devices.*

ACT 1 3.3 Volt Timing Model



Timing Characteristics

Timing characteristics for ACT 1 3.3 V devices fall into three categories: family dependent, device dependent, and design dependent. The input and output buffer characteristics are common to all family members. Internal routing delays are device dependent. Design dependency means actual delays are not determined until after placement and routing of the user design is complete. Delay values may then be determined by using the ALS Timer utility or performing simulation with post-layout delays.

Critical Nets and Typical Nets

Propagation delays are expressed only for typical nets, which are used for initial design performance evaluation. Critical net delays can then be applied to the most time-critical paths. Critical nets are determined by net property assignment prior to placement and routing. Up to 6% of the nets in a design may be designated as critical, while 90% of the nets in a design are typical.

Long Tracks

Some nets in the design use long tracks. Long tracks are special routing resources that span multiple rows, columns, or modules. Long tracks employ three and sometimes four antifuse connections. This increases capacitance and resistance, resulting in longer net delays for macros connected to long tracks. Typically, up to 6% of nets in a fully utilized device require long tracks. Long tracks contribute approximately 5 ns to 10 ns delay. This additional delay is represented statistically in higher fanout (FO=8) routing delays in the data sheet specifications section.

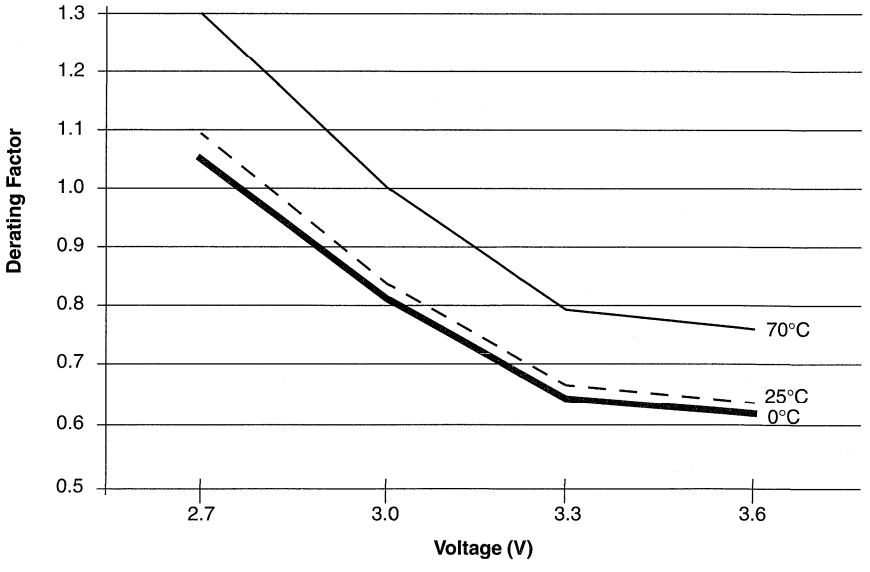
Timing Derating

A best case timing derating factor of 0.45 is used to reflect best case processing. Note that this factor is relative to the "standard speed" timing parameters, and must be multiplied by the appropriate voltage and temperature derating factors for a given application.

Temperature and Voltage Derating Factors (normalized to Worst-Case Commercial, T_J = 3.0 V, 70°C)

	0	25	70
2.7	1.05	1.09	1.30
3.0	0.81	0.84	1.00
3.3	0.64	0.67	0.79
3.6	0.62	0.64	0.76

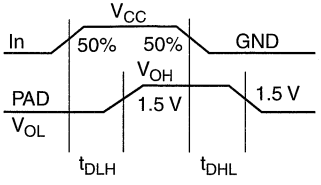
Junction Temperature and Voltage Derating Curves (normalized to Worst-Case Commercial, T_J = 3.0 V, 70°C)

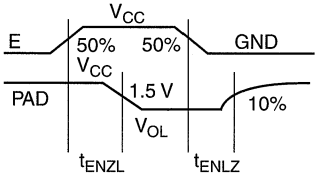


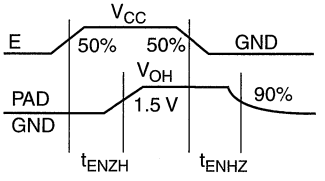
Note: *This derating factor applies to all routing and propagation delays.*

Parameter Measurement

Output Buffer Delays

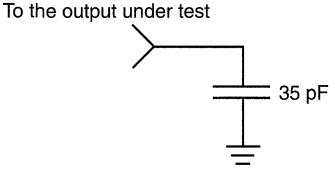




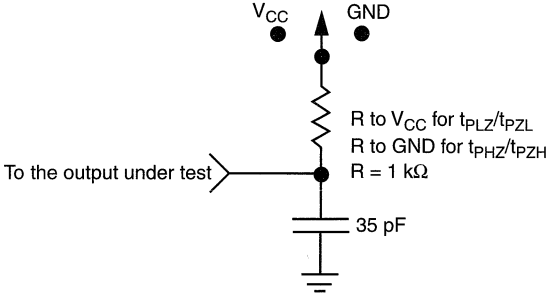


AC Test Loads

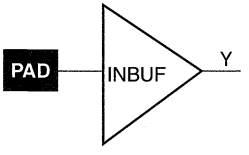
Load 1
(Used to measure propagation delay)

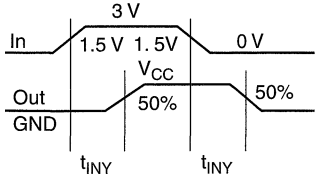


Load 2
(Used to measure rising/falling edges)

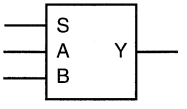


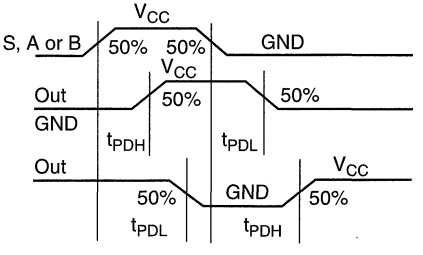
Input Buffer Delays





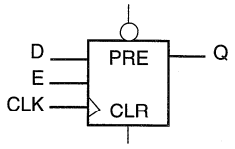
Module Delays



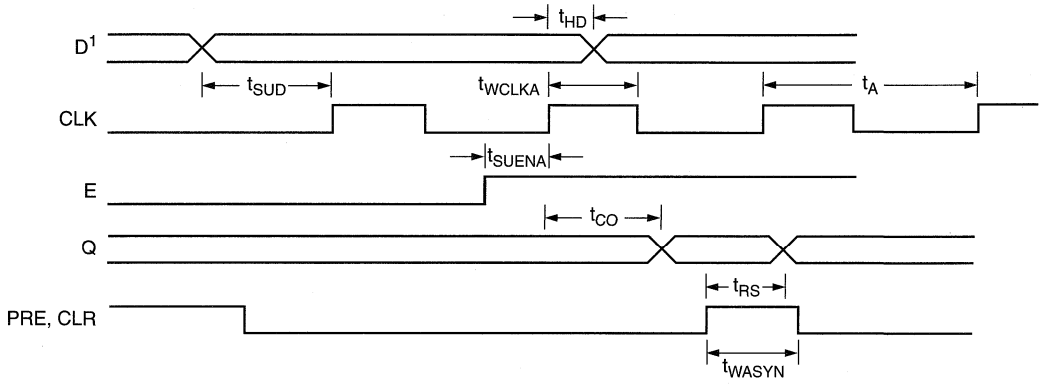


Sequential Module Timing Characteristics

Flip-Flops and Latches



(Positive edge triggered)



Note: *D represents all data functions involving A, B, and S for multiplexed flip-flops.*

ACT 1 Timing Characteristics

(Worst-Case Commercial Conditions, V_{CC} = 3.0 V, T_J = 70°C)

Logic Module Propagation Delays				
Parameter	**Description**	**Min.**	**Max.**	**Units**
t_{PD1}	Single Module		6.5	ns
t_{PD2}	Dual Module Macros		15.1	ns
t_{CO}	Sequential Clk to Q		6.5	ns
t_{GO}	Latch G to Q		6.5	ns
t_{RS}	Flip-Flop (Latch) Reset to Q		6.5	ns
Predicted Routing Delays[1]				
t_{RD1}	FO=1 Routing Delay		2.0	ns
t_{RD2}	FO=2 Routing Delay		3.2	ns
t_{RD3}	FO=3 Routing Delay		4.8	ns
t_{RD4}	FO=4 Routing Delay		7.0	ns
t_{RD8}	FO=8 Routing Delay		14.8	ns
Sequential Timing Characteristics[2]				
t_{SUD}	Flip-Flop (Latch) Data Input Setup	10.0		ns
t_{HD}	Flip-Flop (Latch) Data Input Hold	0.0		ns
t_{SUENA}	Flip-Flop (Latch) Enable Setup	10.0		ns
t_{HENA}	Flip-Flop (Latch) Enable Hold	0.0		ns
t_{WCLKA}	Flip-Flop (Latch) Clock Active Pulse Width	9.8		ns
t_{WASYN}	Flip-Flop (Latch) Asynchronous Pulse Width	9.8		ns
t_A	Flip-Flop Clock Input Period	20.0		ns
f_{MAX}	Flip-Flop (Latch) Clock Frequency (FO = 128)		50	MHz

Notes:

1. *Routing delays are for typical designs across worst-case operating conditions. These parameters should be used for estimating device performance. Post-route timing analysis or simulation is required to determine actual worst-case performance. Post-route timing is based on actual routing delay measurements performed on the device prior to shipment.*
2. *Setup times assume fanout of 3. Further testing information can be obtained from the ALS Timer utility.*

ACT 1 Timing Characteristics (continued)

(Worst-Case Commercial Conditions)

Input Module Propagation Delays					
Parameter	**Description**		**Min.**	**Max.**	**Units**
t_{INYH}	Pad to Y High			6.8	ns
t_{INYL}	Pad to Y Low			6.8	ns
Input Module Predicted Routing Delays[1]					
t_{IRD1}	FO=1 Routing Delay			2.0	ns
t_{IRD2}	FO=2 Routing Delay			3.2	ns
t_{IRD3}	FO=3 Routing Delay			4.8	ns
t_{IRD4}	FO=4 Routing Delay			7.0	ns
t_{IRD8}	FO=8 Routing Delay			14.8	ns
Global Clock Network					
t_{CKH}	Input Low to High	FO = 16 FO = 128		6.7 7.9	ns
t_{CKL}	Input High to Low	FO = 16 FO = 128		8.8 10.0	ns
t_{PWH}	Minimum Pulse Width High	FO = 16 FO = 128	8.9 9.8		ns
t_{PWL}	Minimum Pulse Width Low	FO = 16 FO = 128	8.9 9.8		ns
t_{CKSW}	Maximum Skew	FO = 16 FO = 128		1.5 2.4	ns
t_P	Minimum Period	FO = 16 FO = 128	18.2 20		ns
f_{MAX}	Maximum Frequency	FO = 16 FO = 128		55 50	MHz

Note:

1. *These parameters should be used for estimating device performance. Optimization techniques may further reduce delays by 0 to 8 ns. Routing delays are for typical designs across worst-case operating conditions. Post-route timing analysis or simulation is required to determine actual worst-case performance. Post-route timing is based on actual routing delay measurements performed on the device prior to shipment.*

ACT 1 Timing Characteristics (continued)

(Worst-Case Commercial Conditions)

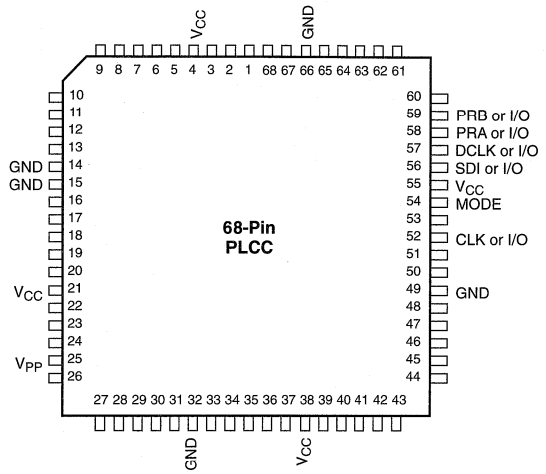
Output Module Timing				
Parameter	**Description**	**Min.**	**Max.**	**Units**
TTL Output Module Timing[1]				
t_{DLH}	Data to Pad High		15.0	ns
t_{DHL}	Data to Pad Low		16.7	ns
t_{ENZH}	Enable Pad Z to High		14.8	ns
t_{ENZL}	Enable Pad Z to Low		17.7	ns
t_{ENHZ}	Enable Pad High to Z		22.4	ns
t_{ENLZ}	Enable Pad Low to Z		20.2	ns
d_{TLH}	Delta Low to High		0.13	ns/pF
d_{THL}	Delta High to Low		0.17	ns/pF
CMOS Output Module Timing[1]				
t_{DLH}	Data to Pad High		17.7	ns
t_{DHL}	Data to Pad Low		14.2	ns
t_{ENZH}	Enable Pad Z to High		13.4	ns
t_{ENZL}	Enable Pad Z to Low		18.5	ns
t_{ENHZ}	Enable Pad High to Z		22.4	ns
t_{ENLZ}	Enable Pad Low to Z		20.2	ns
d_{TLH}	Delta Low to High		0.22	ns/pF
d_{THL}	Delta High to Low		0.13	ns/pF

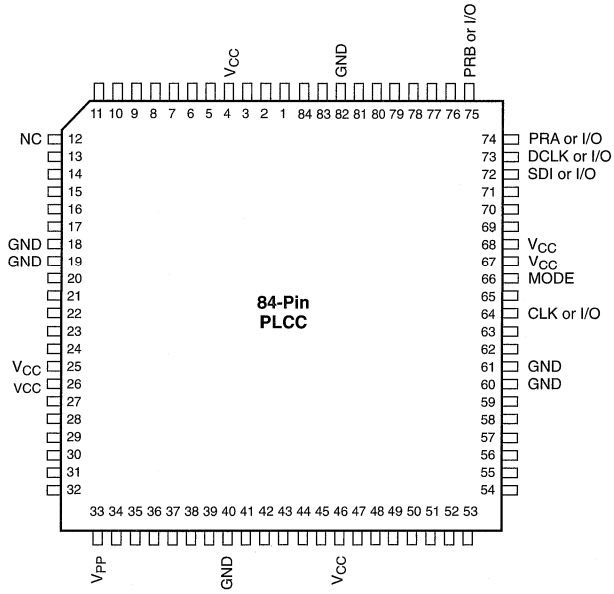
Note:

1. Delays based on 35 pF loading.

Package Pin Assignments

(Top View)



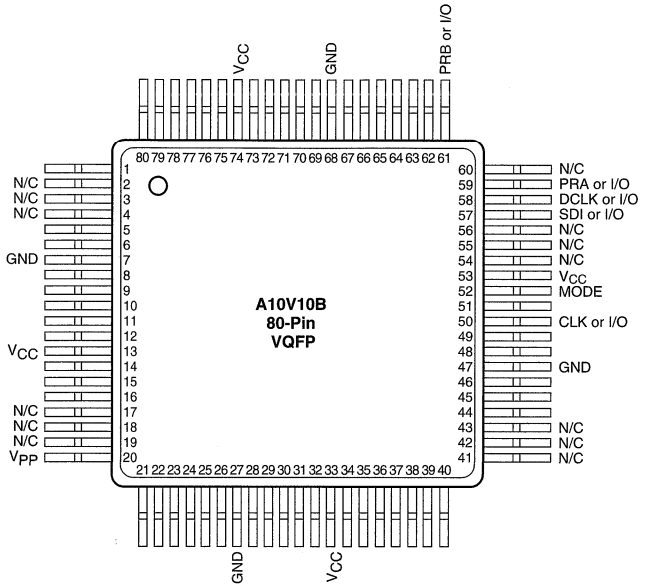


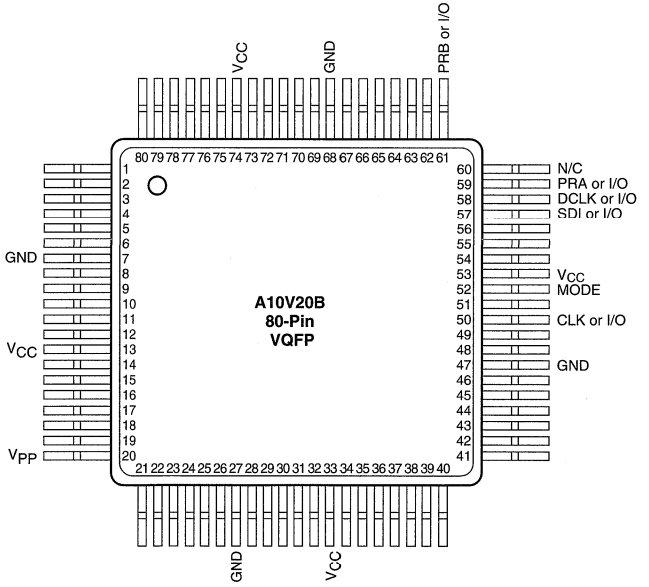
Notes:

1. *V_{PP} must be terminated to V_{CC} except during device programming.*
2. *MODE must be terminated to circuit ground, except during device programming or debugging.*
3. *Unused I/O pins are designated as outputs by ALS and are driven low.*
4. *All unassigned pins are available for use as I/Os.*

Package Pin Assignments (continued)

(Top View)





Notes:

1. *V_{PP} must be terminated to V_{CC} except during device programming.*
2. *MODE must be terminated to circuit ground, except during device programming or debugging.*
3. *Unused I/O pins are designated as outputs by ALS and are driven low.*
4. *All unassigned pins are available for use as I/Os.*

ACT™ 2
Field Programmable Gate Arrays

Features

- Up to 8000 Gate Array Gates (20,000 PLD equivalent gates)
- Replaces up to 200 TTL Packages
- Replaces up to eighty 20-Pin PAL® Packages
- Design Library with over 500 Macro Functions
- Single-Module Sequential Functions
- Wide-Input Combinatorial Functions
- Up to 1232 Programmable Logic Modules
- Up to 998 Flip-Flops
- Datapath Performance at 105 MHz
- 16-Bit Accumulator Performance to 39 MHz
- Two In-Circuit Diagnostic Probe Pins Support Speed Analysis to 50 MHz
- Two High-Speed, Low-Skew Clock Networks
- I/O Drive to 10 mA
- Nonvolatile, User Programmable
- Logic Fully Tested Prior to Shipment
- 1.0-micron CMOS Technology

Product Family Profile

Device	A1225A	A1240A	A1280A
Capacity			
Gate Array Equivalent Gates	2,500	4,000	8,000
PLD Equivalent Gates	6,250	10,000	20,000
TTL Equivalent Packages	63	100	200
20-Pin PAL Equivalent Packages	25	40	80
Logic Modules	451	684	1,232
S-Modules	231	348	624
C-Modules	220	336	608
Flip-Flops (maximum)	382	568	998
Routing Resources			
Horizontal Tracks/Channel	36	36	36
Vertical Tracks/Channel	15	15	15
PLICE Antifuse Elements	250,000	400,000	750,000
User I/Os (maximum)	83	104	140
Packages[1]	100 CPGA 100 PQFP 100 VQFP 84 PLCC	132 CPGA 144 PQFP 176 TQFP 84 PLCC	176 CPGA 160 PQFP 176 TQFP 84 PLCC 172 CQFP
Performance[2]			
16-Bit Prescaled Counters	105 MHz	100 MHz	85 MHz
16-Bit Loadable Counters	70 MHz	69 MHz	67 MHz
16-Bit Accumulators	39 MHz	38 MHz	36 MHz

Notes:

1. *See product plan on page 1-53 for package availability.*
2. *Performance is based on '–2' speed devices at commercial worst-case operating conditions using PREP Benchmarks, Suite #1, Version 1.2, dated 3-28-93, any analysis is not endorsed by PREP.*

Description

The ACT™ 2 family represents Actel's second generation of field programmable gate arrays (FPGAs). The ACT 2 family presents a two-module architecture, consisting of C-modules and S-modules. These modules are optimized for both combinatorial and sequential designs. Based on Actel's patented channeled array architecture, the ACT 2 family provides significant enhancements to gate density and performance while maintaining downward compatibility with the ACT 1 design environment and upward compatibility with the ACT 3 design environment. The devices are implemented in silicon gate, 1.0-µm, two-level metal CMOS, and employ Actel's PLICE® antifuse technology. This revolutionary architecture offers gate array design flexibility, high performance, and fast time-to-production with user programming. The ACT 2 family is supported by the Designer and Designer Advantage Systems, which offers automatic pin assignment, validation of electrical and design rules, automatic placement and routing, timing analysis, user programming, and diagnostic probe capabilities. The systems are supported on the following platforms: 386/486™ PC, Sun™, and HP™ workstations. The systems provide CAE interfaces to the following design environments: Cadence, Viewlogic®, Mentor Graphics®, and OrCAD™.

Ordering Information

A1280 A – 1 PG 176 C

- **C** — Application (Temperature Range)
  - C = Commercial (0 to +70°C)
  - I = Industrial (–40 to +85°C)
  - M = Military (–55 to +125°C)
  - B = MIL-STD-883
  - E = Extended Military Flow
- **176** — Package Lead Count
- **PG** — Package Type
  - PL = Plastic J-Leaded Chip Carrier
  - PQ = Plastic Quad Flatpack
  - CQ = Ceramic Quad Flatpack
  - PG = Ceramic Pin Grid Array
  - TQ = Thin (1.4 mm) Quad Flatpack
  - VQ = Very Thin (1.0 mm) Quad Flatpack
- **1** — Speed Grade
  - Blank = Standard Speed
  - –1 = Approximately 15% faster than Standard
  - –2 = Approximately 25% faster than Standard
- **A** — Die Revision
  - A =1.0-µm CMOS process
- **A1280** — Part Number
  - A1225 =2500 Gates
  - A1240 =4000 Gates
  - A1280 =8000 Gates

Product Plan[1]

	Speed Grade*			Application				
	Std	–1	–2	C	I	M	B	E
A1225A Device								
100-pin Ceramic Pin Grid Array (PG)	✔	✔	✔	✔	—	—	—	—
100-pin Plastic Quad Flatpack (PQ)	✔	✔	✔	✔	✔	—	—	—
100-pin Very Thin (1.0 mm) Quad Flatpack (VQ)	✔	✔	✔	✔	—	—	—	—
84-pin Plastic Leaded Chip Carrier (PL)	✔	✔	✔	✔	✔	—	—	—
A1240A Device								
132-pin Ceramic Pin Grid Array (PG)	✔	✔	✔	✔	—	✔	✔	—
176-pin Thin (1.4 mm) Quad Flatpack (TQ)	✔	✔	✔	✔	—	—	—	—
144-pin Plastic Quad Flatpack (PQ)	✔	✔	✔	✔	✔	—	—	—
84-pin Plastic Leaded Chip Carrier (PL)	✔	✔	✔	✔	✔	—	—	—
A1280A Device								
176-pin Ceramic Pin Grid Array (PG)	✔	✔	✔	✔	—	✔	✔	—
176-pin Thin (1.4 mm) Quad Flatpack (TQ)	✔	✔	✔	✔	—	—	—	—
160-pin Plastic Quad Flatpack (PQ)	✔	✔	✔	✔	✔	—	—	—
172-pin Ceramic Quad Flatpack (CQ)	✔	✔	✔	✔	—	✔	✔	✔

Applications: C = Commercial, I = Industrial, M = Military, B = MIL-STD-883, E = Extended Flow

Availability: ✔ = Available, P = Planned, — = Not Planned

* Speed Grade: –1 = Approx. 15% faster than Standard, –2 = Approx. 25% faster than Standard

Note:

1. Please consult Actel representatives for current availability.

Device Resources

Device Series	Logic Modules	Gates	User I/Os									
			CPGA			PQFP			PLCC	CQFP	TQFP	VQFP
			176-pin	132-pin	100-pin	160-pin	144-pin	100-pin	84-pin	172-pin	176-pin	100-pin
A1225A	451	2500	—	—	83	—	—	83	72	—	—	83
A1240A	684	4000	—	104	—	—	104	—	72	—	104	—
A1280A	1232	8000	140	—	—	125	—	—	72	140	140	—

Pin Description

CLKA **Clock A (Input)**

TTL Clock input for clock distribution networks. The Clock input is buffered prior to clocking the logic modules. This pin can also be used as an I/O.

CLKB **Clock B (Input)**

TTL Clock input for clock distribution networks. The Clock input is buffered prior to clocking the logic modules. This pin can also be used as an I/O.

DCLK **Diagnostic Clock (Input)**

TTL Clock input for diagnostic probe and device programming. DCLK is active when the MODE pin is HIGH. This pin functions as an I/O when the MODE pin is LOW.

GND **Ground**

LOW supply voltage.

I/O **Input/Output (Input, Output)**

The I/O pin functions as an input, output, three-state, or bidirectional buffer. Input and output levels are compatible with standard TTL and CMOS specifications. Unused I/O pins are automatically driven LOW by the ALS software.

MODE **Mode (Input)**

The MODE pin controls the use of multifunction pins (DCLK, PRA, PRB, SDI). When the MODE pin is HIGH, the special functions are active. When the MODE pin is LOW, the pins function as I/Os.

NC **No Connection**

This pin is not connected to circuitry within the device.

PRA **Probe A (Output)**

The Probe A pin is used to output data from any user-defined design node within the device. This independent diagnostic pin is used in conjunction with the Probe B pin to allow real-time diagnostic output of any signal path within the device. The Probe A pin can be used as a user-defined I/O when debugging has been completed. The pin's probe capabilities can be permanently disabled to protect programmed design confidentiality. PRA is active when the MODE pin is HIGH. This pin functions as an I/O when the MODE pin is LOW.

PRB **Probe B (Output)**

The Probe B pin is used to output data from any user-defined design node within the device. This independent diagnostic pin is used in conjunction with the Probe A pin to allow real-time diagnostic output of any signal path within the device. The Probe B pin can be used as a user-defined I/O when debugging has been completed. The pin's probe capabilities can be permanently disabled to protect programmed design confidentiality. PRB is active when the MODE pin is HIGH. This pin functions as an I/O when the MODE pin is LOW.

SDI **Serial Data Input (Input)**

Serial data input for diagnostic probe and device programming. SDI is active when the MODE pin is HIGH. This pin functions as an I/O when the MODE pin is LOW.

V_{CC} **5 V Supply Voltage**

HIGH supply voltage.

V_{KS} **Programming Voltage**

Supply voltage used for device programming. This pin must be connected to GND during normal operation.

V_{PP} **Programming Voltage**

Supply voltage used for device programming. This pin must be connected to V_{CC} during normal operation.

V_{SV} **Programming Voltage**

Supply voltage used for device programming. This pin must be connected to V_{CC} during normal operation.

ACT 2 Architecture

This section of the data sheet is meant to familiarize the user with the architecture of ACT 2 family devices. A generic description of the family will be presented first, followed by a detailed description of the logic blocks, the routing structure, the antifuses, and the special function circuits. Diagrams for the ACT 2 devices are provided at the end of the data sheet. The additional circuitry required to program and test the devices will not be covered.

Array Topology

The ACT 2 family architecture is composed of five key building blocks: Logic modules, I/O modules, Routing Tracks, Global Clock Networks, and Probe Circuits. The basic structure is similar for all devices in the family, differing only in the number of rows, columns, or I/Os (see Table 1).

The logic and I/O modules are arranged in a two-dimensional array (Figure 1). There are three types of modules: Logic, I/O, and Bin. Logic and I/O modules are available as user resources. Bin modules are used during testing and are not available to users.

Table 1 • Array Sizes

Device	Rows	Columns	Logic	I/O
A1225A	13	46	451	83
A1240A	14	62	684	104
A1280A	18	82	1232	140

Logic Modules

Logic modules are classified into two types: combinatorial (C-modules) and sequential (S-modules) (see Figures 2 and 3). The C-module is an enhanced version of the ACT 1 family logic module optimized to implement high fanin combinatorial macros, such as 5-input AND, and 5-input OR. The full ACT 2 combinatorial logic module is available for use as the CM8 hard macro. The S-module is designed to implement high-speed flip-flop functions within a single

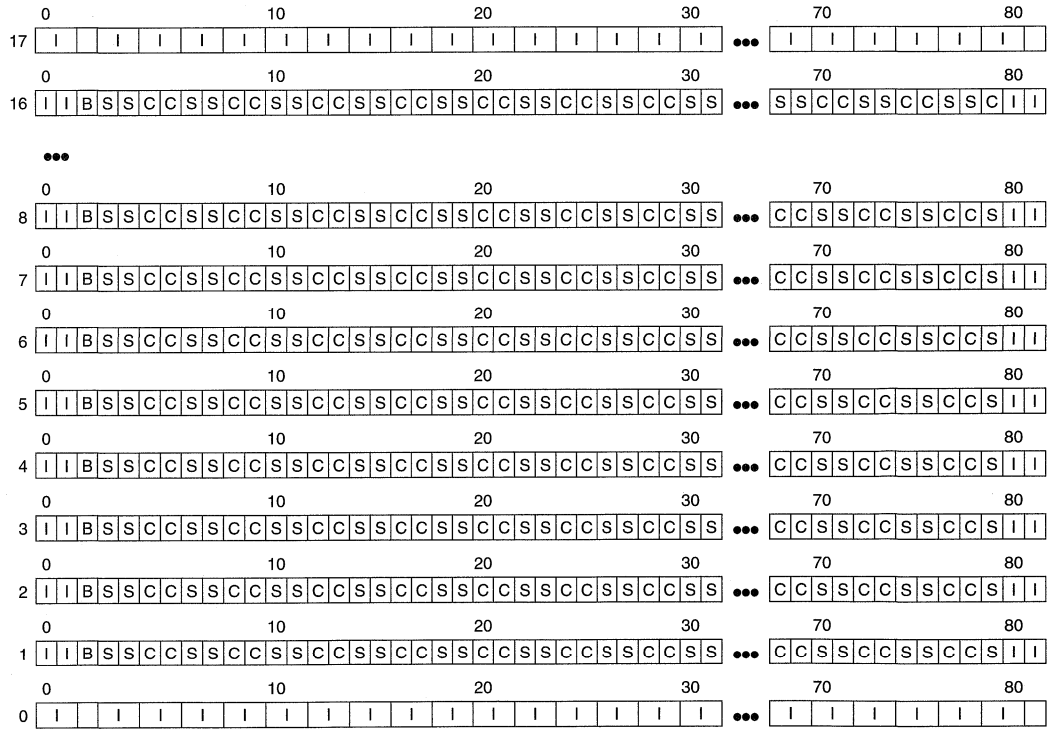


Figure 1 • A1280A Simplified Floor Plan

module. S-modules also include combinatorial logic, which allows an additional level of logic to be implemented without additional propagation delay. C-modules and S-modules are arranged in pairs called module-pairs. Module-pairs are arranged in alternating pairs (shown in Figure 1) and make up the bulk of the array. This arrangement allows the placement software to support two-module macros of four types (CC, CS, SC, and SS). I/O modules are arranged around the periphery of the array.

The combinatorial module (shown in Figure 2) implements the following function:

$Y = !S1 * !S0 * D00 + !S1 * S0 * D01 + S1 * !S0 * D10 + S1 * S0 * D11$

where:

$S0 = A0 * B0$

$S1 = A1 + B1$

The sequential module implements this same function Y (except that S0 = A0 only, since the B0 input is used for reset), followed by a sequential block. The sequential block can implement either a D-type flip-flop or a transparent latch. It can also be fully transparent so that the S-modules can be used to implement purely combinatorial functions. The function of the sequential module is determined by the macro selection from the design library of hard macros. Allowable S-module implementations are shown in Figure 3.

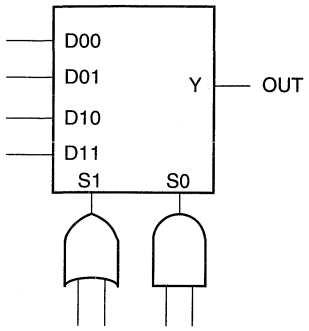
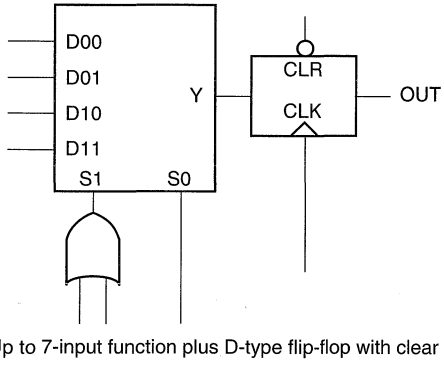
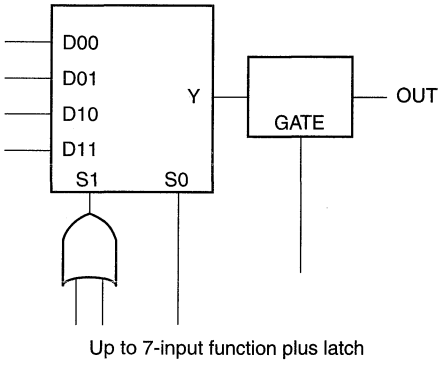
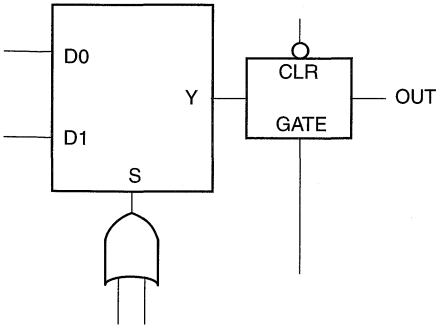


***Figure 2** • C-module Implementation*







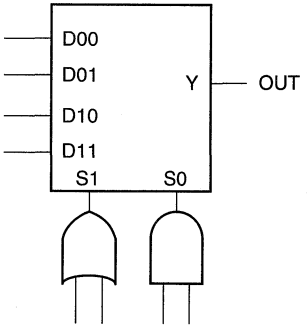


***Figure 3** • S-module Implementations*

I/Os

The I/O architecture consists of pad drivers located near the bonding pads and I/O modules located in the array. Top/bottom I/O modules are located in the top and bottom rows respectively. Side I/O modules occupy the leftmost two columns and the rightmost two columns of the array. The function of all I/O modules is identical, but the top/bottom I/O modules have a different routing interface to the array than the side I/O modules. I/Os implement a variety of user functions determined by library macro selection.

Special Purpose I/Os

Certain I/O pads are temporarily used for programming and testing the device. During normal user operation, these special I/O pads are identical to other I/O pads. The following special I/O pads and their functions are shown in Table 2.

Table 2 • Special I/O Pads

SDI	Serial Data In
DCLK	Serial Data Clock In
PRA	Probe A Output
PRB	Probe B Output

Two other pads, CLKA and CLKB, also differ from normal I/Os in that they can be used to drive the global clock networks. Power, Ground, and Programming pads are not considered I/O functions. Their function is summarized as follows:

V_{CC}	Power
GND	Circuit Ground
V_{SV}, V_{KS}, V_{PP}	Programming Pads
MODE	Program/Debug Control

I/O Pads

I/O pads are located on the periphery of the die and consist of the bonding pad, the high-drive CMOS drivers, and the TTL level-shifter inputs. Each I/O pad is associated with a specific I/O module. Connections from the I/O pad to the I/O module are made using the signals DATAOUT, DATAIN, and EN (shown in Figure 4).

I/O Modules

There are two types of I/O modules: side and top/bottom. The I/O module schematic is shown in Figure 5. In the side I/O modules, there are two inputs supplying the data to be output from the chip UO1 and UO2. (UO stands for user output.) Two are used so that the router can choose to take the signal from either the routing channel above or the routing channel below the I/O module. The top/bottom I/O modules interact with only one channel and therefore have only one UO input.

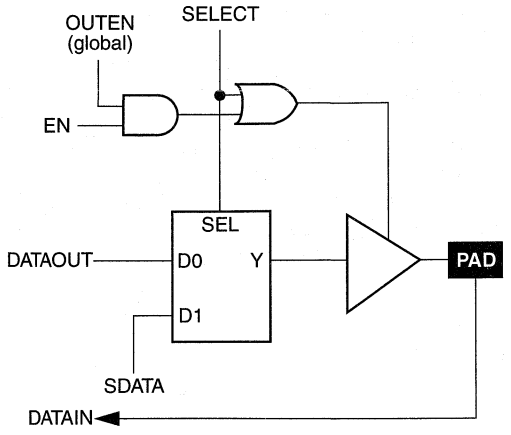


Figure 4 • I/O Pad Signals

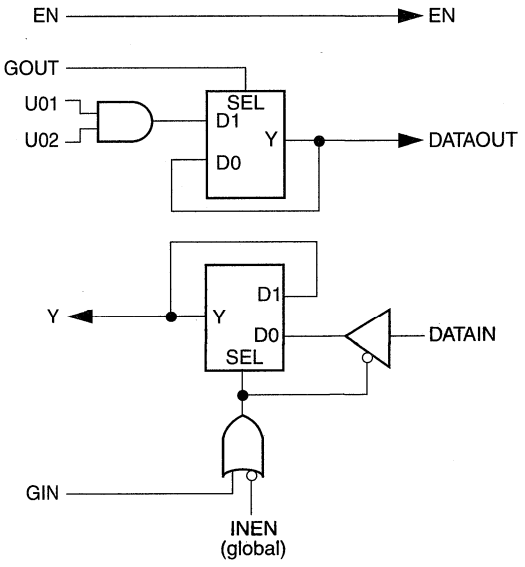


Figure 5 • I/O Module

The EN input enables the tristate output buffer. The global signals INEN and OUTEN (Figure 4 and Figure 5) are used to disable the inputs and outputs during certain test modes. Latches are provided in the input and output path. When GOUT is high, the latch is transparent. The latch can be used as the second stage of a rising-edge flip-flop. GIN is the reverse of GOUT. When GIN is high, the input data is latched; when it is low, the input latch becomes transparent.

The output of the module, Y, is used for data being input to the chip. Side I/O modules have a dedicated output segment for Y extending into the routing channels above and below (similar to logic modules). Side I/O modules may also connect to the array through nondedicated Long Vertical Tracks (LVTs). Top/Bottom I/O modules have no dedicated output segment. Signals coming into the chip from the top or bottom must be routed using F-fuses and LVTs (F-fuses and LVTs are explained in detail in the routing section). I/O signals connected to I/O modules on either the top or bottom of the array may incur a delay penalty over signals connected to I/O modules on the sides.

Hard Macros

Designing within the Actel design environment is accomplished using a building block approach. Over 350 logic function macros are provided in the ACT 2 design library. Hard macro logic functions range from simple SSI gates such as AND, NOR, and Exclusive OR to more complex functions such as flip-flops with 4:1 Multiplexed Data inputs. Hard macros are implemented in the ACT 2 architecture by using one or more C-modules or S-modules. Over 200 of the macros are implemented in a single module, while several two-module macros are also available. Two-module hard macros always utilize a module-pair, either SS, CC, CS, or SC. Because one- and two-module macros have small propagation delay variances, their performances can be predicted very accurately. Hard macro propagation delays are specified in the data sheet. Soft macros comprise multiple hard macros connected together to form complex functions. These functions range from MSI functions to 16-bit counters and accumulators. A large number of TTL equivalent hard and soft macros are also provided. Soft macro delays are not specified in the data sheet.

Routing Structure

The ACT 2 architecture uses Vertical and Horizontal routing tracks to interconnect the various logic and I/O modules. These routing tracks are metal interconnects that may either be of continuous length or broken into pieces called segments. Segments can be joined together at the ends using antifuses to increase their lengths up to the full length of the track.

Horizontal Routing

Horizontal channels are located between the rows of modules and are composed of several routing tracks. The horizontal routing tracks within the channel are divided into one or more segments. The minimum horizontal segment length is the width of a module-pair, and the maximum horizontal segment length is the full length of the channel. Any segment that spans more than one-third the row length is considered a long horizontal segment. A typical channel is shown in Figure 6. Nondedicated horizontal routing tracks are used to route signal nets. Dedicated routing tracks are used for the global clock networks and for power and ground tie-off tracks.

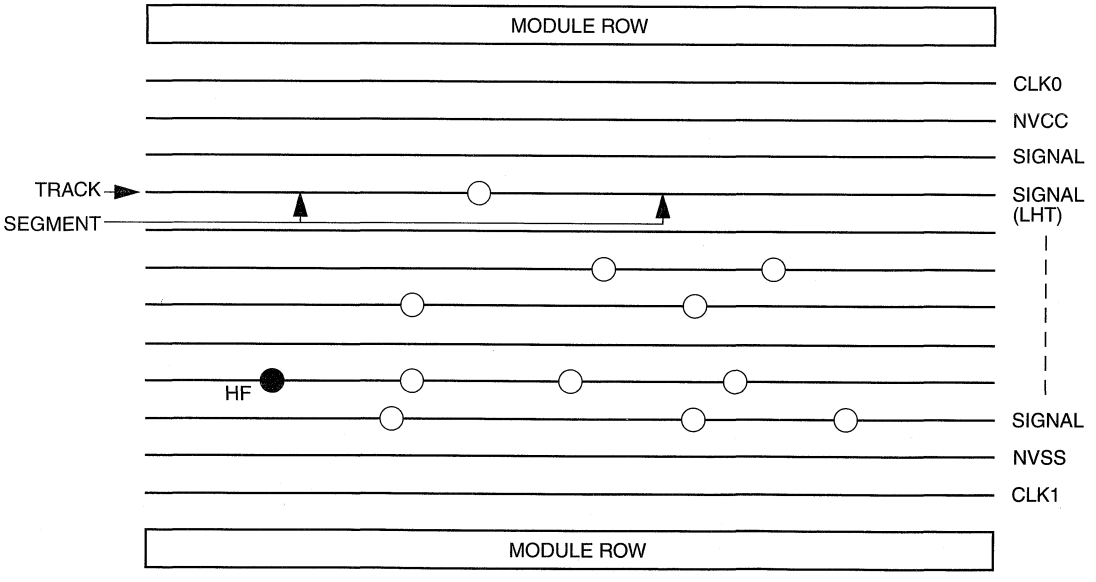


Figure 6 • Horizontal Routing Tracks and Segments

Vertical Routing

Other tracks run vertically through the module. Vertical tracks are of three types: input, output, and long. Vertical tracks are also divided into one or more segments. Each segment in an input track is dedicated to the input of a particular module. Each segment in an output track is dedicated to the output of a particular module. Long segments are uncommitted and can be assigned during routing. Each output segment spans four channels (two above and two below), except near the top and bottom of the array where edge effects occur. LVTs contain either one or two segments. An example of vertical routing tracks and segments is shown in Figure 7.

Antifuse Structures

An antifuse is a "normally open" structure as opposed to the normally closed fuse structure used in PROMs or PALs. The use of antifuses to implement a Programmable Logic Device results in highly testable structures as well as efficient programming algorithms. The structure is highly testable because there are no preexisting connections; therefore, temporary connections can be made using pass transistors. These temporary connections can isolate individual antifuses to be programmed as well as isolate individual circuit structures to be tested. This can be done both before and after programming. For example, all metal tracks can be tested for continuity and shorts between adjacent tracks, and the functionality of all logic modules can be verified.

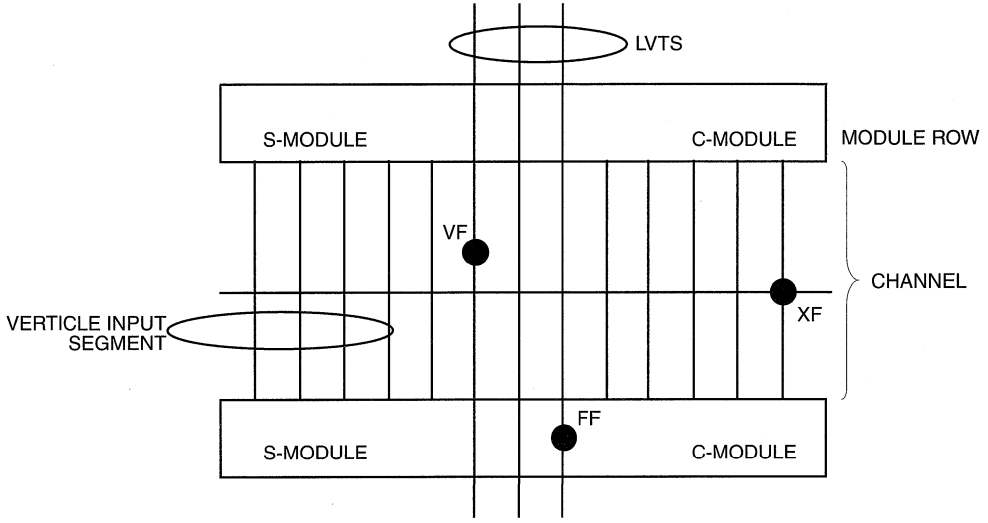


Figure 7 • Vertical Routing Tracks and Segments

Antifuse Connections

Four types of antifuse connections are used in the routing structure of the ACT 2 array. (The physical structure of the antifuse is identical in each case; only the usage differs.) The four types are:

XF	Cross-connected antifuse	Most intersections of horizontal and vertical tracks have an XF that connects the perpendicular tracks.
HF	Horizontally connected antifuses	Adjacent segments in the same horizontal tracks are connected end-to-end by an HF.
VF	Vertically connected antifuse	Some long vertical tracks are divided into two segments. Adjacent long segments are connected end-to-end by a VF.
FF	"Fast-Fuse" antifuse	The FF connects a module output directly to a long vertical track.

Examples of all four antifuse connections are shown in Figures 6 and 7.

Antifuse Programming

The ACT 2 family uses the PLICE antifuse developed by Actel. The PLICE element is programmed by placing a high voltage (~17 V) across the element and supplying current (~5 mA) for a short duration (<1 ms). In the ACT 2 architecture, most antifuses are programmed to ~500 ohms resistance, except for the F-fuses which are programmed to ~250 ohms. The programming circuits are transparent to the user.

Clock Networks

Two low-skew, high fanout clock distribution networks are provided in the ACT 2 architecture (Figure 8). These networks are referred to as CLK0 and CLK1. Each network has a clock module (CLKMOD) that selects the source of the clock signal and may be driven as follows:

1. externally from the CLKA pad
2. externally from the CLKB pad
3. internally from the CLKINA input
4. internally from the CLKINB input

The clock modules are located in the top row of I/O modules. Clock drivers and a dedicated horizontal clock track are located in each horizontal routing channel.

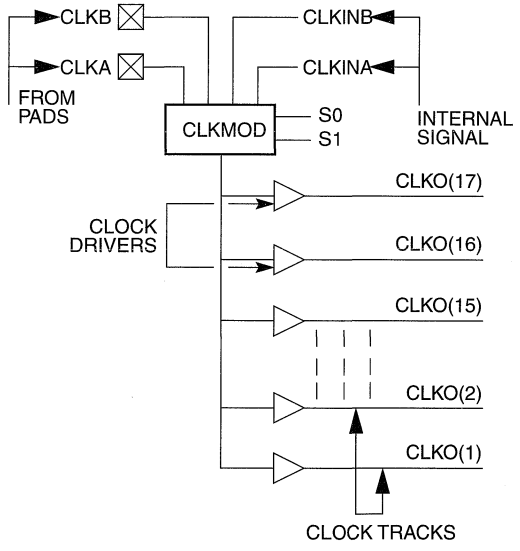


Figure 8 • Clock Networks

The user controls the clock module by selecting one of two clock macros from the macro library. The macro CLKBUF is used to connect one of the two external clock pins to a clock network, and the macro CLKINT is used to connect an internally generated clock signal to a clock network. Since both clock networks are identical, the user does not care whether CLK0 or CLK1 is being used.

The clock input pads may also be used as normal I/Os, bypassing the clock networks.

Module Interface

Connections to logic and I/O modules are made through vertical segments that connect to the module inputs and outputs. These vertical segments lie on vertical tracks that span the entire height of the array.

Module Input Connections

The tracks dedicated to Module inputs are segmented by pass transistors in each module row. During normal user operation, the pass transistors are inactive (off), which isolates the inputs of a module from the inputs of the module directly above or below it. During certain test modes, the pass transistors are active (on) to verify the continuity of the metal tracks. Vertical input segments span only one channel. Inputs to the array modules come either from the channel above or the channel below. The logic modules are arranged so that half of the inputs are connected to the channel above and half of the inputs to segments in the channel below (Figure 9).

Module Output Connections

Module outputs have dedicated output segments. Output segments extend vertically two channels above and two channels below, except at the top or bottom of the array. Output segments twist, as shown in Figure 9, so that only four vertical tracks are required.

LVT Connections

Outputs may also connect to nondedicated segments (LVTs). Each module-pair in the array shares three LVTs that span the length of column as shown in Figure 9. Any module in the column pair can connect to one of the LVTs in the column using an FF connection. The FF connection uses antifuses connected directly to the driver stage of the module output, bypassing the isolation transistor. FF antifuses are programmed at a higher current level than HF, VF, or XF antifuses to produce a lower resistance value.

Antifuse Connections

In general, every intersection of a vertical segment and a horizontal segment contains an unprogrammed antifuse (XF-type). One exception is in the case of the clock networks.

Clock Connections

To minimize loading on the clock networks, only a subset of inputs has fuses on the clock tracks. Only a few of the C-module and S-module inputs can be connected to the clock networks. To further reduce loading on the clock network, only a subset of the horizontal routing tracks can connect to the clock inputs of the S-module. Both of these are illustrated in Figure 10.

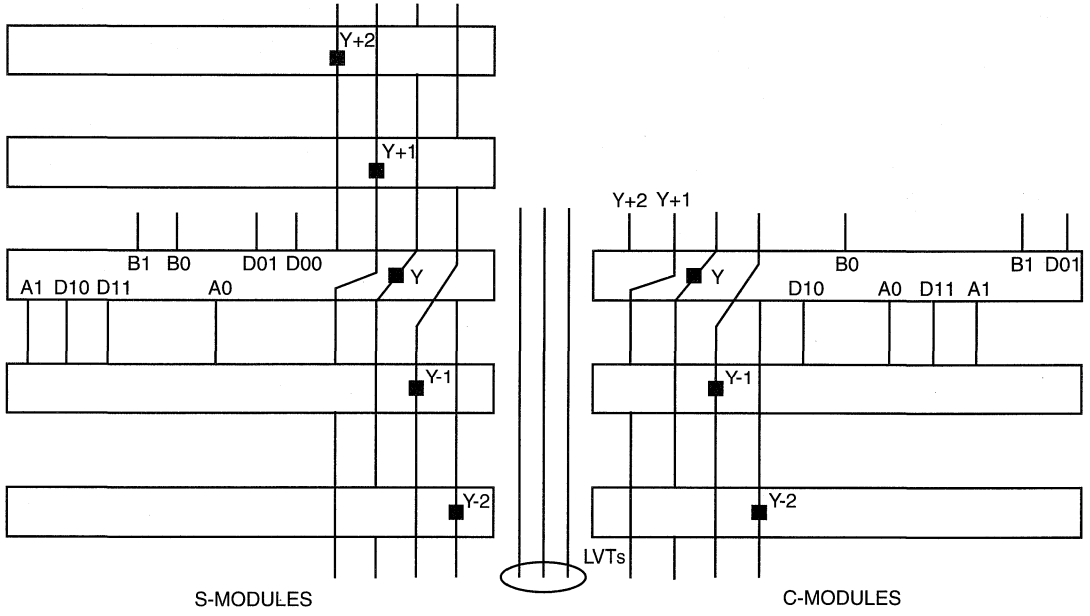


Figure 9 • Logic Module Routing Interface

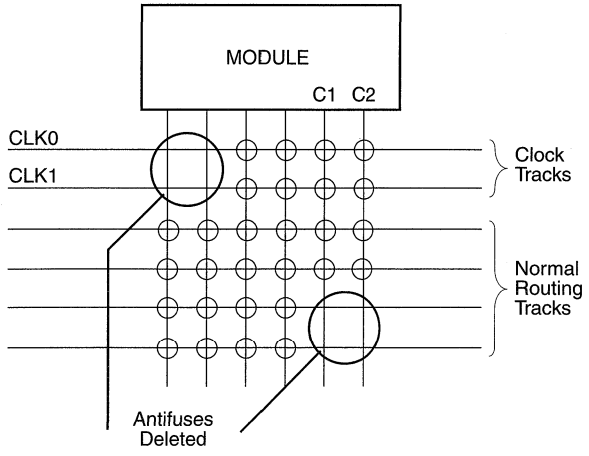


Figure 10 • Fuse Deletion on Clock Networks

Programming and Test Circuits

The array of logic and I/O modules is surrounded by test and programming circuits controlled by the external pins: MODE, SDI, and DCLK. When MODE is low (GND), the device is in normal or user mode. When MODE is high (V_{CC}), the device is placed into one of several programming or test states. The SDI pin (when MODE is high) is used to input serial data to the Mode Register and various address registers surrounding the array. Data is clocked into these registers using the DCLK pin. The registers are connected as a long series of shift registers as shown in Figure 11. The Mode register determines the test or programming state of the device. Many of the test modes are used during wafer sort and final test at the factory. Other test modes are used during programming with the Activator® 2, and some of the modes are available only after programming. The Actionprobe® function is one such function available to users.

Actionprobe

If a device has been successfully programmed and the security fuse has not been programmed, any internal logic or I/O module output can be observed using the Actionprobe circuitry and the PRA and/or PRB pins. The Actionprobe diagnostic system provides the software and hardware required to perform real-time debugging. The software automatically performs the following functions.

A pattern of *ones* and *zeros* is shifted into the device from the SDI pin at each positive edge transition of DCLK. The complete sequence contains 10 bits of counter, 21 bits of Mode Register, n bits of zeros (filler of unused fields, where n depends on the particular device type), R bits of X2, C bits of Y2, R bits of X1, C bits of Y1, and a stop bit ("0" or "1"). After the stop bit has been shifted in, DCLK is left high. X1 and Y1 represent the (X,Y) location in the array for the Actionprobe output, PRA.

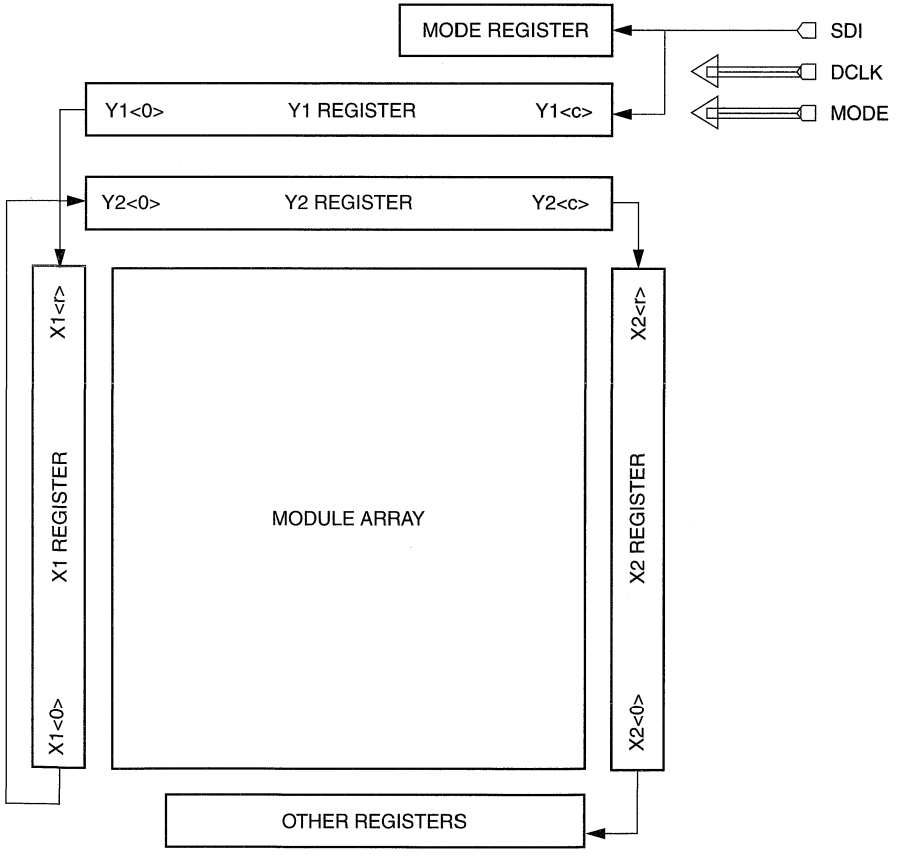


Figure 11 • ACT 2 Shift Register

X2 and Y2 represent the (X,Y) location in the array for the Actionprobe output, PRB. R and C are the row and column size as defined in Table 1. The filler bits, counter pattern, and Mode Register pattern are shown in Table 3. Addressing for rows and columns is active high; that is, unselected rows and columns are "zeros" and the selected row and column is "high." The timing sequence is shown in Figure 12. The recommended frequency is 10 MHz with 10 ns setup and hold times allowing for SDI and DCLK transitions. The selected module output will be present at the PRA or PRB output approximately 20 ns after the stop-bit transition.

Table 3 • Bit Stream Definitions for Actionprobe Diagnostics

Device	Probe_Mode	Filler (n)	Counter_Pattern	Mode_Register_Pattern	# of clocks
A1225A	Probe A only	308	1101011010	000000110001111100000	458
A1225A	Probe B only	308	1101011010	000000101001111100000	458
A1225A	Probe A and B	308	1101011010	000000111001111100000	458
A1240A	Probe A only	361	1111000001	000000110001111100000	545
A1240A	Probe B only	361	1111000001	000000101001111100000	545
A1240A	Probe A and B	361	1111000001	000000111001111100000	545
A1280A	Probe A only	443	0011011111	000000110001111100000	675
A1280A	Probe B only	443	0011011111	000000101001111100000	675
A1280A	Probe A and B	443	0011011111	000000111001111100000	675

For example: Selecting PRA for A1280 results in the following bit stream.
0011011111_000000110001111100000_
(433 zeros)_X2<0>...X2<17>_Y2<81>...Y2<0>_X1<0>...X1<0>...X1<17>_Y1<0>...Y1<81>_0,
where "_" is used for clarity only

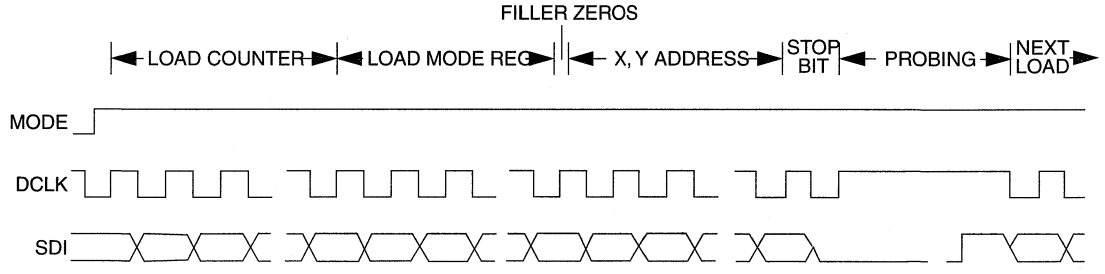


Figure 12 • Timing Waveforms

Absolute Maximum Ratings[1]

Free air temperature range

Symbol	Parameter	Limits	Units
V_{CC}	DC Supply Voltage[2,3,4]	–0.5 to +7.0	V
V_I	Input Voltage	–0.5 to V_{CC} +0.5	V
V_O	Output Voltage	–0.5 to V_{CC} +0.5	V
I_{IO}	I/O Source/Sink Current[5]	±20	mA
T_{STG}	Storage Temperature	–65 to +150	°C

Notes::

1. *Stresses beyond those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. Exposure to absolute maximum rated conditions for extended periods may affect device reliability. Device should not be operated outside the Recommended Operating Conditions.*
2. *$V_{PP} = V_{CC}$, except during device programming.*
3. *$V_{SV} = V_{CC}$, except during device programming.*
4. *V_{KS} = GND, except during device programming.*
5. *Device inputs are normally high impedance and draw extremely low current. However, when input voltage is greater than V_{CC} + 0.5 V or less than GND – 0.5 V, the internal protection diode will be forward biased and can draw excessive current.*

Recommended Operating Conditions

Parameter	Commercial	Industrial	Military	Units
Temperature Range[1]	0 to +70	–40 to +85	–55 to +125	°C
Power Supply Tolerance	±5	±10	±10	%V_{CC}

Note:

1. *Ambient temperature (T_A) is used for commercial and industrial; case temperature (T_C) is used for military.*

Electrical Specifications

Symbol	Parameter	Commercial		Industrial		Military		Units
		Min.	Max.	Min.	Max.	Min.	Max.	
V_{OH}[1]	(I_{OH} = –10 mA)[2]	2.4						V
	(I_{OH} = –6 mA)	3.84						V
	(I_{OH} = –4 mA)			3.7		3.7		V
V_{OL}[1]	(I_{OL} = 10 mA)[2]		0.5					V
	(I_{OL} = 6 mA)		0.33		0.40		0.40	V
V_{IL}		–0.3	0.8	–0.3	0.8	–0.3	0.8	V
V_{IH}		2.0	V_{CC} + 0.3	2.0	V_{CC} + 0.3	2.0	V_{CC} + 0.3	V
Input Transition Time t_R, t_F[2]			500		500		500	ns
C_{IO} I/O Capacitance[2, 3]			10		10		10	pF
Standby Current, I_{CC}[4] (typical = 1 mA)			2		10		20	mA
Leakage Current[5]		–10	10	–10	10	–10	10	µA

Notes:

1. *Only one output tested at a time. V_{CC} = min.*
2. *Not tested, for information only.*
3. *Includes worst-case 176 CPGA package capacitance. V_{OUT} = 0 V, f = 1 MHz.*
4. *All outputs unloaded. All inputs = V_{CC} or GND, typical I_{CC} = 1 mA. I_{CC} limit includes I_{PP} and I_{SV} during normal operation.*
5. *V_{OUT}, V_{IN} = V_{CC} or GND.*

Package Thermal Characteristics

The device junction to case thermal characteristic is θjc, and the junction to ambient air characteristic is θja. The thermal characteristics for θja are shown with two different air flow rates.

Maximum junction temperature is 150°C.

A sample calculation of the absolute maximum power dissipation allowed for a PQFP 160-pin package at commercial temperature is as follows:

$$\frac{\text{Max. junction temp. (°C)} - \text{Max. commercial temp.}}{\theta ja\ (°C/W)} = \frac{150°C - 70°C}{33°C/W} = 2.4\ W$$

Package Type	Pin Count	θja Still Air	θja 300 ft/min	Units
Ceramic Pin Grid Array	100	35	17	°C/W
	132	30	15	°C/W
	176	23	12	°C/W
Ceramic Quad Flatpack	172	25	15	°C/W
Plastic Quad Flatpack[1]	100	48	40	°C/W
	144	40	32	°C/W
	160	38	30	°C/W
Plastic Leaded Chip Carrier[2]	84	37	28	°C/W
Very Thin Quad Flatpack[3]	100	43	35	°C/W
Thin Quad Flatpack[4]	176	32	25	°C/W

Notes:

1. *Maximum Power Dissipation for PQFP packages are 2.1 Watts (100-pin), 2.5 Watts (144-pin), and 2.6 Watts (160-pin).*
2. *Maximum Power Dissipation for PLCC packages is 2.7 Watts.*
3. *Maximum Power Dissipation for VQFP packages is 2.3 Watts.*
4. *Maximum Power Dissipation for TQFP packages is 3.1 Watts.*

Power Dissipation

$$P = [I_{CC}\text{standby} + I_{CC}\text{active}] * V_{CC} + I_{OL} * V_{OL} * N + I_{OH} * (V_{CC} - V_{OH}) * M$$

Where:

I_{CC} standby is the current flowing when no inputs or outputs are changing.

I_{CC} active is the current flowing due to CMOS switching.

I_{OL}, I_{OH} are TTL sink/source currents.

V_{OL}, V_{OH} are TTL level output voltages.

N equals the number of outputs driving TTL loads to V_{OL}.

M equals the number of outputs driving TTL loads to V_{OH}.

An accurate determination of N and M is problematical because their values depend on the family type, design details, and on the system I/O. The power can be divided into two components: static and active.

Static Power Component

Actel FPGAs have small static power components that result in lower power dissipation than PALs or PLDs. By integrating multiple PALs/PLDs into one FPGA, an even greater reduction in board-level power dissipation can be achieved.

The power due to standby current is typically a small component of the overall power. Standby power is calculated below for commercial, worst case conditions.

I_{CC}	V_{CC}	Power
2 mA	5.25 V	10.5 mW

The static power dissipated by TTL loads depends on the number of outputs driving high or low and the DC load current. Again, this value is typically small. For instance, a 32-bit bus sinking 4 mA at 0.33 V will generate 42 mW with all outputs driving low, and 140 mW with all outputs driving high. The actual dissipation will average somewhere between as I/Os switch states with time.

Active Power Component

Power dissipation in CMOS devices is usually dominated by the active (dynamic) power dissipation. This component is frequency dependent, a function of the logic and the external I/O. Active power dissipation results from charging internal chip capacitances of the interconnect, unprogrammed antifuses, module inputs, and module outputs, plus external capacitance due to PC board traces and load device inputs. An additional component of the active power dissipation is the totem-pole current in CMOS transistor pairs. The net

effect can be associated with an equivalent capacitance that can be combined with frequency and voltage to represent active power dissipation.

Equivalent Capacitance

The power dissipated by a CMOS circuit can be expressed by the Equation 1.

$$\text{Power (uW)} = C_{EQ} * V_{CC}^2 * F \quad (1)$$

Where:

C_{EQ} is the equivalent capacitance expressed in pF.

V_{CC} is the power supply in volts.

F is the switching frequency in MHz.

Equivalent capacitance is calculated by measuring ICC active at a specified frequency and voltage for each circuit component of interest. Measurements have been made over a range of frequencies at a fixed value of VCC. Equivalent capacitance is frequency independent so that the results may be used over a wide range of operating conditions. Equivalent capacitance values are shown below.

C_{EQ} Values for Actel FPGAs

Modules (C_{EQM})	5.8
Input Buffers (C_{EQI})	12.9
Output Buffers (C_{EQO})	23.8
Routed Array Clock Buffer Loads (C_{EQCR})	3.9

To calculate the active power dissipated from the complete design, the switching frequency of each part of the logic must be known. Equation 2 shows a piece-wise linear summation over all components.

$$\text{Power} = V_{CC}^2 * [(m * C_{EQM} * f_m)_{modules} + (n * C_{EQI} * f_n)_{inputs} + (p * (C_{EQO} + C_L) * f_p)_{outputs} + 0.5 * (q_1 * C_{EQCR} * f_{q1})_{routed\_Clk1} + (r_1 * f_{q1})_{routed\_Clk1} + 0.5 * (q_2 * C_{EQCR} * f_{q2})_{routed\_Clk2} + (r_2 * f_{q2})_{routed\_Clk2}] \quad (2)$$

Where:

m = Number of logic modules switching at fm

n = Number of input buffers switching at fn

p = Number of output buffers switching at fp

q1 = Number of clock loads on the first routed array clock

q2 = Number of clock loads on the second routed array clock

r_1 = Fixed capacitance due to first routed array clock

r_2 = Fixed capacitance due to second routed array clock

C_{EQM} = Equivalent capacitance of logic modules in pF

C_{EQI} = Equivalent capacitance of input buffers in pF

C_{EQO} = Equivalent capacitance of output buffers in pF

C_{EQCR} = Equivalent capacitance of routed array clock in pF

C_L = Output lead capacitance in pF

f_m = Average logic module switching rate in MHz

f_n = Average input buffer switching rate in MHz

f_p = Average output buffer switching rate in MHz

f_{q1} = Average first routed array clock rate in MHz

f_{q2} = Average second routed array clock rate in MHz

Fixed Capacitance Values for Actel FPGAs (pF)

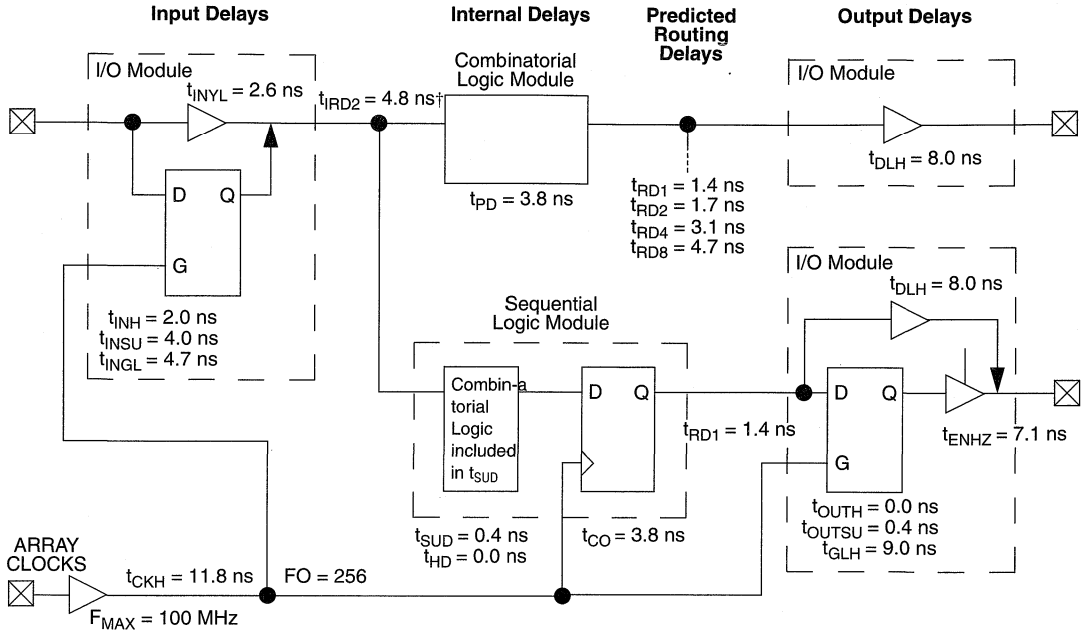
Device Type	r1 routed_Clk1	r2 routed_Clk2
A1225A	106	106.0
A1240A	134	134.2
A1280A	168	167.8

Determining Average Switching Frequency

To determine the switching frequency for a design, you must have a detailed understanding of the data input values to the circuit. The following guidelines are meant to represent worst-case scenarios so that they can be generally used to predict the upper limits of power dissipation. These guidelines are as follows:

Logic Modules (m)	80% of modules
Inputs switching (n)	# inputs/4
Outputs switching (p)	# outputs/4
First routed array clock loads (q_1)	40% of sequential modules
Second routed array clock loads (q_2)	40% of sequential modules
Load capacitance (C_L)	35 pF
Average logic module switching rate (f_m)	F/10
Average input switching rate (f_n)	F/5
Average output switching rate (f_p)	F/10
Average first routed array clock rate (f_{q1})	F
Average second routed array clock rate (f_{q2})	F/2

ACT 2 Timing Model*

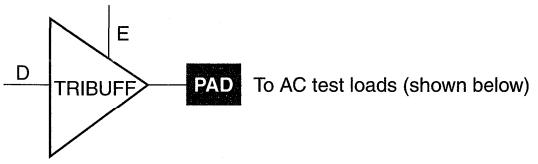


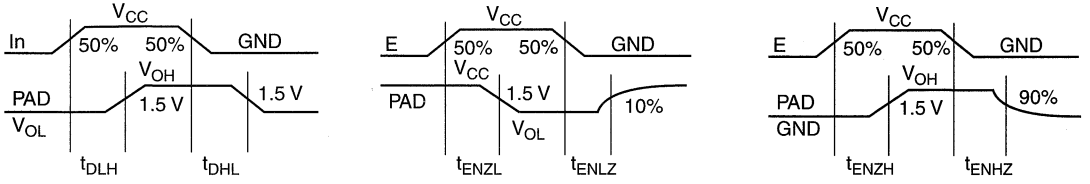
**Values shown for A1240A-2 at worst-case commercial conditions.*

† Input Module Predicted Routing Delay

Parameter Measurement

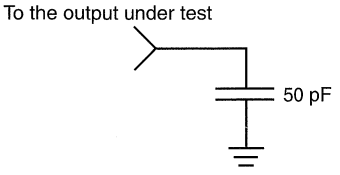
Output Buffer Delays



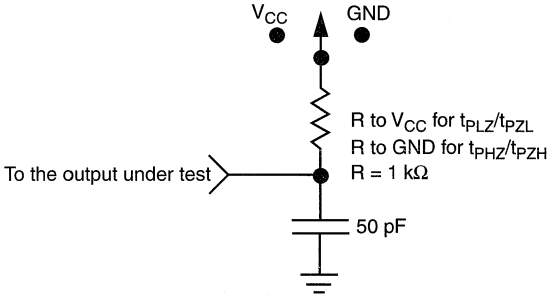


AC Test Loads

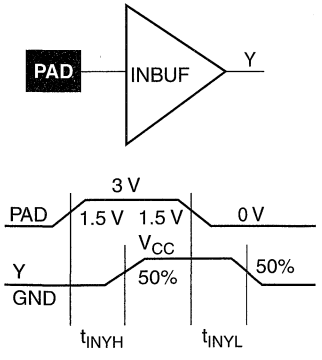
Load 1
(Used to measure propagation delay)



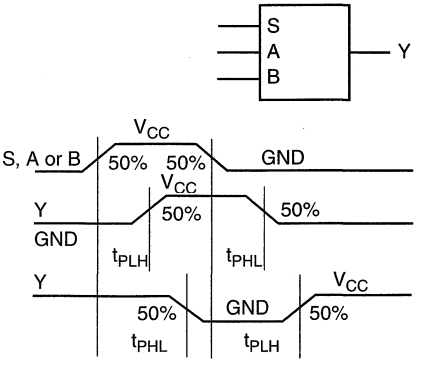
Load 2
(Used to measure rising/falling edges)



Input Buffer Delays

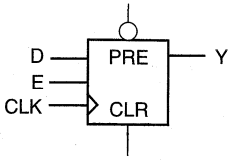


Module Delays

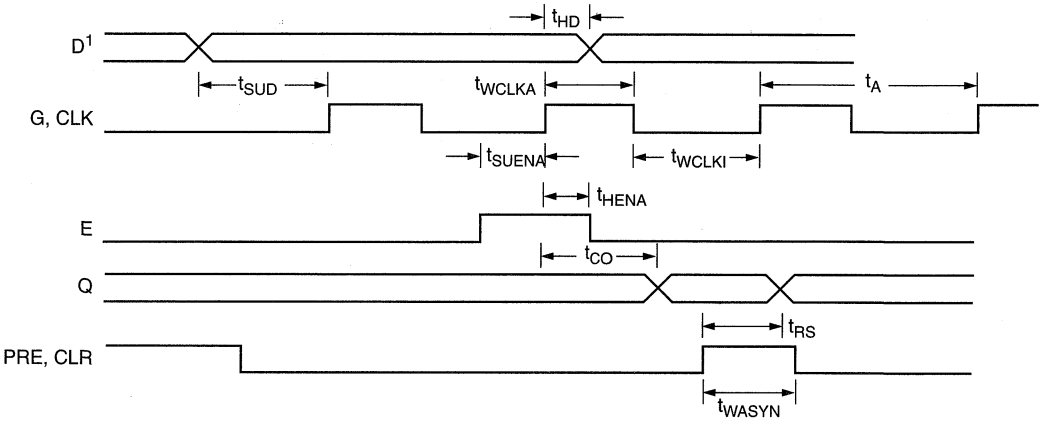


Sequential Module Timing Characteristics

Flip-Flops and Latches



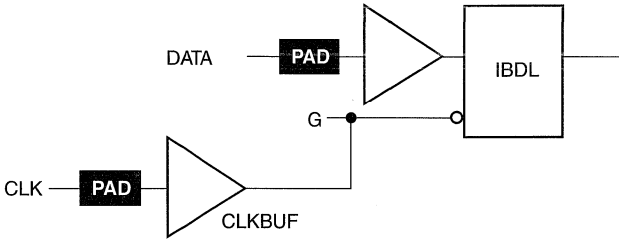
(Positive edge triggered)

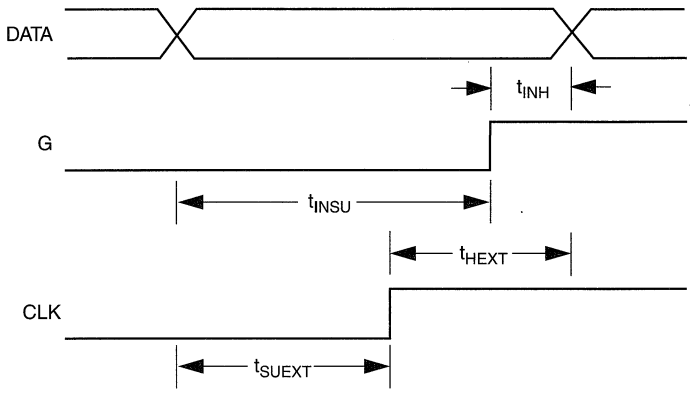


Note: *D represents all data functions involving A, B, and S for multiplexed flip-flops.*

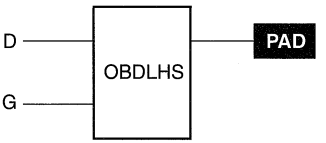
Sequential Timing Characteristics (continued)

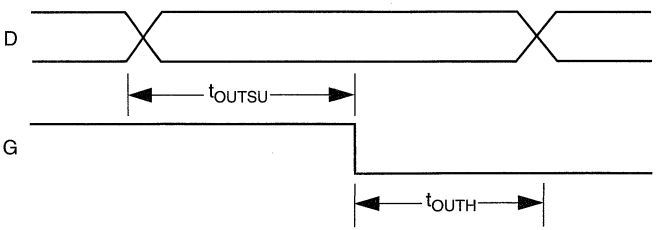
Input Buffer Latches





Output Buffer Latches





Predictable Performance: Tight Delay Distributions

Propagation delay between logic modules depends on the resistive and capacitive loading of the routing tracks, the interconnect elements, and the module inputs being driven. Propagation delay increases as the length of routing tracks, the number of interconnect elements, or the number of inputs increases.

From a design perspective, the propagation delay can be statistically correlated or modeled by the fanout (number of loads) driven by a module. Higher fanout usually requires some paths to have longer routing tracks.

The ACT 2 family delivers a very tight fanout delay distribution. This tight distribution is achieved in two ways: by decreasing the delay of the interconnect elements and by decreasing the number of interconnect elements per path.

Actel's patented PLICE antifuse offers a very low resistive/capacitive interconnect. The ACT 2 family's antifuses, fabricated in 1.0 µm lithography, offer nominal levels of 500 ohms resistance and 7.5 femtofarad (fF) capacitance per antifuse.

The ACT 2 fanout distribution is also tight due to the low number of antifuses required for each interconnect path. The ACT 2 family's proprietary architecture limits the number of antifuses per path to a maximum of four, with 90% of interconnects using two antifuses.

Table 4 • Logic Module + Routing Delay, by Fanout (ns) (Worst-Case Commercial Conditions)

Family	FO=1	FO=2	FO=3	FO=4	FO=8
A1225A-2	4.9	5.5	6.1	6.6	8.2
A1240A-2	5.2	5.5	6.1	6.9	8.5
A1280A-2	5.5	6.3	6.8	7.5	10.5

Timing Characteristics

Timing characteristics for ACT 2 devices fall into three categories: family dependent, device dependent, and design dependent. The input and output buffer characteristics are common to all ACT 2 family members. Internal routing delays are device dependent. Design dependency means actual delays are not determined until after placement and routing of the user's design is complete. Delay values may then be determined by using the ALS Timer utility or performing simulation with post-layout delays.

Critical Nets and Typical Nets

Propagation delays are expressed only for typical nets, which are used for initial design performance evaluation. Critical net delays can then be applied to the most time-critical paths. Critical nets are determined by net property assignment prior to placement and routing. Up to 6% of the nets in a design may be designated as critical, while 90% of the nets in a design are typical.

Long Tracks

Some nets in the design use long tracks. Long tracks are special routing resources that span multiple rows, columns, or modules. Long tracks employ three and sometimes four antifuse connections. This increases capacitance and resistance, resulting in longer net delays for macros connected to long tracks. Typically, up to 6% of nets in a fully utilized device require long tracks. Long tracks contribute approximately 6 ns to 12 ns delay. This additional delay is represented statistically in higher fanout (FO=8) routing delays in the data sheet specifications section.

Timing Derating

A best case timing derating factor of 0.45 is used to reflect best case processing. Note that this factor is relative to the "standard speed" timing parameters, and must be multiplied by the appropriate voltage and temperature derating factors for a given application.

Timing Derating Factor (Temperature and Voltage)

	Industrial		Military	
	Min.	Max.	Min.	Max.
(Commercial Minimum/Maximum Specification) x	0.69	1.11	0.67	1.23

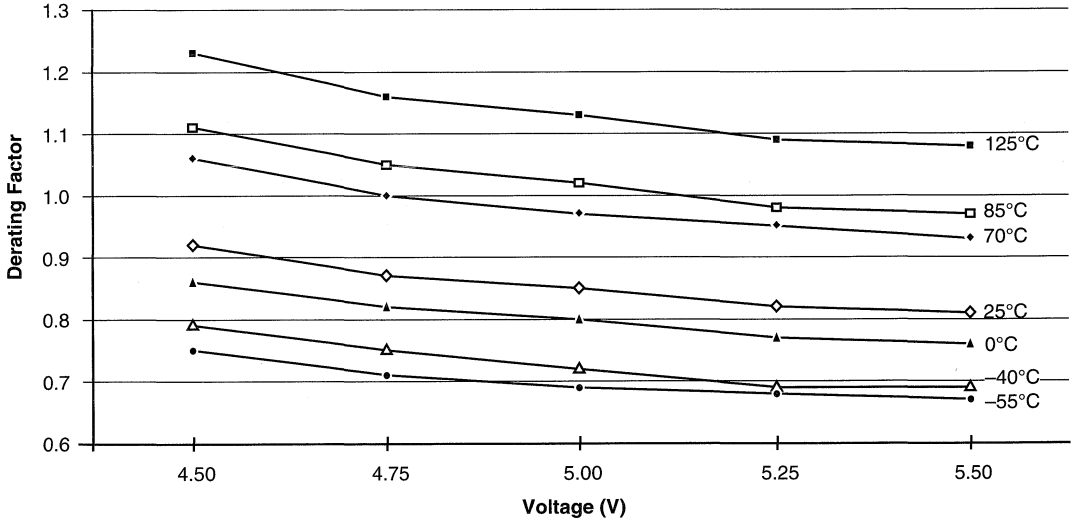
Timing Derating Factor for Designs at Typical Temperature (T_J = 25°C) and Voltage (5.0 V)

(Commercial Maximum Specification) x	0.85

Temperature and Voltage Derating Factors (normalized to Worst-Case Commercial, T_J = 4.75 V, 70°C)

	–55	–40	0	25	70	85	125
4.50	0.75	0.79	0.86	0.92	1.06	1.11	1.23
4.75	0.71	0.75	0.82	0.87	1.00	1.05	1.16
5.00	0.69	0.72	0.80	0.85	0.97	1.02	1.13
5.25	0.68	0.69	0.77	0.82	0.95	0.98	1.09
5.50	0.67	0.69	0.76	0.81	0.93	0.97	1.08

Junction Temperature and Voltage Derating Curves
(normalized to Worst-Case Commercial, T_J = 4.75 V, 70°C)



Note: This derating factor applies to all routing and propagation delays.

A1225A Timing Characteristics

(Worst-Case Commercial Conditions, V_{CC} = 4.75 V, T_J = 70°C)

Logic Module Propagation Delays[1]		'Std' Speed		'–1' Speed		'–2' Speed		
Parameter	**Description**	**Min.**	**Max.**	**Min.**	**Max.**	**Min.**	**Max.**	**Units**
t_{PD1}	Single Module		5.0		4.3		3.8	ns
t_{CO}	Sequential Clk to Q		5.0		4.3		3.8	ns
t_{GO}	Latch G to Q		5.0		4.3		3.8	ns
t_{RS}	Flip-Flop (Latch) Reset to Q		5.0		4.3		3.8	ns
Predicted Routing Delays[2]								
t_{RD1}	FO=1 Routing Delay		1.4		1.2		1.1	ns
t_{RD2}	FO=2 Routing Delay		2.2		1.9		1.7	ns
t_{RD3}	FO=3 Routing Delay		3.0		2.6		2.3	ns
t_{RD4}	FO=4 Routing Delay		3.7		3.1		2.8	ns
t_{RD8}	FO=8 Routing Delay		5.8		4.9		4.4	ns
Sequential Timing Characteristics[3,4]								
t_{SUD}	Flip-Flop (Latch) Data Input Setup	0.5		0.4		0.4		ns
t_{HD}	Flip-Flop (Latch) Data Input Hold	0.0		0.0		0.0		ns
t_{SUENA}	Flip-Flop (Latch) Enable Setup	1.0		0.9		0.8		ns
t_{HENA}	Flip-Flop (Latch) Enable Hold	0.0		0.0		0.0		ns
t_{WCLKA}	Flip-Flop (Latch) Clock Active Pulse Width	6.0		5.0		4.5		ns
t_{WASYN}	Flip-Flop (Latch) Asynchronous Pulse Width	6.0		5.0		4.5		ns
t_A	Flip-Flop Clock Input Period	13.0		11.0		9.4		ns
t_{INH}	Input Buffer Latch Hold	0.0		0.0		0.0		ns
t_{INSU}	Input Buffer Latch Setup	0.5		0.4		0.4		ns
t_{OUTH}	Output Buffer Latch Hold	0.0		0.0		0.0		ns
t_{OUTSU}	Output Buffer Latch Setup	0.5		0.4		0.4		ns
f_{MAX}	Flip-Flop (Latch) Clock Frequency		75.0		90.0		105.0	MHz

Notes:

1. *For dual-module macros, use $t_{PD1} + t_{RD1} + t_{PDn}$, $t_{CO} + t_{RD1} + t_{PDn}$ or $t_{PD1} + t_{RD1} + t_{SUD}$, whichever is appropriate.*
2. *Routing delays are for typical designs across worst-case operating conditions. These parameters should be used for estimating device performance. Post-route timing analysis or simulation is required to determine actual worst-case performance. Post-route timing is based on actual routing delay measurements performed on the device prior to shipment.*
3. *Data applies to macros based on the S-module. Timing parameters for sequential macros constructed from C-modules can be obtained from the ALS Timer utility.*
4. *Setup and hold timing parameters for the Input Buffer Latch are defined with respect to the PAD and the D input. External setup/hold timing parameters must account for delay from an external PAD signal to the G inputs. Delay from an external PAD signal to the G input subtracts (adds) to the internal setup (hold) time.*

A1225A Timing Characteristics (continued)

(Worst-Case Commercial Conditions)

Input Module Propagation Delays			'Std' Speed		'–1' Speed		'–2 Speed		
Parameter	**Description**		**Min.**	**Max.**	**Min.**	**Max.**	**Min.**	**Max.**	**Units**
t_{INYH}	Pad to Y High			3.8		3.3		2.9	ns
t_{INYL}	Pad to Y Low			3.5		3.0		2.6	ns
t_{INGH}	G to Y High			6.6		5.7		5.0	ns
t_{INGL}	G to Y Low			6.3		5.4		4.7	ns
Input Module Predicted Routing Delays[1]									
t_{IRD1}	FO=1 Routing Delay			5.4		4.6		4.1	ns
t_{IRD2}	FO=2 Routing Delay			6.1		5.2		4.6	ns
t_{IRD3}	FO=3 Routing Delay			7.1		6.0		5.3	ns
t_{IRD4}	FO=4 Routing Delay			7.6		6.4		5.7	ns
t_{IRD8}	FO=8 Routing Delay			9.8		8.3		7.4	ns
Global Clock Network									
t_{CKH}	Input Low to High	FO = 32		12.8		11.0		10.2	ns
		FO = 256		15.7		13.0		11.8	
t_{CKL}	Input High to Low	FO = 32		12.8		11.0		10.2	ns
		FO = 256		15.9		13.2		12.0	
t_{PWH}	Minimum Pulse Width High	FO = 32	4.5		4.1		3.4		ns
		FO = 256	5.0		4.5		3.8		
t_{PWL}	Minimum Pulse Width Low	FO = 32	4.5		4.1		3.4		ns
		FO = 256	5.0		4.5		3.8		
t_{CKSW}	Maximum Skew	FO = 32		0.7		0.7		0.7	ns
		FO = 256		3.5		3.5		3.5	
t_{SUEXT}	Input Latch External Setup	FO = 32	0.0		0.0		0.0		ns
		FO = 256	0.0		0.0		0.0		
t_{HEXT}	Input Latch External Hold	FO = 32	7.0		7.0		7.0		ns
		FO = 256	11.2		11.2		11.2		
t_P	Minimum Period	FO = 32	9.1		8.3		7.7		ns
		FO = 256	10.0		8.8		8.1		
f_{MAX}	Maximum Frequency	FO = 32		110.0		120.0		130.0	MHz
		FO = 256		100.0		115.0		125.0	

Note:

1. These parameters should be used for estimating device performance. Optimization techniques may further reduce delays by 0 to 4 ns. Routing delays are for typical designs across worst-case operating conditions. Post-route timing analysis or simulation is required to determine actual worst-case performance. Post-route timing is based on actual routing delay measurements performed on the device prior to shipment.

A1225A Timing Characteristics (continued)

(Worst-Case Commercial Conditions)

Output Module Timing		'Std' Speed		'–1' Speed		'–2 Speed		
Parameter	**Description**	**Min.**	**Max.**	**Min.**	**Max.**	**Min.**	**Max.**	**Units**
TTL Output Module Timing[1, 2]								
t_{DLH}	Data to Pad High		10.6		9.0		8.0	ns
t_{DHL}	Data to Pad Low		13.4		11.4		10.1	ns
t_{ENZH}	Enable Pad Z to High		11.8		10.0		8.9	ns
t_{ENZL}	Enable Pad Z to Low		15.5		13.2		11.6	ns
t_{ENHZ}	Enable Pad High to Z		9.4		8.0		7.1	ns
t_{ENLZ}	Enable Pad Low to Z		11.1		9.5		8.3	ns
t_{GLH}	G to Pad High		11.9		10.2		8.9	ns
t_{GHL}	G to Pad Low		14.9		12.7		11.2	ns
d_{TLH}	Delta Low to High		0.09		0.08		0.07	ns/pF
d_{THL}	Delta High to Low		0.16		0.13		0.12	ns/pF
CMOS Output Module Timing[1, 2]								
t_{DLH}	Data to Pad High		13.5		11.5		10.1	ns
t_{DHL}	Data to Pad Low		11.2		9.6		8.4	ns
t_{ENZH}	Enable Pad Z to High		11.8		10.0		8.9	ns
t_{ENZL}	Enable Pad Z to Low		15.5		13.2		11.6	ns
t_{ENHZ}	Enable Pad High to Z		9.4		8.0		7.1	ns
t_{ENLZ}	Enable Pad Low to Z		11.1		9.5		8.3	ns
t_{GLH}	G to Pad High		11.9		10.2		8.9	ns
t_{GHL}	G to Pad Low		14.9		12.7		11.2	ns
d_{TLH}	Delta Low to High		0.16		0.13		0.12	ns/pF
d_{THL}	Delta High to Low		0.12		0.10		0.09	ns/pF

Notes:

1. *Delays based on 50 pF loading.*
2. *Maximum Recommended Simultaneous Switching Outputs:*

PLCC	*20pF*	*72*
	35pF	*45*
	50pF	*32*
PQFP, CPGA	*20pF*	*80*
	35pF	*45*
	50pF	*32*

A1240A Timing Characteristics

(Worst-Case Commercial Conditions, V_{CC} = 4.75 V, T_J = 70°C)

Logic Module Propagation Delays[1]		'Std' Speed		'–1' Speed		'–2' Speed		
Parameter	**Description**	**Min.**	**Max.**	**Min.**	**Max.**	**Min.**	**Max.**	**Units**
t_{PD1}	Single Module		5.0		4.3		3.8	ns
t_{CO}	Sequential Clk to Q		5.0		4.3		3.8	ns
t_{GO}	Latch G to Q		5.0		4.3		3.8	ns
t_{RS}	Flip-Flop (Latch) Reset to Q		5.0		4.3		3.8	ns
Predicted Routing Delays[2]								
t_{RD1}	FO=1 Routing Delay		1.8		1.5		1.4	ns
t_{RD2}	FO=2 Routing Delay		2.3		2.0		1.7	ns
t_{RD3}	FO=3 Routing Delay		3.0		2.6		2.3	ns
t_{RD4}	FO=4 Routing Delay		4.1		3.5		3.1	ns
t_{RD8}	FO=8 Routing Delay		6.3		5.4		4.7	ns
Sequential Timing Characteristics[3, 4]								
t_{SUD}	Flip-Flop (Latch) Data Input Setup	0.5		0.4		0.4		ns
t_{HD}	Flip-Flop (Latch) Data Input Hold	0.0		0.0		0.0		ns
t_{SUENA}	Flip-Flop (Latch) Enable Setup	1.0		0.9		0.8		ns
t_{HENA}	Flip-Flop (Latch) Enable Hold	0.0		0.0		0.0		ns
t_{WCLKA}	Flip-Flop (Latch) Clock Active Pulse Width	6.5		6.0		4.5		ns
t_{WASYN}	Flip-Flop (Latch) Asynchronous Pulse Width	6.5		6.0		4.5		ns
t_A	Flip-Flop Clock Input Period	15.0		12.0		9.8		ns
t_{INH}	Input Buffer Latch Hold	0.0		0.0		0.0		ns
t_{INSU}	Input Buffer Latch Setup	0.5		0.4		0.4		ns
t_{OUTH}	Output Buffer Latch Hold	0.0		0.0		0.0		ns
t_{OUTSU}	Output Buffer Latch Setup	0.5		0.4		0.4		ns
f_{MAX}	Flip-Flop (Latch) Clock Frequency		66.0		80.0		100.0	MHz

Notes:

1. *For dual-module macros, use $t_{PD1} + t_{RD1} + t_{PDn}$, $t_{CO} + t_{RD1} + t_{PDn}$ or $t_{PD1} + t_{RD1} + t_{SUD}$, whichever is appropriate.*
2. *Routing delays are for typical designs across worst-case operating conditions. These parameters should be used for estimating device performance. Post-route timing analysis or simulation is required to determine actual worst-case performance. Post-route timing is based on actual routing delay measurements performed on the device prior to shipment.*
3. *Data applies to macros based on the S-module. Timing parameters for sequential macros constructed from C-modules can be obtained from the ALS Timer utility.*
4. *Setup and hold timing parameters for the Input Buffer Latch are defined with respect to the PAD and the D input. External setup/hold timing parameters must account for delay from an external PAD signal to the G inputs. Delay from an external PAD signal to the G input subtracts (adds) to the internal setup (hold) time.*

A1240A Timing Characteristics (continued)

(Worst-Case Commercial Conditions)

Input Module Propagation Delays			'Std' Speed		'–1' Speed		'–2' Speed		
Parameter	**Description**		**Min.**	**Max.**	**Min.**	**Max.**	**Min.**	**Max.**	**Units**
t_{INYH}	Pad to Y High			3.8		3.3		2.9	ns
t_{INYL}	Pad to Y Low			3.5		3.0		2.6	ns
t_{INGH}	G to Y High			6.6		5.7		5.0	ns
t_{INGL}	G to Y Low			6.3		5.4		4.7	ns
Input Module Predicted Routing Delays[1]									
t_{IRD1}	FO=1 Routing Delay			5.6		4.8		4.2	ns
t_{IRD2}	FO=2 Routing Delay			6.4		5.4		4.8	ns
t_{IRD3}	FO=3 Routing Delay			7.2		6.1		5.4	ns
t_{IRD4}	FO=4 Routing Delay			7.9		6.7		5.9	ns
t_{IRD8}	FO=8 Routing Delay			10.5		8.9		7.9	ns
Global Clock Network									
t_{CKH}	Input Low to High	FO = 32		12.8		11.0		10.2	ns
		FO = 256		15.7		13.0		11.8	
t_{CKL}	Input High to Low	FO = 32		12.8		11.0		10.2	ns
		FO = 256		15.9		13.2		12.0	
t_{PWH}	Minimum Pulse Width High	FO = 32	5.5		4.5		3.8		ns
		FO = 256	5.8		5.0		4.1		
t_{PWL}	Minimum Pulse Width Low	FO = 32	5.5		4.5		3.8		ns
		FO = 256	5.8		5.0		4.1		
t_{CKSW}	Maximum Skew	FO = 32		0.5		0.5		0.5	ns
		FO = 256		2.5		2.5		2.5	
t_{SUEXT}	Input Latch External Setup	FO = 32	0.0		0.0		0.0		ns
		FO = 256	0.0		0.0		0.0		
t_{HEXT}	Input Latch External Hold	FO = 32	7.0		7.0		7.0		ns
		FO = 256	11.2		11.2		11.2		
t_P	Minimum Period	FO = 32	11.1		9.1		8.1		ns
		FO = 256	11.7		10.0		8.8		
f_{MAX}	Maximum Frequency	FO = 32		90.0		110.0		125.0	MHz
		FO = 256		85.0		100.0		115.0	

Note:

1. *These parameters should be used for estimating device performance. Optimization techniques may further reduce delays by 0 to 4 ns. Routing delays are for typical designs across worst-case operating conditions. Post-route timing analysis or simulation is required to determine actual worst-case performance. Post-route timing is based on actual routing delay measurements performed on the device prior to shipment.*

A1240A Timing Characteristics (continued)

(Worst-Case Commercial Conditions)

Output Module Timing		'Std' Speed		'–1' Speed		'–2' Speed		
Parameter	**Description**	**Min.**	**Max.**	**Min.**	**Max.**	**Min.**	**Max.**	**Units**
TTL Output Module Timing[1, 2]								
t_{DLH}	Data to Pad High		10.6		9.0		8.0	ns
t_{DHL}	Data to Pad Low		13.4		11.4		10.1	ns
t_{ENZH}	Enable Pad Z to High		11.8		10.0		8.9	ns
t_{ENZL}	Enable Pad Z to Low		15.5		13.2		11.7	ns
t_{ENHZ}	Enable Pad High to Z		9.4		8.0		7.1	ns
t_{ENLZ}	Enable Pad Low to Z		11.1		9.5		8.4	ns
t_{GLH}	G to Pad High		11.9		10.2		9.0	ns
t_{GHL}	G to Pad Low		14.9		12.7		11.2	ns
d_{TLH}	Delta Low to High		0.09		0.08		0.07	ns/pF
d_{THL}	Delta High to Low		0.16		0.13		0.12	ns/pF
CMOS Output Module Timing[1, 2]								
t_{DLH}	Data to Pad High		13.5		11.5		10.2	ns
t_{DHL}	Data to Pad Low		11.2		9.6		8.4	ns
t_{ENZH}	Enable Pad Z to High		11.8		10.0		8.9	ns
t_{ENZL}	Enable Pad Z to Low		15.5		13.2		11.7	ns
t_{ENHZ}	Enable Pad High to Z		9.4		8.0		7.1	ns
t_{ENLZ}	Enable Pad Low to Z		11.1		9.5		8.4	ns
t_{GLH}	G to Pad High		11.9		10.2		9.0	ns
t_{GHL}	G to Pad Low		14.9		12.7		11.2	ns
d_{TLH}	Delta Low to High		0.16		0.13		0.12	ns/pF
d_{THL}	Delta High to Low		0.12		0.10		0.09	ns/pF

Notes:

1. *Delays based on 50 pF loading.*
2. *Maximum Recommended Simultaneous Switching Outputs:*

PLCC	*20pF*	*72*
	35pF	*45*
	50pF	*32*
PQFP, CPGA	*20pF*	*104*
	35pF	*68*
	50pF	*48*

A1280A Timing Characteristics

(Worst-Case Commercial Conditions, V_{CC} = 4.75 V, T_J = 70°C)

Logic Module Propagation Delays[1]		'Std' Speed		'–1' Speed		'–2' Speed		
Parameter	**Description**	**Min.**	**Max.**	**Min.**	**Max.**	**Min.**	**Max.**	**Units**
t_{PD1}	Single Module		5.0		4.3		3.8	ns
t_{CO}	Sequential Clk to Q		5.0		4.3		3.8	ns
t_{GO}	Latch G to Q		5.0		4.3		3.8	ns
t_{RS}	Flip-Flop (Latch) Reset to Q		5.0		4.3		3.8	ns
Predicted Routing Delays[2]								
t_{RD1}	FO=1 Routing Delay		2.3		2.0		1.7	ns
t_{RD2}	FO=2 Routing Delay		3.3		2.8		2.5	ns
t_{RD3}	FO=3 Routing Delay		4.0		3.4		3.0	ns
t_{RD4}	FO=4 Routing Delay		4.9		4.2		3.7	ns
t_{RD8}	FO=8 Routing Delay		8.8		7.5		6.7	ns
Sequential Timing Characteristics[3,4]								
t_{SUD}	Flip-Flop (Latch) Data Input Setup	0.5		0.4		0.4		ns
t_{HD}	Flip-Flop (Latch) Data Input Hold	0.0		0.0		0.0		ns
t_{SUENA}	Flip-Flop (Latch) Enable Setup	1.0		0.9		0.8		ns
t_{HENA}	Flip-Flop (Latch) Enable Hold	0.0		0.0		0.0		ns
t_{WCLKA}	Flip-Flop (Latch) Clock Active Pulse Width	7.0		6.0		5.5		ns
t_{WASYN}	Flip-Flop (Latch) Asynchronous Pulse Width	7.0		6.0		5.5		ns
t_A	Flip-Flop Clock Input Period	18.0		13.3		11.7		ns
t_{INH}	Input Buffer Latch Hold	0.0		0.0		0.0		ns
t_{INSU}	Input Buffer Latch Setup	0.5		0.4		0.4		ns
t_{OUTH}	Output Buffer Latch Hold	0.0		0.0		0.0		ns
t_{OUTSU}	Output Buffer Latch Setup	0.5		0.4		0.4		ns
f_{MAX}	Flip-Flop (Latch) Clock Frequency		50.0		75.0		85.0	MHz

Notes:

1. *For dual-module macros, use $t_{PD1} + t_{RD1} + t_{PDn}$, $t_{CO} + t_{RD1} + t_{PDn}$, or $t_{PD1} + t_{RD1} + t_{SUD}$, whichever is appropriate.*
2. *Routing delays are for typical designs across worst-case operating conditions. These parameters should be used for estimating device performance. Post-route timing analysis or simulation is required to determine actual worst-case performance. Post-route timing is based on actual routing delay measurements performed on the device prior to shipment.*
3. *Data applies to macros based on the S-module. Timing parameters for sequential macros constructed from C-modules can be obtained from the ALS Timer utility.*
4. *Setup and hold timing parameters for the Input Buffer Latch are defined with respect to the PAD and the D input. External setup/hold timing parameters must account for delay from an external PAD signal to the G inputs. Delay from an external PAD signal to the G input subtracts (adds) to the internal setup (hold) time.*

A1280A Timing Characteristics (continued)

(Worst-Case Commercial Conditions)

Input Module Propagation Delays			'Std' Speed		'–1' Speed		'–2' Speed		
Parameter	**Description**		**Min.**	**Max.**	**Min.**	**Max.**	**Min.**	**Max.**	**Units**
t_{INYH}	Pad to Y High			3.8		3.3		2.9	ns
t_{INYL}	Pad to Y Low			3.5		3.0		2.7	ns
t_{INGH}	G to Y High			6.6		5.7		5.0	ns
t_{INGL}	G to Y Low			6.3		5.4		4.8	ns
Input Module Predicted Routing Delays[1]									
t_{IRD1}	FO=1 Routing Delay			6.0		5.1		4.6	ns
t_{IRD2}	FO=2 Routing Delay			6.9		5.9		5.2	ns
t_{IRD3}	FO=3 Routing Delay			7.4		6.3		5.6	ns
t_{IRD4}	FO=4 Routing Delay			8.6		7.3		6.5	ns
t_{IRD8}	FO=8 Routing Delay			12.4		10.5		9.4	ns
Global Clock Network									
t_{CKH}	Input Low to High	FO = 32		12.8		11.0		10.2	ns
		FO = 384		17.2		14.6		13.1	
t_{CKL}	Input High to Low	FO = 32		12.8		11.0		10.2	ns
		FO = 384		17.5		14.9		13.3	
t_{PWH}	Minimum Pulse Width High	FO = 32	6.6		5.5		5.0		ns
		FO = 384	7.6		6.4		5.8		
t_{PWL}	Minimum Pulse Width Low	FO = 32	6.6		5.5		5.0		ns
		FO = 384	7.6		6.4		5.8		
t_{CKSW}	Maximum Skew	FO = 32		0.5		0.5		0.5	ns
		FO = 384		2.5		2.5		2.5	
t_{SUEXT}	Input Latch External Setup	FO = 32	0.0		0.0		0.0		ns
		FO = 384	0.0		0.0		0.0		
t_{HEXT}	Input Latch External Hold	FO = 32	7.0		7.0		7.0		ns
		FO = 384	11.2		11.2		11.2		
t_P	Minimum Period	FO = 32	13.3		11.2		9.6		ns
		FO = 384	15.3		12.6		10.6		
f_{MAX}	Maximum Frequency	FO = 32		75.0		90.0		105.0	MHz
		FO = 384		65.0		80.0		95.0	

Note:

1. *These parameters should be used for estimating device performance. Optimization techniques may further reduce delays by 0 to 4 ns. Routing delays are for typical designs across worst-case operating conditions. Post-route timing analysis or simulation is required to determine actual worst-case performance. Post-route timing is based on actual routing delay measurements performed on the device prior to shipment.*

A1280A Timing Characteristics (continued)

(Worst-Case Commercial Conditions)

Output Module Timing		'Std' Speed		'–1' Speed		'–2' Speed		
Parameter	**Description**	**Min.**	**Max.**	**Min.**	**Max.**	**Min.**	**Max.**	**Units**
TTL Output Module Timing[1, 2]								
t_{DLH}	Data to Pad High		10.6		9.0		8.1	ns
t_{DHL}	Data to Pad Low		13.4		11.4		10.2	ns
t_{ENZH}	Enable Pad Z to High		11.8		10.0		9.0	ns
t_{ENZL}	Enable Pad Z to Low		15.5		13.2		11.8	ns
t_{ENHZ}	Enable Pad High to Z		9.4		8.0		7.1	ns
t_{ENLZ}	Enable Pad Low to Z		11.1		9.5		8.4	ns
t_{GLH}	G to Pad High		11.9		10.2		9.0	ns
t_{GHL}	G to Pad Low		14.9		12.7		11.3	ns
d_{TLH}	Delta Low to High		0.09		0.08		0.07	ns/pF
d_{THL}	Delta High to Low		0.16		0.13		0.12	ns/pF
CMOS Output Module Timing[1, 2]								
t_{DLH}	Data to Pad High		13.5		11.5		10.3	ns
t_{DHL}	Data to Pad Low		11.2		9.6		8.5	ns
t_{ENZH}	Enable Pad Z to High		11.8		10.0		9.0	ns
t_{ENZL}	Enable Pad Z to Low		15.5		13.2		11.8	ns
t_{ENHZ}	Enable Pad High to Z		9.4		8.0		7.1	ns
t_{ENLZ}	Enable Pad Low to Z		11.1		9.5		8.4	ns
t_{GLH}	G to Pad High		11.9		10.2		9.0	ns
t_{GHL}	G to Pad Low		14.9		12.7		11.3	ns
d_{TLH}	Delta Low to High		0.16		0.13		0.12	ns/pF
d_{THL}	Delta High to Low		0.12		0.10		0.09	ns/pF

Notes::

1. *Delays based on 50 pF loading.*
2. *Maximum Recommended Simultaneous Switching Outputs:*

PQFP, CPGA, CQFP	*20pF*	*160*
	35pF	*90*
	50pF	*64*

Macro Library

Hard Macros—Combinatorial

Function	Macro	Description	Modules S	Modules C
ACT 2 Combinatorial Logic Module	CM8	Combinational Module (Full ACT 2 Logic Module)		1
ACT 2 Sequential Logic Module	DFM7A	4-input D-Type Flip-Flop with Multiplexed Data, active low Clear, and active high clock	1	
	DFM7B	4-input D-Type Flip-Flop with Multiplexed Data, active low Clear, and clock	1	
Adder	FA1A	1-bit adder, carry in and carry out active low, A-input active low		2
	FA1B	1-bit adder, carry in and carry out active low		2
	FA2A	2-bit adder, carry in and carry out active low, A0 and A1 inputs active low		2
	HA1	Half-Adder		2
	HA1A	Half-Adder with active low A-input		2
	HA1B	Half-Adder with active low carry out and sum		2
	HA1C	Half-Adder with active low carry out		2
AND	AND2	2-input AND		1
	AND2A	2-input AND with active low A-input		1
	AND2B	2-input AND with active low inputs		1
	AND3	3-input AND		1
	AND3A	3-input AND with active low A-input		1
	AND3B	3-input AND with active low A- and B-inputs		1
	AND3C	3-input AND with active low inputs		1
	AND4	4-input AND		1
	AND4A	4-input AND with active low A-input		1
	AND4B	4-input AND with active low A- and B-inputs		1
	AND4C	4-input AND with active low A-, B-, and C-inputs		1
	AND4D	4-input AND with active low inputs		2
	AND5B	5-input AND with active low A- and B-inputs		1
AND-OR	AO1	3-input AND-OR		1
	AO10	5-input AND-OR-AND		1
	AO11	3-input AND-OR		1
	AO1A	3-input AND-OR with active low A-input		1
	AO1B	3-input AND-OR with active low C-input		1
	AO1C	3-input AND-OR with active low A- and C-inputs		1
	AO1D	3-input AND-OR with active low A- and B-inputs		1
	AO1E	3-input AND-OR with active low inputs		1
	AO2	4-input AND-OR		1
	AO2A	4-input AND-OR with active low A-input		1
	AO2B	4-input AND-OR with active low A- and B-inputs		1
	AO2C	4-input AND-OR with active low A- and C-inputs		1
	AO2D	4-input AND-OR with active low A-, B-, and C-inputs		1
	AO2E	4-input AND-OR with active low inputs		1
	AO3	4-input AND-OR		1
	AO3A	4-input AND-OR		1
	AO3B	4-input AND-OR		1
	AO3C	4-input AND-OR		1
	AO4A	4-input AND-OR		1
	AO5A	4-input AND-OR		1
	AO6	2-wide 4-input AND-OR		1
	AO6A	2-wide 4-input AND-OR with active low D-input		1

Hard Macros—Combinatorial (Continued)

Function	Macro	Description	Modules S	Modules C
AND-OR	AO7	5-input AND-OR		1
	AO8	5-input AND-OR with active low C- and D-inputs		1
	AO9	5-input AND-OR		1
	AOI1	3-input AND-OR-INVERT		1
	AOI1A	3-input AND-OR-INVERT with active low A-input		1
	AOI1B	3-input AND-OR-INVERT with active low C-input		1
	AOI1C	3-input AND-OR-INVERT with active low A- and B-inputs		1
	AOI1D	3-input AND-OR-INVERT with active low inputs		1
	AOI2A	4-input AND-OR-INVERT with active low A-input		1
	AOI2B	4-input AND-OR-INVERT with active low A- and C-inputs		1
	AOI3A	4-input AND-OR-INVERT with active low inputs		1
	AOI4	2-wide 4-input AND-OR-INVERT		2
	AOI4A	2-wide 4-input AND-OR-INVERT with active low C-input		1
AND-XOR	AX1	3-input AND-XOR with active low A-input		1
	AX1A	3-input AND-XOR-INVERT with active low A-input		2
	AX1B	3-input AND-XOR with active low A- and B-inputs		1
	AX1C	3-input AND-XOR		1
Buffer	BUF	Buffer with active high input and output		1
	BUFA	Buffer with active low input and output		1
Clock Net	CLKINT	Clock Net Interface	0	0
	GAND2	2-input AND Clock Net		1
	GMX4	4-to-1 Multiplexor Clock Net		1
	GNAND2	2-input NAND Clock Net		1
	GNOR2	2-input NOR Clock Net		1
	GOR2	2-input OR Clock Net		1
	GXOR2	2-input Exclusive OR Clock Net		1
Inverter	INV	Inverter with active low output		1
	INVA	Inverter with active low input		1
Majority	MAJ3	3-input complex AND-OR		1
MUX	MX2	2-to-1 Multiplexor		1
	MX2A	2-to-1 Multiplexor with active low A-input		1
	MX2B	2-to-1 Multiplexor with active low B-input		1
MUX	MX2C	2-to-1 Multiplexor with active low output		1
	MX4	4-to-1 Multiplexor		1
	MXC1	Boolean		2
	MXT	Boolean		2
NAND	NAND2	2-input NAND		1
	NAND2A	2-input NAND with active low A-input		1
	NAND2B	2-input NAND with active low inputs		1
	NAND3	3-input NAND		1
	NAND3A	3-input NAND with active low A-input		1
	NAND3B	3-input NAND with active low A- and B-inputs		1
	NAND3C	3-input NAND with active low inputs		1
	NAND4	4-input NAND		2
	NAND4A	4-input NAND with active low A-input		1
	NAND4B	4-input NAND with active low A- and B-inputs		1
	NAND4C	4-input NAND with active low A-, B-, and C-inputs		1
	NAND4D	4-input NAND with active low inputs		1
	NAND5C	5-input NAND with active low A-, B-, and C-inputs		1

Hard Macros—Combinatorial (Continued)

Function	Macro	Description	Modules S	Modules C
NOR	NOR2	2-input NOR		1
	NOR2A	2-input NOR with active low A-input		1
	NOR2B	2-input NOR with active low inputs		1
	NOR3	3-input NOR		1
	NOR3A	3-input NOR with active low A-input		1
	NOR3B	3-input NOR with active low A- and B-inputs		1
	NOR3C	3-input NOR with active low inputs		1
	NOR4	4-input NOR		2
	NOR4A	4-input NOR with active low A-input		1
	NOR4B	4-input NOR with active low A- and B-inputs		1
	NOR4C	4-input NOR with active low A-, B-, and C-inputs		1
	NOR4D	4-input NOR with active low inputs		1
	NOR5C	5-input NOR with active low A-, B-, and C-inputs		1
OR	OR2	2-input OR		1
	OR2A	2-input OR with active low A-input		1
	OR2B	2-input OR with active low inputs		1
	OR3	3-input OR		1
	OR3A	3-input OR with active low A-input		1
	OR3B	3-input OR with active low A- and B-inputs		1
	OR3C	3-input OR with active low inputs		1
	OR4	4-input OR		1
	OR4A	4-input OR with active low A-input		1
	OR4B	4-input OR with active low A- and B-input		1
	OR4C	4-input OR with active low A-, B-, and C-inputs		1
	OR4D	4-input OR with active low inputs		2
	OR5B	5-input OR with active low A- and B-inputs		1
OR-AND	OA1	3-input OR-AND		1
	OA1A	3-input OR-AND with active low A-input		1
	OA1B	3-input OR-AND with active low C-input		1
	OA1C	3-input OR-AND with active low A- and C-inputs		1
	OA2	2-wide 4-input OR-AND		1
	OA2A	2 wide 4-input OR-AND with active low A-input		1
	OA3	4-input OR-AND		1
	OA3A	4-input OR-AND with active low C-input		1
	OA3B	4-input OR-AND with active low A- and C-inputs		1
	OA4	4-input OR-AND		1
	OA4A	4-input OR-AND with active low C-input		1
	OA5	4-input complex OR-AND		1
	OAI1	3-input OR-AND-INVERT		1
	OAI2A	4-input OR-AND-INVERT with active low D-input		1
	OAI3	4-input OR-AND-INVERT		1
	OAI3A	4-input OR-AND-INVERT with active low C- and D-inputs		1
XNOR	XNOR	2-input XNOR		1
XNOR-AND	XA1A	3-input XNOR-AND		1
XNOR-OR	XO1A	3-input XNOR-OR		1
XOR	XOR	2-input XOR		1
XOR-AND	XA1	3-input XOR-AND		1
XOR-OR	XO1	3-input XOR-OR		1

Hard Macros—Sequential

Function	Macro	Description	Modules S	Modules C
D-Type	DF1	D-Type Flip-Flop	1	
	DF1A	D-Type Flip-Flop with active low output	1	
	DF1B	D-Type Flip-Flop with active low clock	1	
	DF1C	D-Type Flip-Flop with active low clock and output	1	
	DFC1	D-Type Flip-Flop with active high Clear	1	1
	DFC1A	D-Type Flip-Flop with active high Clear and active low clock	1	1
	DFC1B	D-Type Flip-Flop with active low Clear	1	
	DFC1D	D-Type Flip-Flop with active low Clear and clock	1	
	DFE	D-Type Flip-Flop with active high Enable	1	
	DFE1B	D-Type Flip-Flop with active low Enable	1	
	DFE1C	D-Type Flip-Flop with active low Enable and clock	1	
	DFE3A	D-Type Flip-Flop with Enable and active low Clear	1	
	DFE3B	D-Type Flip-Flop with Enable and active low Clear and clock	1	
	DFE3C	D-Type Flip-Flop with active low Enable and Clear	1	
	DFE3D	D-Type Flip-Flop with active low Enable, Clear, and clock	1	
	DFEA	D-Type Flip-Flop with Enable and active low clock	1	
	DFM	2-input D-Type Flip-Flop with Multiplexed Data	1	
	DFM1B	2-input D-Type Flip-Flop with Multiplexed Data and active low output	1	
	DFM1C	2-input D-Type Flip-Flop with Multiplexed Data and active low clock and output	1	
	DFM3	2-input D-Type Flip-Flop with Multiplexed Data and Clear	1	1
	DFM3B	2-input D-Type Flip-Flop with Multiplexed Data and active low Clear and clock	1	
	DFM3E	2-input D-Type Flip-Flop with Multiplexed Data, Clear, and active low clock	1	1
	DFM4C	2-input D-Type Flip-Flop with Multiplexed Data and active low Preset and output	1	
	DFM4D	2-input D-Type Flip-Flop with Multiplexed Data and active low Preset, clock, and output	1	
	DFM6A	4-input D-Type Flip-Flop with Multiplexed Data, active low Clear, and active high clock	1	
	DFM6B	4-input D-Type Flip-Flop with Multiplexed Data, active low Clear, and clock	1	
	DFMA	2-input D-Type Flip-Flop with Multiplexed Data and active low clock	1	
	DFMB	2-input D-Type Flip-Flop with Multiplexed Data and active low Clear	1	
	DFME1A	2-input D-Type Flip-Flop with Multiplexed Data and active low Enable	1	
	DFP1	D-Type Flip-Flop with active high Preset		2
	DFP1A	D-Type Flip-Flop with active high Preset and active low clock		2
	DFP1B	D-Type Flip-Flop with active low Preset		2
	DFP1C	D-Type Flip-Flop with active high Preset and active low output	1	1
	DFP1D	D-Type Flip-Flop with active low Preset and clock		2
	DFP1E	D-Type Flip-Flop with active low Preset and output	1	
	DFP1F	D-Type Flip-Flop with active high Preset and active low clock and output	1	1
	DFP1G	D-Type Flip-Flop with active low Preset, clock, and output	1	
	DFPC	D-Type Flip-Flop with active high Preset, active low Clear, and active high clock		2
	DFPCA	D-Type Flip-Flop with active high Preset and active low Clear and clock		2
J-K Type	JKF	JK Flip-Flop with active low K-input	1	
	JKF1B	JK Flip-Flop with active low clock and K-input	1	
	JKF2A	JK Flip-Flop with active low Clear and K-input	1	

Hard Macros—Sequential (Continued)

Function	Macro	Description	Modules S	Modules C
J-K Type	JKF2B	JK Flip-Flop with active low Clear, clock, and K-input	1	
	JKF2C	JK Flip-Flop with active high Clear and active low K-input	1	1
	JKF2D	JK Flip-Flop with active high Clear and active low clock and K-input	1	1
T-Type	TF1A	T-Type Flip-Flop with active low Clear	1	
	TF1B	T-Type Flip-Flop with active low Clear and clock	1	
Latch	DL1	Data Latch	1	
	DL1A	Data Latch with active low output	1	
	DL1B	Data Latch with active low clock	1	
	DL1C	Data Latch with active low clock and output	1	
	DLC	Data Latch with active low Clear	1	
	DLC1	Data Latch with active high Clear		1
	DLC1A	Data Latch with active high Clear and active low clock		1
	DLC1F	Data Latch with active high Clear and active low output		1
	DLC1G	Data Latch with active high Clear and active low clock and output		1
	DLCA	Data Latch with active low Clock and Clear	1	
	DLE	Data Latch with active high Enable	1	
	DLE1D	Data Latch with active high Enable and clock and active low input and output	1	
	DLE2B	Data Latch with active low Enable, Clear, and clock	1	
	DLE2C	Data Latch with active low Enable and clock and active high Clear		1
	DLE3B	Data Latch with active low Enable and clock and active low Preset		1
	DLE3C	Data Latch with active low Enable, Preset, and clock		1
	DLEA	Data Latch with active low Enable and active high clock	1	
	DLEB	Data Latch with active high Enable and active high clock	1	
	DLEC	Data Latch with active low Enable and clock	1	
	DLM	2-input Data Latch with Multiplexed Data	1	
	DLM3	4-input Data Latch with Multiplexed Data	1	
	DLM3A	4-input Data Latch with Multiplexed Data and active low clock	1	
	DLM4	Data Latch with Multiplexed Data	1	
	DLM4A	Data Latch with Multiplexed Data	1	
	DLMA	2-input Data Latch with Multiplexed Data and active low clock	1	
	DLME1A	2-input Data Latch with Multiplexed Data and Enable and active low clock	1	
	DLP1	Data Latch with active high Preset and clock		1
	DLP1A	Data Latch with active high Preset and active low clock		1
	DLP1B	Data Latch with active low Preset and active high clock		1
	DLP1C	Data Latch with active low Preset and clock		1
	DLP1D	Data Latch with active low Preset and output and active high clock	1	
	DLP1E	Data Latch with active low Preset, clock, and output	1	

Input/Output Macros

Function	Macro	Description	I/O Modules
Buffer	IBDL	Input Buffer with Latch Clock	1
	INBUF	Input Buffer	1
	OBHS	Output Buffer, High Slew	1
	OUTBUF	Output Buffer, High Slew	1
Bidirectional	BBHS	Bidirectional Buffer, High Slew	1
	BBDLHS	Bidirectional with Input Latch and Output Latch	1
	BIBUF	Bidirectional Buffer, High Slew (with hidden buffer at Y pin)	1
	CLKBIBUF	Bidirectional with Input Dedicated to Clock Network	1
Input	CLKBUF	Input for Dedicated Routed Clock Network	1
Output	DBDLKS	Output Buffer with Latch	1
	OBHS	Output Buffer	1
	TBHS	Tristate output, High Slew	1
	TRIBUFF	Tristate output, High Slew	1

Soft Macros

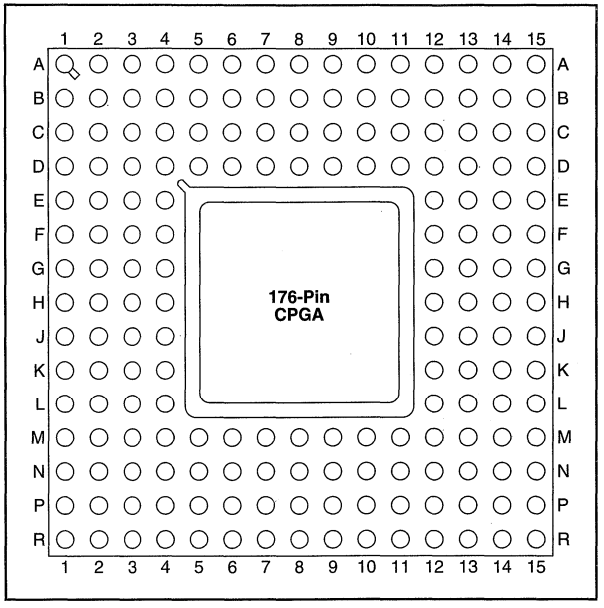
Function	Macro	Description	Maximum Logic Levels	Modules S	Modules C
Adder	FADD10	10-bit adder	3		56
	FADD12	12-bit adder	4		9
	FADD16	16-bit adder	5		97
	FADD8	8-bit adder	4		44
	FADD9	9-bit adder with active low carry out	3		49
	VAD16C	Very fast 16-bit adder, no Carry in	3		91
	VADC16C	Very fast 16-bit adder with Carry in	3		97
Comparator	ICMP4	4-bit Identity Comparator	2		5
	ICMP8	8-bit Identity Comparator	3		9
	MCMPC2	2-bit Magnitude Comparator with Enable	3		9
	MCMPC4	4-bit Magnitude Comparator with Enable	4		18
	MCMPC8	8-bit Magnitude Comparator with Enable	6		36
Counter	CNT4A	4-bit binary counter with load and clear	4	4	8
	CNT4B	4-bit binary counter with load, clear, carry-in, carry-out	4	4	7
	FCTD16C	Fast 16-bit Down Counter, parallel loadable	2	19	33
	FCTD8A	Fast 8-bit Down Counter, parallel loadable	1	10	18
	FCTD8B	Fast 8-bit Down Counter, parallel loadable	1	9	13
	FCTU16C	Fast 16-bit Up Counter, parallel loadable	2	19	31
	FCTU8A	Fast 8-bit Up Counter, parallel loadable	1	10	17
	FCTU8B	Fast 8-bit Up Counter, parallel loadable	1	9	12
	UDCNT4A	4-bit up/down counter with load, carry-in, and carry-out	5	4	13
	VCTD16C	Very fast 16-bit down counter, delay after load, registered control inputs	1	34	41
	VCTD2CP	2-bit down counter, prescaler, delay after load, used to build VCTD counters	1	5	2
	VCTD2CU	2-bit down counter, upper bits, delay after load, used to build VCTD counters	1	2	3
	VCTD4CL	4-bit down counter, lower bits, delay after load, used to build VCTD counters	1	4	7
	VCTD4CM	4-bit down counter, middle bits, delay after load, used to build VCTD counters	1	4	8
Decoder	DEC2X4	2-to-4 decoder	1		4
	DEC2X4A	2-to-4 decoder with active low outputs	1		4
	DEC3X8	3-to-8 decoder	1		8
	DEC3X8A	3-to-8 decoder with active low outputs	1		8
	DEC4X16A	4-to-16 decoder with active low outputs	2		20
	DECE2X4	2-to-4 decoder with enable	1		4
	DECE2X4A	2-to-4 decoder with enable and active low outputs	1		4
	DECE3X8	3-to-8 decoder with enable	2		11
	DECE3X8A	3-to-8 decoder with enable and active low outputs	2		11
Latch	DLC8A	Octal latch with clear active low 8-bit Data Latch with active low Clear	1	8	
	DLE8	Octal latch with enable 8-bit Data Latch with active high Enable	1	8	
	DLM8	Octal latch with multiplexed data 8-bit Data Latch with Multiplexed Data	1	8	
MUX	MX16	16-to-1 Multiplexor	2		5
	MX8	8-to-1 Multiplexor with active high output	2		3
	MX8A	8-to-1 Multiplexor with active low output	2		3
Multiplier	SMULT8	8-bit by 8-bit Multiplier			242
Shift Register	SREG4A	4-bit shift register with clear active low	1	4	
	SREG8A	8-bit shift register with clear active low	1	8	

Soft Macros—TTL Equivalent

Function	Macro	Description	Maximum Logic Levels	Modules S	Modules C
	TA00	2-input NAND	1		1
	TA02	2-input NOR	1		1
	TA04	Inverter	1		1
	TA07	Buffer	1		1
	TA08	2-input AND	1		1
	TA10	3-input NAND	1		1
	TA11	3-input AND	1		1
	TA138	3-to-8 decoder with enable and active low outputs	2		12
	TA139	2-to-4 decoder with active low enable and outputs	1		4
	TA150	16-to-1 multiplexor with active low enable	3		6
	TA151	8-to-1 multiplexor with enable and both active low and active high output	3		5
	TA153	4-to-1 multiplexor with active low enable	2		2
	TA154	4-to-16 decoder with active low outputs and select lines	2		22
	TA157	2-to-1 multiplexor with active low enable	1		1
	TA160	4-bit decade counter with active low clear and load	4	4	8
	TA161	4-bit binary counter with active low clear and load	3	4	6
	TA164	8-bit serial in, parallel out shift register, active low clear	1	8	
	TA169	4-bit Up/Down Counter	6	4	14
	TA174	hex D-type flip-flop with active low clear	1	6	
	TA175	quadruple D-type flip-flop with active low clear	1	4	
	TA181	ALU			37
	TA190	4-bit up/down decode counter with up/down mode	7	4	31
	TA191	4-bit up/down binary counter with up/down mode	7	4	30
	TA194	4-bit bidirectional universal shift register	1	4	4
	TA195	4-bit parallel-access shift register	1	4	1
	TA20	4-input NAND	1		2
	TA21	4-input AND	1		1
	TA269	8-bit up/down binary counter	8	8	28
	TA27	3-input NOR	1		1
	TA273	octal register with clear	1	8	
	TA280	9-bit odd/even parity generator and checker	4		9
	TA32	2-input OR	1		1
	TA377	octal register with active low enable	1	8	
	TA40	4-input NAND	1		2
	TA42	4 to 10 decoder	1		10
	TA51	AND-OR-Invert	1		2
	TA54	4-wide 2-input AND-OR-Invert	2		5
	TA55	2-wide 4-input AND-OR-Invert	2		3
	TA688	8-bit identity comparator	3		9
	TA86	2-input exclusive OR	1		1

Package Pin Assignments

176-Pin CPGA (Top View)



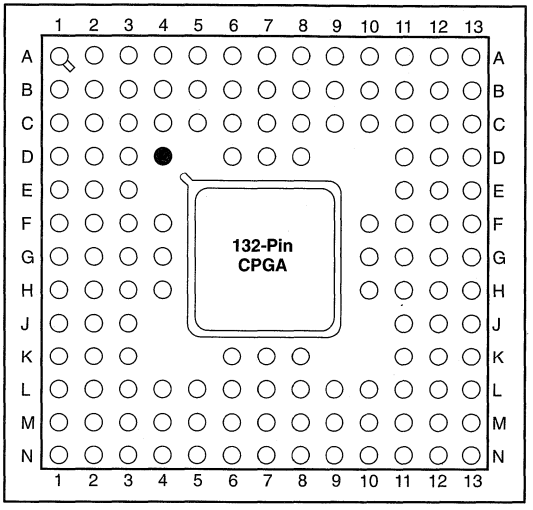
Signal	Pad Number	Location
CLKA or I/O	154	A9
CLKB or I/O	158	B8
DCLK or I/O	175	B3
GND	1, 8, 18, 23, 33, 38, 45, 57, 67, 77, 89 101, 106, 111, 121, 126, 133, 145, 156, 165	D4, E4, G4, H4, K4, L4, M4, M6, M8, M10, M12 K12, J12, H12, F12, E12, D12, D10, C8, D6
MODE	2	C3
PRA or I/O	152	C9
PRB or I/O	160	D7
SDI or I/O	135	B14
V_{CC}	13, 24, 28, 52, 68, 82, 112, 116, 140, 155, 170	F4, H3, J4, M5, N8, M11, H13, G12, D11, D8, D5
V_{KS}	109	J13
V_{PP}	110	J14
V_{SV}	25, 113	H2, H14

Notes:

1. *Unused I/O pins are designated as outputs by ALS and are driven low.*
2. *All unassigned pins are available for use as I/Os.*
3. *MODE = GND, except during device programming or debugging.*
4. *$V_{PP} = V_{CC}$, except during device programming.*
5. *$V_{SV} = V_{CC}$, except during device programming*
6. *V_{KS} = GND, except during device programming.*

Package Pin Assignments (continued)

132-Pin CPGA (Top View)



● Orientation Pin

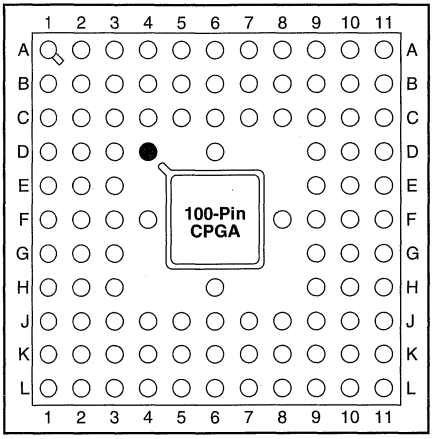
Signal	Pad Number	Location
CLKA or I/O	115	B7
CLKB or I/O	119	B6
DCLK or I/O	132	C3
GND	9, 10, 26, 27, 41, 58, 59, 73, 74, 92, 93, 107, 108, 125, 126	E3, F4, J2, J3, L5, L9, M9, K12, J11, E12, E11, C9, B9, B5, C5
MODE	2	A1
PRA or I/O	113	B8
PRB or I/O	121	C6
SDI or I/O	101	B12
V_{CC}	18, 19, 49, 50, 83, 84, 116, 117	G3, G2, L7, K7, G10, G11, D7, C7
V_{KS}	81	H13
V_{PP}	82	G13
V_{SV}	17, 85	G4, G12

Notes:

1. *Unused I/O pins are designated as outputs by ALS and are driven low.*
2. *All unassigned pins are available for use as I/Os.*
3. *MODE = GND, except during device programming or debugging.*
4. *$V_{PP} = V_{CC}$, except during device programming.*
5. *$V_{SV} = V_{CC}$, except during device programming*
6. *V_{KS} = GND, except during device programming.*

Package Pin Assignments (continued)

100-Pin CPGA (Top View)



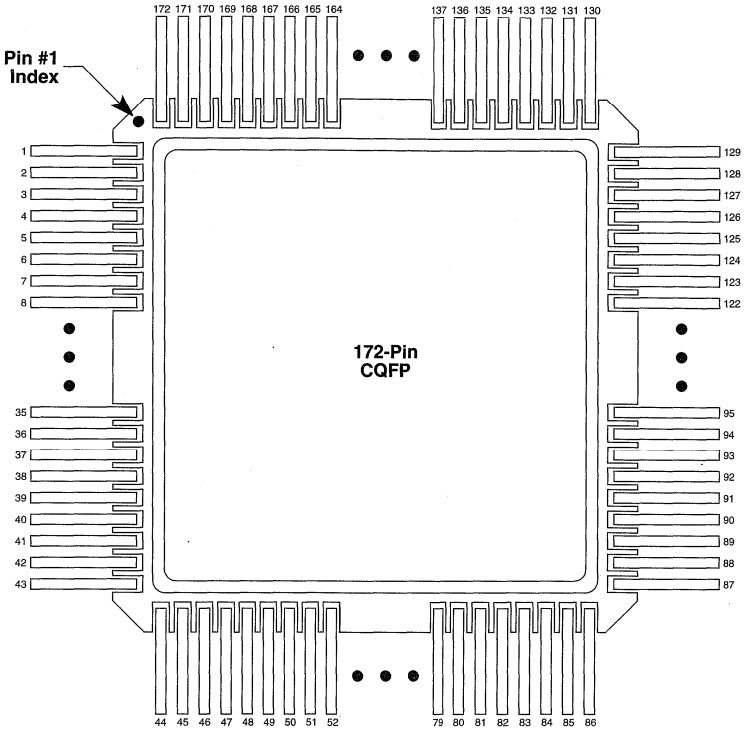
Signal	Pad Number	Location
CLKA or I/O	87	C6
CLKB or I/O	90	D6
DCLK or I/O	100	C3
GND	7, 20, 32, 44, 55, 70, 82, 94	E3, G3, J5, J7, G9, D10, C7, C5
MODE	2	C2
PRA or I/O	85	A7
PRB or I/O	92	A4
SDI or I/O	77	C8
V_{CC}	15, 38, 64, 88	F3, K6, F9, B6
V_{KS}	62	F11
V_{PP}	63	F10
V_{SV}	14, 65	G1, E11

Notes:

1. *Unused I/O pins are designated as outputs by ALS and are driven low.*
2. *All unassigned pins are available for use as I/Os.*
3. *MODE = GND, except during device programming or debugging.*
4. *$V_{PP} = V_{CC}$, except during device programming.*
5. *$V_{SV} = V_{CC}$, except during device programming*
6. *V_{KS} = GND, except during device programming.*

Package Pin Assignments (continued)

172-Pin CQFP (Top View)



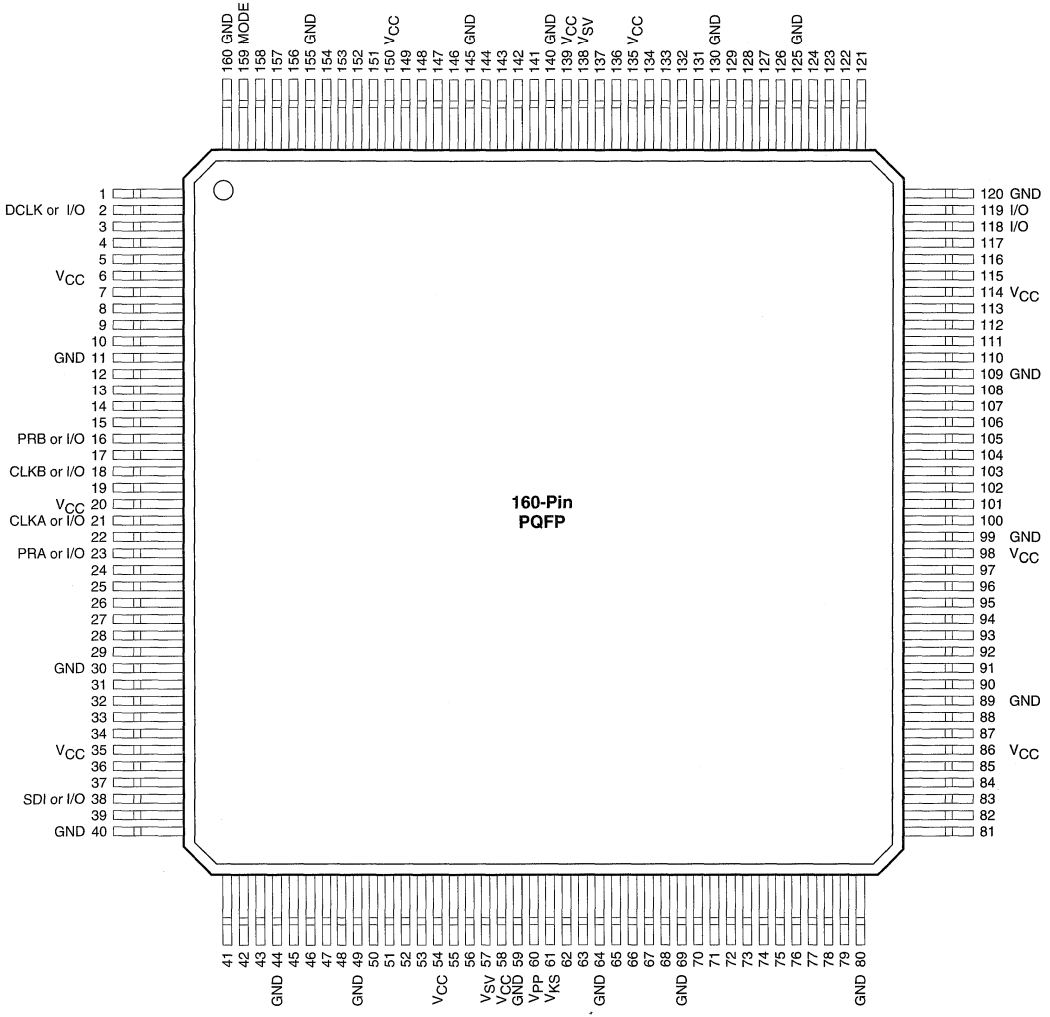
Signal	PIN Number
CLKA or I/O	150
CLKB or I/O	154
DCLK or I/O	171
GND	7, 17, 22, 32, 37, 55, 65, 75, 98, 103, 108, 118, 123, 141, 152, 161
MODE	1
PRA or I/O	148
PRB or I/O	156
SDI or I/O	131
V_{CC}	12, 23, 27, 50, 66, 80, 109, 113, 136, 151, 166
V_{KS}	106
V_{PP}	107
V_{SV}	24, 110

Notes:

1. *Unused I/O pins are designated as outputs by ALS and are driven low.*
2. *All unassigned pins are available for use as I/Os.*
3. *MODE = GND, except during device programming or debugging.*
4. *$V_{PP} = V_{CC}$, except during device programming.*
5. *$V_{SV} = V_{CC}$, except during device programming*
6. *V_{KS} = GND, except during device programming.*

Package Pin Assignments (continued)

160-Pin PQFP (Top View)

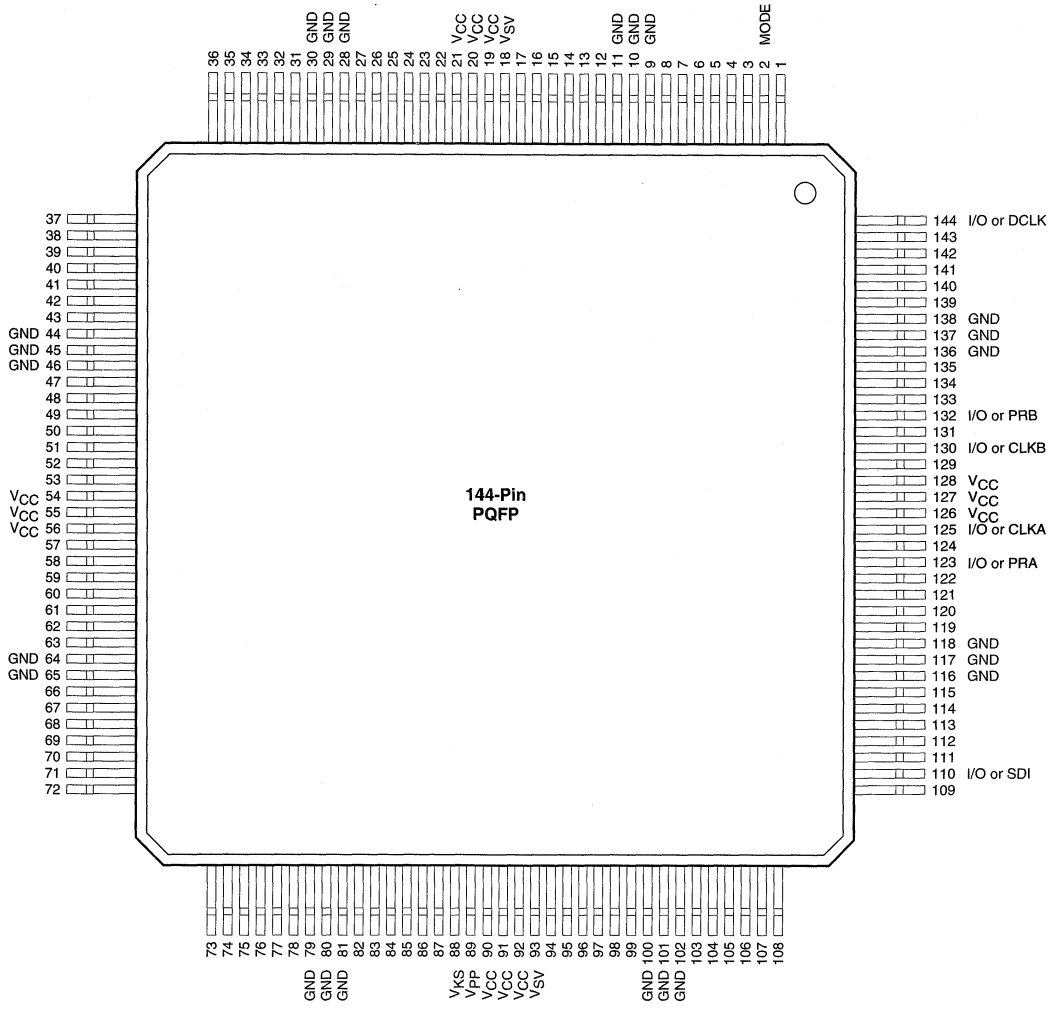


Notes:

1. *Unused I/O pins are designated as outputs by ALS and are driven low.*
2. *All unassigned pins are available for use as I/Os.*
3. *MODE = GND, except during device programming or debugging.*
4. *$V_{PP} = V_{CC}$, except during device programming.*
5. *$V_{SV} = V_{CC}$, except during device programming*
6. *V_{KS} = GND, except during device programming.*

Package Pin Assignments (continued)

144-Pin PQFP (Top View)

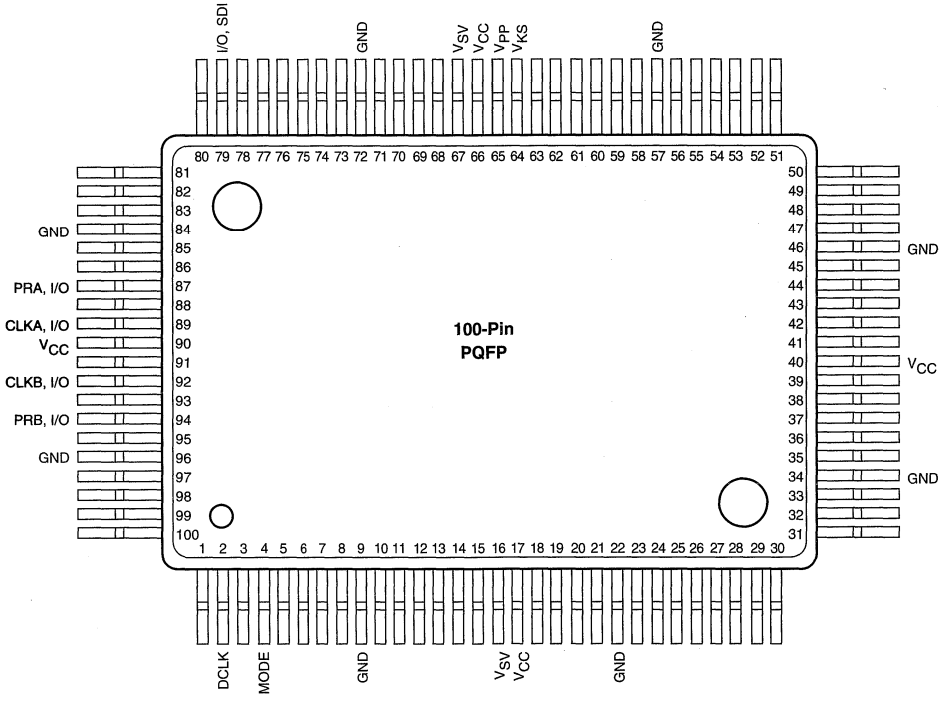


Notes:

1. *Unused I/O pins are designated as outputs by ALS and are driven low.*
2. *All unassigned pins are available for use as I/Os.*
3. *MODE = GND, except during device programming or debugging.*
4. *$V_{PP} = V_{CC}$, except during device programming.*
5. *$V_{SV} = V_{CC}$, except during device programming*
6. *V_{KS} = GND, except during device programming.*

Package Pin Assignments (continued)

100-Pin PQFP (Top View)

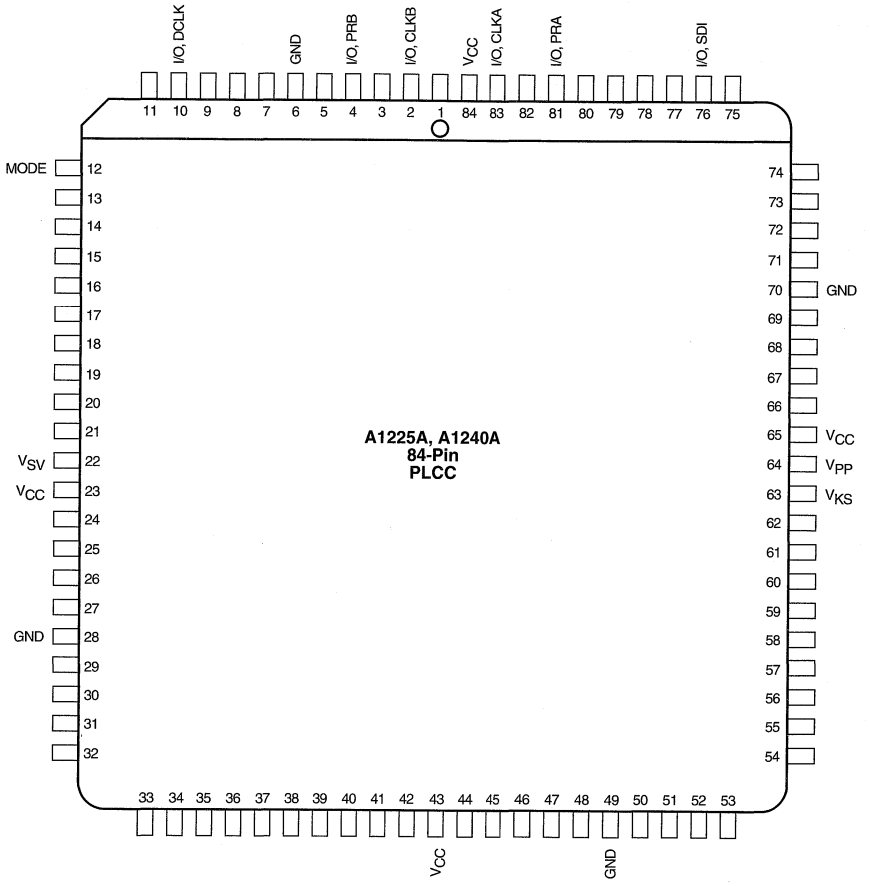


Notes:

1. *Unused I/O pins are designated as outputs by ALS and are driven low.*
2. *All unassigned pins are available for use as I/Os.*
3. *MODE = GND, except during device programming or debugging.*
4. *$V_{PP} = V_{CC}$, except during device programming.*
5. *$V_{SV} = V_{CC}$, except during device programming*
6. *V_{KS} = GND, except during device programming.*

Package Pin Assignments (continued)

84-Pin PLCC (Top View)

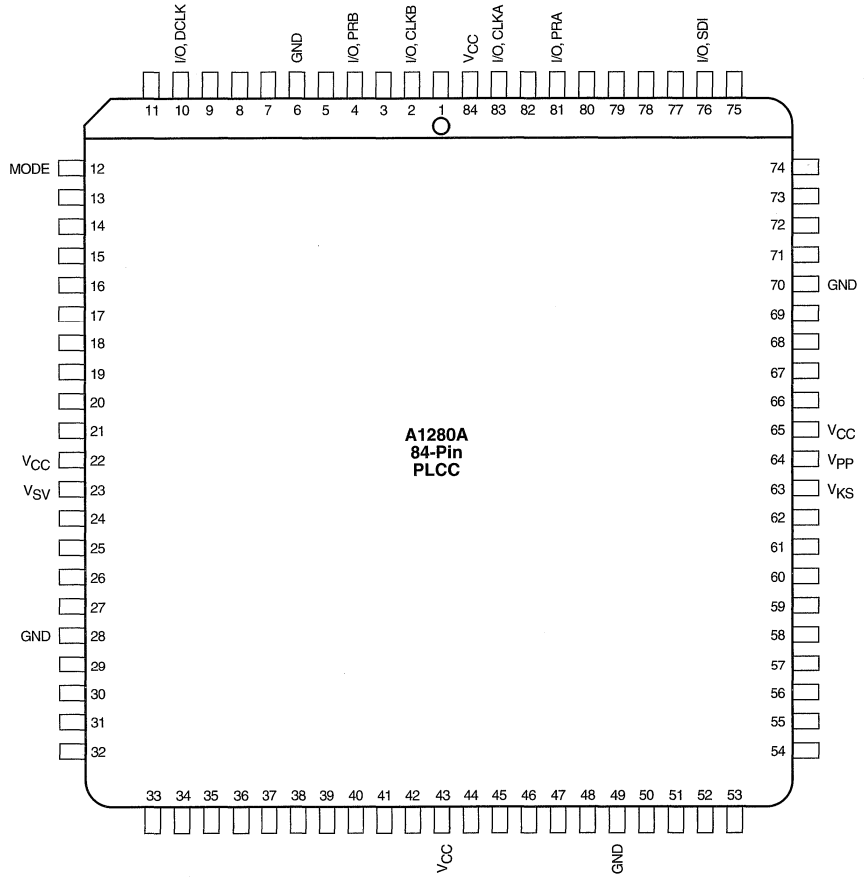


Notes:

1. *Unused I/O pins are designated as outputs by ALS and are driven low.*
2. *All unassigned pins are available for use as I/Os.*
3. *MODE = GND, except during device programming or debugging.*
4. *$V_{PP} = V_{CC}$, except during device programming.*
5. *$V_{SV} = V_{CC}$, except during device programming*
6. *$V_{KS} = GND$, except during device programming.*

Package Pin Assignments (continued)

84-Pin PLCC (Top View)

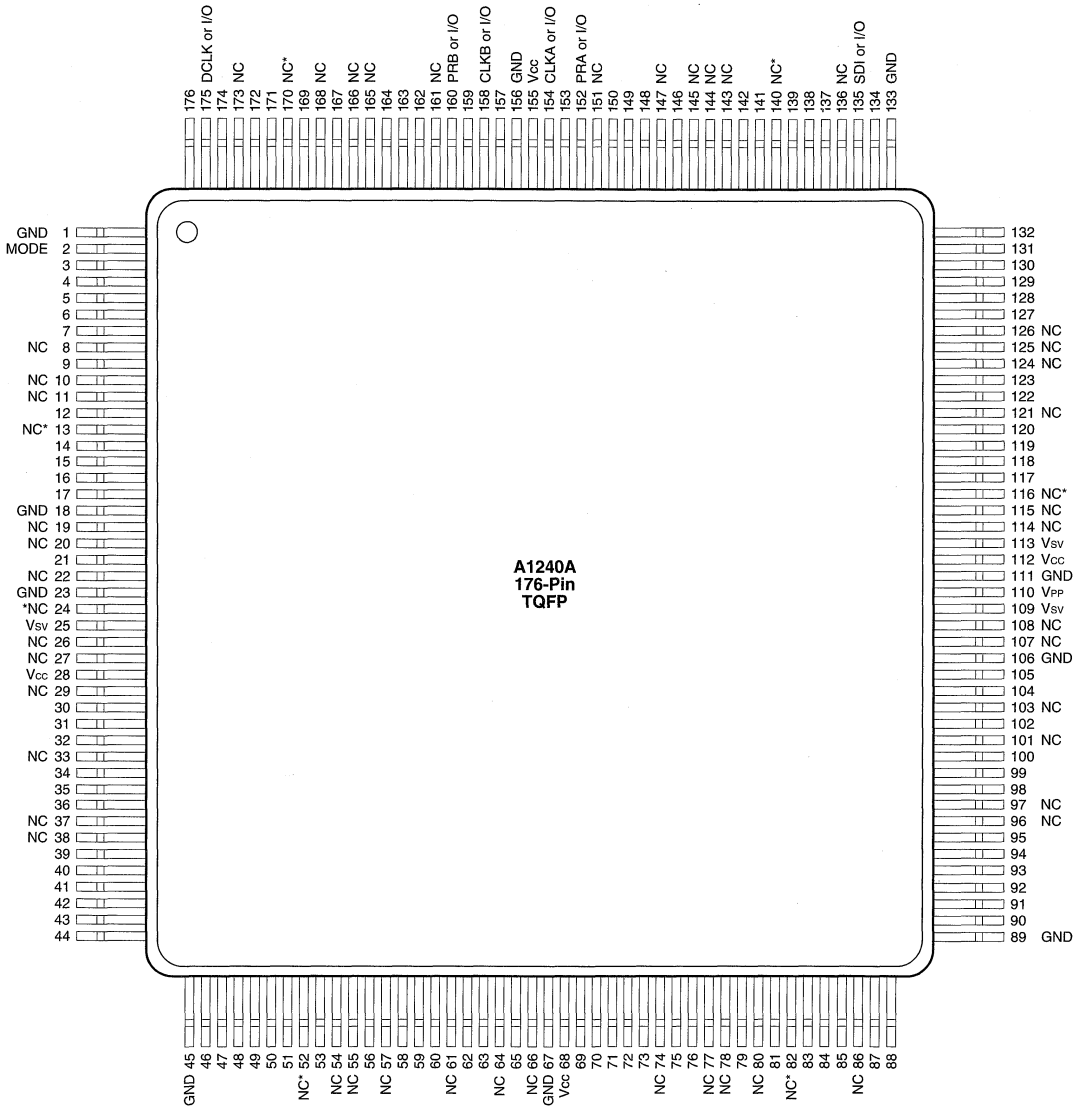


Notes:

1. *Unused I/O pins are designated as outputs by ALS and are driven low.*
2. *All unassigned pins are available for use as I/Os.*
3. *MODE = GND, except during device programming or debugging.*
4. *$V_{PP} = V_{CC}$, except during device programming.*
5. *$V_{SV} = V_{CC}$, except during device programming*
6. *V_{KS} = GND, except during device programming.*

Package Pin Assignments (continued)

176-Pin TQFP (Top View)



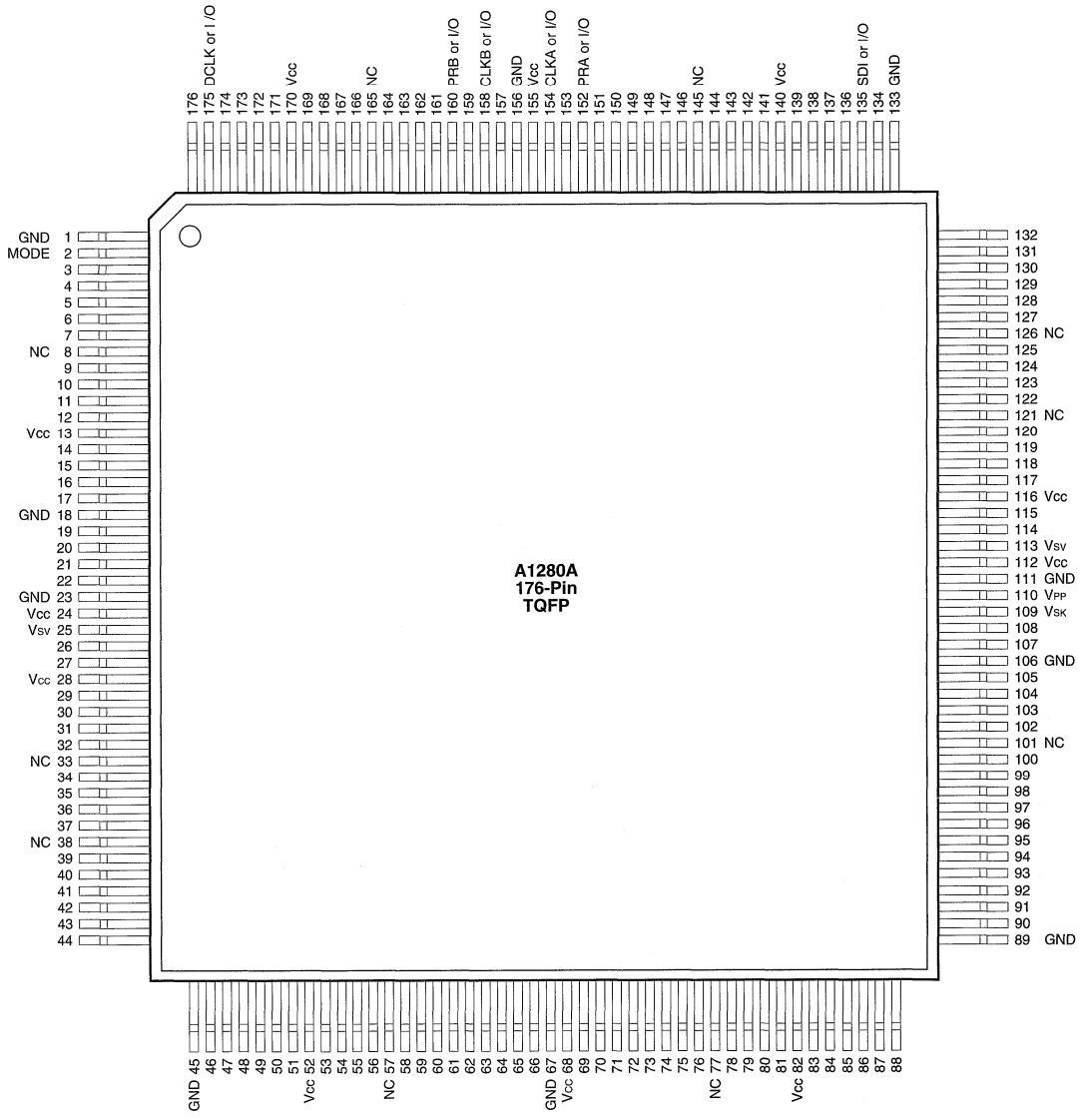
* For pin compatibility, these pins should be reserved for Vcc in higher capacity devices.

Notes:

1. *Unused I/O pins are designated as outputs by ALS and are driven low.*
2. *All unassigned pins are available for use as I/Os.*
3. *MODE = GND, except during device programming or debugging.*
4. $V_{PP} = V_{CC}$, *except during device programming.*
5. $V_{SV} = V_{CC}$, *except during device programming*
6. V_{KS} *= GND, except during device programming.*

Package Pin Assignments (continued)

176-Pin TQFP (Top View)

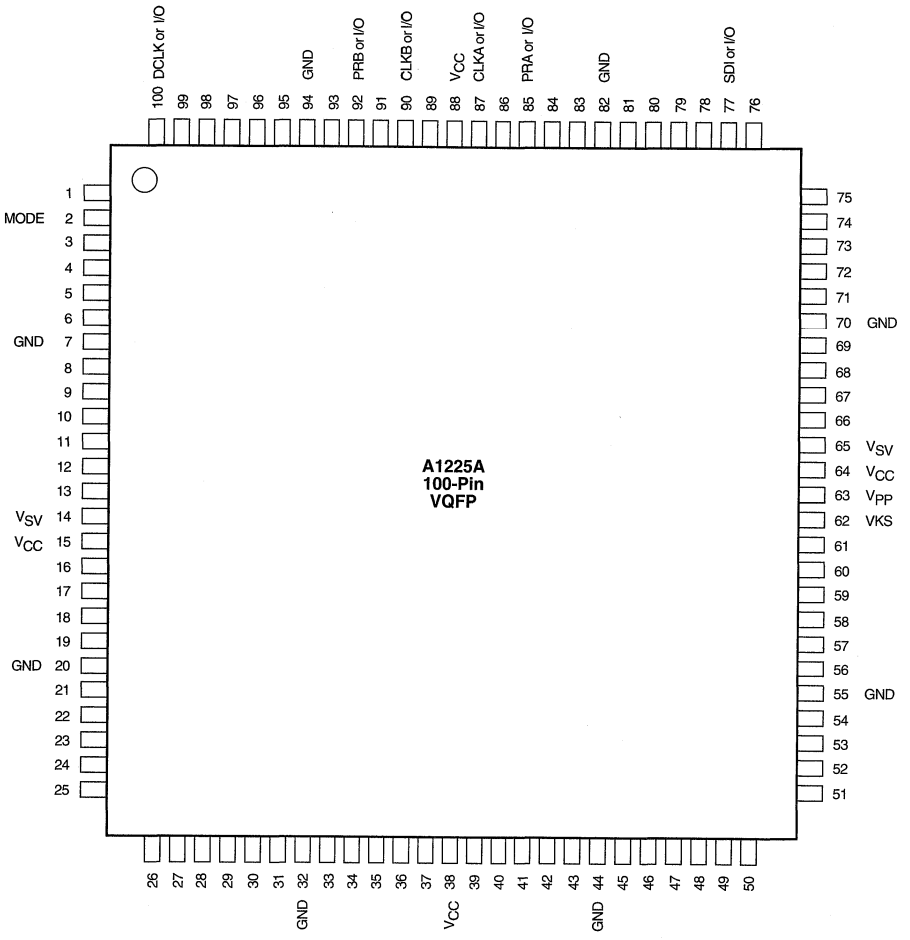


Notes:

1. *Unused I/O pins are designated as outputs by ALS and are driven low.*
2. *All unassigned pins are available for use as I/Os.*
3. *MODE = GND, except during device programming or debugging.*
4. *$V_{PP} = V_{CC}$, except during device programming.*
5. *$V_{SV} = V_{CC}$, except during device programming*
6. *V_{KS} = GND, except during device programming.*

Package Pin Assignments (continued)

100-Pin VQFP (Top View)



Notes:

1. *Unused I/O pins are designated as outputs by ALS and are driven low.*
2. *All unassigned pins are available for use as I/Os.*
3. *MODE = GND, except during device programming or debugging.*
4. *$V_{PP} = V_{CC}$, except during device programming.*
5. *$V_{SV} = V_{CC}$, except during device programming*
6. *V_{KS} = GND, except during device programming.*

1200XL
Field Programmable Gate Arrays

Features

- Up to 8000 Gate Array Gates (20,000 PLD equivalent gates)
- Replaces up to 200 TTL Packages
- Replaces up to eighty 20-Pin PAL® Packages
- Design Library with over 500 Macro Functions
- Single-Module Sequential Functions
- Wide-Input Combinatorial Functions
- Up to 1232 Programmable Logic Modules
- Up to 998 Flip-Flops
- Datapath Performance at 135 MHz
- 16-Bit Accumulator Performance to 50 MHz
- 10 ns Clock-Out speeds
- Two In-Circuit Diagnostic Probe Pins Support Speed Analysis to 50 MHz
- Two High-Speed, Low-Skew Clock Networks
- I/O Drive to 10 mA
- Nonvolatile, User Programmable
- Fabricated in 0.6 micron CMOS technology

Product Family Profile

Device	A1225XL	A1240XL	A1280XL
Capacity			
Gate Array Equivalent Gates	2,500	4,000	8,000
PLD Equivalent Gates	6,250	10,000	20,000
TTL Equivalent Packages	63	100	200
20-Pin PAL Equivalent Packages	25	40	80
Logic Modules	451	684	1,232
S-Modules	231	348	624
C-Modules	220	336	608
Flip-Flops (maximum)	382	568	998
Routing Resources			
Horizontal Tracks/Channel	36	36	36
Vertical Tracks/Channel	15	15	15
PLICE Antifuse Elements	250,000	400,000	750,000
User I/Os (maximum)	83	104	140
Packages[1]	100 PQFP	144 PQFP	160 PQFP
	100 VQFP	176 TQFP	176 TQFP
	84 PLCC	84 PLCC	84 PLCC
	100 CPGA	132 CPGA	176 CPGA
Performance[2]			
16-Bit Prescaled Counters	135 MHz	130 MHz	110 MHz
16-Bit Loadable Counters	87 MHz	83 MHz	75 MHz
16-Bit Accumulators	47 MHz	45 MHz	42 MHz

Notes:

1. *See product plan on page 1-105 for package availability.*
2. *Performance is based on '–1' speed devices at commercial worst-case operating conditions using PREP Benchmarks (mean frequency results), Suite #1, Version 1.2, dated 3-28-93, any analysis is not endorsed by PREP.*

Description

The 1200XL family, Actel's fourth family of field programmable gate arrays (FPGAs), is targeted as a VALUE family, offering very low costs and mid-to-high range performance. The 1200XL family is based on Actel's patented channeled array architecture and consists of two-module types: combinatorial (C-modules) and combinatorial-sequential (S-modules). The 1200XL family is pin and functionally compatible with the ACT 2 family, and is design compatible with the ACT 1 and ACT 3 families. The 1200XL family provides significant performance enhancements in comparison to the ACT 2 family while offering a lower cost solution. The devices are implemented in 0.65 micron two-level metal CMOS technology and employ Actel's patented PLICE antifuse technology.

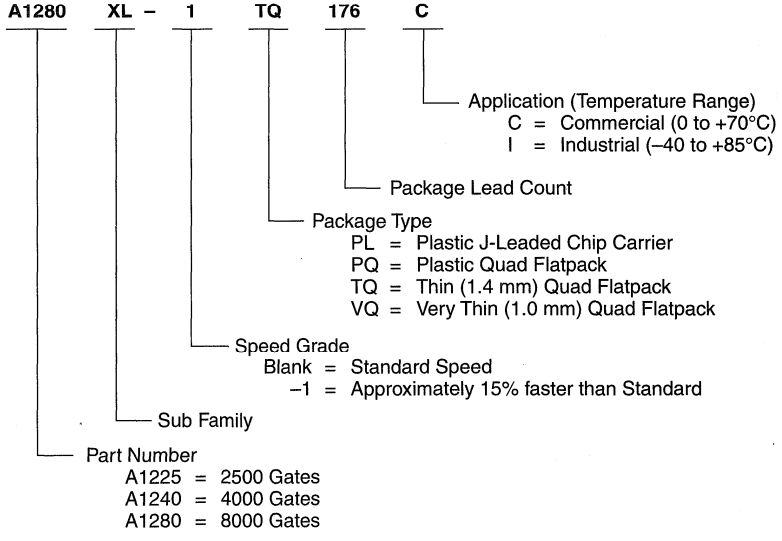
The 1200XL family is supported by Actel's Designer and Designer Advantage Systems, allowing logic design implementation with minimum effort. The systems offer Microsoft® Windows™ and X Window™ graphical user interfaces and integrate with the resident CAE system to provide a complete gate array design environment: schematic capture, simulation, fully automatic placement and routing, timing verification and device programming. The systems also include the ACTmap™ optimization and synthesis tool, and the ACTgen™ Macro Builder, a powerful macro function generator for counters, adders, and other structured blocks. The systems are available for 386/486/Pentium PCs and for HP™, and Sun™ workstations running Viewlogic®, Mentor Graphics®, and OrCAD™ tools.

Performance versus ACT 2

The 1200XL family has been enhanced for performance in comparison to the ACT 2 family. The I/O modules have been redesigned to improve pin-to-pin and clock-out delays. The clock distribution networks have been enhanced to improve both propagation delays and skews for highly loaded designs. Finally, the family is implemented in 0.65 micron CMOS technology, resulting in much faster performance.

	A1225A	A1225A-2	A1225XL	A1225XL-1
Clock-Out (pad-pad, 64 loads)	24.0 ns	18.0 ns	12.0 ns	10.0 ns
Pin-Pin (1 level, FO = 1)	28.3 ns	21.2 ns	15.5 ns	13.2 ns
PREP Datapath (#1)	75 MHz	105 MHz	115 MHz	135 MHz
PREP Accumulator (#6)	28 MHz	37 MHz	40 MHz	47 MHz
PREP State Machine (#3)	32 MHz	43 MHz	51 MHz	60 MHz

Ordering Information



Product Plan[1]

	Speed Grade*		Application	
	Std	–1	C	I
A1225XL Device				
84-pin Plastic Leaded Chip Carrier (PL)	P	P	P	P
100-pin Plastic Quad Flatpack (PQ)	P	P	P	P
100-pin Very Thin (1.0 mm) Quad Flatpack	P	P	P	P
A1240XL Device				
84-pin Plastic Leaded Chip Carrier (PL)	P	P	P	P
144-pin Plastic Quad Flatpack (PQ)	P	P	P	P
176-pin Thin (1.4 mm) Quad Flatpack	P	P	P	P
A1280XL Device				
84-pin Plastic Leaded Chip Carrier (PL)	✔	✔	✔	P
160-pin Plastic Quad Flatpack (PQ)	✔	✔	✔	P
176-pin Thin (1.4 mm) Quad Flatpack	✔	✔	✔	P

Applications: *C = Commercial* *I = Industrial*

Availability: ✔ = *Available* P = *Planned* — = *Not Planned*

**Speed Grade:* *–1 = Approx. 15% faster than Standard*

Note:

1. Please consult Actel representatives for current availability.

Device Resources

Device Series	Logic Modules	Gates	User I/Os					
			PQFP			PLCC	TQFP	VQFP
			160-pin	144-pin	100-pin	84-pin	176-pin	100-pin
A1225XL	451	2500	—	—	83	72	—	83
A1240XL	684	4000	—	104	—	72	104	—
A1280XL	1232	8000	125	—	—	72	140	—

Pin Description

CLKA **Clock A (Input)**

TTL Clock input for clock distribution networks. The Clock input is buffered prior to clocking the logic modules. This pin can also be used as an I/O.

CLKB **Clock B (Input)**

TTL Clock input for clock distribution networks. The Clock input is buffered prior to clocking the logic modules. This pin can also be used as an I/O.

DCLK **Diagnostic Clock (Input)**

TTL Clock input for diagnostic probe and device programming. DCLK is active when the MODE pin is HIGH. This pin functions as an I/O when the MODE pin is LOW.

GND **Ground**

LOW supply voltage.

I/O **Input/Output (Input, Output)**

The I/O pin functions as an input, output, three-state, or bidirectional buffer. Input and output levels are compatible with standard TTL and CMOS specifications. Unused I/O pins are automatically driven LOW by the ALS software.

MODE **Mode (Input)**

The MODE pin controls the use of multifunction pins (DCLK, PRA, PRB, SDI). When the MODE pin is HIGH, the special functions are active. When the MODE pin is LOW, the pins function as I/Os.

NC **No Connection**

This pin is not connected to circuitry within the device.

PRA **Probe A (Output)**

The Probe A pin is used to output data from any user-defined design node within the device. This independent diagnostic pin is used in conjunction with the Probe B pin to allow real-time diagnostic output of any signal path within the device. The Probe A pin can be used as a user-defined I/O when debugging has been completed. The pin's probe capabilities can be permanently disabled to protect programmed design confidentiality. PRA is active when the MODE pin is HIGH. This pin functions as an I/O when the MODE pin is LOW.

PRB **Probe B (Output)**

The Probe B pin is used to output data from any user-defined design node within the device. This independent diagnostic pin is used in conjunction with the Probe A pin to allow real-time diagnostic output of any signal path within the device. The Probe B pin can be used as a user-defined I/O when debugging has been completed. The pin's probe capabilities can be permanently disabled to protect programmed design confidentiality. PRB is active when the MODE pin is HIGH. This pin functions as an I/O when the MODE pin is LOW.

SDI **Serial Data Input (Input)**

Serial data input for diagnostic probe and device programming. SDI is active when the MODE pin is HIGH. This pin functions as an I/O when the MODE pin is LOW.

V_{CC} **5 V Supply Voltage**

HIGH supply voltage.

V_{KS} **Programming Voltage**

Supply voltage used for device programming. This pin must be connected to GND during normal operation.

V_{PP} **Programming Voltage**

Supply voltage used for device programming. This pin must be connected to V_{CC} during normal operation.

V_{SV} **Programming Voltage**

Supply voltage used for device programming. This pin must be connected to V_{CC} during normal operation.

1200XL Architecture

This section of the data sheet is meant to familiarize the user with the architecture of 1200XL family devices. A generic description of the family will be presented first, followed by a detailed description of the logic blocks, the routing structure, the antifuses, and the special function circuits. Diagrams for the ACT 2 devices are provided at the end of the data sheet. The additional circuitry required to program and test the devices will not be covered.

Array Topology

The 1200XL family architecture is composed of five key building blocks: Logic modules, I/O modules, Routing Tracks, Global Clock Networks, and Probe Circuits. The basic structure is similar for all devices in the family, differing only in the number of rows, columns, or I/Os (see Table 1).

The logic and I/O modules are arranged in a two-dimensional array (Figure 1). There are three types of modules: Logic, I/O, and Bin. Logic and I/O modules are available as user resources. Bin modules are used during testing and are not available to users.

Logic Modules

Logic modules are classified into two types: combinatorial (C-modules) and sequential (S-modules) (see Figures 2 and 3). The C-module is an enhanced version of the ACT 1 family logic module optimized to implement high fanin combinatorial macros, such as 5-input AND, and 5-input OR. The full ACT 2 combinatorial logic module is available for use as the CM8 hard macro. The S-module is designed to implement high-speed flip-flop functions within a single module. S-modules also include combinatorial logic, which allows an additional level of logic to be implemented without additional propagation delay. C-modules and S-modules are arranged in pairs called module-pairs. Module-pairs are arranged in alternating pairs (shown in Figure 1) and make up the bulk of the array. This arrangement allows the

Table 1 • Array Sizes

Device	Rows	Columns	Logic	I/O
A1225XL	13	46	451	83
A1240XL	14	62	684	104
A1280XL	18	82	1232	140

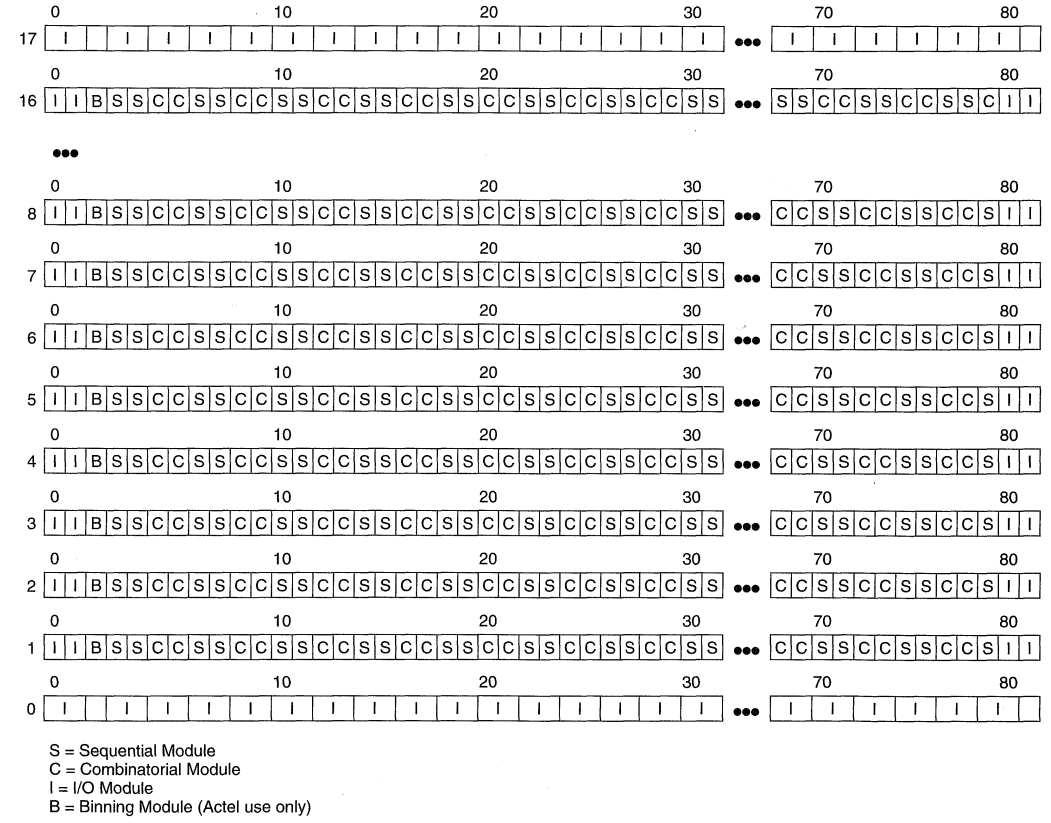


Figure 1 • A1280XL Simplified Floor Plan

placement software to support two-module macros of four types (CC, CS, SC, and SS). I/O modules are arranged around the periphery of the array.

The combinatorial module (shown in Figure 2) implements the following function:

Y = !S1 * !S0 * D00 + !S1 * S0 * D01 * S1 * !S0 * D10 + S1 * S0 * D11

where:

S0 = A0 * B0

S1 = A1 + B1

The sequential module implements this same function Y (except that S0 = A0 only, since the B0 input is used for reset), followed by a sequential block. The sequential block can implement either a D-type flip-flop or a transparent latch. It can also be fully transparent so that the S-modules can be used to implement purely combinatorial functions. The function of the sequential module is determined by the macro selection from the design library of hard macros.

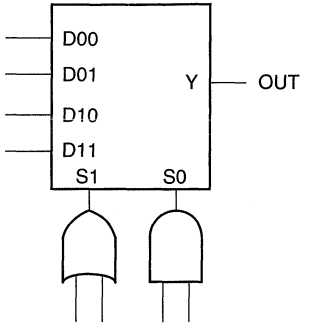
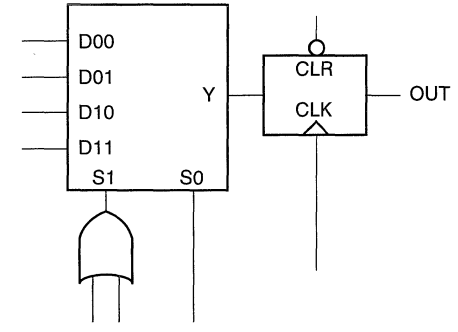
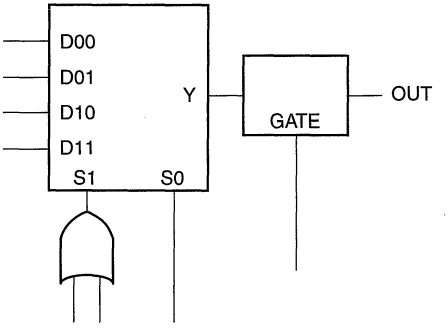
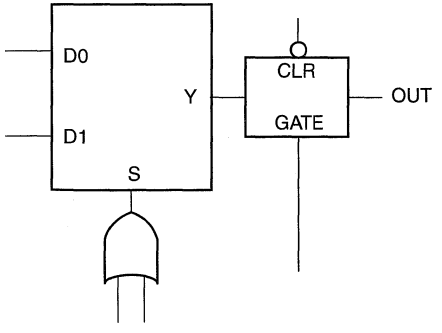


***Figure 2** • C-module Implementation*

Allowable S-module implementations are shown in Figure 3.







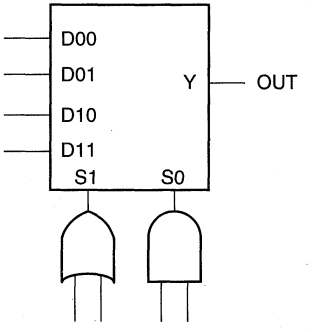


***Figure 3** • S-module Implementations*

I/Os

The I/O architecture consists of pad drivers located near the bonding pads and I/O modules located in the array. Top/bottom I/O modules are located in the top and bottom rows respectively. Side I/O modules occupy the leftmost two columns and the rightmost two columns of the array. The function of all I/O modules is identical, but the top/bottom I/O modules have a different routing interface to the array than the side I/O modules. I/Os implement a variety of user functions determined by library macro selection.

Special Purpose I/Os

Certain I/O pads are temporarily used for programming and testing the device. During normal user operation, these special I/O pads are identical to other I/O pads. The following special I/O pads and their functions are shown in Table 2.

Table 2 • Special I/O Pads

SDI	Serial Data In
DCLK	Serial Data Clock In
PRA	Probe A Output
PRB	Probe B Output

Two other pads, CLKA and CLKB, also differ from normal I/Os in that they can be used to drive the global clock networks. Power, Ground, and Programming pads are not considered I/O functions. Their function is summarized as follows:

V_{CC}	Power
GND	Circuit Ground
V_{SV}, V_{KS}, V_{PP}	Programming Pads
MODE	Program/Debug Control

I/O Pads

I/O pads are located on the periphery of the die and consist of the bonding pad, the high-drive CMOS drivers, and the TTL level-shifter inputs. Each I/O pad is associated with a specific I/O module. Connections from the I/O pad to the I/O module are made using the signals DATAOUT, DATAIN, and EN (shown in Figure 4).

I/O Modules

There are two types of I/O modules: side and top/bottom. The I/O module schematic is shown in Figure 5. In the side I/O modules, there are two inputs supplying the data to be output from the chip UO1 and UO2. (UO stands for user output.) Two are used so that the router can choose to take the signal from either the routing channel above or the routing channel below the I/O module. The top/bottom I/O modules interact with only one channel and therefore have only one UO input.

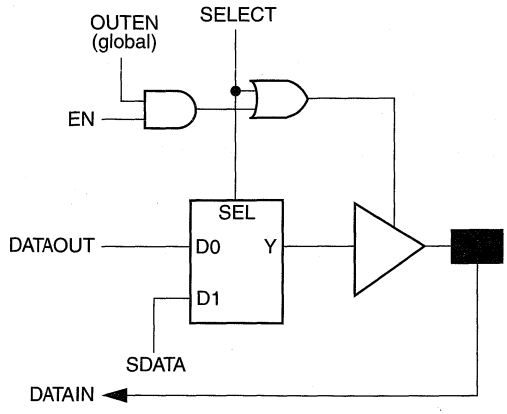


Figure 4 • I/O Pad Signals

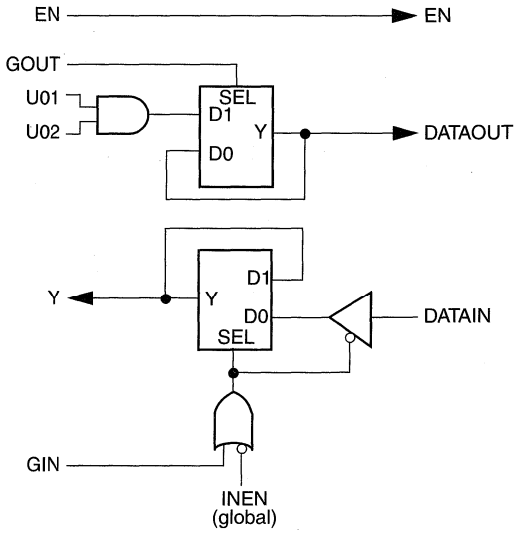


Figure 5 • I/O Module

The EN input enables the tristate output buffer. The global signals INEN and OUTEN (Figures 4 and 5) are used to disable the inputs and outputs during certain test modes. Latches are provided in the input and output path. When GOUT is high, the latch is transparent. The latch can be used as the second stage of a rising-edge flip-flop. GIN is the reverse of GOUT. When GIN is high, the input data is latched; when it is low, the input latch becomes transparent.

The output of the module, Y, is used for data being input to the chip. Side I/O modules have a dedicated output segment for Y extending into the routing channels above and below (similar to logic modules). Side I/O modules may also connect to the array through nondedicated Long Vertical Tracks (LVTs). Top/Bottom I/O modules have no dedicated output segment. Signals coming into the chip from the top or bottom must be routed using F-fuses and LVTs (F-fuses and LVTs are explained in detail in the routing section). I/O signals connected to I/O modules on either the top or bottom of the array may incur a delay penalty over signals connected to I/O modules on the sides.

Hard Macros

Designing within the Actel design environment is accomplished using a building block approach. Over 350 logic function macros are provided in the 1200XL design library. Hard macro logic functions range from simple SSI gates such as AND, NOR, and Exclusive OR to more complex functions such as flip-flops with 4:1 Multiplexed Data inputs. Hard macros are implemented in the ACT 2 architecture by using one or more C-modules or S-modules. Over 200 of the macros are implemented in a single module, while several two-module macros are also available. Two-module hard macros always utilize a module-pair, either SS, CC, CS, or SC. Because one- and two-module macros have small propagation delay variances, their performances can be predicted very accurately. Hard macro propagation delays are specified in the data sheet. Soft macros comprise multiple hard macros connected together to form complex functions. These functions range from MSI functions to 16-bit counters and accumulators. A large number of TTL equivalent hard and soft macros are also provided. Soft macro delays are not specified in the data sheet.

Routing Structure

The 1200XL architecture uses Vertical and Horizontal routing tracks to interconnect the various logic and I/O modules. These routing tracks are metal interconnects that may either be of continuous length or broken into pieces called segments. Segments can be joined together at the ends using antifuses to increase their lengths up to the full length of the track.

Horizontal Routing

Horizontal channels are located between the rows of modules and are composed of several routing tracks. The horizontal routing tracks within the channel are divided into one or more segments. The minimum horizontal segment length is the width of a module-pair, and the maximum horizontal segment length is the full length of the channel. Any segment that spans more than one-third the row length is considered a long horizontal segment. A typical channel is shown in Figure 6. Nondedicated horizontal routing tracks are used to route signal nets. Dedicated routing tracks are used for the global clock networks and for power and ground tie-off tracks.

Vertical Routing

Other tracks run vertically through the module. Vertical tracks are of three types: input, output, and long. Vertical tracks are also divided into one or more segments. Each segment in an input track is dedicated to the input of a particular module. Each segment in an output track is dedicated to the output of a particular module. Long segments are uncommitted and can be assigned during routing. Each output segment spans four channels (two above and two below), except near the top and bottom of the array where edge effects occur. LVTs contain either one or two segments. An example of vertical routing tracks and segments is shown in Figure 7.

Antifuse Structures

An antifuse is a "normally open" structure as opposed to the normally closed fuse structure used in PROMs or PALs. The use of antifuses to implement a Programmable Logic Device results in highly testable structures as well as efficient programming algorithms. The structure is highly testable because there are no preexisting connections; therefore, temporary connections can be made using pass transistors. These temporary connections can isolate individual antifuses to be programmed as well as isolate individual circuit structures to be tested. This can be done both before and after programming. For example, all metal tracks can be tested for continuity and shorts between adjacent tracks, and the functionality of all logic modules can be verified.

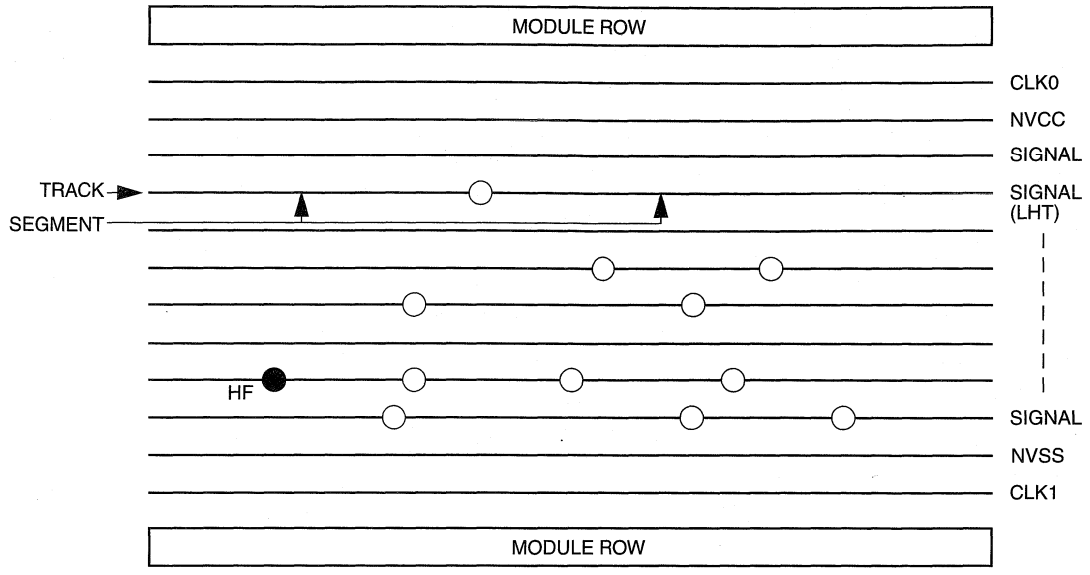


Figure 6 • Horizontal Routing Tracks and Segments

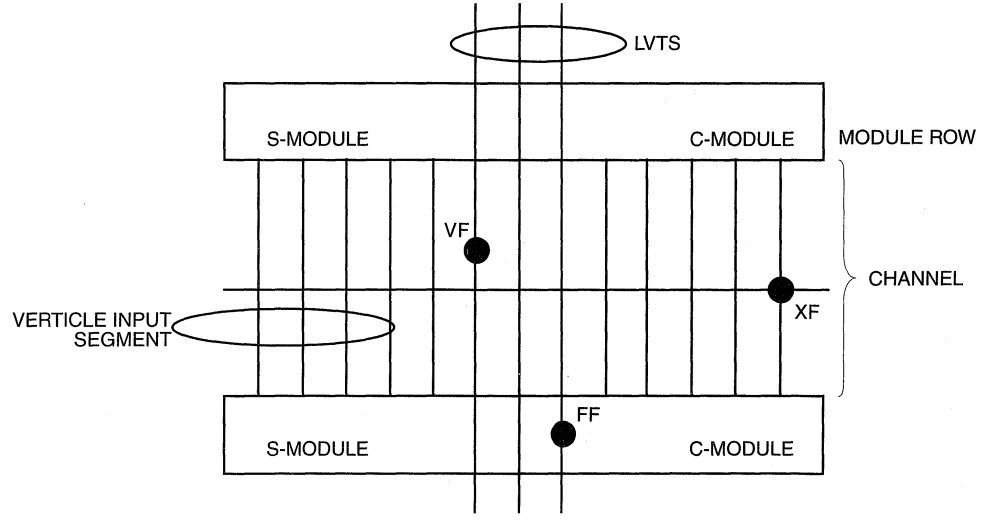


Figure 7 • Vertical Routing Tracks and Segments

Antifuse Connections

Four types of antifuse connections are used in the routing structure of the 1200XL array. (The physical structure of the antifuse is identical in each case; only the usage differs.) The four types are:

Examples of all four antifuse connections are shown in Figures 6 and 7.

XF	Cross-connected antifuse	Most intersections of horizontal and vertical tracks have an XF that connects the perpendicular tracks.
HF	Horizontally connected antifuses	Adjacent segments in the same horizontal tracks are connected end-to-end by an HF.
VF	Vertically connected antifuse	Some long vertical tracks are divided into two segments. Adjacent long segments are connected end-to-end by a VF.
FF	"Fast-Fuse" antifuse	The FF connects a module output directly to a long vertical track.

Antifuse Programming

The 1200XL family uses the PLICE antifuse developed by Actel. The PLICE element is programmed by placing a high voltage (~17 V) across the element and supplying current (~5 mA) for a short duration (<1 ms). In the 1200XL architecture, most antifuses are programmed to ~500 ohms resistance, except for the F-fuses which are programmed to ~250 ohms. The programming circuits are transparent to the user.

Clock Networks

Two low-skew, high fanout clock distribution networks are provided in the 1200XL architecture (Figure 8). These networks are referred to as CLK0 and CLK1. Each network has a clock module (CLKMOD) that selects the source of the clock signal and may be driven as follows:

1. externally from the CLKA pad
2. externally from the CLKB pad
3. internally from the CLKINA input
4. internally from the CLKINB input

The clock modules are located in the top row of I/O modules. Clock drivers and a dedicated horizontal clock track are located in each horizontal routing channel.

The user controls the clock module by selecting one of two clock macros from the macro library. The macro CLKBUF is used to connect one of the two external clock pins to a clock network, and the macro CLKINT is used to connect an internally generated clock signal to a clock network. Since both clock networks are identical, the user does not care whether CLK0 or CLK1 is being used.

The clock input pads may also be used as normal I/Os, bypassing the clock networks.

Module Interface

Connections to logic and I/O modules are made through vertical segments that connect to the module inputs and

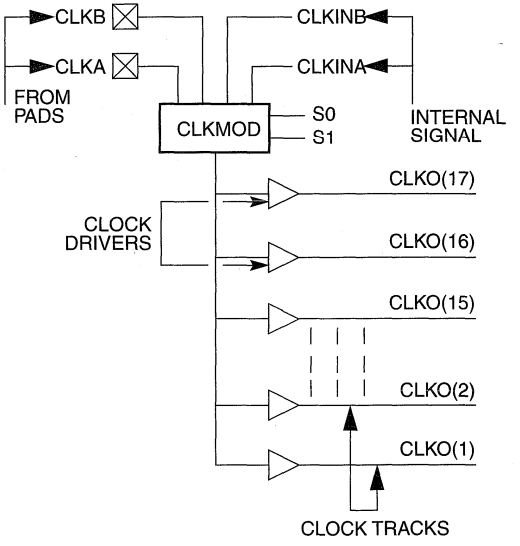


Figure 8 • Clock Networks

outputs. These vertical segments lie on vertical tracks that span the entire height of the array.

Module Input Connections

The tracks dedicated to Module inputs are segmented by pass transistors in each module row. During normal user operation, the pass transistors are inactive (off), which isolates the inputs of a module from the inputs of the module directly above or below it. During certain test modes, the pass transistors are active (on) to verify the continuity of the metal tracks. Vertical input segments span only one channel. Inputs to the array modules come either from the channel above or the channel below. The logic modules are arranged so that half of the inputs are connected to the channel above and half of the inputs to segments in the channel below (Figure 9).

Module Output Connections

Module outputs have dedicated output segments. Output segments extend vertically two channels above and two channels below, except at the top or bottom of the array. Output segments twist, as shown in Figure 9, so that only four vertical tracks are required.

LVT Connections

Outputs may also connect to nondedicated segments (LVTs). Each module-pair in the array shares three LVTs that span the length of column as shown in Figure 9. Any module in the column pair can connect to one of the LVTs in the column using an FF connection. The FF connection uses antifuses connected directly to the driver stage of the module output, bypassing the isolation transistor. FF antifuses are programmed at a higher current level than HF, VF, or XF antifuses to produce a lower resistance value.

Antifuse Connections

In general, every intersection of a vertical segment and a horizontal segment contains an unprogrammed antifuse (XF-type). One exception is in the case of the clock networks.

Clock Connections

To minimize loading on the clock networks, only a subset of inputs has fuses on the clock tracks. Only a few of the C-module and S-module inputs can be connected to the clock networks. To further reduce loading on the clock network, only a subset of the horizontal routing tracks can connect to the clock inputs of the S-module. Both of these are illustrated in Figure 10.

Programming and Test Circuits

The array of logic and I/O modules is surrounded by test and programming circuits controlled by the external pins: MODE, SDI, and DCLK. When MODE is low (GND), the device is in normal or user mode. When MODE is high (V_{CC}), the device is placed into one of several programming or test states. The SDI pin (when MODE is high) is used to input serial data to the Mode Register and various address registers surrounding the array. Data is clocked into these registers using the DCLK pin. The registers are connected as a long series of shift registers as shown in Figure 11. The Mode register determines the test or programming state of the device. Many of the test modes are used during wafer sort and final test at the factory. Other test modes are used during programming with the Activator® 2, and some of the modes are available only after programming. The Actionprobe® function is one such function available to users.

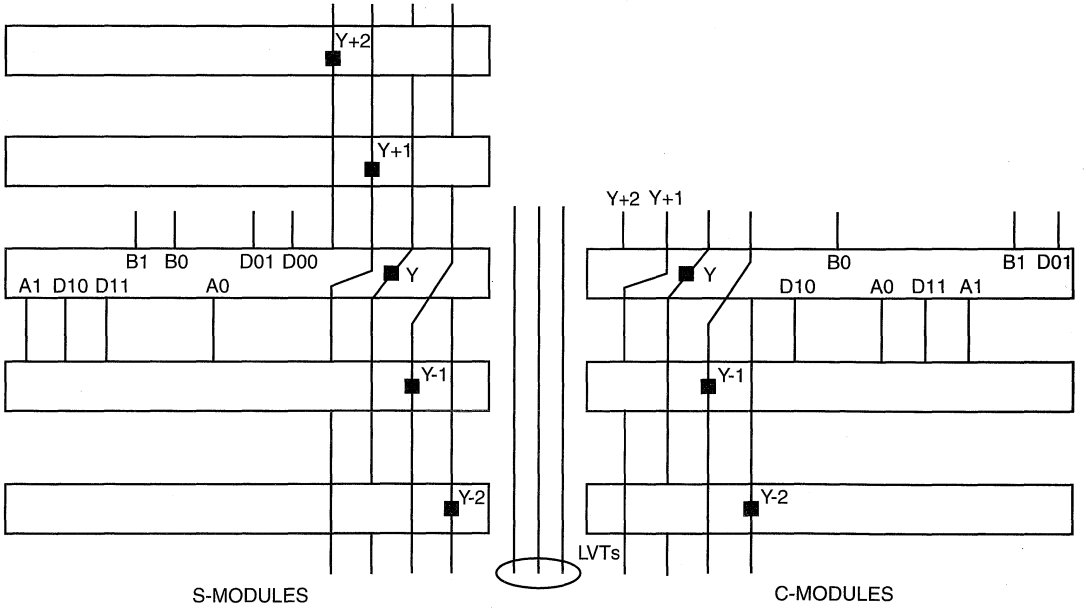


Figure 9 • Logic Module Routing Interface

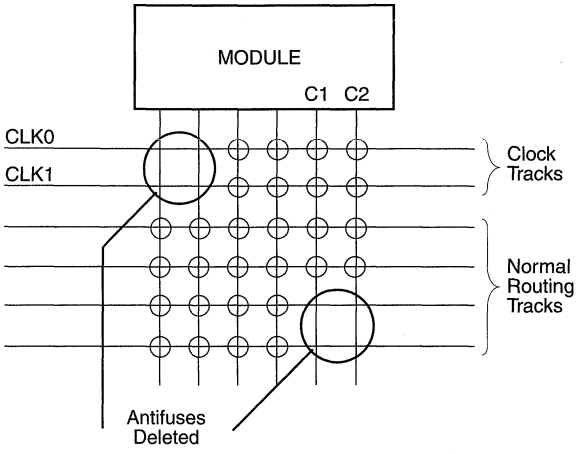


Figure 10 • Fuse Deletion on Clock Networks

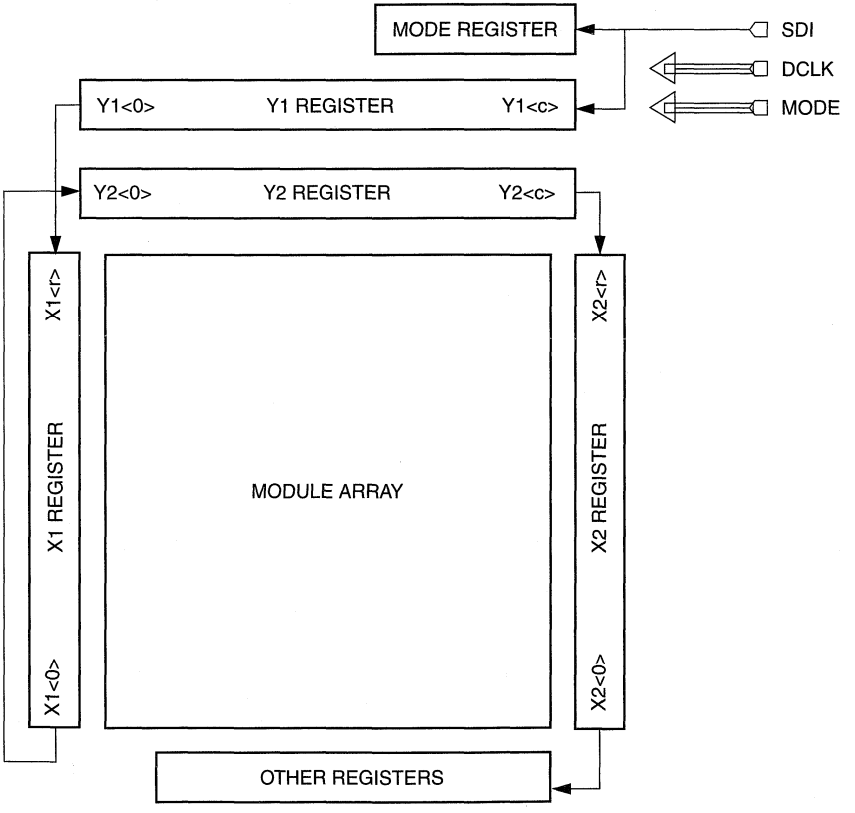


Figure 11 • 1200XL Shift Register

Actionprobe

If a device has been successfully programmed and the security fuse has not been programmed, any internal logic or I/O module output can be observed using the Actionprobe circuitry and the PRA and/or PRB pins. The Actionprobe diagnostic system provides the software and hardware required to perform real-time debugging. The software automatically performs the following functions.

A pattern of *ones* and *zeros* is shifted into the device from the SDI pin at each positive edge transition of DCLK. The complete sequence contains 10 bits of counter, 21 bits of Mode Register, n bits of zeros (filler of unused fields, where n depends on the particular device type), R bits of X2, C bits of Y2, R bits of X1, C bits of Y1, and a stop bit ("0" or "1"). After the stop bit has been shifted in, DCLK is left high. X1 and Y1 represent the (X,Y) location in the array for the Actionprobe output, PRA.

X2 and Y2 represent the (X,Y) location in the array for the Actionprobe output, PRB. R and C are the row and column size as defined in Table 1. The filler bits, counter pattern, and Mode Register pattern are shown in Table 3. Addressing for rows and columns is active high; that is, unselected rows and columns are "zeros" and the selected row and column is "high." The timing sequence is shown in Figure 12. The recommended frequency is 10 MHz with 10 ns setup and hold times allowing for SDI and DCLK transitions. The selected module output will be present at the PRA or PRB output approximately 20 ns after the stop-bit transition.

Table 3 • Bit Stream Definitions for Actionprobe Diagnostics

Device	Probe_Mode	Filler (n)	Counter_Pattern	Mode_Register_Pattern	# of clocks
A1225XL	Probe A only	308	1101011010	00000011000111111000000	458
A1225XL	Probe B only	308	1101011010	00000010100111111000000	458
A1225XL	Probe A and B	308	1101011010	00000011100111111000000	458
A1240XL	Probe A only	361	1111000001	00000011000111111000000	545
A1240XL	Probe B only	361	1111000001	00000010100111111000000	545
A1240XL	Probe A and B	361	1111000001	00000011100111111000000	545
A1280XL	Probe A only	443	0011011111	00000011000111111000000	675
A1280XL	Probe B only	443	0011011111	00000010100111111000000	675
A1280XL	Probe A and B	443	0011011111	00000011100111111000000	675

For example: Selecting PRA for A1280 results in the following bit stream.

0011011111_00000011000111111000000_

(433 zeros)_X2<0>...X2<17>_Y2<81>...Y2<0>_X1<0>...X1<0>...X1<17>_Y1<0>...Y1<81>_0,

where "_" is used for clarity only

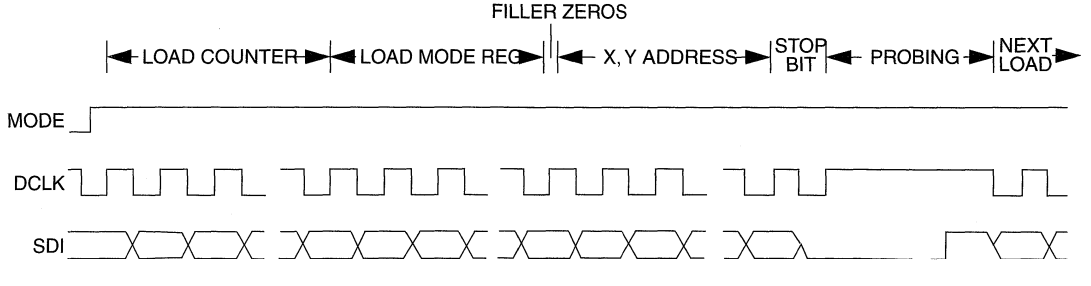


Figure 12 • Timing Waveforms

Absolute Maximum Ratings[1]

Free air temperature range

Symbol	Parameter	Limits	Units
V_{CC}	DC Supply Voltage[2,3,4]	–0.5 to +7.0	V
V_I	Input Voltage	–0.5 to V_{CC} +0.5	V
V_O	Output Voltage	–0.5 to V_{CC} +0.5	V
I_{IO}	I/O Source/Sink Current[5]	±20	mA
T_{STG}	Storage Temperature	–65 to +150	°C

Notes:

1. *Stresses beyond those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. Exposure to absolute maximum rated conditions for extended periods may affect device reliability. Device should not be operated outside the Recommended Operating Conditions.*
2. *$V_{PP} = V_{CC}$, except during device programming.*
3. *$V_{SV} = V_{CC}$, except during device programming.*
4. *V_{KS} = GND, except during device programming.*
5. *Device inputs are normally high impedance and draw extremely low current. However, when input voltage is greater than V_{CC} + 0.5 V or less than GND – 0.5 V, the internal protection diode will be forward biased and can draw excessive current.*

Recommended Operating Conditions

Parameter	Commercial	Industrial	Military	Units
Temperature Range[1]	0 to +70	–40 to +85	–55 to +125	°C
Power Supply Tolerance	±5	±10	±10	%V_{CC}

Note:

1. *Ambient temperature (T_A) is used for commercial and industrial; case temperature (T_C) is used for military.*

Electrical Specifications

Symbol	Parameter	Commercial		Industrial		Military		Units
		Min.	Max.	Min.	Max.	Min.	Max.	
V_{OH}[1]	(I_{OH} = –10 mA)[2]	2.4						V
	(I_{OH} = –6 mA)	3.84						V
	(I_{OH} = –4 mA)			3.7		3.7		V
V_{OL}[1]	(I_{OL} = 10 mA)[2]		0.5					V
	(I_{OL} = 6 mA)		0.33		0.40		0.40	V
V_{IL}		–0.3	0.8	–0.3	0.8	–0.3	0.8	V
V_{IH}		2.0	V_{CC} + 0.3	2.0	V_{CC} + 0.3	2.0	V_{CC} + 0.3	V
Input Transition Time t_R, t_F[2]			500		500		500	ns
C_{IO} I/O Capacitance[2, 3]			10		10		10	pF
Standby Current, I_{CC}[4] (typical = 1 mA)			2		10		20	mA
Leakage Current[5]		–10	10	–10	10	–10	10	µA

Notes:

1. *Only one output tested at a time. V_{CC} = min.*
2. *Not tested, for information only.*
3. *Includes worst-case 176 CPGA package capacitance. V_{OUT} = 0 V, f = 1 MHz.*
4. *All outputs unloaded. All inputs = V_{CC} or GND, typical I_{CC} = 1 mA. I_{CC} limit includes I_{PP} and I_{SV} during normal operation.*
5. *V_{OUT}, V_{IN} = V_{CC} or GND.*

Package Thermal Characteristics

The device junction to case thermal characteristic is θjc, and the junction to ambient air characteristic is θja. The thermal characteristics for θja are shown with two different air flow rates.

Maximum junction temperature is 150°C.

A sample calculation of the absolute maximum power dissipation allowed for a PQFP 160-pin package at commercial temperature is as follows:

$$\frac{\text{Max. junction temp. (°C)} - \text{Max. commercial temp.}}{\theta ja\ (°C/W)} = \frac{150°C - 70°C}{30°C/W} = 2.6W$$

Package Type	Pin Count	θja Still Air	θja 300 ft/min	Units
Ceramic Pin Grid Array	100	35	17	°C/W
	132	30	15	°C/W
	176	23	12	°C/W
Plastic Quad Flatpack[1]	100	51	40	°C/W
	144	36	30	°C/W
	160	33	26	°C/W
Plastic Leaded Chip Carrier[2]	84	37	28	°C/W
Very Thin Quad Flatpack[3]	100	43	35	°C/W
Thin Quad Flatpack[4]	176	32	25	°C/W

Notes:

1. *Maximum Power Dissipation for PQFP packages are 1.6 Watts (100-pin), 2.2 Watts (144-pin), and 2.4 Watts (160-pin).*
2. *Maximum Power Dissipation for PLCC packages is 2.2 Watts.*
3. *Maximum Power Dissipation for VQFP packages is 1.9 Watts.*
4. *Maximum Power Dissipation for TQFP packages is 2.5 Watts.*

General Power Equation

$$P = [I_{CC}standby + I_{CC}active] * V_{CC} + I_{OL} * V_{OL} * N + I_{OH} * (V_{CC} - V_{OH}) * M$$

Where:

I_{CC}standby is the current flowing when no inputs or outputs are changing.

I_{CC}active is the current flowing due to CMOS switching.

I_{OL}, I_{OH} are TTL sink/source currents.

V_{OL}, V_{OH} are TTL level output voltages.

N equals the number of outputs driving TTL loads to V_{OL}.

M equals the number of outputs driving TTL loads to V_{OH}.

An accurate determination of N and M is problematic because their values depend on the family type, design details, and on the system I/O. The power can be divided into two components: static and active.

Static Power Component

Actel FPGAs have small static power components that result in lower power dissipation than PALs or PLDs. By integrating multiple PALs/PLDs into one FPGA, an even greater reduction in board-level power dissipation can be achieved.

The power due to standby current is typically a small component of the overall power. Standby power is calculated below for commercial, worst case conditions.

I_{CC}	V_{CC}	Power
2 mA	5.25 V	10.5 mW

The static power dissipation by TTL loads depends on the number of outputs driving high or low and the DC load current. Again, this number is typically small. For instance, a 32-bit bus sinking 4 mA at 0.33 V will generate 42 mW with all outputs driving low and 140 mW with all outputs driving high. The actual dissipation will average somewhere between as I/Os switch states with time.

Active Power Component

Power dissipation in CMOS devices is usually dominated by the active (dynamic) power dissipation. This component is frequency dependent, a function of the logic and the external I/O. Active power dissipation results from charging internal chip capacitances of the interconnect, unprogrammed antifuses, module inputs, and module outputs, plus external capacitance due to PC board traces and load device inputs. An additional component of the active power dissipation is the totem-pole current in the CMOS transistor pairs. The net effect can be associated with an equivalent capacitance that can be combined with frequency and voltage to represent active power dissipation.

Equivalent Capacitance

The power dissipated by a CMOS circuit can be expressed by Equation 1.

$$\text{Power } (\mu W) = C_{EQ} * V_{CC}^2 * F \quad (1)$$

Where:

C_{EQ} is the equivalent capacitance expressed in picofarads (pF).

V_{CC} is power supply in volts (V).

F is the switching frequency in megahertz (MHz).

Equivalent capacitance is calculated by measuring ICCactive at a specified frequency and voltage for each circuit component of interest. Measurements have been made over a range of frequencies at a fixed value of VCC. Equivalent capacitance is frequency independent so that the results may be used over a wide range of operating conditions. Equivalent capacitance values are shown below.

C_{EQ} Values for Actel FPGAs

Modules (C_{EQM})	5.2
Input Buffers (C_{EQI})	11.6
Output Buffers (C_{EQO})	23.8
Routed Array Clock Buffer Loads (C_{EQCR})	3.5

To calculate the active power dissipated from the complete design, the switching frequency of each part of the logic must be known. Equation 2 shows a piece-wise linear summation over all components.

$$\text{Power} = V_{CC}^2 * [(m \times C_{EQM} * f_m)_{Modules} + (n * C_{EQI} * f_n)_{Inputs} + (p * (C_{EQO} + C_L) * f_p)_{outputs} + 0.5 * (q_1 * C_{EQCR} * f_{q1})_{routed\_Clk1} + (r_1 * f_{q1})_{routed\_Clk1} + 0.5 * (q_2 * C_{EQCR} * f_{q2})_{routed\_Clk2} + (r_2 * f_{q2})_{routed\_Clk2} \quad (2)$$

Where:

m = Number of logic modules switching at frequency f_m

n = Number of input buffers switching at frequency f_n

p = Number of output buffers switching at frequency f_p

q_1 = Number of clock loads on the first routed array clock

q_2 = Number of clock loads on the second routed array clock

r_1 = Fixed capacitance due to first routed array clock

r_2 = Fixed capacitance due to second routed array clock

C_{EQM} = Equivalent capacitance of logic modules in pF

C_{EQI} = Equivalent capacitance of input buffers in pF

C_{EQO} = Equivalent capacitance of output buffers in pF

C_{EQCR} = Equivalent capacitance of routed array clock in pF

C_L = Output load capacitance in pF

f_m = Average logic module switching rate in MHz

f_n = Average input buffer switching rate in MHz

f_p = Average output buffer switching rate in MHz

f_{q1} = Average first routed array clock rate in MHz

f_{q2} = Average second routed array clock rate in MHz

Fixed Capacitance Values for Actel FPGAs (pF)

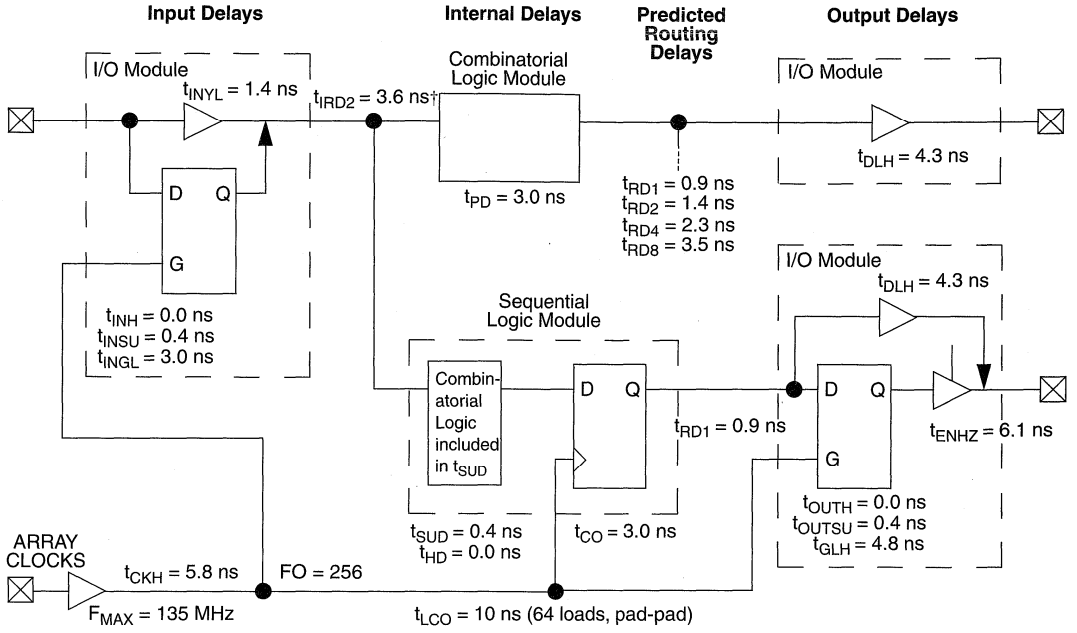
Device Type	r_1 routed_Clk1	r_2 routed_Clk2
A1225XL	106	106
A1240XL	134	134
A1280XL	168	168

Determining Average Switching Frequency

To determine the switching frequency for a design, you must have a detailed understanding of the data input values to the circuit. The following guidelines are meant to represent worst-case scenarios so that they can be generally used to predict the upper limits of power dissipation. These guidelines are as follows:

Logic Modules (m)	= 80% of combinatorial modules
Inputs switching (n)	= # of inputs/4
Outputs switching (p)	= # outputs/4
First routed array clock loads (q_1)	= 40% of sequential modules
Second routed array clock loads (q_2)	= 40% of sequential modules
Load capacitance (C_L)	= 35 pF
Average logic module switching rate (f_m)	= F/10
Average input switching rate (f_n)	= F/5
Average output switching rate (f_p)	= F/10
Average first routed array clock rate (f_{q1})	= F
Average second routed array clock rate (f_{q2})	= F/2

1200XL Timing Model*

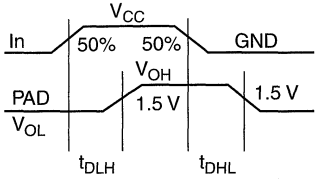


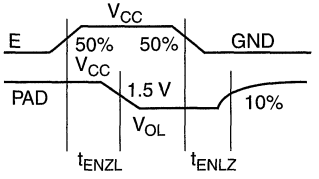
**Values shown for A1225XL-1 at worst-case commercial conditions.*

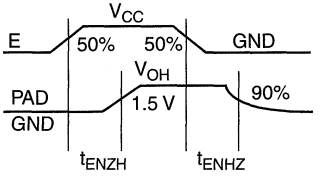
† Input Module Predicted Routing Delay

Parameter Measurement

Output Buffer Delays

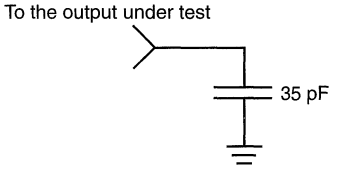




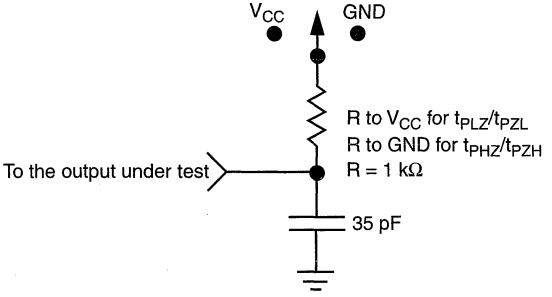


AC Test Loads

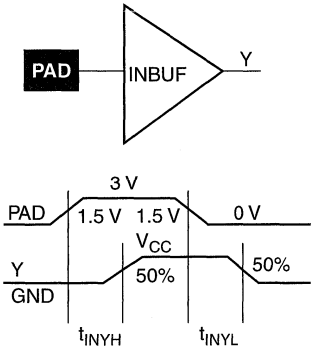
Load 1
(Used to measure propagation delay)



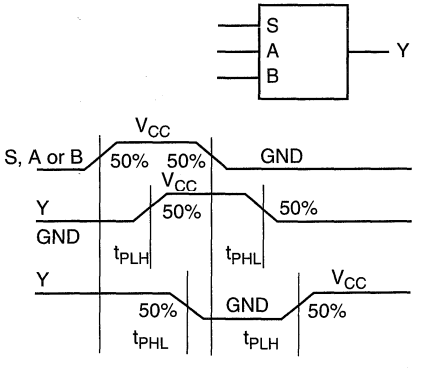
Load 2
(Used to measure rising/falling edges)



Input Buffer Delays

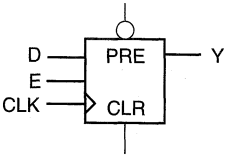


Module Delays

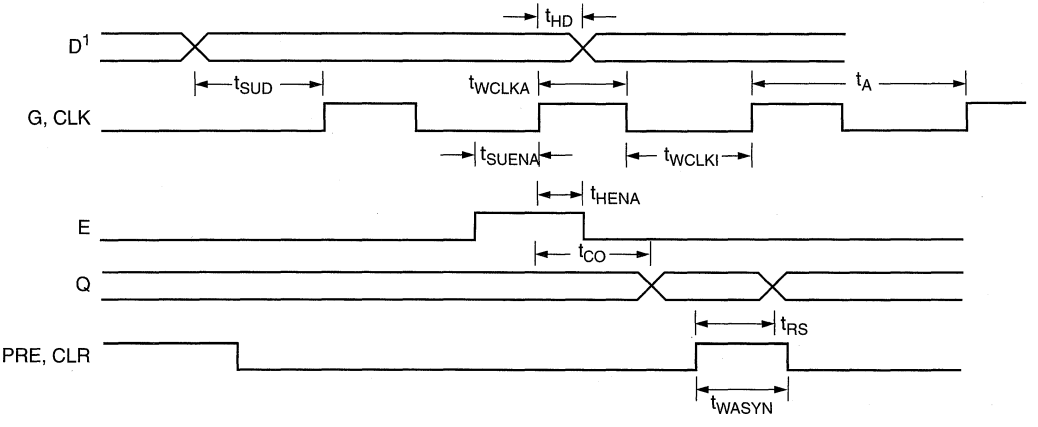


Sequential Module Timing Characteristics

Flip-Flops and Latches



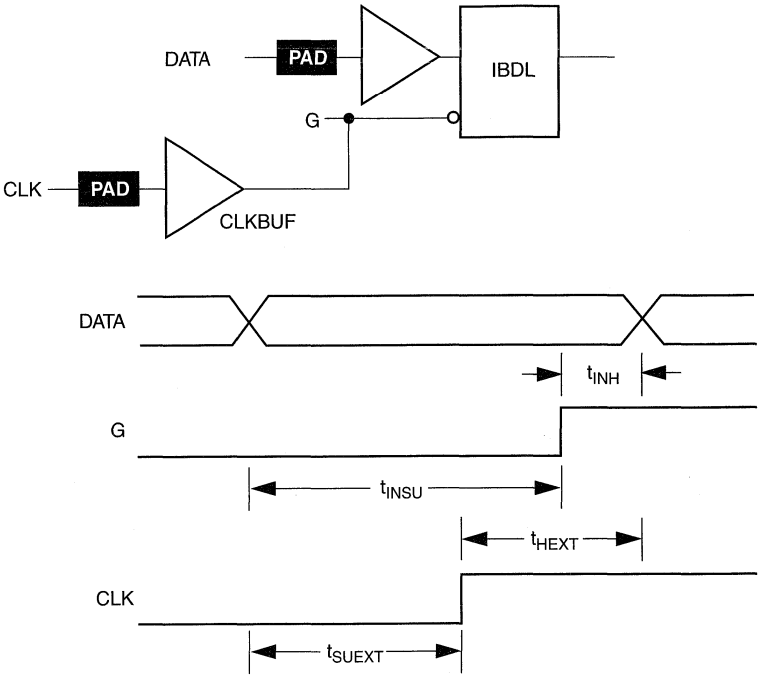
(Positive edge triggered)



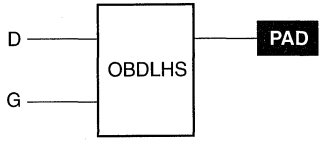
Note: *D represents all data functions involving A, B, and S for multiplexed flip-flops.*

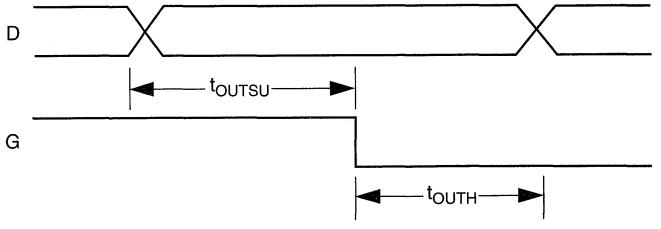
Sequential Timing Characteristics (continued)

Input Buffer Latches



Output Buffer Latches





Predictable Performance: Tight Delay Distributions

Propagation delay between logic modules depends on the resistive and capacitive loading of the routing tracks, the interconnect elements, and the module inputs being driven. Propagation delay increases as the length of routing tracks, the number of interconnect elements, or the number of inputs increases.

From a design perspective, the propagation delay can be statistically correlated or modeled by the fanout (number of loads) driven by a module. Higher fanout usually requires some paths to have longer routing tracks.

The 1200XL family delivers a very tight fanout delay distribution. This tight distribution is achieved in two ways: by decreasing the delay of the interconnect elements and by decreasing the number of interconnect elements per path.

Actel's patented PLICE antifuse offers a very low resistive/capacitive interconnect. The 1200XL family's antifuses, fabricated in 0.6 micron lithography, offer nominal levels of 100 ohms resistance and 7.0 femtofarad (fF) capacitance per antifuse.

The 1200XL fanout distribution is also tight due to the low number of antifuses required for each interconnect path. The 1200XL family's proprietary architecture limits the number of antifuses per path to a maximum of four, with 90% of interconnects using two antifuses.

Table 4 • *Logic Module + Routing Delay, by Fanout (ns)*
(Worst-Case Commercial Conditions)

Family	FO=1	FO=2	FO=3	FO=4	FO=8
A1225XL-1	3.9	4.4	4.8	5.3	6.5
A1240XL-1	4.2	4.4	4.9	5.6	6.8
A1280XL-1	4.4	5.0	5.5	6.0	8.7

Timing Characteristics

Timing characteristics for 1200XL devices fall into three categories: family dependent, device dependent, and design dependent. The input and output buffer characteristics are common to all 1200XL family members. Internal routing delays are device dependent. Design dependency means actual delays are not determined until after placement and routing of the user's design is complete. Delay values may then be determined by using the ALS Timer utility or performing simulation with post-layout delays.

Critical Nets and Typical Nets

Propagation delays are expressed only for typical nets, which are used for initial design performance evaluation. Critical net delays can then be applied to the most time-critical paths. Critical nets are determined by net property assignment prior to placement and routing. Up to 6% of the nets in a design may be designated as critical, while 90% of the nets in a design are typical.

Long Tracks

Some nets in the design use long tracks. Long tracks are special routing resources that span multiple rows, columns, or modules. Long tracks employ three and sometimes four antifuse connections. This increases capacitance and resistance, resulting in longer net delays for macros connected to long tracks. Typically, up to 6% of nets in a fully utilized device require long tracks. Long tracks contribute approximately 4 ns to 8 ns delay. This additional delay is represented statistically in higher fanout (FO=8) routing delays in the data sheet specifications section.

Timing Derating

A best case timing derating factor of 0.45 is used to reflect best case processing. Note that this factor is relative to the "standard speed" timing parameters, and must be multiplied by the appropriate voltage and temperature derating factors for a given application.

Timing Derating Factor (Temperature and Voltage)

	Industrial		Military	
	Min.	Max.	Min.	Max.
(Commercial Minimum/Maximum Specification) x	0.69	1.11	0.67	1.23

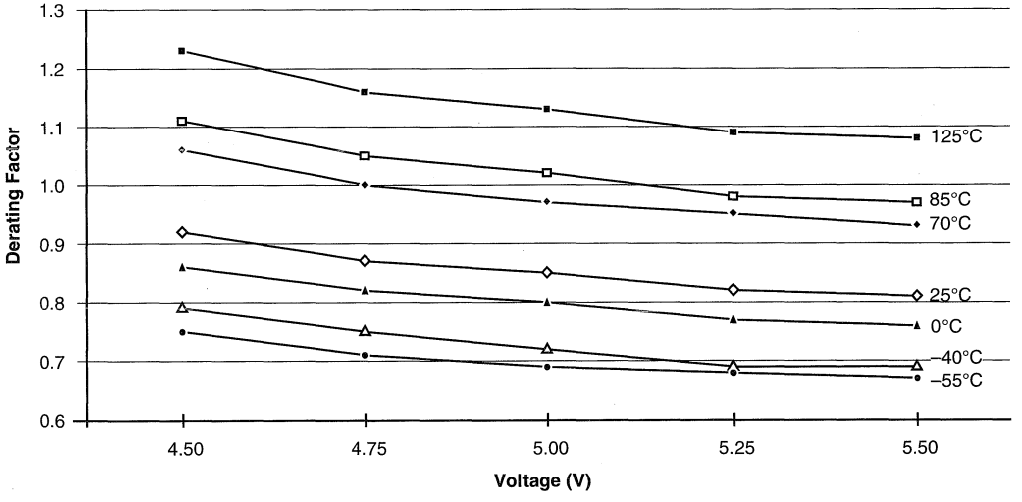
Timing Derating Factor for Designs at Typical Temperature (T_J = 25°C) and Voltage (5.0 V)

(Commercial Maximum Specification) x	0.85

Temperature and Voltage Derating Factors (normalized to Worst-Case Commercial, T_J = 4.75 V, 70°C)

	–55	–40	0	25	70	85	125
4.50	0.75	0.79	0.86	0.92	1.06	1.11	1.23
4.75	0.71	0.75	0.82	0.87	1.00	1.05	1.16
5.00	0.69	0.72	0.80	0.85	0.97	1.02	1.13
5.25	0.68	0.69	0.77	0.82	0.95	0.98	1.09
5.50	0.67	0.69	0.76	0.81	0.93	0.97	1.08

Junction Temperature and Voltage Derating Curves (normalized to Worst-Case Commercial, T_J = 4.75 V, 70°C)



Note: *This derating factor applies to all routing and propagation delays.*

A1225XL Timing Characteristics

(Worst-Case Commercial Conditions, V_{CC} = 4.75 V, T_J = 70°C)

Logic Module Propagation Delays[1]		'Std' Speed		'–1' Speed		
Parameter	**Description**	**Min.**	**Max.**	**Min.**	**Max.**	**Units**
t_{PD1}	Single Module		3.5		3.0	ns
t_{CO}	Sequential Clk to Q		3.5		3.0	ns
t_{GO}	Latch G to Q		3.5		3.0	ns
t_{RS}	Flip-Flop (Latch) Reset to Q		3.5		3.0	ns
Predicted Routing Delays[2]						
t_{RD1}	FO=1 Routing Delay		1.1		0.9	ns
t_{RD2}	FO=2 Routing Delay		1.7		1.4	ns
t_{RD3}	FO=3 Routing Delay		2.2		1.8	ns
t_{RD4}	FO=4 Routing Delay		2.7		2.3	ns
t_{RD8}	FO=8 Routing Delay		4.2		3.5	ns
Sequential Timing Characteristics[3,4]						
t_{SUD}	Flip-Flop (Latch) Data Input Setup	0.5		0.4		ns
t_{HD}	Flip-Flop (Latch) Data Input Hold	0.0		0.0		ns
t_{SUENA}	Flip-Flop (Latch) Enable Setup	1.0		0.9		ns
t_{HENA}	Flip-Flop (Latch) Enable Hold	0.0		0.0		ns
t_{WCLKA}	Flip-Flop (Latch) Clock Active Pulse Width	4.3		3.6		ns
t_{WASYN}	Flip-Flop (Latch) Asynchronous Pulse Width	4.3		3.6		ns
t_A	Flip-Flop Clock Input Period	8.7		7.4		ns
t_{INH}	Input Buffer Latch Hold	0.0		0.0		ns
t_{INSU}	Input Buffer Latch Setup	0.4		0.4		ns
t_{OUTH}	Output Buffer Latch Hold	0.0		0.0		ns
t_{OUTSU}	Output Buffer Latch Setup	0.4		0.4		ns
f_{MAX}	Flip-Flop (Latch) Clock Frequency		115		135	MHz

Notes:

1. *For dual-module macros, use $t_{PD1} + t_{RD1} + t_{PDm}$, $t_{CO} + t_{RD1} + t_{PDn}$ or $t_{PD1} + t_{RD1} + t_{SUD}$, whichever is appropriate.*
2. *Routing delays are for typical designs across worst-case operating conditions. These parameters should be used for estimating device performance. Post-route timing analysis or simulation is required to determine actual worst-case performance. Post-route timing is based on actual routing delay measurements performed on the device prior to shipment.*
3. *Data applies to macros based on the S-module. Timing parameters for sequential macros constructed from C-modules can be obtained from the ALS Timer utility.*
4. *Setup and hold timing parameters for the Input Buffer Latch are defined with respect to the PAD and the D input. External setup/hold timing parameters must account for delay from an external PAD signal to the G inputs. Delay from an external PAD signal to the G input subtracts (adds) to the internal setup (hold) time.*

A1225XL Timing Characteristics (continued)

(Worst-Case Commercial Conditions)

Input Module Propagation Delays			'Std' Speed		'–1' Speed		
Parameter	**Description**		**Min.**	**Max.**	**Min.**	**Max.**	**Units**
t_{INYH}	Pad to Y High			1.4		1.2	ns
t_{INYL}	Pad to Y Low			1.7		1.4	ns
t_{INGH}	G to Y High			2.7		2.3	ns
t_{INGL}	G to Y Low			3.5		3.0	ns
Input Module Predicted Routing Delays[1]							
t_{IRD1}	FO=1 Routing Delay			3.9		3.3	ns
t_{IRD2}	FO=2 Routing Delay			4.3		3.6	ns
t_{IRD3}	FO=3 Routing Delay			5.0		4.2	ns
t_{IRD4}	FO=4 Routing Delay			5.4		4.6	ns
t_{IRD8}	FO=8 Routing Delay			6.9		5.9	ns
Global Clock Network							
t_{CKH}	Input Low to High	FO = 32 FO = 256		6.8 7.6		5.8 6.5	ns
t_{CKL}	Input High to Low	FO = 32 FO = 256		6.7 7.6		5.7 6.5	ns
t_{PWH}	Minimum Pulse Width High	FO = 32 FO = 256	3.5 3.6		3.0 3.1		ns
t_{PWL}	Minimum Pulse Width Low	FO = 32 FO = 256	3.5 3.6		3.0 3.1		ns
t_{CKSW}	Maximum Skew	FO = 32 FO = 256		1.0 1.0		0.9 0.9	ns
t_{SUEXT}	Input Latch External Setup	FO = 32 FO = 256	0.0 0.0		0.0 0.0		ns
t_{HEXT}	Input Latch External Hold	FO = 32 FO = 256	3.4 4.3		2.9 3.7		ns
t_P	Minimum Period	FO = 32 FO = 256	7.2 7.4		6.1 6.3		ns
f_{MAX}	Maximum Frequency	FO = 32 FO = 256		140.0 135.0		165.0 160.0	MHz

Note:

1. *These parameters should be used for estimating device performance. Optimization techniques may further reduce delays by 0 to 3 ns. Routing delays are for typical designs across worst-case operating conditions. Post-route timing analysis or simulation is required to determine actual worst-case performance. Post-route timing is based on actual routing delay measurements performed on the device prior to shipment.*

A1225XL Timing Characteristics (continued)

(Worst-Case Commercial Conditions)

Output Module Timing		'Std' Speed		'–1' Speed		
Parameter	**Description**	**Min.**	**Max.**	**Min.**	**Max.**	**Units**
TTL Output Module **Timing**[1, 2]						
t_{DLH}	Data to Pad High		5.0		4.3	ns
t_{DHL}	Data to Pad Low		5.4		4.6	ns
t_{ENZH}	Enable Pad Z to High		5.0		4.3	ns
t_{ENZL}	Enable Pad Z to Low		5.5		4.7	ns
t_{ENHZ}	Enable Pad High to Z		7.2		6.1	ns
t_{ENLZ}	Enable Pad Low to Z		7.2		6.1	ns
t_{GLH}	G to Pad High		5.6		4.8	ns
t_{GHL}	G to Pad Low		6.3		5.4	ns
t_{LCO}	I/O Latch Clock-Out (pad-to-pad), 64 clock loading		12.0		10.0	ns
t_{ACO}	Array Clock-Out (pad-to-pad), 64 clock loading		17.0		14.4	ns
d_{TLH}	Capacitive Loading, Low to High		0.05		0.04	ns/pF
d_{THL}	Capacitive Loading, High to Low		0.07		0.06	ns/pF
CMOS Output Module **Timing**[1, 2]						
t_{DLH}	Data to Pad High		6.4		5.4	ns
t_{DHL}	Data to Pad Low		4.5		3.8	ns
t_{ENZH}	Enable Pad Z to High		5.0		4.3	ns
t_{ENZL}	Enable Pad Z to Low		5.5		4.7	ns
t_{ENHZ}	Enable Pad High to Z		7.2		6.1	ns
t_{ENLZ}	Enable Pad Low to Z		7.2		6.1	ns
t_{GLH}	G to Pad High		5.6		4.8	ns
t_{GHL}	G to Pad Low		6.3		5.4	ns
t_{LCO}	I/O Latch Clock-Out (pad-to-pad), 64 clock loading		14.2		11.8	ns
t_{ACO}	Array Clock-Out (pad-to-pad), 64 clock loading		20.0		17.0	ns
d_{TLH}	Capacitive Loading, Low to High		0.07		0.06	ns/pF
d_{THL}	Capacitive Loading, High to Low		0.06		0.05	ns/pF

Notes:

1. *Delays based on 35 pF loading.*
2. *Maximum Recommended Simultaneous Switching Outputs:*

PLCC	*20pF*	*72*
	35pF	*45*
	50pF	*32*
PQFP, CPGA	*20pF*	*80*
	35pF	*45*
	50pF	*32*

A1240XL Timing Characteristics

(Worst-Case Commercial Conditions, V_{CC} = 4.75 V, T_J = 70°C)

Logic Module Propagation Delays[1]		'Std' Speed		'–1' Speed		
Parameter	**Description**	**Min.**	**Max.**	**Min.**	**Max.**	**Units**
t_{PD1}	Single Module		3.5		3.0	ns
t_{CO}	Sequential Clk to Q		3.5		3.0	ns
t_{GO}	Latch G to Q		3.5		3.0	ns
t_{RS}	Flip-Flop (Latch) Reset to Q		3.5		3.0	ns
Predicted Routing Delays[2]						
t_{RD1}	FO=1 Routing Delay		1.4		1.2	ns
t_{RD2}	FO=2 Routing Delay		1.7		1.4	ns
t_{RD3}	FO=3 Routing Delay		2.2		1.9	ns
t_{RD4}	FO=4 Routing Delay		3.0		2.6	ns
t_{RD8}	FO=8 Routing Delay		4.5		3.8	ns
Sequential Timing Characteristics[3, 4]						
t_{SUD}	Flip-Flop (Latch) Data Input Setup	0.5		0.4		ns
t_{HD}	Flip-Flop (Latch) Data Input Hold	0.0		0.0		ns
t_{SUENA}	Flip-Flop (Latch) Enable Setup	1.0		0.9		ns
t_{HENA}	Flip-Flop (Latch) Enable Hold	0.0		0.0		ns
t_{WCLKA}	Flip-Flop (Latch) Clock Active Pulse Width	4.5		3.8		ns
t_{WASYN}	Flip-Flop (Latch) Asynchronous Pulse Width	4.5		3.8		ns
t_A	Flip-Flop Clock Input Period	9.1		7.7		ns
t_{INH}	Input Buffer Latch Hold	0.0		0.0		ns
t_{INSU}	Input Buffer Latch Setup	0.4		0.4		ns
t_{OUTH}	Output Buffer Latch Hold	0.0		0.0		ns
t_{OUTSU}	Output Buffer Latch Setup	0.4		0.4		ns
f_{MAX}	Flip-Flop (Latch) Clock Frequency		110.0		130.0	MHz

Notes:

1. *For dual-module macros, use $t_{PD1} + t_{RD1} + t_{PDn}$, $t_{CO} + t_{RD1} + t_{PDn}$ or $t_{PD1} + t_{RD1} + t_{SUD}$, whichever is appropriate.*
2. *Routing delays are for typical designs across worst-case operating conditions. These parameters should be used for estimating device performance. Post-route timing analysis or simulation is required to determine actual worst-case performance. Post-route timing is based on actual routing delay measurements performed on the device prior to shipment.*
3. *Data applies to macros based on the S-module. Timing parameters for sequential macros constructed from C-modules can be obtained from the ALS Timer utility.*
4. *Setup and hold timing parameters for the Input Buffer Latch are defined with respect to the PAD and the D input. External setup/hold timing parameters must account for delay from an external PAD signal to the G inputs. Delay from an external PAD signal to the G input subtracts (adds) to the internal setup (hold) time.*

A1240XL Timing Characteristics (continued)

(Worst-Case Commercial Conditions)

Input Module Propagation Delays			'Std' Speed		'–1' Speed		
Parameter	**Description**		**Min.**	**Max.**	**Min.**	**Max.**	**Units**
t_{INYH}	Pad to Y High			1.4		1.2	ns
t_{INYL}	Pad to Y Low			1.7		1.4	ns
t_{INGH}	G to Y High			2.7		2.3	ns
t_{INGL}	G to Y Low			3.5		3.0	ns
Input Module Predicted Routing Delays[1]							
t_{IRD1}	FO=1 Routing Delay			3.9		3.3	ns
t_{IRD2}	FO=2 Routing Delay			4.5		3.8	ns
t_{IRD3}	FO=3 Routing Delay			5.1		4.3	ns
t_{IRD4}	FO=4 Routing Delay			5.5		4.7	ns
t_{IRD8}	FO=8 Routing Delay			7.4		6.3	ns
Global Clock Network							
t_{CKH}	Input Low to High	FO = 32 FO = 256		6.8 7.6		5.8 6.5	ns
t_{CKL}	Input High to Low	FO = 32 FO = 256		6.7 7.6		5.7 6.5	ns
t_{PWH}	Minimum Pulse Width High	FO = 32 FO = 256	3.6 3.9		3.1 3.3		ns
t_{PWL}	Minimum Pulse Width Low	FO = 32 FO = 256	3.6 3.9		3.1 3.3		ns
t_{CKSW}	Maximum Skew	FO = 32 FO = 256		1.0 1.0		0.9 0.9	ns
t_{SUEXT}	Input Latch External Setup	FO = 32 FO = 256	0.0 0.0		0.0 0.0		ns
t_{HEXT}	Input Latch External Hold	FO = 32 FO = 256	3.4 4.3		2.9 3.7		ns
t_P	Minimum Period	FO = 32 FO = 256	7.4 8.0		6.3 6.8		ns
f_{MAX}	Maximum Frequency	FO = 32 FO = 256		135.0 125.0		160.0 150.0	MHz

Note:

1. *These parameters should be used for estimating device performance. Optimization techniques may further reduce delays by 0 to 3 ns. Routing delays are for typical designs across worst-case operating conditions. Post-route timing analysis or simulation is required to determine actual worst-case*

A1240XL Timing Characteristics (continued)

(Worst-Case Commercial Conditions)

Output Module Timing		'Std' Speed		'–1' Speed		
Parameter	**Description**	**Min.**	**Max.**	**Min.**	**Max.**	**Units**
TTL Output Module **Timing**[1, 2]						
t_{DLH}	Data to Pad High		5.0		4.3	ns
t_{DHL}	Data to Pad Low		5.4		4.6	ns
t_{ENZH}	Enable Pad Z to High		5.0		4.3	ns
t_{ENZL}	Enable Pad Z to Low		5.5		4.7	ns
t_{ENHZ}	Enable Pad High to Z		7.2		6.1	ns
t_{ENLZ}	Enable Pad Low to Z		7.2		6.1	ns
t_{GLH}	G to Pad High		5.6		4.8	ns
t_{GHL}	G to Pad Low		6.3		5.4	ns
t_{LCO}	I/O Latch Clock-Out (pad-to-pad), 64 clock loading		12.3		10.5	ns
t_{ACO}	Array Clock-Out (pad-to-pad), 64 clock loading		17.2		14.6	ns
d_{TLH}	Capacity Loading, Low to High		0.05		0.04	ns/pF
d_{THL}	Capacity Loading, High to Low		0.07		0.06	ns/pF
CMOS Output Module **Timing**[1, 2]						
t_{DLH}	Data to Pad High		6.4		5.4	ns
t_{DHL}	Data to Pad Low		4.5		3.8	ns
t_{ENZH}	Enable Pad Z to High		5.0		4.3	ns
t_{ENZL}	Enable Pad Z to Low		5.5		4.7	ns
t_{ENHZ}	Enable Pad High to Z		7.2		6.1	ns
t_{ENLZ}	Enable Pad Low to Z		7.2		6.1	ns
t_{GLH}	G to Pad High		5.6		4.8	ns
t_{GHL}	G to Pad Low		6.3		5.4	ns
t_{LCO}	I/O Latch Clock-Out (pad-to-pad), 64 clock loading		14.5		12.4	ns
t_{ACO}	Array Clock-Out (pad-to-pad), 64 clock loading		20.3		17.2	ns
d_{TLH}	Capacity Loading, Low to High		0.07		0.06	ns/pF
d_{THL}	Capacity Loading, High to Low		0.06		0.05	ns/pF

Notes:

1. *Delays based on 35 pF loading.*
2. *Maximum Recommended Simultaneous Switching Outputs:*

PLCC	*20pF*	*72*
	35pF	*45*
	50pF	*32*
PQFP, CPGA	*20pF*	*104*
	35pF	*68*
	50pF	*48*

A1280XL Timing Characteristics (continued)

(Worst-Case Commercial Conditions, V_{CC} = 4.75 V, T_J = 70°C)

Logic Module Propagation Delays[1]		'Std' Speed		'–1' Speed		
Parameter	**Description**	**Min.**	**Max.**	**Min.**	**Max.**	**Units**
t_{PD1}	Single Module		3.5		3.0	ns
t_{CO}	Sequential Clk to Q		3.5		3.0	ns
t_{GO}	Latch G to Q		3.5		3.0	ns
t_{RS}	Flip-Flop (Latch) Reset to Q		3.5		3.0	ns
Predicted Routing Delays[2]						
t_{RD1}	FO=1 Routing Delay		1.7		1.4	ns
t_{RD2}	FO=2 Routing Delay		2.4		2.0	ns
t_{RD3}	FO=3 Routing Delay		2.9		2.5	ns
t_{RD4}	FO=4 Routing Delay		3.5		3.0	ns
t_{RD8}	FO=8 Routing Delay		6.7		5.7	ns
Sequential Timing Characteristics[3,4]						
t_{SUD}	Flip-Flop (Latch) Data Input Setup	0.5		0.4		ns
t_{HD}	Flip-Flop (Latch) Data Input Hold	0.0		0.0		ns
t_{SUENA}	Flip-Flop (Latch) Enable Setup	1.0		0.9		ns
t_{HENA}	Flip-Flop (Latch) Enable Hold	0.0		0.0		ns
t_{WCLKA}	Flip-Flop (Latch) Clock Active Pulse Width	4.9		4.3		ns
t_{WASYN}	Flip-Flop (Latch) Asynchronous Pulse Width	4.9		4.3		ns
t_A	Flip-Flop Clock Input Period	10		8.7		ns
t_{INH}	Input Buffer Latch Hold	0.0		0.0		ns
t_{INSU}	Input Buffer Latch Setup	0.4		0.4		ns
t_{OUTH}	Output Buffer Latch Hold	0.0		0.0		ns
t_{OUTSU}	Output Buffer Latch Setup	0.4		0.4		ns
f_{MAX}	Flip-Flop (Latch) Clock Frequency		100.0		115.0	MHz

Notes:

1. *For dual-module macros, use $t_{PD1} + t_{RD1} + t_{PDm}$, $t_{CO} + t_{RD1} + t_{PDm}$, or $t_{PD1} + t_{RD1} + t_{SUD}$, whichever is appropriate.*
2. *Routing delays are for typical designs across worst-case operating conditions. These parameters should be used for estimating device performance. Post-route timing analysis or simulation is required to determine actual worst-case performance. Post-route timing is based on actual routing delay measurements performed on the device prior to shipment.*
3. *Data applies to macros based on the S-module. Timing parameters for sequential macros constructed from C-modules can be obtained from the ALS Timer utility.*
4. *Setup and hold timing parameters for the Input Buffer Latch are defined with respect to the PAD and the D input. External setup/hold timing parameters must account for delay from an external PAD signal to the G inputs. Delay from an external PAD signal to the G input subtracts (adds) to the internal setup (hold) time.*

A1280XL Timing Characteristics (continued)

(Worst-Case Commercial Conditions)

Input Module Propagation Delays			'Std' Speed		'–1' Speed		
Parameter	**Description**		**Min.**	**Max.**	**Min.**	**Max.**	**Units**
t_{INYH}	Pad to Y High			1.4		1.2	ns
t_{INYL}	Pad to Y Low			1.7		1.4	ns
t_{INGH}	G to Y High			2.7		2.3	ns
t_{INGL}	G to Y Low			3.5		3.0	ns
Input Module Predicted Routing Delays[1]							
t_{IRD1}	FO=1 Routing Delay			4.3		3.7	ns
t_{IRD2}	FO=2 Routing Delay			4.9		4.2	ns
t_{IRD3}	FO=3 Routing Delay			5.3		4.5	ns
t_{IRD4}	FO=4 Routing Delay			6.1		5.2	ns
t_{IRD8}	FO=8 Routing Delay			8.8		7.5	ns
Global Clock Network							
t_{CKH}	Input Low to High	FO = 32 FO = 384		6.8 7.6		5.8 6.5	ns
t_{CKL}	Input High to Low	FO = 32 FO = 384		6.7 7.6		5.7 6.5	ns
t_{PWH}	Minimum Pulse Width High	FO = 32 FO = 384	4.3 4.6		3.5 3.9		ns
t_{PWL}	Minimum Pulse Width Low	FO = 32 FO = 384	4.3 4.6		3.5 3.9		ns
t_{CKSW}	Maximum Skew	FO = 32 FO = 384		1.0 1.0		0.9 0.9	ns
t_{SUEXT}	Input Latch External Setup	FO = 32 FO = 384	0.0 0.0		0.0 0.0		ns
t_{HEXT}	Input Latch External Hold	FO = 32 FO = 384	3.4 4.3		2.9 3.7		ns
t_P	Minimum Period	FO = 32 FO = 384	8.7 9.6		7.4 8.0		ns
f_{MAX}	Maximum Frequency	FO = 32 FO = 384		115.0 105.0		135.0 125.0	MHz

Note:

1. *These parameters should be used for estimating device performance. Optimization techniques may further reduce delays by 0 to 4 ns. Routing delays are for typical designs across worst-case operating conditions. Post-route timing analysis or simulation is required to determine actual worst-case performance. Post-route timing is based on actual routing delay measurements performed on the device prior to shipment.*

A1280XL Timing Characteristics (continued)

(Worst-Case Commercial Conditions)

Output Module Timing		'Std' Speed		'–1' Speed		
Parameter	**Description**	**Min.**	**Max.**	**Min.**	**Max.**	**Units**
TTL Output Module **Timing**[1, 2]						
t_{DLH}	Data to Pad High		5.0		4.3	ns
t_{DHL}	Data to Pad Low		5.4		4.6	ns
t_{ENZH}	Enable Pad Z to High		5.0		4.3	ns
t_{ENZL}	Enable Pad Z to Low		5.5		4.7	ns
t_{ENHZ}	Enable Pad High to Z		7.2		6.1	ns
t_{ENLZ}	Enable Pad Low to Z		7.2		6.1	ns
t_{GLH}	G to Pad High		5.6		4.8	ns
t_{GHL}	G to Pad Low		6.3		5.4	ns
t_{LCO}	I/O Latch Clock-Out (pad-to-pad), 64 clock loading		13.1		11.0	ns
t_{ACO}	Array Clock-Out (pad-to-pad), 64 clock loading		18.5		15.7	ns
d_{TLH}	Capacitive Loading, Low to High		0.05		0.04	ns/pF
d_{THL}	Capacitive Loading, High to Low		0.07		0.04	ns/pF
CMOS Output Module **Timing**[1, 2]						
t_{DLH}	Data to Pad High		6.4		5.4	ns
t_{DHL}	Data to Pad Low		4.5		3.8	ns
t_{ENZH}	Enable Pad Z to High		5.0		4.3	ns
t_{ENZL}	Enable Pad Z to Low		5.5		4.7	ns
t_{ENHZ}	Enable Pad High to Z		7.2		6.1	ns
t_{ENLZ}	Enable Pad Low to Z		7.2		6.1	ns
t_{GLH}	G to Pad High		5.6		4.8	ns
t_{GHL}	G to Pad Low		6.3		5.4	ns
t_{LCO}	I/O Latch Clock-Out (pad-to-pad), 64 clock loading		15.5		13.0	ns
t_{ACO}	Array Clock-Out (pad-to-pad), 64 clock loading		21.8		18.5	ns
d_{TLH}	Capacitive Loading, Low to High		0.07		0.06	ns/pF
d_{THL}	Capacitive Loading, High to Low		0.06		0.05	ns/pF

Notes:

1. *Delays based on 35 pF loading.*
2. *Maximum Recommended Simultaneous Switching Outputs:*

PQFP, CPGA, CQFP	*20pF*	*160*
	35pF	*90*
	50pF	*64*

Macro Library

Hard Macros—Combinatorial

Function	Macro	Description	Modules S	Modules C
1200XL Combinatorial Logic Module	CM8	Combinational Module (Full 1200XL Logic Module)		1
1200XL Sequential Logic Module	DFM7A	4-input D-Type Flip-Flop with Multiplexed Data, active low Clear, and active high clock	1	
	DFM7B	4-input D-Type Flip-Flop with Multiplexed Data, active low Clear, and clock	1	
Adder	FA1A	1-bit adder, carry in and carry out active low, A-input active low		2
	FA1B	1-bit adder, carry in and carry out active low		2
	FA2A	2-bit adder, carry in and carry out active low, A0 and A1 inputs active low		2
	HA1	Half-Adder		2
	HA1A	Half-Adder with active low A-input		2
	HA1B	Half-Adder with active low carry out and sum		2
	HA1C	Half-Adder with active low carry out		2
AND	AND2	2-input AND		1
	AND2A	2-input AND with active low A-input		1
	AND2B	2-input AND with active low inputs		1
	AND3	3-input AND		1
	AND3A	3-input AND with active low A-input		1
	AND3B	3-input AND with active low A- and B-inputs		1
	AND3C	3-input AND with active low inputs		1
	AND4	4-input AND		1
	AND4A	4-input AND with active low A-input		1
	AND4B	4-input AND with active low A- and B-inputs		1
	AND4C	4-input AND with active low A-, B-, and C-inputs		1
	AND4D	4-input AND with active low inputs		2
	AND5B	5-input AND with active low A- and B-inputs		1
AND-OR	AO1	3-input AND-OR		1
	AO10	5-input AND-OR-AND		1
	AO11	3-input AND-OR		1
	AO1A	3-input AND-OR with active low A-input		1
	AO1B	3-input AND-OR with active low C-input		1
	AO1C	3-input AND-OR with active low A- and C-inputs		1
	AO1D	3-input AND-OR with active low A- and B-inputs		1
	AO1E	3-input AND-OR with active low inputs		1
	AO2	4-input AND-OR		1
	AO2A	4-input AND-OR with active low A-input		1
	AO2B	4-input AND-OR with active low A- and B-inputs		1
	AO2C	4-input AND-OR with active low A- and C-inputs		1
	AO2D	4-input AND-OR with active low A-, B-, and C-inputs		1
	AO2E	4-input AND-OR with active low inputs		1
	AO3	4-input AND-OR		1
	AO3A	4-input AND-OR		1
	AO3B	4-input AND-OR		1
	AO3C	4-input AND-OR		1
	AO4A	4-input AND-OR		1
	AO5A	4-input AND-OR		1
	AO6	2-wide 4-input AND-OR		1
	AO6A	2-wide 4-input AND-OR with active low D-input		1

Hard Macros—Combinatorial (continued)

Function	Macro	Description	Modules S	Modules C
AND-OR	AO7	5-input AND-OR		1
	AO8	5-input AND-OR with active low C- and D-inputs		1
	AO9	5-input AND-OR		1
	AOI1	3-input AND-OR-INVERT		1
	AOI1A	3-input AND-OR-INVERT with active low A-input		1
	AOI1B	3-input AND-OR-INVERT with active low C-input		1
	AOI1C	3-input AND-OR-INVERT with active low A- and B-inputs		1
	AOI1D	3-input AND-OR-INVERT with active low inputs		1
	AOI2A	4-input AND-OR-INVERT with active low A-input		1
	AOI2B	4-input AND-OR-INVERT with active low A- and C-inputs		1
	AOI3A	4-input AND-OR-INVERT with active low inputs		1
	AOI4	2-wide 4-input AND-OR-INVERT		2
	AOI4A	2-wide 4-input AND-OR-INVERT with active low C-input		1
AND-XOR	AX1	3-input AND-XOR with active low A-input		1
	AX1A	3-input AND-XOR-INVERT with active low A-input		2
	AX1B	3-input AND-XOR with active low A- and B-inputs		1
	AX1C	3-input AND-XOR		1
Buffer	BUF	Buffer with active high input and output		1
	BUFA	Buffer with active low input and output		1
Clock Net	CLKINT	Clock Net Interface	0	0
	GAND2	2-input AND Clock Net		1
	GMX4	4-to-1 Multiplexor Clock Net		1
	GNAND2	2-input NAND Clock Net		1
	GNOR2	2-input NOR Clock Net		1
	GOR2	2-input OR Clock Net		1
	GXOR2	2-input Exclusive OR Clock Net		1
Inverter	INV	Inverter with active low output		1
	INVA	Inverter with active low input		1
Majority	MAJ3	3-input complex AND-OR		1
MUX	MX2	2-to-1 Multiplexor		1
	MX2A	2-to-1 Multiplexor with active low A-input		1
	MX2B	2-to-1 Multiplexor with active low B-input		1
MUX	MX2C	2-to-1 Multiplexor with active low output		1
	MX4	4-to-1 Multiplexor		1
	MXC1	Boolean		2
	MXT	Boolean		2
NAND	NAND2	2-input NAND		1
	NAND2A	2-input NAND with active low A-input		1
	NAND2B	2-input NAND with active low inputs		1
	NAND3	3-input NAND		1
	NAND3A	3-input NAND with active low A-input		1
	NAND3B	3-input NAND with active low A- and B-inputs		1
	NAND3C	3-input NAND with active low inputs		1
	NAND4	4-input NAND		2
	NAND4A	4-input NAND with active low A-input		1
	NAND4B	4-input NAND with active low A- and B-inputs		1
	NAND4C	4-input NAND with active low A-, B-, and C-inputs		1
	NAND4D	4-input NAND with active low inputs		1
	NAND5C	5-input NAND with active low A-, B-, and C-inputs		1

Hard Macros—Combinatorial (continued)

Function	Macro	Description	Modules S	Modules C
NOR	NOR2	2-input NOR		1
	NOR2A	2-input NOR with active low A-input		1
	NOR2B	2-input NOR with active low inputs		1
	NOR3	3-input NOR		1
	NOR3A	3-input NOR with active low A-input		1
	NOR3B	3-input NOR with active low A- and B-inputs		1
	NOR3C	3-input NOR with active low inputs		1
	NOR4	4-input NOR		2
	NOR4A	4-input NOR with active low A-input		1
	NOR4B	4-input NOR with active low A- and B-inputs		1
	NOR4C	4-input NOR with active low A-, B-, and C-inputs		1
	NOR4D	4-input NOR with active low inputs		1
	NOR5C	5-input NOR with active low A-, B-, and C-inputs		1
OR	OR2	2-input OR		1
	OR2A	2-input OR with active low A-input		1
	OR2B	2-input OR with active low inputs		1
	OR3	3-input OR		1
	OR3A	3-input OR with active low A-input		1
	OR3B	3-input OR with active low A- and B-inputs		1
	OR3C	3-input OR with active low inputs		1
	OR4	4-input OR		1
	OR4A	4-input OR with active low A-input		1
	OR4B	4-input OR with active low A- and B-input		1
	OR4C	4-input OR with active low A-, B-, and C-inputs		1
	OR4D	4-input OR with active low inputs		2
	OR5B	5-input OR with active low A- and B-inputs		1
OR-AND	OA1	3-input OR-AND		1
	OA1A	3-input OR-AND with active low A-input		1
	OA1B	3-input OR-AND with active low C-input		1
	OA1C	3-input OR-AND with active low A- and C-inputs		1
	OA2	2-wide 4-input OR-AND		1
	OA2A	2 wide 4-input OR-AND with active low A-input		1
	OA3	4-input OR-AND		1
	OA3A	4-input OR-AND with active low C-input		1
	OA3B	4-input OR-AND with active low A- and C-inputs		1
	OA4	4-input OR-AND		1
	OA4A	4-input OR-AND with active low C-input		1
	OA5	4-input complex OR-AND		1
	OAI1	3-input OR-AND-INVERT		1
	OAI2A	4-input OR-AND-INVERT with active low D-input		1
	OAI3	4-input OR-AND-INVERT		1
	OAI3A	4-input OR-AND-INVERT with active low C- and D-inputs		1
XNOR	XNOR	2-input XNOR		1
XNOR-AND	XA1A	3-input XNOR-AND		1
XNOR-OR	XO1A	3-input XNOR-OR		1
XOR	XOR	2-input XOR		1
XOR-AND	XA1	3-input XOR-AND		1
XOR-OR	XO1	3-input XOR-OR		1

Hard Macros—Sequential

Function	Macro	Description	Modules S	Modules C
D-Type	DF1	D-Type Flip-Flop	1	
	DF1A	D-Type Flip-Flop with active low output	1	
	DF1B	D-Type Flip-Flop with active low clock	1	
	DF1C	D-Type Flip-Flop with active low clock and output	1	
	DFC1	D-Type Flip-Flop with active high Clear	1	1
	DFC1A	D-Type Flip-Flop with active high Clear and active low clock	1	1
	DFC1B	D-Type Flip-Flop with active low Clear	1	
	DFC1D	D-Type Flip-Flop with active low Clear and clock	1	
	DFE	D-Type Flip-Flop with active high Enable	1	
	DFE1B	D-Type Flip-Flop with active low Enable	1	
	DFE1C	D-Type Flip-Flop with active low Enable and clock	1	
	DFE3A	D-Type Flip-Flop with Enable and active low Clear	1	
	DFE3B	D-Type Flip-Flop with Enable and active low Clear and clock	1	
	DFE3C	D-Type Flip-Flop with active low Enable and Clear	1	
	DFE3D	D-Type Flip-Flop with active low Enable, Clear, and clock	1	
	DFEA	D-Type Flip-Flop with Enable and active low clock	1	
	DFM	2-input D-Type Flip-Flop with Multiplexed Data	1	
	DFM1B	2-input D-Type Flip-Flop with Multiplexed Data and active low output	1	
	DFM1C	2-input D-Type Flip-Flop with Multiplexed Data and active low clock and output	1	
	DFM3	2-input D-Type Flip-Flop with Multiplexed Data and Clear	1	1
	DFM3B	2-input D-Type Flip-Flop with Multiplexed Data and active low Clear and clock	1	
	DFM3E	2-input D-Type Flip-Flop with Multiplexed Data, Clear, and active low clock	1	1
	DFM4C	2-input D-Type Flip-Flop with Multiplexed Data and active low Preset and output	1	
	DFM4D	2-input D-Type Flip-Flop with Multiplexed Data and active low Preset, clock, and output	1	
	DFM6A	4-input D-Type Flip-Flop with Multiplexed Data, active low Clear, and active high clock	1	
	DFM6B	4-input D-Type Flip-Flop with Multiplexed Data, active low Clear, and clock	1	
	DFMA	2-input D-Type Flip-Flop with Multiplexed Data and active low clock	1	
	DFMB	2-input D-Type Flip-Flop with Multiplexed Data and active low Clear	1	
	DFME1A	2-input D-Type Flip-Flop with Multiplexed Data and active low Enable	1	
	DFP1	D-Type Flip-Flop with active high Preset		2
	DFP1A	D-Type Flip-Flop with active high Preset and active low clock		2
	DFP1B	D-Type Flip-Flop with active low Preset		2
	DFP1C	D-Type Flip-Flop with active high Preset and active low output	1	1
	DFP1D	D-Type Flip-Flop with active low Preset and clock		2
	DFP1E	D-Type Flip-Flop with active low Preset and output	1	
	DFP1F	D-Type Flip-Flop with active high Preset and active low clock and output	1	1
	DFP1G	D-Type Flip-Flop with active low Preset, clock, and output	1	
	DFPC	D-Type Flip-Flop with active high Preset, active low Clear, and active high clock		2
	DFPCA	D-Type Flip-Flop with active high Preset and active low Clear and clock		2

Hard Macros—Sequential (continued)

Function	Macro	Description	Modules S	Modules C
J-K Type	JKF	JK Flip-Flop with active low K-input	1	
	JKF1B	JK Flip-Flop with active low clock and K-input	1	
	JKF2A	JK Flip-Flop with active low Clear and K-input	1	
	JKF2B	JK Flip-Flop with active low Clear, clock, and K-input	1	
	JKF2C	JK Flip-Flop with active high Clear and active low K-input	1	1
	JKF2D	JK Flip-Flop with active high Clear and active low clock and K-input	1	1
T-Type	TF1A	T-Type Flip-Flop with active low Clear	1	
	TF1B	T-Type Flip-Flop with active low Clear and clock	1	
Latch	DL1	Data Latch	1	
	DL1A	Data Latch with active low output	1	
	DL1B	Data Latch with active low clock	1	
	DL1C	Data Latch with active low clock and output	1	
	DLC	Data Latch with active low Clear	1	
	DLC1	Data Latch with active high Clear		1
	DLC1A	Data Latch with active high Clear and active low clock		1
	DLC1F	Data Latch with active high Clear and active low output		1
	DLC1G	Data Latch with active high Clear and active low clock and output		1
	DLCA	Data Latch with active low Clock and Clear	1	
	DLE	Data Latch with active high Enable	1	
	DLE1D	Data Latch with active high Enable and clock and active low input and output	1	
	DLE2B	Data Latch with active low Enable, Clear, and clock	1	
	DLE2C	Data Latch with active low Enable and clock and active high Clear		1
	DLE3B	Data Latch with active low Enable and clock and active low Preset		1
	DLE3C	Data Latch with active low Enable, Preset, and clock		1
	DLEA	Data Latch with active low Enable and active high clock	1	
	DLEB	Data Latch with active high Enable and active high clock	1	
	DLEC	Data Latch with active low Enable and clock	1	
	DLM	2-input Data Latch with Multiplexed Data	1	
	DLM3	4-input Data Latch with Multiplexed Data	1	
	DLM3A	4-input Data Latch with Multiplexed Data and active low clock	1	
	DLM4	Data Latch with Multiplexed Data	1	
	DLM4A	Data Latch with Multiplexed Data	1	
	DLMA	2-input Data Latch with Multiplexed Data and active low clock	1	
	DLME1A	2-input Data Latch with Multiplexed Data and Enable and active low clock	1	
	DLP1	Data Latch with active high Preset and clock		1
	DLP1A	Data Latch with active high Preset and active low clock		1
	DLP1B	Data Latch with active low Preset and active high clock		1
	DLP1C	Data Latch with active low Preset and clock		1
	DLP1D	Data Latch with active low Preset and output and active high clock	1	
	DLP1E	Data Latch with active low Preset, clock, and output	1	

Input/Output Macros

Function	Macro	Description	I/O Modules
Buffer	IBDL	Input Buffer with Latch Clock	1
	INBUF	Input Buffer	1
	OBHS	Output Buffer, High Slew	1
	OUTBUF	Output Buffer, High Slew	1
Bidirectional	BBHS	Bidirectional Buffer, High Slew	1
	BBDLHS	Bidirectional with Input Latch and Output Latch	1
	BIBUF	Bidirectional Buffer, High Slew (with hidden buffer at Y pin)	1
	CLKBIBUF	Bidirectional with Input Dedicated to Clock Network	1
Input	CLKBUF	Input for Dedicated Routed Clock Network	1
Output	DBDLKS	Output Buffer with Latch	1
	OBHS	Output Buffer	1
	TBHS	Tristate output, High Slew	1
	TRIBUFF	Tristate output, High Slew	1

Soft Macros (continued)

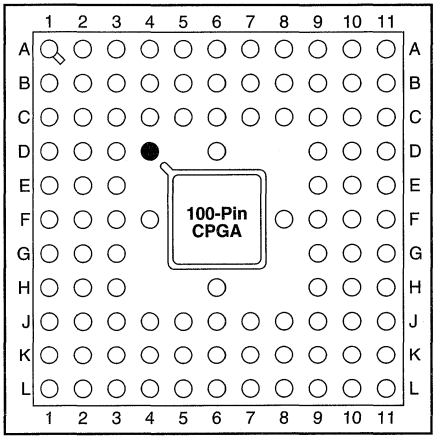
Function	Macro	Description	Maximum Logic Levels	Modules S	Modules C
Adder	FADD10	10-bit adder	3		56
	FADD12	12-bit adder	4		9
	FADD16	16-bit adder	5		97
	FADD8	8-bit adder	4		44
	FADD9	9-bit adder with active low carry out	3		49
	VAD16C	Very fast 16-bit adder, no Carry in	3		91
	VADC16C	Very fast 16-bit adder with Carry in	3		97
Comparator	ICMP4	4-bit Identity Comparator	2		5
	ICMP8	8-bit Identity Comparator	3		9
	MCMPC2	2-bit Magnitude Comparator with Enable	3		9
	MCMPC4	4-bit Magnitude Comparator with Enable	4		18
	MCMPC8	8-bit Magnitude Comparator with Enable	6		36
Counter	CNT4A	4-bit binary counter with load and clear	4	4	8
	CNT4B	4-bit binary counter with load, clear, carry-in, carry-out	4	4	7
	FCTD16C	Fast 16-bit Down Counter, parallel loadable	2	19	33
	FCTD8A	Fast 8-bit Down Counter, parallel loadable	1	10	18
	FCTD8B	Fast 8-bit Down Counter, parallel loadable	1	9	13
	FCTU16C	Fast 16-bit Up Counter, parallel loadable	2	19	31
	FCTU8A	Fast 8-bit Up Counter, parallel loadable	1	10	17
	FCTU8B	Fast 8-bit Up Counter, parallel loadable	1	9	12
	UDCNT4A	4-bit up/down counter with load, carry-in, and carry-out	5	4	13
	VCTD16C	Very fast 16-bit down counter, delay after load, registered control inputs	1	34	41
	VCTD2CP	2-bit down counter, prescaler, delay after load, used to build VCTD counters	1	5	2
	VCTD2CU	2-bit down counter, upper bits, delay after load, used to build VCTD counters	1	2	3
	VCTD4CL	4-bit down counter, lower bits, delay after load, used to build VCTD counters	1	4	7
	VCTD4CM	4-bit down counter, middle bits, delay after load, used to build VCTD counters	1	4	8
Decoder	DEC2X4	2-to-4 decoder	1		4
	DEC2X4A	2-to-4 decoder with active low outputs	1		4
	DEC3X8	3-to-8 decoder	1		8
	DEC3X8A	3-to-8 decoder with active low outputs	1		8
	DEC4X16A	4-to-16 decoder with active low outputs	2		20
	DECE2X4	2-to-4 decoder with enable	1		4
	DECE2X4A	2-to-4 decoder with enable and active low outputs	1		4
	DECE3X8	3-to-8 decoder with enable	2		11
	DECE3X8A	3-to-8 decoder with enable and active low outputs	2		11
Latch	DLC8A	Octal latch with clear active low 8-bit Data Latch with active low Clear	1	8	
	DLE8	Octal latch with enable 8-bit Data Latch with active high Enable	1	8	
	DLM8	Octal latch with multiplexed data 8-bit Data Latch with Multiplexed Data	1	8	
MUX	MX16	16-to-1 Multiplexor	2		5
	MX8	8-to-1 Multiplexor with active high output	2		3
	MX8A	8-to-1 Multiplexor with active low output	2		3
Multiplier	SMULT8	8-bit by 8-bit Multiplier			242
Shift Register	SREG4A	4-bit shift register with clear active low	1	4	
	SREG8A	8-bit shift register with clear active low	1	8	

Soft Macros—TTL Equivalent

Function	Macro	Description	Maximum Logic Levels	Modules S	Modules C
	TA00	2-input NAND	1		1
	TA02	2-input NOR	1		1
	TA04	Inverter	1		1
	TA07	Buffer	1		1
	TA08	2-input AND	1		1
	TA10	3-input NAND	1		1
	TA11	3-input AND	1		1
	TA138	3-to-8 decoder with enable and active low outputs	2		12
	TA139	2-to-4 decoder with active low enable and outputs	1		4
	TA150	16-to-1 multiplexor with active low enable	3		6
	TA151	8-to-1 multiplexor with enable and both active low and active high output	3		5
	TA153	4-to-1 multiplexor with active low enable	2		2
	TA154	4-to-16 decoder with active low outputs and select lines	2		22
	TA157	2-to-1 multiplexor with active low enable	1		1
	TA160	4-bit decade counter with active low clear and load	4	4	8
	TA161	4-bit binary counter with active low clear and load	3	4	6
	TA164	8-bit serial in, parallel out shift register, active low clear	1	8	
	TA169	4-bit Up/Down Counter	6	4	14
	TA174	hex D-type flip-flop with active low clear	1	6	
	TA175	quadruple D-type flip-flop with active low clear	1	4	
	TA181	ALU			37
	TA190	4-bit up/down decode counter with up/down mode	7	4	31
	TA191	4-bit up/down binary counter with up/down mode	7	4	30
	TA194	4-bit bidirectional universal shift register	1	4	4
	TA195	4-bit parallel-access shift register	1	4	1
	TA20	4-input NAND	1		2
	TA21	4-input AND	1		1
	TA269	8-bit up/down binary counter	8	8	28
	TA27	3-input NOR	1		1
	TA273	octal register with clear	1	8	
	TA280	9-bit odd/even parity generator and checker	4		9
	TA32	2-input OR	1		1
	TA377	octal register with active low enable	1	8	
	TA40	4-input NAND	1		2
	TA42	4 to 10 decoder	1		10
	TA51	AND-OR-Invert	1		2
	TA54	4-wide 2-input AND-OR-Invert	2		5
	TA55	2-wide 4-input AND-OR-Invert	2		3
	TA688	8-bit identity comparator	3		9
	TA86	2-input exclusive OR	1		1

Package Pin Assignments

100-Pin CPGA (Top View)



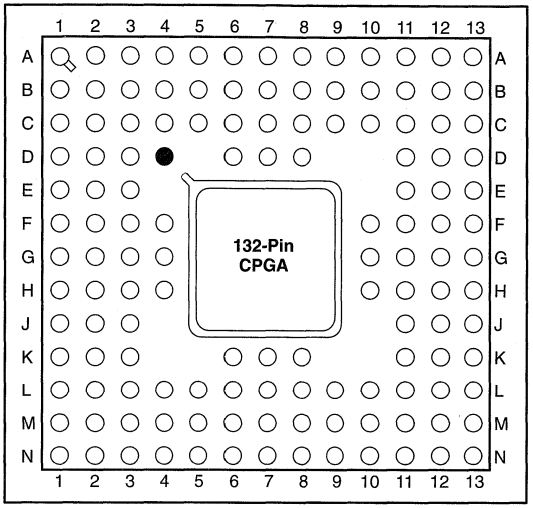
Signal	Pad Number	Location
PRA or I/O	85	A7
PRB or I/O	92	A4
MODE	2	C2
SDI or I/O	77	C8
DCLK or I/O	100	C3
CLKA or I/O	87	C6
CLKB or I/O	90	D6
GND	7, 20, 32, 44, 55, 70, 82, 94	E3, G3, J5, J7, G9, D10, C7, C5
V_{CC}	15, 38, 64, 88	F3, K6, F9, B6
V_{PP}	63	F10
V_{SV}	14, 65	G1, E11
V_{KS}	62	F11

Notes:

1. *Unused I/O pins are designated as outputs by ALS and are driven low.*
2. *All unassigned pins are available for use as I/Os.*
3. *MODE = GND, except during device programming or debugging.*
4. *$V_{PP} = V_{CC}$, except during device programming.*
5. *$V_{SV} = V_{CC}$, except during device programming.*
6. *V_{KS} = GND, except during device programming.*

Package Pin Assignments (continued)

132-Pin CPGA (Top View)



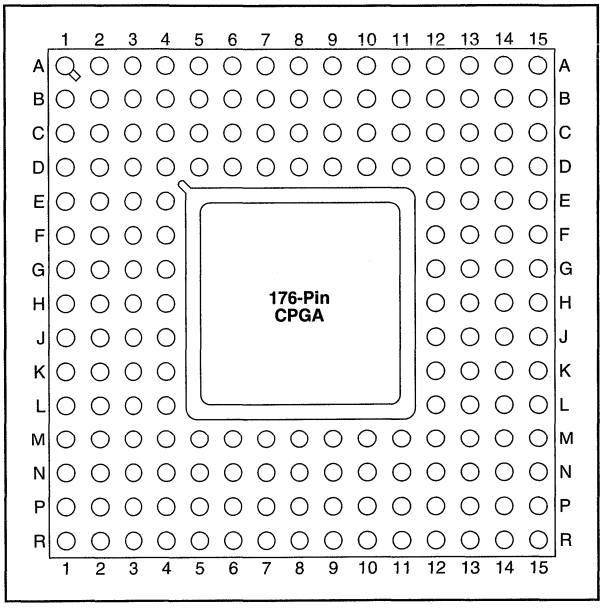
Signal	Pad Number	Location
PRA or I/O	113	B8
PRB or I/O	121	C6
MODE	2	A1
SDI or I/O	101	B12
DCLK or I/O	132	C3
CLKA or I/O	115	B7
CLKB or I/O	119	B6
GND	9, 10, 26, 27, 41, 58, 59, 73, 74, 92, 93, 107, 108, 125, 126	E3, F4, J2, J3, L5, L9, M9, K12, J11, E12, E11, C9, B9, B5, C5
V_{CC}	18, 19, 49, 50, 83, 84, 116, 117	G3, G2, L7, K7, G10, G11, D7, C7
V_{PP}	82	G13
V_{SV}	17, 85	G4, G12
V_{KS}	81	H13

Notes:

1. *Unused I/O pins are designated as outputs by ALS and are driven low.*
2. *All unassigned pins are available for use as I/Os.*
3. *MODE = GND, except during device programming or debugging.*
4. *$V_{PP} = V_{CC}$, except during device programming.*
5. *$V_{SV} = V_{CC}$, except during device programming.*
6. *V_{KS} = GND, except during device programming.*

Package Pin Assignments (continued)

176-Pin CPGA (Top View)



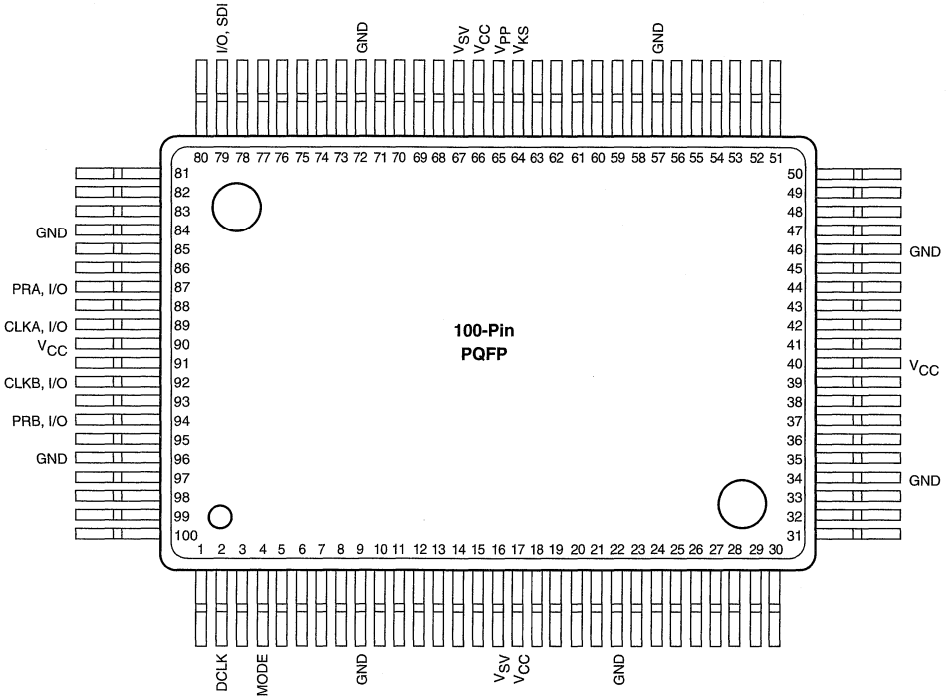
Signal	Pad Number	Location
PRA or I/O	152	C9
PRB or I/O	160	D7
MODE	2	C3
SDI or I/O	135	B14
DCLK or I/O	175	B3
CLKA or I/O	154	A9
CLKB or I/O	158	B8
GND	1, 8, 18, 23, 33, 38, 45, 57, 67, 77, 89 101, 106, 111, 121, 126, 133, 145, 156, 165	D4, E4, G4, H4, K4, L4, M4, M6, M8, M10, M12 K12, J12, H12, F12, E12, D12, D10, C8, D6
V_{CC}	13, 24, 28, 52, 68, 82, 112, 116, 140, 155, 170	F4, H3, J4, M5, N8, M11, H13, G12, D11, D8, D5
V_{PP}	110	J14
V_{SV}	25, 113	H2, H14
V_{KS}	109	J13

Notes:

1. *Unused I/O pins are designated as outputs by ALS and are driven low.*
2. *All unassigned pins are available for use as I/Os.*
3. *MODE = GND, except during device programming or debugging.*
4. *$V_{PP} = V_{CC}$, except during device programming.*
5. *$V_{SV} = V_{CC}$, except during device programming.*
6. *V_{KS} = GND, except during device programming.*

Package Pin Assignments (continued)

100-Pin PQFP (Top View)

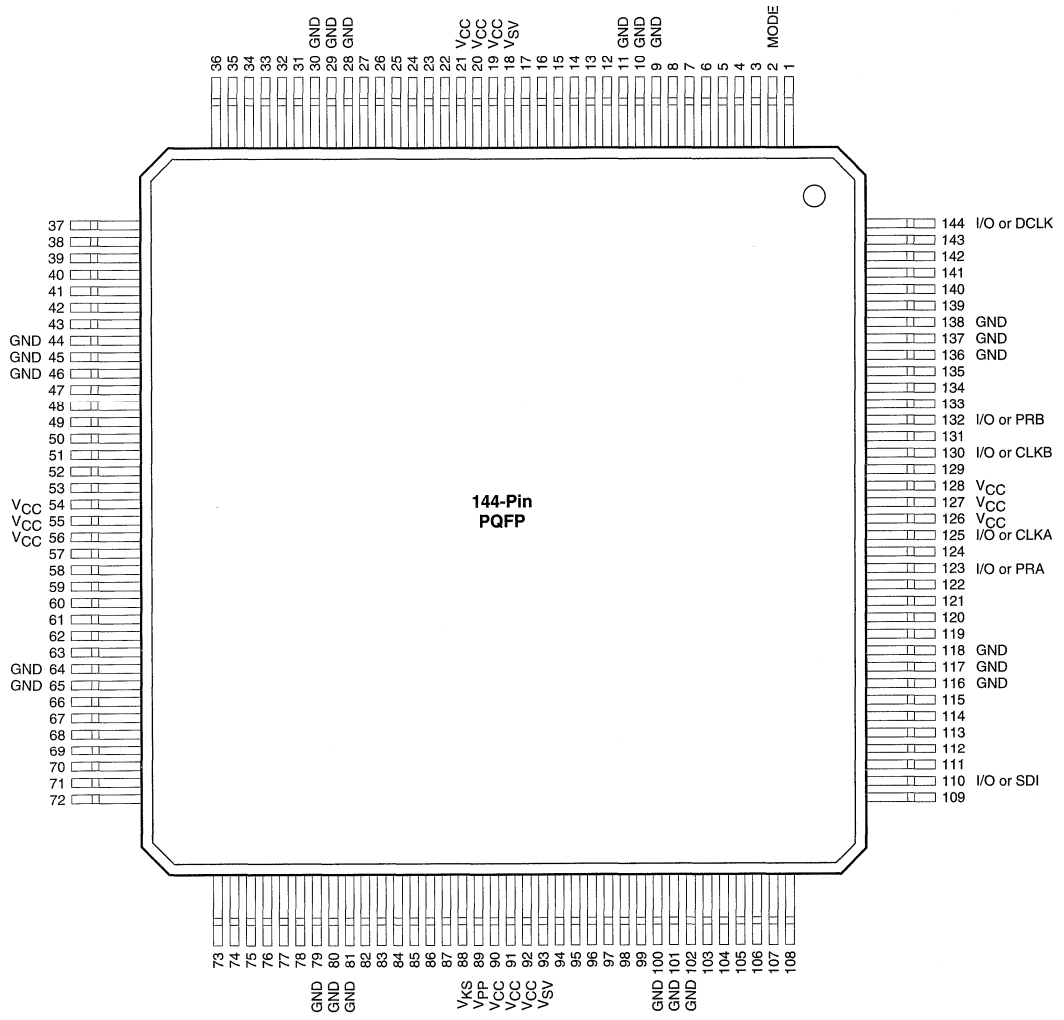


Notes:

1. *Unused I/O pins are designated as outputs by ALS and are driven low.*
2. *All unassigned pins are available for use as I/Os.*
3. *MODE = GND, except during device programming or debugging.*
4. *$V_{PP} = V_{CC}$, except during device programming.*
5. *$V_{SV} = V_{CC}$, except during device programming.*
6. *V_{KS} = GND, except during device programming.*

Package Pin Assignments (continued)

144-Pin PQFP (Top View)

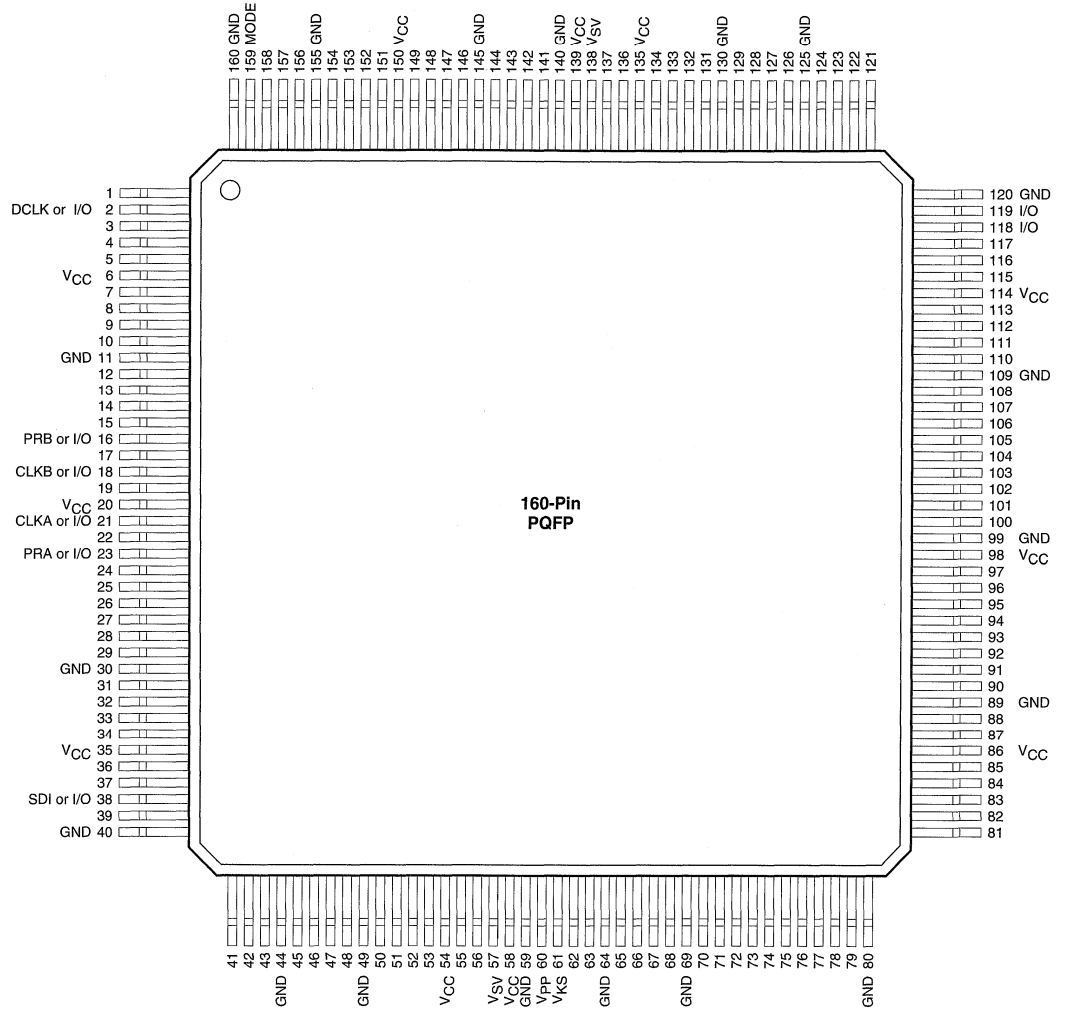


Notes:

1. *Unused I/O pins are designated as outputs by ALS and are driven low.*
2. *All unassigned pins are available for use as I/Os.*
3. *MODE = GND, except during device programming or debugging.*
4. *$V_{PP} = V_{CC}$, except during device programming.*
5. *$V_{SV} = V_{CC}$, except during device programming.*
6. *V_{KS} = GND, except during device programming.*

Package Pin Assignments (continued)

160-Pin CPGA (Top View)

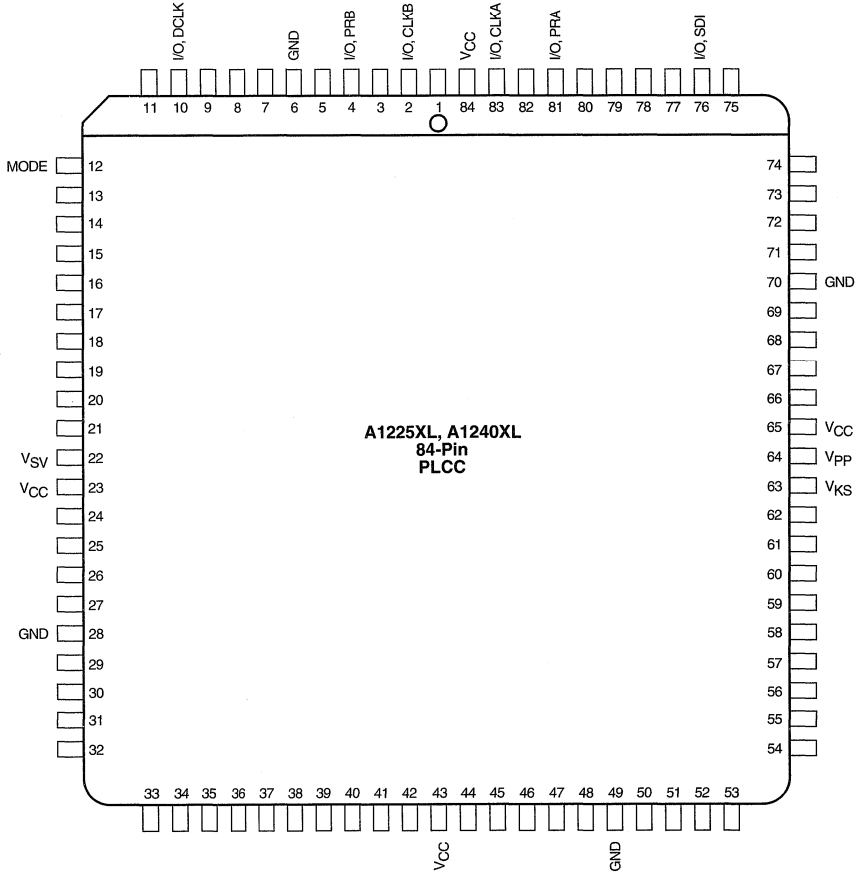


Notes:

1. *Unused I/O pins are designated as outputs by ALS and are driven low.*
2. *All unassigned pins are available for use as I/Os.*
3. *MODE = GND, except during device programming or debugging.*
4. *$V_{PP} = V_{CC}$, except during device programming.*
5. *$V_{SV} = V_{CC}$, except during device programming.*
6. *V_{KS} = GND, except during device programming.*

Package Pin Assignments (continued)

84-Pin PLCC (Top View)

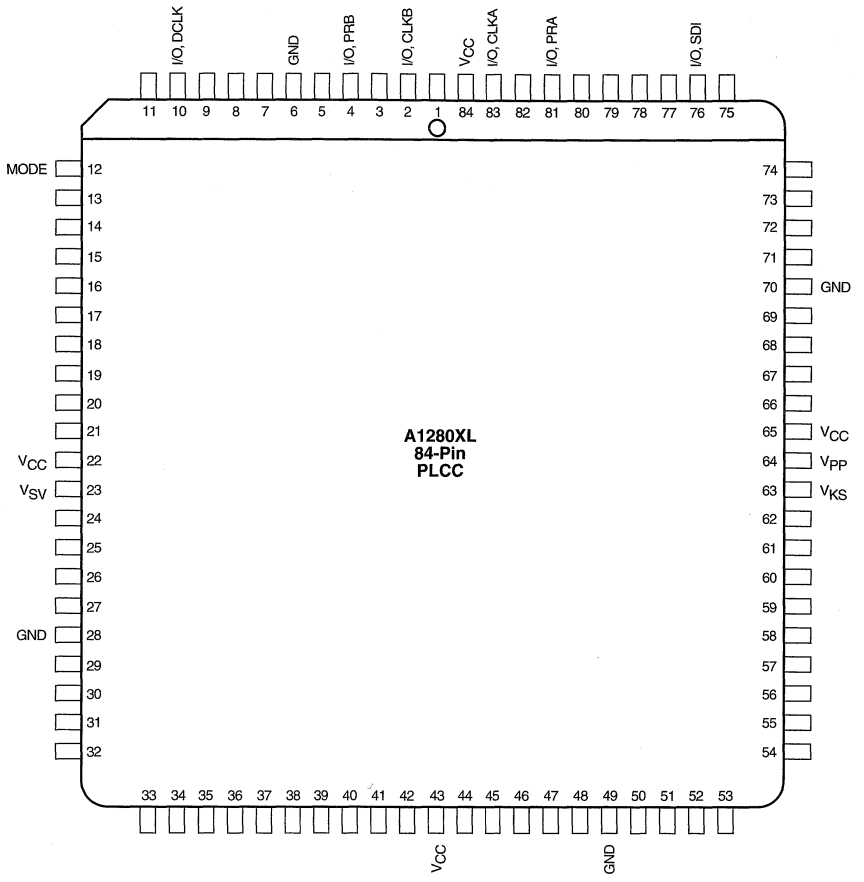


Notes:

1. *Unused I/O pins are designated as outputs by ALS and are driven low.*
2. *All unassigned pins are available for use as I/Os.*
3. *MODE = GND, except during device programming or debugging.*
4. *$V_{PP} = V_{CC}$, except during device programming.*
5. *$V_{SV} = V_{CC}$, except during device programming.*
6. *V_{KS} = GND, except during device programming.*

Package Pin Assignments (continued)

84-Pin PLCC (Top View)

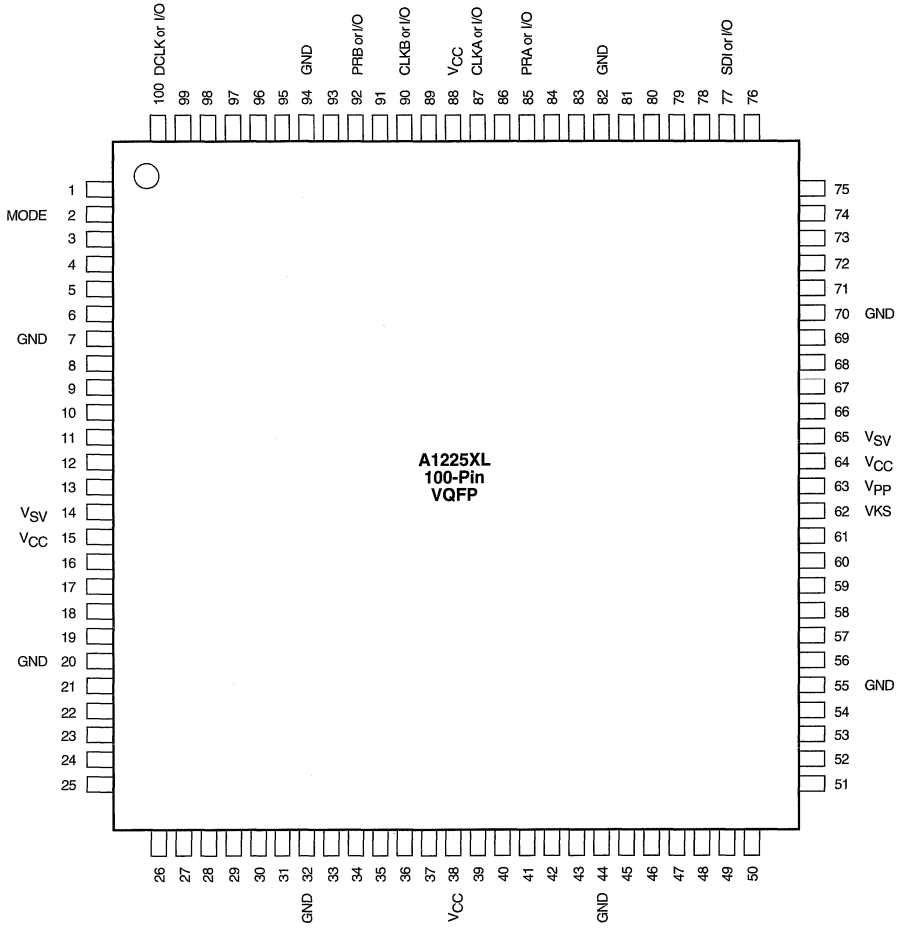


Notes:

1. *Unused I/O pins are designated as outputs by ALS and are driven low.*
2. *All unassigned pins are available for use as I/Os.*
3. *MODE = GND, except during device programming or debugging.*
4. *$V_{PP} = V_{CC}$, except during device programming.*
5. *$V_{SV} = V_{CC}$, except during device programming.*
6. *V_{KS} = GND, except during device programming.*

Package Pin Assignments (continued)

100-Pin VQFP (Top View)

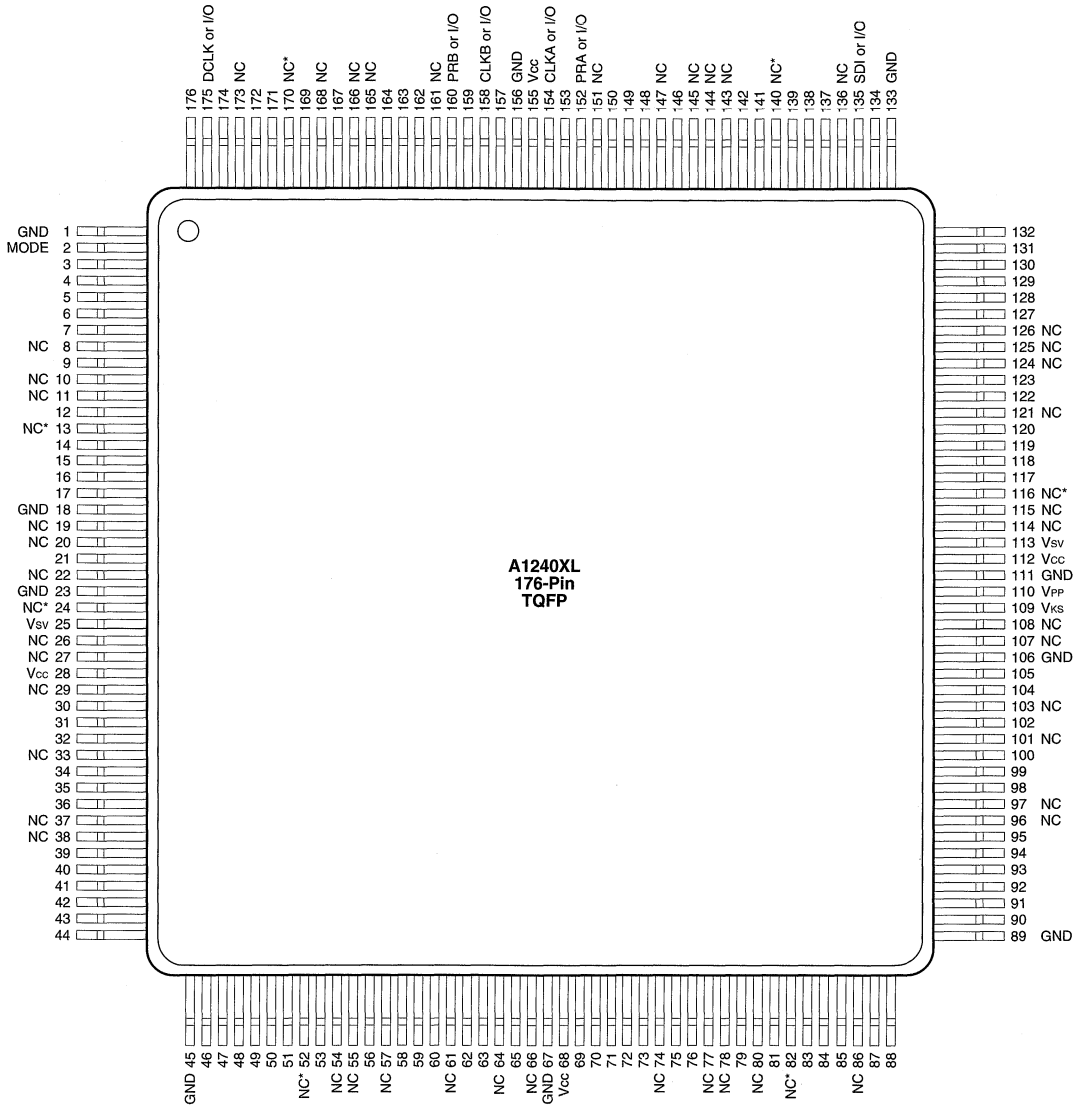


Notes:

1. *Unused I/O pins are designated as outputs by ALS and are driven low.*
2. *All unassigned pins are available for use as I/Os.*
3. *MODE = GND, except during device programming or debugging.*
4. *$V_{PP} = V_{CC}$, except during device programming.*
5. *$V_{SV} = V_{CC}$, except during device programming.*
6. *V_{KS} = GND, except during device programming.*

Package Pin Assignments (continued)

176-Pin TQFP (Top View)



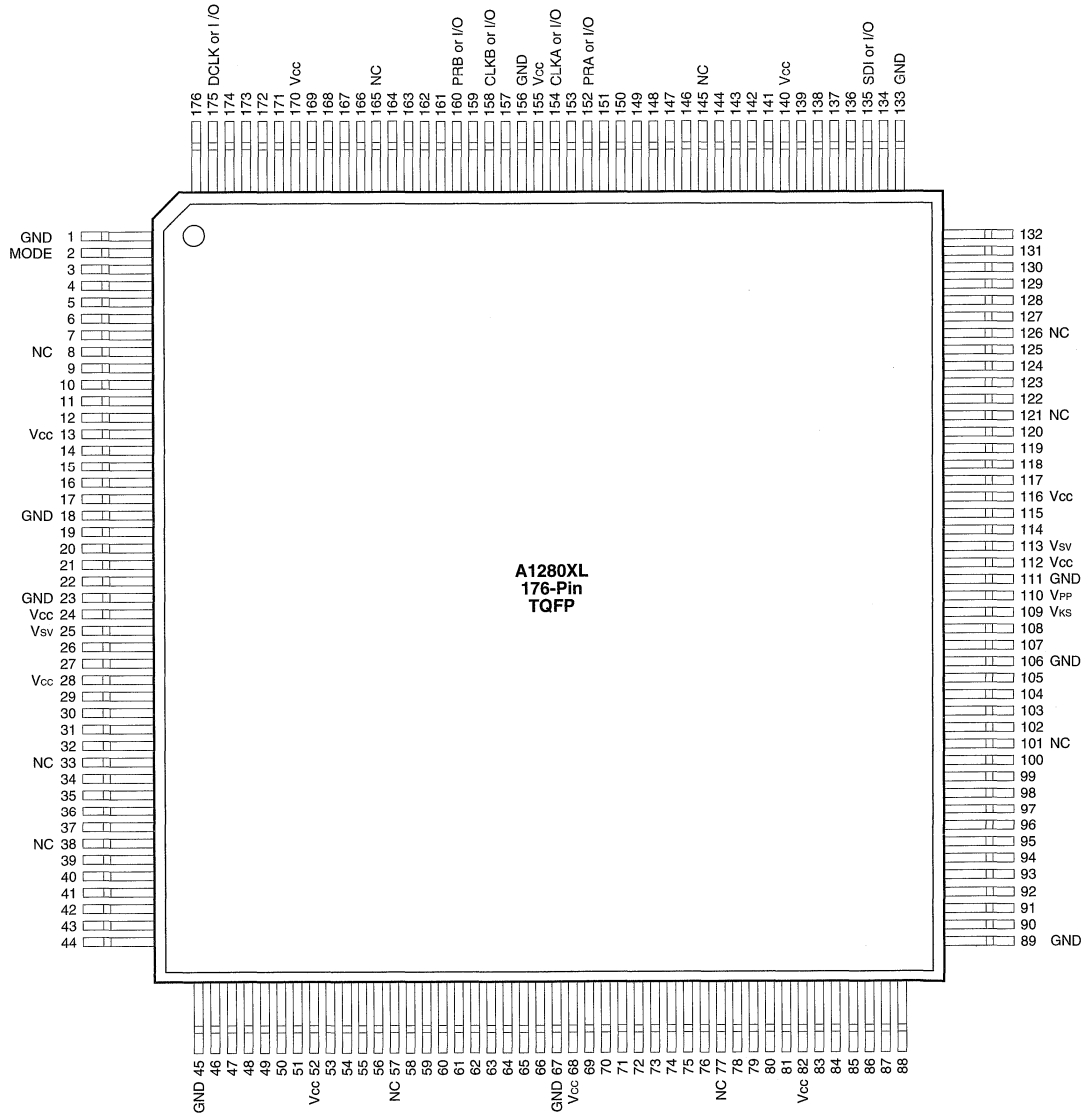
* For pin compatibility, these pins should be reserved for Vcc in higher capacity devices.

Notes:

1. *Unused I/O pins are designated as outputs by ALS and are driven low.*
2. *All unassigned pins are available for use as I/Os.*
3. *MODE = GND, except during device programming or debugging.*
4. $V_{PP} = V_{CC}$, *except during device programming.*
5. $V_{SV} = V_{CC}$, *except during device programming.*
6. V_{KS} = *GND, except during device programming.*

Package Pin Assignments (continued)

176-Pin TQFP (Top View)



Notes:

1. *Unused I/O pins are designated as outputs by ALS and are driven low.*
2. *All unassigned pins are available for use as I/Os.*
3. *MODE = GND, except during device programming or debugging.*
4. $V_{PP} = V_{CC}$, *except during device programming.*
5. $V_{SV} = V_{CC}$, *except during device programming.*
6. V_{KS} = *GND, except during device programming.*

ACT™ 3
Field Programmable Gate Arrays

Features

- Highly Predictable Performance with 100% Automatic Placement and Routing
- 7.5 ns Clock-to-Output Times
- Up to 250 MHz On-Chip Performance
- Up to 228 User-Programmable I/O Pins
- Four Fast, Low-Skew Clock Networks
- More Than 500 Macro Functions
- Up to 10,000 Gate Array Equivalent Gates (up to 25,000 equivalent PLD Gates)
- Replaces up to twenty 32 macro-cell CPLDs
- Replaces up to one hundred 20-pin PAL® Packages
- Up to 1153 Dedicated Flip-Flops
- I/O Drive to 12 mA
- VQFP, TQFP, BGA, and PQFP Packages
- Nonvolatile, User Programmable
- Low-power 0.8 micron CMOS Technology
- Fully Tested Prior to Shipment

Product Family Profile

Device	A1415A	A1425A	A1440A	A1460A	A14100A
Capacity					
Gate Array Equivalent Gates	1,500	2,500	4,000	6,000	10,000
PLD Equivalent Gates	3,750	6,250	10,000	15,000	25,000
TTL Equivalent Packages (40 gates)	40	60	100	150	250
20-Pin PAL Equivalent Packages (100 gates)	15	25	40	60	100
Logic Modules	200	310	564	848	1,377
S-Module	104	160	288	432	697
C-Module	96	150	276	416	680
Dedicated Flip-Flops[1]	264	360	568	768	1,153
User I/Os (maximum)	80	100	140	168	228
Packages[2] (by pin count)					
CPGA	100	133	175	207	257
PLCC	84	84	84	—	—
PQFP	100	100, 160	160	160, 208	—
RQFP	—	—	—	—	208
VQFP	100	100	100	—	—
TQFP	—	—	176	176	—
BGA	—	—	—	225	313
CQFP	—	132	—	196	256
Performance[3] (maximum, worst-case commercial)					
Chip-to-Chip[4]	108 MHz	108 MHz	100 MHz	97 MHz	93 MHz
Accumulators (16-bit)	63 MHz	63 MHz	63 MHz	63 MHz	63 MHz
Loadable Counter (16-bit)	110 MHz	110 MHz	110 MHz	110 MHz	105 MHz
Prescaled Loadable Counters (16-bit)	250 MHz	250 MHz	250 MHz	200 MHz	200 MHz
Datapath, Shift Registers	250 MHz	250 MHz	250 MHz	200 MHz	200 MHz
Clock-to-Output (pad-to-pad)	7.5 ns	7.5 ns	8.5 ns	9.0 ns	9.5 ns

Notes:

1. *One flip-flop per S-Module, two flip-flops per I/O-Module.*
2. *See product plan on page 1-156 for package availability.*
3. *Based on A1415A-3, A1425A-3, A1440A-3, A1460A-3, and A14100A-3.*
4. *Clock-to-Output + Setup*

Description

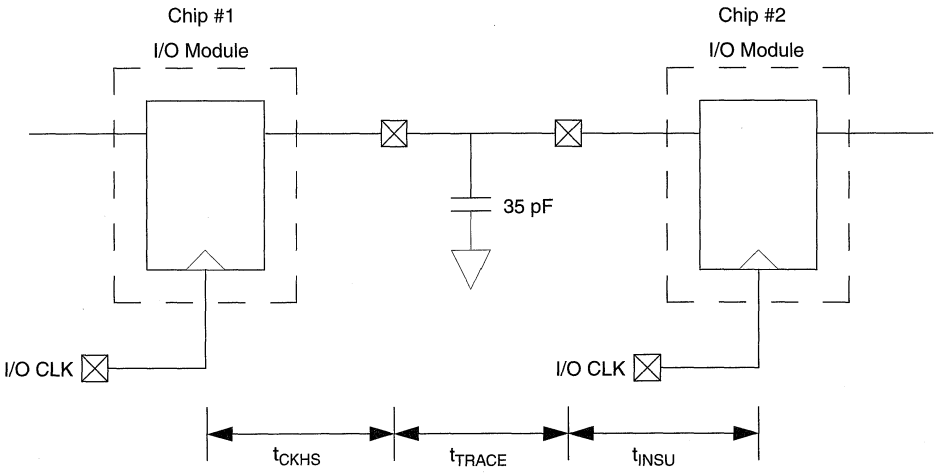
The ACT 3 family, based on Actel's proprietary PLICE® antifuse technology and 0.8-micron double-metal, double-poly CMOS process, offers a high-performance programmable solution capable of 250 MHz on-chip performance and 7.5 nanosecond clock-to-output speeds. The ACT 3 family spans capacities from 1,500 to 10,000 gate array equivalent gates (up to 25,000 PLD gates), and offers very high pin-to-gate ratios, with up to 228 user I/Os for 10,000 gate designs.

Predictable Performance* (Worst-Case Commercial)	
Accumulators (16-bit)	58–63 MHz
Loadable Counters (16-bit)	95–110 MHz
Prescaled Loadable Counters (16-bit)	230–250 MHz
Shift Registers	250–250 MHz

The ACT 3 family represents the third generation of Actel Field Programmable Gate Arrays (FPGAs). The family improves on the proven ACT 2 family two-module architecture, consisting of combinatorial and sequential-combinatorial logic modules. The ACT 3 family offers registered I/O modules delivering 9 ns clock-to-out times. The devices contain four clock distribution networks, including dedicated array and I/O clocks, supporting very fast synchronous and asynchronous designs. In addition, routed clocks can be used to drive high fanout signals like resets or output enables, reducing buffering requirements.

The ACT 3 family is supported by the Designer and Designer Advantage systems, allowing logic design implementation with minimum effort. The systems offer Microsoft® Windows™ and X Window™ graphical user interfaces and integrate with the resident CAE system to provide a complete gate array design environment: schematic capture, simulation, fully automatic placement and routing, timing verification and device programming. The systems also include the ACTmap™ optimization and synthesis tool, and the ACTgen™ Macro Builder, a powerful macro function generator for counters, adders, and other structured blocks. The systems are available for 386/486/Pentium PCs and for HP™, and Sun™ workstations running Viewlogic®, Mentor Graphics®, and OrCAD™ tools.

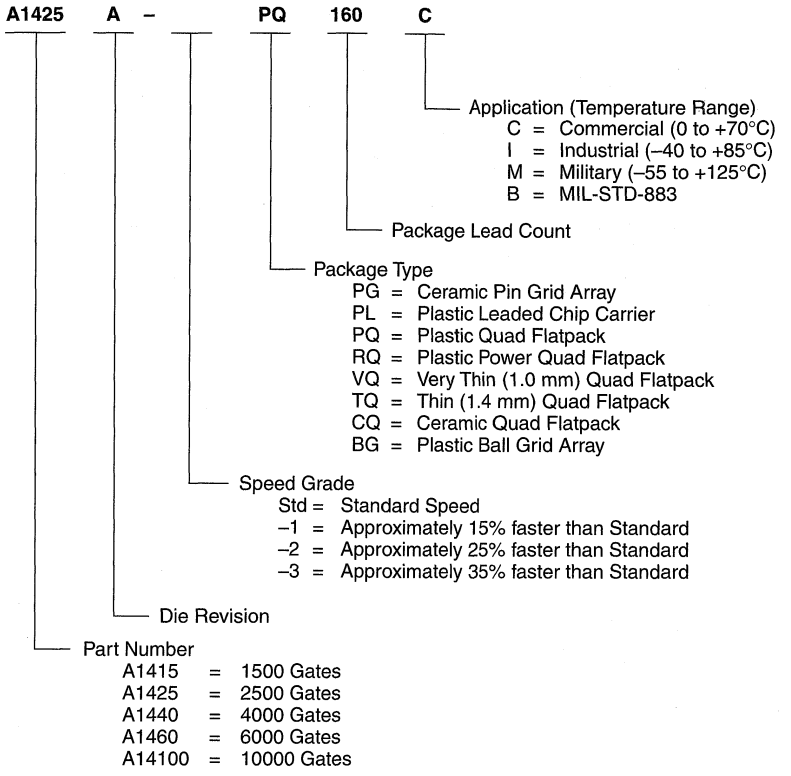
Chip-to-Chip Performance



Chip-to-Chip Performance (Worst-Case Commercial)

	t_{CKHS}	t_{TRACE}	t_{INSU}	Total	MHz
A1425A-3	7.5	1.0	1.8	10.3 ns	97
A1460A-3	9.0	1.0	1.3	11.3 ns	88

Ordering Information



Product Plan[1]

	Speed Grade*				Application				
	Std	–1	–2	–3	C	I	M	B	E
A1415A Device									
84-pin Plastic Leaded Chip Carrier (PLCC)	✔	✔	✔	P	✔	✔	—	—	—
100-pin Plastic Quad Flatpack (PQFP)	✔	✔	✔	P	✔	✔	—	—	—
100-pin Very Thin Quad Flatpack (VQFP)	✔	✔	✔	P	✔	—	—	—	—
100-pin Ceramic Pin Grid Array (CPGA)	✔	✔	✔	P	✔	—	—	—	—
A1425A Device									
84-pin Plastic Leaded Chip Carrier (PLCC)	✔	✔	✔	✔	✔	✔	—	—	—
100-pin Plastic Quad Flatpack (PQFP)	✔	✔	✔	✔	✔	✔	—	—	—
100-pin Very Thin Quad Flatpack (VQFP)	✔	✔	✔	✔	✔	—	—	—	—
132-pin Ceramic Quad Flatpack (CQFP)	✔	—	—	—	✔	—	P	P	—
133-pin Ceramic Pin Grid Array (CPGA)	✔	✔	✔	✔	✔	—	✔	P	—
160-pin Plastic Quad Flatpack (PQFP)	✔	✔	✔	✔	✔	✔	—	—	—
A1440A Device									
84-pin Plastic Leaded Chip Carrier (PLCC)	✔	✔	✔	P	✔	✔	—	—	—
100-pin Very Thin Quad Flatpack (VQFP)	✔	✔	✔	P	✔	—	—	—	—
160-pin Plastic Quad Flatpack (PQFP)	✔	✔	✔	P	✔	✔	—	—	—
176-pin Thin Quad Flatpack (TQFP)	✔	✔	✔	P	✔	—	—	—	—
177-pin Ceramic Pin Grid Array (CPGA)	✔	✔	✔	P	✔	—	—	—	—
A1460A Device									
160-pin Plastic Quad Flatpack (PQFP)	✔	✔	✔	✔	✔	✔	—	—	—
176-pin Thin Quad Flatpack (TQFP)	✔	✔	✔	✔	✔	—	—	—	—
196-pin Ceramic Quad Flatpack (CQFP)	✔	—	—	—	✔	—	P	P	—
207-pin Ceramic Pin Grid Array (CPGA)	✔	✔	✔	✔	✔	—	✔	P	—
208-pin Plastic Quad Flatpack (PQFP)	✔	✔	✔	✔	✔	✔	—	—	—
225-pin Platic Ball Grid Array (BGA)	✔	✔	✔	✔	✔	—	—	—	—
A14100A Device									
208-pin Power Quad Flatpack (RQFP)	✔	✔	✔	P	✔	✔	—	—	—
257-pin Ceramic Pin Grid Array (CPGA)	✔	✔	✔	P	✔	—	✔	P	—
313-pin Plastic Ball Grid Array (BGA)	✔	✔	✔	P	✔	P	—	—	—
256-pin Ceramic Quad Flatpack (CQFP)	✔	—	—	—	✔	—	P	P	—

Applications: *C = Commercial*, *I = Industrial*, *M = Military*, *B = MIL-STD-883*, *E = Extended*

Availability: ✔ = *Available*, *P = Planned*, — = *Not Planned*

** Speed Grade:* *–1 = Approx. 15% faster than Standard*, *–2 = Approx. 25% faster than Standard*, *–3 = Approx. 35 % faster than Standard.*

Note:

1. Availability as of October 1994. Please consult Actel Representatives for current availability.

Plastic Device Resources

Device Series	Logic Modules	Gates	User I/Os							
			PLCC	PQFP, RQFP			VQFP	TQFP	BGA	
			84-pin	100-pin	160-pin	208-pin	100-pin	176-pin	225-pin	313-pin
A1415A	200	1500	70	80	—	—	80	—	—	—
A1425A	310	2500	70	80	100	—	83	—	—	—
A1440A	564	4000	70	—	131	—	83	140	—	—
A1460A	848	6000	—	—	131	167	—	151	168	—
A14100A	1377	10000	—	—	—	175	—	—	—	228

Hermetic Device Resources

Device Series	Logic Modules	Gates	User I/Os							
			CPGA					CQFP		
			100-pin	133-pin	175-pin	207-pin	257-pin	132-pin	196-pin	256-pin
A1415A	200	1500	80	—	—	—	—	—	—	—
A1425A	310	2500	—	100	—	—	—	100	—	—
A1440A	564	4000	—	—	140	—	—	—	—	—
A1460A	848	6000	—	—	—	168	—	—	168	—
A14100A	1377	10000	—	—	—	—	228	—	—	228

Pin Description

CLKA **Clock A (Input)**

TTL Clock input for clock distribution networks. The Clock input is buffered prior to clocking the logic modules. This pin can also be used as an I/O.

CLKB **Clock B (Input)**

TTL Clock input for clock distribution networks. The Clock input is buffered prior to clocking the logic modules. This pin can also be used as an I/O.

DCLK **Diagnostic Clock (Input)**

TTL Clock input for diagnostic probe and device programming. DCLK is active when the MODE pin is HIGH. This pin functions as an I/O when the MODE pin is LOW.

GND **Ground**

LOW supply voltage.

HCLK **Dedicated (Hard-wired) Array Clock (Input)**

TTL Clock input for sequential modules. This input is directly wired to each S-Module and offers clock speeds independent of the number of S-Modules being driven. This pin can also be used as an I/O.

I/O **Input/Output (Input, Output)**

The I/O pin functions as an input, output, three-state, or bidirectional buffer. Input and output levels are compatible with standard TTL and CMOS specifications. Unused I/O pins are automatically driven LOW by the ALS software.

IOCLK **Dedicated (Hard-wired) I/O Clock (Input)**

TTL Clock input for I/O modules. This input is directly wired to each I/O module and offers clock speeds independent of the number of I/O modules being driven. This pin can also be used as an I/O.

IOPCL **Dedicated (Hard-wired) I/O Preset/Clear (Input)**

TTL input for I/O preset or clear. This global input is directly wired to the preset and clear inputs of all I/O registers. This pin functions as an I/O when no I/O preset or clear macros are used.

MODE **Mode (Input)**

The MODE pin controls the use of diagnostic pins (DCLK, PRA, PRB, SDI). When the MODE pin is HIGH, the special functions are active. When the MODE pin is LOW, the pins function as I/Os.

NC **No Connection**

This pin is not connected to circuitry within the device.

PRA **Probe A (Output)**

The Probe A pin is used to output data from any user-defined design node within the device. This independent diagnostic pin can be used in conjunction with the Probe B pin to allow real-time diagnostic output of any signal path within the device. The Probe A pin can be used as a user-defined I/O when debugging has been completed. The pin's probe capabilities can be permanently disabled to protect programmed design confidentiality. PRA is accessible when the MODE pin is HIGH. This pin functions as an I/O when the MODE pin is LOW.

PRB **Probe B (Output)**

The Probe B pin is used to output data from any user-defined design node within the device. This independent diagnostic pin can be used in conjunction with the Probe A pin to allow real-time diagnostic output of any signal path within the device. The Probe B pin can be used as a user-defined I/O when debugging has been completed. The pin's probe capabilities can be permanently disabled to protect programmed design confidentiality. PRB is accessible when the MODE pin is HIGH. This pin functions as an I/O when the MODE pin is LOW.

SDI **Serial Data Input (Input)**

Serial data input for diagnostic probe and device programming. SDI is active when the MODE pin is HIGH. This pin functions as an I/O when the MODE pin is LOW.

V_{CC} **5 V Supply Voltage**

HIGH supply voltage.

V_{KS} **Programming Voltage**

Supply voltage used for device programming. This pin must be connected to GND during normal operation.

V_{PP} **Programming Voltage**

Supply voltage used for device programming. This pin must be connected to V_{CC} during normal operation.

V_{SV} **Programming Voltage**

Supply voltage used for device programming. This pin must be connected to V_{CC} during normal operation.

Architecture

This section of the data sheet is meant to familiarize the user with the architecture of the ACT 3 family of FPGA devices. A generic description of the family will be presented first, followed by a detailed description of the logic blocks, the routing structure, the antifuses, and the special function circuits. The on-chip circuitry required to program the devices is not covered.

Topology

The ACT 3 family architecture is composed of six key elements: Logic modules, I/O modules, I/O Pad Drivers, Routing Tracks, Clock Networks, and Programming and Test Circuits. The basic structure is similar for all devices in the family, differing only in the number of rows, columns, and I/Os. The array itself consists of alternating rows of modules and channels. The logic modules and channels are in the center of the array; the I/O modules are located along the array periphery. A simplified floor plan is depicted in Figure 1.

Logic Modules

ACT 3 logic modules are enhanced versions of the ACT 2 family logic modules. As in the ACT 2 family, there are two types of modules: C-modules and S-modules. The C-module is functionally equivalent to the ACT 2 C-module and implements high fanin combinatorial macros, such as 5-input AND, 5-input OR, and so on. It is available for use as the CM8 hard macro. The S-module is designed to implement high-speed sequential functions within a single module. S-modules consist of a full C-module driving a flip-flop, which allows an additional level of logic to be implemented without additional propagation delay. It is available for use as the DFM8A/B and DLM8A/B hard macros. C-modules and S-modules are arranged in pairs called module-pairs. Module-pairs are arranged in alternating patterns and make up the bulk of the array. This arrangement allows the placement software to support two-module macros of four types (CC, CS, SC, and SS). The C-module implements the following function:

$Y = !S1 * !S0 * D00 + !S1 * S0 * D01 + S1 * !S0 * D10 + S1 * S0 * D11$

where: $S0 = A0 * B0$ and $S1 = A1 + B1$

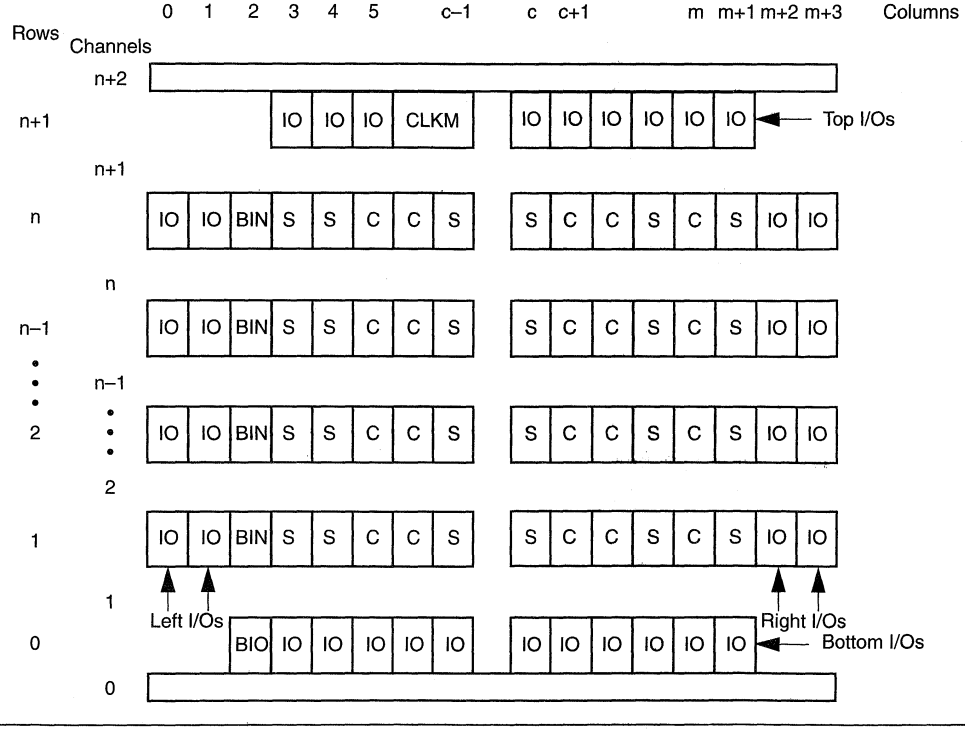


Figure 1 • Generalized Floor Plan of ACT 3 Device

The S-module contains a full implementation of the C-module plus a clearable sequential element that can either implement a latch or flip-flop function. The S-module can therefore implement any function implemented by the C-module. This allows complex combinatorial-sequential functions to be implemented with no delay penalty. The Action Logic System will automatically combine any C-module macro driving an S-module macro into the S-module, thereby freeing up a logic module and eliminating a module delay.

The clear input CLR is accessible from the routing channel. In addition, the clock input may be connected to one of three clock networks: CLK0, CLK1, or HCLK. The C-module and S-module functional descriptions are shown in Figures 2 and 3. The clock selection multiplexor selects the clock input to the S-module.

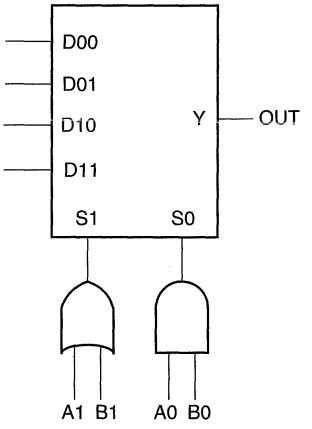


Figure 2 • C-Module Diagram

I/Os

I/O Modules

I/O modules provide an interface between the array and the I/O Pad Drivers. I/O modules are located in the array and access the routing channels in a similar fashion to logic modules. There are two types of I/O modules: side and top/bottom. The I/O module schematic is shown in Figure 4. UO1 and UO2 are inputs from the routing channel, one for the routing channel above and one for the routing channel below the module. The top/bottom I/O modules interact with only one channel and therefore have only one UO input. The signals DataIn and DataOut connect to the I/O pad driver. Each I/O module contains two D-type flip-flops. Each flip-flop is connected to the dedicated I/O clock (IOCLK). Each flip-flop can be bypassed by nonsequential I/Os. In addition, each flip-flop contains a data enable input that can be accessed from the routing channels (ODE and IDE). The asynchronous preset/clear input is driven by the dedicated preset/clear network (IOPCL). Either preset or clear can be selected individually on an I/O module by I/O module basis.

The I/O module output Y is used to bring Pad signals into the array *or* to feed the output register back into the array. This allows the output register to be used in high-speed state machine applications. Side I/O modules have a dedicated output segment for Y extending into the routing channels above and below (similar to logic modules). Top/Bottom I/O modules have no dedicated output segment. Signals coming

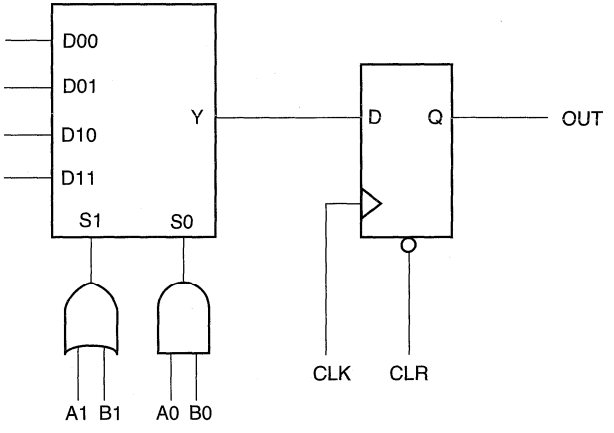


Figure 3 • S-Module Diagram

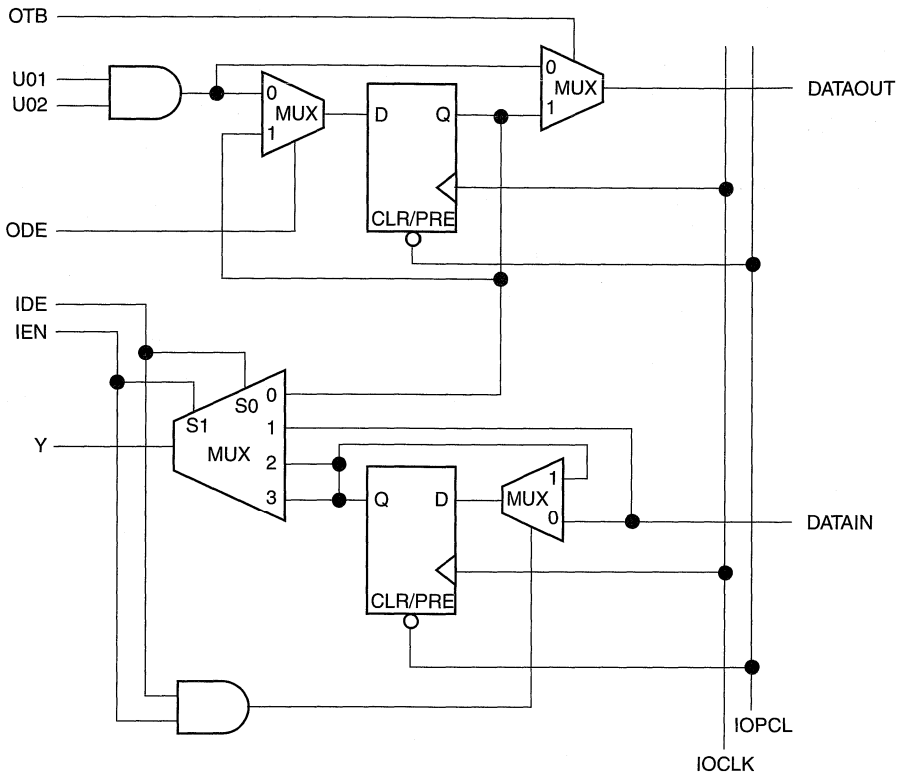


Figure 4 • Functional Diagram for I/O Module

into the chip from the top or bottom are routed using F-fuses and LVTs (F-fuses and LVTs are explained in detail in the routing section).

I/O Pad Drivers

All pad drivers are capable of being tristate. Each buffer connects to an associated I/O module with four signals: OE (Output Enable), IE (Input Enable), DataOut, and DataIn. Certain special signals used only during programming and test also connect to the pad drivers: OUTEN (global output enable), INEN (global input enable), and SLEW (individual slew selection). See Figure 5.

Special I/Os

The special I/Os are of two types: temporary and permanent. Temporary special I/Os are used during programming and testing. They function as normal I/Os when the MODE pin is inactive. Permanent special I/Os are user programmed as either normal I/Os or special I/Os. Their function does not change once the device has been programmed. The permanent special I/Os consist of the array clock input buffers (CLKA and CLKB), the hard-wired array clock input buffer (HCLK), the hard-wired I/O clock input buffer (IOCLK), and the hard-wired I/O register preset/clear input buffer (IOPCL). Their function is determined by the I/O macros selected.

Clock Networks

The ACT 3 architecture contains four clock networks: two high-performance dedicated clock networks and two general purpose routed networks. The high-performance networks function up to 200 MHz, while the general purpose routed networks function up to 150 MHz.

Dedicated Clocks

Dedicated clock networks support high performance by providing sub-nanosecond skew and guaranteed performance. Dedicated clock networks contain no programming elements in the path from the I/O Pad Driver to the input of S-modules or I/O modules. There are two dedicated clock networks: one for the array registers (HCLK), and one for the I/O registers (IOCLK). The clock networks are accessed by special I/Os.

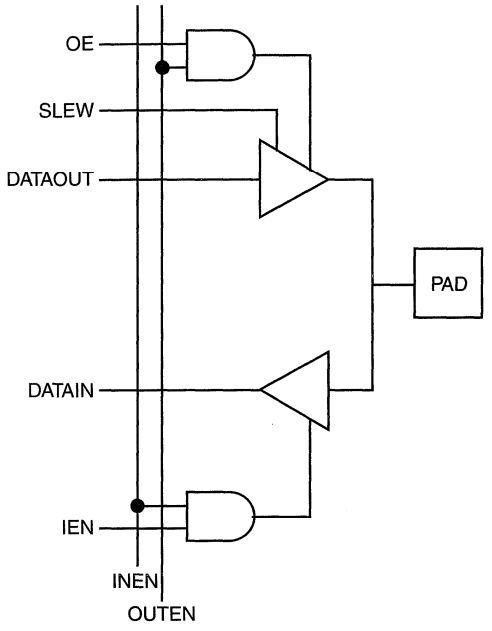


Figure 5 • Function Diagram for I/O Pad Driver

Routed Clocks

The routed clock networks are referred to as CLK0 and CLK1. Each network is connected to a clock module (CLKMOD) that selects the source of the clock signal and may be driven as follows (see Figure 6):

- externally from the CLKA pad
- externally from the CLKB pad
- internally from the CLKINA input
- internally from the CLKINB input

The clock modules are located in the top row of I/O modules. Clock drivers and a dedicated horizontal clock track are located in each horizontal routing channel. The function of the clock module is determined by the selection of clock macros from the macro library. The macro CLKBUF is used to connect one of the two external clock pins to a clock network, and the macro CLKINT is used to connect an internally generated clock signal to a clock network. Since both clock networks are identical, the user does not care whether CLK0 or CLK1 is being used. Routed clocks can also be used to drive high fanout nets like resets, output enables, or data enables. This saves logic modules and results in performance increases in some cases.

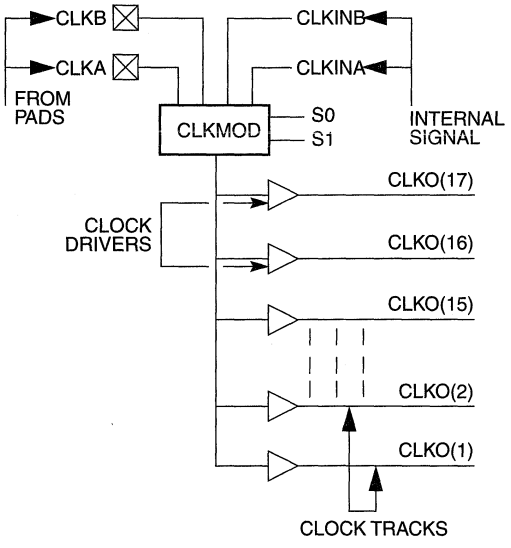


Figure 6 • Clock Networks

Routing Structure

The ACT 3 architecture uses vertical and horizontal routing tracks to connect the various logic and I/O modules. These routing tracks are metal interconnects that may either be of continuous length or broken into segments. Segments can be joined together at the ends using antifuses to increase their lengths up to the full length of the track.

Horizontal Routing

Horizontal channels are located between the rows of modules and are composed of several routing tracks. The horizontal routing tracks within the channel are divided into one or more segments. The minimum horizontal segment length is the width of a module-pair, and the maximum horizontal segment length is the full length of the channel. Any segment that spans more than one-third the row length is considered a long horizontal segment. A typical channel is shown in Figure 7. Undedicated horizontal routing tracks are used to route signal nets. Dedicated routing tracks are used for the global clock networks and for power and ground tie-off tracks.

Vertical Routing

Other tracks run vertically through the modules. Vertical tracks are of three types: input, output, and long. Vertical tracks are also divided into one or more segments. Each segment in an input track is dedicated to the input of a particular module. Each segment in an output track is

dedicated to the output of a particular module. Long segments are uncommitted and can be assigned during routing. Each output segment spans four channels (two above and two below), except near the top and bottom of the array where edge effects occur. LVTs contain either one or two segments. An example of vertical routing tracks and segments is shown in Figure 8.

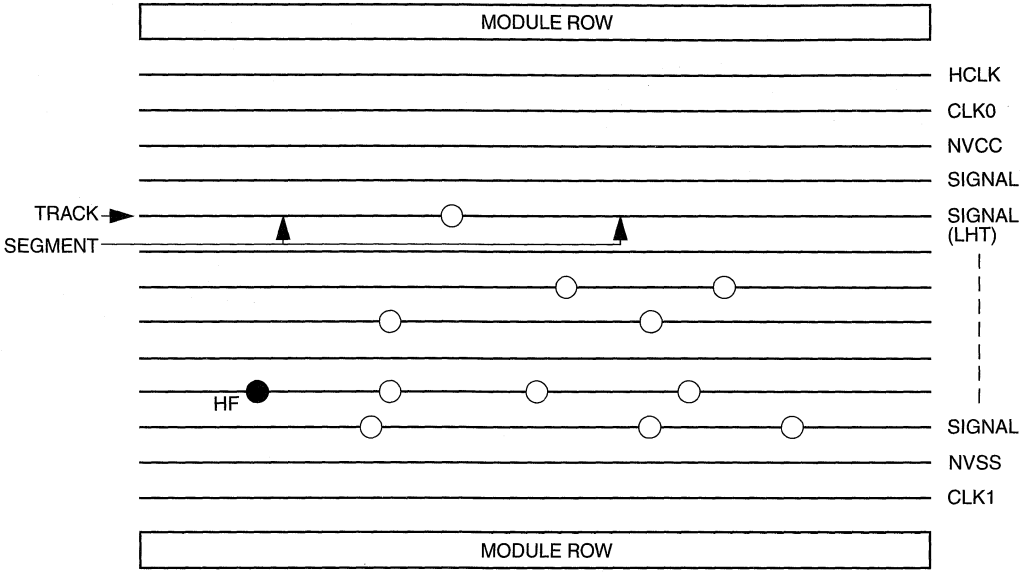


Figure 7 • Horizontel Routing Tracks and Segments

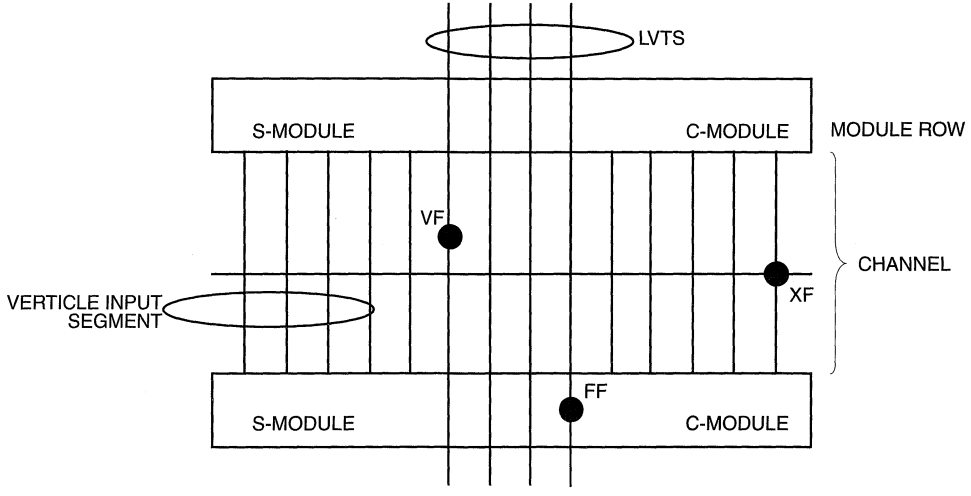


Figure 8 • Vertical Routing Tracks and Segments

Antifuse Connections

An antifuse is a "normally open" structure as opposed to the normally closed fuse structure used in PROMs or PALs. The use of antifuses to implement a programmable logic device results in highly testable structures as well as an efficient programming architecture. The structure is highly testable because there are no preexisting connections; temporary connections can be made using pass transistors. These temporary connections can isolate individual antifuses to be programmed as well as isolate individual circuit structures to be tested. This can be done both before and after programming. For example, all metal tracks can be tested for continuity and shorts between adjacent tracks, and the functionality of all logic modules can be verified.

Four types of antifuse connections are used in the routing structure of the ACT 3 array. (The physical structure of the antifuse is identical in each case; only the usage differs.) Table 1 shows four types of antifuses.

Table 1 • Antifuse Types

XF	Horizontal-to-Vertical Connection
HF	Horizontal-to-Horizontal Connection
VF	Vertical-to-Vertical Connection
FF	"Fast" Vertical Connection

Examples of all four types of connections are shown in Figures 7 and 8.

Module Interface

Connections to Logic and I/O modules are made through vertical segments that connect to the module inputs and outputs. These vertical segments lie on vertical tracks that span the entire height of the array.

Module Input Connections

The tracks dedicated to module inputs are segmented by pass transistors in each module row. During normal user operation, the pass transistors are inactive, which isolates the inputs of a module from the inputs of the module directly above or below it. During certain test modes, the pass transistors are active to verify the continuity of the metal tracks. Vertical input segments span only the channel above or the channel below. The logic modules are arranged such that half of the inputs are connected to the channel above and half of the inputs to segments in the channel below as shown in Figure 9.

Module Output Connections

Module outputs have dedicated output segments. Output segments extend vertically two channels above and two channels below, except at the top or bottom of the array. Output segments twist, as shown in Figure 10, so that only four vertical tracks are required.

LVT Connections

Outputs may also connect to nondedicated segments called Long Vertical Tracks (LVTs). Each module pair in the array shares four LVTs that span the length of the column. Any module in the column pair can connect to one of the LVTs in the column using an FF connection. The FF connection uses antifuses connected directly to the driver stage of the module output, bypassing the isolation transistor. FF antifuses are programmed at a higher current level than HF, VF, or XF antifuses to produce a lower resistance value.

Antifuse Connections

In general every intersection of a vertical segment and a horizontal segment contains an unprogrammed antifuse (XF-type). One exception is in the case of the clock networks.

Clock Connections

To minimize loading on the clock networks, a subset of inputs has antifuses on the clock tracks. Only a few of the C-module and S-module inputs can be connected to the clock networks. To further reduce loading on the clock network, only a subset of the horizontal routing tracks can connect to the clock inputs of the S-module.

Programming and Test Circuits

The array of logic and I/O modules is surrounded by test and programming circuits controlled by the temporary special I/O pins MODE, SDI, and DCLK. The function of these pins is similar to all ACT family devices. The ACT 3 family also includes support for two Actionprobe® circuits allowing complete observability of any logic or I/O module in the array using the temporary special I/O pins, PRA and PRB.

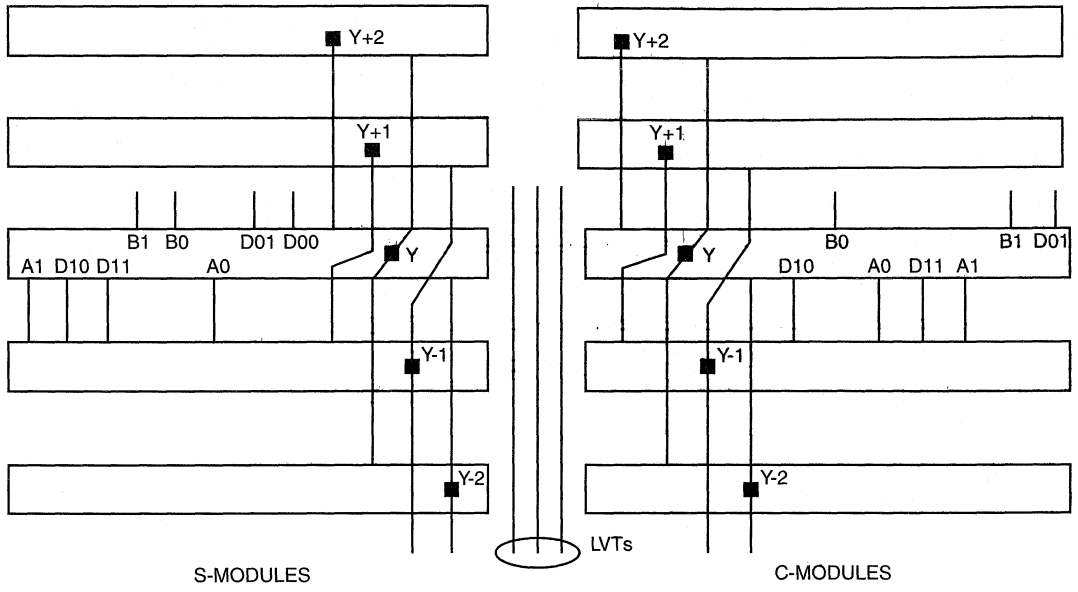


Figure 9 • Logic Module Routing Interface

Absolute Maximum Ratings[1]

Free air temperature range

Symbol	Parameter	Limits	Units
V_{CC}	DC Supply Voltage[2]	–0.5 to +7.0	V
V_I	Input Voltage	–0.5 to V_{CC} +0.5	V
V_O	Output Voltage	–0.5 to V_{CC} +0.5	V
I_{IO}	I/O Source Sink Current[3]	±20	mA
T_{STG}	Storage Temperature	–65 to +150	°C

Notes:

1. *Stresses beyond those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. Exposure to absolute maximum rated conditions for extended periods may affect device reliability. Device should not be operated outside the Recommended Operating Conditions.*
2. *V_{PP}, $V_{SV} = V_{CC}$, except during device programming.*
3. *Device inputs are normally high impedance and draw extremely low current. However, when input voltage is greater than V_{CC} + 0.5 V or less than GND – 0.5 V, the internal protection diodes will forward bias and can draw excessive current.*

Recommended Operating Conditions

Parameter	Commercial	Industrial	Military	Units
Temperature Range[1]	0 to +70	–40 to +85	–55 to +125	°C
Power Supply Tolerance	±5	±10	±10	%V_{CC}

Note:

1. *Ambient temperature (T_A) is used for commercial and industrial; case temperature (T_C) is used for military.*

Electrical Specifications

Symbol	Parameter	Test Condition	Commercial		Industrial		Military		Units
			Min.	Max.	Min.	Max.	Min.	Max.	
V_{OH}[1,2]	HIGH Level Output	I_{OH} = –4 mA (CMOS)			3.7		3.7		V
		I_{OH} = –6 mA (CMOS)	3.84						V
		I_{OH} = –10 mA (TTL)[3]	2.40						V
V_{OL}[1,2]	LOW Level Output	I_{OL} = +6 mA (CMOS)		0.33		0.4		0.4	V
		I_{OL} = +12 mA (TTL)[3]		0.50					V
V_{IH}	HIGH Level Input	TTL Inputs	2.0	V_{CC} + 0.3	2.0	V_{CC} + 0.3	2.0	V_{CC} + 0.3	V
V_{IL}	LOW Level Input	TTL Inputs	–0.3	0.8	–0.3	0.8	–0.3	0.8	V
I_{IN}	Input Leakage	$V_I = V_{CC}$ or GND	–10	+10	–10	+10	–10	+10	µA
I_{OZ}	3-state Output Leakage	$V_O = V_{CC}$ or GND	–10	+10	–10	+10	–10	+10	µA
C_{IO}	I/O Capacitance[3,4]			10		10		10	pF
$I_{CC(S)}$	Standby V_{CC} Supply Current	(typical = 0.7 mA)		2		10		20	mA
$I_{CC(D)}$	Dynamic V_{CC} Supply Current	See "Power Dissipation" Section							

Notes:

1. *Actel devices can drive and receive either CMOS or TTL signal levels. No assignment of I/Os as TTL or CMOS is required.*
2. *Tested one output at a time, V_{CC} = min.*
3. *Not tested, for information only.*
4. *V_{OUT} = 0V, f = 1 MHz.*
5. *Typical standby current = 0.7 mA. All outputs unloaded. All inputs = V_{CC} or GND.*

Package Thermal Characteristics

The device junction to case thermal characteristic is θjc, and the junction to ambient air characteristic is θja. The thermal characteristics for θja are shown with two different air flow rates.

Maximum junction temperature is 150°C.

A sample calculation of the absolute maximum power dissipation allowed for a CPGA 175-pin package at commercial temperature and still air is as follows:

$$\text{Absolute Maximum Power Allowed} = \frac{\text{Max. junction temp. (°C)} - \text{Max. ambient temp. (°C)}}{\theta ja\ (°C/W)} = \frac{150°C - 70°C}{25°C/W} = 3.2\ W$$

Package Type[1]	Pin Count	θja Still Air	θja 300 ft/min	Units
Ceramic Pin Grid Array	100	35	17	°C/W
	133	30	15	°C/W
	175	25	14	°C/W
	207	22	13	°C/W
	257	15	8	°C/W
Ceramic Quad Flatpack	132	55	30	°C/W
	196	36	24	°C/W
	256	30	18	°C/W
Plastic Quad Flatpack	100	51	40	°C/W
	160	33	26	°C/W
	208	33	26	°C/W
Very Thin Quad Flatpack	100	43	35	°C/W
Thin Quad Flatpack	176	32	25	°C/W
Power Quad Flatpack	208	17	13	°C/W
Plastic Leaded Chip Carrier	84	37	28	°C/W
Plastic Ball Grid Array	225	25	19	°C/W
	313	23	17	°C/W

Notes:

1. *Maximum Power Dissipation for 160-pin PQFP package is 2.4 Watts, 208-pin PQFP package is 2.4 Watts, 100-pin PQFP package is 1.6 Watts, 100-pin VQFP package is 1.9 Watts, 176-pin TQFP package is 2.5 Watts, 84-pin PLCC package is 2.2 Watts, 208-pin RQFP package is 4.7 Watts, 225-pin BGA package is 3.2 Watts, 313-pin BGA package is 3.5 Watts.*

Power Dissipation

$$P = [I_{CC\ standby} + I_{active}] * V_{CC} + I_{OL} * V_{OL} * N + I_{OH} * (V_{CC} - V_{OH}) * M \quad (1)$$

Where:

$I_{CC\ standby}$ is the current flowing when no inputs or outputs are changing.

I_{active} is the current flowing due to CMOS switching.

I_{OL}, I_{OH} are TTL sink/source currents.

V_{OL}, V_{OH} are TTL level output voltages.

N equals the number of outputs driving TTL loads to V_{OL}.

M equals the number of outputs driving TTL loads to V_{OH}.

An accurate determination of N and M is problematical because their values depend on the design and on the system I/O. The power can be divided into two components: static and active.

Static Power Component

Actel FPGAs have small static power components that result in lower power dissipation than PALs or PLDs. By integrating multiple PALs/PLDs into one FPGA, an even greater reduction in board-level power dissipation can be achieved.

The power due to standby current is typically a small component of the overall power. Standby power is calculated below for commercial, worst case conditions.

I_{CC}	V_{CC}	Power
2mA	5.25 V	10.5 mW

The static power dissipated by TTL loads depends on the number of outputs driving high or low and the DC load current. Again, this value is typically small. For instance, a 32-bit bus sinking 4 mA at 0.33 V will generate 42 mW with all outputs driving low, and 140 mW with all outputs driving high. The actual dissipation will average somewhere between as I/Os switch states with time.

Active Power Component

Power dissipation in CMOS devices is usually dominated by the active (dynamic) power dissipation. This component is frequency dependent, a function of the logic and the external I/O. Active power dissipation results from charging internal chip capacitances of the interconnect, unprogrammed antifuses, module inputs, and module outputs, plus external capacitance due to PC board traces and load device inputs. An additional component of the active power dissipation is the totem-pole current in CMOS transistor pairs. The net effect can be associated with an equivalent capacitance that can be combined with frequency and voltage to represent active power dissipation.

Equivalent Capacitance

The power dissipated by a CMOS circuit can be expressed by the Equation 2.

$$\text{Power (uW)} = C_{EQ} * V_{CC}^2 * F \quad (2)$$

Where:

C_{EQ} is the equivalent capacitance expressed in pF.

V_{CC} is the power supply in volts.

F is the switching frequency in MHz.

Equivalent capacitance is calculated by measuring I_{CC}active at a specified frequency and voltage for each circuit component of interest. Measurements have been made over a range of frequencies at a fixed value of V_{CC}. Equivalent capacitance is frequency independent so that the results may be used over a wide range of operating conditions. Equivalent capacitance values are shown below.

C_{EQ} Values for Actel FPGAs

Modules (C_{EQM})	6.7
Input Buffers (C_{EQI})	7.2
Output Buffers (C_{EQO})	10.4
Routed Array Clock Buffer Loads (C_{EQCR})	1.6
Dedicated Clock Buffer Loads (C_{EQCD})	0.7
I/O Clock Buffer Loads (C_{EQCI})	0.9

To calculate the active power dissipated from the complete design, the switching frequency of each part of the logic must be known. Equation 3 shows a piece-wise linear summation over all components.

$$\text{Power} = V_{CC}2 * [(m * C_{EQM} * f_m)_{modules} + (n * C_{EQI} * f_n)_{inputs} + (p * (C_{EQO} + C_L) * f_p)_{outputs} + 0.5 * (q_1 * C_{EQCR} * f_{q1})_{routed\_Clk1} + (r_1 * f_{q1})_{routed\_Clk1} + 0.5 * (q_2 * C_{EQCR} * f_{q2})_{routed\_Clk2} + (r_2 * f_{q2})_{routed\_Clk2} + 0.5 * (s_1 * CEQCD * f_{s1})_{dedicated\_Clk} + (s_2 * CEQCI * f_{s2})_{IO\_Clk}] \quad (3)$$

Where:

m = Number of logic modules switching at f_m

n = Number of input buffers switching at f_n

p = Number of output buffers switching at f_p

q_1 = Number of clock loads on the first routed array clock

q_2 = Number of clock loads on the second routed array clock

r_1 = Fixed capacitance due to first routed array clock

r_2 = Fixed capacitance due to second routed array clock

s_1 = Fixed number of clock loads on the dedicated array clock

s_2 = Fixed number of clock loads on the dedicated I/O clock

C_{EQM} = Equivalent capacitance of logic modules in pF

C_{EQI} = Equivalent capacitance of input buffers in pF

C_{EQO} = Equivalent capacitance of output buffers in pF

C_{EQCR} = Equivalent capacitance of routed array clock in pF

C_{EQCD} = Equivalent capacitance of dedicated array clock in pF

C_{EQCI} = Equivalent capacitance of dedicated I/O clock in pF

C_L = Output lead capacitance in pF

f_m = Average logic module switching rate in MHz

f_n = Average input buffer switching rate in MHz

fp = Average output buffer switching rate in MHz

f_{q1} = Average first routed array clock rate in MHz

f_{q2} = Average second routed array clock rate in MHz

f_{s1} = Average dedicated array clock rate in MHz

f_{s2} = Average dedicated I/O clock rate in MHz

Fixed Capacitance Values for Actel FPGAs (pF)

Device Type	r_1 routed_Clk1	r_2 routed_Clk2
A1415A	60	60
A1425A	75	75
A1440A	105	105
A1460A	165	165
A14100A	195	195

Fixed Clock Loads (s_1/s_2)

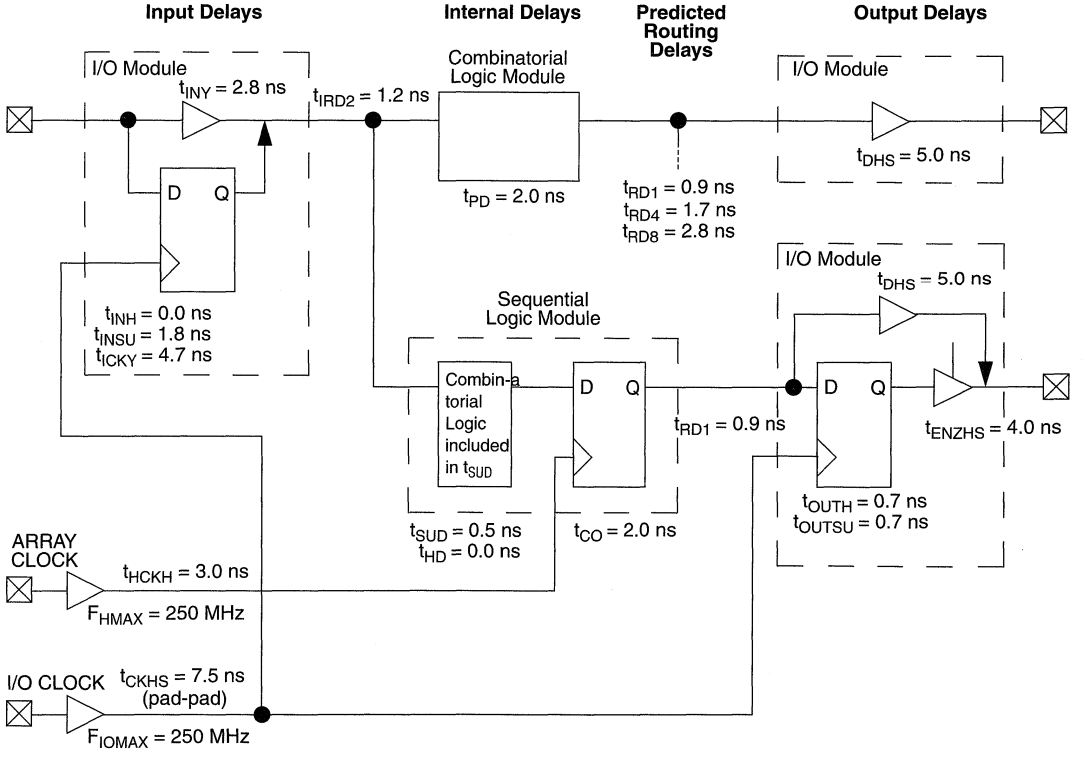
Device Type	s_1 Clock Loads on dedicated array clock	s_2 Clock Loads on dedicated I/O clock
A1415A	104	80
A1425A	160	100
A1440A	288	140
A1460A	432	168
A14100A	697	228

Determining Average Switching Frequency

To determine the switching frequency for a design, you must have a detailed understanding of the data input values to the circuit. The following guidelines are meant to represent worst-case scenarios so that they can be generally used to predict the upper limits of power dissipation. These guidelines are as follows:

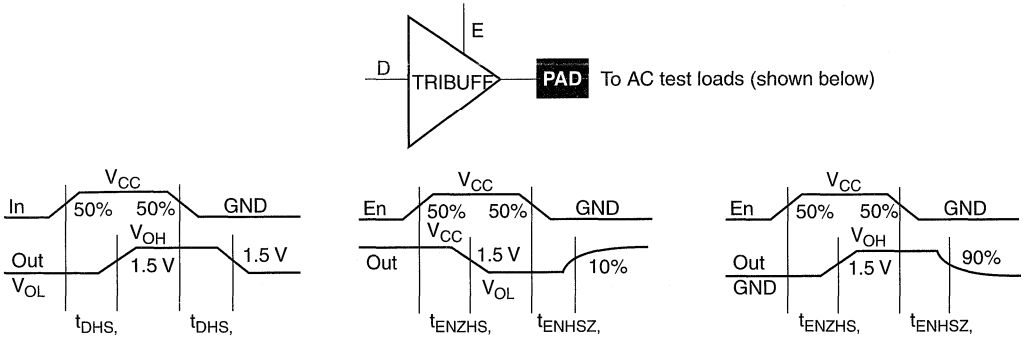
Logic Modules (m)	=	80% of modules
Inputs switching (n)	=	# inputs/4
Outputs switching (p)	=	# output/4
First routed array clock loads (q_1)	=	40% of sequential modules
Second routed array clock loads (q_2)	=	40% of sequential modules
Load capacitance (C_L)	=	35 pF
Average logic module switching rate (f_m)	=	F/10
Average input switching rate (f_n)	=	F/5
Average output switching rate (f_p)	=	F/10
Average first routed array clock rate (f_{q1})	=	F/2
Average second routed array clock rate (f_{q2})	=	F/2
Average dedicated array clock rate (f_{s1})	=	F
Average dedicated I/O clock rate (f_{s2})	=	F

ACT 3 Timing Model*

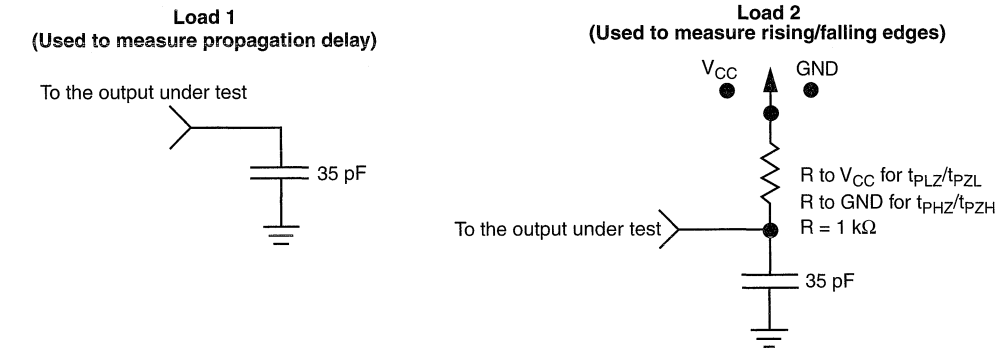


Values shown for A1425A-3.

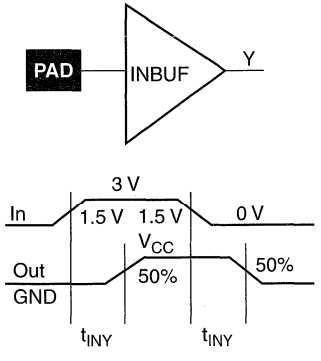
Output Buffer Delays



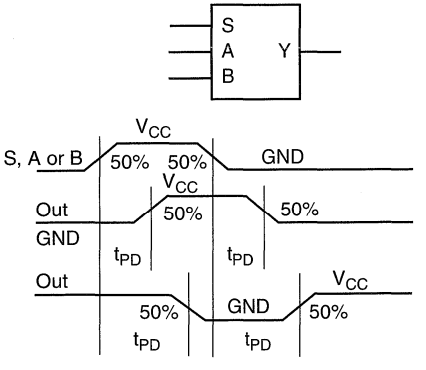
AC Test Loads



Input Buffer Delays

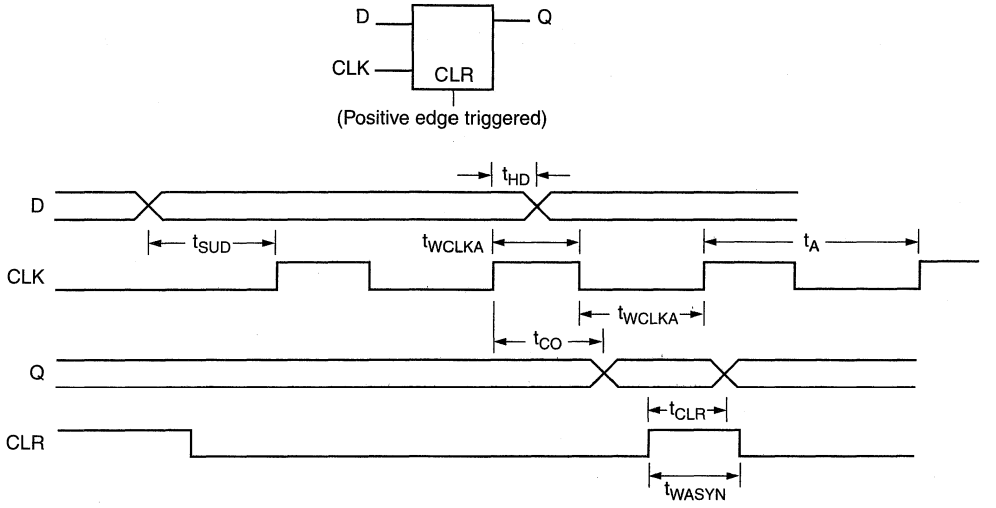


Module Delays

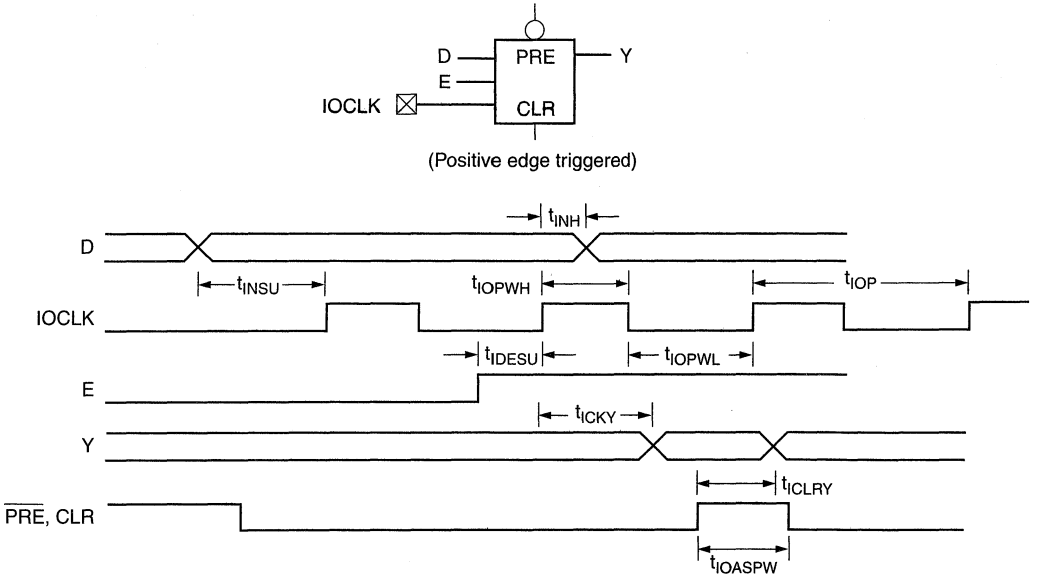


Sequential Module Timing Characteristics

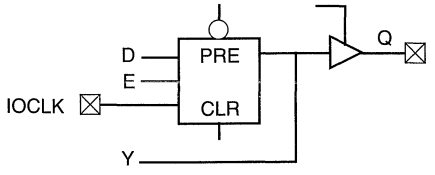
Flip-Flops



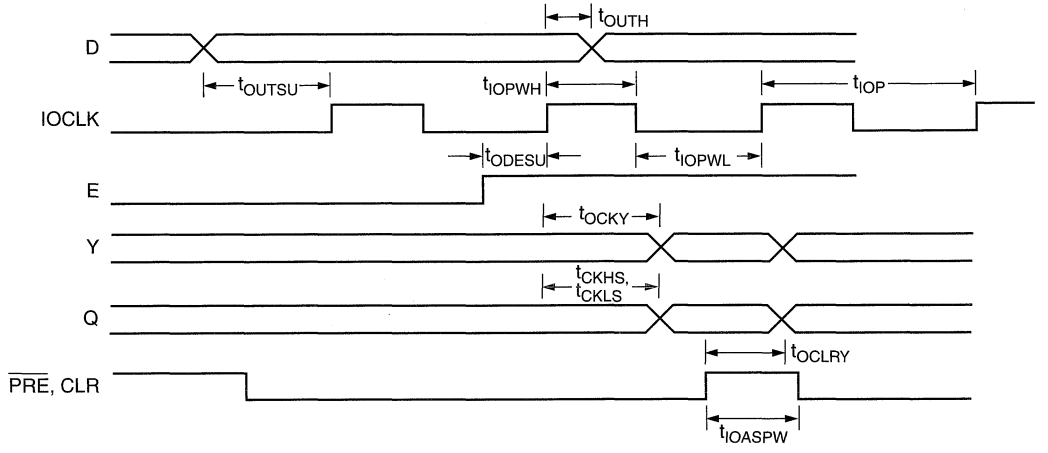
I/O Module: Sequential Input Timing Characteristics



I/O Module: Sequential Output Timing Characteristics



(Positive edge triggered)



Predictable Performance: Tightest Delay Distributions

Propagation delay between logic modules depends on the resistive and capacitive loading of the routing tracks, the interconnect elements, and the module inputs being driven. Propagation delay increases as the length of routing tracks, the number of interconnect elements, or the number of inputs increases.

From a design perspective, the propagation delay can be statistically correlated or modeled by the fanout (number of loads) driven by a module. Higher fanout usually requires some paths to have longer lengths of routing track.

The ACT 3 family delivers the tightest fanout delay distribution of any FPGA. This tight distribution is achieved in two ways: by decreasing the delay of the interconnect elements and by decreasing the number of interconnect elements per path.

Actel's patented PLICE antifuse offers a very low resistive/capacitive interconnect. The ACT 3 family's antifuses, fabricated in 0.8 micron m lithography, offer nominal levels of 200Ω resistance and 6 femtofarad (fF) capacitance per antifuse.

The ACT 3 fanout distribution is also tighter than alternative devices due to the low number of antifuses required per interconnect path. The ACT 3 family's proprietary architecture limits the number of antifuses per path to only four, with 90% of interconnects using only two antifuses.

Table 2 • *Logic Module + Routing Delay, by fanout (ns)*
(Worst-Case Commercial Conditions)

Family	FO=1	FO=2	FO=3	FO=4	FO=8
ACT 1 –2	4.5	5.1	5.9	7.0	11.1
ACT 2 –2	4.9	5.5	6.1	6.6	8.2
ACT 3 –3	2.9	3.2	3.4	3.7	4.8

The ACT 3 family's tight fanout delay distribution offers an FPGA design environment in which fanout can be traded for the increased performance of reduced logic level designs. This also simplifies performance estimates when designing with ACT 3 devices.

Timing Characteristics

Timing characteristics for ACT 3 devices fall into three categories: family dependent, device dependent, and design dependent. The input and output buffer characteristics are common to all ACT 3 family members. Internal routing delays are device dependent. Design dependency means actual delays are not determined until after placement and routing of the user's design is complete. Delay values may then be determined by using the ALS Timer utility or performing simulation with post-layout delays.

Critical Nets and Typical Nets

Propagation delays are expressed only for typical nets, which are used for initial design performance evaluation. Critical net delays can then be applied to the most time-critical paths. Critical nets are determined by net property assignment prior to placement and routing. Up to 6% of the nets in a design may be designated as critical, while 90% of the nets in a design are typical.

Long Tracks

Some nets in the design use long tracks. Long tracks are special routing resources that span multiple rows, columns, or modules. Long tracks employ three and sometimes four antifuse connections. This increases capacitance and resistance, resulting in longer net delays for macros connected to long tracks. Typically up to 6% of nets in a fully utilized device require long tracks. Long tracks contribute approximatley 4 ns to 14 ns delay. This additional delay is represented statistically in higher fanout (FO=8) routing delays in the data sheet specifications section.

Timing Derating

ACT 3 devices are manufactured in a CMOS process. Therefore, device performance varies according to temperature, voltage, and process variations. Minimum timing parameters reflect maximum operating voltage, minimum operating temperature, and best-case processing. Maximum timing parameters reflect minimum operating voltage, maximum operating temperature, and worst-case processing.

Timing Derating Factor (Temperature and Voltage)

	Industrial		Military	
	Min.	Max.	Min.	Max.
(Commercial Minimum/Maximum Specification) x	0.66	1.07	0.63	1.17

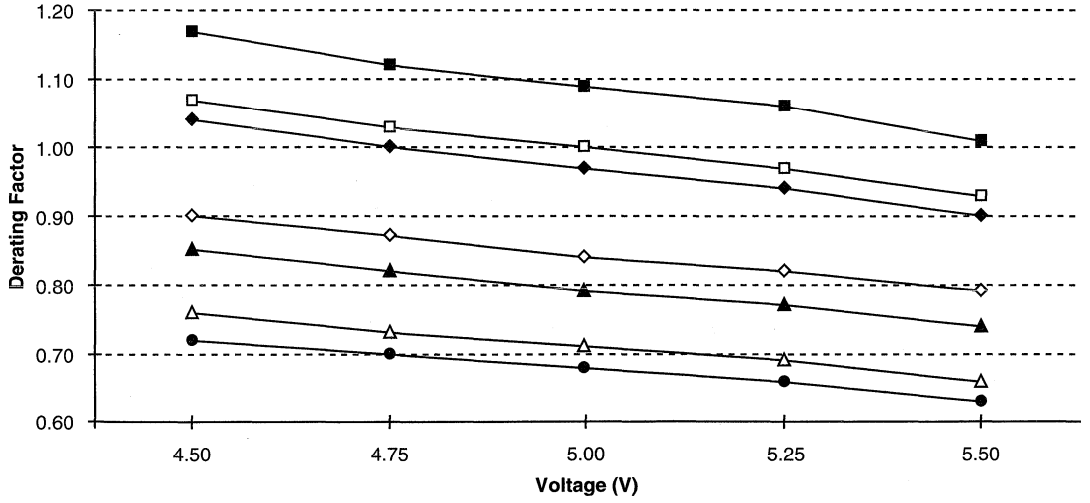
Timing Derating Factor for Designs at Typical Temperature (T_J = 25°C) and Voltage (5.0 V)

(Commercial Maximum Specification) x	0.85

Temperature and Voltage Derating Factors (normalized to Worst-Case Commercial, T_J = 4.75 V, 70°C)

	–55	–40	0	25	70	85	125
4.50	0.72	0.76	0.85	0.90	1.04	1.07	1.17
4.75	0.70	0.73	0.82	0.87	1.00	1.03	1.12
5.00	0.68	0.71	0.79	0.84	0.97	1.00	1.09
5.25	0.66	0.69	0.77	0.82	0.94	0.97	1.06
5.50	0.63	0.66	0.74	0.79	0.90	0.93	1.01

Junction Temperature and Voltage Derating Curves (normalized to Worst-Case Commercial, T_J = 4.75 V, 70°C)



Note: *This derating factor applies to all routing and propagation dealys.*

A1415A Timing Characteristics

(Worst-Case Commercial Conditions, V_{CC} = 4.75 V, T_J = 70°C)

								Preliminary Information		
Logic Module Propagation Delays[1]		**'Std' Speed**		**'–1' Speed**		**'–2' Speed**		**'–3' Speed**		
Parameter	**Description**	**Min.**	**Max.**	**Min.**	**Max.**	**Min.**	**Max.**	**Min.**	**Max.**	**Units**
t_{PD}	Internal Array Module		3.0		2.6		2.3		2.0	ns
t_{CO}	Sequential Clock to Q		3.0		2.6		2.3		2.0	ns
t_{CLR}	Asynchronous Clear to Q		3.0		2.6		2.3		2.0	ns
Predicted Routing Delays[2]										
t_{RD1}	FO=1 Routing Delay		1.3		1.1		1.0		0.9	ns
t_{RD2}	FO=2 Routing Delay		1.8		1.6		1.4		1.2	ns
t_{RD3}	FO=3 Routing Delay		2.1		1.8		1.6		1.4	ns
t_{RD4}	FO=4 Routing Delay		2.5		2.2		1.9		1.7	ns
t_{RD8}	FO=8 Routing Delay		4.2		3.6		3.2		2.8	ns
Logic Module Sequential Timing										
t_{SUD}	Flip-Flop Data Input Setup	0.8		0.7		0.6		0.5		ns
t_{HD}	Flip-Flop Data Input Hold	0.0		0.0		0.0		0.0		ns
t_{SUD}	Latch Data Input Setup	0.8		0.7		0.6		0.5		ns
t_{HD}	Latch Data Input Hold	0.0		0.0		0.0		0.0		ns
t_{WASYN}	Asynchronous Pulse Width	3.8		3.2		2.4		1.9		ns
t_{WCLKA}	Flip-Flop Clock Pulse Width	3.8		3.2		2.4		1.9		ns
t_A	Flip-Flop Clock Input Period	8.0		6.8		5.0		4.0		ns
f_{MAX}	Flip-Flop Clock Frequency		125		150		200		250	MHz

Notes:

1. *For dual-module macros, use $t_{PD} + t_{RD1} + t_{PDn}$, $t_{CO} + t_{RD1} + t_{PDn}$ or $t_{PD1} + t_{RD1} + t_{SUD}$, whichever is appropriate.*
2. *Routing delays are for typical designs across worst-case operating conditions. These parameters should be used for estimating device performance. Post-route timing analysis or simulation is required to determine actual worst-case performance. Post-route timing is based on actual routing delay measurements performed on the device prior to shipment.*

A1415A Timing Characteristics (continued)

(Worst-Case Commercial Conditions)

							Preliminary Information		
I/O Module Input Propagation Delays		**'Std' Speed**		**'–1' Speed**		**'–2' Speed**		**'–3' Speed**	
Parameter	**Description**	**Min.**	**Max.**	**Min.**	**Max.**	**Min.**	**Max.**	**Min.**	**Max.**	**Units**
t_{INY}	Input Data Pad to Y		4.2		3.6		3.2		2.8	ns
t_{ICKY}	Input Reg IOCLK Pad to Y		7.0		6.0		5.3		4.7	ns
t_{OCKY}	Output Reg IOCLK Pad to Y		7.0		6.0		5.3		4.7	ns
t_{ICLRY}	Input Asynchronous Clear to Y		7.0		6.0		5.3		4.7	ns
t_{OCLRY}	Output Asynchronous Clear to Y		7.0		6.0		5.3		4.7	ns
Predicted Input Routing Delays[1]										
t_{IRD1}	FO=1 Routing Delay		1.3		1.1		1.0		0.9	ns
t_{IRD2}	FO=2 Routing Delay		1.8		1.6		1.4		1.2	ns
t_{IRD3}	FO=3 Routing Delay		2.1		1.8		1.6		1.4	ns
t_{IRD4}	FO=4 Routing Delay		2.5		2.2		1.9		1.7	ns
t_{IRD8}	FO=8 Routing Delay		4.2		3.6		3.2		2.8	ns
I/O Module Sequential Timing										
t_{INH}	Input F-F Data Hold (w.r.t. IOCLK Pad)	0.0		0.0		0.0		0.0		ns
t_{INSU}	Input F-F Data Setup (w.r.t. IOCLK Pad)	3.0		2.5		2.3		2.0		ns
t_{IDEH}	Input Data Enable Hold (w.r.t. IOCLK Pad)	0.0		0.0		0.0		0.0		ns
t_{IDESU}	Input Data Enable Setup (w.r.t. IOCLK Pad)	8.6		7.5		6.5		5.8		ns
t_{OUTH}	Output F-F Data Hold (w.r.t. IOCLK Pad)	1.0		0.9		0.8		0.7		ns
t_{OUTSU}	Output F-F Data Setup (w.r.t. IOCLK Pad)	1.0		0.9		0.8		0.7		ns
t_{ODEH}	Output Data Enable Hold (w.r.t. IOCLK Pad)	0.5		0.4		0.4		0.3		ns
t_{ODESU}	Output Data Enable Setup (w.r.t. IOCLK Pad)	2.0		1.7		1.5		1.3		ns

Note:

1. *Routing delays are for typical designs across worst-case operating conditions. These parameters should be used for estimating device performance. Post-route timing analysis or simulation is required to determine actual worst-case performance. Post-route timing is based on actual routing delay measurements performed on the device prior to shipment.*

A1415A Timing Characteristics (continued)

(Worst-Case Commercial Conditions)

								Preliminary Information		
I/O Module – TTL Output Timing[1]		**'Std' Speed**		**'–1' Speed**		**'–2' Speed**		**'–3' Speed**		
Parameter	**Description**	**Min.**	**Max.**	**Min.**	**Max.**	**Min.**	**Max.**	**Min.**	**Max.**	**Units**
t_{DHS}	Data to Pad, High Slew		7.5		6.4		5.6		5.0	ns
t_{DLS}	Data to Pad, Low Slew		12.0		10.2		9.0		8.0	ns
t_{ENZHS}	Enable to Pad, Z to H/L, Hi Slew		6.0		5.1		4.5		4.0	ns
t_{ENZLS}	Enable to Pad, Z to H/L, Lo Slew		11.0		9.4		8.3		7.4	ns
t_{ENHSZ}	Enable to Pad, H/L to Z, Hi Slew		10.0		8.5		7.5		6.5	ns
t_{ENLSZ}	Enable to Pad, H/L to Z, Lo Slew		10.0		8.5		7.5		6.5	ns
t_{CKHS}	IOCLK Pad to Pad H/L, Hi Slew		10.0		9.0		7.5		7.5	ns
t_{CKLS}	IOCLK Pad to Pad H/L, Lo Slew		15.0		13.5		11.3		11.3	ns
d_{TLHHS}	Delta Low to High, Hi Slew		0.03		0.03		0.02		0.02	ns/pF
d_{TLHLS}	Delta Low to High, Lo Slew		0.07		0.06		0.05		0.05	ns/pF
d_{THLHS}	Delta High to Low, Hi Slew		0.05		0.04		0.04		0.04	ns/pF
d_{THLLS}	Delta High to Low, Lo Slew		0.07		0.06		0.05		0.05	ns/pF
I/O Module – CMOS Output Timing[1]										
t_{DHS}	Data to Pad, High Slew		9.3		7.9		7.0		6.2	ns
t_{DLS}	Data to Pad, Low Slew		17.5		14.9		13.1		11.7	ns
t_{ENZHS}	Enable to Pad, Z to H/L, Hi Slew		7.8		6.6		5.9		5.2	ns
t_{ENZLS}	Enable to Pad, Z to H/L, Lo Slew		13.3		11.3		10.0		8.9	ns
t_{ENHSZ}	Enable to Pad, H/L to Z, Hi Slew		10.0		8.5		7.5		6.7	ns
t_{ENLSZ}	Enable to Pad, H/L to Z, Lo Slew		10.0		9.0		7.5		6.7	ns
t_{CKHS}	IOCLK Pad to Pad H/L, Hi Slew		11.8		10.7		8.9		8.9	ns
t_{CKLS}	IOCLK Pad to Pad H/L, Lo Slew		17.3		15.6		13.0		13.0	ns
d_{TLHHS}	Delta Low to High, Hi Slew		0.06		0.05		0.04		0.04	ns/pF
d_{TLHLS}	Delta Low to High, Lo Slew		0.11		0.09		0.08		0.07	ns/pF
d_{THLHS}	Delta High to Low, Hi Slew		0.04		0.03		0.03		0.03	ns/pF
d_{THLLS}	Delta High to Low, Lo Slew		0.05		0.04		0.04		0.04	ns/pF

Note:

1. Delays based on 35pF loading.

A1415A Timing Characteristics (continued)

(Worst-Case Commercial Conditions)

								Preliminary Information		
Dedicated (Hard-Wired) I/O Clock Network		**'Std' Speed**		**'–1' Speed**		**'–2' Speed**		**'–3' Speed**		
Parameter	**Description**	**Min.**	**Max.**	**Min.**	**Max.**	**Min.**	**Max.**	**Min.**	**Max.**	**Units**
t_{IOCKH}	Input Low to High (Pad to I/O Module Input)		3.0		2.6		2.3		2.0	ns
t_{IOPWH}	Minimum Pulse Width High	3.8		3.3		2.4		1.9		ns
t_{IOPWL}	Minimum Pulse Width Low	3.8		3.3		2.4		1.9		ns
t_{IOSAPW}	Minimum Asynchronous Pulse Width	3.8		3.3		2.4		1.9		ns
t_{IOCKSW}	Maximum Skew		0.4		0.4		0.4		0.4	ns
t_{IOP}	Minimum Period	8.0		6.8		5.0		4.0		ns
f_{IOMAX}	Maximum Frequency		125		150		200		250	MHz
Dedicated (Hard-Wired) Array Clock Network										
t_{HCKH}	Input Low to High (Pad to S-Module Input)		4.5		3.9		3.4		3.0	ns
t_{HCKL}	Input High to Low (Pad to S-Module Input)		4.5		3.9		3.4		3.0	ns
t_{HPWH}	Minimum Pulse Width High	3.8		3.3		2.4		1.9		ns
t_{HPWL}	Minimum Pulse Width Low	3.8		3.3		2.4		1.9		ns
t_{HCKSW}	Maximum Skew		0.3		0.3		0.3		0.3	ns
t_{HP}	Minimum Period	8.0		6.8		5.0		4.0		ns
f_{HMAX}	Maximum Frequency		125		150		200		250	MHz
Routed Array Clock Networks										
t_{RCKH}	Input Low to High (FO=64)		5.5		4.7		4.1		3.7	ns
t_{RCKL}	Input High to Low (FO=64)		6.0		5.1		4.5		4.0	ns
t_{RPWH}	Min. Pulse Width High (FO=64)	4.9		4.2		3.8		3.3		ns
t_{RPWL}	Min. Pulse Width Low (FO=64)	4.9		4.2		3.8		3.3		ns
t_{RCKSW}	Maximum Skew (FO=128)		1.0		0.9		0.8		0.7	ns
t_{RP}	Minimum Period (FO=64)	10.0		8.7		8.0		6.8		ns
f_{RMAX}	Maximum Frequency (FO=64)		100		115		125		150	MHz
Clock-to-Clock Skews										
$t_{IOHCKSW}$	I/O Clock to H-Clock Skew	0.0	2.2	0.0	2.0	0.0	1.8	0.0	1.7	ns
$t_{IORCKSW}$	I/O Clock to R-Clock Skew (FO = 64)	0.0	1.0	0.0	1.0	0.0	1.0	0.0	1.0	ns
t_{HRCKSW}	H-Clock to R-Clock Skew (FO = 64)	0.0	1.0	0.0	1.0	0.0	1.0	0.0	1.0	ns

Note:

1. Delays based on 35pF loading.

A1425A Timing Characteristics

(Worst-Case Commercial Conditions, V_{CC} = 4.75 V, T_J = 70°C)

								Preliminary Information		
Logic Module Propagation Delays[1]		**'Std' Speed**		**'–1' Speed**		**'–2' Speed**		**'–3' Speed**		
Parameter	**Description**	**Min.**	**Max.**	**Min.**	**Max.**	**Min.**	**Max.**	**Min.**	**Max.**	**Units**
t_{PD}	Internal Array Module		3.0		2.6		2.3		2.0	ns
t_{CO}	Sequential Clock to Q		3.0		2.6		2.3		2.0	ns
t_{CLR}	Asynchronous Clear to Q		3.0		2.6		2.3		2.0	ns
Predicted Routing Delays[2]										
t_{RD1}	FO=1 Routing Delay		1.3		1.1		1.0		0.9	ns
t_{RD2}	FO=2 Routing Delay		1.8		1.6		1.4		1.2	ns
t_{RD3}	FO=3 Routing Delay		2.1		1.8		1.6		1.4	ns
t_{RD4}	FO=4 Routing Delay		2.5		2.2		1.9		1.7	ns
t_{RD8}	FO=8 Routing Delay		4.2		3.6		3.2		2.8	ns
Logic Module Sequential Timing										
t_{SUD}	Flip-Flop Data Input Setup	0.8		0.7		0.6		0.5		ns
t_{HD}	Flip-Flop Data Input Hold	0.0		0.0		0.0		0.0		ns
t_{SUD}	Latch Data Input Setup	0.8		0.7		0.6		0.5		ns
t_{HD}	Latch Data Input Hold	0.0		0.0		0.0		0.0		ns
t_{WASYN}	Asynchronous Pulse Width	3.8		3.2		2.4		1.9		ns
t_{WCLKA}	Flip-Flop Clock Pulse Width	3.8		3.2		2.4		1.9		ns
t_A	Flip-Flop Clock Input Period	8.0		6.8		5.0		4.0		ns
f_{MAX}	Flip-Flop Clock Frequency		125		150		200		250	MHz

Notes:

1. *For dual-module macros, use $t_{PD} + t_{RD1} + t_{PDn}$, $t_{CO} + t_{RD1} + t_{PDn}$ or $t_{PD1} + t_{RD1} + t_{SUD}$, whichever is appropriate.*
2. *Routing delays are for typical designs across worst-case operating conditions. These parameters should be used for estimating device performance. Post-route timing analysis or simulation is required to determine actual worst-case performance. Post-route timing is based on actual routing delay measurements performed on the device prior to shipment.*

A1425A Timing Characteristics (continued)

(Worst-Case Commercial Conditions)

								Preliminary Information		
I/O Module Input Propagation Delays		**'Std' Speed**		**'–1' Speed**		**'–2' Speed**		**'–3' Speed**		
Parameter	**Description**	**Min.**	**Max.**	**Min.**	**Max.**	**Min.**	**Max.**	**Min.**	**Max.**	**Units**
t_{INY}	Input Data Pad to Y		4.2		3.6		3.2		2.8	ns
t_{ICKY}	Input Reg IOCLK Pad to Y		7.0		6.0		5.3		4.7	ns
t_{OCKY}	Output Reg IOCLK Pad to Y		7.0		6.0		5.3		4.7	ns
t_{ICLRY}	Input Asynchronous Clear to Y		7.0		6.0		5.3		4.7	ns
t_{OCLRY}	Output Asynchronous Clear to Y		7.0		6.0		5.3		4.7	ns
Predicted Input Routing Delays[1]										
t_{IRD1}	FO=1 Routing Delay		1.3		1.1		1.0		0.9	ns
t_{IRD2}	FO=2 Routing Delay		1.8		1.6		1.4		1.2	ns
t_{IRD3}	FO=3 Routing Delay		2.1		1.8		1.6		1.4	ns
t_{IRD4}	FO=4 Routing Delay		2.5		2.2		1.9		1.7	ns
t_{IRD8}	FO=8 Routing Delay		4.2		3.6		3.2		2.8	ns
I/O Module Sequential Timing										
t_{INH}	Input F-F Data Hold (w.r.t. IOCLK Pad)	0.0		0.0		0.0		0.0		ns
t_{INSU}	Input F-F Data Setup (w.r.t. IOCLK Pad)	2.7		2.3		2.0		1.8		ns
t_{IDEH}	Input Data Enable Hold (w.r.t. IOCLK Pad)	0.0		0.0		0.0		0.0		ns
t_{IDESU}	Input Data Enable Setup (w.r.t. IOCLK Pad)	8.6		7.5		6.5		5.8		ns
t_{OUTH}	Output F-F Data Hold (w.r.t. IOCLK Pad)	1.0		0.9		0.8		0.7		ns
t_{OUTSU}	Output F-F Data Setup (w.r.t. IOCLK Pad)	1.0		0.9		0.8		0.7		ns
t_{ODEH}	Output Data Enable Hold (w.r.t. IOCLK Pad)	0.5		0.4		0.4		0.3		ns
t_{ODESU}	Output Data Enable Setup (w.r.t. IOCLK Pad)	2.0		1.7		1.5		1.3		ns

Note:

1. *Routing delays are for typical designs across worst-case operating conditions. These parameters should be used for estimating device performance. Post-route timing analysis or simulation is required to determine actual worst-case performance. Post-route timing is based on actual routing delay measurements performed on the device prior to shipment.*

A1425A Timing Characteristics (continued)

(Worst-Case Commercial Conditions)

								Preliminary Information		
I/O Module – TTL Output Timing[1]		**'Std' Speed**		**'–1' Speed**		**'–2' Speed**		**'–3' Speed**		
Parameter	**Description**	**Min.**	**Max.**	**Min.**	**Max.**	**Min.**	**Max.**	**Min.**	**Max.**	**Units**
t_{DHS}	Data to Pad, High Slew		7.5		6.4		5.6		5.0	ns
t_{DLS}	Data to Pad, Low Slew		12.0		10.2		9.0		8.0	ns
t_{ENZHS}	Enable to Pad, Z to H/L, Hi Slew		6.0		5.1		4.5		4.0	ns
t_{ENZLS}	Enable to Pad, Z to H/L, Lo Slew		11.0		9.4		8.3		7.4	ns
t_{ENHSZ}	Enable to Pad, H/L to Z, Hi Slew		10.0		8.5		7.5		6.5	ns
t_{ENLSZ}	Enable to Pad, H/L to Z, Lo Slew		10.0		8.5		7.5		6.5	ns
t_{CKHS}	IOCLK Pad to Pad H/L, Hi Slew		10.0		9.0		7.5		7.5	ns
t_{CKLS}	IOCLK Pad to Pad H/L, Lo Slew		15.0		13.5		11.3		11.3	ns
d_{TLHHS}	Delta Low to High, Hi Slew		0.03		0.03		0.02		0.02	ns/pF
d_{TLHLS}	Delta Low to High, Lo Slew		0.07		0.06		0.05		0.05	ns/pF
d_{THLHS}	Delta High to Low, Hi Slew		0.05		0.04		0.04		0.04	ns/pF
d_{THLLS}	Delta High to Low, Lo Slew		0.07		0.06		0.05		0.05	ns/pF
I/O Module – CMOS Output Timing[1]										
t_{DHS}	Data to Pad, High Slew		9.3		7.9		7.0		6.2	ns
t_{DLS}	Data to Pad, Low Slew		17.5		14.9		13.1		11.7	ns
t_{ENZHS}	Enable to Pad, Z to H/L, Hi Slew		7.8		6.6		5.9		5.2	ns
t_{ENZLS}	Enable to Pad, Z to H/L, Lo Slew		13.3		11.3		10.0		8.9	ns
t_{ENHSZ}	Enable to Pad, H/L to Z, Hi Slew		10.0		8.5		7.5		6.7	ns
t_{ENLSZ}	Enable to Pad, H/L to Z, Lo Slew		10.0		9.0		7.5		6.7	ns
t_{CKHS}	IOCLK Pad to Pad H/L, Hi Slew		11.8		10.7		8.9		8.9	ns
t_{CKLS}	IOCLK Pad to Pad H/L, Lo Slew		17.3		15.6		13.0		13.0	ns
d_{TLHHS}	Delta Low to High, Hi Slew		0.06		0.05		0.04		0.04	ns/pF
d_{TLHLS}	Delta Low to High, Lo Slew		0.11		0.09		0.08		0.07	ns/pF
d_{THLHS}	Delta High to Low, Hi Slew		0.04		0.03		0.03		0.03	ns/pF
d_{THLLS}	Delta High to Low, Lo Slew		0.05		0.04		0.04		0.04	ns/pF

Note:

1. Delays based on 35pF loading.

A1425A Timing Characteristics (continued)

(Worst-Case Commercial Conditions)

								Preliminary Information		
Dedicated (Hard-Wired) I/O Clock Network		**'Std' Speed**		**'–1' Speed**		**'–2' Speed**		**'–3' Speed**		
Parameter	**Description**	**Min.**	**Max.**	**Min.**	**Max.**	**Min.**	**Max.**	**Min.**	**Max.**	**Units**
t_{IOCKH}	Input Low to High (Pad to I/O Module Input)		3.0		2.6		2.3		2.0	ns
t_{IOPWH}	Minimum Pulse Width High	3.8		3.3		2.4		1.9		ns
t_{IOPWL}	Minimum Pulse Width Low	3.8		3.3		2.4		1.9		ns
t_{IOSAPW}	Minimum Asynchronous Pulse Width	3.8		3.3		2.4		1.9		ns
t_{IOCKSW}	Maximum Skew		0.4		0.4		0.4		0.4	ns
t_{IOP}	Minimum Period	8.0		6.8		5.0		4.0		ns
f_{IOMAX}	Maximum Frequency		125		150		200		250	MHz
Dedicated (Hard-Wired) Array Clock Network										
t_{HCKH}	Input Low to High (Pad to S-Module Input)		4.5		3.9		3.4		3.0	ns
t_{HCKL}	Input High to Low (Pad to S-Module Input)		4.5		3.9		3.4		3.0	ns
t_{HPWH}	Minimum Pulse Width High	3.8		3.3		2.4		1.9		ns
t_{HPWL}	Minimum Pulse Width Low	3.8		3.3		2.4		1.9		ns
t_{HCKSW}	Maximum Skew		0.3		0.3		0.3		0.3	ns
t_{HP}	Minimum Period	8.0		6.8		5.0		4.0		ns
f_{HMAX}	Maximum Frequency		125		150		200		250	MHz
Routed Array Clock Networks										
t_{RCKH}	Input Low to High (FO=64)		5.5		4.7		4.1		3.7	ns
t_{RCKL}	Input High to Low (FO=64)		6.0		5.1		4.5		4.0	ns
t_{RPWH}	Min. Pulse Width High (FO=64)	4.9		4.2		3.8		3.3		ns
t_{RPWL}	Min. Pulse Width Low (FO=64)	4.9		4.2		3.8		3.3		ns
t_{RCKSW}	Maximum Skew (FO=128)		1.0		0.9		0.8		0.7	ns
t_{RP}	Minimum Period (FO=64)	10.0		8.7		8.0		6.8		ns
f_{RMAX}	Maximum Frequency (FO=64)		100		115		125		150	MHz
Clock-to-Clock Skews										
$t_{IOHCKSW}$	I/O Clock to H-Clock Skew	0.0	2.2	0.0	2.0	0.0	1.8	0.0	1.7	ns
$t_{IORCKSW}$	I/O Clock to R-Clock Skew									
	(FO = 64)	0.0	1.0	0.0	1.0	0.0	1.0	0.0	1.0	ns
	(FO = 80)	0.0	3.0	0.0	3.0	0.0	3.0	0.0	3.0	ns
t_{HRCKSW}	H-Clock to R-Clock Skew									
	(FO = 64)	0.0	1.0	0.0	1.0	0.0	1.0	0.0	1.0	ns
	(FO = 80)	0.0	3.0	0.0	3.0	0.0	3.0	0.0	3.0	ns

Note:

1. Delays based on 35pF loading.

A1440A Timing Characteristics

(Worst-Case Commercial Conditions, V_{CC} = 4.75 V, T_J = 70°C)

								Preliminary Information		
Logic Module Propagation Delays[1]		**'Std' Speed**		**'–1' Speed**		**'–2' Speed**		**'–3' Speed**		
Parameter	**Description**	**Min.**	**Max.**	**Min.**	**Max.**	**Min.**	**Max.**	**Min.**	**Max.**	**Units**
t_{PD}	Internal Array Module		3.0		2.6		2.3		2.0	ns
t_{CO}	Sequential Clock to Q		3.0		2.6		2.3		2.0	ns
t_{CLR}	Asynchronous Clear to Q		3.0		2.6		2.3		2.0	ns
Predicted Routing Delays[2]										
t_{RD1}	FO=1 Routing Delay		1.3		1.1		1.0		0.9	ns
t_{RD2}	FO=2 Routing Delay		1.8		1.6		1.4		1.2	ns
t_{RD3}	FO=3 Routing Delay		2.1		1.8		1.6		1.4	ns
t_{RD4}	FO=4 Routing Delay		2.5		2.2		1.9		1.7	ns
t_{RD8}	FO=8 Routing Delay		4.2		3.6		3.2		2.8	ns
Logic Module Sequential Timing										
t_{SUD}	Flip-Flop Data Input Setup	0.8		0.7		0.6		0.5		ns
t_{HD}	Flip-Flop Data Input Hold	0.0		0.0		0.0		0.0		ns
t_{SUD}	Latch Data Input Setup	0.8		0.7		0.6		0.5		ns
t_{HD}	Latch Data Input Hold	0.0		0.0		0.0		0.0		ns
t_{WASYN}	Asynchronous Pulse Width	3.8		3.2		2.4		1.9		ns
t_{WCLKA}	Flip-Flop Clock Pulse Width	3.8		3.2		2.4		1.9		ns
t_A	Flip-Flop Clock Input Period	8.0		6.8		5.0		4.0		ns
f_{MAX}	Flip-Flop Clock Frequency		125		150		200		250	MHz

Notes:

1. *For dual-module macros, use $t_{PD} + t_{RD1} + t_{PDn}$, $t_{CO} + t_{RD1} + t_{PDn}$ or $t_{PD1} + t_{RD1} + t_{SUD}$, whichever is appropriate.*
2. *Routing delays are for typical designs across worst-case operating conditions. These parameters should be used for estimating device performance. Post-route timing analysis or simulation is required to determine actual worst-case performance. Post-route timing is based on actual routing delay measurements performed on the device prior to shipment.*

A1440A Timing Characteristics (continued)

(Worst-Case Commercial Conditions)

										Preliminary Information		
I/O Module Input Propagation Delays		**'Std' Speed**		**'–1' Speed**		**'–2' Speed**		**'–3' Speed**				
Parameter	**Description**	**Min.**	**Max.**	**Min.**	**Max.**	**Min.**	**Max.**	**Min.**	**Max.**	**Units**
t_{INY}	Input Data Pad to Y		4.2		3.6		3.2		2.8	ns
t_{ICKY}	Input Reg IOCLK Pad to Y		7.0		6.0		5.3		4.7	ns
t_{OCKY}	Output Reg IOCLK Pad to Y		7.0		6.0		5.3		4.7	ns
t_{ICLRY}	Input Asynchronous Clear to Y		7.0		6.0		5.3		4.7	ns
t_{OCLRY}	Output Asynchronous Clear to Y		7.0		6.0		5.3		4.7	ns
Predicted Input Routing Delays[1]										
t_{IRD1}	FO=1 Routing Delay		1.3		1.1		1.0		0.9	ns
t_{IRD2}	FO=2 Routing Delay		1.8		1.6		1.4		1.2	ns
t_{IRD3}	FO=3 Routing Delay		2.1		1.8		1.6		1.4	ns
t_{IRD4}	FO=4 Routing Delay		2.5		2.2		1.9		1.7	ns
t_{IRD8}	FO=8 Routing Delay		4.2		3.6		3.2		2.8	ns
I/O Module Sequential Timing										
t_{INH}	Input F-F Data Hold (w.r.t. IOCLK Pad)	0.0		0.0		0.0		0.0		ns
t_{INSU}	Input F-F Data Setup (w.r.t. IOCLK Pad)	2.3		2.0		1.7		1.5		ns
t_{IDEH}	Input Data Enable Hold (w.r.t. IOCLK Pad)	0.0		0.0		0.0		0.0		ns
t_{IDESU}	Input Data Enable Setup (w.r.t. IOCLK Pad)	8.6		7.5		6.5		5.8		ns
t_{OUTH}	Output F-F Data Hold (w.r.t. IOCLK Pad)	1.0		0.9		0.8		0.7		ns
t_{OUTSU}	Output F-F Data Setup (w.r.t. IOCLK Pad)	1.0		0.9		0.8		0.7		ns
t_{ODEH}	Output Data Enable Hold (w.r.t. IOCLK Pad)	0.5		0.4		0.4		0.3		ns
t_{ODESU}	Output Data Enable Setup (w.r.t. IOCLK Pad)	2.0		1.7		1.5		1.3		ns

Note:

1. *Routing delays are for typical designs across worst-case operating conditions. These parameters should be used for estimating device performance. Post-route timing analysis or simulation is required to determine actual worst-case performance. Post-route timing is based on actual routing delay measurements performed on the device prior to shipment.*

A1440A Timing Characteristics (continued)

(Worst-Case Commercial Conditions)

								Preliminary Information		
I/O Module – TTL Output Timing[1]		**'Std' Speed**		**'–1' Speed**		**'–2' Speed**		**'–3' Speed**		
Parameter	**Description**	**Min.**	**Max.**	**Min.**	**Max.**	**Min.**	**Max.**	**Min.**	**Max.**	**Units**
t_{DHS}	Data to Pad, High Slew		7.5		6.4		5.6		5.0	ns
t_{DLS}	Data to Pad, Low Slew		12.0		10.2		9.0		8.0	ns
t_{ENZHS}	Enable to Pad, Z to H/L, Hi Slew		6.0		5.1		4.5		4.0	ns
t_{ENZLS}	Enable to Pad, Z to H/L, Lo Slew		11.0		9.4		8.3		7.4	ns
t_{ENHSZ}	Enable to Pad, H/L to Z, Hi Slew		11.0		9.4		8.3		7.4	ns
t_{ENLSZ}	Enable to Pad, H/L to Z, Lo Slew		11.0		9.4		8.3		7.4	ns
t_{CKHS}	IOCLK Pad to Pad H/L, Hi Slew		11.0		9.5		8.5		8.5	ns
t_{CKLS}	IOCLK Pad to Pad H/L, Lo Slew		15.0		13.5		11.3		11.3	ns
d_{TLHHS}	Delta Low to High, Hi Slew		0.03		0.03		0.02		0.02	ns/pF
d_{TLHLS}	Delta Low to High, Lo Slew		0.07		0.06		0.05		0.05	ns/pF
d_{THLHS}	Delta High to Low, Hi Slew		0.05		0.04		0.04		0.04	ns/pF
d_{THLLS}	Delta High to Low, Lo Slew		0.07		0.06		0.05		0.05	ns/pF
I/O Module – CMOS Output Timing[1]										
t_{DHS}	Data to Pad, High Slew		9.3		7.9		7.0		6.2	ns
t_{DLS}	Data to Pad, Low Slew		17.5		14.9		13.1		11.7	ns
t_{ENZHS}	Enable to Pad, Z to H/L, Hi Slew		7.8		6.6		5.9		5.2	ns
t_{ENZLS}	Enable to Pad, Z to H/L, Lo Slew		13.3		11.3		10.0		8.9	ns
t_{ENHSZ}	Enable to Pad, H/L to Z, Hi Slew		11.0		9.4		8.3		7.4	ns
t_{ENLSZ}	Enable to Pad, H/L to Z, Lo Slew		11.0		9.4		8.3		7.4	ns
t_{CKHS}	IOCLK Pad to Pad H/L, Hi Slew		11.8		10.1		9.0		9.0	ns
t_{CKLS}	IOCLK Pad to Pad H/L, Lo Slew		17.3		15.6		13.0		13.0	ns
d_{TLHHS}	Delta Low to High, Hi Slew		0.06		0.05		0.04		0.04	ns/pF
d_{TLHLS}	Delta Low to High, Lo Slew		0.11		0.09		0.08		0.07	ns/pF
d_{THLHS}	Delta High to Low, Hi Slew		0.04		0.03		0.03		0.03	ns/pF
d_{THLLS}	Delta High to Low, Lo Slew		0.05		0.04		0.04		0.04	ns/pF

Note:

1. Delays based on 35pF loading.

A1440A Timing Characteristics (continued)

(Worst-Case Commercial Conditions)

								Preliminary Information		
Dedicated (Hard-Wired) I/O Clock Network		**'Std' Speed**		**'–1' Speed**		**'–2' Speed**		**'–3' Speed**		
Parameter	**Description**	**Min.**	**Max.**	**Min.**	**Max.**	**Min.**	**Max.**	**Min.**	**Max.**	**Units**
t_{IOCKH}	Input Low to High (Pad to I/O Module Input)		3.0		2.6		2.3		2.0	ns
t_{IOPWH}	Minimum Pulse Width High	3.8		3.3		2.4		1.9		ns
t_{IOPWL}	Minimum Pulse Width Low	3.8		3.3		2.4		1.9		ns
t_{IOSAPW}	Minimum Asynchronous Pulse Width	3.8		3.3		2.4		1.9		ns
t_{IOCKSW}	Maximum Skew		0.4		0.4		0.4		0.4	ns
t_{IOP}	Minimum Period	8.0		6.8		5.0		4.0		ns
f_{IOMAX}	Maximum Frequency		125		150		200		250	MHz
Dedicated (Hard-Wired) Array Clock Network										
t_{HCKH}	Input Low to High (Pad to S-Module Input)		4.5		3.9		3.4		3.0	ns
t_{HCKL}	Input High to Low (Pad to S-Module Input)		4.5		3.9		3.4		3.0	ns
t_{HPWH}	Minimum Pulse Width High	3.8		3.3		2.4		1.9		ns
t_{HPWL}	Minimum Pulse Width Low	3.8		3.3		2.4		1.9		ns
t_{HCKSW}	Maximum Skew		0.3		0.3		0.3		0.3	ns
t_{HP}	Minimum Period	8.0		6.8		5.0		4.0		ns
f_{HMAX}	Maximum Frequency		125		150		200		250	MHz
Routed Array Clock Networks										
t_{RCKH}	Input Low to High (FO=64)		5.5		4.7		4.1		3.7	ns
t_{RCKL}	Input High to Low (FO=64)		6.0		5.1		4.5		4.0	ns
t_{RPWH}	Min. Pulse Width High (FO=64)	4.9		4.2		3.8		3.3		ns
t_{RPWL}	Min. Pulse Width Low (FO=64)	4.9		4.2		3.8		3.3		ns
t_{RCKSW}	Maximum Skew (FO=128)		1.0		0.9		0.8		0.7	ns
t_{RP}	Minimum Period (FO=64)	10.0		8.7		8.0		6.8		ns
f_{RMAX}	Maximum Frequency (FO=64)		100		115		125		150	MHz
Clock-to-Clock Skews										
$t_{IOHCKSW}$	I/O Clock to H-Clock Skew	0.0	2.2	0.0	2.0	0.0	1.8	0.0	1.7	ns
$t_{IORCKSW}$	I/O Clock to R-Clock Skew									
	(FO = 64)	0.0	1.0	0.0	1.0	0.0	1.0	0.0	1.0	ns
	(FO = 144)	0.0	3.0	0.0	3.0	0.0	3.0	0.0	3.0	ns
t_{HRCKSW}	H-Clock to R-Clock Skew									
	(FO = 64)	0.0	1.0	0.0	1.0	0.0	1.0	0.0	1.0	ns
	(FO = 144)	0.0	3.0	0.0	3.0	0.0	3.0	0.0	3.0	ns

Note:

1. Delays based on 35pF loading.

A1460A Timing Characteristics

(Worst-Case Commercial Conditions, V_{CC} = 4.75 V, T_J = 70°C)

							Preliminary information		
Logic Module Propagation Delays[1]		**'Std' Speed**		**'–1' Speed**		**'–2' Speed**		**'–3' Speed**	
Parameter	**Description**	**Min.**	**Max.**	**Min.**	**Max.**	**Min.**	**Max.**	**Min.**	**Max.**	**Units**
t_{PD}	Internal Array Module		3.0		2.6		2.3		2.0	ns
t_{CO}	Sequential Clock to Q		3.0		2.6		2.3		2.0	ns
t_{CLR}	Asynchronous Clear to Q		3.0		2.6		2.3		2.0	ns
Predicted Routing Delays[2]										
t_{RD1}	FO=1 Routing Delay		1.3		1.1		1.0		0.9	ns
t_{RD2}	FO=2 Routing Delay		1.8		1.6		1.4		1.2	ns
t_{RD3}	FO=3 Routing Delay		2.1		1.8		1.6		1.4	ns
t_{RD4}	FO=4 Routing Delay		2.5		2.2		1.9		1.7	ns
t_{RD8}	FO=8 Routing Delay		4.2		3.6		3.2		2.8	ns
Logic Module Sequential Timing										
t_{SUD}	Flip-Flop Data Input Setup	0.8		0.7		0.6		0.5		ns
t_{HD}	Flip-Flop Data Input Hold	0.0		0.0		0.0		0.0		ns
t_{SUD}	Latch Data Input Setup	0.8		0.7		0.6		0.5		ns
t_{HD}	Latch Data Input Hold	0.0		0.0		0.0		0.0		ns
t_{WASYN}	Asynchronous Pulse Width	4.8		3.8		3.2		2.4		ns
t_{WCLKA}	Flip-Flop Clock Pulse Width	4.8		3.8		3.2		2.4		ns
t_A	Flip-Flop Clock Input Period	10.0		8.0		6.8		5.0		ns
f_{MAX}	Flip-Flop Clock Frequency		100		125		150		200	MHz

Note:

1. *For dual-module macros, use $t_{PD} + t_{RD1} + t_{PDn}$, $t_{CO} + t_{RD1} + t_{PDn}$ or $t_{PD1} + t_{RD1} + t_{SUD}$, whichever is appropriate.*
2. *Routing delays are for typical designs across worst-case operating conditions. These parameters should be used for estimating device performance. Post-route timing analysis or simulation is required to determine actual worst-case performance. Post-route timing is based on actual routing delay measurements performed on the device prior to shipment.*

A1460A Timing Characteristics (continued)

(Worst-Case Commercial Conditions)

										Preliminary Information		
I/O Module Input Propagation Delays		**'Std' Speed**		**'–1' Speed**		**'–2' Speed**		**'–3' Speed**				
Parameter	**Description**	**Min.**	**Max.**	**Min.**	**Max.**	**Min.**	**Max.**	**Min.**	**Max.**			**Units**
t_{INY}	Input Data Pad to Y		4.2		3.6		3.2		2.8			ns
t_{ICKY}	Input Reg IOCLK Pad to Y		7.0		6.0		5.3		4.7			ns
t_{OCKY}	Output Reg IOCLK Pad to Y		7.0		6.0		5.3		4.7			ns
t_{ICLRY}	Input Asynchronous Clear to Y		7.0		6.0		5.3		4.7			ns
t_{OCLRY}	Output Asynchronous Clear to Y		7.0		6.0		5.3		4.7			ns
Predicted Input Routing Delays[1]												
t_{IRD1}	FO=1 Routing Delay		1.3		1.1		1.0		0.9			ns
t_{IRD2}	FO=2 Routing Delay		1.8		1.6		1.4		1.2			ns
t_{IRD3}	FO=3 Routing Delay		2.1		1.8		1.6		1.4			ns
t_{IRD4}	FO=4 Routing Delay		2.5		2.2		1.9		1.7			ns
t_{IRD8}	FO=8 Routing Delay		4.2		3.6		3.2		2.8			ns
I/O Module Sequential Timing												
t_{INH}	Input F-F Data Hold (w.r.t. IOCLK Pad)	0.0		0.0		0.0		0.0				ns
t_{INSU}	Input F-F Data Setup (w.r.t. IOCLK Pad)	2.0		1.8		1.5		1.3				ns
t_{IDEH}	Input Data Enable Hold (w.r.t. IOCLK Pad)	0.0		0.0		0.0		0.0				ns
t_{IDESU}	Input Data Enable Setup (w.r.t. IOCLK Pad)	8.6		7.5		6.5		5.8				ns
t_{OUTH}	Output F-F Data Hold (w.r.t. IOCLK Pad)	1.0		0.9		0.8		0.7				ns
t_{OUTSU}	Output F-F Data Setup (w.r.t. IOCLK Pad)	1.0		0.9		0.8		0.7				ns
t_{ODEH}	Output Data Enable Hold (w.r.t. IOCLK Pad)	0.5		0.4		0.4		0.3				ns
t_{ODESU}	Output Data Enable Setup (w.r.t. IOCLK Pad)	2.0		1.7		1.5		1.3				ns

Note:

1. *Routing delays are for typical designs across worst-case operating conditions. These parameters should be used for estimating device performance. Post-route timing analysis or simulation is required to determine actual worst-case performance. Post-route timing is based on actual routing delay measurements performed on the device prior to shipment.*

A1460A Timing Characteristics (continued)

(Worst-Case Commercial Conditions)

							Preliminary information		
I/O Module – TTL Output Timing[1]		**'Std' Speed**		**'–1' Speed**		**'–2' Speed**		**'–3' Speed**	
Parameter	**Description**	**Min.**	**Max.**	**Min.**	**Max.**	**Min.**	**Max.**	**Min.**	**Max.**	**Units**
t_{DHS}	Data to Pad, High Slew		7.5		6.4		5.6		5.0	ns
t_{DLS}	Data to Pad, Low Slew		12.0		10.2		9.0		8.0	ns
t_{ENZHS}	Enable to Pad, Z to H/L, Hi Slew		6.0		5.1		4.5		4.0	ns
t_{ENZLS}	Enable to Pad, Z to H/L, Lo Slew		11.0		9.4		8.3		7.4	ns
t_{ENHSZ}	Enable to Pad, H/L to Z, Hi Slew		11.6		9.9		8.7		7.8	ns
t_{ENLSZ}	Enable to Pad, H/L to Z, Lo Slew		11.0		9.4		8.3		7.4	ns
t_{CKHS}	IOCLK Pad to Pad H/L, Hi Slew		11.5		10.0		9.0		9.0	ns
t_{CKLS}	IOCLK Pad to Pad H/L, Lo Slew		17.0		15.3		12.8		12.8	ns
d_{TLHHS}	Delta Low to High, Hi Slew		0.03		0.03		0.02		0.02	ns/pF
d_{TLHLS}	Delta Low to High, Lo Slew		0.07		0.06		0.05		0.05	ns/pF
d_{THLHS}	Delta High to Low, Hi Slew		0.05		0.04		0.04		0.04	ns/pF
d_{THLLS}	Delta High to Low, Lo Slew		0.07		0.06		0.05		0.05	ns/pF
I/O Module – CMOS Output Timing[1]										
t_{DHS}	Data to Pad, High Slew		9.3		7.9		7.0		6.2	ns
t_{DLS}	Data to Pad, Low Slew		17.5		14.9		13.1		11.7	ns
t_{ENZHS}	Enable to Pad, Z to H/L, Hi Slew		7.8		6.6		5.9		5.2	ns
t_{ENZLS}	Enable to Pad, Z to H/L, Lo Slew		13.3		11.3		10.0		8.9	ns
t_{ENHSZ}	Enable to Pad, H/L to Z, Hi Slew		11.0		9.4		8.3		7.4	ns
t_{ENLSZ}	Enable to Pad, H/L to Z, Lo Slew		11.0		9.4		8.3		7.4	ns
t_{CKHS}	IOCLK Pad to Pad H/L, Hi Slew		13.8		12.1		10.4		10.4	ns
t_{CKLS}	IOCLK Pad to Pad H/L, Lo Slew		19.3		17.4		14.5		14.5	ns
d_{TLHHS}	Delta Low to High, Hi Slew		0.06		0.05		0.04		0.04	ns/pF
d_{TLHLS}	Delta Low to High, Lo Slew		0.11		0.09		0.08		0.07	ns/pF
d_{THLHS}	Delta High to Low, Hi Slew		0.04		0.03		0.03		0.03	ns/pF
d_{THLLS}	Delta High to Low, Lo Slew		0.05		0.04		0.04		0.04	ns/pF

Note:

1. Delays based on 35pF loading.

A1460A Timing Characteristics (continued)

(Worst-Case Commercial Conditions)

								Preliminary Information		
Dedicated (Hard-Wired) I/O Clock Network		**'Std' Speed**		**'–1' Speed**		**'–2' Speed**		**'–3' Speed**		
Parameter	**Description**	**Min.**	**Max.**	**Min.**	**Max.**	**Min.**	**Max.**	**Min.**	**Max.**	**Units**
t_{IOCKH}	Input Low to High (Pad to I/O Module Input)		3.5		3.0		2.6		2.3	ns
t_{IOPWH}	Minimum Pulse Width High	4.8		3.8		3.2		2.4		ns
t_{IOPWL}	Minimum Pulse Width Low	4.8		3.8		3.2		2.4		ns
t_{IOSAPW}	Minimum Asynchronous Pulse Width	4.8		3.8		3.2		2.4		ns
t_{IOCKSW}	Maximum Skew		0.6		0.6		0.6		0.6	ns
t_{IOP}	Minimum Period	10.0		8.0		6.8		5.0		ns
f_{IOMAX}	Maximum Frequency		100		125		150		200	MHz
Dedicated (Hard-Wired) Array Clock Network										
t_{HCKH}	Input Low to High (Pad to S-Module Input)		5.5		4.7		4.1		3.7	ns
t_{HCKL}	Input High to Low (Pad to S-Module Input)		5.5		4.7		4.1		3.7	ns
t_{HPWH}	Minimum Pulse Width High	4.8		3.8		3.2		2.4		ns
t_{HPWL}	Minimum Pulse Width Low	4.8		3.8		3.2		2.4		ns
t_{HCKSW}	Maximum Skew		0.6		0.6		0.6		0.6	ns
t_{HP}	Minimum Period	10.0		8.0		6.8		5.0		ns
f_{HMAX}	Maximum Frequency		100		125		150		200	MHz
Routed Array Clock Networks										
t_{RCKH}	Input Low to High (FO=256)		9.0		7.7		6.8		6.0	ns
t_{RCKL}	Input High to Low (FO=256)		9.0		7.7		6.8		6.0	ns
t_{RPWH}	Min. Pulse Width High (FO=256)	6.1		5.4		4.5		4.1		ns
t_{RPWL}	Min. Pulse Width Low (FO=256)	6.1		5.4		4.5		4.1		ns
t_{RCKSW}	Maximum Skew (FO=128)		1.8		1.6		1.4		1.2	ns
t_{RP}	Minimum Period (FO=256)	12.5		11.1		9.3		8.3		ns
f_{RMAX}	Maximum Frequency (FO=256)		80		90		105		120	MHz
Clock-to-Clock Skews										
$t_{IOHCKSW}$	I/O Clock to H-Clock Skew	0.0	3.0	0.0	2.9	0.0	2.7	0.0	2.6	ns
$t_{IORCKSW}$	I/O Clock to R-Clock Skew									
	(FO = 64)	0.0	1.7	0.0	1.7	0.0	1.7	0.0	1.7	ns
	(FO = 216)	0.0	5.0	0.0	5.0	0.0	5.0	0.0	5.0	ns
t_{HRCKSW}	H-Clock to R-Clock Skew									
	(FO = 64)	0.0	1.0	0.0	1.0	0.0	1.0	0.0	1.3	ns
	(FO = 216)	0.0	3.0	0.0	3.0	0.0	3.0	0.0	3.0	ns

Note:

1. Delays based on 35pF loading.

A14100A Timing Characteristics

(Worst-Case Commercial Conditions, V_{CC} = 4.75 V, T_J = 70°C)

								Preliminary information		
Logic Module Propagation Delays[1]		**'Std' Speed**		**'–1' Speed**		**'–2' Speed**		**'–3' Speed**		
Parameter	**Description**	**Min.**	**Max.**	**Min.**	**Max.**	**Min.**	**Max.**	**Min.**	**Max.**	**Units**
t_{PD}	Internal Array Module		3.0		2.6		2.3		2.0	ns
t_{CO}	Sequential Clock to Q		3.0		2.6		2.3		2.0	ns
t_{CLR}	Asynchronous Clear to Q		3.0		2.6		2.3		2.0	ns
Predicted Routing Delays[2]										
t_{RD1}	FO=1 Routing Delay		1.3		1.1		1.0		0.9	ns
t_{RD2}	FO=2 Routing Delay		1.8		1.6		1.4		1.2	ns
t_{RD3}	FO=3 Routing Delay		2.1		1.8		1.6		1.4	ns
t_{RD4}	FO=4 Routing Delay		2.5		2.2		1.9		1.7	ns
t_{RD8}	FO=8 Routing Delay		4.2		3.6		3.2		2.8	ns
Logic Module Sequential Timing										
t_{SUD}	Flip-Flop Data Input Setup	0.8		0.8		0.6		0.5		ns
t_{HD}	Flip-Flop Data Input Hold	0.5		0.5		0.0		0.0		ns
t_{SUD}	Latch Data Input Setup	0.8		0.8		0.6		0.5		ns
t_{HD}	Latch Data Input Hold	0.5		0.5		0.0		0.0		ns
t_{WASYN}	Asynchronous Pulse Width	4.8		3.8		3.2		2.4		ns
t_{WCLKA}	Flip-Flop Clock Pulse Width	4.8		3.8		3.2		2.4		ns
t_A	Flip-Flop Clock Input Period	10.0		8.0		6.8		5.0		ns
f_{MAX}	Flip-Flop Clock Frequency		100		125		150		200	MHz

Notes:

1. *For dual-module macros, use $t_{PD} + t_{RD1} + t_{PDn}$, $t_{CO} + t_{RD1} + t_{PDn}$ or $t_{PD1} + t_{RD1} + t_{SUD}$, whichever is appropriate.*
2. *Routing delays are for typical designs across worst-case operating conditions. These parameters should be used for estimating device performance. Post-route timing analysis or simulation is required to determine actual worst-case performance. Post-route timing is based on actual routing delay measurements performed on the device prior to shipment.*

A14100A Timing Characteristics (continued)

(Worst-Case Commercial Conditions)

											Preliminary Information		
I/O Module Input Propagation Delays		**'Std' Speed**		**'–1' Speed**		**'–2' Speed**		**'–3' Speed**					
Parameter	**Description**	**Min.**	**Max.**	**Min.**	**Max.**	**Min.**	**Max.**	**Min.**	**Max.**	**Units**			
t_{INY}	Input Data Pad to Y		4.2		3.6		3.2		2.8	ns			
t_{ICKY}	Input Reg IOCLK Pad to Y		7.0		6.0		5.3		4.7	ns			
t_{OCKY}	Output Reg IOCLK Pad to Y		7.0		6.0		5.3		4.7	ns			
t_{ICLRY}	Input Asynchronous Clear to Y		7.0		6.0		5.3		4.7	ns			
t_{OCLRY}	Output Asynchronous Clear to Y		7.0		6.0		5.3		4.7	ns			
Predicted Input Routing Delays[1]													
t_{IRD1}	FO=1 Routing Delay		1.3		1.1		1.0		0.9	ns			
t_{IRD2}	FO=2 Routing Delay		1.8		1.6		1.4		1.2	ns			
t_{IRD3}	FO=3 Routing Delay		2.1		1.8		1.6		1.4	ns			
t_{IRD4}	FO=4 Routing Delay		2.5		2.2		1.9		1.7	ns			
t_{IRD8}	FO=8 Routing Delay		4.2		3.6		3.2		2.8	ns			
I/O Module Sequential Timing													
t_{INH}	Input F-F Data Hold (w.r.t. IOCLK Pad)	0.0		0.0		0.0		0.0		ns			
t_{INSU}	Input F-F Data Setup (w.r.t. IOCLK Pad)	1.8		1.5		1.4		1.2		ns			
t_{IDEH}	Input Data Enable Hold (w.r.t. IOCLK Pad)	0.0		0.0		0.0		0.0		ns			
t_{IDESU}	Input Data Enable Setup (w.r.t. IOCLK Pad)	8.6		7.5		6.5		5.8		ns			
t_{OUTH}	Output F-F Data Hold (w.r.t. IOCLK Pad)	1.0		1.0		0.8		0.7		ns			
t_{OUTSU}	Output F-F Data Setup (w.r.t. IOCLK Pad)	1.0		1.0		0.8		0.7		ns			
t_{ODEH}	Output Data Enable Hold (w.r.t. IOCLK Pad)	0.5		0.5		0.4		0.3		ns			
t_{ODESU}	Output Data Enable Setup (w.r.t. IOCLK Pad)	2.0		2.0		1.5		1.3		ns			

Note:

1. *Routing delays are for typical designs across worst-case operating conditions. These parameters should be used for estimating device performance. Post-route timing analysis or simulation is required to determine actual worst-case performance. Post-route timing is based on actual routing delay measurements performed on the device prior to shipment.*

A14100A Timing Characteristics (continued)

(Worst-Case Commercial Conditions)

								Advanced information		
I/O Module – TTL Output Timing[1]		**'Std' Speed**		**'–1' Speed**		**'–2' Speed**		**'–3' Speed**		
Parameter	**Description**	**Min.**	**Max.**	**Min.**	**Max.**	**Min.**	**Max.**	**Min.**	**Max.**	**Units**
t_{DHS}	Data to Pad, High Slew		7.5		6.4		5.6		5.0	ns
t_{DLS}	Data to Pad, Low Slew		12.0		10.2		9.0		8.0	ns
t_{ENZHS}	Enable to Pad, Z to H/L, Hi Slew		6.0		5.1		4.5		4.0	ns
t_{ENZLS}	Enable to Pad, Z to H/L, Lo Slew		11.0		9.4		8.3		7.4	ns
t_{ENHSZ}	Enable to Pad, H/L to Z, Hi Slew		12.0		10.2		9.0		8.0	ns
t_{ENLSZ}	Enable to Pad, H/L to Z, Lo Slew		11.0		9.4		8.3		7.4	ns
t_{CKHS}	IOCLK Pad to Pad H/L, Hi Slew		12.0		10.5		9.5		9.5	ns
t_{CKLS}	IOCLK Pad to Pad H/L, Lo Slew		17.0		15.3		12.8		12.8	ns
d_{TLHHS}	Delta Low to High, Hi Slew		0.03		0.03		0.02		0.02	ns/pF
d_{TLHLS}	Delta Low to High, Lo Slew		0.07		0.06		0.05		0.05	ns/pF
d_{THLHS}	Delta High to Low, Hi Slew		0.05		0.04		0.04		0.04	ns/pF
d_{THLLS}	Delta High to Low, Lo Slew		0.07		0.06		0.05		0.05	ns/pF
I/O Module – CMOS Output Timing[1]										
t_{DHS}	Data to Pad, High Slew		9.3		7.9		7.0		6.2	ns
t_{DLS}	Data to Pad, Low Slew		17.5		14.9		13.1		11.7	ns
t_{ENZHS}	Enable to Pad, Z to H/L, Hi Slew		7.8		6.6		5.9		5.2	ns
t_{ENZLS}	Enable to Pad, Z to H/L, Lo Slew		13.3		11.3		10.0		8.9	ns
t_{ENHSZ}	Enable to Pad, H/L to Z, Hi Slew		12.0		10.0		9.0		8.0	ns
t_{ENLSZ}	Enable to Pad, H/L to Z, Lo Slew		11.0		9.4		8.3		7.4	ns
t_{CKHS}	IOCLK Pad to Pad H/L, Hi Slew		13.8		12.4		10.4		10.4	ns
t_{CKLS}	IOCLK Pad to Pad H/L, Lo Slew		19.3		17.4		14.5		14.5	ns
d_{TLHHS}	Delta Low to High, Hi Slew		0.06		0.05		0.04		0.04	ns/pF
d_{TLHLS}	Delta Low to High, Lo Slew		0.11		0.09		0.08		0.07	ns/pF
d_{THLHS}	Delta High to Low, Hi Slew		0.04		0.03		0.03		0.03	ns/pF
d_{THLLS}	Delta High to Low, Lo Slew		0.05		0.04		0.04		0.04	ns/pF

Note:

1. Delays based on 35pF loading.

A14100A Timing Characteristics (continued)

(Worst-Case Commercial Conditions)

								Preliminary Information		
Dedicated (Hard-Wired) I/O Clock Network		**'Std' Speed**		**'–1' Speed**		**'–2' Speed**		**'–3' Speed**		
Parameter	**Description**	**Min.**	**Max.**	**Min.**	**Max.**	**Min.**	**Max.**	**Min.**	**Max.**	**Units**
t_{IOCKH}	Input Low to High (Pad to I/O Module Input)		3.5		3.0		2.6		2.3	ns
t_{IOPWH}	Minimum Pulse Width High	4.8		3.8		3.3		2.4		ns
t_{IOPWL}	Minimum Pulse Width Low	4.8		3.8		3.3		2.4		ns
t_{IOSAPW}	Minimum Asynchronous Pulse Width	4.8		3.8		3.3		2.4		ns
t_{IOCKSW}	Maximum Skew		0.8		0.7		0.6		0.6	ns
t_{IOP}	Minimum Period	10.0		8.0		6.8		5.0		ns
f_{IOMAX}	Maximum Frequency		100		125		150		200	MHz
Dedicated (Hard-Wired) Array Clock Network										
t_{HCKH}	Input Low to High (Pad to S-Module Input)		5.5		4.7		4.1		3.7	ns
t_{HCKL}	Input High to Low (Pad to S-Module Input)		5.5		4.7		4.1		3.7	ns
t_{HPWH}	Minimum Pulse Width High	4.8		3.8		3.3		2.4		ns
t_{HPWL}	Minimum Pulse Width Low	4.8		3.8		3.3		2.4		ns
t_{HCKSW}	Maximum Skew		0.8		0.7		0.6		0.6	ns
t_{HP}	Minimum Period	10.0		8.0		6.8		5.0		ns
f_{HMAX}	Maximum Frequency		100		125		150		200	MHz
Routed Array Clock Networks										
t_{RCKH}	Input Low to High (FO=256)		9.0		7.7		6.8		6.0	ns
t_{RCKL}	Input High to Low (FO=256)		9.0		7.7		6.8		6.0	ns
t_{RPWH}	Min. Pulse Width High (FO=256)	6.1		5.4		4.5		4.1		ns
t_{RPWL}	Min. Pulse Width Low (FO=256)	6.1		5.4		4.5		4.1		ns
t_{RCKSW}	Maximum Skew (FO=128)		1.8		1.6		1.4		1..2	ns
t_{RP}	Minimum Period (FO=256)	12.5		11.1		9.3		8.3		ns
f_{RMAX}	Maximum Frequency (FO=256)		80		90		105		120	MHz
Clock-to-Clock Skews										
$t_{IOHCKSW}$	I/O Clock to H-Clock Skew	0.0	3.0	0.0	2.9	0.0	2.7	0.0	2.6	ns
$t_{IORCKSW}$	I/O Clock to R-Clock Skew (FO = 64)	0.0	1.7	0.0	1.7	0.0	17	0.0	1.7	ns
	(FO = 350)	0.0	5.0	0.0	5.0	0.0	5.0	0.0	5.0	
t_{HRCKSW}	H-Clock to R-Clock Skew (FO = 64)	0.0	1.0	0.0	1.0	0.0	1.0	0.0	1.3	ns
	(FO = 350)	0.0	3.0	0.0	3.0	0.0	3.0	0.0	3.0	

Note:

1. Delays based on 35pF loading.

Macro Library

Hard Macros—Combinatorial

Function	Macro	Description	Modules S	Modules C
ACT 3 Combinatorial Logic Module	CM8	Combinational Module (Full ACT 3 Logic Module)		1
ACT 3 Sequential Logic Module	DFM8A	4-bit D-Type Flip-Flop with Multiplexed Data, active low Clear, and active high clock	1	
	DFM8B	4-bit D-Type Flip-Flop with Multiplexed Data, active low Clear, and active low clock	1	
Adder	FA1A	1-bit adder, carry in and carry out active low, A-input active low		2
	FA1B	1-bit adder, carry in and carry out active low		2
	FA2A	2-bit adder, carry in and carry out active low, A0 and A1 inputs active low		2
	HA1	Half-Adder		2
	HA1A	Half-Adder with active low A-input		2
	HA1B	Half-Adder with active low carry out and sum		2
	HA1C	Half-Adder with active low carry out		2
AND	AND2	2-input AND		1
	AND2A	2-input AND with active low A-input		1
	AND2B	2-input AND with active low inputs		1
	AND3	3-input AND		1
	AND3A	3-input AND with active low A-input		1
	AND3B	3-input AND with active low A- and B-inputs		1
	AND3C	3-input AND with active low inputs		1
	AND4	4-input AND		1
	AND4A	4-input AND with active low A-input		1
	AND4B	4-input AND with active low A- and B-inputs		1
	AND4C	4-input AND with active low A-, B-, and C-inputs		1
	AND4D	4-input AND with active low inputs		2
	AND5B	5-input AND with active low A- and B-inputs		1
AND-OR	AO1	3-input AND-OR		1
	AO10	5-input AND-OR-AND		1
	AO11	3-input AND-OR		1
	AO1A	3-input AND-OR with active low A-input		1
	AO1B	3-input AND-OR with active low C-input		1
	AO1C	3-input AND-OR with active low A- and C-inputs		1
	AO1D	3-input AND-OR with active low A- and B-inputs		1
	AO1E	3-input AND-OR with active low inputs		1
	AO2	4-input AND-OR		1
	AO2A	4-input AND-OR with active low A-input		1
	AO2B	4-input AND-OR with active low A- and B-inputs		1
	AO2C	4-input AND-OR with active low A- and C-inputs		1
	AO2D	4-input AND-OR with active low A-, B-, and C-inputs		1
	AO2E	4-input AND-OR with active low inputs		1
	AO3	4-input AND-OR		1
	AO3A	4-input AND-OR		1
	AO3B	4-input AND-OR		1
	AO3C	4-input AND-OR		1
	AO4A	4-input AND-OR		1
	AO5A	4-input AND-OR		1
	AO6	2-wide 4-input AND-OR		1

Hard Macros—Combinatorial (Continued)

Function	Macro	Description	Modules S	Modules C
AND-OR	AO6A	2-wide 4-input AND-OR with active low D-input		1
	AO7	5-input AND-OR		1
	AO8	5-input AND-OR with active low C- and D-inputs		1
	AO9	5-input AND-OR		1
	AOI1	3-input AND-OR-INVERT		1
	AOI1A	3-input AND-OR-INVERT with active low A-input		1
	AOI1B	3-input AND-OR-INVERT with active low C-input		1
	AOI1C	3-input AND-OR-INVERT with active low A- and B-inputs		1
	AOI1D	3-input AND-OR-INVERT with active low inputs		1
	AOI2A	4-input AND-OR-INVERT with active low A-input		1
	AOI2B	4-input AND-OR-INVERT with active low A- and C-inputs		1
	AOI3A	4-input AND-OR-INVERT with active low inputs		1
	AOI4	2-wide 4-input AND-OR-INVERT		2
	AOI4A	2-wide 4-input AND-OR-INVERT with active low C-input		1
AND-XOR	AX1	3-input AND-XOR with active low A-input		1
	AX1A	3-input AND-XOR-INVERT with active low A-input		2
	AX1B	3-input AND-XOR with active low A- and B-inputs		1
	AX1C	3-input AND-XOR		1
Buffer	BUF	Buffer, with active high input and output		1
	BUFA	Buffer, with active low input and output		1
Clock Net	CLKINT	Clock Net Interface	0	0
	GAND2	2-input AND Clock Net		1
	GMX4	4-to-1 Multiplexor Clock Net		1
	GNAND2	2-input NAND Clock Net		1
	GNOR2	2-input NOR Clock Net		1
	GOR2	2-input OR Clock Net		1
	GXOR2	2-input Exclusive OR Clock Net		1
Inverter	INV	Inverter with active low output		1
	INVA	Inverter with active low input		1
Majority	MAJ3	3-input complex AND-OR		1
MUX	MX2	2-to-1 Multiplexor		1
	MX2A	2-to-1 Multiplexor with active low A-input		1
	MX2B	2-to-1 Multiplexor with active low B-input		1
MUX	MX2C	2-to-1 Multiplexor with active low output		1
	MX4	4-to-1 Multiplexor		1
	MXC1	Boolean		2
	MXT	Boolean		2
NAND	NAND2	2-input NAND		1
	NAND2A	2-input NAND with active low A-input		1
	NAND2B	2-input NAND with active low inputs		1
	NAND3	3-input NAND		1
	NAND3A	3-input NAND with active low A-input		1
	NAND3B	3-input NAND with active low A- and B-inputs		1
	NAND3C	3-input NAND with active low inputs		1
	NAND4	4-input NAND		2
	NAND4A	4-input NAND with active low A-input		1
	NAND4B	4-input NAND with active low A- and B-inputs		1
	NAND4C	4-input NAND with active low A-, B-, and C-inputs		1
	NAND4D	4-input NAND with active low inputs		1

Hard Macros—Combinatorial (Continued)

Function	Macro	Description	Modules S	Modules C
NAND	NAND5C	5-input NAND with active low A-, B-, and C-inputs		1
NOR	NOR2	2-input NOR		1
NOR	NOR2A	2-input NOR with active low A-input		1
	NOR2B	2-input NOR with active low inputs		1
	NOR3	3-input NOR		1
	NOR3A	3-input NOR with active low A-input		1
	NOR3B	3-input NOR with active low A- and B-inputs		1
	NOR3C	3-input NOR with active low inputs		1
	NOR4	4-input NOR		2
	NOR4A	4-input NOR with active low A-input		1
	NOR4B	4-input NOR with active low A- and B-inputs		1
	NOR4C	4-input NOR with active low A-, B-, and C-inputs		1
	NOR4D	4-input NOR with active low inputs		1
	NOR5C	5-input NOR with active low A-, B-, and C-inputs		1
OR	OR2	2-input OR		1
	OR2A	2-input OR with active low A-input		1
	OR2B	2-input OR with active low inputs		1
	OR3	3-input OR		1
	OR3A	3-input OR with active low A-input		1
	OR3B	3-input OR with active low A- and B-inputs		1
	OR3C	3-input OR with active low inputs		1
	OR4	4-input OR		1
	OR4A	4-input OR with active low A-input		1
	OR4B	4-input OR with active low A- and B-input		1
	OR4C	4-input OR with active low A-, B-, and C-inputs		1
	OR4D	4-input OR with active low inputs		2
	OR5B	5-input OR with active low A- and B-inputs		1
OR-AND	OA1	3-input OR-AND		1
	OA1A	3-input OR-AND with active low A-input		1
	OA1B	3-input OR-AND with active low C-input		1
	OA1C	3-input OR-AND with active low A- and C-inputs		1
	OA2	2-wide 4-input OR-AND		1
	OA2A	2 wide 4-input OR-AND with active low A-input		1
	OA3	4-input OR-AND		1
	OA3A	4-input OR-AND with active low C-input		1
	OA3B	4-input OR-AND with active low A- and C-inputs		1
	OA4	4-input OR-AND		1
	OA4A	4-input OR-AND with active low C-input		1
	OA5	4-input complex OR-AND		1
	OAI1	3-input OR-AND-INVERT		1
	OAI2A	4-input OR-AND-INVERT with active low D-input		1
	OAI3	4-input OR-AND-INVERT		1
	OAI3A	4-input OR-AND-INVERT with active low C- and D-inputs		1
XNOR	XNOR	2-input XNOR		1
XNOR-AND	XA1A	3-input XNOR-AND		1
XNOR-OR	XO1A	3-input XNOR-OR		1
XOR	XOR	2-input XOR		1
XOR-AND	XA1	3-input XOR-AND		1
XOR-OR	XO1	3-input XOR-OR		1

Hard Macros—Sequential

Function	Macro	Description	Modules S	Modules C
D-Type	DF1	D-Type Flip-Flop	1	
	DF1A	D-Type Flip-Flop with active low output	1	
	DF1B	D-Type Flip-Flop with active low clock	1	
	DF1C	D-Type Flip-Flop with active low clock and output	1	
	DFC1	D-Type Flip-Flop with active high Clear	1	1
	DFC1A	D-Type Flip-Flop with active high Clear and active low clock	1	1
	DFC1B	D-Type Flip-Flop with active low Clear	1	
	DFC1D	D-Type Flip-Flop with active low Clear and clock	1	
	DFE	D-Type Flip-Flop with active high Enable	1	
	DFE1B	D-Type Flip-Flop with active low Enable	1	
	DFE1C	D-Type Flip-Flop with active low Enable and clock	1	
	DFE3A	D-Type Flip-Flop with Enable and active low Clear	1	
	DFE3B	D-Type Flip-Flop with Enable and active low Clear and clock	1	
	DFE3C	D-Type Flip-Flop with active low Enable and Clear	1	
	DFE3D	D-Type Flip-Flop with active low Enable, Clear, and clock	1	
	DFEA	D-Type Flip-Flop with Enable and active low clock	1	
	DFM	2-bit D-Type Flip-Flop with Multiplexed Data	1	
	DFM1B	2-bit D-Type Flip-Flop with Multiplexed Data and active low output	1	
	DFM1C	2-bit D-Type Flip-Flop with Multiplexed Data and active low clock and output	1	
	DFM3	2-bit D-Type Flip-Flop with Multiplexed Data and Clear	1	1
	DFM3B	2-bit D-Type Flip-Flop with Multiplexed Data and active low Clear and clock	1	
	DFM3E	2-bit D-Type Flip-Flop with Multiplexed Data, Clear, and active low clock	1	1
	DFM4C	2-bit D-Type Flip-Flop with Multiplexed Data and active low Preset and output	1	
	DFM4D	2-bit D-Type Flip-Flop with Multiplexed Data and active low Preset, clock, and output	1	
	DFM6A	4-bit D-Type Flip-Flop with Multiplexed Data, active low Clear, and active high Clock	1	
	DFM6B	4-bit D-Type Flip-Flop with Multiplexed Data, active low Clear, and clock	1	
	DFM7A	4-bit D-Type Flip-Flop with Multiplexed Data, active low Clear, and active high clock	1	
	DFM7B	4-bit D-Type Flip-Flop with Multiplexed Data, active low Clear and clock	1	
	DFMA	2-bit D-Type Flip-Flop with Multiplexed Data and active low clock	1	
	DFMB	2-bit D-Type Flip-Flop with Multiplexed Data and active low Clear	1	
	DFME1A	2-bit D-Type Flip-Flop with Multiplexed Data and active low Enable	1	
	DFP1	D-Type Flip-Flop with active high Preset		2
	DFP1A	D-Type Flip-Flop with active high Preset and active low clock		2
	DFP1B	D-Type Flip-Flop with active low Preset		2
	DFP1C	D-Type Flip-Flop with active high Preset and active low output	1	1
	DFP1D	D-Type Flip-Flop with active low Preset and clock		2
	DFP1E	D-Type Flip-Flop with active low Preset and output	1	
	DFP1F	D-Type Flip-Flop with active high Preset and active low clock and output	1	1
	DFP1G	D-Type Flip-Flop with active low Preset, clock, and output	1	
	DFPC	D-Type Flip-Flop with active high Preset, active low Clear, and active high clock		2
	DFPCA	D-Type Flip-Flop with active high Preset, and active low Clear and clock		2

Hard Macros—Sequential (Continued)

Function	Macro	Description	Modules S	Modules C
J-K Type	JKF	JK Flip-Flop with active low K-input	1	
	JKF1B	JK Flip-Flop with active low clock and K-input	1	
	JKF2A	JK Flip-Flop with active low Clear and K-input	1	
	JKF2B	JK Flip-Flop with active low Clear, clock, and K-input	1	
	JKF2C	JK Flip-Flop with active high Clear and active low K-input	1	1
	JKF2D	JK Flip-Flop with active high Clear and active low clock and K-input	1	1
T-Type	TF1A	T-Type Flip-Flop with active low Clear	1	
	TF1B	T-Type Flip-Flop with active low Clear and clock	1	
Latch	DL1	Data Latch	1	
	DL1A	Data Latch with active low output	1	
	DL1B	Data Latch with active low clock	1	
	DL1C	Data Latch with active low clock and output	1	
	DLC	Data Latch with active low Clear	1	
	DLC1	Data Latch with active high Clear		1
	DLC1A	Data Latch with active high Clear and active low clock		1
	DLC1F	Data Latch with active high Clear and active low output		1
	DLC1G	Data Latch with active high Clear and active low clock and output		1
	DLCA	Data Latch with active low Clock and Clear	1	
	DLE	Data Latch with active high Enable	1	
	DLE1D	Data Latch with active high Enable and clock and active low input and output	1	
	DLE2B	Data Latch with active low Enable, Clear, and clock	1	
	DLE2C	Data Latch with active low Enable and clock and active high Clear		1
	DLE3B	Data Latch with active low Enable and clock and active low Preset		1
	DLE3C	Data Latch with active low Enable, Preset, and clock		1
	DLEA	Data Latch with active low Enable and active high clock	1	
	DLEB	Data Latch with active high Enable and active high clock	1	
	DLEC	Data Latch with active low Enable and clock	1	
	DLM	2-bit Data Latch with Multiplexed Data	1	
	DLM3	4-bit Data Latch with Multiplexed Data	1	
	DLM3A	4-bit Data Latch with Multiplexed Data and active low clock	1	
	DLM4	Data Latch with Multiplexed Data	1	
	DLM4A	Data Latch with Multiplexed Data	1	
	DLMA	2-bit Data Latch with Multiplexed Data, and active low clock	1	
	DLME1A	2-bit Data Latch with Multiplexed Data and Enable and active low clock	1	
	DLP1	Data Latch with active high Preset and clock		1
	DLP1A	Data Latch with active high Preset and active low clock		1
	DLP1B	Data Latch with active low Preset and active high clock		1
	DLP1C	Data Latch with active low Preset and clock		1
	DLP1D	Data Latch with active low Preset and output and active high clock	1	
	DLP1E	Data Latch with active low Preset, clock, and output	1	

Input/Output Macros

Function	Macro	Description	I/O Modules
Buffer	BBHS	Bidirectional Buffer, High Slew	1
	BBUFTH	Bidirectional Buffer, Tristate Enable, High Slew	1
	BBUFTL	Bidirectional Buffer, Tristate Enable, Low Slew	1
	BIBUF	Bidirectional Buffer, High Slew (with hidden buffer at Y pin)	1
	HCLKBUF	Dedicated High-Speed S-Module Clock Buffer	1
	IBUF	Input Buffer	1
	INBUF	Input Buffer	1
	IOCLKBUF	Dedicated I/O Module Clock Buffer	1
	IOPCLBUF	Dedicated I/O Module IOPCL Buffer	1
	OBHS	Output buffer, High Slew	1
	OBUFTH	Output Buffer, Tristate Enable, High Slew	1
	OBUFTL	Output Buffer, Tristate Enable, Low Slew	1
	OUTBUF	Output Buffer, High Slew	1
Bidirectional	BRECTH	Bidirectional, Output Register with Clear, Data Enable, Tristate Enable, High Slew	1
	BRECTL	Bidirectional, Output Register with Clear, Data Enable, Tristate Enable, Low Slew	1
	BREPTH	Bidirectional, Output Register with Preset, Data Enable, Tristate Enable, High Slew	1
	BREPTL	Bidirectional, Output Register with Preset, Data Enable, Tristate Enable, Low Slew	1
	CLKBIBUF	Bidirectional with Input Dedicated to Clock Network	1
	DECETH	Bidirectional, Double Registered with Clear, Data Enable, Tristate Enable, High Slew	1
	DECETL	Bidirectional, Double Registered with Clear, Data Enable, Tristate Enable, Low Slew	1
	DEPETH	Bidirectional, Double Registered with Preset, Data Enable, Tristate Enable, High Slew	1
	DEPETL	Bidirectional, Double Registered with Preset, Data Enable, Tristate Enable, Low Slew	1
Input	CLKBUF	Input for Dedicated Routed Clock Network	1
	IREC	Input Register with Clear	1
	IREP	Input Register with Preset	1
Output	FECTMH	Output Register with Muxed Feedback, Clear, Data Enable, Tristate Enable, High Slew	1
	FECTML	Output Register with Muxed Feedback, Clear, Data Enable, Tristate Enable, Low Slew	1
	FEPTMH	Output Register with Muxed Feedback, Preset, Data Enable, Tristate Enable, High Slew	1
	FEPTML	Output Register with Muxed Feedback, Preset, Data Enable, Tristate Enable, Low Slew	1
	ORECTH	Output Register with Clear, Data Enable, Tristate Enable, High Slew	1
	ORECTL	Output Register with Clear, Data Enable, Tristate Enable, Low Slew	1
	OREPTH	Output Register with Preset, Data Enable, Tristate Enable, High Slew	1
	OREPTL	Output Register with Preset, Data Enable, Tristate Enable, Low Slew	1
	TBHS	Tristate output, High Slew	1
	TRIBUFF	Tristate output, High Slew	1

Soft Macros

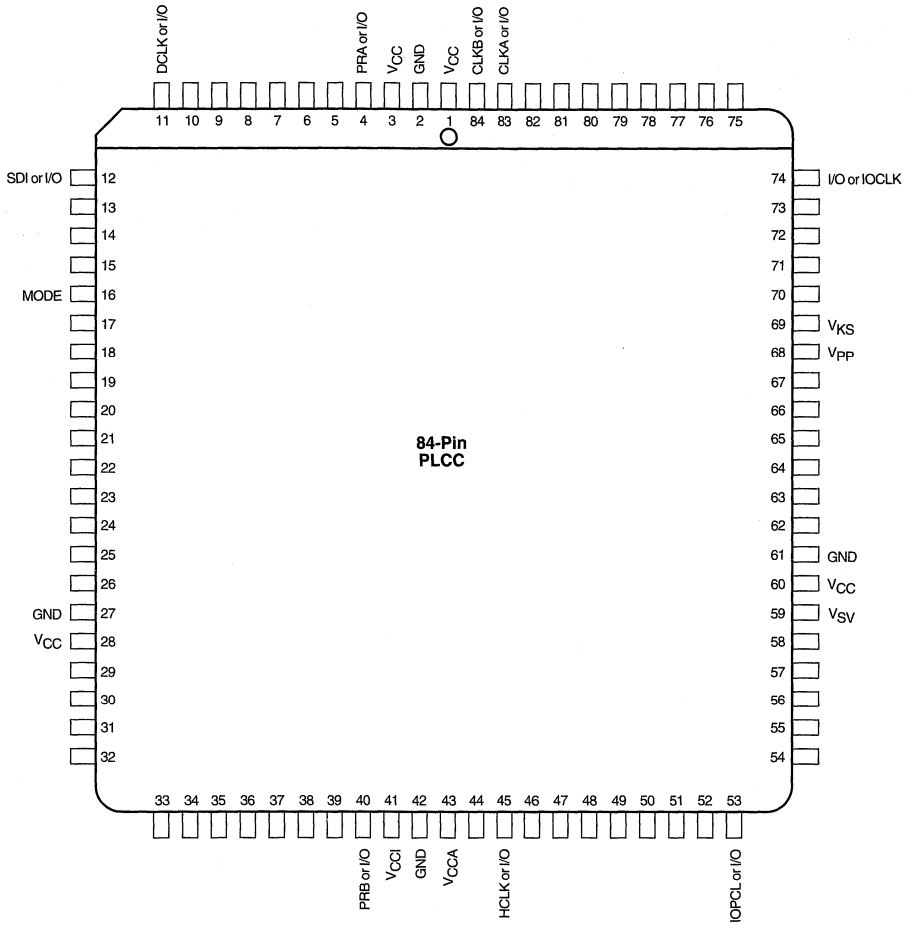
Function	Macro	Description	Maximum Logic Levels	Modules S	Modules C
Adder	FADD10	10-bit adder	3		56
	FADD12	12-bit adder	4		9
	FADD16	16-bit adder	5		97
	FADD8	8-bit adder	4		44
	FADD9	9-bit adder with active low carry out	3		49
	VAD16C	Very fast 16-bit adder, no Carry in	3		97
	VADC16C	Very fast 16-bit adder with Carry in	3		97
Comparator	ICMP4	4-bit Identity Comparator	2		5
	ICMP8	8-bit Identity Comparator	3		9
	MCMPC2	2-bit Magnitude Comparator with Enable	3		9
	MCMPC4	4-bit Magnitude Comparator with Enable	4		18
	MCMPC8	8-bit Magnitude Comparator with Enable	6		36
Counter	CNT4A	4-bit binary counter with load and clear	4	4	8
	CNT4B	4-bit binary counter with load, clear, carry-in, carry-out	4	4	7
	FCTD16C	Fast 16-bit Down Counter, parallel loadable	2	19	33
	FCTD8A	Fast 8-bit Down Counter, parallel loadable	1	10	18
	FCTD8B	Fast 8-bit Down Counter, parallel loadable	1	9	13
	FCTU16C	Fast 16-bit Up Counter, parallel loadable	2	19	31
	FCTU8A	Fast 8-bit Up Counter, parallel loadable	1	10	17
	FCTU8B	Fast 8-bit Up Counter, parallel loadable	1	9	12
	UDCNT4A	4-bit up/down counter with load, carry-in, and carry-out	5	4	13
	VCTD16C	Very fast 16-bit down counter, delay after load, registered control inputs	1	34	41
	VCTD2CP	2-bit down counter, prescaler, delay after load, use to build VCTD counters	1	5	2
	VCTD2CU	2-bit down counter, upper bits, delay after load, use to build VCTD counters	1	2	3
	VCTD4CL	4-bit down counter, lower bits, delay after load, use to build VCTD counters	1	4	7
	VCTD4CM	4-bit down counter, middle bits, delay after load, use to build VCTD counters	1	4	8
Decoder	DEC2X4	2-to-4 decoder	1		4
	DEC2X4A	2-to-4 decoder with active low outputs	1		4
	DEC3X8	3-to-8 decoder	1		8
	DEC3X8A	3-to-8 decoder with active low outputs	1		8
	DEC4X16A	4-to-16 decoder with active low outputs	2		20
	DECE2X4	2-to-4 decoder with enable	1		4
	DECE2X4A	2-to-4 decoder with enable and active low outputs	1		4
	DECE3X8	3-to-8 decoder with enable	2		11
	DECE3X8A	3-to-8 decoder with enable and active low outputs	2		11
Latch	DLC8A	octal latch with clear active low 8-bit Data Latch with active low Clear	1	8	
	DLE8	octal latch with enable 8-bit Data Latch with active high Enable	1	8	
	DLM8	octal latch with multiplexed data 8-bit Data Latch with Multiplexed Data	1	8	
MUX	MX16	16-to-1 Multiplexor	2		5
	MX8	8-to-1 Multiplexor with active high output	2		3
	MX8A	8-to-1 Multiplexor with active low output	2		3
Multiplier	SMULT8	8-bit by 8-bit Multiplier			242
Shift Register	SREG4A	4-bit shift register with clear active low	1	4	
	SREG8A	8-bit shift register with clear active low	1	8	

Soft Macros—TTL Equivalent

Function	Macro	Description	Maximum Logic Levels	Modules S	Modules C
	TA00	2-input NAND	1		1
	TA02	2-input NOR	1		1
	TA04	Inverter	1		1
	TA07	Buffer	1		1
	TA08	2-input AND	1		1
	TA10	3-input NAND	1		1
	TA11	3-input AND	1		1
	TA138	3-to-8 decoder with enable and active low outputs	2		12
	TA139	2-to-4 decoder with active low enable and outputs	1		4
	TA150	16-to-1 multiplexor with active low enable	3		6
	TA151	8-to-1 multiplexor with enable and both active low and active high output	3		5
	TA153	4-to-1 multiplexor with active low enable	2		2
	TA154	4-to-16 decoder with active low outputs and select lines	2		22
	TA157	2-to-1 multiplexor with active low enable	1		1
	TA160	4-bit decade counter with active low clear and load	4	4	8
	TA161	4-bit binary counter with active low clear and load	3	4	6
	TA164	8-bit serial in, parallel out shift register, active low clear	1	8	
	TA169	4-bit Up/Down Counter	6	4	14
	TA174	hex D-type flip-flop with active low clear	1	6	
	TA175	quadruple D-type flip-flop with active low clear	1	4	
	TA181	ALU			37
	TA190	4-bit up/down decode counter with up/down mode	7	4	31
	TA191	4-bit up/down binary counter with up/down mode	7	4	30
	TA194	4-bit bidirectional universal shift register	1	4	4
	TA195	4-bit parallel-access shift register	1	4	1
	TA20	4-input NAND	1		2
	TA21	4-input AND	1		1
	TA269	8-bit up/down binary counter	8	8	28
	TA27	3-input NOR	1		1
	TA273	octal register with clear	1	8	
	TA280	9-bit odd/even parity generator and checker	4		9
	TA32	2-input OR	1		1
	TA377	octal register with active low enable	1	8	
	TA40	4-input NAND	1		2
	TA42	4 to 10 decoder	1		10
	TA51	AND-OR-Invert	1		2
	TA54	4-wide 2-input AND-OR-Invert	2		5
	TA55	2-wide 4-input AND-OR-Invert	2		3
	TA688	8-bit identity comparator	3		9
	TA86	2-input exclusive OR	1		1

Package Pin Assignments

84-Pin PLCC (Top View)

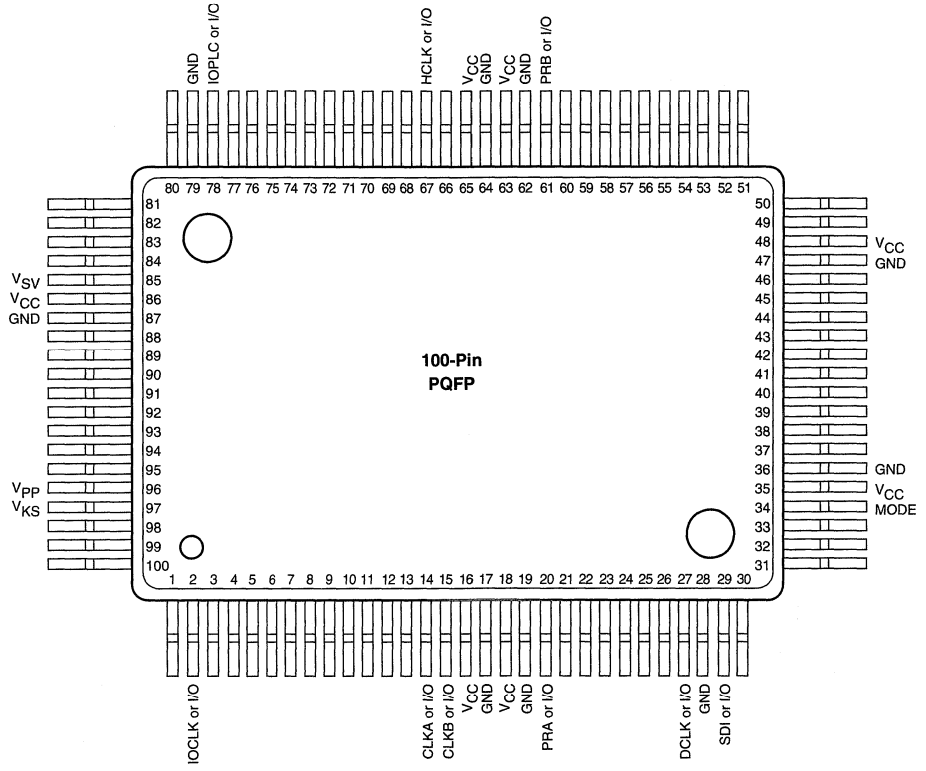


Notes:

1. *Unused I/O pins are designated as outputs by ALS and are driven low.*
2. *All unassigned pins are available for use as I/Os.*
3. *MODE = GND, except during device programming or debugging.*
4. *$V_{PP} = V_{CC}$, except during device programming.*
5. *$V_{SV} = V_{CC}$, except during device programming.*
6. *V_{KS} = GND, except during device programming.*

Package Pin Assignments (continued)

100-Pin PQFP (Top View)

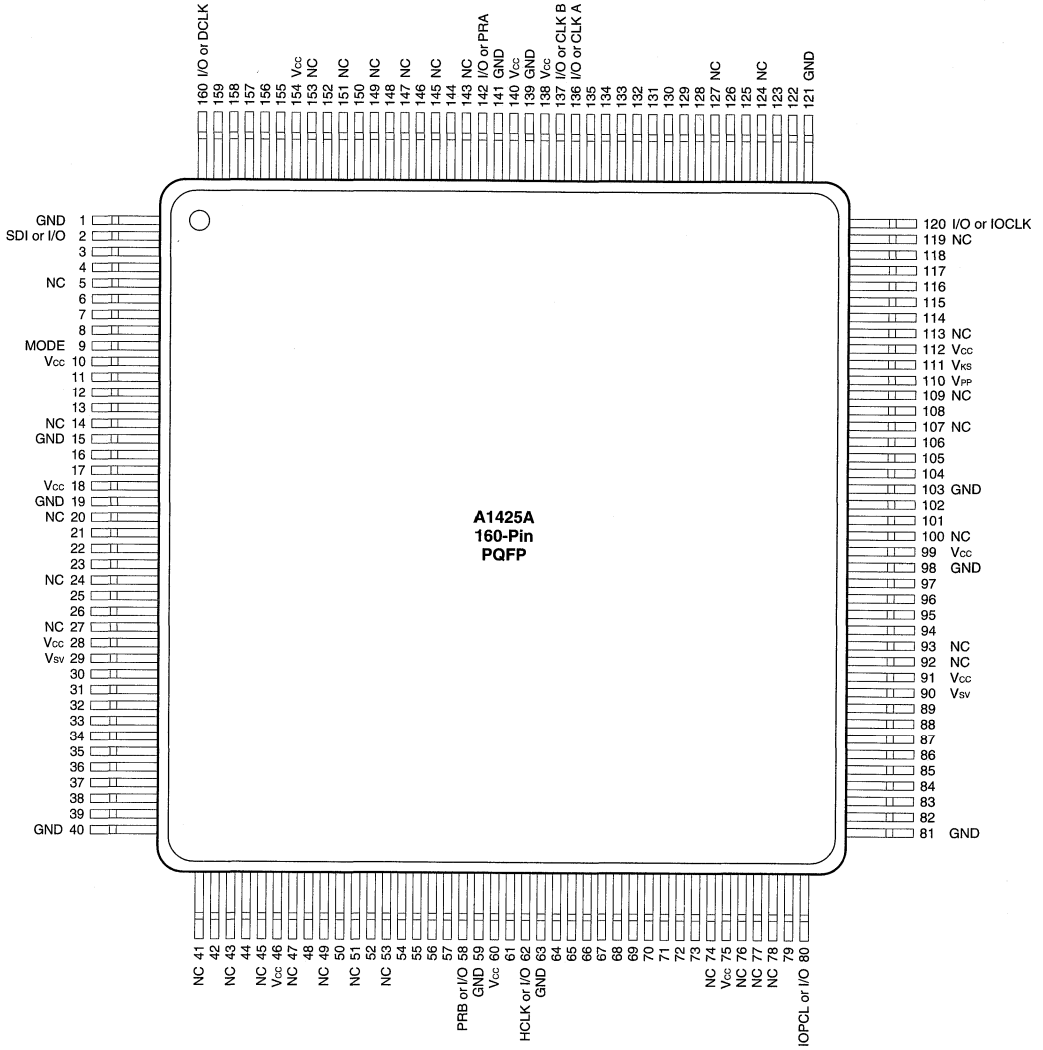


Notes:

1. *Unused I/O pins are designated as outputs by ALS and are driven low.*
2. *All unassigned pins are available for use as I/Os.*
3. *MODE = GND, except during device programming or debugging.*
4. *$V_{PP} = V_{CC}$, except during device programming.*
5. *$V_{SV} = V_{CC}$, except during device programming.*
6. *V_{KS} = GND, except during device programming.*

Package Pin Assignments (continued)

160-Pin PQFP (Top View)

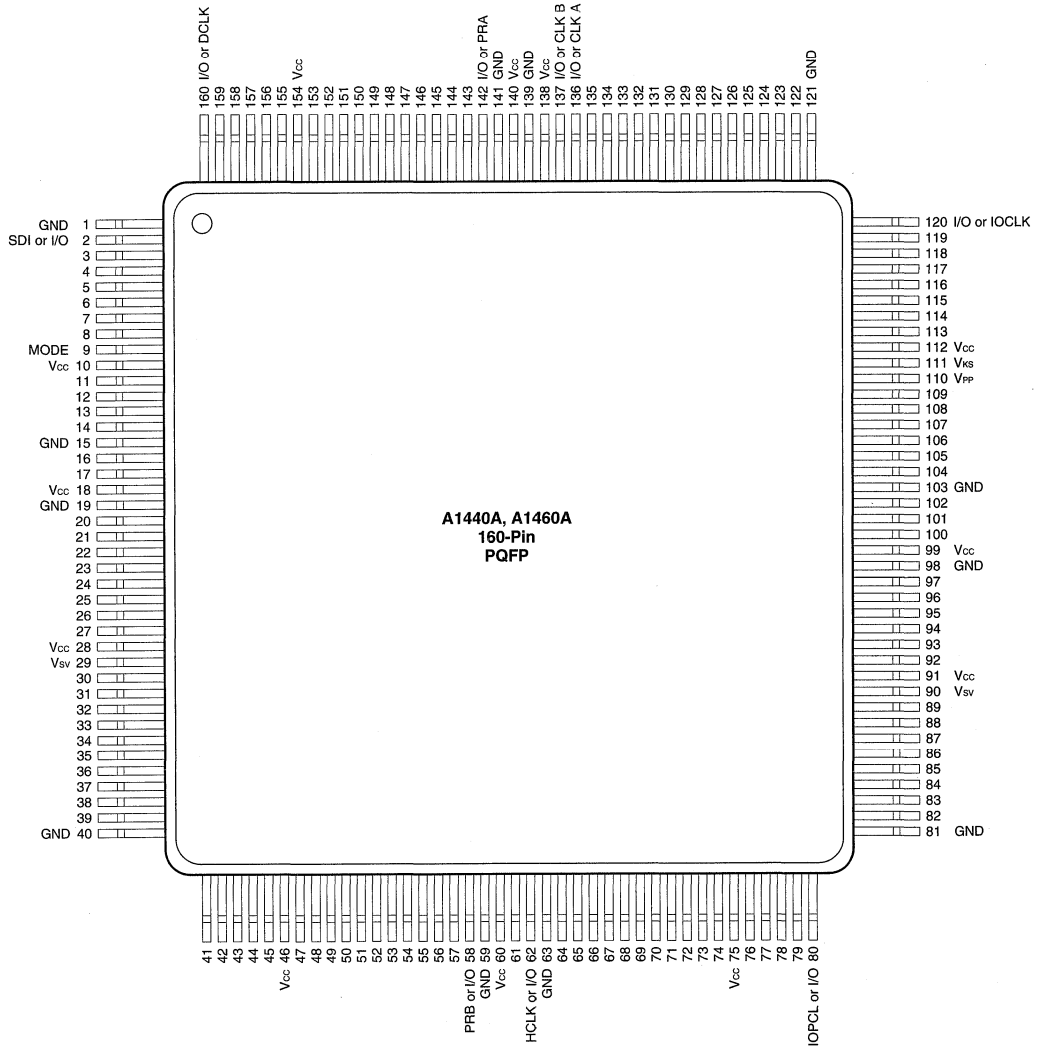


Notes:

1. *Unused I/O pins are designated as outputs by ALS and are driven low.*
2. *All unassigned pins are available for use as I/Os.*
3. *MODE = GND, except during device programming or debugging.*
4. *$V_{PP} = V_{CC}$, except during device programming.*
5. *$V_{SV} = V_{CC}$, except during device programming.*
6. *V_{KS} = GND, except during device programming.*

Package Pin Assignments (continued)

160-Pin PQFP (Top View)

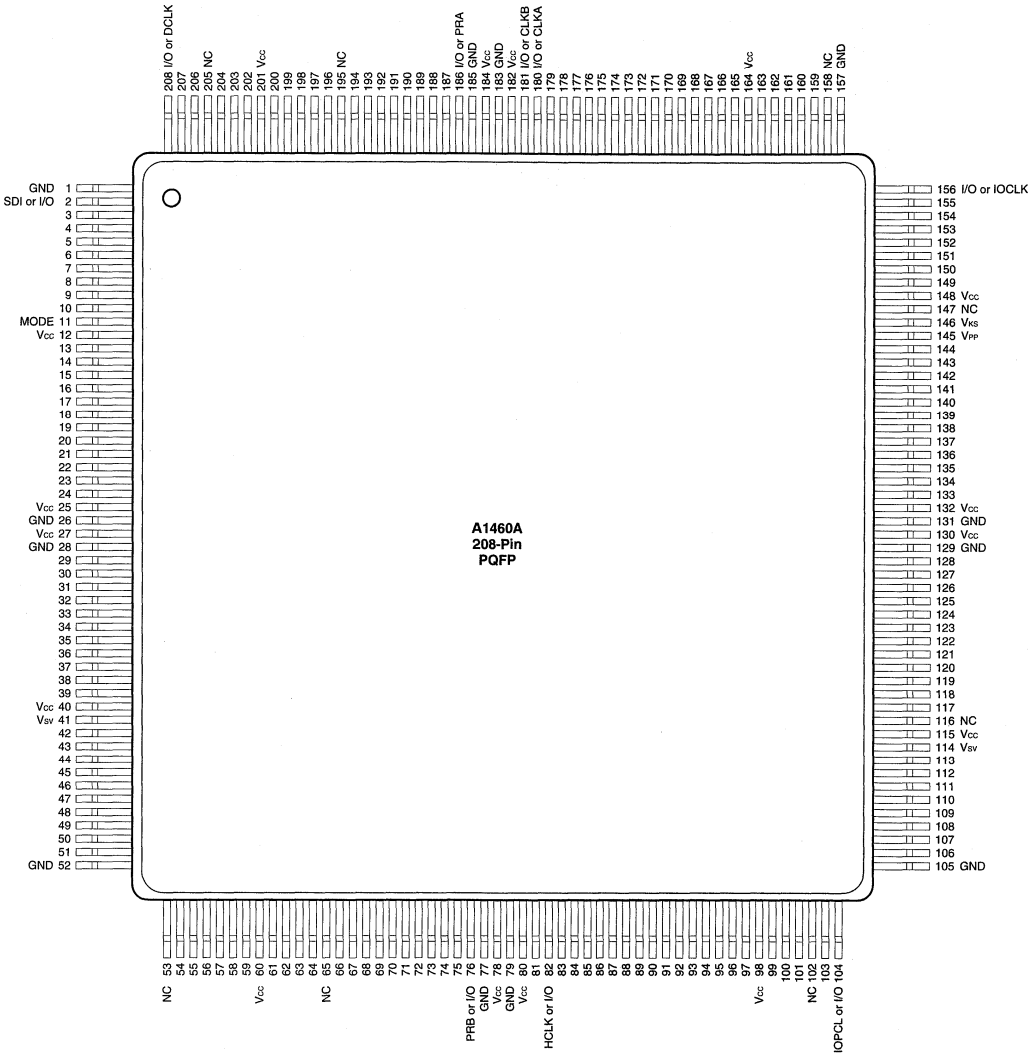


Notes:

1. *Unused I/O pins are designated as outputs by ALS and are driven low.*
2. *All unassigned pins are available for use as I/Os.*
3. *MODE = GND, except during device programming or debugging.*
4. *$V_{PP} = V_{CC}$, except during device programming.*
5. *$V_{SV} = V_{CC}$, except during device programming.*
6. *V_{KS} = GND, except during device programming.*

Package Pin Assignments (continued)

208-Pin PQFP (Top View)

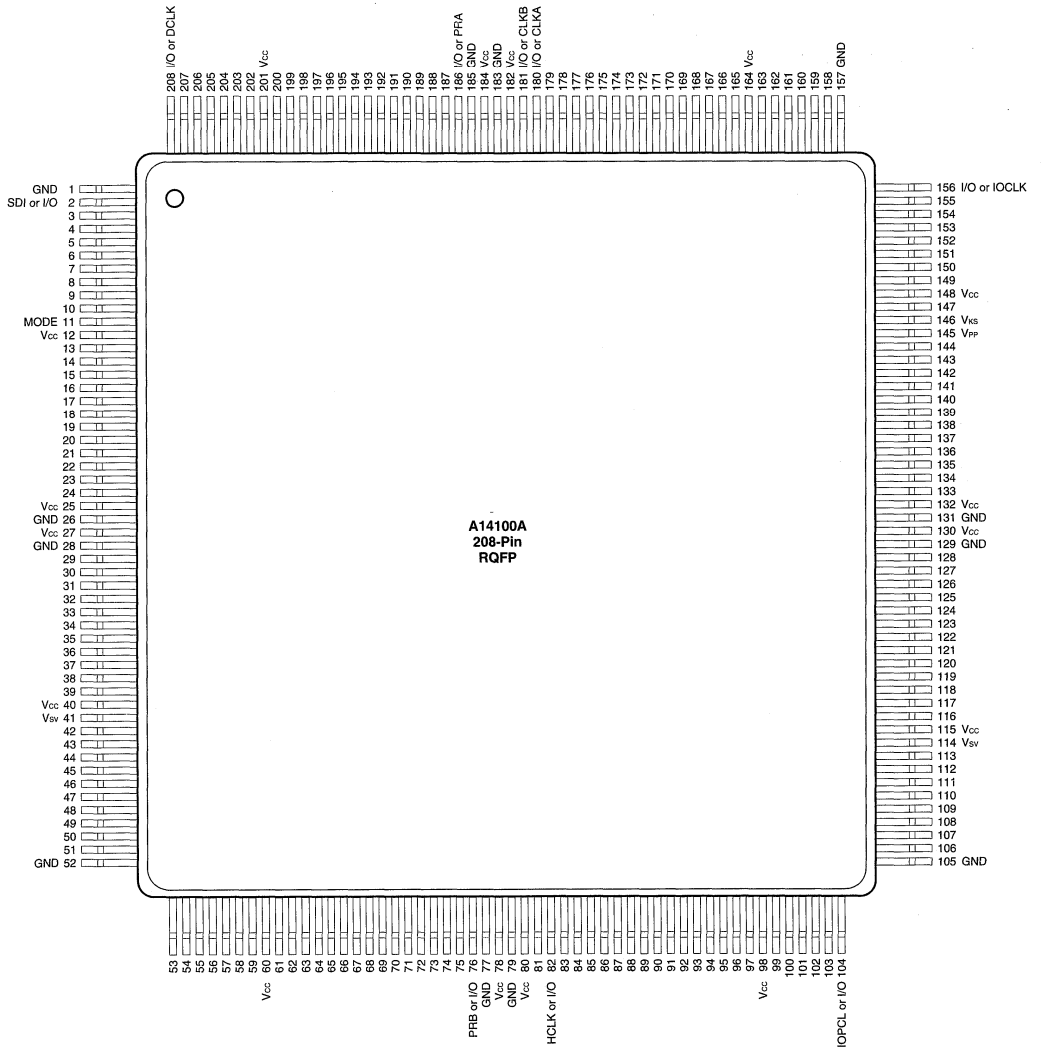


Notes:

1. *Unused I/O pins are designated as outputs by ALS and are driven low.*
2. *All unassigned pins are available for use as I/Os.*
3. *MODE = GND, except during device programming or debugging.*
4. $V_{PP} = V_{CC}$, *except during device programming.*
5. $V_{SV} = V_{CC}$, *except during device programming.*
6. V_{KS} = *GND, except during device programming.*

Package Pin Assignments (continued)

208-Pin RQFP (Top View)

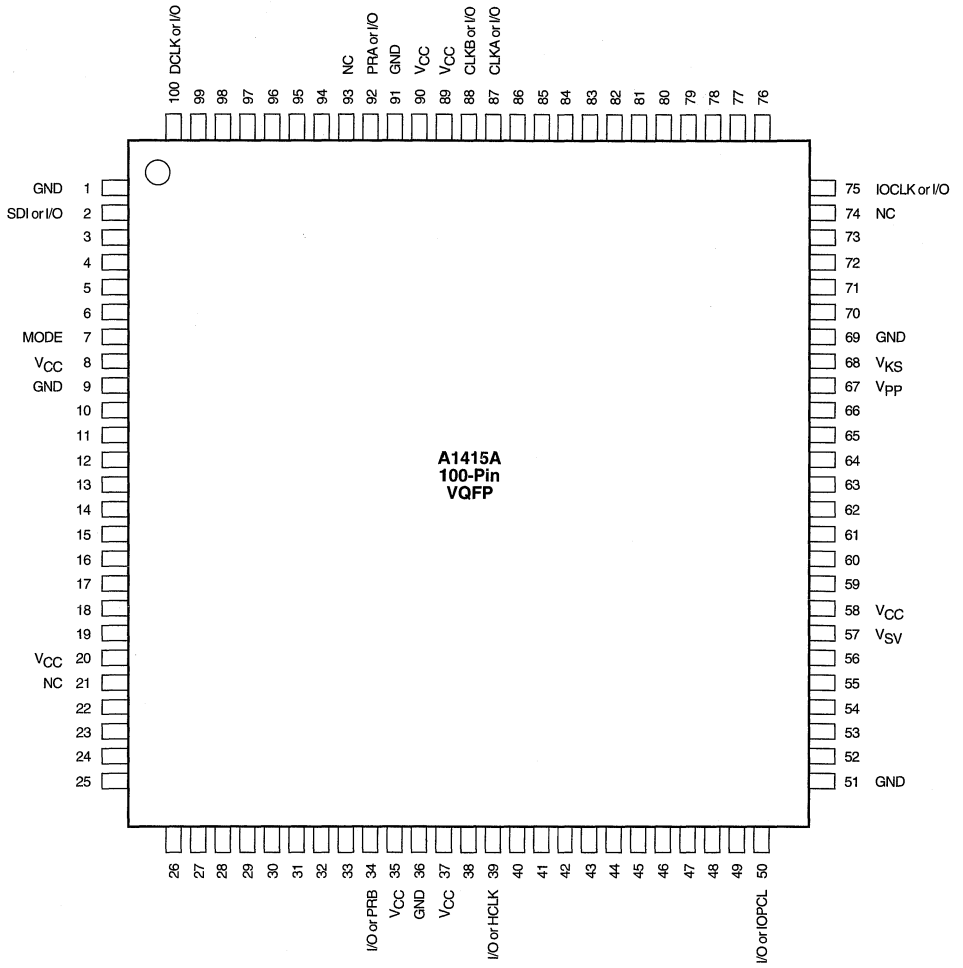


Notes:

1. *Unused I/O pins are designated as outputs by ALS and are driven low.*
2. *All unassigned pins are available for use as I/Os.*
3. *MODE = GND, except during device programming or debugging.*
4. *$V_{PP} = V_{CC}$, except during device programming.*
5. *$V_{SV} = V_{CC}$, except during device programming.*
6. *V_{KS} = GND, except during device programming.*

Package Pin Assignments (continued)

100-Pin VQFP (Top View)

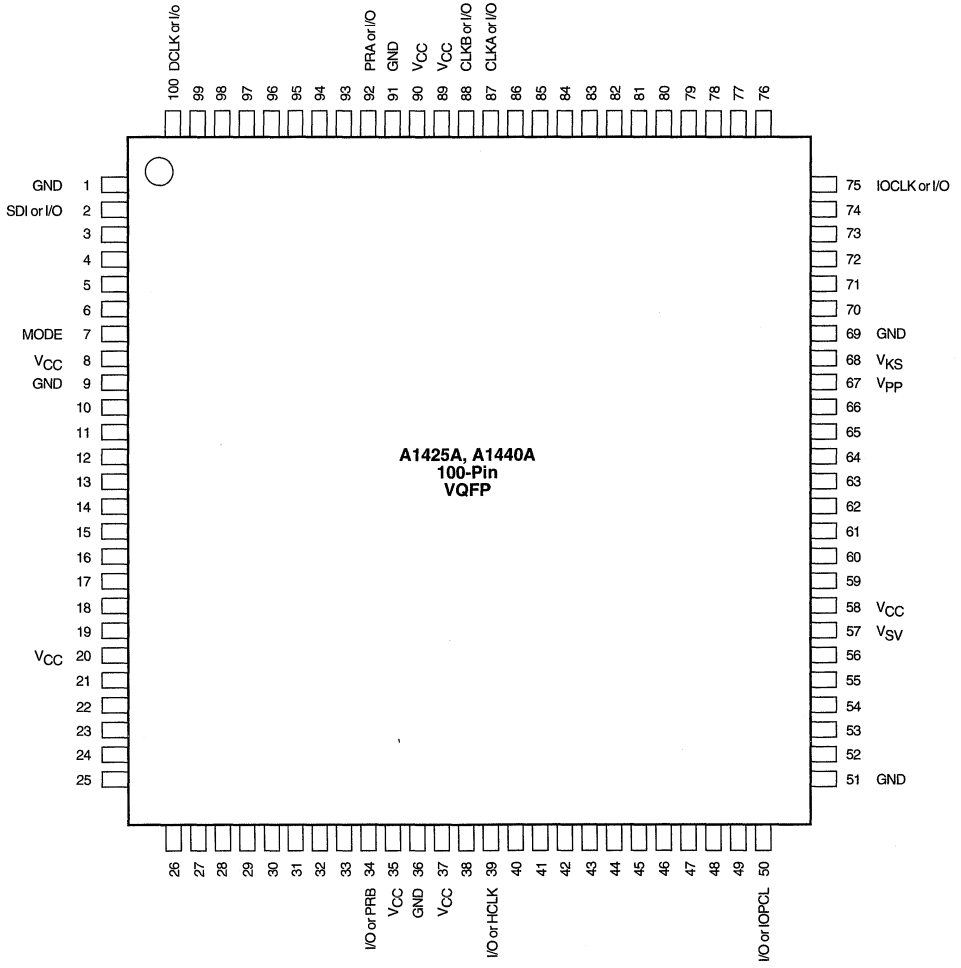


Notes:

1. *Unused I/O pins are designated as outputs by ALS and are driven low.*
2. *All unassigned pins are available for use as I/Os.*
3. *MODE = GND, except during device programming or debugging.*
4. *$V_{PP} = V_{CC}$, except during device programming.*
5. *$V_{SV} = V_{CC}$, except during device programming.*
6. *V_{KS} = GND, except during device programming.*

Package Pin Assignments (continued)

100-Pin VQFP (Top View)

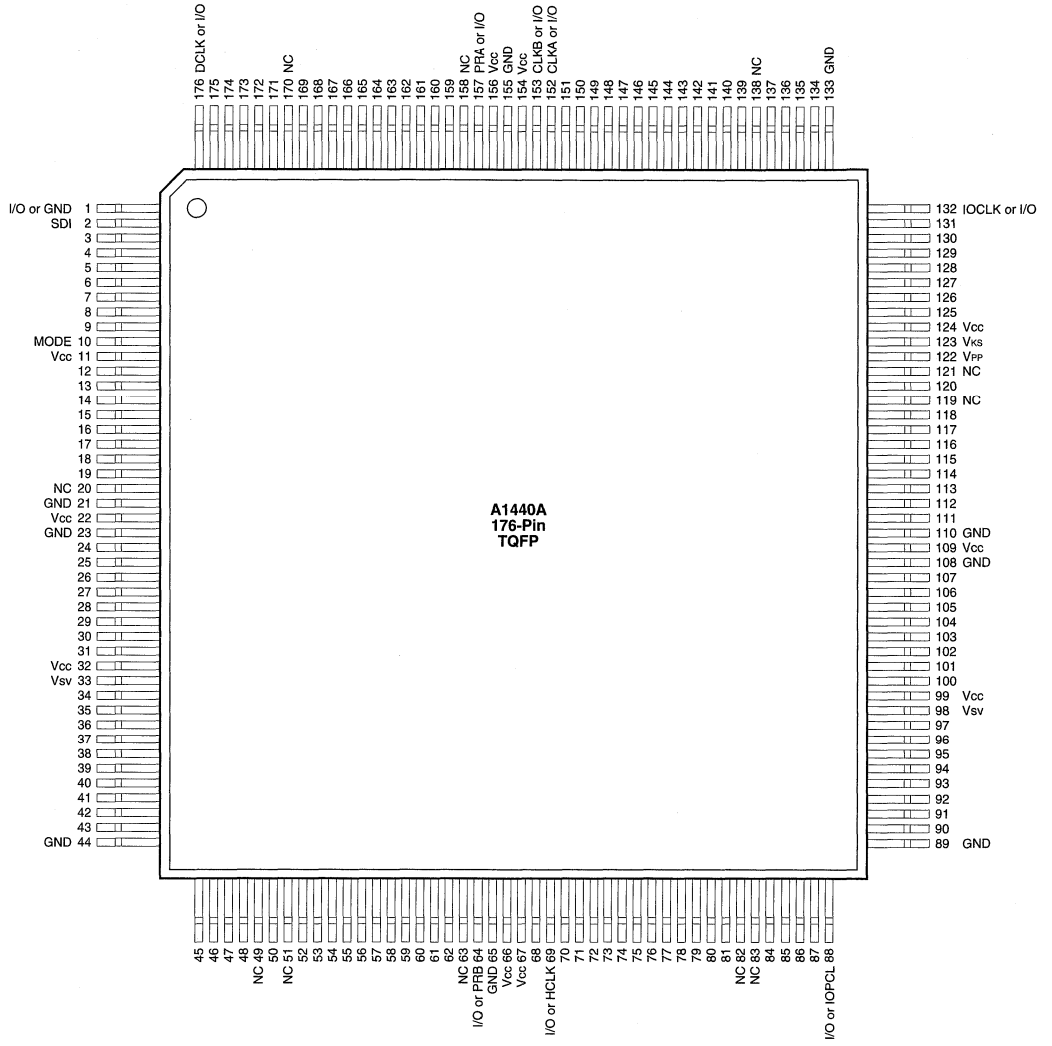


Notes:

1. *Unused I/O pins are designated as outputs by ALS and are driven low.*
2. *All unassigned pins are available for use as I/Os.*
3. *MODE = GND, except during device programming or debugging.*
4. *$V_{PP} = V_{CC}$, except during device programming.*
5. *$V_{SV} = V_{CC}$, except during device programming.*
6. *V_{KS} = GND, except during device programming.*

Package Pin Assignments (continued)

176-Pin TQFP (Top View)

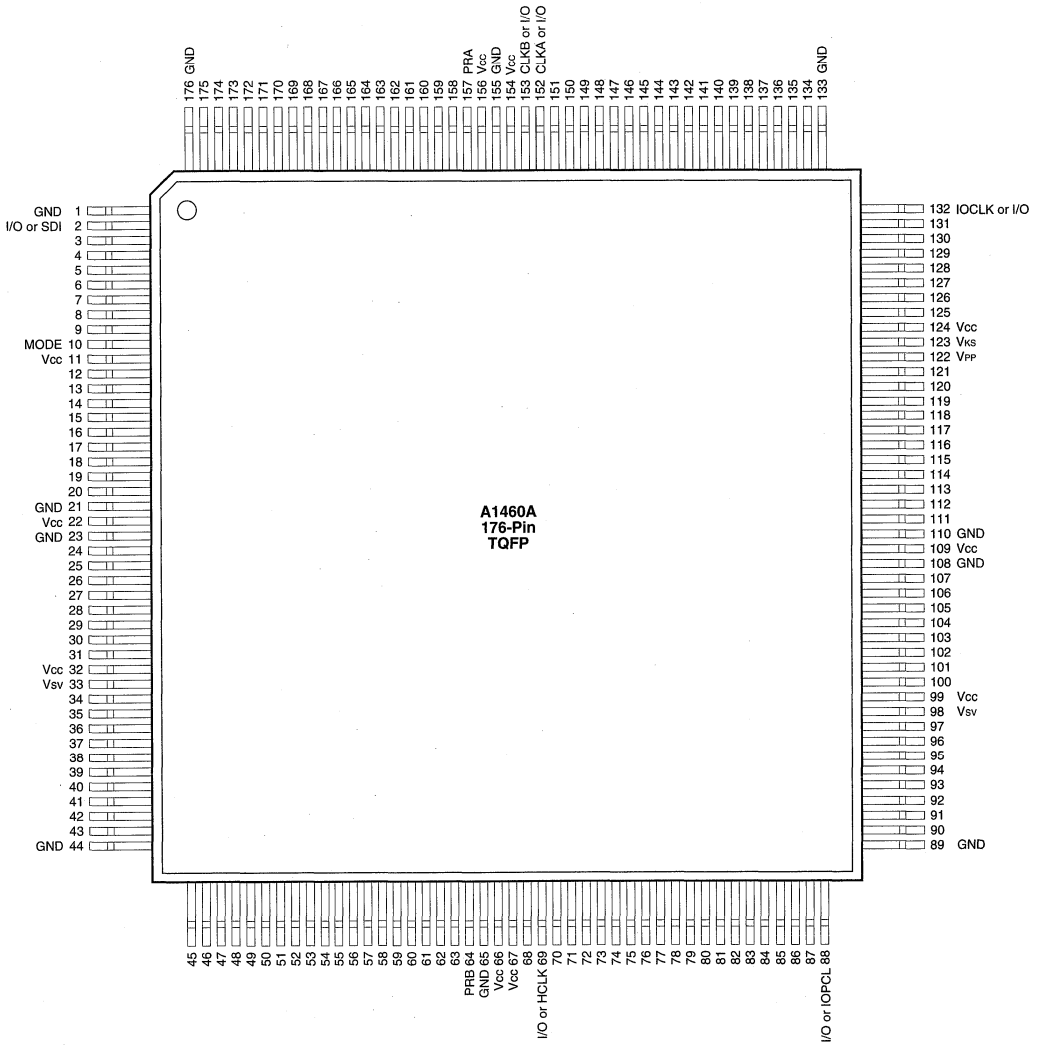


Notes:

1. *Unused I/O pins are designated as outputs by ALS and are driven low.*
2. *All unassigned pins are available for use as I/Os.*
3. *MODE = GND, except during device programming or debugging.*
4. $V_{PP} = V_{CC}$, *except during device programming.*
5. $V_{SV} = V_{CC}$, *except during device programming.*
6. $V_{KS} = GND$, *except during device programming.*

Package Pin Assignments (continued)

176-Pin TQFP (Top View)

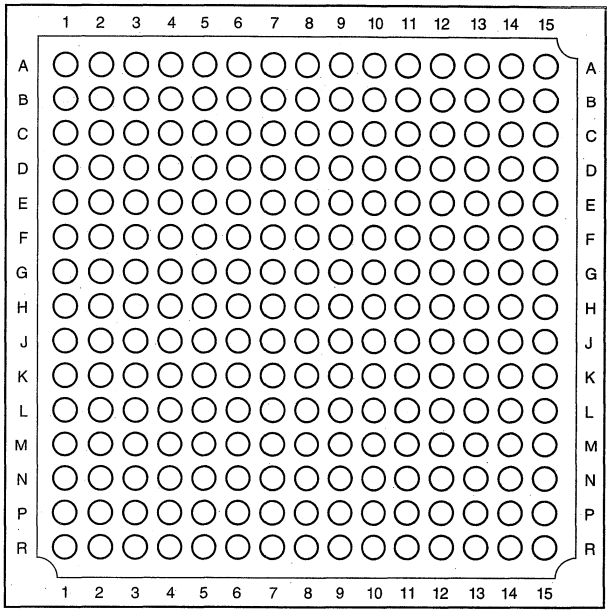


Notes:

1. *Unused I/O pins are designated as outputs by ALS and are driven low.*
2. *All unassigned pins are available for use as I/Os.*
3. *MODE = GND, except during device programming or debugging.*
4. *$V_{PP} = V_{CC}$, except during device programming.*
5. *$V_{SV} = V_{CC}$, except during device programming.*
6. *V_{KS} = GND, except during device programming.*

Package Pin Assignments (continued)

225-Pin BGA (Top View)



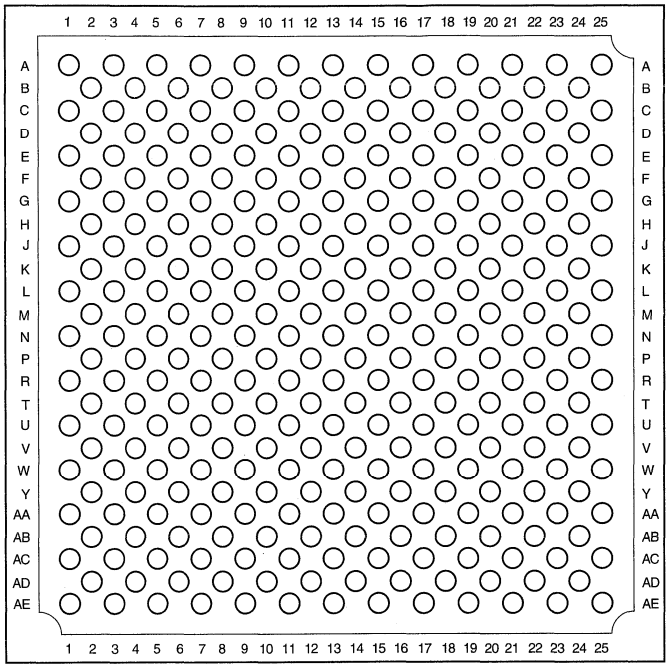
Signal	Location
CLKA or I/O	C8
CLKB or I/O	B8
DCLK or I/O	B2
GND	A1, A15, F8, G7, G8, G9, H6, H7, H8, H9, H10, J7, J8, J9, K8, P2, R15
HCLK or I/O	P9
IOCLK or I/O	B14
IOPCL or I/O	P14
MODE	D1
NC	A11, B5, B7, D8, D12, F6, F11, H1, H12, H14, K11, L1, L13, N8, P5, R1, R8, R11, R14
PRA OR I/O	A7
PRB or I/O	L7
SDI or I/O	D4
V_{CC}	A8, B12, D5, D14, E3, E8, H2, H3, H11, H15, L2, M8, M15, P4, P8, R13
V_{KS}	D15
V_{PP}	E13
V_{SV}	K4, L12

Notes:

1. *Unused I/O pins are designated as outputs by ALS and are driven low.*
2. *All unassigned pins are available for use as I/Os.*
3. *MODE = GND, except during device programming or debugging.*
4. *$V_{PP} = V_{CC}$, except during device programming.*
5. *$V_{SV} = V_{CC}$, except during device programming.*
6. *V_{KS} = GND, except during device programming.*

Package Pin Assignments (continued)

313-Pin BGA (Top View)



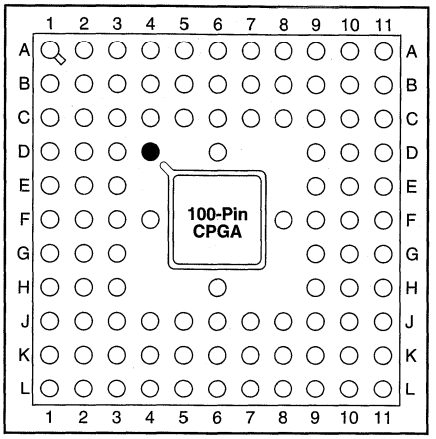
Signal	Location
CLKA or I/O	J13
CLKB or I/O	G13
DCLK or I/O	B2
GND	A1, A25, AD2, AE25, L13, M12, M14, N11, N13, N15, P12, P14, R13
HCLK or I/O	T14
IOCLK or I/O	B24
IOPCL or I/O	AD24
MODE	G3
NC	A3, A13, A23, AA5, AA9, AA23, AB2, AB4, AB20, AC13, AC25, AD22, AE1, AE21, B14, C5, C25, D4, D24, E3, E21, F6, F10, F16, G1, G25, H18, H24, J1, J7, J25, K12, L15, L17, M6, N1, N5, N7, N21, N23, P20, R11, T6, T8, U9, U13, U21, V16, W7, Y20, Y24
PRA OR I/O	H12
PRB or I/O	AD12
SDI or I/O	C1
V_{CC}	AB18, AD6, AE13, C13, C19, E13, G9, H22, K8, M16, N3, N9, N25, U5, W13, V24
V_{KS}	J21
V_{PP}	K20
V_{SV}	V2, V22

Notes:

1. *Unused I/O pins are designated as outputs by ALS and are driven low.*
2. *All unassigned pins are available for use as I/Os.*
3. *MODE = GND, except during device programming or debugging.*
4. *$V_{PP} = V_{CC}$, except during device programming.*
5. *$V_{SV} = V_{CC}$, except during device programming.*
6. *V_{KS} = GND, except during device programming.*

Package Pin Assignments (continued)

100-Pin CPGA (Top View)



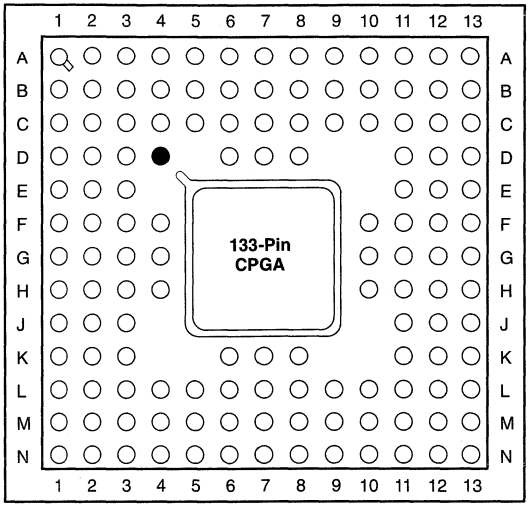
Signal	Location
CLKA or I/O	C7
CLKB or I/O	D6
DCLK or I/O	C4
GND	C3, F3, J3, C6, J6, J8, C9, F9, J9
HCLK or I/O	H6
IOCLK or I/O	C10
IOPCL or I/O	K9
MODE	C2
PRA OR I/O	A6
PRB or I/O	L3
SDI or I/O	B3
V_{CC}	F2, K2, B6, K6, B10, F10, K10
V_{KS}	E9
V_{PP}	E11
V_{SV}	G2

Notes:

1. *Unused I/O pins are designated as outputs by ALS and are driven low.*
2. *All unassigned pins are available for use as I/Os.*
3. *MODE = GND, except during device programming or debugging.*
4. *$V_{PP} = V_{CC}$, except during device programming.*
5. *$V_{SV} = V_{CC}$, except during device programming.*
6. *V_{KS} = GND, except during device programming.*

Package Pin Assignments (continued)

133-Pin CPGA (Top View)



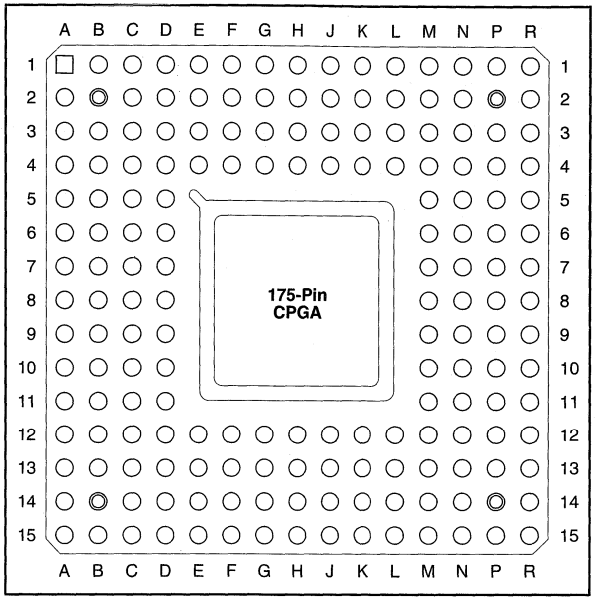
Signal	Location
CLKA or I/O	D7
CLKB or I/O	B6
DCLK or I/O	D4
GND	A2, C3, C7, C11, C12, G3, G11, L3, L7, L11, M3, N12
HCLK or I/O	K7
IOCLK or I/O	C10
IOPCL or I/O	L10
MODE	E3
NC	A1, A7, A13, G1, G13, N1, N7, N13
PRA OR I/O	A6
PRB or I/O	L6
SDI or I/O	C2
V_{CC}	B2, B7, B12, G2, G12, M2, M7, M12
V_{KS}	F10
V_{PP}	E11
V_{SV}	J2, J12

Notes:

1. *Unused I/O pins are designated as outputs by ALS and are driven low.*
2. *All unassigned pins are available for use as I/Os.*
3. *MODE = GND, except during device programming or debugging.*
4. *$V_{PP} = V_{CC}$, except during device programming.*
5. *$V_{SV} = V_{CC}$, except during device programming.*
6. *V_{KS} = GND, except during device programming.*

Package Pin Assignments (continued)

175-Pin CPGA (Top View)



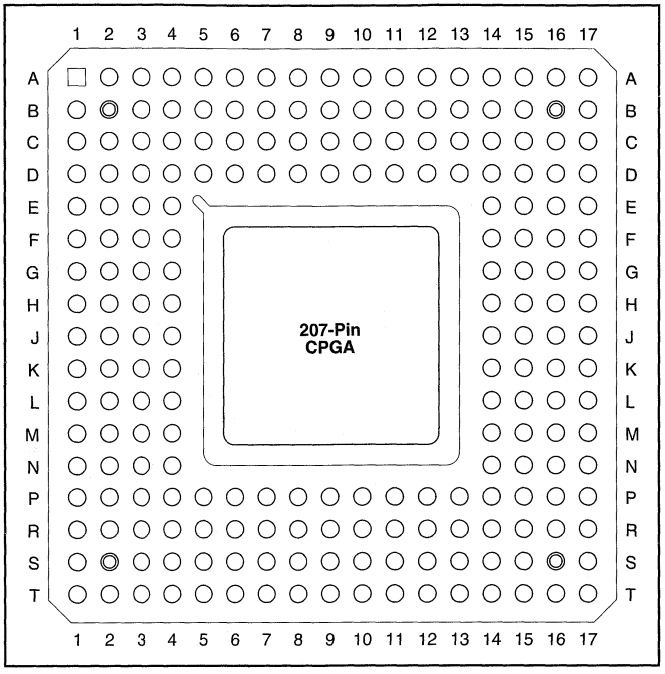
Signal	Location
CLKA or I/O	C9
CLKB or I/O	A9
DCLK or I/O	D5
GND	D4, D8, D11, D12, E4, H4, H12, L4, L12, M4, M8, M12
HCLK or I/O	R8
IOCLK or I/O	E12
IOPCL or I/O	P13
MODE	F3
NC	A1, A2, A15, B2, B3, P2, P14, R1, R2, R14, R15
PRA OR I/O	B8
PRB or I/O	R7
SDI or I/O	D3
V_{CC}	C3, C8, C13, H3, H13, N3, N8, N13
V_{KS}	E14
V_{PP}	E15
V_{SV}	L1, L14

Notes:

1. *Unused I/O pins are designated as outputs by ALS and are driven low.*
2. *All unassigned pins are available for use as I/Os.*
3. *MODE = GND, except during device programming or debugging.*
4. *$V_{PP} = V_{CC}$, except during device programming.*
5. *$V_{SV} = V_{CC}$, except during device programming.*
6. *V_{KS} = GND, except during device programming.*

Package Pin Assignments (continued)

207-Pin CPGA (Top View)



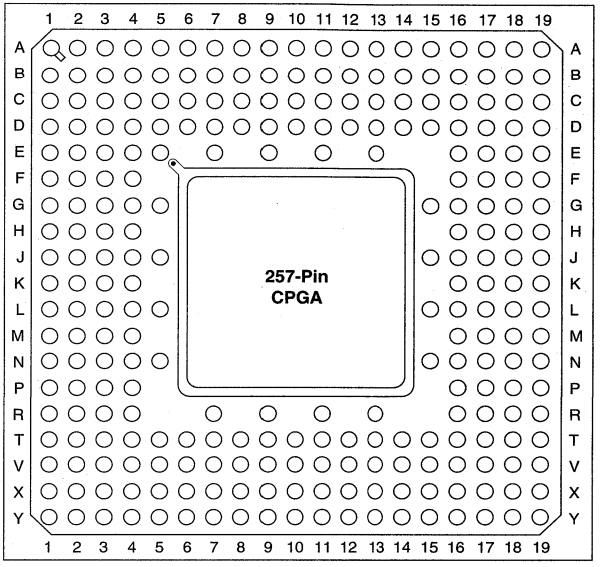
Signal	Location
CLKA or I/O	K1
CLKB or I/O	J3
DCLK or I/O	E4
GND	C15, D4, D5, D9, D14, J4, J14, P3, P4, P9, P14, R15
HCKL or I/O	J15
IOCLK or I/O	P5
IOPCL or I/O	N14
MODE	D7
NC	A1, A2, A16, A17, B1, B17, C1, C2, S1, S3, S17, T1, T2, T16, T17
PRA OR I/O	H1
PRB or I/O	K16
SDI or I/O	C3
V_{CC}	B2, B9, B16, J2, J16, S2, S9, S16
V_{KS}	P7
V_{PP}	T5
V_{SV}	D11, P12

Notes:

1. *Unused I/O pins are designated as outputs by ALS and are driven low.*
2. *All unassigned pins are available for use as I/Os.*
3. *MODE = GND, except during device programming or debugging.*
4. *$V_{PP} = V_{CC}$, except during device programming.*
5. *$V_{SV} = V_{CC}$, except during device programming.*
6. *V_{KS} = GND, except during device programming.*

Package Pin Assignments (continued)

257-Pin CPGA (Top View)



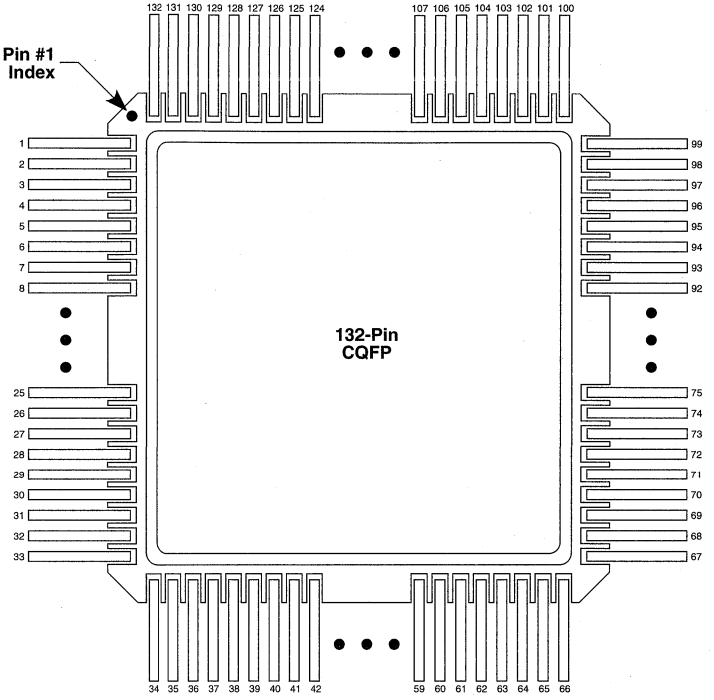
Signal	Location
CLKA or I/O	L4
CLKB or I/O	L5
DCLK or I/O	E4
GND	C4, B16, T17, R4, D4, D10, D16, E11, J5, K4, K16, L15, T4, T10, T16
HCLK or I/O	J16
IOCLK or I/O	T5
IOPCL or I/O	R16
MODE	A5
NC	E5
PRA OR I/O	J1
PRB or I/O	J17
SDI or I/O	B4
V_{CC}	C3, C10, C17, K3, K17, V3, V10, V17
V_{KS}	X7
V_{PP}	V7
V_{SV}	C13, X14

Notes:

1. *Unused I/O pins are designated as outputs by ALS and are driven low.*
2. *All unassigned pins are available for use as I/Os.*
3. *MODE = GND, except during device programming or debugging.*
4. *$V_{PP} = V_{CC}$, except during device programming.*
5. *$V_{SV} = V_{CC}$, except during device programming.*
6. *V_{KS} = GND, except during device programming.*

Package Pin Assignments (continued)

132-Pin CQFP (Top View)



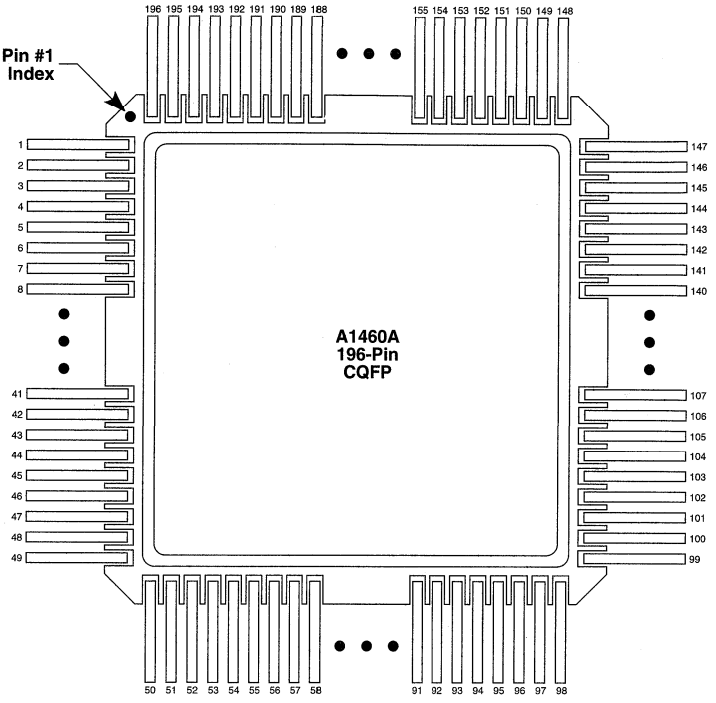
Signal	PIN Number
CLKA or I/O	116
CLKB or I/O	117
DCLK or I/O	131
GND	2, 10, 26, 36, 42, 58, 65, 74, 90, 101, 106, 122
HCLK or I/O	50
IOCLK or I/O	98
IOPCL or I/O	64
MODE	9
NC	1, 34, 66, 67, 99, 100, 132
PRA or I/O	118
PRB or I/O	48
SDI or I/O	3
V_{CC}	11, 27, 43, 59, 75, 91, 107, 123
V_{KS}	92
V_{PP}	89
V_{SV}	22, 78

Notes:

1. *Unused I/O pins are designated as outputs by ALS and are driven low.*
2. *All unassigned pins are available for use as I/Os.*
3. *MODE = GND, except during device programming or debugging.*
4. *$V_{PP} = V_{CC}$, except during device programming.*
5. *$V_{SV} = V_{CC}$, except during device programming*
6. *V_{KS} = GND, except during device programming.*

Package Pin Assignments (continued)

196-Pin CQFP (Top View)



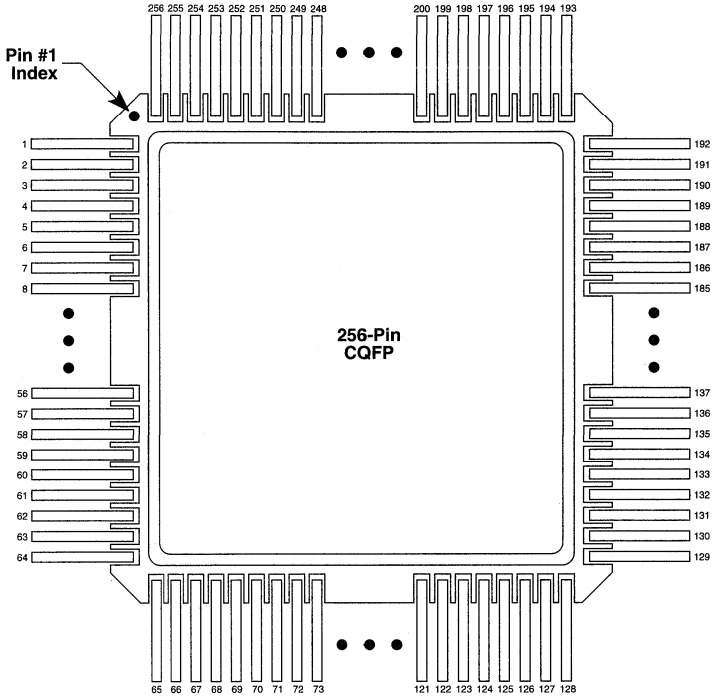
Signal	PIN Number
CLKA or I/O	172
CLKB or I/O	173
DCLK or I/O	196
GND	1, 13, 37, 51, 52, 64, 86, 98, 101, 112, 139, 149, 162, 183, 193
HCLK or I/O	77
IOCLK or I/O	148
IOPCL or I/O	100
MODE	11
PRA or I/O	174
PRB or I/O	75
SDI or I/O	2
V_{CC}	12, 38, 59, 94, 111, 140, 155, 189
V_{KS}	138
V_{PP}	137
V_{SV}	39, 110

Notes:

1. *Unused I/O pins are designated as outputs by ALS and are driven low.*
2. *All unassigned pins are available for use as I/Os.*
3. *MODE = GND, except during device programming or debugging.*
4. *$V_{PP} = V_{CC}$, except during device programming.*
5. *$V_{SV} = V_{CC}$, except during device programming*
6. *V_{KS} = GND, except during device programming.*

Package Pin Assignments (continued)

256-Pin CQFP (Top View)



Signal	PIN Number
CLKA or I/O	219
CLKB or I/O	220
DCLK or I/O	256
GND	1, 29, 31, 59, 91, 93, 110, 128, 158, 160, 176, 189, 222, 224, 240
HCLK or I/O	96
IOCLK or I/O	188
IOPCL or I/O	127
MODE	11
PRA or I/O	225
PRB or I/O	90
SDI or I/O	2
V_{CC}	28, 30, 92, 94, 159, 161, 221, 223
V_{KS}	175
V_{PP}	174
V_{SV}	141

Notes:

1. *Unused I/O pins are designated as outputs by ALS and are driven low.*
2. *All unassigned pins are available for use as I/Os.*
3. *MODE = GND, except during device programming or debugging.*
4. *$V_{PP} = V_{CC}$, except during device programming.*
5. *$V_{SV} = V_{CC}$, except during device programming*
6. *V_{KS} = GND, except during device programming.*

ACT™ 3 3.3 Volt Field Programmable Gate Arrays

Features

- 3.3V Functionality fully compliant with JEDEC specifications
- Highly Predictable Performance with 100% Automatic Placement and Routing
- 13 ns Clock-to-Output Times
- Up to 100 MHz On-Chip Performance
- Up to 200 User-Programmable I/O Pins
- Four Fast, Low-Skew Clock Networks
- More Than 500 Macro Functions
- Up to 10,000 Gate Array Equivalent Gates (up to 25,000 equivalent PLD Gates)
- Replaces up to 250 TTL Packages
- Replaces up to 100 20-pin PAL® Packages
- Up to 1153 Dedicated Flip-Flops
- I/O Drive to 12 mA
- VQFP, TQFP, BGA, and PLCC Packages
- Nonvolatile, User Programmable
- Low-power 0.8 micron CMOS Technology
- Fully Tested Prior to Shipment

Product Family Profile

Device	A14V15A	A14V25A	A14V40A	A14V60A	A14V100A
Capacity					
Gate Array Equivalent Gates	1,500	2,500	4,000	6,000	10,000
PLD Equivalent Gates	3,750	6,250	10,000	15,000	25,000
TTL Equivalent Packages (40 gates)	40	60	100	150	250
20-Pin PAL Equivalent Packages (100 gates)	15	25	40	60	100
Logic Modules	200	310	564	848	1,377
S-Module	104	160	288	432	697
C-Module	96	150	276	416	680
Dedicated Flip-Flops[1]	264	360	568	768	1,153
User I/Os (maximum)	80	100	140	168	228
Packages[2] (by pin count)					
PLCC	84	84	84	—	—
PQFP	—	—	—	208	—
RQFP	—	—	—	—	208
VQFP	100	100	100	—	—
TQFP	—	—	176	176	—
BGA	—	—	—	—	313
Performance (maximum, worst-case commercial)					
Chip-to-Chip[3]	63 MHz	63 MHz	60 MHz	59 MHz	57 MHz
Accumulators (16-bit)	33 MHz	33 MHz	33 MHz	33 MHz	33 MHz
Loadable Counter (16-bit)	56 MHz	56 MHz	56 MHz	52 MHz	52 MHz
Prescaled Loadable Counters (16-bit)	100 MHz	100 MHz	100 MHz	75 MHz	75 MHz
Datapath, Shift Registers	100 MHz	100 MHz	100 MHz	75 MHz	75 MHz
Clock-to-Output (pad-to-pad)	13.0 ns	13.0 ns	14.4 ns	15.0 ns	15.7 ns

Note:

1. *One flip-flop per S-Module, two flip-flops per I/O-Module.*
2. *See product plan on page 1-225 for package availability.*
3. *Clock-to-Output + Setup*

Description

The ACT 3 family, based on Actel's proprietary PLICE® antifuse technology and 0.8-micron double-metal, double-poly CMOS process, offers a high-performance programmable solution capable of 200 MHz on-chip performance and 6.5 nanosecond clock-to-output speeds. The ACT 3 family spans capacities from 1,500 to 10,000 gate array equivalent gates (up to 25,000 PLD gates), and offers very high pin-to-gate ratios, with up to 228 user I/Os for 10,000 gate designs.

Predictable Performance* (Worst-Case Commercial)

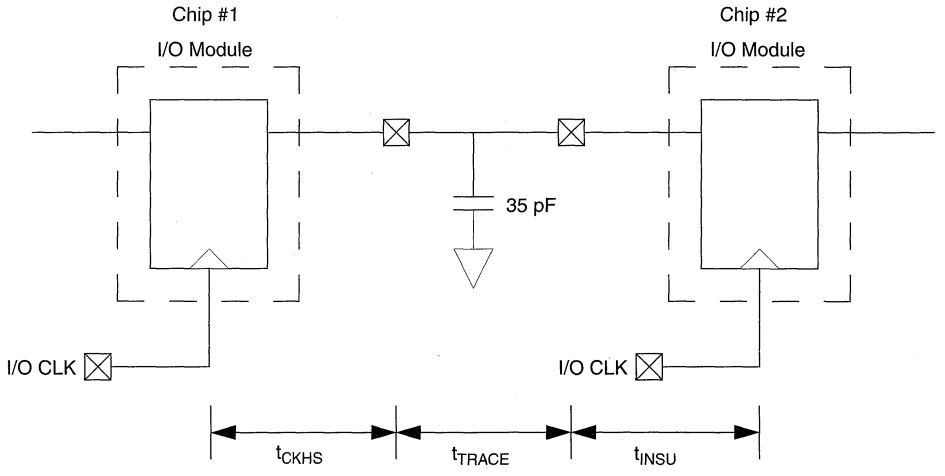
Accumulators (16-bit)	30–33 MHz
Loadable Counters (16-bit)	50–56 MHz
Prescaled Loadable Counters (16-bit)	90–100 MHz
Shift Registers	100–100 MHz

*See page 5 for further details.

The ACT 3 family represents the third generation of Actel Field Programmable Gate Arrays (FPGAs). The family improves on the proven ACT 2 family two-module architecture, consisting of combinatorial and sequential-combinatorial logic modules. The ACT 3 family offers registered I/O modules delivering 9 ns clock-to-out times. The devices contain four clock distribution networks, including dedicated array and I/O clocks, supporting very fast synchronous and asynchronous designs. In addition, routed clocks can be used to drive high fanout signals like resets or output enables, reducing buffering requirements.

The ACT 3 family is supported by the Designer and Designer Advantage systems, allowing logic design implementation with minimum effort. The systems offer Microsoft® Windows™ and X Window™ graphical user interfaces and integrate with the resident CAE system to provide a complete gate array design environment: schematic capture, simulation, fully automatic placement and routing, timing verification and device programming. The systems also include the ACTmap™ optimization and synthesis tool, and the ACTgen™ Macro Builder, a powerful macro function generator for counters, adders, and other structured blocks. The systems are available for 386/486/Pentium PCs and for HP™, and Sun™ workstations running Viewlogic®, Mentor Graphics®, and OrCAD™ tools.

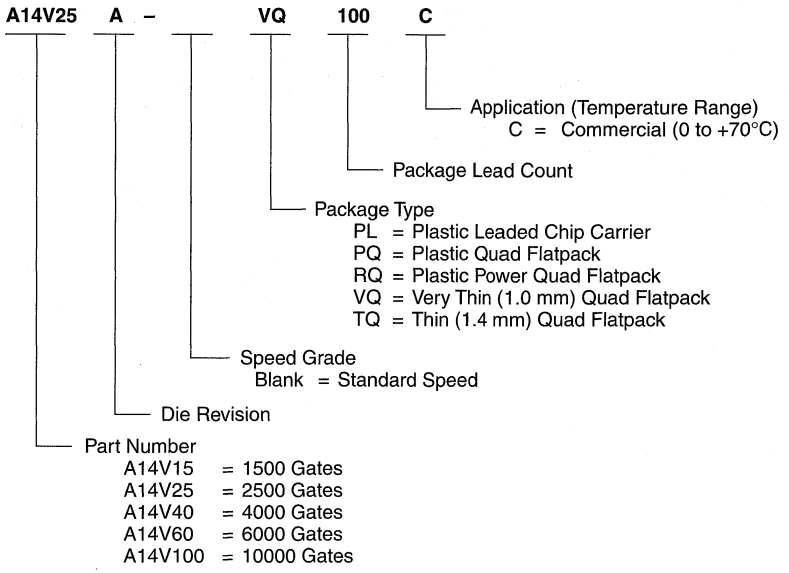
Chip-to-Chip Performance



Chip-to-Chip Performance (Worst-Case Commercial)

	t_{CKHS}	t_{TRACE}	t_{INSU}	Total	MHz
A14V25A	13.0	1.0	2.6	16.6	60
A14V60A	15.0	1.0	2.0	18.0	56

Ordering Information



Product Plan

	Speed Grade	Application
	Std	C
A14V15A Device		
84-pin Plastic Leaded Chip Carrier (PLCC)	✔	✔
100-pin Very Thin Quad Flatpack (VQFP)	✔	✔
A14V25A Device		
84-pin Plastic Leaded Chip Carrier (PLCC)	✔	✔
100-pin Very Thin Quad Flatpack (VQFP)	✔	✔
A14V40A Device		
84-pin Plastic Leaded Chip Carrier (PLCC)	✔	✔
100-pin Very Thin Quad Flatpack (VQFP)	✔	✔
176-pin Thin Quad Flatpack (TQFP)	✔	✔
A14V60A Device		
176-pin Thin Quad Flatpack (TQFP)	✔	✔
208-pin Plastic Quad Flatpack (PQFP)	✔	✔
A14V100A Device		
208-pin Power Quad Flatpack (RQFP)	✔	✔
313-pin Plastic Ball Grid Array (BGA)	✔	✔

Applications: *C = Commercial*

Availability: ✔ = *Available*, *P* = *Planned*, — = *Not Planned.*

Plastic Device Resources

Device Series	Logic Modules	Gates	User I/Os				
			PLCC	PQFP, RQFP	VQFP	TQFP	BGA
			84-pin	208-pin	100-pin	176-pin	313-pin
A14V15A	200	1500	70	—	80	—	—
A14V25A	310	2500	70	—	83	—	—
A14V40A	564	4000	70	—	83	140	—
A14V60A	848	6000	—	167	—	151	—
A14V100A	1377	10000	—	175	—	—	228

Pin Description

CLKA **Clock A (Input)**

TTL Clock input for clock distribution networks. The Clock input is buffered prior to clocking the logic modules. This pin can also be used as an I/O.

CLKB **Clock B (Input)**

TTL Clock input for clock distribution networks. The Clock input is buffered prior to clocking the logic modules. This pin can also be used as an I/O.

DCLK **Diagnostic Clock (Input)**

TTL Clock input for diagnostic probe and device programming. DCLK is active when the MODE pin is HIGH. This pin functions as an I/O when the MODE pin is LOW.

GND **Ground**

LOW supply voltage.

HCLK **Dedicated (Hard-wired) Array Clock (Input)**

TTL Clock input for sequential modules. This input is directly wired to each S-Module and offers clock speeds independent of the number of S-Modules being driven. This pin can also be used as an I/O.

I/O **Input/Output (Input, Output)**

The I/O pin functions as an input, output, three-state, or bidirectional buffer. Input and output levels are compatible with standard TTL and CMOS specifications. Unused I/O pins are automatically driven LOW by the ALS software.

IOCLK **Dedicated (Hard-wired) I/O Clock (Input)**

TTL Clock input for I/O modules. This input is directly wired to each I/O module and offers clock speeds independent of the number of I/O modules being driven. This pin can also be used as an I/O.

IOPCL **Dedicated (Hard-wired) I/O Preset/Clear (Input)**

TTL input for I/O preset or clear. This global input is directly wired to the preset and clear inputs of all I/O registers. This pin functions as an I/O when no I/O preset or clear macros are used.

MODE **Mode (Input)**

The MODE pin controls the use of diagnostic pins (DCLK, PRA, PRB, SDI). When the MODE pin is HIGH, the special functions are active. When the MODE pin is LOW, the pins function as I/Os.

NC **No Connection**

This pin is not connected to circuitry within the device.

PRA **Probe A (Output)**

The Probe A pin is used to output data from any user-defined design node within the device. This independent diagnostic pin can be used in conjunction with the Probe B pin to allow real-time diagnostic output of any signal path within the device. The Probe A pin can be used as a user-defined I/O when debugging has been completed. The pin's probe capabilities can be permanently disabled to protect programmed design confidentiality. PRA is accessible when the MODE pin is HIGH. This pin functions as an I/O when the MODE pin is LOW.

PRB **Probe B (Output)**

The Probe B pin is used to output data from any user-defined design node within the device. This independent diagnostic pin can be used in conjunction with the Probe A pin to allow real-time diagnostic output of any signal path within the device. The Probe B pin can be used as a user-defined I/O when debugging has been completed. The pin's probe capabilities can be permanently disabled to protect programmed design confidentiality. PRB is accessible when the MODE pin is HIGH. This pin functions as an I/O when the MODE pin is LOW.

SDI **Serial Data Input (Input)**

Serial data input for diagnostic probe and device programming. SDI is active when the MODE pin is HIGH. This pin functions as an I/O when the MODE pin is LOW.

V_{CC} **3.3 V Supply Voltage**

HIGH supply voltage.

V_{KS} **Programming Voltage**

Supply voltage used for device programming. This pin must be connected to GND during normal operation.

V_{PP} Programming Voltage

Supply voltage used for device programming. This pin must be connected to V_{CC} during normal operation.

V_{SV} Programming Voltage

Supply voltage used for device programming. This pin must be connected to V_{CC} during normal operation.

Architecture

This section of the data sheet is meant to familiarize the user with the architecture of the ACT 3 family of FPGA devices. A generic description of the family will be presented first, followed by a detailed description of the logic blocks, the routing structure, the antifuses, and the special function circuits. The on-chip circuitry required to program the devices is not covered.

Topology

The ACT 3 family architecture is composed of six key elements: Logic modules, I/O modules, I/O Pad Drivers, Routing Tracks, Clock Networks, and Programming and Test Circuits. The basic structure is similar for all devices in the family, differing only in the number of rows, columns, and I/Os. The array itself consists of alternating rows of modules and channels. The logic modules and channels are in the center of the array; the I/O modules are located along the array periphery. A simplified floor plan is depicted in Figure 1.

Logic Modules

ACT 3 logic modules are enhanced versions of the ACT 2 family logic modules. As in the ACT 2 family, there are two types of modules: C-modules and S-modules. The C-module is functionally equivalent to the ACT 2 C-module and implements high fanin combinatorial macros, such as 5-input AND, 5-input OR, and so on. It is available for use as the CM8 hard macro. The S-module is designed to implement high-speed sequential functions within a single module. S-modules consist of a full C-module driving a flip-flop, which allows an additional level of logic to be implemented without additional propagation delay. It is available for use as the

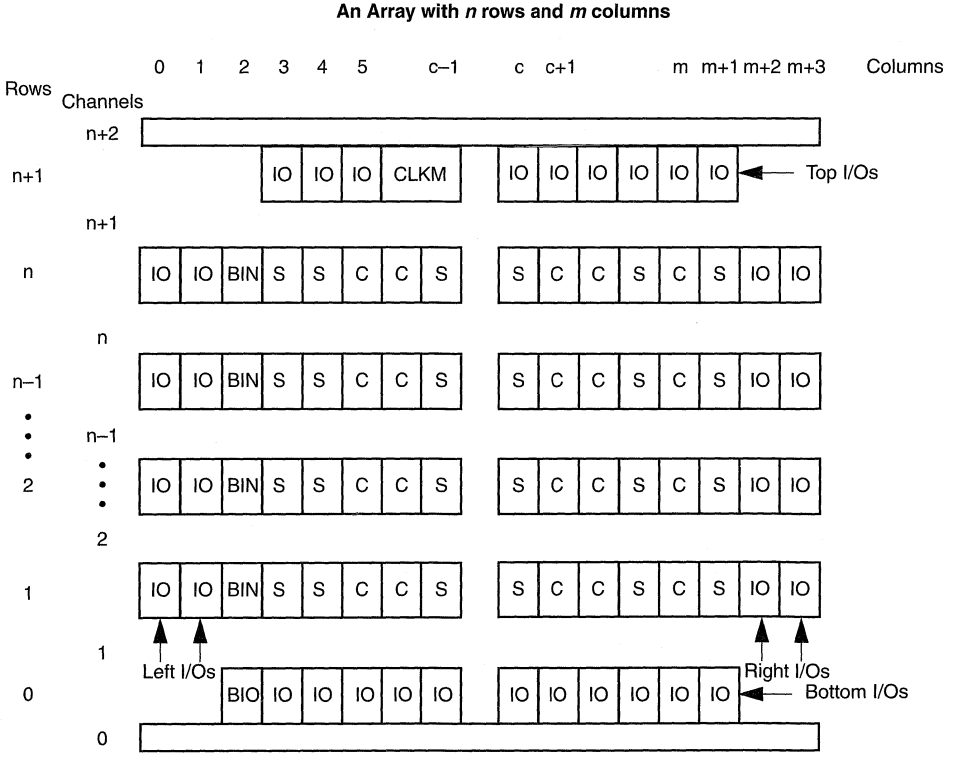


Figure 1 • Generalized Floor Plan of ACT 3 Device

DFM8A/B and DLM8A/B hard macros. C-modules and S-modules are arranged in pairs called module-pairs. Module-pairs are arranged in alternating patterns and make up the bulk of the array. This arrangement allows the placement software to support two-module macros of four types (CC, CS, SC, and SS). The C-module implements the following function:

Y = !S1 * !S0 * D00 + !S1 * S0 * D01 + S1 * !S0 * D10 + S1 * S0 * D11

where: S0 = A0 * B0 and S1 = A1 + B1

The S-module contains a full implementation of the C-module plus a clearable sequential element that can either implement a latch or flip-flop function. The S-module can therefore implement any function implemented by the C-module. This allows complex combinatorial-sequential functions to be implemented with no delay penalty. The Action Logic System will automatically combine any C-module macro driving an S-module macro into the S-module, thereby freeing up a logic module and eliminating a module delay.

The clear input CLR is accessible from the routing channel. In addition, the clock input may be connected to one of three clock networks: CLK0, CLK1, or HCLK. The C-module and S-module functional descriptions are shown in Figures 2 and 3. The clock selection multiplexor selects the clock input to the S-module.

I/Os

I/O Modules

I/O modules provide an interface between the array and the I/O Pad Drivers. I/O modules are located in the array and

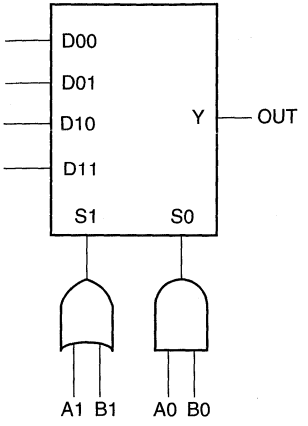


Figure 2 • C-Module Diagram

access the routing channels in a similar fashion to logic modules. There are two types of I/O modules: side and top/bottom. The I/O module schematic is shown in Figure 4. UO1 and UO2 are inputs from the routing channel, one for the routing channel above and one for the routing channel below the module. The top/bottom I/O modules interact with only one channel and therefore have only one UO input. The signals DataIn and DataOut connect to the I/O pad driver. Each I/O module contains two D-type flip-flops. Each flip-flop is connected to the dedicated I/O clock (IOCLK). Each flip-flop can be bypassed by nonsequential I/Os. In addition, each flip-flop contains a data enable input that can be accessed from the routing channels (ODE and IDE). The

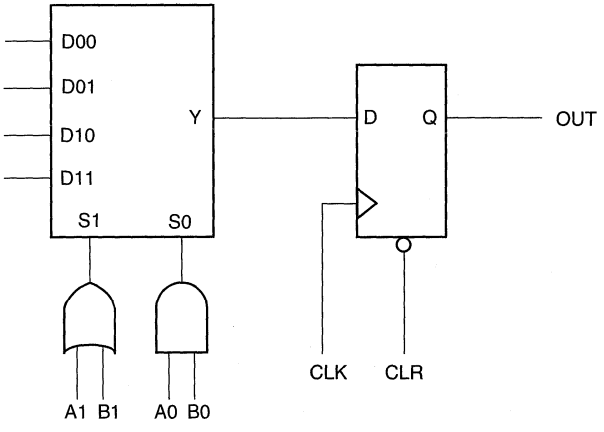


Figure 3 • S-Module Diagram

asynchronous preset/clear input is driven by the dedicated preset/clear network (IOPCL). Either preset or clear can be selected individually on an I/O module by I/O module basis.

The I/O module output Y is used to bring Pad signals into the array *or* to feed the output register back into the array. This allows the output register to be used in high-speed state machine applications. Side I/O modules have a dedicated output segment for Y extending into the routing channels above and below (similar to logic modules). Top/Bottom I/O modules have no dedicated output segment. Signals coming into the chip from the top or bottom are routed using F-fuses and LVTs (F-fuses and LVTs are explained in detail in the routing section).

I/O Pad Drivers

All pad drivers are capable of being tristate. Each buffer connects to an associated I/O module with four signals: OE (Output Enable), IE (Input Enable), DataOut, and DataIn. Certain special signals used only during programming and test also connect to the pad drivers: OUTEN (global output enable), INEN (global input enable), and SLEW (individual slew selection). See Figure 5.

Special I/Os

The special I/Os are of two types: temporary and permanent. Temporary special I/Os are used during programming and testing. They function as normal I/Os when the MODE pin is inactive. Permanent special I/Os are user programmed as either normal I/Os or special I/Os. Their function does not change once the device has been programmed. The permanent special I/Os consist of the array clock input buffers (CLKA and CLKB), the hard-wired array clock input buffer (HCLK), the hard-wired I/O clock input buffer (IOCLK), and the hard-wired I/O register preset/clear input buffer (IOPCL). Their function is determined by the I/O macros selected.

Clock Networks

The ACT 3 architecture contains four clock networks: two high-performance dedicated clock networks and two general purpose routed networks. The high-performance networks function up to 100 MHz, while the general purpose routed networks function up to 75 MHz.

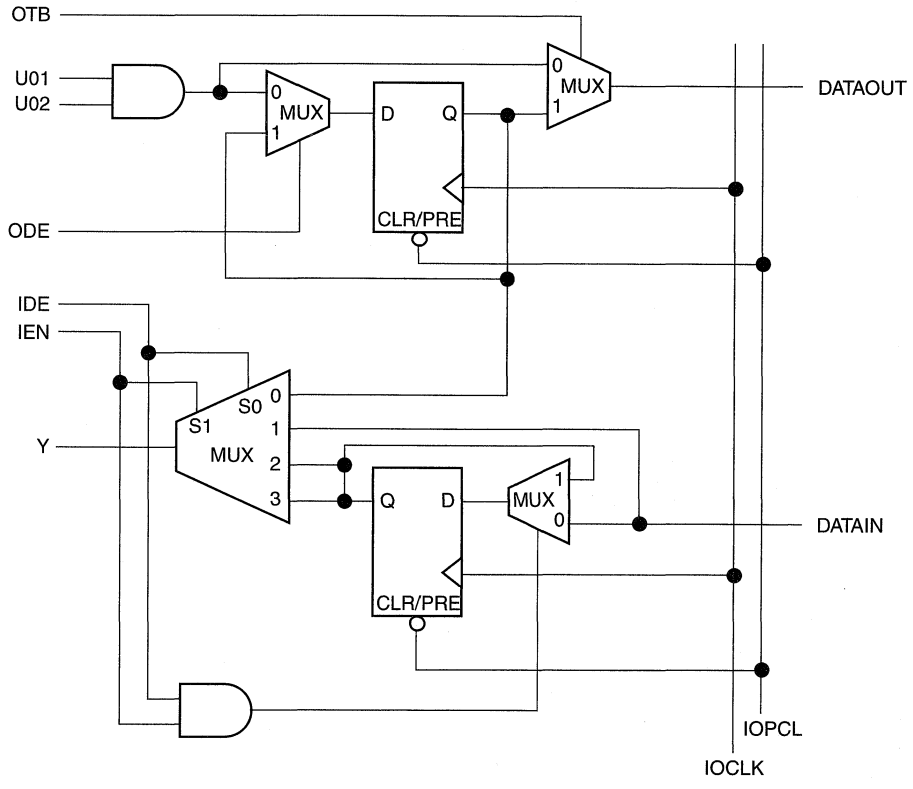


Figure 4 • Functional Diagram for I/O Module

Dedicated Clocks

Dedicated clock networks support high performance by providing sub-nanosecond skew and guaranteed performance. Dedicated clock networks contain no programming elements in the path from the I/O Pad Driver to the input of S-modules or I/O modules. There are two dedicated clock networks: one for the array registers (HCLK), and one for the I/O registers (IOCLK). The clock networks are accessed by special I/Os.

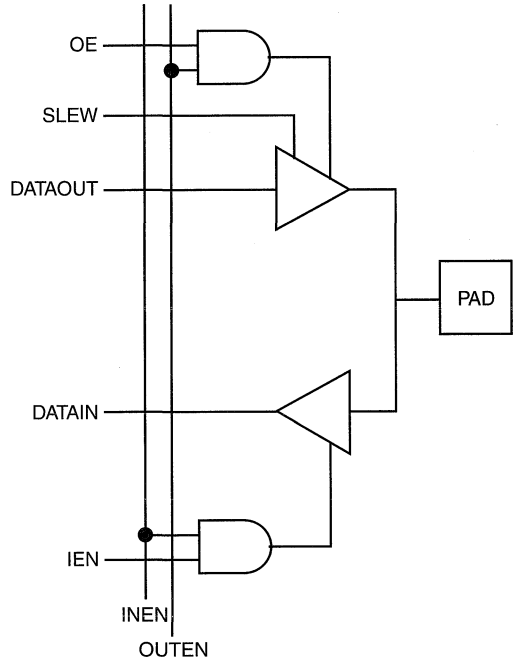


Figure 5 • Function Diagram for I/O Pad Driver

Routed Clocks

The routed clock networks are referred to as CLK0 and CLK1. Each network is connected to a clock module (CLKMOD) that selects the source of the clock signal and may be driven as follows (see Figure 6):

- externally from the CLKA pad
- externally from the CLKB pad
- internally from the CLKINA input
- internally from the CLKINB input

The clock modules are located in the top row of I/O modules. Clock drivers and a dedicated horizontal clock track are located in each horizontal routing channel. The function of the clock module is determined by the selection of clock macros from the macro library. The macro CLKBUF is used to connect one of the two external clock pins to a clock network, and the macro CLKINT is used to connect an internally generated clock signal to a clock network. Since both clock networks are identical, the user does not care whether CLK0 or CLK1 is being used. Routed clocks can also be used to drive high fanout nets like resets, output enables, or data enables. This saves logic modules and results in performance increases in some cases.

Routing Structure

The ACT 3 architecture uses vertical and horizontal routing tracks to connect the various logic and I/O modules. These routing tracks are metal interconnects that may either be of continuous length or broken into segments. Segments can be joined together at the ends using antifuses to increase their lengths up to the full length of the track.

Horizontal Routing

Horizontal channels are located between the rows of modules and are composed of several routing tracks. The horizontal routing tracks within the channel are divided into one or more segments. The minimum horizontal segment length is the width of a module-pair, and the maximum horizontal segment length is the full length of the channel. Any segment that spans more than one-third the row length is considered a long horizontal segment. A typical channel is shown in Figure 7. Undedicated horizontal routing tracks are used to route signal nets. Dedicated routing tracks are used for the global clock networks and for power and ground tie-off tracks.

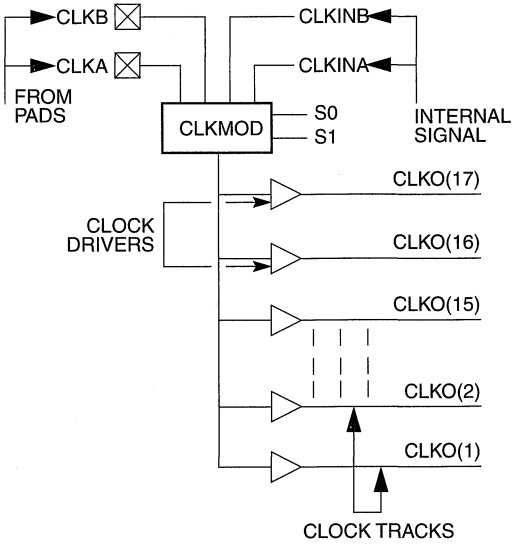


Figure 6 • Clock Networks

Vertical Routing

Other tracks run vertically through the modules. Vertical tracks are of three types: input, output, and long. Vertical tracks are also divided into one or more segments. Each segment in an input track is dedicated to the input of a particular module. Each segment in an output track is dedicated to the output of a particular module. Long segments are uncommitted and can be assigned during routing. Each output segment spans four channels (two above and two below), except near the top and bottom of the array where edge effects occur. LVTs contain either one or two segments. An example of vertical routing tracks and segments is shown in Figure 8.

Antifuse Connections

An antifuse is a "normally open" structure as opposed to the normally closed fuse structure used in PROMs or PALs. The use of antifuses to implement a programmable logic device results in highly testable structures as well as an efficient programming architecture. The structure is highly testable because there are no preexisting connections; temporary connections can be made using pass transistors. These temporary connections can isolate individual antifuses to be programmed as well as isolate individual circuit structures to be tested. This can be done both before and after programming. For example, all metal tracks can be tested for continuity and shorts between adjacent tracks, and the functionality of all logic modules can be verified.

Four types of antifuse connections are used in the routing structure of the ACT 3 array. (The physical structure of the antifuse is identical in each case; only the usage differs.) Table 1 shows four types of antifuses.

Table 1 • Antifuse Types

XF	Horizontal-to-Vertical Connection
HF	Horizontal-to-Horizontal Connection
VF	Vertical-to-Vertical Connection
FF	"Fast" Vertical Connection

Examples of all four types of connections are shown in Figures 7 and 8.

Module Interface

Connections to Logic and I/O modules are made through vertical segments that connect to the module inputs and outputs. These vertical segments lie on vertical tracks that span the entire height of the array.

Module Input Connections

The tracks dedicated to module inputs are segmented by pass transistors in each module row. During normal user

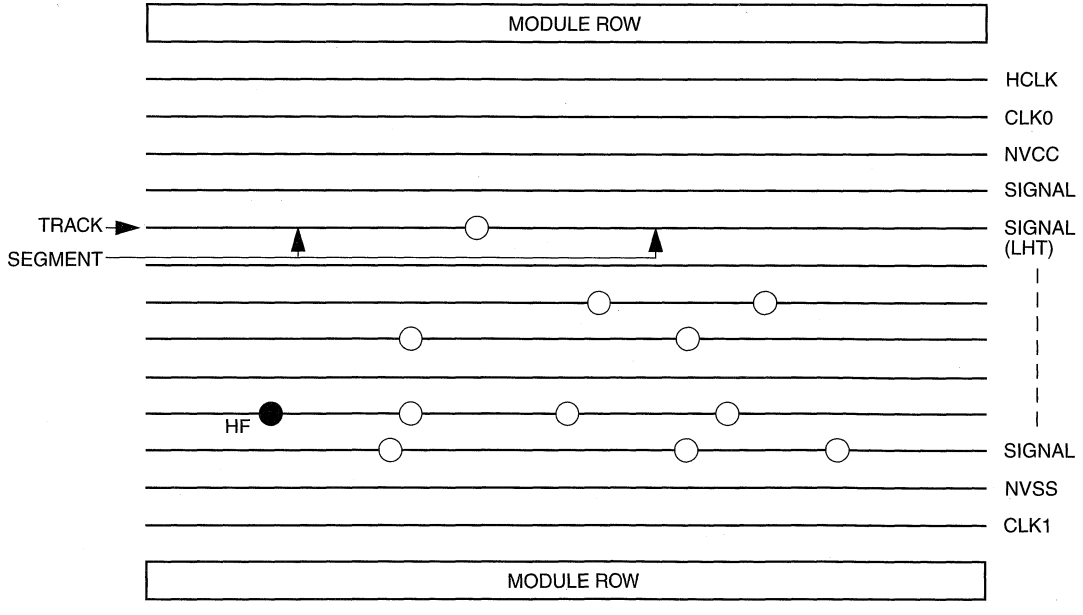


Figure 7 • Horizontal Routing Tracks and Segments

operation, the pass transistors are inactive, which isolates the inputs of a module from the inputs of the module directly above or below it. During certain test modes, the pass transistors are active to verify the continuity of the metal tracks. Vertical input segments span only the channel above or the channel below. The logic modules are arranged such that half of the inputs are connected to the channel above and half of the inputs to segments in the channel below as shown in Figure 9.

Module Output Connections

Module outputs have dedicated output segments. Output segments extend vertically two channels above and two channels below, except at the top or bottom of the array. Output segments twist, as shown in Figure 9, so that only four vertical tracks are required.

LVT Connections

Outputs may also connect to nondedicated segments called Long Vertical Tracks (LVTs). Each module pair in the array shares four LVTs that span the length of the column. Any module in the column pair can connect to one of the LVTs in the column using an FF connection. The FF connection uses antifuses connected directly to the driver stage of the module output, bypassing the isolation transistor. FF antifuses are programmed at a higher current level than HF, VF, or XF antifuses to produce a lower resistance value.

Antifuse Connections

In general every intersection of a vertical segment and a horizontal segment contains an unprogrammed antifuse (XF-type). One exception is in the case of the clock networks.

Clock Connections

To minimize loading on the clock networks, a subset of inputs has antifuses on the clock tracks. Only a few of the C-module and S-module inputs can be connected to the clock networks. To further reduce loading on the clock network, only a subset of the horizontal routing tracks can connect to the clock inputs of the S-module.

Programming and Test Circuits

The array of logic and I/O modules is surrounded by test and programming circuits controlled by the temporary special I/O pins MODE, SDI, and DCLK. The function of these pins is similar to all ACT family devices. The ACT 3 family also includes support for two Actionprobe® circuits allowing complete observability of any logic or I/O module in the array using the temporary special I/O pins, PRA and PRB.

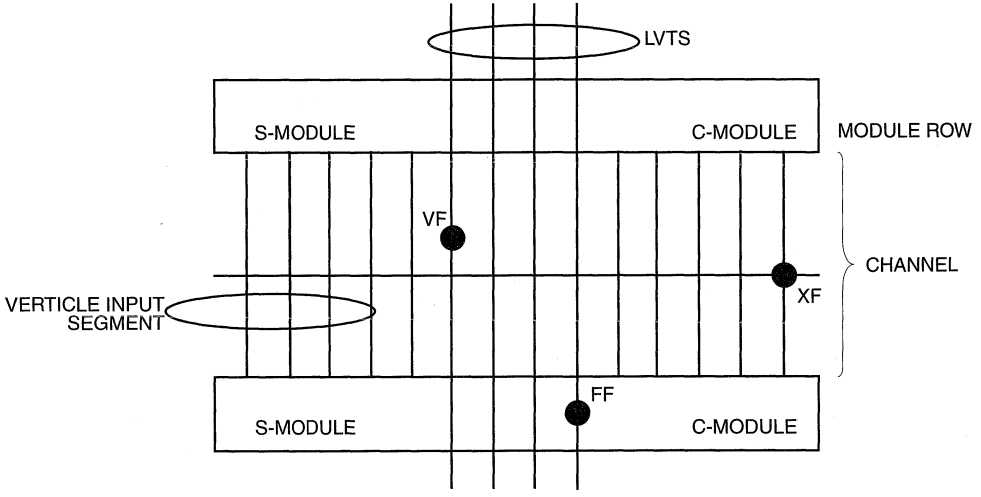


Figure 8 • Vertical Routing Tracks and Segments

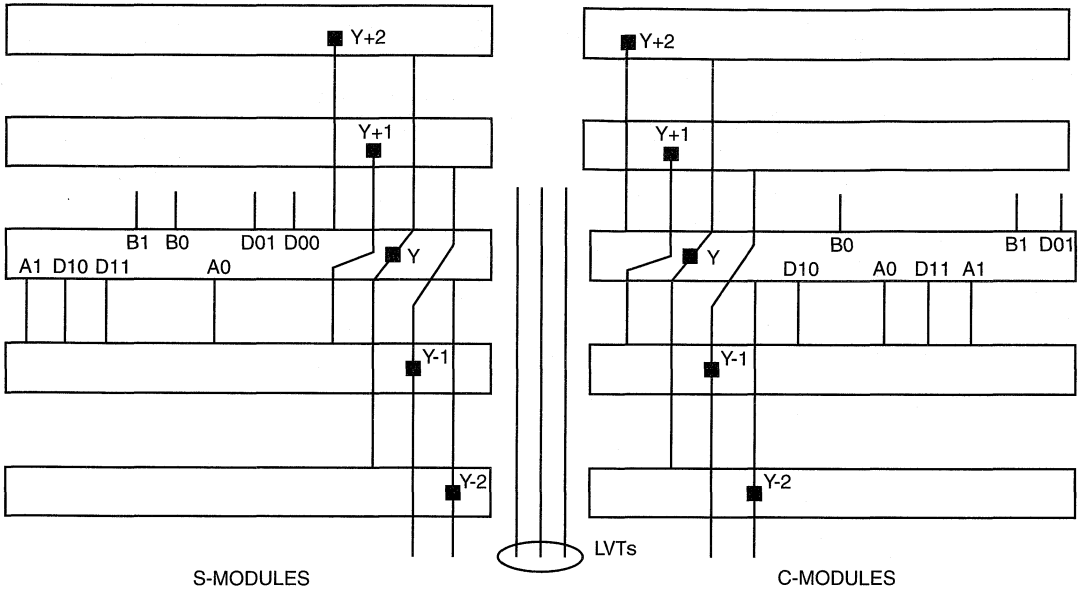


Figure 9 • Logic Module Routing Interface

Absolute Maximum Ratings[1]

Free air temperature range

Symbol	Parameter	Limits	Units
V_{CC}	DC Supply Voltage[2]	–0.5 to +7.0	V
V_I	Input Voltage	–0.5 to V_{CC} +0.5	V
V_O	Output Voltage	–0.5 to V_{CC} +0.5	V
I_{IO}	I/O Source Sink Current[3]	±20	mA
T_{STG}	Storage Temperature	–65 to +150	°C

Notes:

1. *Stresses beyond those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. Exposure to absolute maximum rated conditions for extended periods may affect device reliability. Device should not be operated outside the Recommended Operating Conditions.*
2. *V_{PP}, $V_{SV} = V_{CC}$, except during device programming.*
3. *Device inputs are normally high impedance and draw extremely low current. However, when input voltage is greater than V_{CC} + 0.5 V or less than GND – 0.5 V, the internal protection diodes will forward bias and can draw excessive current.*

Recommended Operating Conditions

Parameter	Commercial	Units
Temperature Range[1]	0 to +70	°C
Power Supply Tolerance	3.0 to 3.6	V

Note:

1. *Ambient temperature (T_A) is used for commercial.*

Electrical Specifications

Parameter		Commercial		Units
		Min.	Max.	
V_{OH}[1]	(I_{OH} = –4 mA)	2.15		V
	(I_{OH} = –3.2 mA)	2.4		V
V_{OL}[1]	(I_{OL} = 6 mA)		0.4	V
V_{IL}		–0.3	0.8	V
V_{IH}		2.0	V_{CC} + 0.3	V
Input Transition Time t_R, t_F[2]			500	ns
C_{IO} I/O Capacitance[2, 3]			10	pF
Standby Current, I_{CC}[4] (typical = 0.3 mA)			0.75	mA
Leakage Current[5]		–10	10	µA

Notes:

1. *Only one output tested at a time. V_{CC} = min.*
2. *Not tested, for information only.*
3. *Includes worst-case 84-pin PLCC package capacitance. V_{OUT} = 0 V, f = 1 MHz.*
4. *Typical standby current = 0.3 mA. All outputs unloaded. All inputs = V_{CC} or GND.*
5. *V_O, V_{IN} = V_{CC} or GND.*

Package Thermal Characteristics

The device junction to case thermal characteristic is θjc, and the junction to ambient air characteristic is θja. The thermal characteristics for θja are shown with two different air flow rates.

Maximum junction temperature is 150°C.

A sample calculation of the absolute maximum power dissipation allowed for a RQFP 208-pin package at commercial temperature and still air is as follows:

$$\text{Absolute Maximum Power Allowed} = \frac{\text{Max. junction temp. (°C)} - \text{Max. ambient temp. (°C)}}{\theta ja\ (°C/W)} = \frac{150°C - 70°C}{17°C/W} = 4.7\ W$$

Package Type	Pin Count	θja Still Air	θja 300 ft/min	Units
Plastic Quad Flatpack[1]	208	33	26	°C/W
Very Thin Quad Flatpack	100	43	35	°C/W
Thin Quad Flatpack	176	32	25	°C/W
Power Quad Flatpack	208	17	13	°C/W
Plastic Leaded Chip Carrier[2]	84	37	28	°C/W
Plastic Ball Grid Array	313	23	17	°C/W

Notes:

1. Maximum Power Dissipation for 176-pin TQFP package is 2.5 Watts, 208-pin PQFP package is 2.4 Watts, and 100-pin VQFP package is 1.9 Watts.

2. Maximum Power Dissipation for PLCC package is 2.2 Watts, 208-pin RQFP package is 4.7 Watts, and 313-pin BGA packages is 3.5 Watts.

General Power Equation

$$P = [I_{CC}standby + I_{CC}active] * V_{CC} + I_{OL} * V_{OL} * N + I_{OH} * (V_{CC} - V_{OH}) * M$$

Where:

I_{CC}standby is the current flowing when no inputs or outputs are changing.

I_{CC}active is the current flowing due to CMOS switching.

I_{OL}, I_{OH} are TTL sink/source currents.

V_{OL}, V_{OH} are TTL level output voltages.

N equals the number of outputs driving TTL loads to V_{OL}.

M equals the number of outputs driving TTL loads to V_{OH}.

An accurate determination of N and M is problematical because their values depend on the family type, design details, and on the system I/O. The power can be divided into two components: static and active.

Static Power Component

Actel FPGAs have small static power components that result in lower power dissipation than PALs or PLDs. By integrating multiple PALs/PLDs into one FPGA, an even greater reduction in board-level power dissipation can be achieved.

The power due to standby current is typically a small component of the overall power. Standby power is calculated below for commercial, worst case conditions.

I_{CC}	V_{CC}	Power
0.75 mA	3.6 V	2.70 mW (max)
0.30 mA	3.3 V	0.99 mW (typ)

Active Power Component

Power dissipation in CMOS devices is usually dominated by the active (dynamic) power dissipation. This component is frequency dependent, a function of the logic and the

external I/O. Active power dissipation results from charging internal chip capacitances of the interconnect, unprogrammed antifuses, module inputs, and module outputs, plus

external capacitance due to PC board traces and load device inputs. An additional component of the active power dissipation is the totem-pole current in CMOS transistor pairs. The net effect can be associated with an equivalent capacitance that can be combined with frequency and voltage to represent active power dissipation.

Equivalent Capacitance

The power dissipated by a CMOS circuit can be expressed in Equation 1.

$$\text{Power (uW)} = C_{EQ} * V_{CC}^2 * F(1)$$

Where:

C_{EQ} is the equivalent capacitance expressed in pF.

V_{CC} is the power supply in volts.

F is the switching frequency in MHz.

Equivalent capacitance is calculated by measuring I_{CC} active at a specified frequency and voltage for each circuit component of interest. Measurements have been made over a range of frequencies at a fixed value of VCC. Equivalent capacitance is frequency independent so that the results may be used over a wide range of operating conditions. Equivalent capacitance values are shown below.

C_{EQ} Values for Actel FPGAs

Modules (C_{EQM})	5.7
Input Buffers (C_{EQI})	5.4
Output Buffers (C_{EQO})	7.8
Routed Array Clock Buffer Loads (C_{EQCR})	1.4
Dedicated Clock Buffer Loads (C_{EQCD})	0.6
I/O Clock Buffer Loads (C_{EQCI})	0.7

To calculate the active power dissipated from the complete design, the switching frequency of each part of the logic must be known. Equation 2 show a piece-wise linear summation over all components.

$$\text{Power} = V_{CC}^2 * [(m * C_{EQM} * f_m)_{modules} + (n * C_{EQI} * f_n)_{inputs} + (p * (C_{EQO} + C_L) * f_p)_{outputs} + 0.5 * (q_1 * C_{EQCR} * f_{q1})_{routed\_Clk1} + (r_1 * f_{q1})_{routed\_Clk1} + 0.5 * (q_2 * C_{EQCR} * f_{q2})_{routed\_Clk2} + (r_2 * f_{q2})_{routed\_Clk2} + 0.5 * (s_1 * C_{EQCD} * f_{s1})_{dedicated\_Clk} + (s_2 * C_{EQCI} * f_{s2})_{IO\_Clk}] \quad (2)$$

Where:

m = Number of logic modules switching at f_m

n = Number of input buffers switching at f_n

p = Number of output buffers switching at f_p

q_1 = Number of clock loads on the first routed array clock

q_2 = Number of clock loads on the second routed array clock

r_1 = Fixed capacitance due to first routed array clock

r_2 = Fixed capacitance due to second routed array clock

s_1 = Fixed number of clock loads on the dedicated array clock

s_2 = Fixed number of clock loads on the dedicated I/O clock

C_{EQM} = Equivalent capacitance of logic modules in pF

C_{EQI} = Equivalent capacitance of input buffers in pF

C_{EQO} = Equivalent capacitance of output buffers in pF

C_{EQCR} = Equivalent capacitance of routed array clock in pF

C_{EQCD} = Equivalent capacitance of dedicated array clock in pF

C_{EQCI} = Equivalent capacitance of dedicated I/O clock in pF

C_L = Output lead capacitance in pF

f_m = Average logic module switching rate in MHz

f_n = Average input buffer switching rate in MHz

f_p = Average output buffer switching rate in MHz

f_{q1} = Average first routed array clock rate in MHz

f_{q2} = Average second routed array clock rate in MHz

f_{s1} = Average dedicated array clock rate in MHz

f_{s2} = Average dedicated I/O clock rate in MHz

Fixed Capacitance Values for Actel FPGAs (pF)

Device Type	r_1 routed clock 1	r_2 routed clock 2
A14V15A	57	57
A14V25A	72	72
A14V40A	100	100
A14V60A	157	157
A14V100A	185	185

Fixed Clock Loads

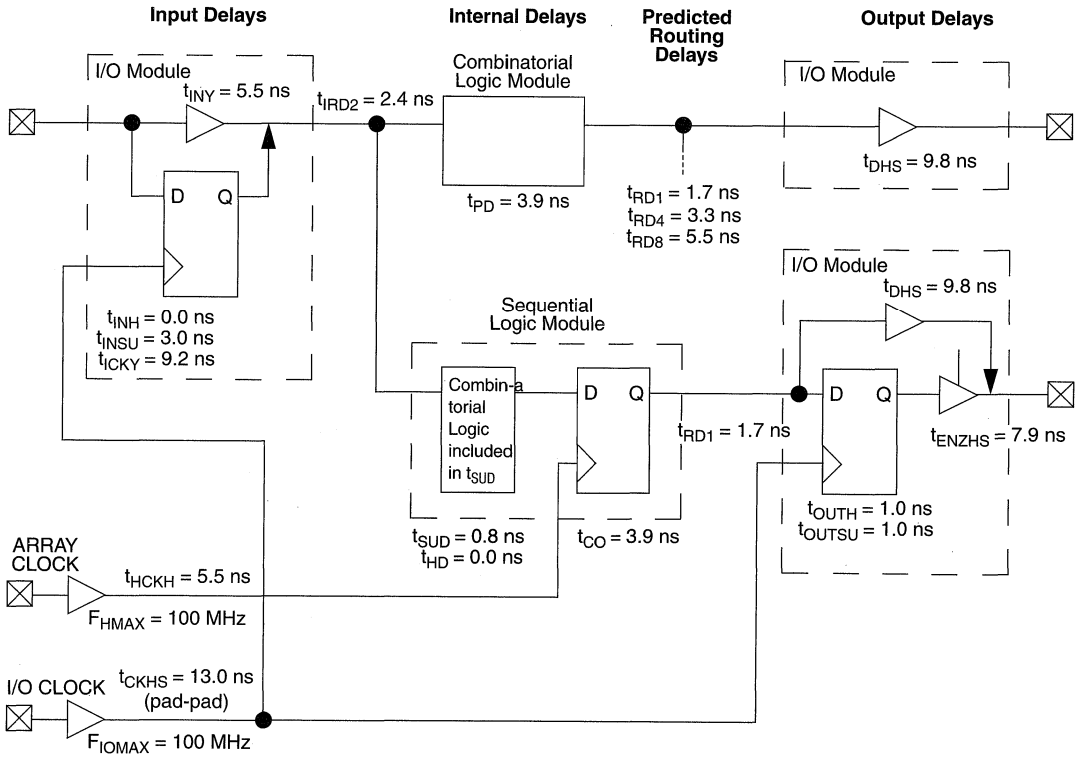
Device Type	S_1 dedicated array clock	S_2 dedicated I/O clock
A14V15A	104	80
A14V25A	160	100
A14V40A	288	140
A14V60A	432	168
A14V100A	697	228V

Determining Average Switching Frequency

To determine the switching frequency for a design, you must have a detailed understanding of the data input values to the circuit. The following guidelines are meant to represent worst-case scenarios so that they can be generally used to predict the upper limits of power dissipation. These guidelines are as follows:

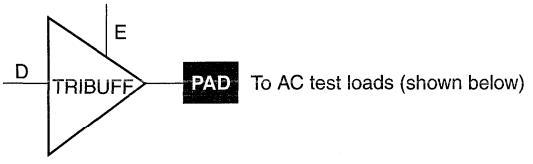
Logic Modules (m)	80% of modules
Inputs switching (n)	# inputs/4
Outputs switching (p)	# output/4
First routed array clock loads (q_1)	40% of sequential modules
Second routed array clock loads (q_2)	40% of sequential modules
Load capacitance (C_L)	35 pF
Average logic module switching rate (f_m)	F/10
Average input switching rate (f_n)	F/5
Average output switching rate (f_p)	F/10
Average first routed array clock rate (f_{q1})	F/2
Average second routed array clock rate (f_{q2})	F/2
Average dedicated array clock rate (f_{s1})	F
Average dedicated I/O clock rate (f_{s2})	F

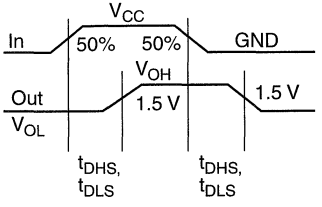
ACT 3 Timing Model*

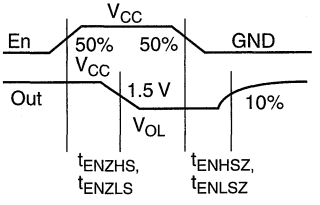


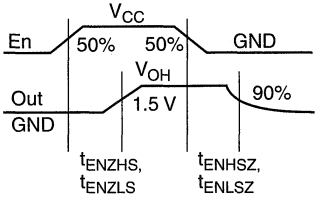
**Values shown for A14V25A.*

Output Buffer Delays



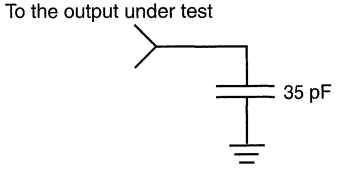




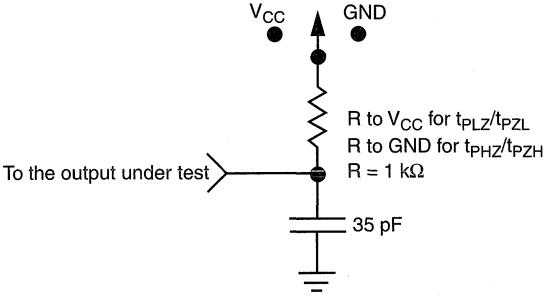


AC Test Loads

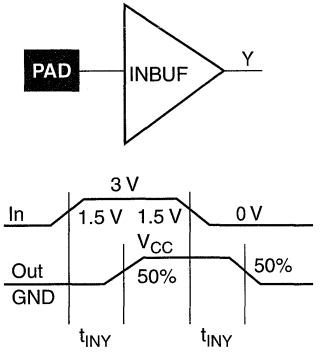
Load 1
(Used to measure propagation delay)



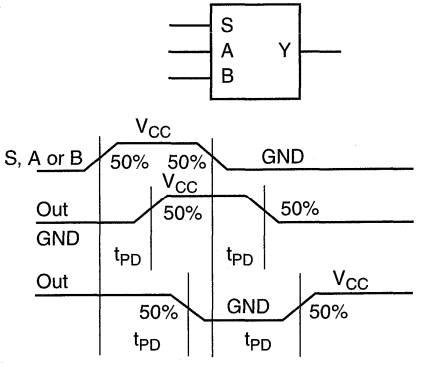
Load 2
(Used to measure rising/falling edges)



Input Buffer Delays

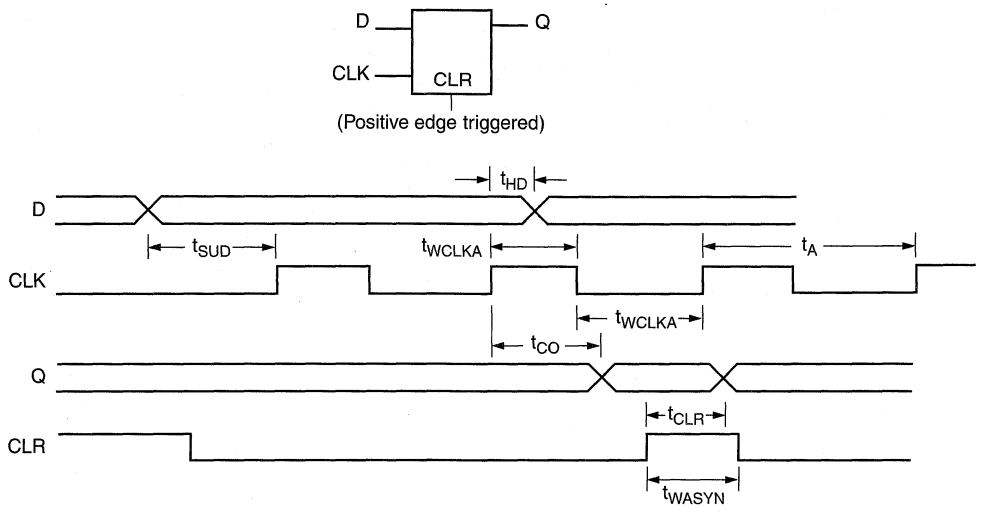


Module Delays

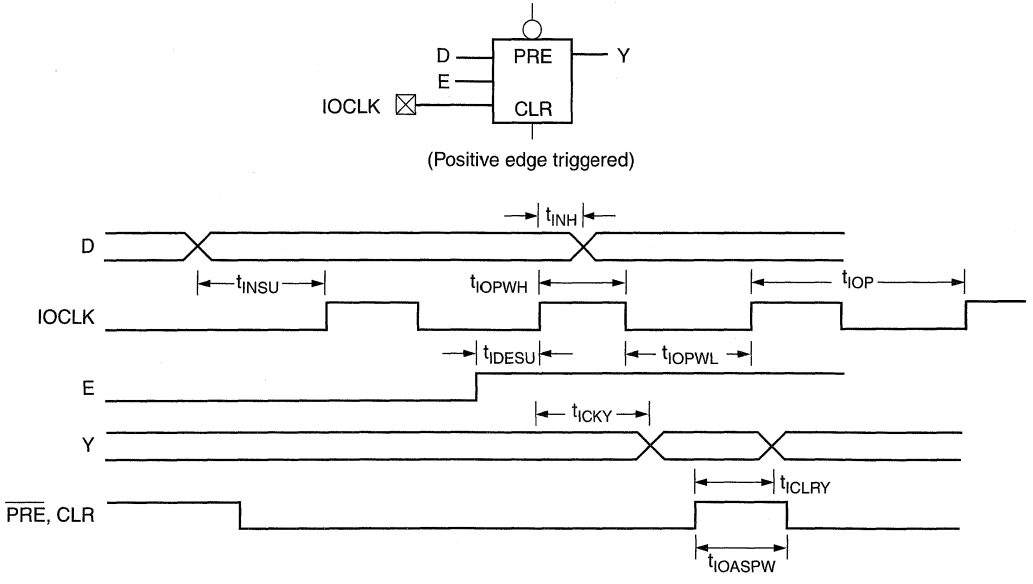


Sequential Module Timing Characteristics

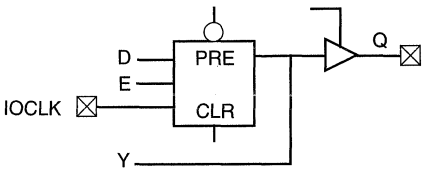
Flip-Flops



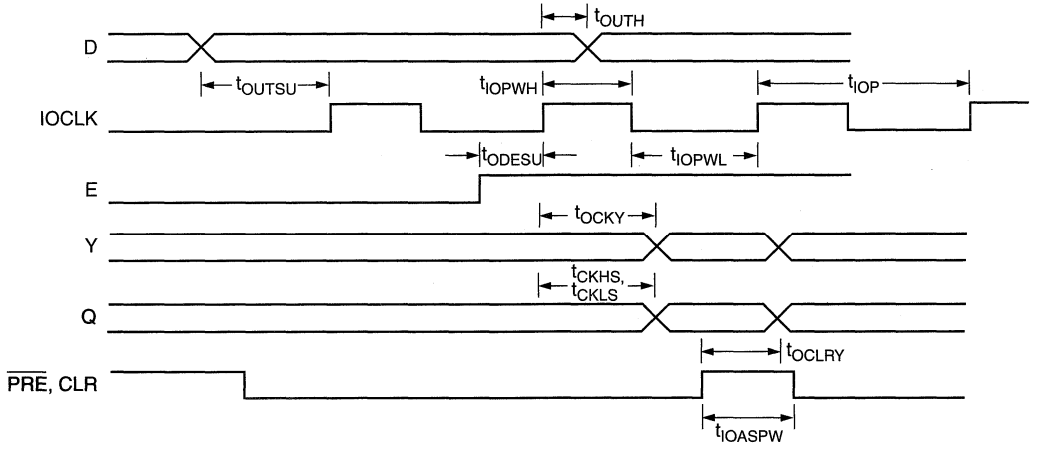
I/O Module: Sequential Input Timing Characteristics



I/O Module: Sequential Output Timing Characteristics



(Positive edge triggered)



Timing Characteristics

Timing characteristics for ACT 3 devices fall into three categories: family dependent, device dependent, and design dependent. The input and output buffer characteristics are common to all ACT 3 family members. Internal routing delays are device dependent. Design dependency means actual delays are not determined until after placement and routing of the user's design is complete. Delay values may then be determined by using the ALS Timer utility or performing simulation with post-layout delays.

Critical Nets and Typical Nets

Propagation delays are expressed only for typical nets, which are used for initial design performance evaluation. Critical net delays can then be applied to the most time-critical paths. Critical nets are determined by net property assignment prior to placement and routing. Up to 6% of the nets in a design may be designated as critical, while 90% of the nets in a design are typical.

Long Tracks

Some nets in the design use long tracks. Long tracks are special routing resources that span multiple rows, columns, or modules. Long tracks employ three and sometimes four antifuse connections. This increases capacitance and resistance, resulting in longer net delays for macros connected to long tracks. Typically up to 6% of nets in a fully utilized device require long tracks. Long tracks contribute approximatley 4 ns to 14 ns delay. This additional delay is represented statistically in higher fanout (FO=8) routing delays in the data sheet specifications section.

Timing Derating

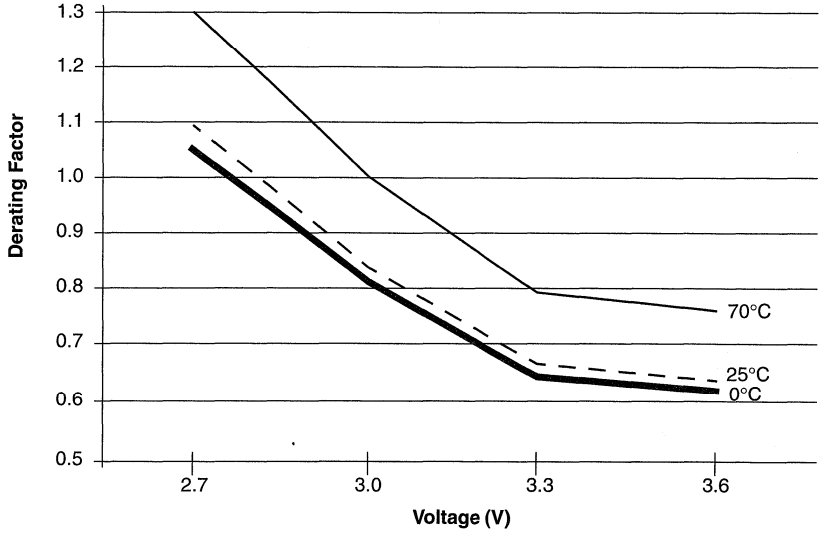
ACT 3 devices are manufactured in a CMOS process. Therefore, device performance varies according to temperature, voltage, and process variations. Minimum timing parameters reflect maximum operating voltage, minimum operating temperature, and best-case processing. Maximum timing parameters reflect minimum operating voltage, maximum operating temperature, and worst-case processing.

ACT 3 Timing Model*

Temperature and Voltage Derating Factors (normalized to Worst-Case Commercial, T_J = 3.0 V, 70°C)

	0	25	70
2.7	1.05	1.09	1.30
3.0	0.81	0.84	1.00
3.3	0.64	0.67	0.79
3.6	0.62	0.64	0.76

Junction Temperature and Voltage Derating Curves (normalized to Worst-Case Commercial, T_J = 3.0 V, 70°C)



Note: *This derating factor applies to all routing and propagation delays.*

A14V15A Timing Characteristics

(Worst-Case Commercial Conditions, V_{CC} = 3.0 V, T_J = 70°C)

Logic Module Propagation Delays[1]		'Std' Speed		
Parameter	**Description**	**Min.**	**Max.**	**Units**
t_{PD}	Internal Array Module		3.9	ns
t_{CO}	Sequential Clock to Q		3.9	ns
t_{CLR}	Asynchronous Clear to Q		3.9	ns
Predicted Routing Delays[2]				
t_{RD1}	FO=1 Routing Delay		1.7	ns
t_{RD2}	FO=2 Routing Delay		2.4	ns
t_{RD3}	FO=3 Routing Delay		2.8	ns
t_{RD4}	FO=4 Routing Delay		3.3	ns
t_{RD8}	FO=8 Routing Delay		5.5	ns
Logic Module Sequential Timing				
t_{SUD}	Flip-Flop Data Input Setup	0.8		ns
t_{HD}	Flip-Flop Data Input Hold	0.0		ns
t_{SUD}	Latch Data Input Setup	0.8		ns
t_{HD}	Latch Data Input Hold	0.0		ns
t_{WASYN}	Asynchronous Pulse Width	4.8		ns
t_{WCLKA}	Flip-Flop Clock Pulse Width	4.8		ns
t_A	Flip-Flop Clock Input Period	10.0		ns
f_{MAX}	Flip-Flop Clock Frequency		100	MHz

Notes:

1. *For dual-module macros, use $t_{PD} + t_{RD1} + t_{PDn}$, $t_{CO} + t_{RD1} + t_{PDn}$ or $t_{PD1} + t_{RD1} + t_{SUD}$, whichever is appropriate.*
2. *Routing delays are for typical designs across worst-case operating conditions. These parameters should be used for estimating device performance. Post-route timing analysis or simulation is required to determine actual worst-case performance. Post-route timing is based on actual routing delay measurements performed on the device prior to shipment.*

A14V15A Timing Characteristics (continued)

(Worst-Case Commercial Conditions)

I/O Module Input Propagation Delays		'Std' Speed		
Parameter	**Description**	**Min.**	**Max.**	**Units**
t_{INY}	Input Data Pad to Y		5.5	ns
t_{ICKY}	Input Reg IOCLK Pad to Y		9.2	ns
t_{OCKY}	Output Reg IOCLK Pad to Y		9.2	ns
t_{ICLRY}	Input Asynchronous Clear to Y		9.2	ns
t_{OCLRY}	Output Asynchronous Clear to Y		9.2	ns
Predicted Input Routing Delays[1]				
t_{IRD1}	FO=1 Routing Delay		1.7	ns
t_{IRD2}	FO=2 Routing Delay		2.4	ns
t_{IRD3}	FO=3 Routing Delay		2.8	ns
t_{IRD4}	FO=4 Routing Delay		3.3	ns
t_{IRD8}	FO=8 Routing Delay		5.5	ns
I/O Module Sequential Timing				
t_{INH}	Input F-F Data Hold (w.r.t. IOCLK Pad)	0.0		ns
t_{INSU}	Input F-F Data Setup (w.r.t. IOCLK Pad)	3.0		ns
t_{IDEH}	Input Data Enable Hold (w.r.t. IOCLK Pad)	0.0		ns
t_{IDESU}	Input Data Enable Setup (w.r.t. IOCLK Pad)	8.6		ns
t_{OUTH}	Output F-F Data Hold (w.r.t. IOCLK Pad)	1.0		ns
t_{OUTSU}	Output F-F Data Setup (w.r.t. IOCLK Pad)	1.0		ns
t_{ODEH}	Output Data Enable Hold (w.r.t. IOCLK Pad)	0.5		ns
t_{ODESU}	Output Data Enable Setup (w.r.t. IOCLK Pad)	2.0		ns

Note:

1. Routing delays are for typical designs across worst-case operating conditions. These parameters should be used for estimating device performance. Post-route timing analysis or simulation is required to determine actual worst-case performance. Post-route timing is based on actual routing delay measurements performed on the device prior to shipment.

A14V15A Timing Characteristics (continued)

(Worst-Case Commercial Conditions)

I/O Module – Output Timing[1]		'Std' Speed		
Parameter	**Description**	**Min.**	**Max.**	**Units**
t_{DHS}	Data to Pad, High Slew		9.8	ns
t_{DLS}	Data to Pad, Low Slew		15.7	ns
t_{ENZHS}	Enable to Pad, Z to H/L, Hi Slew		7.9	ns
t_{ENZLS}	Enable to Pad, Z to H/L, Lo Slew		14.4	ns
t_{ENHSZ}	Enable to Pad, H/L to Z, Hi Slew		13.0	ns
t_{ENLSZ}	Enable to Pad, H/L to Z, Lo Slew		13.0	ns
t_{CKHS}	IOCLK Pad to Pad H/L, Hi Slew		13.0	ns
t_{CKLS}	IOCLK Pad to Pad H/L, Lo Slew		19.7	ns
d_{TLHHS}	Delta Low to High, Hi Slew		0.03	ns/pF
d_{TLHLS}	Delta Low to High, Lo Slew		0.07	ns/pF
d_{THLHS}	Delta High to Low, Hi Slew		0.05	ns/pF
d_{THLLS}	Delta High to Low, Lo Slew		0.07	ns/pF

Note:

1. Delays based on 35pF loading.

A14V15A Timing Characteristics (continued)

(Worst-Case Commercial Conditions)

Dedicated (Hard-Wired) I/O Clock Network		'Std' Speed		
Parameter	**Description**	**Min.**	**Max.**	**Units**
t_{IOCKH}	Input Low to High (Pad to I/O Module Input)		3.5	ns
t_{IOPWH}	Minimum Pulse Width High	4.8		ns
t_{IOPWL}	Minimum Pulse Width Low	4.8		ns
t_{IOSAPW}	Minimum Asynchronous Pulse Width	4.8		ns
t_{IOCKSW}	Maximum Skew		0.4	ns
t_{IOP}	Minimum Period	10.0		ns
f_{IOMAX}	Maximum Frequency		100	MHz
Dedicated (Hard-Wired) Array Clock Network				
t_{HCKH}	Input Low to High (Pad to S-Module Input)		5.5	ns
t_{HCKL}	Input High to Low (Pad to S-Module Input)		5.5	ns
t_{HPWH}	Minimum Pulse Width High	4.8		ns
t_{HPWL}	Minimum Pulse Width Low	4.8		ns
t_{HCKSW}	Maximum Skew		0.3	ns
t_{HP}	Minimum Period	10.0		ns
f_{HMAX}	Maximum Frequency		100	MHz
Routed Array Clock Networks				
t_{RCKH}	Input Low to High (FO=64)		9.0	ns
t_{RCKL}	Input High to Low (FO=64)		9.0	ns
t_{RPWH}	Min. Pulse Width High (FO=64)	6.5		ns
t_{RPWL}	Min. Pulse Width Low (FO=64)	6.5		ns
t_{RCKSW}	Maximum Skew (FO=128)		1.0	ns
t_{RP}	Minimum Period (FO=64)	13.4		ns
f_{RMAX}	Maximum Frequency (FO=64)		75	MHz
Clock-to-Clock Skews				
$t_{IOHCKSW}$	I/O Clock to H-Clock Skew	0.0	3.0	ns
$t_{IOHCKSW}$	I/O Clock to R-Clock Skew	0.0	3.0	ns
t_{HRCKSW}	H-Clock to R-Clock Skew			
	(FO = 64)	0.0	1.0	ns
	(FO = 50% max.)	0.0	3.0	ns

Note:

1. *Delays based on 35pF loading.*

A14V25A Timing Characteristics

(Worst-Case Commercial Conditions, V_{CC} = 3.0 V, T_J = 70°C)

Logic Module Propagation Delays[1]		'Std' Speed		
Parameter	**Description**	**Min.**	**Max.**	**Units**
t_{PD}	Internal Array Module		3.9	ns
t_{CO}	Sequential Clock to Q		3.9	ns
t_{CLR}	Asynchronous Clear to Q		3.9	ns
Predicted Routing Delays[2]				
t_{RD1}	FO=1 Routing Delay		1.7	ns
t_{RD2}	FO=2 Routing Delay		2.4	ns
t_{RD3}	FO=3 Routing Delay		2.8	ns
t_{RD4}	FO=4 Routing Delay		3.3	ns
t_{RD8}	FO=8 Routing Delay		5.5	ns
Logic Module Sequential Timing				
t_{SUD}	Flip-Flop Data Input Setup	0.8		ns
t_{HD}	Flip-Flop Data Input Hold	0.0		ns
t_{SUD}	Latch Data Input Setup	0.8		ns
t_{HD}	Latch Data Input Hold	0.0		ns
t_{WASYN}	Asynchronous Pulse Width	4.8		ns
t_{WCLKA}	Flip-Flop Clock Pulse Width	4.8		ns
t_A	Flip-Flop Clock Input Period	10.0		ns
f_{MAX}	Flip-Flop Clock Frequency		100	MHz

Notes:

1. *For dual-module macros, use $t_{PD} + t_{RD1} + t_{PDm}$, $t_{CO} + t_{RD1} + t_{PDm}$ or $t_{PD1} + t_{RD1} + t_{SUD}$, whichever is appropriate.*
2. *Routing delays are for typical designs across worst-case operating conditions. These parameters should be used for estimating device performance. Post-route timing analysis or simulation is required to determine actual worst-case performance. Post-route timing is based on actual routing delay measurements performed on the device prior to shipment.*

A14V25A Timing Characteristics (continued)

(Worst-Case Commercial Conditions)

I/O Module Input Propagation Delays		'Std' Speed		
Parameter	**Description**	**Min.**	**Max.**	**Units**
t_{INY}	Input Data Pad to Y		5.5	ns
t_{ICKY}	Input Reg IOCLK Pad to Y		9.2	ns
t_{OCKY}	Output Reg IOCLK Pad to Y		9.2	ns
t_{ICLRY}	Input Asynchronous Clear to Y		9.2	ns
t_{OCLRY}	Output Asynchronous Clear to Y		9.2	ns
Predicted Input Routing Delays[1]				
t_{IRD1}	FO=1 Routing Delay		1.7	ns
t_{IRD2}	FO=2 Routing Delay		2.4	ns
t_{IRD3}	FO=3 Routing Delay		2.8	ns
t_{IRD4}	FO=4 Routing Delay		3.3	ns
t_{IRD8}	FO=8 Routing Delay		5.5	ns
I/O Module Sequential Timing				
t_{INH}	Input F-F Data Hold (w.r.t. IOCLK Pad)	0.0		ns
t_{INSU}	Input F-F Data Setup (w.r.t. IOCLK Pad)	3.0		ns
t_{IDEH}	Input Data Enable Hold (w.r.t. IOCLK Pad)	0.0		ns
t_{IDESU}	Input Data Enable Setup (w.r.t. IOCLK Pad)	8.6		ns
t_{OUTH}	Output F-F Data Hold (w.r.t. IOCLK Pad)	1.0		ns
t_{OUTSU}	Output F-F Data Setup (w.r.t. IOCLK Pad)	1.0		ns
t_{ODEH}	Output Data Enable Hold (w.r.t. IOCLK Pad)	0.5		ns
t_{ODESU}	Output Data Enable Setup (w.r.t. IOCLK Pad)	2.0		ns

Note:

1. *Routing delays are for typical designs across worst-case operating conditions. These parameters should be used for estimating device performance. Post-route timing analysis or simulation is required to determine actual worst-case performance. Post-route timing is based on actual routing delay measurements performed on the device prior to shipment.*

A14V25A Timing Characteristics (continued)

(Worst-Case Commercial Conditions)

I/O Module – Output Timing[1]		'Std' Speed		
Parameter	**Description**	**Min.**	**Max.**	**Units**
t_{DHS}	Data to Pad, High Slew		9.8	ns
t_{DLS}	Data to Pad, Low Slew		15.7	ns
t_{ENZHS}	Enable to Pad, Z to H/L, Hi Slew		7.9	ns
t_{ENZLS}	Enable to Pad, Z to H/L, Lo Slew		14.4	ns
t_{ENHSZ}	Enable to Pad, H/L to Z, Hi Slew		13.0	ns
t_{ENLSZ}	Enable to Pad, H/L to Z, Lo Slew		13.0	ns
t_{CKHS}	IOCLK Pad to Pad H/L, Hi Slew		13.0	ns
t_{CKLS}	IOCLK Pad to Pad H/L, Lo Slew		19.7	ns
d_{TLHHS}	Delta Low to High, Hi Slew		0.03	ns/pF
d_{TLHLS}	Delta Low to High, Lo Slew		0.07	ns/pF
d_{THLHS}	Delta High to Low, Hi Slew		0.05	ns/pF
d_{THLLS}	Delta High to Low, Lo Slew		0.07	ns/pF

Note:

1. Delays based on 35pF loading.

A14V25A Timing Characteristics (continued)

(Worst-Case Commercial Conditions)

Dedicated (Hard-Wired) I/O Clock Network		'Std' Speed		
Parameter	**Description**	**Min.**	**Max.**	**Units**
t_{IOCKH}	Input Low to High (Pad to I/O Module Input)		3.5	ns
t_{IOPWH}	Minimum Pulse Width High	4.8		ns
t_{IOPWL}	Minimum Pulse Width Low	4.8		ns
t_{IOSAPW}	Minimum Asynchronous Pulse Width	4.8		ns
t_{IOCKSW}	Maximum Skew		0.4	ns
t_{IOP}	Minimum Period	10.0		ns
f_{IOMAX}	Maximum Frequency		100	MHz
Dedicated (Hard-Wired) Array Clock Network				
t_{HCKH}	Input Low to High (Pad to S-Module Input)		5.5	ns
t_{HCKL}	Input High to Low (Pad to S-Module Input)		5.5	ns
t_{HPWH}	Minimum Pulse Width High	4.8		ns
t_{HPWL}	Minimum Pulse Width Low	4.8		ns
t_{HCKSW}	Maximum Skew		0.3	ns
t_{HP}	Minimum Period	10.0		ns
f_{HMAX}	Maximum Frequency		100	MHz
Routed Array Clock Networks				
t_{RCKH}	Input Low to High (FO=64)		9.0	ns
t_{RCKL}	Input High to Low (FO=64)		9.0	ns
t_{RPWH}	Min. Pulse Width High (FO=64)	6.5		ns
t_{RPWL}	Min. Pulse Width Low (FO=64)	6.5		ns
t_{RCKSW}	Maximum Skew (FO=128)		1.0	ns
t_{RP}	Minimum Period (FO=64)	13.4		ns
f_{RMAX}	Maximum Frequency (FO=64)		75	MHz
Clock-to-Clock Skews				
$t_{IOHCKSW}$	I/O Clock to H-Clock Skew	0.0	3.0	ns
$t_{IOHCKSW}$	I/O Clock to R-Clock Skew	0.0	3.0	ns
t_{HRCKSW}	H-Clock to R-Clock Skew			
	(FO = 64)	0.0	1.0	ns
	(FO = 50% max.)	0.0	3.0	ns

Note:

1. Delays based on 35pF loading.

A14V40A Timing Characteristics

(Worst-Case Commercial Conditions, V_{CC} = 3.0 V, T_J = 70°C)

Logic Module Propagation Delays[1]		'Std' Speed		
Parameter	**Description**	**Min.**	**Max.**	**Units**
t_{PD}	Internal Array Module		3.9	ns
t_{CO}	Sequential Clock to Q		3.9	ns
t_{CLR}	Asynchronous Clear to Q		3.9	ns
Predicted Routing Delays[2]				
t_{RD1}	FO=1 Routing Delay		1.7	ns
t_{RD2}	FO=2 Routing Delay		2.4	ns
t_{RD3}	FO=3 Routing Delay		2.8	ns
t_{RD4}	FO=4 Routing Delay		3.3	ns
t_{RD8}	FO=8 Routing Delay		5.5	ns
Logic Module Sequential Timing				
t_{SUD}	Flip-Flop Data Input Setup	0.8		ns
t_{HD}	Flip-Flop Data Input Hold	0.0		ns
t_{SUD}	Latch Data Input Setup	0.8		ns
t_{HD}	Latch Data Input Hold	0.0		ns
t_{WASYN}	Asynchronous Pulse Width	4.8		ns
t_{WCLKA}	Flip-Flop Clock Pulse Width	4.8		ns
t_A	Flip-Flop Clock Input Period	10.0		ns
f_{MAX}	Flip-Flop Clock Frequency		100	MHz

Notes:

1. *For dual-module macros, use $t_{PD} + t_{RD1} + t_{PDn}$, $t_{CO} + t_{RD1} + t_{PDn}$ or $t_{PD1} + t_{RD1} + t_{SUD}$, whichever is appropriate.*
2. *Routing delays are for typical designs across worst-case operating conditions. These parameters should be used for estimating device performance. Post-route timing analysis or simulation is required to determine actual worst-case performance. Post-route timing is based on actual routing delay measurements performed on the device prior to shipment.*

A14V40A Timing Characteristics (continued)

(Worst-Case Commercial Conditions)

I/O Module Input Propagation Delays		'Std' Speed		
Parameter	**Description**	**Min.**	**Max.**	**Units**
t_{INY}	Input Data Pad to Y		5.5	ns
t_{ICKY}	Input Reg IOCLK Pad to Y		9.2	ns
t_{OCKY}	Output Reg IOCLK Pad to Y		9.2	ns
t_{ICLRY}	Input Asynchronous Clear to Y		9.2	ns
t_{OCLRY}	Output Asynchronous Clear to Y		9.2	ns
Predicted Input Routing Delays[1]				
t_{IRD1}	FO=1 Routing Delay		1.7	ns
t_{IRD2}	FO=2 Routing Delay		2.4	ns
t_{IRD3}	FO=3 Routing Delay		2.8	ns
t_{IRD4}	FO=4 Routing Delay		3.3	ns
t_{IRD8}	FO=8 Routing Delay		5.5	ns
I/O Module Sequential Timing				
t_{INH}	Input F-F Data Hold (w.r.t. IOCLK Pad)	0.0		ns
t_{INSU}	Input F-F Data Setup (w.r.t. IOCLK Pad)	2.3		ns
t_{IDEH}	Input Data Enable Hold (w.r.t. IOCLK Pad)	0.0		ns
t_{IDESU}	Input Data Enable Setup (w.r.t. IOCLK Pad)	8.6		ns
t_{OUTH}	Output F-F Data Hold (w.r.t. IOCLK Pad)	1.0		ns
t_{OUTSU}	Output F-F Data Setup (w.r.t. IOCLK Pad)	1.0		ns
t_{ODEH}	Output Data Enable Hold (w.r.t. IOCLK Pad)	0.5		ns
t_{ODESU}	Output Data Enable Setup (w.r.t. IOCLK Pad)	2.0		ns

Note:

1. *Routing delays are for typical designs across worst-case operating conditions. These parameters should be used for estimating device performance. Post-route timing analysis or simulation is required to determine actual worst-case performance. Post-route timing is based on actual routing delay measurements performed on the device prior to shipment.*

A14V40A Timing Characteristics (continued)

(Worst-Case Commercial Conditions)

I/O Module – Output Timing[1]		'Std' Speed		
Parameter	**Description**	**Min.**	**Max.**	**Units**
t_{DHS}	Data to Pad, High Slew		9.8	ns
t_{DLS}	Data to Pad, Low Slew		15.7	ns
t_{ENZHS}	Enable to Pad, Z to H/L, Hi Slew		7.9	ns
t_{ENZLS}	Enable to Pad, Z to H/L, Lo Slew		14.4	ns
t_{ENHSZ}	Enable to Pad, H/L to Z, Hi Slew		14.4	ns
t_{ENLSZ}	Enable to Pad, H/L to Z, Lo Slew		14.4	ns
t_{CKHS}	IOCLK Pad to Pad H/L, Hi Slew		14.4	ns
t_{CKLS}	IOCLK Pad to Pad H/L, Lo Slew		19.7	ns
d_{TLHHS}	Delta Low to High, Hi Slew		0.03	ns/pF
d_{TLHLS}	Delta Low to High, Lo Slew		0.07	ns/pF
d_{THLHS}	Delta High to Low, Hi Slew		0.05	ns/pF
d_{THLLS}	Delta High to Low, Lo Slew		0.07	ns/pF

Note:

1. Delays based on 35pF loading.

A14V40A Timing Characteristics (continued)

(Worst-Case Commercial Conditions)

Dedicated (Hard-Wired) I/O Clock Network		'Std' Speed		
Parameter	**Description**	**Min.**	**Max.**	**Units**
t_{IOCKH}	Input Low to High (Pad to I/O Module Input)		3.5	ns
t_{IOPWH}	Minimum Pulse Width High	4.8		ns
t_{IOPWL}	Minimum Pulse Width Low	4.8		ns
t_{IOSAPW}	Minimum Asynchronous Pulse Width	4.8		ns
t_{IOCKSW}	Maximum Skew		0.4	ns
t_{IOP}	Minimum Period	10.0		ns
f_{IOMAX}	Maximum Frequency		100	MHz
Dedicated (Hard-Wired) Array Clock Network				
t_{HCKH}	Input Low to High (Pad to S-Module Input)		5.5	ns
t_{HCKL}	Input High to Low (Pad to S-Module Input)		5.5	ns
t_{HPWH}	Minimum Pulse Width High	4.8		ns
t_{HPWL}	Minimum Pulse Width Low	4.8		ns
t_{HCKSW}	Maximum Skew		0.3	ns
t_{HP}	Minimum Period	10.0		ns
f_{HMAX}	Maximum Frequency		100	MHz
Routed Array Clock Networks				
t_{RCKH}	Input Low to High (FO=64)		9.0	ns
t_{RCKL}	Input High to Low (FO=64)		9.0	ns
t_{RPWH}	Min. Pulse Width High (FO=64)	6.5		ns
t_{RPWL}	Min. Pulse Width Low (FO=64)	6.5		ns
t_{RCKSW}	Maximum Skew (FO=128)		1.0	ns
t_{RP}	Minimum Period (FO=64)	13.4		ns
f_{RMAX}	Maximum Frequency (FO=64)		75	MHz
Clock-to-Clock Skews				
$t_{IOHCKSW}$	I/O Clock to H-Clock Skew	0.0	3.0	ns
$t_{IOHCKSW}$	I/O Clock to R-Clock Skew	0.0	3.0	ns
t_{HRCKSW}	H-Clock to R-Clock Skew			
	(FO = 64)	0.0	1.0	ns
	(FO = 50% max.)	0.0	3.0	ns

Note:

1. Delays based on 35pF loading.

A14V60A Timing Characteristics

(Worst-Case Commercial Conditions, V_{CC} = 3.0 V, T_J = 70°C)

Logic Module Propagation Delays[1]		'Std' Speed		
Parameter	**Description**	**Min.**	**Max.**	**Units**
t_{PD}	Internal Array Module		3.9	ns
t_{CO}	Sequential Clock to Q		3.9	ns
t_{CLR}	Asynchronous Clear to Q		3.9	ns
Predicted Routing Delays[2]				
t_{RD1}	FO=1 Routing Delay		1.7	ns
t_{RD2}	FO=2 Routing Delay		2.4	ns
t_{RD3}	FO=3 Routing Delay		2.8	ns
t_{RD4}	FO=4 Routing Delay		3.3	ns
t_{RD8}	FO=8 Routing Delay		5.5	ns
Logic Module Sequential Timing				
t_{SUD}	Flip-Flop Data Input Setup	0.8		ns
t_{HD}	Flip-Flop Data Input Hold	0.0		ns
t_{SUD}	Latch Data Input Setup	0.8		ns
t_{HD}	Latch Data Input Hold	0.0		ns
t_{WASYN}	Asynchronous Pulse Width	6.5		ns
t_{WCLKA}	Flip-Flop Clock Pulse Width	6.5		ns
t_A	Flip-Flop Clock Input Period	13.4		ns
f_{MAX}	Flip-Flop Clock Frequency		75	MHz

Notes:

1. *For dual-module macros, use $t_{PD} + t_{RD1} + t_{PDn}$, $t_{CO} + t_{RD1} + t_{PDn}$ or $t_{PD1} + t_{RD1} + t_{SUD}$, whichever is appropriate.*
2. *Routing delays are for typical designs across worst-case operating conditions. These parameters should be used for estimating device performance. Post-route timing analysis or simulation is required to determine actual worst-case performance. Post-route timing is based on actual routing delay measurements performed on the device prior to shipment.*

A14V60A Timing Characteristics (continued)

(Worst-Case Commercial Conditions)

I/O Module Input Propagation Delays		'Std' Speed		
Parameter	**Description**	**Min.**	**Max.**	**Units**
t_{INY}	Input Data Pad to Y		5.5	ns
t_{ICKY}	Input Reg IOCLK Pad to Y		9.2	ns
t_{OCKY}	Output Reg IOCLK Pad to Y		9.2	ns
t_{ICLRY}	Input Asynchronous Clear to Y		9.2	ns
t_{OCLRY}	Output Asynchronous Clear to Y		9.2	ns
Predicted Input Routing Delays[1]				
t_{IRD1}	FO=1 Routing Delay		1.7	ns
t_{IRD2}	FO=2 Routing Delay		2.4	ns
t_{IRD3}	FO=3 Routing Delay		2.8	ns
t_{IRD4}	FO=4 Routing Delay		3.3	ns
t_{IRD8}	FO=8 Routing Delay		5.5	ns
I/O Module Sequential Timing				
t_{INH}	Input F-F Data Hold (w.r.t. IOCLK Pad)	0.0		ns
t_{INSU}	Input F-F Data Setup (w.r.t. IOCLK Pad)	2.0		ns
t_{IDEH}	Input Data Enable Hold (w.r.t. IOCLK Pad)	0.0		ns
t_{IDESU}	Input Data Enable Setup (w.r.t. IOCLK Pad)	8.6		ns
t_{OUTH}	Output F-F Data Hold (w.r.t. IOCLK Pad)	1.0		ns
t_{OUTSU}	Output F-F Data Setup (w.r.t. IOCLK Pad)	1.0		ns
t_{ODEH}	Output Data Enable Hold (w.r.t. IOCLK Pad)	0.5		ns
t_{ODESU}	Output Data Enable Setup (w.r.t. IOCLK Pad)	2.0		ns

Note:

1. *Routing delays are for typical designs across worst-case operating conditions. These parameters should be used for estimating device performance. Post-route timing analysis or simulation is required to determine actual worst-case performance. Post-route timing is based on actual routing delay measurements performed on the device prior to shipment.*

A14V60A Timing Characteristics (continued)

(Worst-Case Commercial Conditions)

I/O Module – Output Timing[1]		'Std' Speed		
Parameter	**Description**	**Min.**	**Max.**	**Units**
t_{DHS}	Data to Pad, High Slew		9.8	ns
t_{DLS}	Data to Pad, Low Slew		15.7	ns
t_{ENZHS}	Enable to Pad, Z to H/L, Hi Slew		7.9	ns
t_{ENZLS}	Enable to Pad, Z to H/L, Lo Slew		14.4	ns
t_{ENHSZ}	Enable to Pad, H/L to Z, Hi Slew		15.2	ns
t_{ENLSZ}	Enable to Pad, H/L to Z, Lo Slew		14.4	ns
t_{CKHS}	IOCLK Pad to Pad H/L, Hi Slew		15.1	ns
t_{CKLS}	IOCLK Pad to Pad H/L, Lo Slew		22.3	ns
d_{TLHHS}	Delta Low to High, Hi Slew		0.03	ns/pF
d_{TLHLS}	Delta Low to High, Lo Slew		0.07	ns/pF
d_{THLHS}	Delta High to Low, Hi Slew		0.05	ns/pF
d_{THLLS}	Delta High to Low, Lo Slew		0.07	ns/pF

Note:

1. Delays based on 35pF loading.

A14V60A Timing Characteristics (continued)

(Worst-Case Commercial Conditions)

Dedicated (Hard-Wired) I/O Clock Network		'Std' Speed		
Parameter	**Description**	**Min.**	**Max.**	**Units**
t_{IOCKH}	Input Low to High (Pad to I/O Module Input)		4.5	ns
t_{IOPWH}	Minimum Pulse Width High	6.5		ns
t_{IOPWL}	Minimum Pulse Width Low	6.5		ns
t_{IOSAPW}	Minimum Asynchronous Pulse Width	6.5		ns
t_{IOCKSW}	Maximum Skew		0.6	ns
t_{IOP}	Minimum Period	13.4		ns
f_{IOMAX}	Maximum Frequency		75	MHz
Dedicated (Hard-Wired) Array Clock Network				
t_{HCKH}	Input Low to High (Pad to S-Module Input)		7.0	ns
t_{HCKL}	Input High to Low (Pad to S-Module Input)		7.0	ns
t_{HPWH}	Minimum Pulse Width High	6.5		ns
t_{HPWL}	Minimum Pulse Width Low	6.5		ns
t_{HCKSW}	Maximum Skew		0.6	ns
t_{HP}	Minimum Period	13.4		ns
f_{HMAX}	Maximum Frequency		75	MHz
Routed Array Clock Networks				
t_{RCKH}	Input Low to High (FO=256)		11.8	ns
t_{RCKL}	Input High to Low (FO=256)		11.8	ns
t_{RPWH}	Min. Pulse Width High (FO=256)	8.2		ns
t_{RPWL}	Min. Pulse Width Low (FO=256)	8.2		ns
t_{RCKSW}	Maximum Skew (FO=128)		1.8	ns
t_{RP}	Minimum Period (FO=256)	16.7		ns
f_{RMAX}	Maximum Frequency (FO=256)		60	MHz
Clock-to-Clock Skews				
$t_{IOHCKSW}$	I/O Clock to H-Clock Skew	0.0	3.0	ns
$t_{IOHCKSW}$	I/O Clock to R-Clock Skew	0.0	5.0	ns
t_{HRCKSW}	H-Clock to R-Clock Skew			
	(FO = 64)	0.0	1.0	ns
	(FO = 50% max.)	0.0	3.0	ns

Note:

1. Delays based on 35pF loading.

A14V100A Timing Characteristics

(Worst-Case Commercial Conditions, V_{CC} = 3.0 V, T_J = 70°C)

Logic Module Propagation Delays[1]		'Std' Speed		
Parameter	**Description**	**Min.**	**Max.**	**Units**
t_{PD}	Internal Array Module		3.9	ns
t_{CO}	Sequential Clock to Q		3.9	ns
t_{CLR}	Asynchronous Clear to Q		3.9	ns
Predicted Routing Delays[2]				
t_{RD1}	FO=1 Routing Delay		1.7	ns
t_{RD2}	FO=2 Routing Delay		2.4	ns
t_{RD3}	FO=3 Routing Delay		2.8	ns
t_{RD4}	FO=4 Routing Delay		3.3	ns
t_{RD8}	FO=8 Routing Delay		5.5	ns
Logic Module Sequential Timing				
t_{SUD}	Flip-Flop Data Input Setup	0.8		ns
t_{HD}	Flip-Flop Data Input Hold	0.5		ns
t_{SUD}	Latch Data Input Setup	0.8		ns
t_{HD}	Latch Data Input Hold	0.5		ns
t_{WASYN}	Asynchronous Pulse Width	6.5		ns
t_{WCLKA}	Flip-Flop Clock Pulse Width	6.5		ns
t_A	Flip-Flop Clock Input Period	13.4		ns
f_{MAX}	Flip-Flop Clock Frequency		75	MHz

Notes:

1. *For dual-module macros, use $t_{PD} + t_{RD1} + t_{PDn}$, $t_{CO} + t_{RD1} + t_{PDn}$ or $t_{PD1} + t_{RD1} + t_{SUD}$, whichever is appropriate.*
2. *Routing delays are for typical designs across worst-case operating conditions. These parameters should be used for estimating device performance. Post-route timing analysis or simulation is required to determine actual worst-case performance. Post-route timing is based on actual routing delay measurements performed on the device prior to shipment.*

A14V100A Timing Characteristics (continued)

(Worst-Case Commercial Conditions)

I/O Module Input Propagation Delays		'Std' Speed		
Parameter	**Description**	**Min.**	**Max.**	**Units**
t_{INY}	Input Data Pad to Y		5.5	ns
t_{ICKY}	Input Reg IOCLK Pad to Y		9.2	ns
t_{OCKY}	Output Reg IOCLK Pad to Y		9.2	ns
t_{ICLRY}	Input Asynchronous Clear to Y		9.2	ns
t_{OCLRY}	Output Asynchronous Clear to Y		9.2	ns
Predicted Input Routing Delays[1]				
t_{IRD1}	FO=1 Routing Delay		1.7	ns
t_{IRD2}	FO=2 Routing Delay		2.4	ns
t_{IRD3}	FO=3 Routing Delay		2.8	ns
t_{IRD4}	FO=4 Routing Delay		3.3	ns
t_{IRD8}	FO=8 Routing Delay		5.5	ns
I/O Module Sequential Timing				
t_{INH}	Input F-F Data Hold (w.r.t. IOCLK Pad)	0.0		ns
t_{INSU}	Input F-F Data Setup (w.r.t. IOCLK Pad)	1.8		ns
t_{IDEH}	Input Data Enable Hold (w.r.t. IOCLK Pad)	0.0		ns
t_{IDESU}	Input Data Enable Setup (w.r.t. IOCLK Pad)	8.6		ns
t_{OUTH}	Output F-F Data Hold (w.r.t. IOCLK Pad)	1.0		ns
t_{OUTSU}	Output F-F Data Setup (w.r.t. IOCLK Pad)	1.0		ns
t_{ODEH}	Output Data Enable Hold (w.r.t. IOCLK Pad)	0.5		ns
t_{ODESU}	Output Data Enable Setup (w.r.t. IOCLK Pad)	2.0		ns

Note:

1. *Routing delays are for typical designs across worst-case operating conditions. These parameters should be used for estimating device performance. Post-route timing analysis or simulation is required to determine actual worst-case performance. Post-route timing is based on actual routing delay measurements performed on the device prior to shipment.*

A14V100A Timing Characteristics (continued)

(Worst-Case Commercial Conditions)

I/O Module – Output Timing[1]		'Std' Speed		
Parameter	**Description**	**Min.**	**Max.**	**Units**
t_{DHS}	Data to Pad, High Slew		9.8	ns
t_{DLS}	Data to Pad, Low Slew		15.7	ns
t_{ENZHS}	Enable to Pad, Z to H/L, Hi Slew		7.9	ns
t_{ENZLS}	Enable to Pad, Z to H/L, Lo Slew		14.4	ns
t_{ENHSZ}	Enable to Pad, H/L to Z, Hi Slew		15.7	ns
t_{ENLSZ}	Enable to Pad, H/L to Z, Lo Slew		14.4	ns
t_{CKHS}	IOCLK Pad to Pad H/L, Hi Slew		15.7	ns
t_{CKLS}	IOCLK Pad to Pad H/L, Lo Slew		22.3	ns
d_{TLHHS}	Delta Low to High, Hi Slew		0.03	ns/pF
d_{TLHLS}	Delta Low to High, Lo Slew		0.07	ns/pF
d_{THLHS}	Delta High to Low, Hi Slew		0.05	ns/pF
d_{THLLS}	Delta High to Low, Lo Slew		0.07	ns/pF

Note:

1. Delays based on 35pF loading.

A14V100A Timing Characteristics (continued)

(Worst-Case Commercial Conditions)

Dedicated (Hard-Wired) I/O Clock Network		'Std' Speed		
Parameter	**Description**	**Min.**	**Max.**	**Units**
t_{IOCKH}	Input Low to High (Pad to I/O Module Input)		4.5	ns
t_{IOPWH}	Minimum Pulse Width High	6.5		ns
t_{IOPWL}	Minimum Pulse Width Low	6.5		ns
t_{IOSAPW}	Minimum Asynchronous Pulse Width	6.5		ns
t_{IOCKSW}	Maximum Skew		0.6	ns
t_{IOP}	Minimum Period	13.4		ns
f_{IOMAX}	Maximum Frequency		75	MHz
Dedicated (Hard-Wired) Array Clock Network				
t_{HCKH}	Input Low to High (Pad to S-Module Input)		7.0	ns
t_{HCKL}	Input High to Low (Pad to S-Module Input)		7.0	ns
t_{HPWH}	Minimum Pulse Width High	6.5		ns
t_{HPWL}	Minimum Pulse Width Low	6.5		ns
t_{HCKSW}	Maximum Skew		0.6	ns
t_{HP}	Minimum Period	13.4		ns
f_{HMAX}	Maximum Frequency		75	MHz
Routed Array Clock Networks				
t_{RCKH}	Input Low to High (FO=256)		11.8	ns
t_{RCKL}	Input High to Low (FO=256)		11.8	ns
t_{RPWH}	Min. Pulse Width High (FO=256)	8.2		ns
t_{RPWL}	Min. Pulse Width Low (FO=256)	8.2		ns
t_{RCKSW}	Maximum Skew (FO=128)		1.8	ns
t_{RP}	Minimum Period (FO=256)	16.7		ns
f_{RMAX}	Maximum Frequency (FO=256)		60	MHz
Clock-to-Clock Skews				
$t_{IOHCKSW}$	I/O Clock to H-Clock Skew	0.0	3.0	ns
$t_{IOHCKSW}$	I/O Clock to R-Clock Skew	0.0	5.0	ns
t_{HRCKSW}	H-Clock to R-Clock Skew (FO = 64) (FO = 50% max.)	 0.0 0.0	 1.0 3.0	ns

Note:

1. Delays based on 35pF loading.

Macro Library

Hard Macros—Combinatorial

Function	Macro	Description	Modules S	Modules C
ACT 3 Combinatorial Logic Module	CM8	Combinational Module (Full ACT 3 Logic Module)		1
ACT 3 Sequential Logic Module	DFM8A	4-bit D-Type Flip-Flop with Multiplexed Data, active low Clear, and active high clock	1	
	DFM8B	4-bit D-Type Flip-Flop with Multiplexed Data, active low Clear, and active low clock	1	
Adder	FA1A	1-bit adder, carry in and carry out active low, A-input active low		2
	FA1B	1-bit adder, carry in and carry out active low		2
	FA2A	2-bit adder, carry in and carry out active low, A0 and A1 inputs active low		2
	HA1	Half-Adder		2
	HA1A	Half-Adder with active low A-input		2
	HA1B	Half-Adder with active low carry out and sum		2
	HA1C	Half-Adder with active low carry out		2
AND	AND2	2-input AND		1
	AND2A	2-input AND with active low A-input		1
	AND2B	2-input AND with active low inputs		1
	AND3	3-input AND		1
	AND3A	3-input AND with active low A-input		1
	AND3B	3-input AND with active low A- and B-inputs		1
	AND3C	3-input AND with active low inputs		1
	AND4	4-input AND		1
	AND4A	4-input AND with active low A-input		1
	AND4B	4-input AND with active low A- and B-inputs		1
	AND4C	4-input AND with active low A-, B-, and C-inputs		1
	AND4D	4-input AND with active low inputs		2
	AND5B	5-input AND with active low A- and B-inputs		1
AND-OR	AO1	3-input AND-OR		1
	AO10	5-input AND-OR-AND		1
	AO11	3-input AND-OR		1
	AO1A	3-input AND-OR with active low A-input		1
	AO1B	3-input AND-OR with active low C-input		1
	AO1C	3-input AND-OR with active low A- and C-inputs		1
	AO1D	3-input AND-OR with active low A- and B-inputs		1
	AO1E	3-input AND-OR with active low inputs		1
	AO2	4-input AND-OR		1
	AO2A	4-input AND-OR with active low A-input		1
	AO2B	4-input AND-OR with active low A- and B-inputs		1
	AO2C	4-input AND-OR with active low A- and C-inputs		1
	AO2D	4-input AND-OR with active low A-, B-, and C-inputs		1
	AO2E	4-input AND-OR with active low inputs		1
	AO3	4-input AND-OR		1
	AO3A	4-input AND-OR		1
	AO3B	4-input AND-OR		1
	AO3C	4-input AND-OR		1
	AO4A	4-input AND-OR		1
	AO5A	4-input AND-OR		1
	AO6	2-wide 4-input AND-OR		1

Hard Macros—Combinatorial (Continued)

Function	Macro	Description	Modules S	Modules C
AND-OR	AO6A	2-wide 4-input AND-OR with active low D-input		1
	AO7	5-input AND-OR		1
	AO8	5-input AND-OR with active low C- and D-inputs		1
	AO9	5-input AND-OR		1
	AOI1	3-input AND-OR-INVERT		1
	AOI1A	3-input AND-OR-INVERT with active low A-input		1
	AOI1B	3-input AND-OR-INVERT with active low C-input		1
	AOI1C	3-input AND-OR-INVERT with active low A- and B-inputs		1
	AOI1D	3-input AND-OR-INVERT with active low inputs		1
	AOI2A	4-input AND-OR-INVERT with active low A-input		1
	AOI2B	4-input AND-OR-INVERT with active low A- and C-inputs		1
	AOI3A	4-input AND-OR-INVERT with active low inputs		1
	AOI4	2-wide 4-input AND-OR-INVERT		2
	AOI4A	2-wide 4-input AND-OR-INVERT with active low C-input		1
AND-XOR	AX1	3-input AND-XOR with active low A-input		1
	AX1A	3-input AND-XOR-INVERT with active low A-input		2
	AX1B	3-input AND-XOR with active low A- and B-inputs		1
	AX1C	3-input AND-XOR		1
Buffer	BUF	Buffer, with active high input and output		1
	BUFA	Buffer, with active low input and output		1
Clock Net	CLKINT	Clock Net Interface	0	0
	GAND2	2-input AND Clock Net		1
	GMX4	4-to-1 Multiplexor Clock Net		1
	GNAND2	2-input NAND Clock Net		1
	GNOR2	2-input NOR Clock Net		1
	GOR2	2-input OR Clock Net		1
	GXOR2	2-input Exclusive OR Clock Net		1
Inverter	INV	Inverter with active low output		1
	INVA	Inverter with active low input		1
Majority	MAJ3	3-input complex AND-OR		1
MUX	MX2	2-to-1 Multiplexor		1
	MX2A	2-to-1 Multiplexor with active low A-input		1
	MX2B	2-to-1 Multiplexor with active low B-input		1
MUX	MX2C	2-to-1 Multiplexor with active low output		1
	MX4	4-to-1 Multiplexor		1
	MXC1	Boolean		2
	MXT	Boolean		2
NAND	NAND2	2-input NAND		1
	NAND2A	2-input NAND with active low A-input		1
	NAND2B	2-input NAND with active low inputs		1
	NAND3	3-input NAND		1
	NAND3A	3-input NAND with active low A-input		1
	NAND3B	3-input NAND with active low A- and B-inputs		1
	NAND3C	3-input NAND with active low inputs		1
	NAND4	4-input NAND		2
	NAND4A	4-input NAND with active low A-input		1
	NAND4B	4-input NAND with active low A- and B-inputs		1
	NAND4C	4-input NAND with active low A-, B-, and C-inputs		1
	NAND4D	4-input NAND with active low inputs		1

Hard Macros—Combinatorial (Continued)

Function	Macro	Description	Modules S	Modules C
NAND	NAND5C	5-input NAND with active low A-, B-, and C-inputs		1
NOR	NOR2	2-input NOR		1
NOR	NOR2A	2-input NOR with active low A-input		1
	NOR2B	2-input NOR with active low inputs		1
	NOR3	3-input NOR		1
	NOR3A	3-input NOR with active low A-input		1
	NOR3B	3-input NOR with active low A- and B-inputs		1
	NOR3C	3-input NOR with active low inputs		1
	NOR4	4-input NOR		2
	NOR4A	4-input NOR with active low A-input		1
	NOR4B	4-input NOR with active low A- and B-inputs		1
	NOR4C	4-input NOR with active low A-, B-, and C-inputs		1
	NOR4D	4-input NOR with active low inputs		1
	NOR5C	5-input NOR with active low A-, B-, and C-inputs		1
OR	OR2	2-input OR		1
	OR2A	2-input OR with active low A-input		1
	OR2B	2-input OR with active low inputs		1
	OR3	3-input OR		1
	OR3A	3-input OR with active low A-input		1
	OR3B	3-input OR with active low A- and B-inputs		1
	OR3C	3-input OR with active low inputs		1
	OR4	4-input OR		1
	OR4A	4-input OR with active low A-input		1
	OR4B	4-input OR with active low A- and B-input		1
	OR4C	4-input OR with active low A-, B-, and C-inputs		1
	OR4D	4-input OR with active low inputs		2
	OR5B	5-input OR with active low A- and B-inputs		1
OR-AND	OA1	3-input OR-AND		1
	OA1A	3-input OR-AND with active low A-input		1
	OA1B	3-input OR-AND with active low C-input		1
	OA1C	3-input OR-AND with active low A- and C-inputs		1
	OA2	2-wide 4-input OR-AND		1
	OA2A	2 wide 4-input OR-AND with active low A-input		1
	OA3	4-input OR-AND		1
	OA3A	4-input OR-AND with active low C-input		1
	OA3B	4-input OR-AND with active low A- and C-inputs		1
	OA4	4-input OR-AND		1
	OA4A	4-input OR-AND with active low C-input		1
	OA5	4-input complex OR-AND		1
	OAI1	3-input OR-AND-INVERT		1
	OAI2A	4-input OR-AND-INVERT with active low D-input		1
	OAI3	4-input OR-AND-INVERT		1
	OAI3A	4-input OR-AND-INVERT with active low C- and D-inputs		1
XNOR	XNOR	2-input XNOR		1
XNOR-AND	XA1A	3-input XNOR-AND		1
XNOR-OR	XO1A	3-input XNOR-OR		1
XOR	XOR	2-input XOR		1
XOR-AND	XA1	3-input XOR-AND		1
XOR-OR	XO1	3-input XOR-OR		1

Hard Macros—Sequential

Function	Macro	Description	Modules S	Modules C
D-Type	DF1	D-Type Flip-Flop	1	
	DF1A	D-Type Flip-Flop with active low output	1	
	DF1B	D-Type Flip-Flop with active low clock	1	
	DF1C	D-Type Flip-Flop with active low clock and output	1	
	DFC1	D-Type Flip-Flop with active high Clear	1	1
	DFC1A	D-Type Flip-Flop with active high Clear and active low clock	1	1
	DFC1B	D-Type Flip-Flop with active low Clear	1	
	DFC1D	D-Type Flip-Flop with active low Clear and clock	1	
	DFE	D-Type Flip-Flop with active high Enable	1	
	DFE1B	D-Type Flip-Flop with active low Enable	1	
	DFE1C	D-Type Flip-Flop with active low Enable and clock	1	
	DFE3A	D-Type Flip-Flop with Enable and active low Clear	1	
	DFE3B	D-Type Flip-Flop with Enable and active low Clear and clock	1	
	DFE3C	D-Type Flip-Flop with active low Enable and Clear	1	
	DFE3D	D-Type Flip-Flop with active low Enable, Clear, and clock	1	
	DFEA	D-Type Flip-Flop with Enable and active low clock	1	
	DFM	2-bit D-Type Flip-Flop with Multiplexed Data	1	
	DFM1B	2-bit D-Type Flip-Flop with Multiplexed Data and active low output	1	
	DFM1C	2-bit D-Type Flip-Flop with Multiplexed Data and active low clock and output	1	
	DFM3	2-bit D-Type Flip-Flop with Multiplexed Data and Clear	1	1
	DFM3B	2-bit D-Type Flip-Flop with Multiplexed Data and active low Clear and clock	1	
	DFM3E	2-bit D-Type Flip-Flop with Multiplexed Data, Clear, and active low clock	1	1
	DFM4C	2-bit D-Type Flip-Flop with Multiplexed Data and active low Preset and output	1	
	DFM4D	2-bit D-Type Flip-Flop with Multiplexed Data and active low Preset, clock, and output	1	
	DFM6A	4-bit D-Type Flip-Flop with Multiplexed Data, active low Clear, and active high Clock	1	
	DFM6B	4-bit D-Type Flip-Flop with Multiplexed Data, active low Clear, and clock	1	
	DFM7A	4-bit D-Type Flip-Flop with Multiplexed Data, active low Clear, and active high clock	1	
	DFM7B	4-bit D-Type Flip-Flop with Multiplexed Data, active low Clear and clock	1	
	DFMA	2-bit D-Type Flip-Flop with Multiplexed Data and active low clock	1	
	DFMB	2-bit D-Type Flip-Flop with Multiplexed Data and active low Clear	1	
	DFME1A	2-bit D-Type Flip-Flop with Multiplexed Data and active low Enable	1	
	DFP1	D-Type Flip-Flop with active high Preset		2
	DFP1A	D-Type Flip-Flop with active high Preset and active low clock		2
	DFP1B	D-Type Flip-Flop with active low Preset		2
	DFP1C	D-Type Flip-Flop with active high Preset and active low output	1	1
	DFP1D	D-Type Flip-Flop with active low Preset and clock		2
	DFP1E	D-Type Flip-Flop with active low Preset and output	1	
	DFP1F	D-Type Flip-Flop with active high Preset and active low clock and output	1	1
	DFP1G	D-Type Flip-Flop with active low Preset, clock, and output	1	
	DFPC	D-Type Flip-Flop with active high Preset, active low Clear, and active high clock		2
	DFPCA	D-Type Flip-Flop with active high Preset, and active low Clear and clock		2
J-K Type	JKF	JK Flip-Flop with active low K-input	1	

Hard Macros—Sequential (Continued)

Function	Macro	Description	Modules S	Modules C
J-K Type	JKF1B	JK Flip-Flop with active low clock and K-input	1	
	JKF2A	JK Flip-Flop with active low Clear and K-input	1	
	JKF2B	JK Flip-Flop with active low Clear, clock, and K-input	1	
	JKF2C	JK Flip-Flop with active high Clear and active low K-input	1	1
	JKF2D	JK Flip-Flop with active high Clear and active low clock and K-input	1	1
T-Type	TF1A	T-Type Flip-Flop with active low Clear	1	
	TF1B	T-Type Flip-Flop with active low Clear and clock	1	
Latch	DL1	Data Latch	1	
	DL1A	Data Latch with active low output	1	
	DL1B	Data Latch with active low clock	1	
	DL1C	Data Latch with active low clock and output	1	
	DLC	Data Latch with active low Clear	1	
	DLC1	Data Latch with active high Clear		1
	DLC1A	Data Latch with active high Clear and active low clock		1
	DLC1F	Data Latch with active high Clear and active low output		1
	DLC1G	Data Latch with active high Clear and active low clock and output		1
	DLCA	Data Latch with active low Clock and Clear	1	
	DLE	Data Latch with active high Enable	1	
	DLE1D	Data Latch with active high Enable and clock and active low input and output	1	
	DLE2B	Data Latch with active low Enable, Clear, and clock	1	
	DLE2C	Data Latch with active low Enable and clock and active high Clear		1
	DLE3B	Data Latch with active low Enable and clock and active low Preset		1
	DLE3C	Data Latch with active low Enable, Preset, and clock		1
	DLEA	Data Latch with active low Enable and active high clock	1	
	DLEB	Data Latch with active high Enable and active high clock	1	
	DLEC	Data Latch with active low Enable and clock	1	
	DLM	2-bit Data Latch with Multiplexed Data	1	
	DLM3	4-bit Data Latch with Multiplexed Data	1	
	DLM3A	4-bit Data Latch with Multiplexed Data and active low clock	1	
	DLM4	Data Latch with Multiplexed Data	1	
	DLM4A	Data Latch with Multiplexed Data	1	
	DLMA	2-bit Data Latch with Multiplexed Data, and active low clock	1	
	DLME1A	2-bit Data Latch with Multiplexed Data and Enable and active low clock	1	
	DLP1	Data Latch with active high Preset and clock		1
	DLP1A	Data Latch with active high Preset and active low clock		1
	DLP1B	Data Latch with active low Preset and active high clock		1
	DLP1C	Data Latch with active low Preset and clock		1
	DLP1D	Data Latch with active low Preset and output and active high clock	1	
	DLP1E	Data Latch with active low Preset, clock, and output	1	

Soft Macros—TTL Equivalent

Function	Macro	Description	Maximum Logic Levels	Modules S	Modules C
	TA00	2-input NAND	1		1
	TA02	2-input NOR	1		1
	TA04	Inverter	1		1
	TA07	Buffer	1		1
	TA08	2-input AND	1		1
	TA10	3-input NAND	1		1
	TA11	3-input AND	1		1
	TA138	3-to-8 decoder with enable and active low outputs	2		12
	TA139	2-to-4 decoder with active low enable and outputs	1		4
	TA150	16-to-1 multiplexor with active low enable	3		6
	TA151	8-to-1 multiplexor with enable and both active low and active high output	3		5
	TA153	4-to-1 multiplexor with active low enable	2		2
	TA154	4-to-16 decoder with active low outputs and select lines	2		22
	TA157	2-to-1 multiplexor with active low enable	1		1
	TA160	4-bit decade counter with active low clear and load	4	4	8
	TA161	4-bit binary counter with active low clear and load	3	4	6
	TA164	8-bit serial in, parallel out shift register, active low clear	1	8	
	TA169	4-bit Up/Down Counter	6	4	14
	TA174	hex D-type flip-flop with active low clear	1	6	
	TA175	quadruple D-type flip-flop with active low clear	1	4	
	TA181	ALU			37
	TA190	4-bit up/down decode counter with up/down mode	7	4	31
	TA191	4-bit up/down binary counter with up/down mode	7	4	30
	TA194	4-bit bidirectional universal shift register	1	4	4
	TA195	4-bit parallel-access shift register	1	4	1
	TA20	4-input NAND	1		2
	TA21	4-input AND	1		1
	TA269	8-bit up/down binary counter	8	8	28
	TA27	3-input NOR	1		1
	TA273	octal register with clear	1	8	
	TA280	9-bit odd/even parity generator and checker	4		9
	TA32	2-input OR	1		1
	TA377	octal register with active low enable	1	8	
	TA40	4-input NAND	1		2
	TA42	4 to 10 decoder	1		10
	TA51	AND-OR-Invert	1		2
	TA54	4-wide 2-input AND-OR-Invert	2		5
	TA55	2-wide 4-input AND-OR-Invert	2		3
	TA688	8-bit identity comparator	3		9
	TA86	2-input exclusive OR	1		1

Input/Output Macros

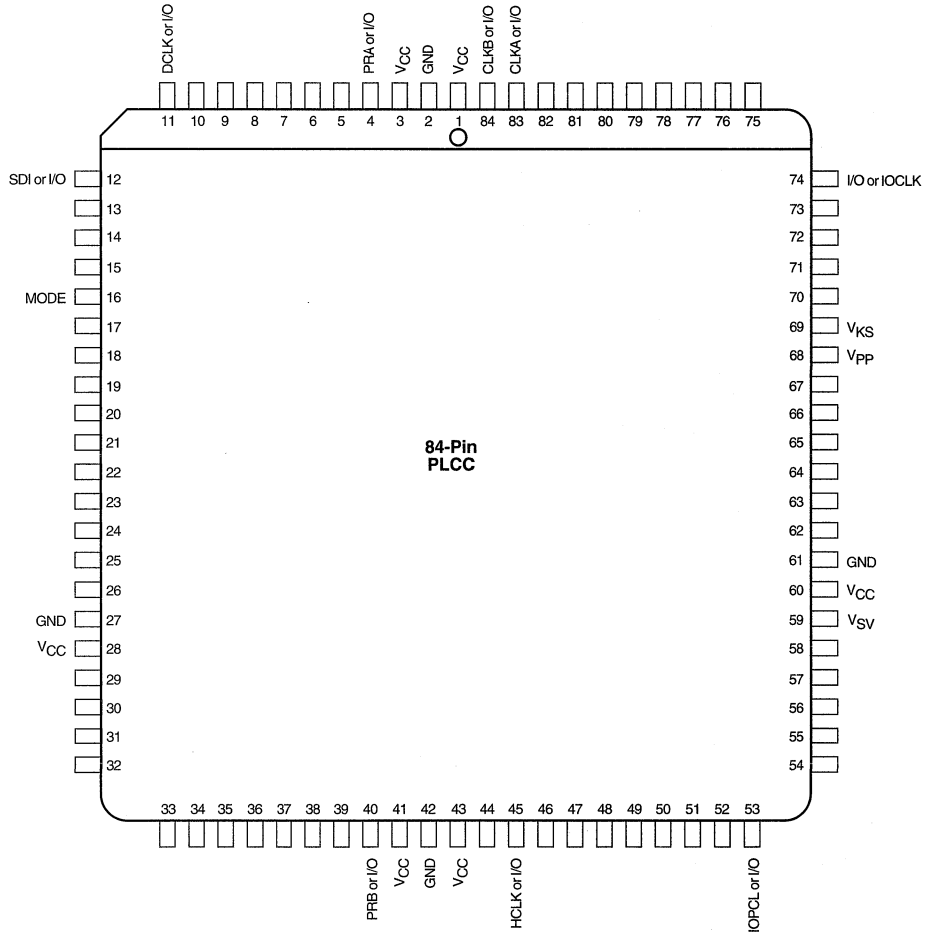
Function	Macro	Description	I/O Modules
Buffer	BBHS	Bidirectional Buffer, High Slew	1
	BBUFTH	Bidirectional Buffer, Tristate Enable, High Slew	1
	BBUFTL	Bidirectional Buffer, Tristate Enable, Low Slew	1
	BIBUF	Bidirectional Buffer, High Slew (with hidden buffer at Y pin)	1
	HCLKBUF	Dedicated High-Speed S-Module Clock Buffer	1
	IBUF	Input Buffer	1
	INBUF	Input Buffer	1
	IOCLKBUF	Dedicated I/O Module Clock Buffer	1
	IOPCLBUF	Dedicated I/O Module IOPCL Buffer	1
	OBHS	Output buffer, High Slew	1
	OBUFTH	Output Buffer, Tristate Enable, High Slew	1
	OBUFTL	Output Buffer, Tristate Enable, Low Slew	1
	OUTBUF	Output Buffer, High Slew	1
Bidirectional	BRECTH	Bidirectional, Output Register with Clear, Data Enable, Tristate Enable, High Slew	1
	BRECTL	Bidirectional, Output Register with Clear, Data Enable, Tristate Enable, Low Slew	1
	BREPTH	Bidirectional, Output Register with Preset, Data Enable, Tristate Enable, High Slew	1
	BREPTL	Bidirectional, Output Register with Preset, Data Enable, Tristate Enable, Low Slew	1
	CLKBIBUF	Bidirectional with Input Dedicated to Clock Network	1
	DECETH	Bidirectional, Double Registered with Clear, Data Enable, Tristate Enable, High Slew	1
	DECETL	Bidirectional, Double Registered with Clear, Data Enable, Tristate Enable, Low Slew	1
	DEPETH	Bidirectional, Double Registered with Preset, Data Enable, Tristate Enable, High Slew	1
	DEPETL	Bidirectional, Double Registered with Preset, Data Enable, Tristate Enable, Low Slew	1
Input	CLKBUF	Input for Dedicated Routed Clock Network	1
	IREC	Input Register with Clear	1
	IREP	Input Register with Preset	1
Output	FECTMH	Output Register with Muxed Feedback, Clear, Data Enable, Tristate Enable, High Slew	1
	FECTML	Output Register with Muxed Feedback, Clear, Data Enable, Tristate Enable, Low Slew	1
	FEPTMH	Output Register with Muxed Feedback, Preset, Data Enable, Tristate Enable, High Slew	1
	FEPTML	Output Register with Muxed Feedback, Preset, Data Enable, Tristate Enable, Low Slew	1
	ORECTH	Output Register with Clear, Data Enable, Tristate Enable, High Slew	1
	ORECTL	Output Register with Clear, Data Enable, Tristate Enable, Low Slew	1
	OREPTH	Output Register with Preset, Data Enable, Tristate Enable, High Slew	1
	OREPTL	Output Register with Preset, Data Enable, Tristate Enable, Low Slew	1
	TBHS	Tristate output, High Slew	1
	TRIBUFF	Tristate output, High Slew	1

Soft Macros

Function	Macro	Description	Maximum Logic Levels	Modules S	Modules C
Adder	FADD10	10-bit adder	3		56
	FADD12	12-bit adder	4		9
	FADD16	16-bit adder	5		97
	FADD8	8-bit adder	4		44
	FADD9	9-bit adder with active low carry out	3		49
	VAD16C	Very fast 16-bit adder, no Carry in	3		97
	VADC16C	Very fast 16-bit adder with Carry in	3		97
Comparator	ICMP4	4-bit Identity Comparator	2		5
	ICMP8	8-bit Identity Comparator	3		9
	MCMPC2	2-bit Magnitude Comparator with Enable	3		9
	MCMPC4	4-bit Magnitude Comparator with Enable	4		18
	MCMPC8	8-bit Magnitude Comparator with Enable	6		36
Counter	CNT4A	4-bit binary counter with load and clear	4	4	8
	CNT4B	4-bit binary counter with load, clear, carry-in, carry-out	4	4	7
	FCTD16C	Fast 16-bit Down Counter, parallel loadable	2	19	33
	FCTD8A	Fast 8-bit Down Counter, parallel loadable	1	10	18
	FCTD8B	Fast 8-bit Down Counter, parallel loadable	1	9	13
	FCTU16C	Fast 16-bit Up Counter, parallel loadable	2	19	31
	FCTU8A	Fast 8-bit Up Counter, parallel loadable	1	10	17
	FCTU8B	Fast 8-bit Up Counter, parallel loadable	1	9	12
	UDCNT4A	4-bit up/down counter with load, carry-in, and carry-out	5	4	13
	VCTD16C	Very fast 16-bit down counter, delay after load, registered control inputs	1	34	41
	VCTD2CP	2-bit down counter, prescaler, delay after load, use to build VCTD counters	1	5	2
	VCTD2CU	2-bit down counter, upper bits, delay after load, use to build VCTD counters	1	2	3
	VCTD4CL	4-bit down counter, lower bits, delay after load, use to build VCTD counters	1	4	7
	VCTD4CM	4-bit down counter, middle bits, delay after load, use to build VCTD counters	1	4	8
Decoder	DEC2X4	2-to-4 decoder	1		4
	DEC2X4A	2-to-4 decoder with active low outputs	1		4
	DEC3X8	3-to-8 decoder	1		8
	DEC3X8A	3-to-8 decoder with active low outputs	1		8
	DEC4X16A	4-to-16 decoder with active low outputs	2		20
	DECE2X4	2-to-4 decoder with enable	1		4
	DECE2X4A	2-to-4 decoder with enable and active low outputs	1		4
	DECE3X8	3-to-8 decoder with enable	2		11
	DECE3X8A	3-to-8 decoder with enable and active low outputs	2		11
Latch	DLC8A	octal latch with clear active low 8-bit Data Latch with active low Clear	1	8	
	DLE8	octal latch with enable 8-bit Data Latch with active high Enable	1	8	
	DLM8	octal latch with multiplexed data 8-bit Data Latch with Multiplexed Data	1	8	
MUX	MX16	16-to-1 Multiplexor	2		5
	MX8	8-to-1 Multiplexor with active high output	2		3
	MX8A	8-to-1 Multiplexor with active low output	2		3
Multiplier	SMULT8	8-bit by 8-bit Multiplier			242
Shift Register	SREG4A	4-bit shift register with clear active low	1	4	
	SREG8A	8-bit shift register with clear active low	1	8	

Package Pin Assignments

84-Pin PLCC (Top View)

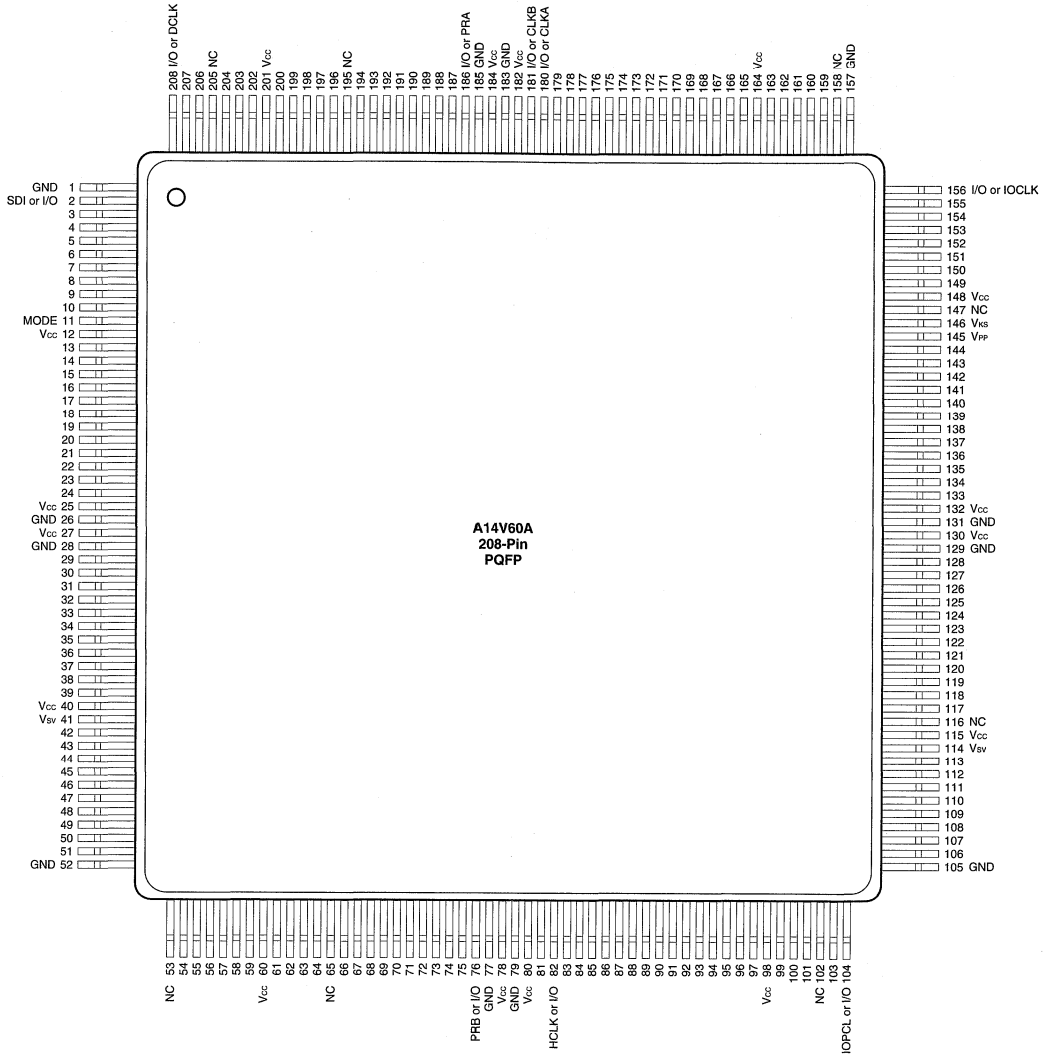


Notes:

1. *Unused I/O pins are designated as outputs by ALS and are driven low.*
2. *All unassigned pins are available for use as I/Os.*
3. *MODE = GND, except during device programming or debugging.*
4. *$V_{PP} = V_{CC}$, except during device programming.*
5. *$V_{SV} = V_{CC}$, except during device programming.*
6. *V_{KS} = GND, except during device programming.*

Package Pin Assignments (continued)

208-Pin PQFP (Top View)

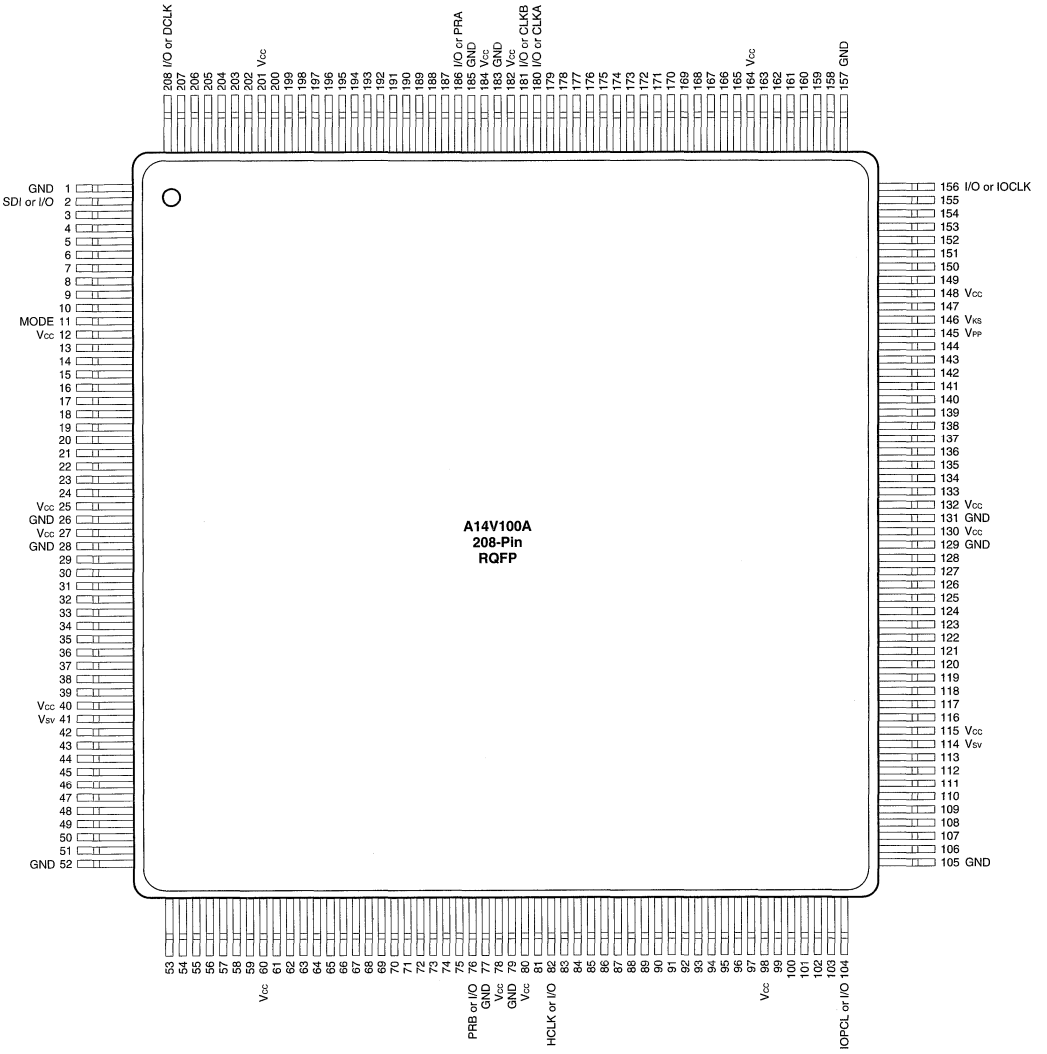


Notes:

1. *Unused I/O pins are designated as outputs by ALS and are driven low.*
2. *All unassigned pins are available for use as I/Os.*
3. *MODE = GND, except during device programming or debugging.*
4. $V_{PP} = V_{CC}$, *except during device programming.*
5. $V_{SV} = V_{CC}$, *except during device programming.*
6. V_{KS} = *GND, except during device programming.*

Package Pin Assignments (continued)

208-Pin RQFP (Top View)

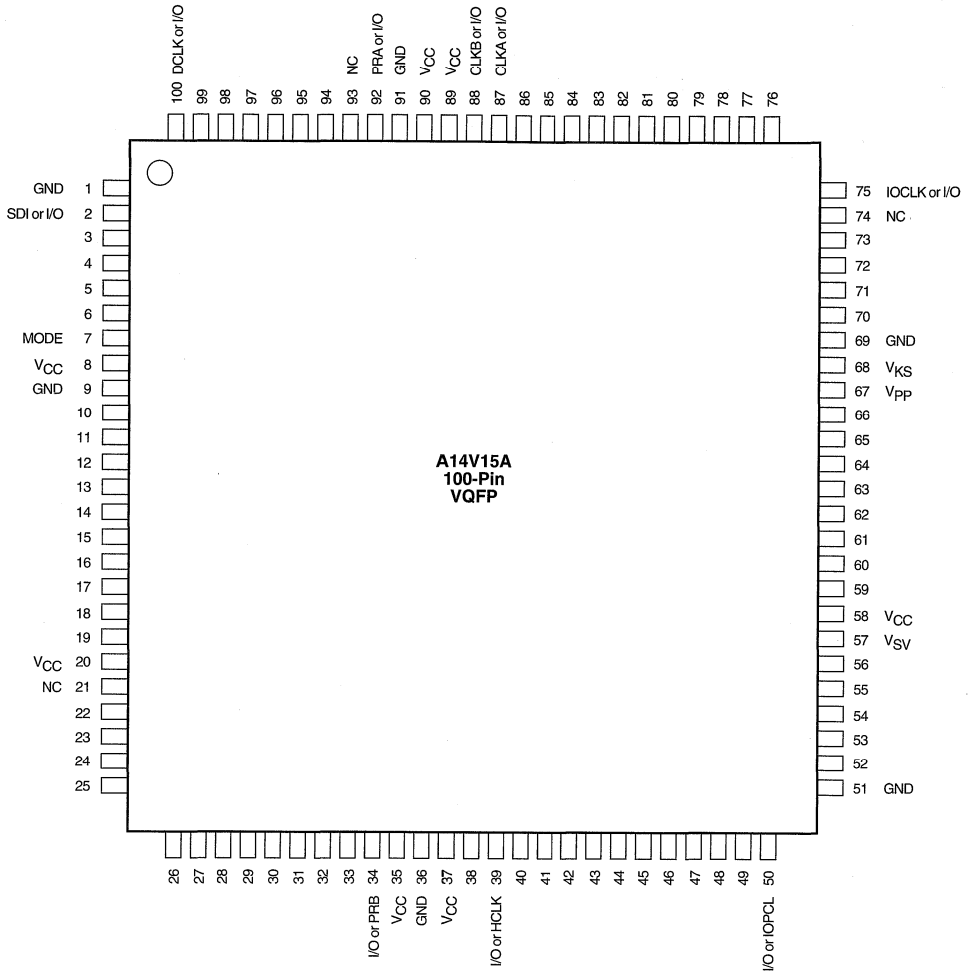


Notes:

1. *Unused I/O pins are designated as outputs by ALS and are driven low.*
2. *All unassigned pins are available for use as I/Os.*
3. *MODE = GND, except during device programming or debugging.*
4. $V_{PP} = V_{CC}$, *except during device programming.*
5. $V_{SV} = V_{CC}$, *except during device programming.*
6. V_{KS} = *GND, except during device programming.*

Package Pin Assignments (continued)

100-Pin VQFP (Top View)

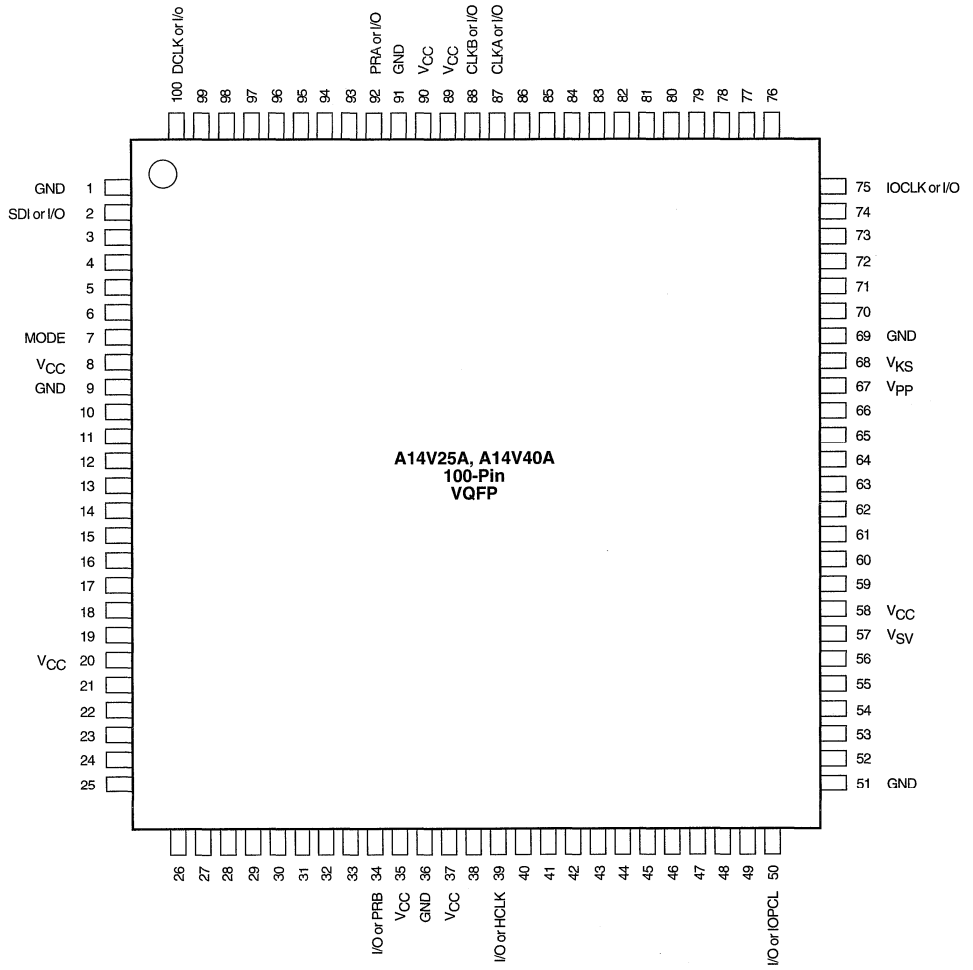


Notes:

1. *Unused I/O pins are designated as outputs by ALS and are driven low.*
2. *All unassigned pins are available for use as I/Os.*
3. *MODE = GND, except during device programming or debugging.*
4. *$V_{PP} = V_{CC}$, except during device programming.*
5. *$V_{SV} = V_{CC}$, except during device programming.*
6. *V_{KS} = GND, except during device programming.*

Package Pin Assignments (continued)

100-Pin VQFP (Top View)

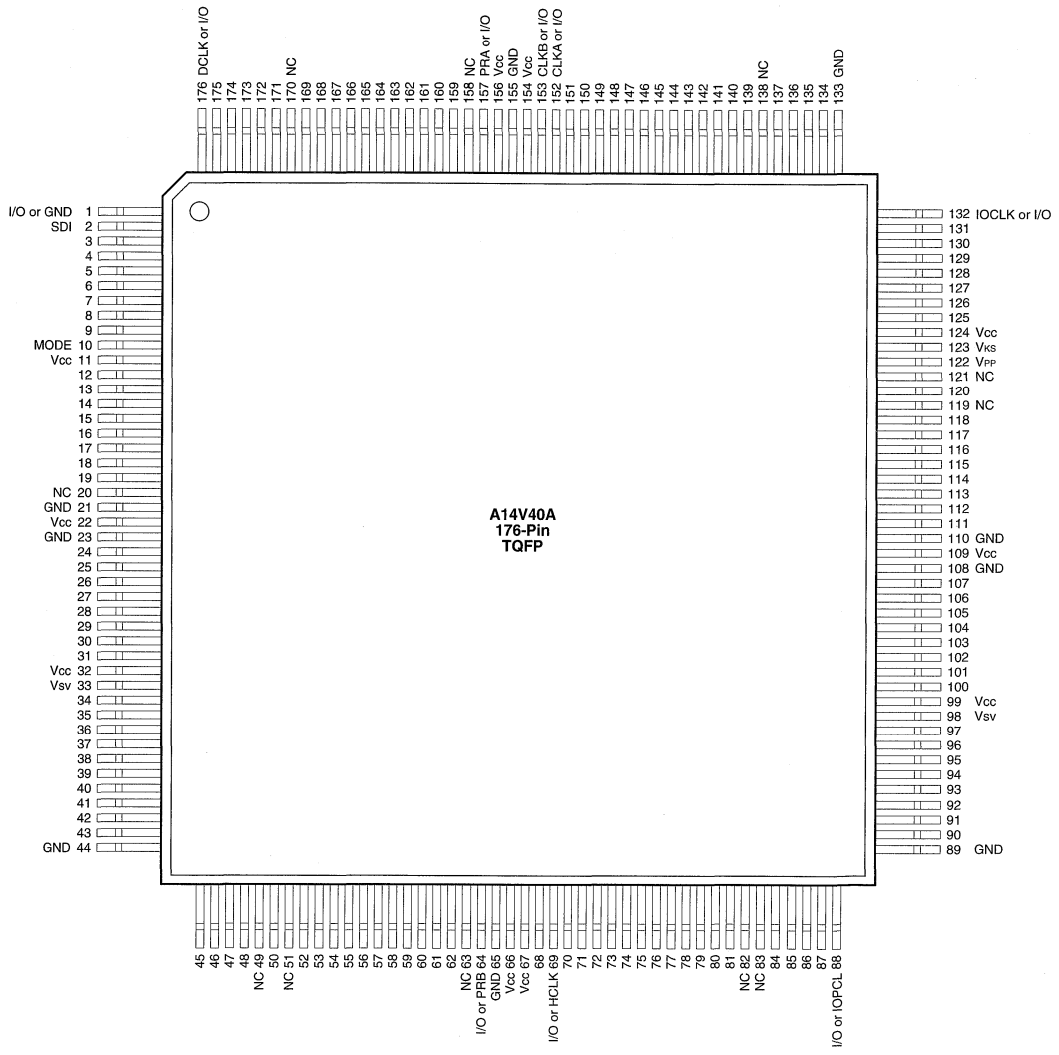


Notes:

1. *Unused I/O pins are designated as outputs by ALS and are driven low.*
2. *All unassigned pins are available for use as I/Os.*
3. *MODE = GND, except during device programming or debugging.*
4. *$V_{PP} = V_{CC}$, except during device programming.*
5. *$V_{SV} = V_{CC}$, except during device programming.*
6. *V_{KS} = GND, except during device programming.*

Package Pin Assignments (continued)

176-Pin TQFP (Top View)

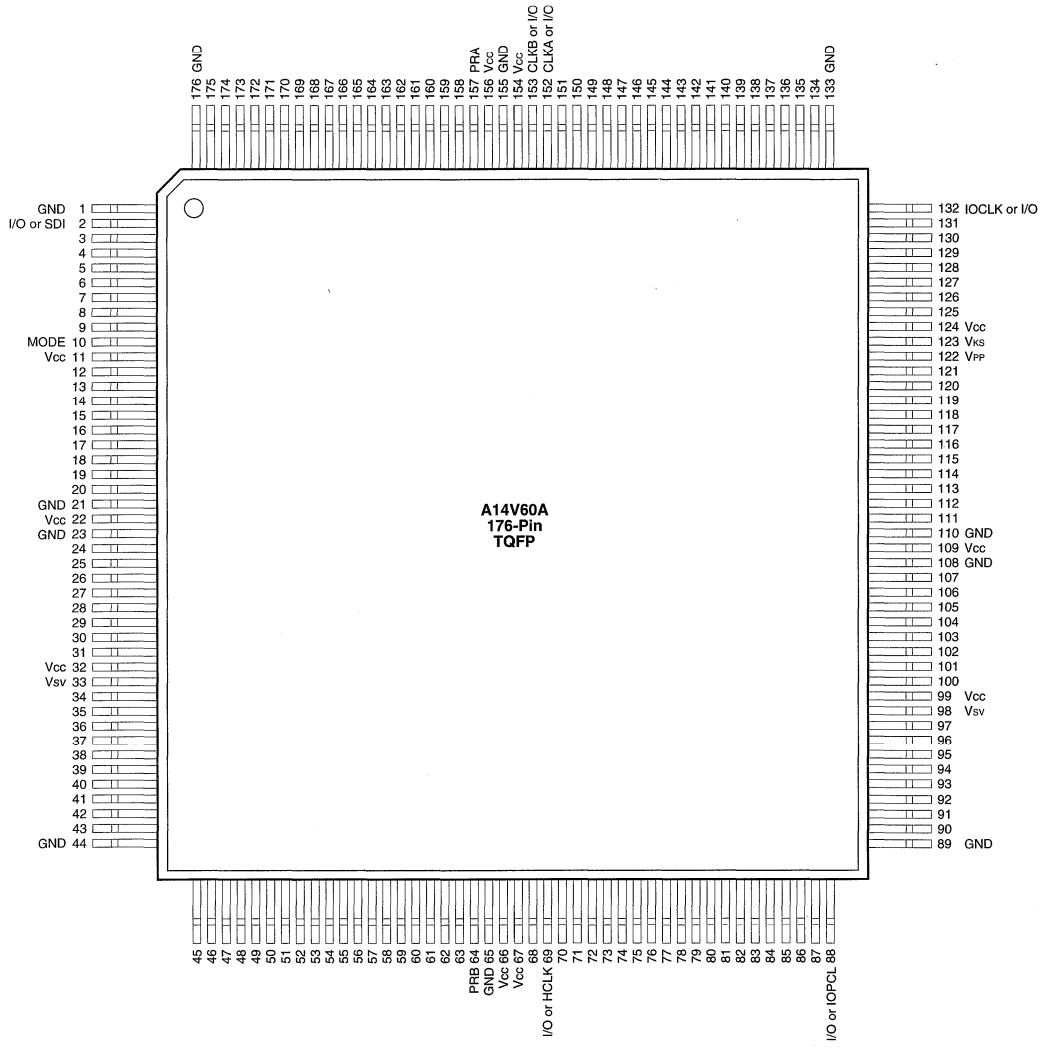


Notes:

1. *Unused I/O pins are designated as outputs by ALS and are driven low.*
2. *All unassigned pins are available for use as I/Os.*
3. *MODE = GND, except during device programming or debugging.*
4. *$V_{PP} = V_{CC}$, except during device programming.*
5. *$V_{SV} = V_{CC}$, except during device programming.*
6. *V_{KS} = GND, except during device programming.*

Package Pin Assignments (continued)

176-Pin TQFP (Top View)

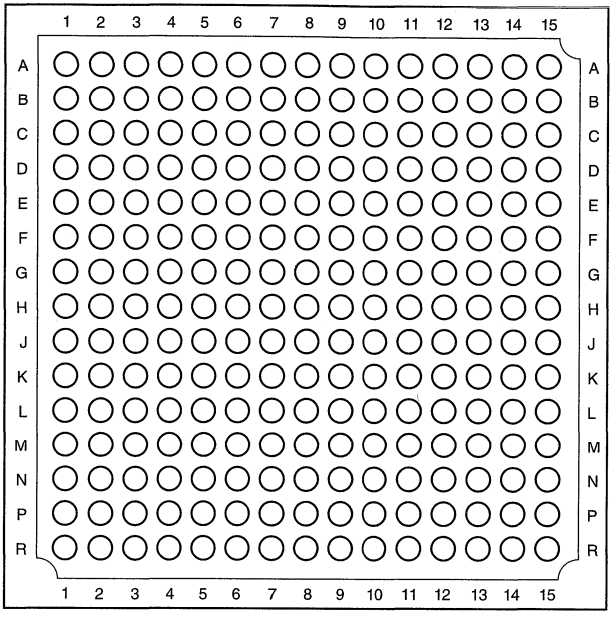


Notes:

1. *Unused I/O pins are designated as outputs by ALS and are driven low.*
2. *All unassigned pins are available for use as I/Os.*
3. *MODE = GND, except during device programming or debugging.*
4. *$V_{PP} = V_{CC}$, except during device programming.*
5. *$V_{SV} = V_{CC}$, except during device programming.*
6. *V_{KS} = GND, except during device programming.*

Package Pin Assignments (continued)

225-Pin BGA (Top View)



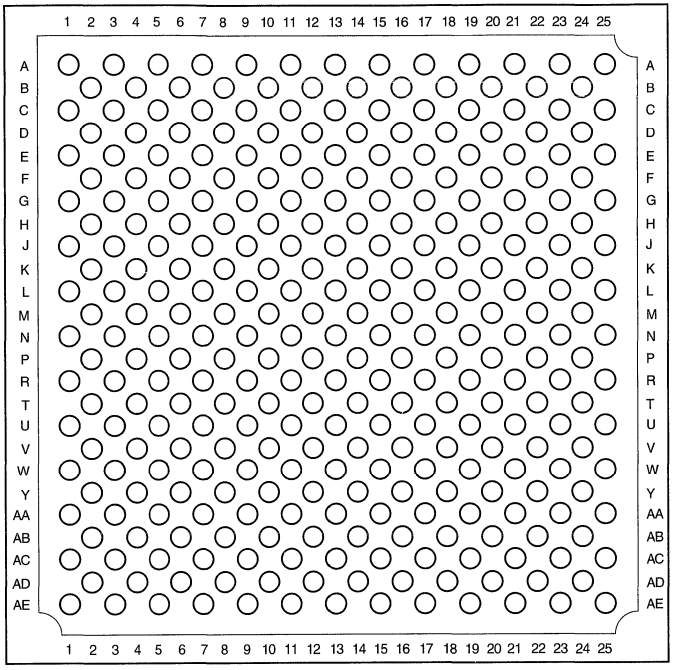
Signal	Location
CLKA or I/O	C8
CLKB or I/O	B8
DCLK or I/O	B2
GND	A1, A15, F8, G7, G8, G9, H6, H7, H8, H9, H10, J7, J8, J9, K8, P2, R15
HCLK or I/O	P9
IOCLK or I/O	B14
IOPCL or I/O	P14
MODE	D1
NC	A11, B5, B7, D8, D12, F6, F11, H1, H12, H14, K11, L1, L13, N8, P5, R1, R8, R11, R14
PRA OR I/O	A7
PRB or I/O	L7
SDI or I/O	D4
V_{CC}	A8, B12, D5, D14, E3, E8, H2, H3, H11, H15, L2, M8, M15, P4, P8, R13
V_{KS}	D15
V_{PP}	E13
V_{SV}	K4, L12

Notes:

1. *Unused I/O pins are designated as outputs by ALS and are driven low.*
2. *All unassigned pins are available for use as I/Os.*
3. *MODE = GND, except during device programming or debugging.*
4. *$V_{PP} = V_{CC}$, except during device programming.*
5. *$V_{SV} = V_{CC}$, except during device programming.*
6. *V_{KS} = GND, except during device programming.*

Package Pin Assignments (continued)

313-Pin BGA (Top View)



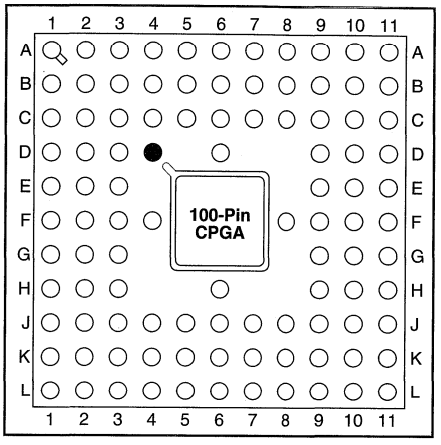
Signal	Location
CLKA or I/O	J13
CLKB or I/O	G13
DCLK or I/O	B2
GND	A1, A25, AD2, AE25, L13, M12, M14, N11, N13, N15, P12, P14, R13
HCLK or I/O	T14
IOCLK or I/O	B24
IOPCL or I/O	AD24
MODE	G3
NC	A3, A13, A23, AA5, AA9, AA23, AB2, AB4, AB20, AC13, AC25, AD22, AE1, AE21, B14, C5, C25, D4, D24, E3, E21, F6, F10, F16, G1, G25, H18, H24, J1, J7, J25, K12, L15, L17, M6, N1, N5, N7, N21, N23, P20, R11, T6, T8, U9, U13, U21, V16, W7, Y20, Y24
PRA OR I/O	H12
PRB or I/O	AD12
SDI or I/O	C1
V_{CC}	AB18, AD6, AE13, C13, C19, E13, G9, H22, K8, M16, N3, N9, N25, U5, W13, V24
V_{KS}	J21
V_{PP}	K20
V_{SV}	V2, V22

Notes:

1. *Unused I/O pins are designated as outputs by ALS and are driven low.*
2. *All unassigned pins are available for use as I/Os.*
3. *MODE = GND, except during device programming or debugging.*
4. *$V_{PP} = V_{CC}$, except during device programming.*
5. *$V_{SV} = V_{CC}$, except during device programming.*
6. *V_{KS} = GND, except during device programming.*

Package Pin Assignments (continued)

100-Pin CPGA (Top View)



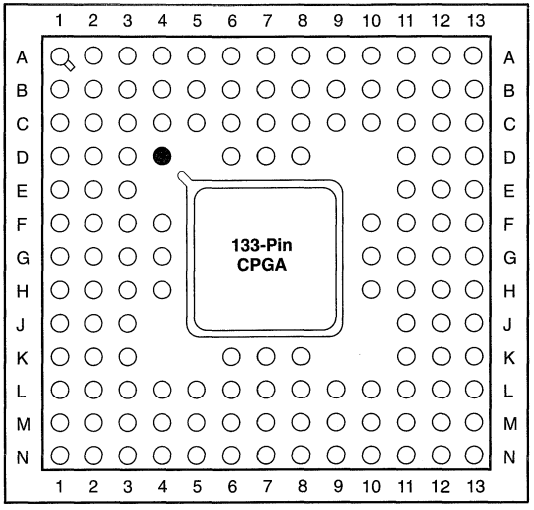
Signal	Location
CLKA or I/O	C7
CLKB or I/O	D6
DCLK or I/O	C4
GND	C3, F3, J3, C6, J6, J8, C9, F9, J9
HCLK or I/O	H6
IOCLK or I/O	C10
IOPCL or I/O	K9
MODE	C2
PRA OR I/O	A6
PRB or I/O	L3
SDI or I/O	B3
V_{CC}	F2, K2, B6, K6, B10, F10, K10
V_{KS}	E9
V_{PP}	E11
V_{SV}	G2

Notes:

1. *Unused I/O pins are designated as outputs by ALS and are driven low.*
2. *All unassigned pins are available for use as I/Os.*
3. *MODE = GND, except during device programming or debugging.*
4. *$V_{PP} = V_{CC}$, except during device programming.*
5. *$V_{SV} = V_{CC}$, except during device programming.*
6. *V_{KS} = GND, except during device programming.*

Package Pin Assignments (continued)

133-Pin CPGA (Top View)



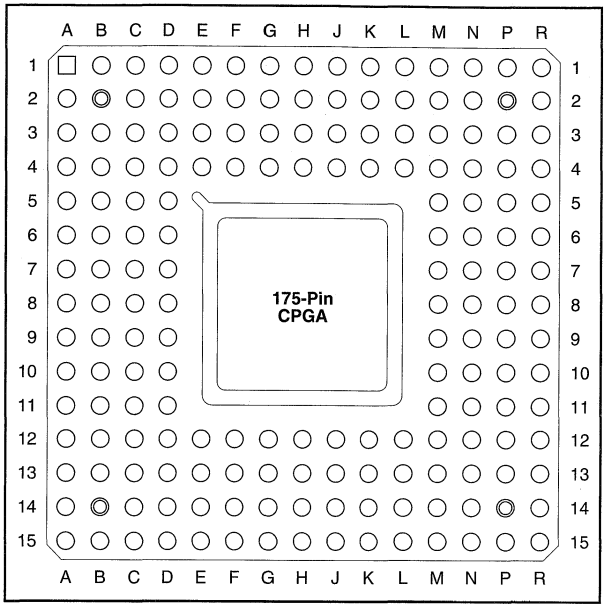
Signal	Location
CLKA or I/O	D7
CLKB or I/O	B6
DCLK or I/O	D4
GND	A2, C3, C7, C11, C12, G3, G11, L3, L7, L11, M3, N12
HCLK or I/O	K7
IOCLK or I/O	C10
IOPCL or I/O	L10
MODE	E3
NC	A1, A7, A13, G1, G13, N1, N7, N13
PRA OR I/O	A6
PRB or I/O	L6
SDI or I/O	C2
V_{CC}	B2, B7, B12, G2, G12, M2, M7, M12
V_{KS}	F10
V_{PP}	E11
V_{SV}	J2, J12

Notes:

1. *Unused I/O pins are designated as outputs by ALS and are driven low.*
2. *All unassigned pins are available for use as I/Os.*
3. *MODE = GND, except during device programming or debugging.*
4. *$V_{PP} = V_{CC}$, except during device programming.*
5. *$V_{SV} = V_{CC}$, except during device programming.*
6. *V_{KS} = GND, except during device programming.*

Package Pin Assignments (continued)

175-Pin CPGA (Top View)



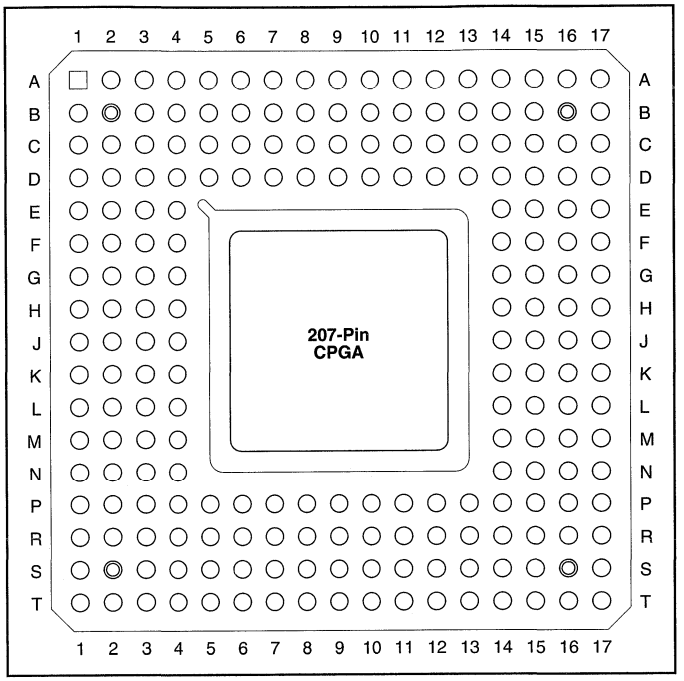
Signal	Location
CLKA or I/O	C9
CLKB or I/O	A9
DCLK or I/O	D5
GND	D4, D8, D11, D12, E4, H4, H12, L4, L12, M4, M8, M12
HCLK or I/O	R8
IOCLK or I/O	E12
IOPCL or I/O	P13
MODE	F3
NC	A1, A2, A15, B2, B3, P2, P14, R1, R2, R14, R15
PRA OR I/O	B8
PRB or I/O	R7
SDI or I/O	D3
V_{CC}	C3, C8, C13, H3, H13, N3, N8, N13
V_{KS}	E14
V_{PP}	E15
V_{SV}	L1, L14

Notes:

1. *Unused I/O pins are designated as outputs by ALS and are driven low.*
2. *All unassigned pins are available for use as I/Os.*
3. *MODE = GND, except during device programming or debugging.*
4. *$V_{PP} = V_{CC}$, except during device programming.*
5. *$V_{SV} = V_{CC}$, except during device programming.*
6. *V_{KS} = GND, except during device programming.*

Package Pin Assignments (continued)

207-Pin CPGA (Top View)



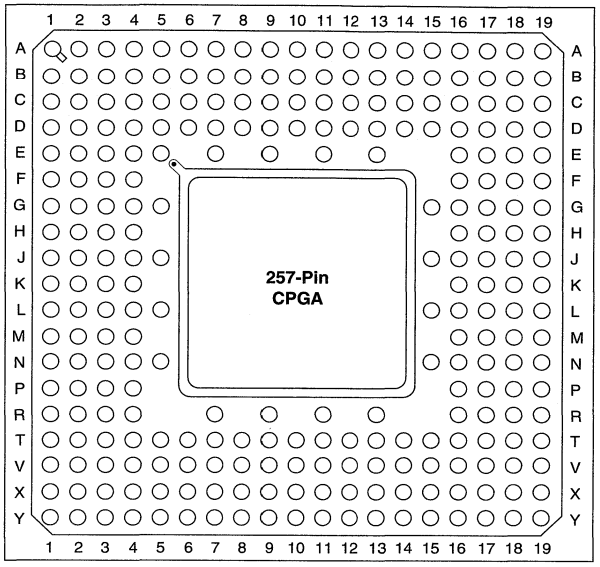
Signal	Location
CLKA or I/O	K1
CLKB or I/O	J3
DCLK or I/O	E4
GND	C15, D4, D5, D9, D14, J4, J14, P3, P4, P9, P14, R15
HCKL or I/O	J15
IOCLK or I/O	P5
IOPCL or I/O	N14
MODE	D7
NC	A1, A2, A16, A17, B1, B17, C1, C2, S1, S3, S17, T1, T2, T16, T17
PRA OR I/O	H1
PRB or I/O	K16
SDI or I/O	C3
V_{CC}	B2, B9, B16, J2, J16, S2, S9, S16
V_{KS}	P7
V_{PP}	T5
V_{SV}	D11, P12

Notes:

1. *Unused I/O pins are designated as outputs by ALS and are driven low.*
2. *All unassigned pins are available for use as I/Os.*
3. *MODE = GND, except during device programming or debugging.*
4. *$V_{PP} = V_{CC}$, except during device programming.*
5. *$V_{SV} = V_{CC}$, except during device programming.*
6. *V_{KS} = GND, except during device programming.*

Package Pin Assignments (continued)

257-Pin CPGA (Top View)



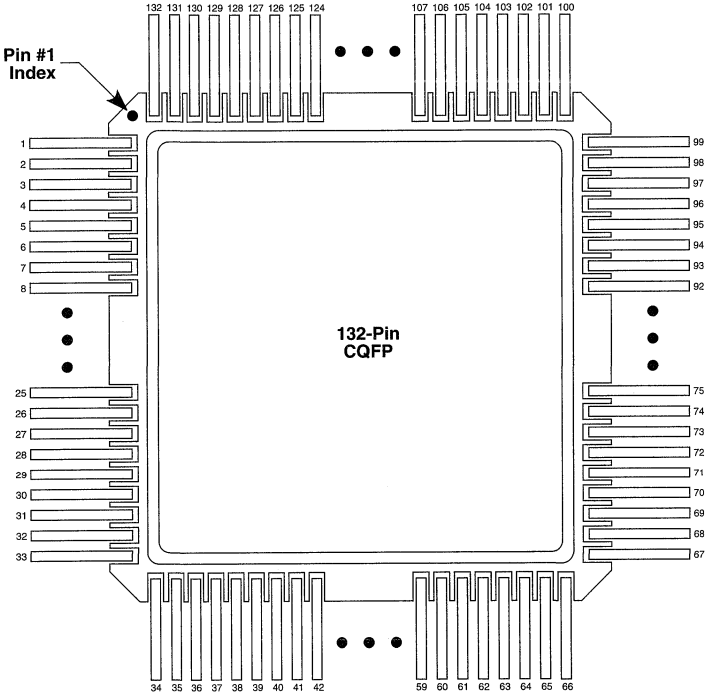
Signal	Location
CLKA or I/O	L4
CLKB or I/O	L5
DCLK or I/O	E4
GND	C4, B16, T17, R4, D4, D10, D16, E11, J5, K4, K16, L15, T4, T10, T16
HCLK or I/O	J16
IOCLK or I/O	T5
IOPCL or I/O	R16
MODE	A5
NC	E5
PRA OR I/O	J1
PRB or I/O	J17
SDI or I/O	B4
V_{CC}	C3, C10, C17, K3, K17, V3, V10, V17
V_{KS}	X7
V_{PP}	V7
V_{SV}	C13, X14

Notes:

1. *Unused I/O pins are designated as outputs by ALS and are driven low.*
2. *All unassigned pins are available for use as I/Os.*
3. *MODE = GND, except during device programming or debugging.*
4. *$V_{PP} = V_{CC}$, except during device programming.*
5. *$V_{SV} = V_{CC}$, except during device programming.*
6. *V_{KS} = GND, except during device programming.*

Package Pin Assignments (continued)

132-Pin CQFP (Top View)



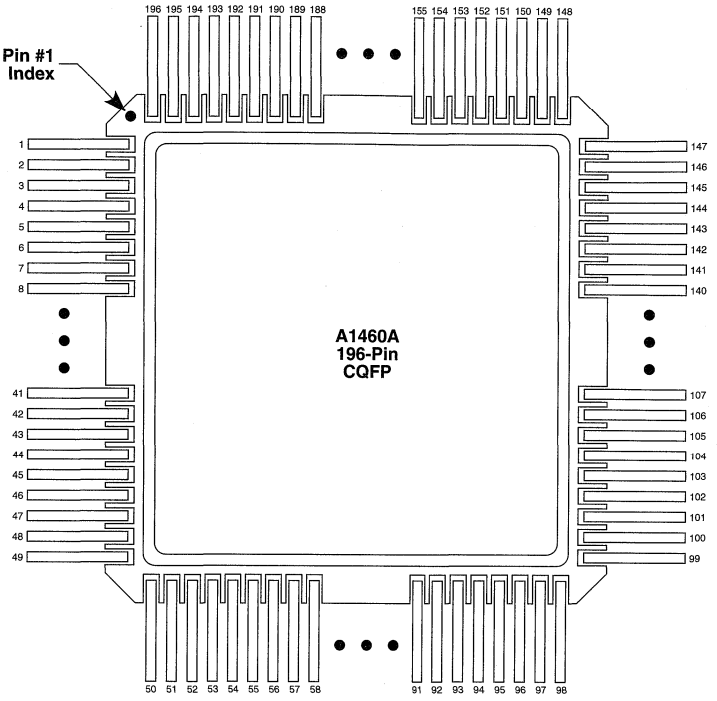
Signal	PIN Number
CLKA or I/O	116
CLKB or I/O	117
DCLK or I/O	131
GND	2, 10, 26, 36, 42, 58, 65, 74, 90, 101, 106, 122
HCLK or I/O	50
IOCLK or I/O	98
IOPCL or I/O	64
MODE	9
NC	1, 34, 66, 67, 99, 100, 132
PRA or I/O	118
PRB or I/O	48
SDI or I/O	3
V_{CC}	11, 27, 43, 59, 75, 91, 107, 123
V_{KS}	92
V_{PP}	89
V_{SV}	22, 78

Notes:

1. *Unused I/O pins are designated as outputs by ALS and are driven low.*
2. *All unassigned pins are available for use as I/Os.*
3. *MODE = GND, except during device programming or debugging.*
4. *$V_{PP} = V_{CC}$, except during device programming.*
5. *$V_{SV} = V_{CC}$, except during device programming*
6. *V_{KS} = GND, except during device programming.*

Package Pin Assignments (continued)

196-Pin CQFP (Top View)



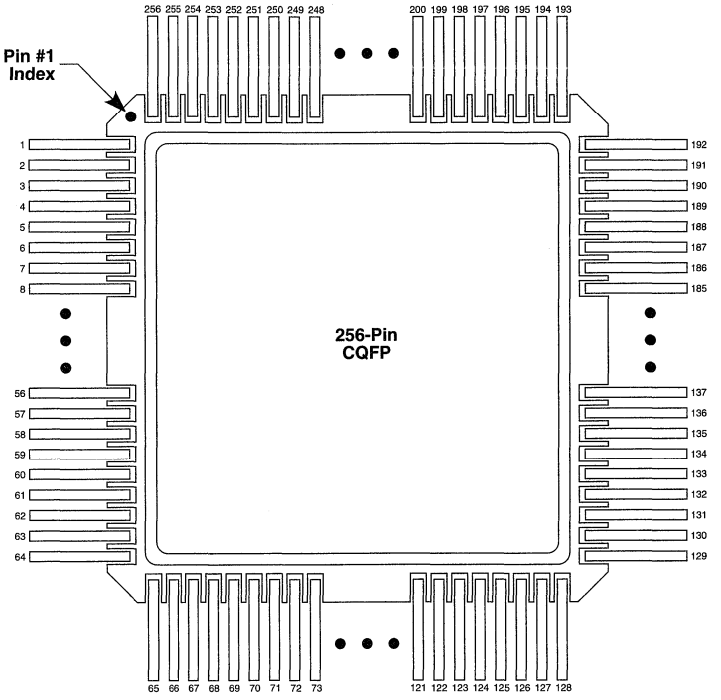
Signal	PIN Number
CLKA or I/O	172
CLKB or I/O	173
DCLK or I/O	196
GND	1, 13, 37, 51, 52, 64, 86, 98, 101, 112, 139, 149, 162, 183, 193
HCLK or I/O	77
IOCLK or I/O	148
IOPCL or I/O	100
MODE	11
PRA or I/O	174
PRB or I/O	75
SDI or I/O	2
V_{CC}	12, 38, 59, 94, 111, 140, 155, 189
V_{KS}	138
V_{PP}	137
V_{SV}	39, 110

Notes:

1. *Unused I/O pins are designated as outputs by ALS and are driven low.*
2. *All unassigned pins are available for use as I/Os.*
3. *MODE = GND, except during device programming or debugging.*
4. *$V_{PP} = V_{CC}$, except during device programming.*
5. *$V_{SV} = V_{CC}$, except during device programming*
6. *V_{KS} = GND, except during device programming.*

Package Pin Assignments (continued)

256-Pin CQFP (Top View)



Signal	PIN Number
CLKA or I/O	219
CLKB or I/O	220
DCLK or I/O	256
GND	1, 29, 31, 59, 91, 93, 110, 128, 158, 160, 176, 189, 222, 224, 240
HCLK or I/O	96
IOCLK or I/O	188
IOPCL or I/O	127
MODE	11
PRA or I/O	225
PRB or I/O	90
SDI or I/O	2
V_{CC}	28, 30, 92, 94, 159, 161, 221, 223
V_{KS}	175
V_{PP}	174
V_{SV}	141

Notes:

1. *Unused I/O pins are designated as outputs by ALS and are driven low.*
2. *All unassigned pins are available for use as I/Os.*
3. *MODE = GND, except during device programming or debugging.*
4. *$V_{PP} = V_{CC}$, except during device programming.*
5. *$V_{SV} = V_{CC}$, except during device programming*
6. *V_{KS} = GND, except during device programming.*

Military
Field Programmable Gate Arrays

Features

- Highly Predictable Performance with 100 Percent Automatic Placement and Routing
- Device Sizes from 1200 to 10,000 gates (up to 25,000 PLD equivalent gates)
- Up to 4, Fast, Low-Skew Clock Networks
- Up to 228 User-Programmable I/O Pins
- More Than 500 Macro Functions
- Replaces up to 250 TTL Packages
- Replaces up to 100 20-pin PAL Packages
- Up to 1153 Dedicated Flip-Flops
- I/O Drive to 10 mA
- Devices Available to DESC SMD
- CQFP and CPGA Packaging
- Nonvolatile, User Programmable
- Logic Fully Tested Prior to Shipment

ACT 3 Features

- Highest-Performance, Highest-Capacity FPGA Family
- System Performance to 60 MHz over Military Temperature
- Low-Power 0.8-micron CMOS Technology

ACT 2 Features

- Best-Value, High-Capacity FPGA Family
- System Performance to 40 MHz over Military Temperature
- Low-Power 1.0-micron CMOS Technology

ACT 1 Features

- Lowest-Cost FPGA Family
- System Performance to 20 MHz over Military Temperature
- Low-Power 1.0-micron CMOS Technology

Product Family Profile

Family	ACT 3			ACT 2		ACT 1	
Device	**A1425A**	**A1460A**	**A14100A**	**A1240A**	**A1280A**	**A1010B**	**A1020B**
Capacity							
Gate Array Equivalent Gates	2500	6000	10,000	4000	8000	1200	2000
PLD Equivalent Gates	6250	15,000	25,000	10,000	20,000	3000	6000
TTL Equivalent Packages (40 gates)	60	150	250	100	200	30	50
20-Pin PAL Equivalent Packages (100 gates)	25	60	100	40	80	12	20
Logic Modules	310	848	1377	684	1232	295	547
S-Modules	160	432	697	348	624	—	—
C-Modules	150	416	680	336	608	295	547
Flip-Flops (maximum)	435	976	1493	568	998	147	273
User I/Os (maximum)	100	168	228	104	140	57	69
Packages[1] (by pin count)							
CPGA	133	207	257	132	176	84	84
CQFP	132	196	256	—	172	—	84
Performance							
System Speed (maximum)	60 MHz	60 MHz	60 MHz	40 MHz	40 MHz	20 MHz	20 MHz

Note:

1. See Product Plan on page 1-291 for package availability.

High-Reliability, Low-Risk Solution

Actel builds the most reliable field programmable gate arrays (FPGAs) in the industry, with overall antifuse reliability ratings of less than 10 Failures-In-Time (FITs), corresponding to a useful life of more than 40 years. Actel FGPAs have been production proven, with more than five million devices shipped and more than one trillion antifuses manufactured. Actel devices are fully tested prior to shipment, with an outgoing defect level of only 122 ppm. (Further reliability data is available in the "Actel Reliability Report.")

100 Percent Tested

Device functionality is fully tested before shipment and during device programming. Routing tracks, logic modules, and programming, debug, and test circuits are 100 percent tested before shipment. Antifuse integrity also is tested before shipment. Programming algorithms are tested when a device is programmed using Actel's Activator® 2 or Activator 2S programming stations.

Benefits

No Cost Risk—Once you have a Designer/Designer Advantage™ System, Actel's CAE software and programming package, you can produce as many chips as you like for just the cost of the device itself, with no NRE charges to eat up your development budget every time you want to try out a new design.

No Time Risk—After entering your design, placement and routing is automatic, and programming the device takes only about 5 to 15 minutes for an average design. You save time in the design entry process by using tools that are familiar to you. The Action Logic System software interfaces with popular CAE packages such as Cadence, Mentor Graphics, OrCAD, and Viewlogic, running on platforms such as HP, Sun, and PC. In addition, synthesis capability is provided with support of synthesis tools from Synopsys, IST, Exemplar, and DATA I/O.

No Reliability Risk—The PLICE® antifuse is a one-time programmable, nonvolatile connection. Since Actel devices are permanently programmed, no downloading from EPROM or SRAM storage is required. Inadvertent erasure is impossible, and there is no need to reload the program after power disruptions. Both the PLICE antifuses and the base process are radiation tolerant. Fabrication using a low-power CMOS process means cooler junction temperatures. Actel's non-PLD architecture delivers lower dynamic operating current. Our reliability tests show a very low failure rate of 66 FITs at 90°C junction temperature with no degradation in AC performance. Special stress testing at wafer test eliminates infant mortalities prior to packaging.

No Security Risk—Reverse engineering of programmed Actel devices from optical or electrical data is extremely difficult. Programmed antifuses cannot be identified from a photograph or by using a SEM. The antifuse map cannot be deciphered either electrically or by microprobing. Each device has a silicon signature that identifies its origins, down to the wafer lot and fabrication facility.

No Testing Risk—Unprogrammed Actel parts are fully tested at the factory. This includes the logic modules, interconnect tracks, and I/Os. AC performance is ensured by special speed path tests, and programming circuitry is verified on test antifuses. During the programming process, an algorithm is run to ensure that all antifuses are correctly programmed. In addition, Actel's Actionprobe® diagnostic tools allow 100 percent observability of all internal nodes to check and debug your design.

Actel FPGA Description

The Actel families of FPGAs offer a variety of packages, speed/performance characteristics, and processing levels for use in all high-reliability and military applications. Devices are implemented in a silicon gate, two-level metal CMOS process, utilizing Actel's PLICE antifuse technology. This unique architecture offers gate array flexibility, high performance, and quick turnaround through user programming. Device utilization is typically 95 percent of available logic modules.

Actel devices also provide system designers with on-chip diagnostic probe/debug capability, allowing the user to observe 100 percent of the nodes within the design, even while the device is operating in-system. All Actel devices include on-chip clock drivers and a hard-wired distribution network.

User-definable I/Os are capable of driving at both TTL and CMOS drive levels. Available packages for the military are the Ceramic Quad Flat Pack (CQFP) and the Ceramic Pin Grid Array (CPGA). See Product Plan on page 2-293 for details.

All Actel FPGAs are supported by the Actel Designer Series, which offers automatic or user-definable pin assignment, validation of electrical and design rules, automatic placement and routing, timing analysis, user programming, and debug/diagnostic probe capabilities. The Designer Series fully supports schematic capture and backannotated simulation through design kits for Cadence, Mentor Graphics, OrCAD, and Viewlogic. Synthesis is supported with kits for use with synthesis tools from Synopsys, IST, Exemplar, and DATA I/O.

Also available is an FPGA fitter (ACTmap) that provides logic synthesis and optimization from PAL language or VHDL description inputs. An FPGA macro generator (ACTgen) is provided, allowing the user easily to create higher-level functions such as counters and adders. Finally, ChipEdit is a graphical/visual design tool that allows the user to modify the automatic place and route results.

ACT 3 Description

The ACT 3 family is the third-generation Actel FPGA family. This family offers the highest-performance and highest-capacity devices, ranging from 2,500 to 10,000 gates, with system performance to 60 MHz over the military temperature range. The devices have four clock distribution networks, including dedicated array and I/O clocks. In addition, the ACT 3 family offers the highest I/O-to-gate ratio available. ACT 3 devices are manufactured using 0.8 micron CMOS technology.

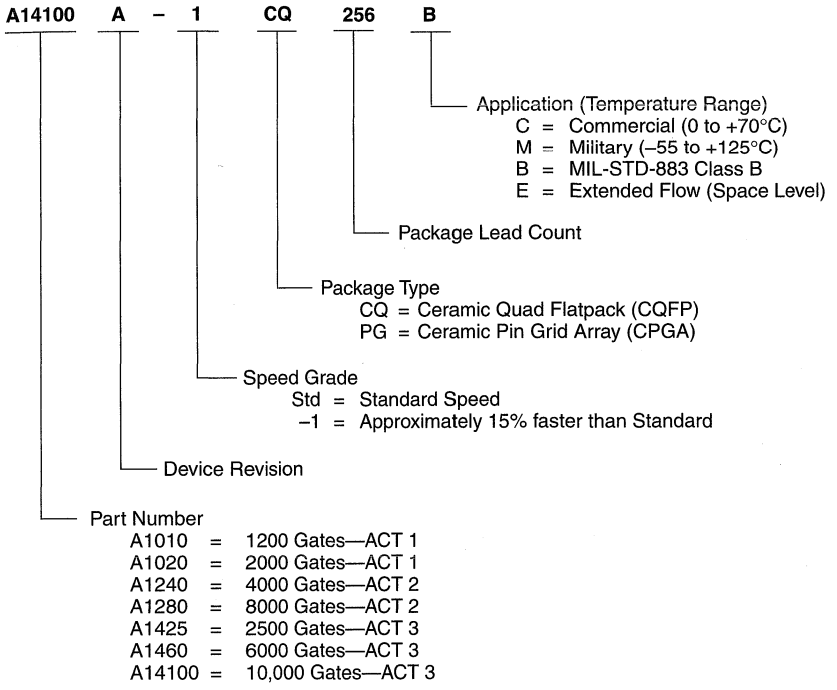
ACT 2 Description

The ACT 2 family is the second-generation Actel FPGA family. This family offers the best-value, high-capacity devices, ranging from 4,000 to 8,000 gates, with system performance to 40 MHz over the military temperature range. The devices have two routed array clock distribution networks. ACT 2 devices are manufactured using 1.0 micron CMOS technology.

ACT 1 Description

The ACT 1 family is the first Actel FPGA family and the first antifuse-based FPGA. This family offers the lowest-cost logic integration, with devices ranging from 1,200 to 2,000 gates, with system performance to 20 MHz over the military temperature range. The devices have one routed array clock distribution network. ACT 1 devices are manufactured using 1.0 micron CMOS technology.

Military Device Ordering Information



DESC SMD/Actel Part Number Cross-Reference

Actel Part Number (Gold Leads)	DESC SMD (Gold Leads)	DESC SMD (Solder Dipped)
A1010B-PG84B	5962-9096403MXC	5962-9096403MXA
A1010B-1PG84B	5962-9096404MXC	5962-9096404MXA
A1020B-PG84B	5962-9096503MUC	5962-9096503MUA
A1020B-1PG84B	5962-9096504MUC	5962-9096504MUA
A1020B-CQ84B	5962-9096503MTC	5962-9096503MTA
A1020B-1CQ84B	5962-9096504MTC	5962-9096504MTA
A1240A-PG132B	5962-9322101MXC	5962-9322101MXA
A1240A-1PG132B	5962-9322102MXC	5962-9322102MXA
A1280A-PG176B	5962-9215601MXC	5962-9215601MXA
A1280A-1PG176B	5962-9215602MXC	5962-9215602MXA
A1280A-CQ172B	5962-9215601MYC	5962-9215601MYA
A1280A-1CQ172B	5962-9215602MYC	5962-9215602MYA
A1425A	TBD	TBD
A1460A	TBD	TBD
A14100A	TBD	TBD

Product Plan

ACT 3 Family	Speed Grade		Application			
	Std	–1	C	M	B	E
A1425A Device						
132-pin Ceramic Quad Flatpack (CQFP)	✔	✔	✔	✔	✔	—
133-pin Ceramic Pin Grid Array (CPGA)	✔	✔	✔	✔	✔	—
A1460A Device						
196-pin Ceramic Quad Flatpack (CQFP)	✔	P	✔	P	P	—
207-pin Ceramic Pin Grid Array (CPGA)	✔	P	✔	P	P	—
A14100A Device						
256-pin Ceramic Quad Flatpack (CQFP)	✔	P	✔	P	P	—
257-pin Ceramic Pin Grid Array (CPGA)	✔	P	✔	P	P	—

ACT 2 Family	Speed Grade		Application			
	Std	–1	C	M	B	E
A1240A Device						
132-pin Ceramic Pin Grid Array (CPGA)	✔	✔	✔	✔	✔	—
A1280A Device						
172-pin Ceramic Quad Flatpack (CQFP)	✔	✔	✔	✔	✔	✔
176-pin Ceramic Pin Grid Array (CPGA)	✔	✔	✔	✔	✔	✔

ACT 1 Family	Speed Grade		Application			
	Std	–1	C	M	B	E
A1010B Device						
84-pin Ceramic Pin Grid Array (CPGA)	✔	✔	✔	✔	✔	—
A1020B Device						
84-pin Ceramic Quad Flatpack (CQFP)	✔	✔	✔	✔	✔	✔
84-pin Ceramic Pin Grid Array (CPGA)	✔	✔	✔	✔	✔	✔

Applications: C = Commercial
M = Military
B = MIL-STD-883
E = Extended Flow

Availability: ✔ = Available Now
P = Planned
— = Not Planned

* Speed Grade: –1 = Approx. 15% faster than Standard

ACT 3 Device Resources

FPGA Device Type	Logic Modules	Gate Array Equivalent Gates	User I/Os					
			CQFP			CPGA		
			132-pin	196-pin	256-pin	133-pin	207-pin	257-pin
A1425A	310	2500	100	—	—	100	—	—
A1460A	848	6000	—	168	—	—	168	—
A14100A	1377	10,000	—	—	228	—	—	228

ACT 2 Device Resources

FPGA Device Type	Logic Modules	Gate Array Equivalent Gates	User I/Os		
			CQFP	CPGA	
			172-pin	132-pin	176-pin
A1240A	684	4000	—	104	—
A1280A	1232	8000	140	—	140

ACT 1 Device Resources

FPGA Device Type	Logic Modules	Gate Array Equivalent Gates	User I/Os	
			CQFP	CPGA
			84-pin	84-pin
A1010B	295	1200	—	57
A1020B	547	2000	69	69

Pin Description

CLK Clock (Input)

ACT 1 only. TTL Clock input for global clock distribution network. The Clock input is buffered prior to clocking the logic modules. This pin can also be used as an I/O.

CLKA Clock A (Input)

ACT 2 and ACT 3 only. TTL Clock input for global clock distribution networks. The Clock input is buffered prior to clocking the logic modules. This pin can also be used as an I/O.

CLKB Clock B (Input)

ACT 2 and ACT 3 only. TTL Clock input for global clock distribution networks. The Clock input is buffered prior to clocking the logic modules. This pin can also be used as an I/O.

DCLK Diagnostic Clock (Input)

TTL Clock input for diagnostic probe and device programming. DCLK is active when the MODE pin is HIGH. This pin functions as an I/O when the MODE pin is LOW.

GND Ground

LOW supply voltage.

HCLK Dedicated (Hard-wired) Array Clock (Input)

ACT 3 only. TTL Clock input for sequential modules. This input is directly wired to each S-module and offers clock speeds independent of the number of S-modules being driven. This pin can also be used as an I/O.

I/O Input/Output (Input, Output)

I/O pin functions as an input, output, tristate, or bidirectional buffer. Input and output levels are compatible with standard TTL and CMOS specifications. Unused I/O pins are automatically driven LOW.

IOCLK Dedicated (Hard-wired) I/O Clock (Input)

ACT 3 only. TTL Clock input for I/O modules. This input is directly wired to each I/O module and offers clock speeds independent of the number of I/O modules being driven. This pin can also be used as an I/O.

IOPCL Dedicated (Hard-wired) I/O Preset/Clear (Input)

ACT 3 only. TTL input for I/O preset or clear. This global input is directly wired to the preset and clear inputs of all I/O registers. This pin functions as an I/O when no I/O preset or clear macros are used.

MODE Mode (Input)

The MODE pin controls the use of diagnostic pins (DCLK, PRA, PRB, SDI). When the MODE pin is HIGH, the special functions are active. When the MODE pin is LOW, the pins function as I/Os.

NC No Connection

This pin is not connected to circuitry within the device.

PRA Probe A (Output)

The Probe A pin is used to output data from any user-defined design node within the device. This independent diagnostic pin can be used in conjunction with the Probe B pin to allow real-time diagnostic output of any signal path within the device. The Probe A pin can be used as a user-defined I/O when debugging has been completed. The pin's probe capabilities can be permanently disabled to protect programmed design confidentiality. PRA is accessible when the MODE pin is HIGH. This pin functions as an I/O when the MODE pin is LOW.

PRB Probe B (Output)

The Probe B pin is used to output data from any user-defined design node within the device. This independent diagnostic pin can be used in conjunction with the Probe A pin to allow real-time diagnostic output of any signal path within the device. The Probe B pin can be used as a user-defined I/O when debugging has been completed. The pin's probe capabilities can be permanently disabled to protect programmed design confidentiality. PRB is accessible when the MODE pin is HIGH. This pin functions as an I/O when the MODE pin is LOW.

SDI Serial Data Input (Input)

Serial data input for diagnostic probe and device programming. SDI is active when the MODE pin is HIGH. This pin functions as an I/O when the MODE pin is LOW.

V_{CC} 5V Supply Voltage

HIGH supply voltage.

V_{KS} Programming Voltage

ACT 2 and ACT 3 only. Supply voltage used for device programming. This pin must be connected to GND during normal operation.

V_{PP} Programming Voltage

Supply voltage used for device programming. This pin must be connected to V_{CC} during normal operation.

V_{SV} Programming Voltage

ACT 2 and ACT 3 only. Supply voltage used for device programming. This pin must be connected to V_{CC} during normal operation.

Actel Military Product Flow

Step	Screen	833—Class B 833 Method	833—Class B Requirement	Military Datasheet Requirement
1.0	Internal Visual	2010, Test Condition B	100%	100%
2.0	Temperature Cycling	1010, Test Condition C	100%	100%
3.0	Constant Acceleration	2001, Test Condition E (min), Y1, Orientation Only	100%	100%
4.0	Seal	1014		
	a. Fine		100%	100%
	b. Gross		100%	100%
5.0	Visual Inspection	2009	100%	100%
6.0	Pre-burn-in Electrical Parameters	In accordance with Actel applicable device specification	100%	N/A
7.0	Burn-in Test	1015 Condition D 160 hours @ 125°C Min.	100%	N/A
8.0	Interim (Post-burn-in) Electrical Parameters	In accordance with Actel applicable device specification	100%	100% (as final test)
9.0	Percent Defective Allowable	5%	All Lots	N/A
10.0	Final Electrical Test	In accordance with Actel applicable device specification		
	a. Static Tests (1) 25°C (Subgroup 1, Table I, 5005) (2) –55°C and +125°C (Subgroups 2, 3, Table I, 5005)		100%	100%
	b. Dynamic and Functional Tests (1) 25°C (Subgroup 7, Table I, 5005) (2) –55°C and +125°C (Subgroups 8A and 8B, Table I, 5005)		100%	100%
	c. Switching Tests at 25°C (Subgroup 9, Table I, 5005)		100%	100%
11.0	Qualification or Quality Confirmation Inspection Test Sample Selection (Group A)	5005	All Lots	N/A
12.0	External Visual	2009	100%	Actel specification

Actel Extended Flow[1, 2]

	Screen	Method	Requirement
1.	Wafer Lot Acceptance[3]	5007 with step coverage waiver	All Lots
2.	Destructive In-Line Bond Pull[4]	2011, condition D	Sample
3.	Internal Visual	2010, condition A	100%
4.	Serialization		100%
5.	Temperature Cycling	1010, condition C	100%
6.	Constant Acceleration	2001, condition E (min), Y_1 orientation only	100%
7.	Visual Inspection	2009	100%
8.	Particle Impact Noise Detection	2020, condition A	100%
9.	Radiographic	2012	100%
10.	Pre-burn-in Test	In accordance with Actel applicable device specification	100%
11.	Burn-in Test	1015, condition D, 240 hours @ 125°C minimum	100%
12.	Interim (Post-burn-in) Electrical Parameters	In accordance with Actel applicable device specification	100%
13.	Reverse Bias Burn-in	1010, condition C, 72 hours @ 150°C minimum	100%
14.	Interim (Post-burn-in) Electrical Parameters	In accordance with Actel applicable device specification	100%
15.	Percent Defective Allowable (PDA) Calculation	5%, 3% functional parameters @ 25°C	All Lots
16.	Final Electrical Test	In accordance with Actel applicable device specification	100%
	a. Static Tests		100%
	(1) 25°C (Subgroup 1, Table1)	5005	
	(2) –55°C and +125°C (Subgroups 2, 3, Table 1)	5005	
	b. Dynamic and Functional Tests		100%
	(1) 25°C (Subgroup 7, Table 15)	5005	
	(2) –55°C and +125°C (Subgroups 5 and 6, 8a and b, Table 1)	5005	
	c. Switching Tests at 25°C (Subgroup 9, Table I, 5005)	5005	100%
17.	Seal	1014	100%
	a.Fine		
	b.Gross		
18.	Qualification or Quality Conformance Inspection Test Sample Selection	5005	Group A & Group B
19	External Visual	2009	100%

Note:

1. *Actel offers the Extended Flow in order to satisfy those customers that require additional screening beyond the requirements of MIL-STD-883, Class B. Actel is compliant to the requirements of MIL-STD-883, Paragraph 1.2.1, and MIL-I-38535, Appendix A. Actel is offering this extended flow incorporating the majority of the screening procedures as outlined in Method 5004 of MIL-STD-883 Class S. The exceptions to Method 5004 are shown in notes 2 to 4 below.*
2. *Method 5004 requires a 100 percent Radiation latch-up testing to Method 1020. Actel will not be performing any radiation testing, and this requirement must be waived in its entirety.*
3. *Wafer lot acceptance is performed to Method 5007; however the step coverage requirement as specified in Method 2018 must be waived.*
4. *Method 5004 requires a 100 percent, nondestructive bond pull to Method 2023. Actel substitutes a destructive bond pull to Method 2011, condition D on a sample basis only.*

Absolute Maximum Ratings[1]

Free air temperature range

Symbol	Parameter	Limits	Units
V_{CC}	DC Supply Voltage[2, 3, 4]	–0.5 to +7.0	V
V_I	Input Voltage	–0.5 to V_{CC} +0.5	V
V_O	Output Voltage	–0.5 to V_{CC} +0.5	V
I_{IO}	I/O Source Sink Current[5]	±20	mA
T_{STG}	Storage Temperature	–65 to +150	°C

Notes:

1. *Stresses beyond those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. Exposure to absolute maximum rated conditions for extended periods may affect device reliability. Device should not be operated outside the recommended operating conditions.*
2. *$V_{PP} = V_{CC}$, except during device programming.*
3. *$V_{SV} = V_{CC}$, except during device programming.*
4. *V_{KS} = GND, except during device programming.*
5. *Device inputs are normally high impedance and draw extremely low current. However, when input voltage is greater than V_{CC} + 0.5 V or less than GND – 0.5 V, the internal protection diode will be forward biased and can draw excessive current.*

Recommended Operating Conditions

Parameter	Commercial	Military	Units
Temperature Range[1]	0 to +70	–55 to +125	°C
Power Supply Tolerance	±5	±10	%V_{CC}

Note:

1. *Ambient temperature (T_A) is used for commercial and industrial; case temperature (T_C) is used for military.*

Package Thermal Characteristics

The device junction to case thermal characteristic is θjc, and the junction to ambient air characteristic is θja. The thermal characteristics for θja are shown with two different air flow rates.

Maximum junction temperature is 150°C.

A sample calculation of the absolute maximum power dissipation allowed for a CPGA 176-pin package at military temperature is as follows:

$$\frac{\text{Max. junction temp. (°C)} - \text{Max. military temp.}}{\theta ja\ (°C/W)} = \frac{150°C - 125°C}{23°C/W} = 1.1\ W$$

Package Type	Pin Count	θja Still Air	θja 300 ft/min	Units
Ceramic Pin Grid Array	84	33	20	°C/W
	132	26	16	°C/W
	133	37	24	°C/W
	176	23	12	°C/W
	207	22	14	°C/W
	257	21	13	°C/W
Ceramic Quad Flatpack	84	40	25	°C/W
	132	55	30	°C/W
	172	25	15	°C/W
	196	36	24	°C/W
	256	30	18	°C/W

Electrical Specifications

Symbol	Parameter	Test Condition	Commercial		Military		Units
			Min.	Max.	Min.	Max.	
V_{OH}[1, 2]	HIGH Level Output	I_{OH} = –4 mA (CMOS)			3.7		V
		I_{OH} = –6 mA (CMOS)	3.84				V
V_{OL}[1, 2]	LOW Level Output	I_{OL} = +6 mA (CMOS)		0.33		0.4	V
V_{IH}	HIGH Level Input	TTL Inputs	2.0	V_{CC} + 0.3	2.0	V_{CC} + 0.3	V
V_{IL}	LOW Level Input	TTL Inputs	–0.3	0.8	–0.3	0.8	V
I_{IN}	Input Leakage	V_I = V_{CC} or GND	–10	+10	–10	+10	µA
I_{OZ}	3-state Output Leakage	V_O = V_{CC} or GND	–10	+10	–10	+10	µA
C_{IO}	I/O Capacitance[3, 4]			10		10	pF
$I_{CC(S)}$	Standby V_{CC} Supply Current	V_I = V_{CC} or GND,					
		I_O = 0 mA		2		20	mA
		ACT 1		3		20	mA
		ACT 2/3		2		20	mA
$I_{CC(D)}$	Dynamic V_{CC} Supply Current	See "Power Dissipation" Section					

Notes:

1. *Actel devices can drive and receive either CMOS or TTL signal levels. No assignment of I/Os as TTL or CMOS is required.*
2. *Tested one output at a time, V_{CC} = min.*
3. *Not tested; for information only.*
4. *V_{OUT} = 0V, f = 1 MHz*

General Power Equation

$$P = [I_{CC}standby + I_{CC}active] * V_{CC} + I_{OL} * V_{OL} * N + I_{OH} * (V_{CC} - V_{OH}) * M$$

Where:

I_{CC}standby is the current flowing when no inputs or outputs are changing.

I_{CC}active is the current flowing due to CMOS switching.

I_{OL}, I_{OH} are TTL sink/source currents.

V_{OL}, V_{OH} are TTL level output voltages.

N equals the number of outputs driving TTL loads to V_{OL}.

M equals the number of outputs driving TTL loads to V_{OH}.

An accurate determination of N and M is problematical because their values depend on the family type, on design details, and on the system I/O. The power can be divided into two components—static and active.

Static Power Component

Actel FPGAs have small static power components that result in power dissipation lower than that of PALs or PLDs. By integrating multiple PALs or PLDs into one FPGA, an even greater reduction in board-level power dissipation can be achieved.

The power due to standby current is typically a small component of the overall power. Standby power is calculated below for commercial, worst-case conditions.

Family	I_{CC}	V_{CC}	Power
ACT 1	3 mA	5.25 V	15.8 mW
ACT 2	2 mA	5.25 V	10.5 mW
ACT 3	2 mA	5.25 V	10.5 mW

The static power dissipated by TTL loads depends on the number of outputs driving high or low and the DC load current. Again, this value is typically small. For instance, a 32-bit bus sinking 4 mA at 0.33 V will generate 42 mW with all outputs driving low, and 140 mW with all outputs driving high.

Active Power Component

Power dissipation in CMOS devices is usually dominated by the active (dynamic) power dissipation. This component is frequency dependent, a function of the logic and the external I/O. Active power dissipation results from charging internal chip capacitances of the interconnect, unprogrammed antifuses, module inputs, and module outputs, plus external capacitance due to PC board traces and load device inputs. An additional component of the active power dissipation is the totem-pole current in CMOS transistor pairs. The net effect can be associated with an equivalent capacitance that

can be combined with frequency and voltage to represent active power dissipation.

Equivalent Capacitance

The power dissipated by a CMOS circuit can be expressed by the equation 1

$$\text{Power (uW)} = C_{EQ} * V_{CC}^2 * F \quad (1)$$

Where:

C_{EQ} is the equivalent capacitance expressed in pF.

V_{CC} is the power supply in volts.

F is the switching frequency in MHz.

Equivalent capacitance is calculated by measuring I_{CC}active at a specified frequency and voltage for each circuit component of interest. Measurements are made over a range of frequencies at a fixed value of V_{CC}. Equivalent capacitance is frequency independent so that the results can be used over a wide range of operating conditions. Equivalent capacitance values are shown below.

CEQ Values for Actel FPGAs

	ACT 1	ACT 2	ACT 3
Modules (C_{EQM})	3.7	5.8	6.7
Input Buffers (C_{EQI})	22.1	12.9	7.2
Output Buffers (C_{EQO})	31.2	23.8	10.4
Routed Array Clock Buffer Loads (C_{EQCR})	4.6	3.9	1.6
Dedicated Clock Buffer Loads (C_{EQCD})	n/a	n/a	0.7
I/O Clock Buffer Loads (C_{EQCI})	n/a	n/a	0.9

To calculate the active power dissipated from the complete design, the switching frequency of each part of the logic must be known. Equation 2 shows a piecewise linear summation over all components, it applies to all ACT 1, ACT 2, and ACT 3 devices. Since the ACT 1 family has only one routed array clock, the terms labeled routed_Clk2, dedicated_Clk, and IO_Clk do not apply. Similarly, the ACT 2 family has two routed array clocks, and the dedicated_Clk and IO_Clk terms do not apply. For ACT 3 devices, all terms will apply.

$$\text{Power} = V_{CC}^2 * [(m * C_{EQM} * f_m)_{modules} + (n * C_{EQI} * f_n)_{inputs} + (p * (C_{EQO} + C_L) * f_p)_{outputs} + 0.5 * (q_1 * C_{EQCR} * f_{q1})_{routed\_Clk1} + (r_1 * f_{q1})_{routed\_Clk1} + 0.5 * (q_2 * C_{EQCR} * f_{q2})_{routed\_Clk2} + (r_2 * f_{q2})_{routed\_Clk2} + 0.5 * (s_1 * C_{EQCD} * f_{s1})_{dedicated\_Clk} + (s_2 * C_{EQCI} * f_{s2})_{IO\_Clk}] \quad (2)$$

Where:

m = Number of logic modules switching at fm

n = Number of input buffers switching at fn

p = Number of output buffers switching at fp

q_1 = Number of clock loads on the first routed array clock (all families)

q_2 = Number of clock loads on the second routed array clock (ACT 2, 3 only)

r_1 = Fixed capacitance due to first routed array clock (all families)

r_2 = Fixed capacitance due to second routed array clock (ACT 2, 3 only)

s_1 = Fixed number of clock loads on the dedicated array clock (ACT 3 only)

s_2 = Fixed number of clock loads on the dedicated I/O clock (ACT 3 only)

C_{EQM} = Equivalent capacitance of logic modules in pF

C_{EQI} = Equivalent capacitance of input buffers in pF

C_{EQO} = Equivalent capacitance of output buffers in pF

C_{EQCR} = Equivalent capacitance of routed array clock in pF

C_{EQCD} = Equivalent capacitance of dedicated array clock in pF

C_{EQCI} = Equivalent capacitance of dedicated I/O clock in pF

C_L = Output lead capacitance in pF

f_m = Average logic module switching rate in MHz

f_n = Average input buffer switching rate in MHz

f_p = Average output buffer switching rate in MHz

f_{q1} = Average first routed array clock rate in MHz (all families)

f_{q2} = Average second routed array clock rate in MHz (ACT 2, 3 only)

f_{s1} = Average dedicated array clock rate in MHz (ACT 3 only)

f_{s2} = Average dedicated I/O clock rate in MHz (ACT 3 only)

Fixed Capacitance Values for Actel FPGAs (pF)

Device Type	r_1 routed_Clk1	r_2 routed_Clk2
A1010B	41	n/a
A1020B	69	n/a
A1240A	134	134
A1280A	168	168
A1425A	75	75
A1460A	165	165
A14100A	195	195

Fixed Clock Loads (s_1/s_2—ACT 3 Only)

Device Type	s_1 Clock Loads on Dedicated Array Clock	s_2 Clock Loads on Dedicated I/O Clock
A1425A	160	100
A1460A	432	168
A14100A	697	228

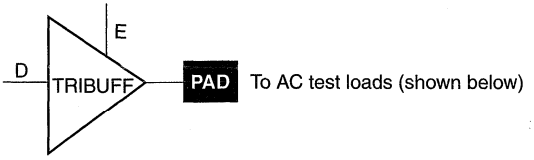
Determining Average Switching Frequency

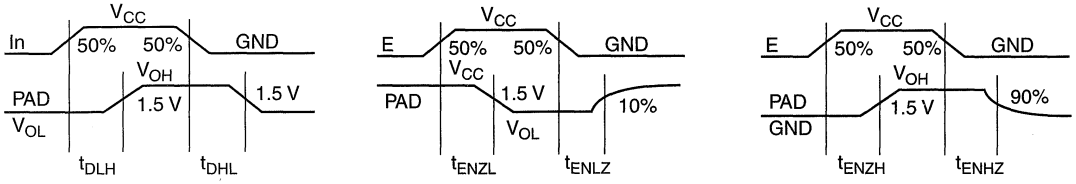
To determine the switching frequency for a design, you must have a detailed understanding of the data values input to the circuit. The guidelines in the table below are meant to represent worst-case scenarios so that they can be generally used to predict the upper limits of power dissipation.

Type	ACT 1	ACT 2	ACT 3
Logic modules (m)	90% of modules	80% of modules	80% of modules
Input switching (n)	# inputs/4	# inputs/4	# inputs/4
Outputs switching (p)	#outputs/4	#outputs/4	#outputs/4
First routed array clock loads (q_1)	40% of modules	40% of sequential modules	40% of sequential modules
Second routed array clock loads (q_2)	n/a	40% of sequential modules	40% of sequential modules
Load capacitance (C_L)	35 pF	35 pF	35 pF
Average logic module switching rate (f_m)	F/10	F/10	F/10
Average input switching rate (f_n)	F/5	F/5	F/5
Average output switching rate (f_p)	F/10	F/10	F/10
Average first routed array clock rate (f_{q1})	F	F	F/2
Average second routed array clock rate (f_{q2})	n/a	F/2	F/2
Average dedicated array clock rate (f_{s1})	n/a	n/a	F
Average dedicated I/O clock rate (f_{s2})	n/a	n/a	F

Parameter Measurement

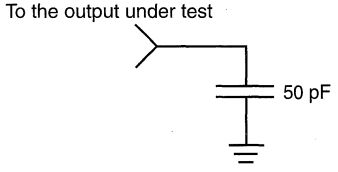
Output Buffer Delays



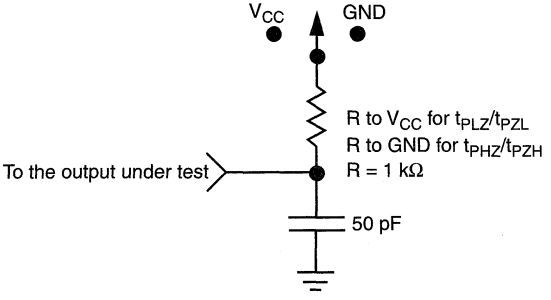


AC Test Load

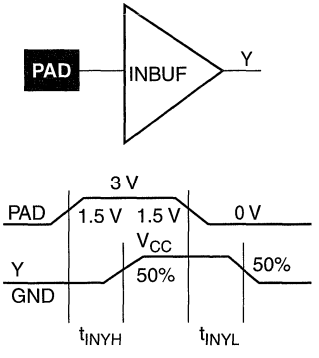
Load 1
(Used to measure propagation delay)



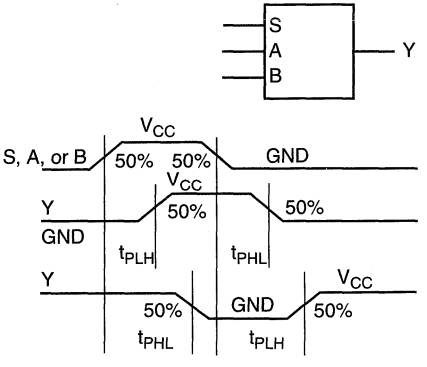
Load 2
(Used to measure rising/falling edges)



Input Buffer Delays

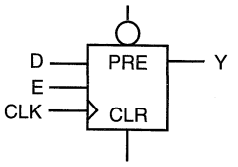


Combinatorial Macro Delays

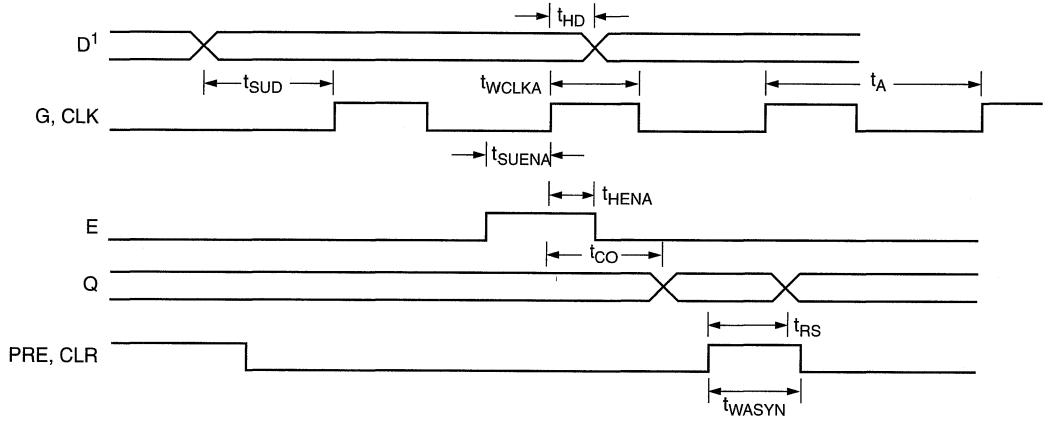


Sequential Timing Characteristics

Flip-Flops and Latches (ACT 1 and ACT 2)



(Positive edge triggered)

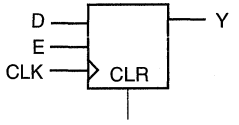


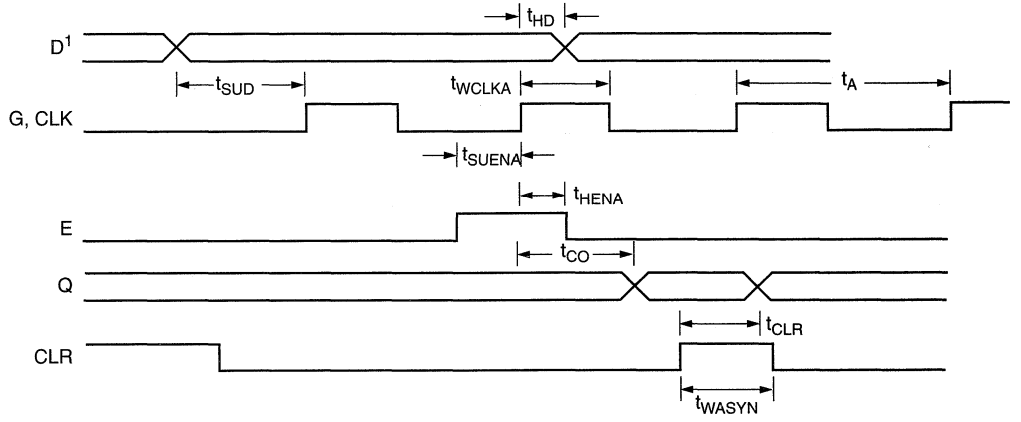
Note:

1. *D represents all data functions involving A, B, and S for multiplexed flip-flops.*

Sequential Timing Characteristics

Flip-Flops and Latches (ACT 3)



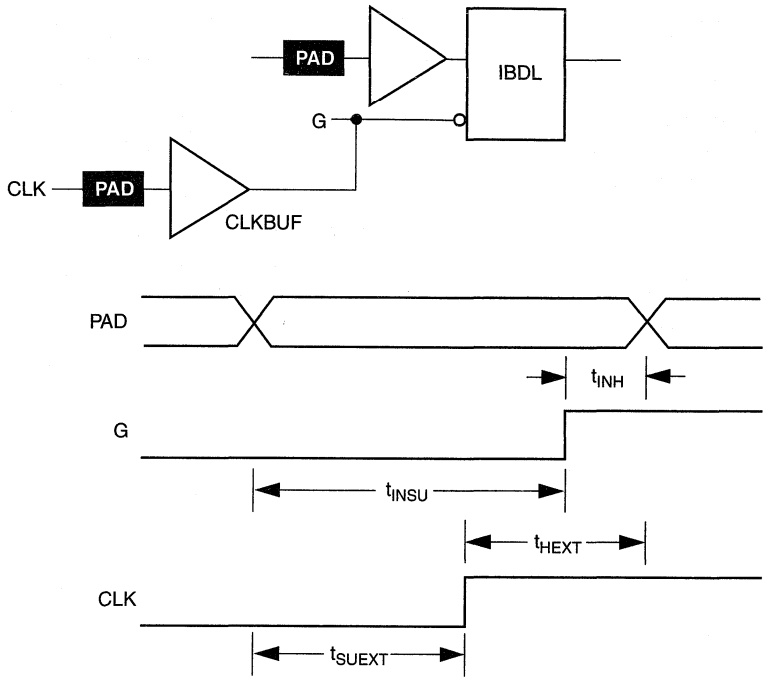


Note:

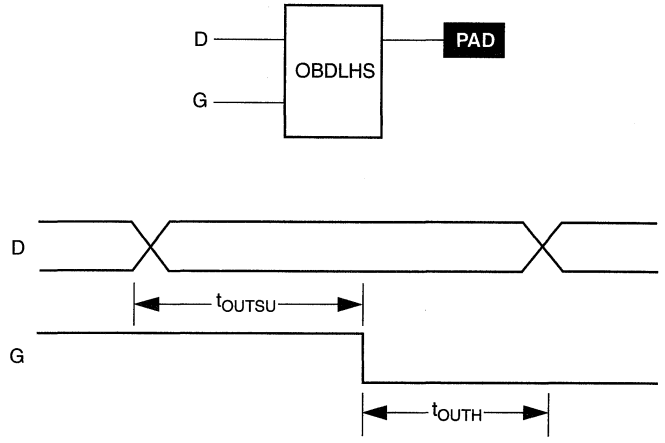
1. *D represents all data functions involving A, B, and S for multiplexed flip-flops.*

Sequential Timing Characteristics (continued)

Input Buffer Latches (ACT 2 only)



Output Buffer Latches (ACT 2 only)



ACT 1 Timing Characteristics

(Worst-Case Military Conditions)

Logic Module Propagation Delays		Std Speed		–1 Speed		
Parameter	**Description**	**Min.**	**Max.**	**Min.**	**Max.**	**Units**
t_{PD1}	Single Module		5.5		4.7	ns
t_{PD2}	Dual Module Macros		12.7		10.8	ns
t_{CO}	Sequential Clk to Q		5.5		4.7	ns
t_{GO}	Latch G to Q		5.5		4.7	ns
t_{RS}	Flip-Flop (Latch) Reset to Q		5.5		4.7	ns
Predicted Routing Delays[1]						
t_{RD1}	FO=1 Routing Delay		1.7		1.5	ns
t_{RD2}	FO=2 Routing Delay		2.7		2.3	ns
t_{RD3}	FO=3 Routing Delay		4.0		3.4	ns
t_{RD4}	FO=4 Routing Delay		5.9		5.0	ns
t_{RD8}	FO=8 Routing Delay		12.5		10.6	ns
Sequential Timing Characteristics[2]						
t_{SUD}	Flip-Flop (Latch) Data Input Setup	10.4		8.8		ns
t_{HD}	Flip-Flop (Latch) Data Input Hold	0.0		0.0		ns
t_{SUENA}	Flip-Flop (Latch) Enable Setup	10.4		8.8		ns
t_{HENA}	Flip-Flop (Latch) Enable Hold	0.0		0.0		ns
t_{WCLKA}	Flip-Flop (Latch) Clock Active Pulse Width	12.9		10.9		ns
t_{WASYN}	Flip-Flop (Latch) Asynchronous Pulse Width	12.9		10.9		ns
t_A	Flip-Flop Clock Input Period	27.3		23.2		ns
f_{MAX}	Flip-Flop (Latch) Clock Frequency		37		44	MHz

Notes:

1. *Routing delays are for typical designs across worst-case operating conditions. These parameters should be used for estimating device performance. Post-route timing analysis or simulation is required to determine actual worst-case performance. Post-route timing is based on actual routing delay measurements performed on the device prior to shipment.*
2. *Setup times assume fanout of 3. Further derating information can be obtained from the ALS Timer utility.*

ACT 1 Timing Characteristics (continued)

(Worst-Case Military Conditions)

Input Module Propagation Delays			Std Speed		–1 Speed		
Parameter	**Description**		**Min.**	**Max.**	**Min.**	**Max.**	**Units**
t_{INYH}	Pad to Y High			5.8		4.9	ns
t_{INYL}	Pad to Y Low			5.8		4.9	ns
Input Module Predicted Routing Delays[1]							
t_{IRD1}	FO=1 Routing Delay			1.7		1.5	ns
t_{IRD2}	FO=2 Routing Delay			2.7		2.3	ns
t_{IRD3}	FO=3 Routing Delay			4.0		3.4	ns
t_{IRD4}	FO=4 Routing Delay			5.9		5.0	ns
t_{IRD8}	FO=8 Routing Delay			12.5		10.6	ns
Global Clock Network							
t_{CKH}	Input Low to High	FO = 16		9.2		7.8	ns
		FO = 128		10.5		8.9	
t_{CKL}	Input High to Low	FO = 16		12.1		10.3	ns
		FO = 128		13.2		11.2	
t_{PWH}	Minimum Pulse Width High	FO = 16	12.2		10.4		ns
		FO = 128	12.9		10.9		
t_{PWL}	Minimum Pulse Width Low	FO = 16	12.2		10.4		ns
		FO = 128	12.9		10.9		
t_{CKSW}	Maximum Skew	FO = 16		2.2		1.9	ns
		FO = 128		3.4		2.9	
t_P	Minimum Period	FO = 16	25.6		21.7		ns
		FO = 128	27.3		23.2		
f_{MAX}	Maximum Frequency	FO = 16		40		46	MHz
		FO = 128		37		44	

Note:

1. *These parameters should be used for estimating device performance. Routing delays are for typical designs across worst-case operating conditions. Post-route timing analysis or simulation is required to determine actual worst-case performance. Post-route timing is based on actual routing delay measurements performed on the device prior to shipment.*

ACT 1 Timing Characteristics (continued)

(Worst-Case Military Conditions)

Output Module Timing		Std Speed		–1 Speed		
Parameter	**Description**	**Min.**	**Max.**	**Min.**	**Max.**	**Units**
TTL Output Module Timing[1]						
t_{DLH}	Data to Pad High		14.2		12.1	ns
t_{DHL}	Data to Pad Low		16.3		13.8	ns
t_{ENZH}	Enable Pad Z to High		14.1		12.0	ns
t_{ENZL}	Enable Pad Z to Low		17.1		14.6	ns
t_{ENHZ}	Enable Pad High to Z		18.8		16.0	ns
t_{ENLZ}	Enable Pad Low to Z		17.0		14.5	ns
d_{TLH}	Delta Low to High		0.11		0.09	ns/pF
d_{THL}	Delta High to Low		0.15		0.12	ns/pF
CMOS Output Module Timing[1]						
t_{DLH}	Data to Pad High		17.7		15.1	ns
t_{DHL}	Data to Pad Low		13.6		11.5	ns
t_{ENZH}	Enable Pad Z to High		14.1		12.0	ns
t_{ENZL}	Enable Pad Z to Low		17.1		14.6	ns
t_{ENHZ}	Enable Pad High to Z		18.8		16.0	ns
t_{ENLZ}	Enable Pad Low to Z		17.0		14.5	ns
d_{TLH}	Delta Low to High		0.18		0.16	ns/pF
d_{THL}	Delta High to Low		0.11		0.09	ns/pF

Note:

2. Delays based on 50 pF loading.

A1240A Timing Characteristics

(Worst-Case Military Conditions)

Logic Module Propagation Delays[1]		Std Speed		–1 Speed		
Parameter	**Description**	**Min.**	**Max.**	**Min.**	**Max.**	**Units**
t_{PD1}	Single Module		6.1		5.2	ns
t_{CO}	Sequential Clk to Q		6.1		5.2	ns
t_{GO}	Latch G to Q		6.1		5.2	ns
t_{RS}	Flip-Flop (Latch) Reset to Q		6.1		5.2	ns
Predicted Routing Delays[2]						
t_{RD1}	FO=1 Routing Delay		2.2		1.9	ns
t_{RD2}	FO=2 Routing Delay		2.8		2.4	ns
t_{RD3}	FO=3 Routing Delay		3.7		3.1	ns
t_{RD4}	FO=4 Routing Delay		5.0		4.3	ns
t_{RD8}	FO=8 Routing Delay		7.7		6.6	ns
Sequential Timing Characteristics[3, 4]						
t_{SUD}	Flip-Flop (Latch) Data Input Setup	0.5		0.5		ns
t_{HD}	Flip-Flop (Latch) Data Input Hold	0.0		0.0		ns
t_{SUENA}	Flip-Flop (Latch) Enable Setup	1.3		1.3		ns
t_{HENA}	Flip-Flop (Latch) Enable Hold	0.0		0.0		ns
t_{WCLKA}	Flip-Flop (Latch) Clock Active Pulse Width	8.1		7.4		ns
t_{WASYN}	Flip-Flop (Latch) Asynchronous Pulse Width	8.1		7.4		ns
t_{A}	Flip-Flop Clock Input Period	18.6		14.8		ns
t_{INH}	Input Buffer Latch Hold	2.5		2.5		ns
t_{INSU}	Input Buffer Latch Setup	–3.5		–3.5		ns
t_{OUTH}	Output Buffer Latch Hold	0.0		0.0		ns
t_{OUTSU}	Output Buffer Latch Setup	0.5		0.5		ns
f_{MAX}	Flip-Flop (Latch) Clock Frequency		54		63	MHz

Notes:

1. *For dual-module macros, use $t_{PD1} + t_{RD1} + t_{PDn}$, $t_{CO} + t_{RD1} + t_{PDn}$, or $t_{PD1} + t_{RD1} + t_{SUD}$, whichever is appropriate.*
2. *Routing delays are for typical designs across worst-case operating conditions. These parameters should be used for estimating device performance. Post-route timing analysis or simulation is required to determine actual worst-case performance. Post-route timing is based on actual routing delay measurements performed on the device prior to shipment.*
3. *Data applies to macros based on the S-module. Timing parameters for sequential macros constructed from C-modules can be obtained from the ALS Timer utility.*
4. *Setup and hold timing parameters for the Input Buffer Latch are defined with respect to the PAD and the D input. External setup/hold timing parameters must account for delay from an external PAD signal to the G inputs. Delay from an external PAD signal to the G input subtracts (adds) to the internal setup (hold) time.*

A1240A Timing Characteristics (continued)

(Worst-Case Military Conditions)

Input Module Propagation Delays			Std Speed		–1 Speed		
Parameter	**Description**		**Min.**	**Max.**	**Min.**	**Max.**	**Units**
t_{INYH}	Pad to Y High			4.7		4.0	ns
t_{INYL}	Pad to Y Low			4.3		3.6	ns
t_{INGH}	G to Y High			8.1		6.9	ns
t_{INGL}	G to Y Low			7.7		6.6	ns
Input Module Predicted Routing Delays[1]							
t_{IRD1}	FO=1 Routing Delay			6.9		5.8	ns
t_{IRD2}	FO=2 Routing Delay			7.8		6.7	ns
t_{IRD3}	FO=3 Routing Delay			8.8		7.5	ns
t_{IRD4}	FO=4 Routing Delay			9.7		8.2	ns
t_{IRD8}	FO=8 Routing Delay			12.9		10.9	ns
Global Clock Network							
t_{CKH}	Input Low to High	FO = 32 FO = 256		15.7 19.2		13.3 16.3	ns
t_{CKL}	Input High to Low	FO = 32 FO = 256		15.7 19.5		13.3 16.5	ns
t_{PWH}	Minimum Pulse Width High	FO = 32 FO = 256	6.7 7.1		5.7 6.0		ns
t_{PWL}	Minimum Pulse Width Low	FO = 32 FO = 256	6.7 7.1		5.7 6.0		ns
t_{CKSW}	Maximum Skew	FO = 32 FO = 256		0.6 3.1		0.6 3.1	ns
t_{SUEXT}	Input Latch External Setup	FO = 32 FO = 256	0.0 0.0		0.0 0.0		ns
t_{HEXT}	Input Latch External Hold	FO = 32 FO = 256	8.6 13.8		8.6 13.8		ns
t_P	Minimum Period	FO = 32 FO = 256	13.5 14.3		11.5 12.2		ns
f_{MAX}	Maximum Frequency	FO = 32 FO = 256		74 70		87 82	MHz

Note:

1. *These parameters should be used for estimating device performance. Optimization techniques may further reduce delays by 0 to 4 ns. Routing delays are for typical designs across worst-case operating conditions. Post-route timing analysis or simulation is required to determine actual worst-case performance. Post-route timing is based on actual routing delay measurements performed on the device prior to shipment.*

A1240A Timing Characteristics (continued)

(Worst-Case Military Conditions)

Output Module Timing		Std Speed		–1 Speed		
Parameter	**Description**	**Min.**	**Max.**	**Min.**	**Max.**	**Units**
TTL Output Module Timing[1]						
t_{DLH}	Data to Pad High		13.0		11.0	ns
t_{DHL}	Data to Pad Low		16.4		13.9	ns
t_{ENZH}	Enable Pad Z to High		14.4		12.3	ns
t_{ENZL}	Enable Pad Z to Low		19.0		16.1	ns
t_{ENHZ}	Enable Pad High to Z		11.5		9.8	ns
t_{ENLZ}	Enable Pad Low to Z		13.6		11.5	ns
t_{GLH}	G to Pad High		14.6		12.4	ns
t_{GHL}	G to Pad Low		18.2		15.5	ns
d_{TLH}	Delta Low to High		0.11		0.09	ns/pF
d_{THL}	Delta High to Low		0.20		0.17	ns/pF
CMOS Output Module Timing[1]						
t_{DLH}	Data to Pad High		16.5		14.0	ns
t_{DHL}	Data to Pad Low		13.7		11.7	ns
t_{ENZH}	Enable Pad Z to High		14.4		12.3	ns
t_{ENZL}	Enable Pad Z to Low		19.0		16.1	ns
t_{ENHZ}	Enable Pad High to Z		11.5		9.8	ns
t_{ENLZ}	Enable Pad Low to Z		13.6		11.5	ns
t_{GLH}	G to Pad High		14.6		12.4	ns
t_{GHL}	G to Pad Low		18.2		15.5	ns
d_{TLH}	Delta Low to High		0.20		0.17	ns/pF
d_{THL}	Delta High to Low		0.15		0.12	ns/pF

Note:

1. Delays based on 50 pF loading.

A1280A Timing Characteristics

(Worst-Case Military Conditions)

Logic Module Propagation Delays[1]		Std Speed		–1 Speed		
Parameter	**Description**	**Min.**	**Max.**	**Min.**	**Max.**	**Units**
t_{PD1}	Single Module		6.1		5.2	ns
t_{CO}	Sequential Clk to Q		6.1		5.2	ns
t_{GO}	Latch G to Q		6.1		5.2	ns
t_{RS}	Flip-Flop (Latch) Reset to Q		6.1		5.2	ns
Predicted Routing Delays[2]						
t_{RD1}	FO=1 Routing Delay		2.8		2.4	ns
t_{RD2}	FO=2 Routing Delay		4.0		3.4	ns
t_{RD3}	FO=3 Routing Delay		4.9		4.2	ns
t_{RD4}	FO=4 Routing Delay		6.0		5.1	ns
t_{RD8}	FO=8 Routing Delay		10.8		9.2	ns
Sequential Timing Characteristics[3, 4]						
t_{SUD}	Flip-Flop (Latch) Data Input Setup	0.5		0.5		ns
t_{HD}	Flip-Flop (Latch) Data Input Hold	0.0		0.0		ns
t_{SUENA}	Flip-Flop (Latch) Enable Setup	1.3		1.3		ns
t_{HENA}	Flip-Flop (Latch) Enable Hold	0.0		0.0		ns
t_{WCLKA}	Flip-Flop (Latch) Clock Active Pulse Width	8.6		7.4		ns
t_{WASYN}	Flip-Flop (Latch) Asynchronous Pulse Width	8.6		7.4		ns
t_A	Flip-Flop Clock Input Period	22.1		16.4		ns
t_{INH}	Input Buffer Latch Hold	2.5		2.5		ns
t_{INSU}	Input Buffer Latch Setup	–3.5		–3.5		ns
t_{OUTH}	Output Buffer Latch Hold	0.0		0.0		ns
t_{OUTSU}	Output Buffer Latch Setup	0.5		0.5		ns
f_{MAX}	Flip-Flop (Latch) Clock Frequency		41		60	MHz

Notes:

1. *For dual-module macros, use $t_{PD1} + t_{RD1} + t_{PDn}$, $t_{CO} + t_{RD1} + t_{PDn}$, or $t_{PD1} + t_{RD1} + t_{SUD}$, whichever is appropriate.*
2. *Routing delays are for typical designs across worst-case operating conditions. These parameters should be used for estimating device performance. Post-route timing analysis or simulation is required to determine actual worst-case performance. Post-route timing is based on actual routing delay measurements performed on the device prior to shipment.*
3. *Data applies to macros based on the S-module. Timing parameters for sequential macros constructed from C-modules can be obtained from the ALS Timer utility.*
4. *Setup and hold timing parameters for the input buffer latch are defined with respect to the PAD and the D input. External setup/hold timing parameters must account for delay from an external PAD signal to the G inputs. Delay from an external PAD signal to the G input subtracts (adds) to the internal setup (hold) time.*

A1280A Timing Characteristics (continued)

(Worst-Case Military Conditions)

Input Module Propagation Delays			Std Speed		–1 Speed		
Parameter	**Description**		**Min.**	**Max.**	**Min.**	**Max.**	**Units**
t_{INYH}	Pad to Y High			4.7		4.0	ns
t_{INYL}	Pad to Y Low			4.3		3.6	ns
t_{INGH}	G to Y High			8.1		6.9	ns
t_{INGL}	G to Y Low			7.7		6.6	ns
Input Module Predicted Routing Delays[1]							
t_{RD1}	FO=1 Routing Delay			7.3		6.2	ns
t_{RD2}	FO=2 Routing Delay			8.4		7.2	ns
t_{RD3}	FO=3 Routing Delay			9.1		7.7	ns
t_{RD4}	FO=4 Routing Delay			10.5		8.9	ns
t_{RD8}	FO=8 Routing Delay			15.2		12.9	ns
Global Clock Network							
t_{CKH}	Input Low to High	FO = 32 FO = 384		15.7 21.1		13.3 17.9	ns
t_{CKL}	Input High to Low	FO = 32 FO = 384		15.7 21.4		13.3 18.2	ns
t_{PWH}	Minimum Pulse Width High	FO = 32 FO = 384	8.1 9.3		6.9 7.9		ns
t_{PWL}	Minimum Pulse Width Low	FO = 32 FO = 384	8.1 9.3		6.9 7.9		ns
t_{CKSW}	Maximum Skew	FO = 32 FO = 384		0.6 3.1		0.6 3.1	ns
t_{SUEXT}	Input Latch External Setup	FO = 32 FO = 384	0.0 0.0		0.0 0.0		ns
t_{HEXT}	Input Latch External Hold	FO = 32 FO = 384	8.6 13.8		8.6 13.8		ns
t_P	Minimum Period	FO = 32 FO = 384	16.2 18.9		13.7 16.0		ns
f_{MAX}	Maximum Frequency	FO = 32 FO = 384		62 53		73 63	MHz

Note:

1. *These parameters should be used for estimating device performance. Optimization techniques may further reduce delays by 0 to 4 ns. Routing delays are for typical designs across worst-case operating conditions. Post-route timing analysis or simulation is required to determine actual worst-case performance. Post-route timing is based on actual routing delay measurements performed on the device prior to shipment.*

A1280A Timing Characteristics (continued)

(Worst-Case Military Conditions)

Output Module Timing		Std Speed		–1 Speed		
Parameter	**Description**	**Min.**	**Max.**	**Min.**	**Max.**	**Units**
TTL Output Module Timing[1]						
t_{DLH}	Data to Pad High		13.0		11.0	ns
t_{DHL}	Data to Pad Low		16.4		13.9	ns
t_{ENZH}	Enable Pad Z to High		14.4		12.3	ns
t_{ENZL}	Enable Pad Z to Low		19.0		16.1	ns
t_{ENHZ}	Enable Pad High to Z		11.5		9.8	ns
t_{ENLZ}	Enable Pad Low to Z		13.6		11.5	ns
t_{GLH}	G to Pad High		14.6		12.4	ns
t_{GHL}	G to Pad Low		18.2		15.5	ns
d_{TLH}	Delta Low to High		0.11		0.09	ns/pF
d_{THL}	Delta High to Low		0.20		0.17	ns/pF
CMOS Output Module Timing[1]						
t_{DLH}	Data to Pad High		16.5		14.0	ns
t_{DHL}	Data to Pad Low		13.7		11.7	ns
t_{ENZH}	Enable Pad Z to High		14.4		12.3	ns
t_{ENZL}	Enable Pad Z to Low		19.0		16.1	ns
t_{ENHZ}	Enable Pad High to Z		11.5		9.8	ns
t_{ENLZ}	Enable Pad Low to Z		13.6		11.5	ns
t_{GLH}	G to Pad High		14.6		12.4	ns
t_{GHL}	G to Pad Low		18.2		15.5	ns
d_{TLH}	Delta Low to High		0.20		0.17	ns/pF
d_{THL}	Delta High to Low		0.15		0.12	ns/pF

Note:

1. Delays based on 50 pF loading.

A1425A Timing Characteristics

(Worst-Case Military Conditions)

Logic Module Propagation Delays[1]		Std Speed		–1 Speed		
Parameter	**Description**	**Min.**	**Max.**	**Min.**	**Max.**	**Units**
t_{PD}	Internal Array Module		3.5		3.0	ns
t_{CO}	Sequential Clock to Q		3.5		3.0	ns
t_{CLR}	Asynchronous Clear to Q		3.5		3.0	ns
Predicted Routing Delays[2]						
t_{RD1}	FO=1 Routing Delay		1.5		1.3	ns
t_{RD2}	FO=2 Routing Delay		2.1		1.9	ns
t_{RD3}	FO=3 Routing Delay		2.5		2.1	ns
t_{RD4}	FO=4 Routing Delay		2.9		2.6	ns
t_{RD8}	FO=8 Routing Delay		4.9		4.2	ns
Logic Module Sequential Timing						
t_{SUD}	Flip-Flop (Latch) Data Input Setup	1.0		0.9		ns
t_{HD}	Flip-Flop (Latch) Data Input Hold	0.0		0.0		ns
t_{SUENA}	Flip-Flop (Latch) Enable Setup	1.0		0.9		ns
t_{HENA}	Flip-Flop (Latch) Enable Hold	0.0		0.0		ns
t_{WASYN}	Asynchronous Pulse Width	4.4		3.8		ns
t_{WCLKA}	Flip-Flop Clock Pulse Width	4.4		3.8		ns
t_A	Flip-Flop Clock Input Period	9.3		7.9		ns
f_{MAX}	Flip-Flop Clock Frequency		100		125	MHz

Notes:

1. *For dual-module macros, use $t_{PD} + t_{RD1} + t_{PDn}$, $t_{CO} + t_{RD1} + t_{PDn}$, or $t_{PD1} + t_{RD1} + t_{SUD}$, whichever is appropriate.*
2. *Routing delays are for typical designs across worst-case operating conditions. These parameters should be used for estimating device performance. Post-route timing analysis or simulation is required to determine actual worst-case performance. Post-route timing is based on actual routing delay measurements performed on the device prior to shipment.*

A1425A Timing Characteristics (continued)

(Worst-Case Military Conditions)

I/O Module Input Propagation Delays		Std Speed		–1 Speed		
Parameter	**Description**	**Min.**	**Max.**	**Min.**	**Max.**	**Units**
t_{INY}	Input Data Pad to Y		4.9		4.2	ns
t_{ICKY}	Input Reg IOCLK Pad to Y		8.2		7.0	ns
t_{OCKY}	Output Reg IOCLK Pad to Y		8.2		7.0	ns
t_{ICLRY}	Input Asynchronous Clear to Y		8.2		7.0	ns
t_{OCLRY}	Output Asynchronous Clear to Y		8.2		7.0	ns
Predicted Input Routing Delays[1]						
t_{IRD1}	FO=1 Routing Delay		1.5		1.3	ns
t_{IRD2}	FO=2 Routing Delay		2.1		1.9	ns
t_{IRD3}	FO=3 Routing Delay		2.5		2.1	ns
t_{IRD4}	FO=4 Routing Delay		2.9		2.6	ns
t_{IRD8}	FO=8 Routing Delay		4.9		4.2	ns
I/O Module Sequential Timing						
t_{INH}	Input F-F Data Hold (w.r.t. IOCLK Pad)	0.0		0.0		ns
t_{INSU}	Input F-F Data Setup (w.r.t. IOCLK Pad)	2.4		2.1		ns
t_{IDEH}	Input Data Enable Hold (w.r.t. IOCLK Pad)	0.0		0.0		ns
t_{IDESU}	Input Data Enable Setup (w.r.t. IOCLK Pad)	10.0		8.7		ns
t_{OUTH}	Output F-F Data Hold (w.r.t. IOCLK Pad)	1.2		1.1		ns
t_{OUTSU}	Output F-F Data Setup (w.r.t. IOCLK Pad)	1.2		1.1		ns
t_{ODEH}	Output Data Enable Hold (w.r.t. IOCLK Pad)	0.6		0.5		ns
t_{ODESU}	Output Data Enable Setup (w.r.t. IOCLK Pad)	2.4		2.0		ns

Note:

1. *Routing delays are for typical designs across worst-case operating conditions. These parameters should be used for estimating device performance. Post-route timing analysis or simulation is required to determine actual worst-case performance. Post-route timing is based on actual routing delay measurements performed on the device prior to shipment.*

A1425A Timing Characteristics (continued)

(Worst-Case Military Conditions)

I/O Module – TTL Output Timing[1]		Std Speed		–1 Speed		
Parameter	**Description**	**Min.**	**Max.**	**Min.**	**Max.**	**Units**
t_{DHS}	Data to Pad, High Slew		8.9		7.5	ns
t_{DLS}	Data to Pad, Low Slew		14.0		11.9	ns
t_{ENZHS}	Enable to Pad, Z to H/L, Hi Slew		7.0		6.0	ns
t_{ENZLS}	Enable to Pad, Z to H/L, Lo Slew		12.8		10.9	ns
t_{ENHSZ}	Enable to Pad, H/L to Z, Hi Slew		11.6		9.9	ns
t_{ENLSZ}	Enable to Pad, H/L to Z, Lo Slew		11.6		9.9	ns
t_{CKHS}	IOCLK Pad to Pad H/L, Hi Slew		11.6		10.5	ns
t_{CKLS}	IOCLK Pad to Pad H/L, Lo Slew		17.4		15.7	ns
d_{TLHHS}	Delta Low to High, Hi Slew		0.04		0.04	ns/pF
d_{TLHLS}	Delta Low to High, Lo Slew		0.08		0.07	ns/pF
d_{THLHS}	Delta High to Low, Hi Slew		0.06		0.05	ns/pF
d_{THLLS}	Delta High to Low, Lo Slew		0.08		0.07	ns/pF
I/O Module – CMOS Output Timing[1]						
t_{DHS}	Data to Pad, High Slew		10.8		9.2	ns
t_{DLS}	Data to Pad, Low Slew		20.3		17.3	ns
t_{ENZHS}	Enable to Pad, Z to H/L, Hi Slew		9.1		7.7	ns
t_{ENZLS}	Enable to Pad, Z to H/L, Lo Slew		15.5		13.1	ns
t_{ENHSZ}	Enable to Pad, H/L to Z, Hi Slew		11.6		9.9	ns
t_{ENLSZ}	Enable to Pad, H/L to Z, Lo Slew		11.6		10.5	ns
t_{CKHS}	IOCLK Pad to Pad H/L, Hi Slew		13.7		12.5	ns
t_{CKLS}	IOCLK Pad to Pad H/L, Lo Slew		20.1		18.1	ns
d_{TLHHS}	Delta Low to High, Hi Slew		0.07		0.06	ns/pF
d_{TLHLS}	Delta Low to High, Lo Slew		0.13		0.11	ns/pF
d_{THLHS}	Delta High to Low, Hi Slew		0.05		0.04	ns/pF
d_{THLLS}	Delta High to Low, Lo Slew		0.06		0.05	ns/pF

Note:

1. Delays based on 35 pF loading.

A1425A Timing Characteristics (continued)

(Worst-Case Military Conditions)

Dedicated (Hard-Wired) I/O Clock Network		Std Speed		–1 Speed		
Parameter	**Description**	**Min.**	**Max.**	**Min.**	**Max.**	**Units**
t_{IOCKH}	Input Low to High (Pad to I/O Module Input)		3.5		3.0	ns
t_{IOPWH}	Minimum Pulse Width High	4.4		3.9		ns
t_{IOPWL}	Minimum Pulse Width Low	4.4		3.9		ns
t_{IOSAPW}	Minimum Asynchronous Pulse Width	4.4		3.9		ns
t_{IOCKSW}	Maximum Skew		0.5		0.5	ns
t_{IOP}	Minimum Period	9.3		7.9		ns
f_{IOMAX}	Maximum Frequency		100		125	MHz
Dedicated (Hard-Wired) Array Clock Network						
t_{HCKH}	Input Low to High (Pad to S-Module Input)		5.3		4.6	ns
t_{HCKL}	Input High to Low (Pad to S-Module Input)		5.3		4.6	ns
t_{HPWH}	Minimum Pulse Width High	4.4		3.9		ns
t_{HPWL}	Minimum Pulse Width Low	4.4		3.9		ns
t_{HCKSW}	Maximum Skew		0.4		0.4	ns
t_{HP}	Minimum Period	9.3		7.9		ns
f_{HMAX}	Maximum Frequency		100		125	MHz
Routed Array Clock Networks						
t_{RCKH}	Input Low to High (FO=64)		6.4		5.5	ns
t_{RCKL}	Input High to Low (FO=64)		7.0		6.0	ns
t_{RPWH}	Min. Pulse Width High (FO=64)	5.7		4.9		ns
t_{RPWL}	Min. Pulse Width Low (FO=64)	5.7		4.9		ns
t_{RCKSW}	Maximum Skew (FO=128)		1.2		1.1	ns
t_{RP}	Minimum Period (FO=64)	11.6		10.1		ns
f_{RMAX}	Maximum Frequency (FO=64)		85		100	MHz
Clock-to-Clock Skews						
$t_{IOHCKSW}$	I/O Clock to H-Clock Skew	0.0	3.0	0.0	3.0	ns
$t_{IORCKSW}$	I/O Clock to R-Clock Skew	0.0	3.0	0.0	3.0	ns
t_{HRCKSW}	H-Clock to R-Clock Skew					
	(FO = 64)	0.0	1.0	0.0	1.0	ns
	(FO = 50% max.)	0.0	3.0	0.0	3.0	ns

Note:

1. Delays based on 35 pF loading.

A1460A Timing Characteristics

(Worst-Case Military Conditions)

Logic Module Propagation Delays[1]		Std Speed		–1 Speed		
Parameter	**Description**	**Min.**	**Max.**	**Min.**	**Max.**	**Units**
t_{PD}	Internal Array Module		3.5		3.0	ns
t_{CO}	Sequential Clock to Q		3.5		3.0	ns
t_{CLR}	Asynchronous Clear to Q		3.5		3.0	ns
Predicted Routing Delays[2]						
t_{RD1}	FO=1 Routing Delay		1.5		1.3	ns
t_{RD2}	FO=2 Routing Delay		2.1		1.9	ns
t_{RD3}	FO=3 Routing Delay		2.5		2.1	ns
t_{RD4}	FO=4 Routing Delay		2.9		2.6	ns
t_{RD8}	FO=8 Routing Delay		4.9		4.2	ns
Logic Module Sequential Timing						
t_{SUD}	Flip-Flop (Latch) Data Input Setup	1.0		0.9		ns
t_{HD}	Flip-Flop (Latch) Data Input Hold	0.0		0.0		ns
t_{SUENA}	Flip-Flop (Latch) Enable Setup	1.0		0.9		ns
t_{HENA}	Flip-Flop (Latch) Enable Hold	0.0		0.0		ns
t_{WASYN}	Asynchronous Pulse Width	5.6		4.8		ns
t_{WCLKA}	Flip-Flop Clock Pulse Width	5.6		4.8		ns
t_{A}	Flip-Flop Clock Input Period	11.6		9.9		ns
f_{MAX}	Flip-Flop Clock Frequency		85		100	MHz

Notes:

1. *For dual-module macros, use $t_{PD} + t_{RD1} + t_{PDm}$, $t_{CO} + t_{RD1} + t_{PDm}$, or $t_{PD1} + t_{RD1} + t_{SUD}$, whichever is appropriate.*
2. *Routing delays are for typical designs across worst-case operating conditions. These parameters should be used for estimating device performance. Post-route timing analysis or simulation is required to determine actual worst-case performance. Post-route timing is based on actual routing delay measurements performed on the device prior to shipment.*

A1460A Timing Characteristics (continued)

(Worst-Case Military Conditions)

I/O Module Input Propagation Delays		Std Speed		–1 Speed		
Parameter	**Description**	**Min.**	**Max.**	**Min.**	**Max.**	**Units**
t_{INY}	Input Data Pad to Y		4.9		4.2	ns
t_{ICKY}	Input Reg IOCLK Pad to Y		8.2		7.0	ns
t_{OCKY}	Output Reg IOCLK Pad to Y		8.2		7.0	ns
t_{ICLRY}	Input Asynchronous Clear to Y		8.2		7.0	ns
t_{OCLRY}	Output Asynchronous Clear to Y		8.2		7.0	ns
Predicted Input Routing Delays[1]						
t_{IRD1}	FO=1 Routing Delay		1.5		1.3	ns
t_{IRD2}	FO=2 Routing Delay		2.1		1.9	ns
t_{IRD3}	FO=3 Routing Delay		2.5		2.1	ns
t_{IRD4}	FO=4 Routing Delay		2.9		2.6	ns
t_{IRD8}	FO=8 Routing Delay		4.9		4.2	ns
I/O Module Sequential Timing						
t_{INH}	Input F-F Data Hold (w.r.t. IOCLK Pad)	0.0		0.0		ns
t_{INSU}	Input F-F Data Setup (w.r.t. IOCLK Pad)	2.4		2.1		ns
t_{IDEH}	Input Data Enable Hold (w.r.t. IOCLK Pad)	0.0		0.0		ns
t_{IDESU}	Input Data Enable Setup (w.r.t. IOCLK Pad)	10.0		8.7		ns
t_{OUTH}	Output F-F Data Hold (w.r.t. IOCLK Pad)	1.2		1.1		ns
t_{OUTSU}	Output F-F Data Setup (w.r.t. IOCLK Pad)	1.2		1.1		ns
t_{ODEH}	Output Data Enable Hold (w.r.t. IOCLK Pad)	0.6		0.5		ns
t_{ODESU}	Output Data Enable Setup (w.r.t. IOCLK Pad)	2.4		2.0		ns

Note:

1. *Routing delays are for typical designs across worst-case operating conditions. These parameters should be used for estimating device performance. Post-route timing analysis or simulation is required to determine actual worst-case performance. Post-route timing is based on actual routing delay measurements performed on the device prior to shipment.*

A1460A Timing Characteristics (continued)

(Worst-Case Military Conditions)

I/O Module – TTL Output Timing[1]		Std Speed		–1 Speed		
Parameter	**Description**	**Min.**	**Max.**	**Min.**	**Max.**	**Units**
t_{DHS}	Data to Pad, High Slew		8.9		7.5	ns
t_{DLS}	Data to Pad, Low Slew		14.0		11.9	ns
t_{ENZHS}	Enable to Pad, Z to H/L, Hi Slew		7.0		6.0	ns
t_{ENZLS}	Enable to Pad, Z to H/L, Lo Slew		12.8		10.9	ns
t_{ENHSZ}	Enable to Pad, H/L to Z, Hi Slew		13.5		11.5	ns
t_{ENLSZ}	Enable to Pad, H/L to Z, Lo Slew		12.8		10.9	ns
t_{CKHS}	IOCLK Pad to Pad H/L, Hi Slew		13.4		11.6	ns
t_{CKLS}	IOCLK Pad to Pad H/L, Lo Slew		19.8		17.8	ns
d_{TLHHS}	Delta Low to High, Hi Slew		0.04		0.04	ns/pF
d_{TLHLS}	Delta Low to High, Lo Slew		0.08		0.07	ns/pF
d_{THLHS}	Delta High to Low, Hi Slew		0.06		0.05	ns/pF
d_{THLLS}	Delta High to Low, Lo Slew		0.08		0.07	ns/pF
I/O Module – CMOS Output Timing[1]						
t_{DHS}	Data to Pad, High Slew		10.8		9.2	ns
t_{DLS}	Data to Pad, Low Slew		20.3		17.3	ns
t_{ENZHS}	Enable to Pad, Z to H/L, Hi Slew		9.1		7.7	ns
t_{ENZLS}	Enable to Pad, Z to H/L, Lo Slew		15.5		13.1	ns
t_{ENHSZ}	Enable to Pad, H/L to Z, Hi Slew		12.8		10.9	ns
t_{ENLSZ}	Enable to Pad, H/L to Z, Lo Slew		12.8		10.9	ns
t_{CKHS}	IOCLK Pad to Pad H/L, Hi Slew		16.0		14.1	ns
t_{CKLS}	IOCLK Pad to Pad H/L, Lo Slew		22.4		20.2	ns
d_{TLHHS}	Delta Low to High, Hi Slew		0.07		0.06	ns/pF
d_{TLHLS}	Delta Low to High, Lo Slew		0.13		0.11	ns/pF
d_{THLHS}	Delta High to Low, Hi Slew		0.05		0.04	ns/pF
d_{THLLS}	Delta High to Low, Lo Slew		0.06		0.05	ns/pF

Note:

1. Delays based on 35 pF loading.

A1460A Timing Characteristics (continued)

(Worst-Case Military Conditions)

Dedicated (Hard-Wired) I/O Clock Network		Std Speed		–1 Speed		
Parameter	**Description**	**Min.**	**Max.**	**Min.**	**Max.**	**Units**
t_{IOCKH}	Input Low to High (Pad to I/O Module Input)		4.1		3.5	ns
t_{IOPWH}	Minimum Pulse Width High	5.7		4.8		ns
t_{IOPWL}	Minimum Pulse Width Low	5.7		4.8		ns
t_{IOSAPW}	Minimum Asynchronous Pulse Width	4.4		3.9		ns
t_{IOCKSW}	Maximum Skew		1.0		0.9	ns
t_{IOP}	Minimum Period	11.6		9.9		ns
f_{IOMAX}	Maximum Frequency		85		100	MHz
Dedicated (Hard-Wired) Array Clock Network						
t_{HCKH}	Input Low to High (Pad to S-Module Input)		6.4		5.5	ns
t_{HCKL}	Input High to Low (Pad to S-Module Input)		6.4		5.5	ns
t_{HPWH}	Minimum Pulse Width High	5.7		4.8		ns
t_{HPWL}	Minimum Pulse Width Low	5.7		4.8		ns
t_{HCKSW}	Maximum Skew		1.0		0.9	ns
t_{HP}	Minimum Period	11.6		9.9		ns
f_{HMAX}	Maximum Frequency		85		100	MHz
Routed Array Clock Networks						
t_{RCKH}	Input Low to High (FO=256)		10.5		9.0	ns
t_{RCKL}	Input High to Low (FO=256)		10.5		9.0	ns
t_{RPWH}	Min. Pulse Width High (FO=256)	7.1		6.3		ns
t_{RPWL}	Min. Pulse Width Low (FO=256)	7.1		6.3		ns
t_{RCKSW}	Maximum Skew (FO=128)		2.1		1.9	ns
t_{RP}	Minimum Period (FO=256)	14.5		12.9		ns
f_{RMAX}	Maximum Frequency (FO=256)		65		75	MHz
Clock-to-Clock Skews						
$t_{IOHCKSW}$	I/O Clock to H-Clock Skew	0.0	3.0	0.0	3.0	ns
$t_{IORCKSW}$	I/O Clock to R-Clock Skew	0.0	5.0	0.0	5.0	ns
t_{HRCKSW}	H-Clock to R-Clock Skew					
	(FO = 64)	0.0	1.0	0.0	1.0	ns
	(FO = 50% max.)	0.0	3.0	0.0	3.0	ns

Note:

1. Delays based on 35 pF loading.

A14100A Timing Characteristics

(Worst-Case Military Conditions)

Logic Module Propagation Delays[1]		Std Speed		–1 Speed		
Parameter	**Description**	**Min.**	**Max.**	**Min.**	**Max.**	**Units**
t_{PD}	Internal Array Module		3.5		3.0	ns
t_{CO}	Sequential Clock to Q		3.5		3.0	ns
t_{CLR}	Asynchronous Clear to Q		3.5		3.0	ns
Predicted Routing Delays[2]						
t_{RD1}	FO=1 Routing Delay		1.5		1.3	ns
t_{RD2}	FO=2 Routing Delay		2.1		1.9	ns
t_{RD3}	FO=3 Routing Delay		2.5		2.1	ns
t_{RD4}	FO=4 Routing Delay		2.9		2.6	ns
t_{RD8}	FO=8 Routing Delay		4.9		4.2	ns
Logic Module Sequential Timing						
t_{SUD}	Flip-Flop (Latch) Data Input Setup	1.0		1.0		ns
t_{HD}	Flip-Flop (Latch) Data Input Hold	0.6		0.6		ns
t_{SUENA}	Flip-Flop (Latch) Enable Setup	1.0		1.0		ns
t_{HENA}	Flip-Flop (Latch) Enable Hold	0.6		0.6		ns
t_{WASYN}	Asynchronous Pulse Width	5.6		4.8		ns
t_{WCLKA}	Flip-Flop Clock Pulse Width	5.6		4.8		ns
t_A	Flip-Flop Clock Input Period	11.6		9.9		ns
f_{MAX}	Flip-Flop Clock Frequency		85		100	MHz

Notes:

1. *For dual-module macros, use $t_{PD} + t_{RD1} + t_{PDn}$, $t_{CO} + t_{RD1} + t_{PDn}$, or $t_{PD1} + t_{RD1} + t_{SUD}$, whichever is appropriate.*
2. *Routing delays are for typical designs across worst-case operating conditions. These parameters should be used for estimating device performance. Post-route timing analysis or simulation is required to determine actual worst-case performance. Post-route timing is based on actual routing delay measurements performed on the device prior to shipment.*

A14100A Timing Characteristics (continued)

(Worst-Case Military Conditions)

I/O Module Input Propagation Delays		Std Speed		–1 Speed		
Parameter	**Description**	**Min.**	**Max.**	**Min.**	**Max.**	**Units**
t_{INY}	Input Data Pad to Y		4.9		4.2	ns
t_{ICKY}	Input Reg IOCLK Pad to Y		8.2		7.0	ns
t_{OCKY}	Output Reg IOCLK Pad to Y		8.2		7.0	ns
t_{ICLRY}	Input Asynchronous Clear to Y		8.2		7.0	ns
t_{OCLRY}	Output Asynchronous Clear to Y		8.2		7.0	ns
Predicted Input Routing Delays[1]						
t_{IRD1}	FO=1 Routing Delay		1.5		1.3	ns
t_{IRD2}	FO=2 Routing Delay		2.1		1.9	ns
t_{IRD3}	FO=3 Routing Delay		2.5		2.1	ns
t_{IRD4}	FO=4 Routing Delay		2.9		2.6	ns
t_{IRD8}	FO=8 Routing Delay		4.9		4.2	ns
I/O Module Sequential Timing						
t_{INH}	Input F-F Data Hold (w.r.t. IOCLK Pad)	0.0		0.0		ns
t_{INSU}	Input F-F Data Setup (w.r.t. IOCLK Pad)	2.4		2.1		ns
t_{IDEH}	Input Data Enable Hold (w.r.t. IOCLK Pad)	0.0		0.0		ns
t_{IDESU}	Input Data Enable Setup (w.r.t. IOCLK Pad)	10.0		8.7		ns
t_{OUTH}	Output F-F Data Hold (w.r.t. IOCLK Pad)	1.2		1.2		ns
t_{OUTSU}	Output F-F Data Setup (w.r.t. IOCLK Pad)	1.2		1.2		ns
t_{ODEH}	Output Data Enable Hold (w.r.t. IOCLK Pad)	0.6		0.6		ns
t_{ODESU}	Output Data Enable Setup (w.r.t. IOCLK Pad)	2.4		2.4		ns

Note:

1. *Routing delays are for typical designs across worst-case operating conditions. These parameters should be used for estimating device performance. Post-route timing analysis or simulation is required to determine actual worst-case performance. Post-route timing is based on actual routing delay measurements performed on the device prior to shipment.*

A14100A Timing Characteristics (continued)

(Worst-Case Military Conditions)

I/O Module – TTL Output Timing[1]		Std Speed		–1 Speed		
Parameter	**Description**	**Min.**	**Max.**	**Min.**	**Max.**	**Units**
t_{DHS}	Data to Pad, High Slew		8.9		7.5	ns
t_{DLS}	Data to Pad, Low Slew		14.0		11.9	ns
t_{ENZHS}	Enable to Pad, Z to H/L, Hi Slew		7.0		6.0	ns
t_{ENZLS}	Enable to Pad, Z to H/L, Lo Slew		12.8		10.9	ns
t_{ENHSZ}	Enable to Pad, H/L to Z, Hi Slew		14.0		11.9	ns
t_{ENLSZ}	Enable to Pad, H/L to Z, Lo Slew		12.8		10.9	ns
t_{CKHS}	IOCLK Pad to Pad H/L, Hi Slew		14.0		12.2	ns
t_{CKLS}	IOCLK Pad to Pad H/L, Lo Slew		17.8		17.8	ns
d_{TLHHS}	Delta Low to High, Hi Slew		0.04		0.04	ns/pF
d_{TLHLS}	Delta Low to High, Lo Slew		0.08		0.07	ns/pF
d_{THLHS}	Delta High to Low, Hi Slew		0.06		0.05	ns/pF
d_{THLLS}	Delta High to Low, Lo Slew		0.08		0.07	ns/pF
I/O Module – CMOS Output Timing[1]						
t_{DHS}	Data to Pad, High Slew		10.8		9.2	ns
t_{DLS}	Data to Pad, Low Slew		20.3		17.3	ns
t_{ENZHS}	Enable to Pad, Z to H/L, Hi Slew		9.1		7.7	ns
t_{ENZLS}	Enable to Pad, Z to H/L, Lo Slew		15.5		13.1	ns
t_{ENHSZ}	Enable to Pad, H/L to Z, Hi Slew		14.0		11.6	ns
t_{ENLSZ}	Enable to Pad, H/L to Z, Lo Slew		12.8		10.9	ns
t_{CKHS}	IOCLK Pad to Pad H/L, Hi Slew		16.0		14.4	ns
t_{CKLS}	IOCLK Pad to Pad H/L, Lo Slew		22.4		20.2	ns
d_{TLHHS}	Delta Low to High, Hi Slew		0.07		0.06	ns/pF
d_{TLHLS}	Delta Low to High, Lo Slew		0.13		0.11	ns/pF
d_{THLHS}	Delta High to Low, Hi Slew		0.05		0.04	ns/pF
d_{THLLS}	Delta High to Low, Lo Slew		0.06		0.05	ns/pF

Note:

1. Delays based on 35 pF loading.

A14100A Timing Characteristics (continued)

(Worst-Case Military Conditions)

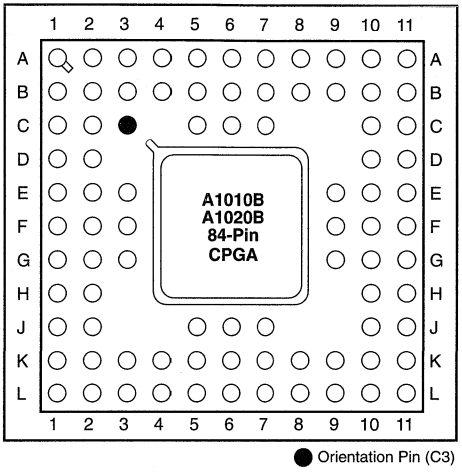
Dedicated (Hard-Wired) I/O Clock Network		Std Speed		–1 Speed		
Parameter	**Description**	**Min.**	**Max.**	**Min.**	**Max.**	**Units**
t_{IOCKH}	Input Low to High (Pad to I/O Module Input)		4.1		3.5	ns
t_{IOPWH}	Minimum Pulse Width High	5.7		4.8		ns
t_{IOPWL}	Minimum Pulse Width Low	5.7		4.8		ns
t_{IOSAPW}	Minimum Asynchronous Pulse Width	4.4		3.9		ns
t_{IOCKSW}	Maximum Skew		1.0		0.9	ns
t_{IOP}	Minimum Period	11.6		9.9		ns
f_{IOMAX}	Maximum Frequency		85		100	MHz
Dedicated (Hard-Wired) Array Clock Network						
t_{HCKH}	Input Low to High (Pad to S-Module Input)		6.4		5.5	ns
t_{HCKL}	Input High to Low (Pad to S-Module Input)		6.4		5.5	ns
t_{HPWH}	Minimum Pulse Width High	5.7		4.8		ns
t_{HPWL}	Minimum Pulse Width Low	5.7		4.8		ns
t_{HCKSW}	Maximum Skew		1.0		0.9	ns
t_{HP}	Minimum Period	11.6		9.9		ns
f_{HMAX}	Maximum Frequency		85		100	MHz
Routed Array Clock Networks						
t_{RCKH}	Input Low to High (FO=256)		10.5		9.0	ns
t_{RCKL}	Input High to Low (FO=256)		10.5		9.0	ns
t_{RPWH}	Min. Pulse Width High (FO=256)	7.1		6.3		ns
t_{RPWL}	Min. Pulse Width Low (FO=256)	7.1		6.3		ns
t_{RCKSW}	Maximum Skew (FO=128)		2.1		1.9	ns
t_{RP}	Minimum Period (FO=256)	14.5		12.9		ns
f_{RMAX}	Maximum Frequency (FO=256)		65		75	MHz
Clock-to-Clock Skews						
$t_{IOHCKSW}$	I/O Clock to H-Clock Skew	0.0	3.5	0.0	3.5	ns
$t_{IORCKSW}$	I/O Clock to R-Clock Skew	0.0	5.0	0.0	5.0	ns
t_{HRCKSW}	H-Clock to R-Clock Skew (FO = 64) (FO = 50% max.)	 0.0 0.0	 1.0 3.0	 0.0 0.0	 1.0 3.0	ns

Note:

1. Delays based on 35 pF loading.

Package Pin Assignments

84-Pin CPGA (Top View)



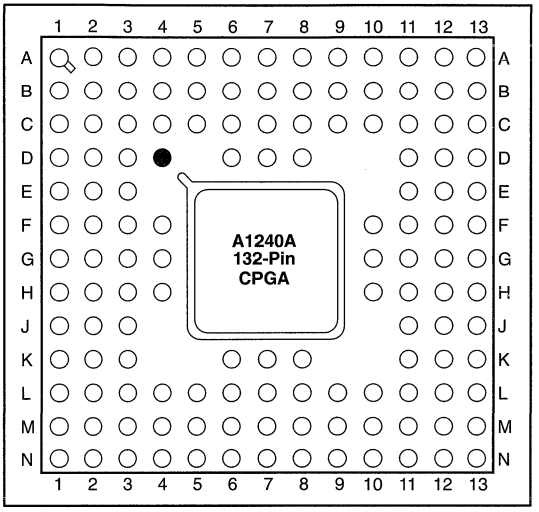
Signal	A1010B Devices	A1020B Devices
CLK or I/O	F9	F9
DCLK or I/O	C10	C10
GND	B7, E2, E3, F10, G10, K5	B7, E2, E3, F10, G10, K5
MODE	E11	E11
N/C (No Connection)	B1, B2, C1, C2, C11, D10, D11, J2, J10, K1, K10, K11, L1	B2
PRA or I/O	A11	A11
PRB or I/O	B10	B10
SDI or I/O	B11	B11
V_{CC}	B5, E9, E10, F1, G2, K7	B5, E9, E10, F1, G2, K7
V_{PP}	K2	K2

Notes:

1. *V_{PP} must be terminated to V_{CC}, except during device programming.*
2. *MODE must be terminated to circuit ground, except during device programming or debugging.*
3. *Unused I/O pins are designated as outputs by ALS and are driven low.*
4. *All unassigned pins are available for use as I/Os.*

Package Pin Assignments (continued)

132-Pin CPGA (Top View)



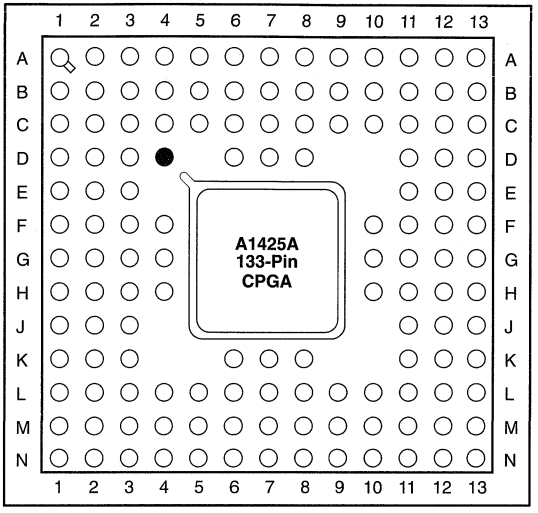
Signal	Location
CLKA or I/O	B7
CLKB or I/O	B6
DCLK or I/O	C3
GND	B5, B9, C5, C9, E3, E11, E12, F4, J2, J3, J11, K12, L5, L9, M9
MODE	A1
PRA or I/O	B8
PRB or I/O	C6
SDI or I/O	B12
V_{CC}	C7, D7, G2, G3, G10, G11, K7, L7
V_{PP}	G13
V_{KS}	H13
V_{SV}	G4, G12

Notes:

1. *Unused I/O pins are designated as outputs by ALS and are driven low.*
2. *All unassigned pins are available for use as I/Os.*
3. *MODE = GND, except during device programming or debugging.*
4. *$V_{PP} = V_{CC}$, except during device programming.*
5. *$V_{SV} = V_{CC}$, except during device programming.*
6. *V_{KS} = GND, except during device programming.*

Package Pin Assignments (continued)

133-Pin CPGA (Top View)



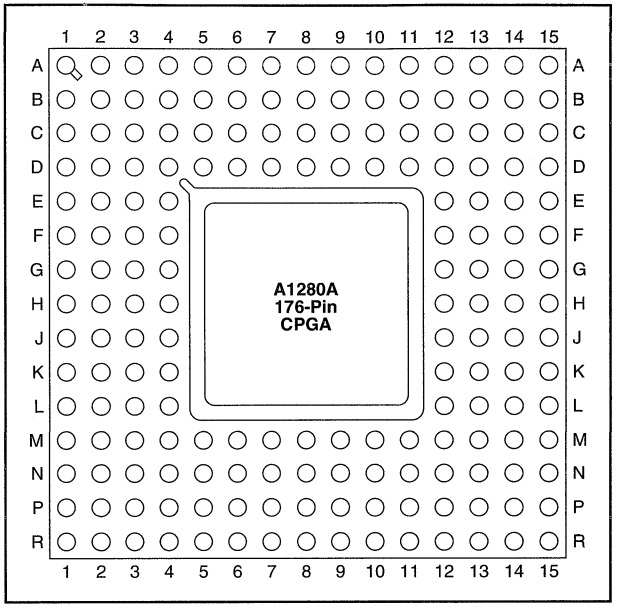
Signal	Location
CLKA or I/O	D7
CLKB or I/O	B6
DCLK or I/O	D4
GND	A2, C3, C7, C11, C12, G3, G11, L3, L7, L11, M3, N12
HCLKA or I/O	K7
IOCLK or I/O	C10
IOPCL or I/O	L10
MODE	E3
NC	A1, A7, A13, G1, G13, N1, N7, N13
PRA or I/O	A6
PRB or I/O	L6
SDI or I/O	C2
V_{CC}	B2, B7, B12, G2, G12, M2, M7, M12
V_{KS}	F10
V_{PP}	E11
V_{SV}	J2, J12

Notes:

1. *Unused I/O pins are designated as outputs by ALS and are driven low.*
2. *All unassigned pins are available for use as I/Os.*
3. *MODE = GND, except during device programming or debugging.*
4. *$V_{PP} = V_{CC}$, except during device programming.*
5. *$V_{SV} = V_{CC}$, except during device programming.*
6. *V_{KS} = GND, except during device programming.*

Package Pin Assignments (continued)

176-Pin CPGA (Top View)



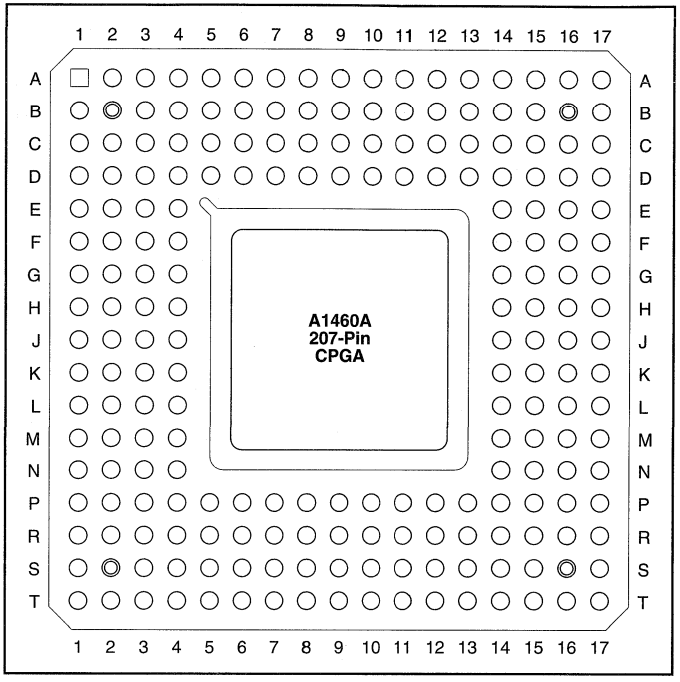
Signal	Location
CLKA or I/O	A9
CLKB or I/O	B8
DCLK or I/O	B3
GND	C8, D4, D6, D10, D12, E4, E12, F12, G4, H4, H12, J12, K4, K12, L4, M4, M6, M8, M10, M12
MODE	C3
PRA or I/O	C9
PRB or I/O	D7
SDI or I/O	B14
V_{CC}	D5, D8, D11, F4, G12, H3, H13, J4, M5, M11, N8
V_{KS}	J13
V_{PP}	J14
V_{SV}	H2, H14

Notes:

1. *Unused I/O pins are designated as outputs by ALS and are driven low.*
2. *All unassigned pins are available for use as I/Os.*
3. *MODE = GND, except during device programming or debugging.*
4. *$V_{PP} = V_{CC}$, except during device programming.*
5. *$V_{SV} = V_{CC}$, except during device programming.*
6. *V_{KS} = GND, except during device programming.*

Package Pin Assignments (continued)

207-Pin CPGA (Top View)



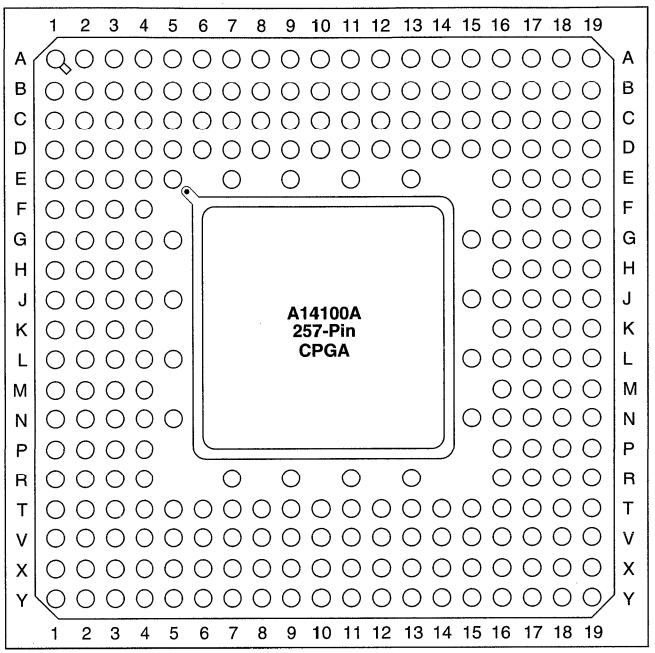
Signal	Location
CLKA or I/O	K1
CLKB or I/O	J3
DCLK or I/O	E4
GND	C15, D4, D5, D9, D14, J4, J14, P3, P4, P9, P14, R15
HCKL or I/O	J15
IOCLK or I/O	P5
IOPCL or I/O	N14
MODE	D7
NC	A1, A2, A16, A17, B1, B17, C1, C2, S1, S3, S17, T1, T2, T16, T17
PRA OR I/O	H1
PRB or I/O	K16
SDI or I/O	C3
V_{CC}	B2, B9, B16, J2, J16, S2, S9, S16
V_{KS}	P7
V_{PP}	T5
V_{SV}	D11, P12

Notes:

1. *Unused I/O pins are designated as outputs by ALS and are driven low.*
2. *All unassigned pins are available for use as I/Os.*
3. *MODE = GND, except during device programming or debugging.*
4. *$V_{PP} = V_{CC}$, except during device programming.*
5. *$V_{SV} = V_{CC}$, except during device programming.*
6. *V_{KS} = GND, except during device programming.*

Package Pin Assignments (continued)

257-Pin CPGA (Top View)



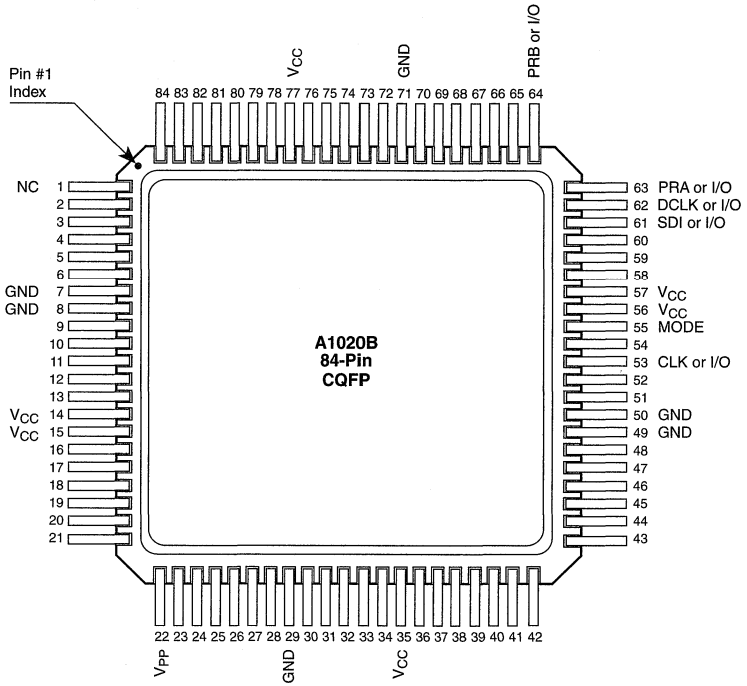
Signal	Location
CLKA or I/O	L4
CLKB or I/O	L5
DCLK or I/O	E4
GND	B16, C4, D4, D10, D16, E11, J5, K4, K16, L15, R4, T4, T10, T16, T17
HCLK or I/O	J16
IOCLK or I/O	T5
IOPCL or I/O	R16
MODE	A5
NC	E5
PRA OR I/O	J1
PRB or I/O	J17
SDI or I/O	B4
V_{CC}	C3, C10, C17, K3, K17, V3, V10, V17
V_{KS}	X7
V_{PP}	V7
V_{SV}	C13, X14

Notes:

1. *Unused I/O pins are designated as outputs by ALS and are driven low.*
2. *All unassigned pins are available for use as I/Os.*
3. *MODE = GND, except during device programming or debugging.*
4. *$V_{PP} = V_{CC}$, except during device programming.*
5. *$V_{SV} = V_{CC}$, except during device programming.*
6. *V_{KS} = GND, except during device programming.*

Package Pin Assignments (continued)

84-Pin CQFP (Top View)

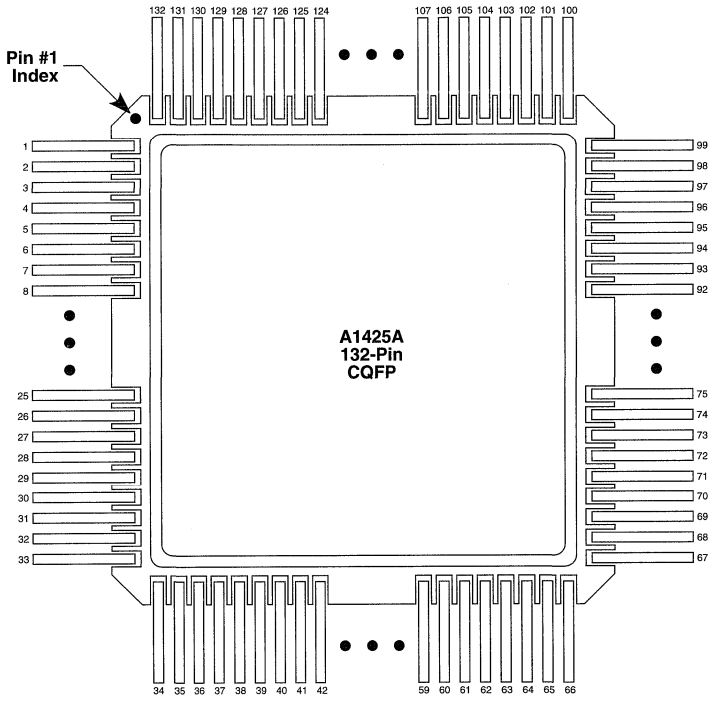


Notes:

1. *V_{PP} must be terminated to V_{CC}, except during device programming.*
2. *MODE must be terminated to circuit ground, except during device programming or debugging.*
3. *Unused I/O pins are designated as outputs by ALS and are driven low.*
4. *All unassigned pins are available for use as I/Os.*

Package Pin Assignments (continued)

132-Pin CQFP (Top View)



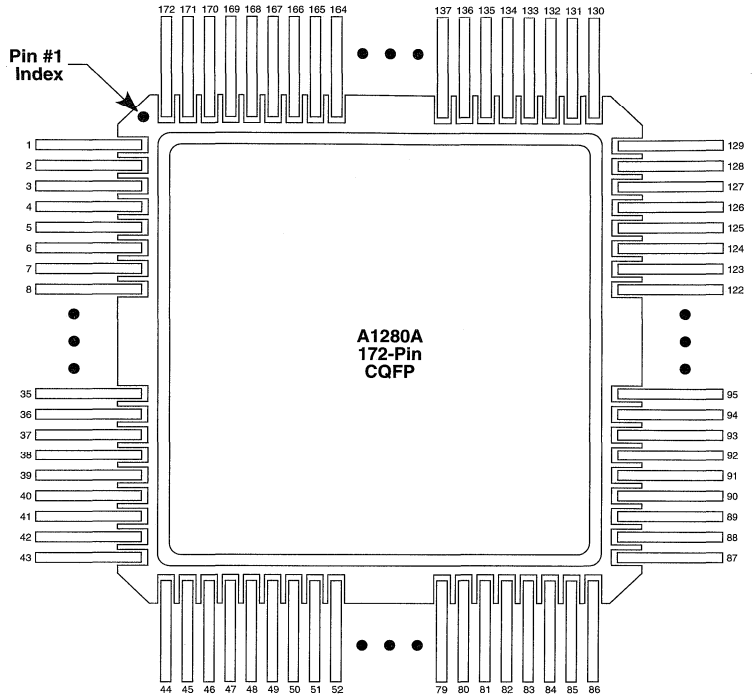
Signal	Pin Number
CLKA or I/O	116
CLKB or I/O	117
DCLK or I/O	131
GND	2, 10, 26, 36, 42, 58, 65, 74, 90, 101, 106, 122
HCLK or I/O	50
IOCLK or I/O	98
IOPCL or I/O	64
MODE	9
NC	1, 34, 66, 67, 99, 100, 132
PRA or I/O	118
PRB or I/O	48
SDI or I/O	3
V_{CC}	11, 27, 43, 59, 75, 91, 107, 123
V_{KS}	92
V_{PP}	89
V_{SV}	22, 78

Notes:

1. *Unused I/O pins are designated as outputs by ALS and are driven low.*
2. *All unassigned pins are available for use as I/Os.*
3. *MODE = GND, except during device programming or debugging.*
4. *$V_{PP} = V_{CC}$, except during device programming.*
5. *$V_{SV} = V_{CC}$, except during device programming.*
6. *V_{KS} = GND, except during device programming.*

Package Pin Assignments (continued)

172-Pin CQFP (Top View)



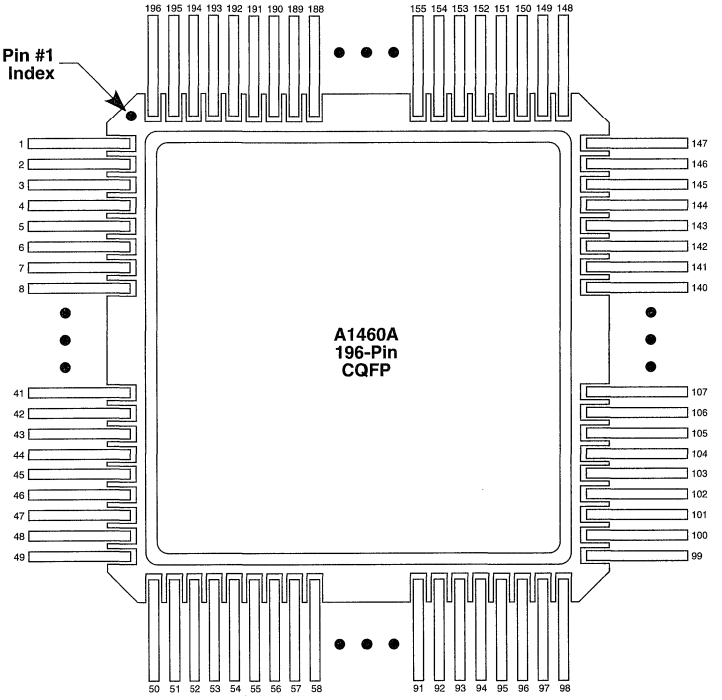
Signal	Pin Number
CLKA or I/O	150
CLKB or I/O	154
DCLK or I/O	171
GND	7, 17, 22, 32, 37, 55, 65, 75, 98, 103, 108, 118, 123, 141, 152, 161
MODE	1
PRA or I/O	148
PRB or I/O	156
SDI or I/O	131
V_{CC}	12, 23, 27, 50, 66, 80, 109, 113, 136, 151, 166
V_{KS}	106
V_{PP}	107
V_{SV}	24, 110

Notes:

1. *Unused I/O pins are designated as outputs by ALS and are driven low.*
2. *All unassigned pins are available for use as I/Os.*
3. *MODE = GND, except during device programming or debugging.*
4. *$V_{PP} = V_{CC}$, except during device programming.*
5. *$V_{SV} = V_{CC}$, except during device programming.*
6. *V_{KS} = GND, except during device programming.*

Package Pin Assignments (continued)

196-Pin CQFP (Top View)



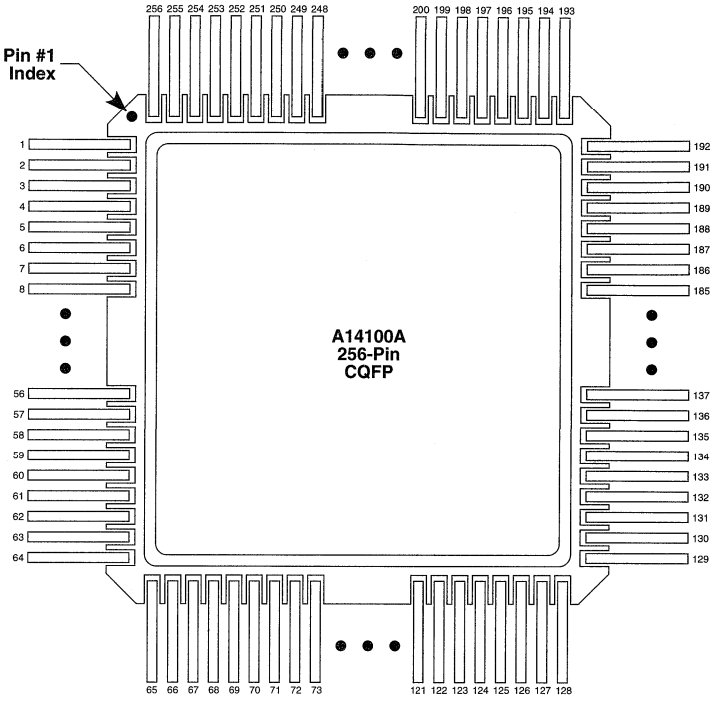
Signal	Pin Number
CLKA or I/O	172
CLKB or I/O	173
DCLK or I/O	196
GND	1, 13, 37, 51, 52, 64, 86, 98, 101, 112, 139, 149, 162, 183, 193
HCLK or I/O	77
IOCLK or I/O	148
IOPCL or I/O	100
MODE	11
PRA or I/O	174
PRB or I/O	75
SDI or I/O	2
V_{CC}	12, 38, 59, 94, 111, 140, 155, 189
V_{KS}	138
V_{PP}	137
V_{SV}	39, 110

Notes:

1. *Unused I/O pins are designated as outputs by ALS and are driven low.*
2. *All unassigned pins are available for use as I/Os.*
3. *MODE = GND, except during device programming or debugging.*
4. *$V_{PP} = V_{CC}$, except during device programming.*
5. *$V_{SV} = V_{CC}$, except during device programming.*
6. *V_{KS} = GND, except during device programming.*

Package Pin Assignments(continued)

256-Pin CQFP (Top View)



Signal	Pin Number
CLKA or I/O	222
CLKB or I/O	223
DCLK or I/O	256
GND	1, 13, 44, 59, 60, 78, 109, 125, 129, 144, 175, 189, 190, 208, 239
HCLK or I/O	95
IOCLK or I/O	188
IOPCL or I/O	128
MODE	11
PRA or I/O	224
PRB or I/O	93
SDI or I/O	2
V_{CC}	12, 45, 70, 115, 143, 177, 202, 245
V_{KS}	174
V_{PP}	173
V_{SV}	46, 142

Notes:

1. *Unused I/O pins are designated as outputs by ALS and are driven low.*
2. *All unassigned pins are available for use as I/Os.*
3. *MODE = GND, except during device programming or debugging.*
4. *$V_{PP} = V_{CC}$, except during device programming.*
5. *$V_{SV} = V_{CC}$, except during device programming.*
6. *V_{KS} = GND, except during device programming.*

Actel
Mask Programmed Gate Arrays

Features

- Mask Programmed versions of Actel Field Programmable Gate Arrays (FPGAs)
- Significant cost reduction for medium- to high-volume applications
- Pin-for-pin compatible with Actel FPGAs
- PCI Local Bus Revision 2 Compliant
- Automatic translation from Actel FPGA netlist to MPGA
- Test vectors generated from customer simulation vectors
- Short lead times for prototype and production devices
- MPGA available for all ACT 1, ACT 2, and ACT 3 devices
- Device sizes from 1,200 to 10,000 gates
- Up to 175 user I/Os
- Available in commercial or industrial temperature ranges
- PLCC, PQFP, VQFP, and TQFP packages available
- Meets all internal worst-case FPGA performance specifications
- Lower I/O capacitance than FPGA
- Lower power dissipation than FPGA

Description

The Actel Mask Programmed Gate Array (MPGA) products are masked versions of the popular Actel FPGA families. These semi-custom devices offer the customer a design path that provides significant cost reduction without significant risk or engineering effort. For medium- to high-volume applications in which the design is fixed, the Actel FPGA used for prototyping and initial production can be replaced by the corresponding MPGA device.

The granular, regular structure of the Actel antifuse-based FPGA products enables easy conversion to MPGA. Actel provides all required engineering services to convert the customer design from FPGA to MPGA, using proprietary software to automatically convert the FPGA logic design into the MPGA device. Test vector generation is made easy by software that converts the customer's third-party simulation vectors into the final vectors used to test the device in production.

All Actel MPGA devices are pin-for-pin compatible with the corresponding FPGA, and therefore no board redesign is required. MPGA devices meet all worst-case timing specifications of the FPGA devices. MPGA devices are available for all plastic packaged devices from ACT 1, ACT 2, and ACT 3 families. See the Product Plan on page 3 for a detailed list of available device and package combinations.

Product Family Profile

	Capacity				Available Packages			
MPGA Device Type	Gate Array Equivalent Gates	PLD Equivalent Gates	Flip-Flops (Maximum)	User I/Os (Maximum)	PLCC	PQFP	VQFP	TQFP
M1010	1200	3000	147	57	44, 68-pin	100-pin	80-pin	—
M1020	2000	6000	273	69	44, 68, 84-pin	100-pin	80-pin	—
M1225	2500	6250	382	83	84-pin	100-pin	100-pin	—
M1240	4000	10000	568	104	84-pin	144-pin	—	176-pin
M1280	8000	20000	998	140	84-pin	160-pin	—	176-pin
M1415	1500	3750	312	80	84-pin	100-pin	100-pin	—
M1425	2500	6250	435	100	84-pin	100, 160-pin	100-pin	—
M1440	4000	10000	706	140	84-pin	160-pin	100-pin	176-pin
M1460	6000	15000	976	167	—	160, 208-pin	—	176-pin
M14100	10000	25000	1153	175	—	208-pin	—	—

Actel FPGA to MPGA Design Flow

Actel's three families of FPGA devices offer a wide selection of device sizes, package choices, performance characteristics, and price points. The FPGA families provide the ideal prototyping tool and are cost-effective for low- to medium-volume applications. As volumes increase, a cost-reduction path becomes a key factor to ensure continued success and profitability of the end product. Once the design has stabilized and volumes are increasing, a choice can be made to convert the design to an MPGA. Since the MPGA product is pin-for-pin compatible with the FPGA, no board redesign is required, and the MPGA can directly replace the FPGA.

A typical design process uses the FPGA device as the prototyping and initial production product of choice and converts to the MPGA as volumes warrant. Figure 1 shows the design process for Actel FPGA and MPGA devices. This option gives you the flexibility to adjust volumes as the demand for the end product changes. Since the MPGA is a semicustom device, all production is built to your order. If the design is already completed in the FPGA, any demand upsides can be satisfied by temporarily switching production back to the FPGA. Since Actel FPGAs are standard off-the-shelf devices, additional product requirements can be met within a short lead time.

The Actel FPGA devices offer the easiest and fastest way to bring a new product to market, and the three FPGA families offer a wide selection of low-cost, high-performance devices. The addition of the MPGA devices offers a simple, low-risk cost-reduction path as production volumes increase.

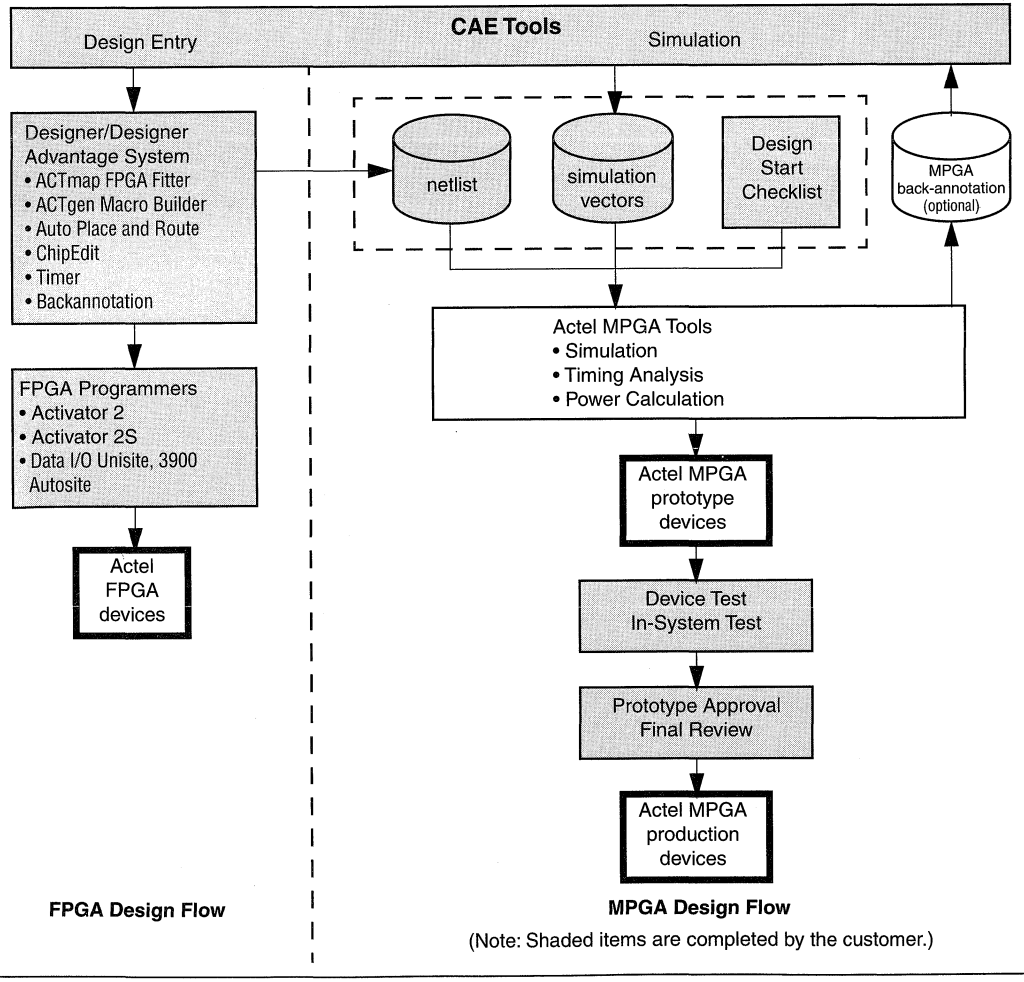


Figure 1 • Actel Device Design Flow

Product Plan

	Availability	Application	
ACT 1 Family		**Commercial**	**Industrial**
M1010 Device			
44-pin Plastic Leaded Chip Carrier (PLCC)	✔	✔	✔
68-pin Plastic Leaded Chip Carrier (PLCC)	✔	✔	✔
80-pin Very Thin Plastic Quad Flatpack (VQFP)	✔	✔	—
100-pin Plastic Quad Flatpack (PQFP)	✔	✔	✔
M1020 Device			
44-pin Plastic Leaded Chip Carrier (PLCC)	✔	✔	✔
68-pin Plastic Leaded Chip Carrier (PLCC)	✔	✔	✔
80-pin Very Thin Plastic Quad Flatpack (VQFP)	✔	✔	—
84-pin Plastic Leaded Chip Carrier (PLCC)	✔	✔	✔
100-pin Plastic Quad Flatpack (PQFP)	✔	✔	✔
ACT 2 Family			
M1225 Device			
84-pin Plastic Leaded Chip Carrier (PLCC)	✔	✔	✔
100-pin Plastic Quad Flatpack (PQFP)	✔	✔	✔
100-pin Thin Plastic Quad Flatpack (TQFP)	✔	✔	—
M1240 Device			
84-pin Plastic Leaded Chip Carrier (PLCC)	✔	✔	✔
144-pin Plastic Quad Flatpack (PQFP)	✔	✔	✔
176-pin Thin Plastic Quad Flatpack (TQFP)	✔	✔	—
M1280 Device			
84-pin Plastic Leaded Chip Carrier (PLCC)	✔	✔	✔
160-pin Plastic Quad Flatpack (PQFP)	✔	✔	✔
176-pin Thin Plastic Quad Flatpack (TQFP)	✔	✔	—
ACT 3 Family			
M1415 Device			
84-pin Plastic Leaded Chip Carrier (PLCC)	✔	✔	✔
100-pin Plastic Quad Flatpack (PQFP)	✔	✔	✔
100-pin Very Thin Plastic Quad Flatpack (VQFP)	✔	✔	—
M1425 Device			
84-pin Plastic Leaded Chip Carrier (PLCC)	✔	✔	✔
100-pin Plastic Quad Flatpack (PQFP)	✔	✔	✔
100-pin Very Thin Plastic Quad Flatpack (VQFP)	✔	✔	—
160-pin Plastic Quad Flatpack (PQFP)	✔	✔	✔
M1440 Device			
84-pin Plastic Leaded Chip Carrier (PLCC)	✔	✔	✔
100-pin Very Thin Plastic Quad Flatpack (VQFP)	✔	✔	—
160-pin Plastic Quad Flatpack (PQFP)	✔	✔	✔
176-pin Thin Plastic Quad Flatpack (TQFP)	✔	✔	—
M1460 Device			
160-pin Plastic Quad Flatpack (PQFP)	✔	✔	✔
176-pin Thin Plastic Quad Flatpack (TQFP)	✔	✔	—
208-pin Plastic Quad Flatpack (PQFP)	✔	✔	✔
M14100 Device			
208-pin Plastic Quad Flatpack (PQFP)	✔	✔	✔

Availability: ✔ = Available
P = Planned
— = Not Planned

ACT 1 Device Resources

MPGA Device Type	Gate Array Equivalent Gates	User I/Os				
		PLCC			PQFP	VQFP
		44-pin	68-pin	84-pin	100-pin	80-pin
M1010	1200	34	57	57	57	57
M1020	2000	34	57	69	69	69

ACT 2 Device Resources

MPGA Device Type	Gate Array Equivalent Gates	User I/Os					
		PLCC	PQFP			VQFP	TQFP
		84-pin	100-pin	144-pin	160-pin	100-pin	176-pin
M1225	2500	72	83	—	—	83	—
M1240	4000	72	—	104	—	—	104
M1280	8000	72	—	—	125	—	140

ACT 3 Device Resources

MPGA Device Type	Gate Array Equivalent Gates	User I/Os					
		PLCC	PQFP			VQFP	TQFP
		84-pin	100-pin	160-pin	208-pin	100-pin	176-pin
M1415	1500	70	80	—	—	80	—
M1425	2500	70	80	100	—	83	—
M1440	4000	70	—	131	—	83	140
M1460	6000	—	—	131	167	—	151
M14100	10000	—	—	—	175	—	—

Ordering Information

M1460 – PQ 208 C

- **C** — Application (Temperature Range)
  - C = Commercial (0 to +70°C)
  - I = Industrial (–40 to +85°C)
- **208** — Package Lead Count
- **PQ** — Package Type
  - PL = Plastic Leaded Chip Carrier (PLCC)
  - PQ = Plastic Quad Flatpack (PQFP)
  - TQ = Thin (1.4 mm) Plastic Quad Flatpack (TQFP)
  - VQ = Very Thin (1.0 mm) Plastic Quad Flatpack (VQFP)
- **M1460** — Part Number
  - M1010 = 1200 Gates—ACT 1
  - M1020 = 2000 Gates—ACT 1
  - M1225 = 2500 Gates—ACT 2
  - M1240 = 4000 Gates—ACT 2
  - M1280 = 8000 Gates—ACT 2
  - M1415 = 1500 Gates—ACT 3
  - M1425 = 2500 Gates—ACT 3
  - M1440 = 4000 Gates—ACT 3
  - M1460 = 6000 Gates—ACT 3
  - M14100 = 10000 Gates—ACT 3

Absolute Maximum Ratings[1]

Free air temperature range

Symbol	Parameter	Limits	Units
V_{CC}	DC Supply Voltage	–0.3 to +7.0	V
V_I	Input Voltage	–0.3 to V_{CC} +0.3	V
V_O	Output Voltage	–0.3 to V_{CC} +0.3	V
I_{IO}	I/O Source Sink Current	±20	mA
T_{STG}	Storage Temperature	–55 to +125	°C

Note:

1. *Stresses beyond those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. Exposure to absolute maximum rated conditions for extended periods may affect device reliability. Device should not be operated outside the Recommended Operating Conditions.*

Recommended Operating Conditions

Parameter	Commercial	Industrial	Units
Temperature Range	0 to +70	–40 to +85	°C
Power Supply Tolerance	±5	±10	%V_{CC}

1

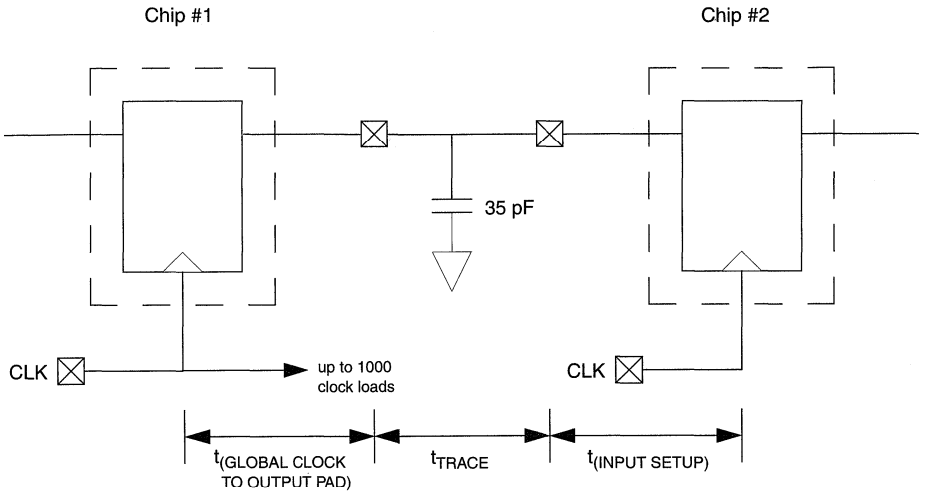
Electrical Specifications

Symbol	Parameter	Test Condition	Commercial		Industrial		Units
			Min.	Max.	Min.	Max.	
V_{OH}[1,2]	HIGH Level Output	I_{OH} = –6 mA (CMOS)	3.7		3.7		V
		I_{OH} = –8 mA (TTL)[3]	2.4		2.4		V
V_{OL}[1,2]	LOW Level Output	I_{OL} = +6 mA (CMOS)		0.4		0.4	V
		I_{OL} = +8 mA (TTL)[3]		0.4		0.4	V
V_{IH}	HIGH Level Input	TTL Inputs	2.0	V_{CC} + 0.3	2.0	V_{CC} + 0.3	V
V_{IL}	LOW Level Input	TTL Inputs	–0.3	0.8	–0.3	0.8	V
I_{IN}	Input Leakage	V_I = V_{CC} or GND	–1	+1	–1	+1	µA
I_{OZ}	3-state Output Leakage	V_O = V_{CC} or GND	–10	+10	–10	+10	µA
C_{IO}	I/O Capacitance[3]			10		10	pF
$I_{CC(S)}$	Standby Supply Current	V_I = V_{CC} or GND, I_O = 0 mA		100		500	µA

Notes:

1. *Actel devices can drive and receive either CMOS or TTL signal levels. No assignment of I/Os as TTL or CMOS is required.*
2. *Tested one output at a time, V_{CC} = min.*
3. *Not tested, for information only.*

Chip-to-Chip Performance



Chip-to-Chip Performance (Worst-Case Commercial)

	$t_{(GLOBAL\ CLOCK\ TO\ OUPUT\ PAD)}$	t_{TRACE}	$t_{(INPUT\ SETUP)}$	Total	MHz
Actel MPGA	12.7	1.0	3.1	16.8	60

Pin Description

Package pin assignments for an FPGA design are directly transferred to the equivalent MPGA package because all I/O and power pins are located in identical positions. While the conversion of package pin assignments is transparent in the end product, there are two small functional differences to note between the device types. First, dedicated FPGA global and debugging pins are general purpose MPGA I/O pins. Also, dedicated FPGA programming voltage pins are Vcc or ground pins on an MPGA. Refer to Table 1 for a complete cross-reference of pin descriptions between the FPGA and MPGA.

Table 1 • FPGA-to-MPGA Pin Cross-Reference

FPGA Pin Description		MPGA Pin Description
CLK **Clock (ACT 1 only)** TTL Clock input for ACT 1 global clock distribution network. This pin can also be used as an I/O.	→	**no change** If desired, TTL Clock input signals may be moved to any MPGA I/O location.
CLKA **Clock A (ACT 2 and ACT 3 only)** TTL Clock input for ACT 2 and ACT 3 clock distribution networks. This pin can also be used as an I/O.	→	**no change** If desired, TTL Clock input signals may be moved to any MPGA I/O location.
CLKB **Clock B (ACT 2 and ACT 3 only)** TTL Clock input for ACT 2 and ACT 3 clock distribution networks. This pin can also be used as an I/O.	→	**no change** If desired, TTL Clock input signals may be moved to any MPGA I/O location.
DCLK **Diagnostic Clock** TTL Clock input for diagnostic probe and device programming. Function is controlled by the MODE pin.	→	**I/O** This pin is used as an I/O only. It is not used for diagnostic probe or device programming functions on an MPGA.
GND **Ground** LOW supply voltage.	→	**Ground** LOW supply voltage.
HCLK **Dedicated (Hard-wired) Array Clock (ACT 3 only)** TTL Clock input for ACT 3 sequential modules. This pin can also be used as an I/O.	→	**no change** If desired, TTL Clock input signals may be moved to any MPGA I/O location.
I/O **Input/Output** The I/O pin functions as an input, output, three-state, or bidirectional buffer. Unused pins are automatically driven LOW by the Designer software.	→	**I/O** User-defined MPGA I/O pins function identically to their FPGA counterparts. However, unused pins are NC (no connection) pins.
IOCLK **Dedicated (Hard-wired) I/O Clock (ACT 3 only)** TTL Clock input for ACT 3 I/O modules. This pin can also be used as an I/O.	→	**no change** If desired, TTL Clock input signals may be moved to any MPGA I/O location.
IOPCL **Dedicated (Hard-wired) I/O Preset/Clear (ACT 3 only)** TTL input for ACT 3 I/O preset or clear. This pin can also be used as an I/O.	→	**no change** If desired, this input signal may be moved to any MPGA I/O location.
MODE **Mode** The MODE pin controls the use of diagnostic pins (DCLK, PRA, PRB, SDI). When the MODE pin is HIGH, the special functions are active. When the MODE pin is LOW, the pins function as I/Os.	→	**TEST (No Connection)** This pin is reserved for parametric testing and should be connected to ground (LOW supply voltage).

Table 1 • FPGA-to-MPGA Pin Cross-Reference (continued)

FPGA Pin Description		MPGA Pin Description
NC **No Connection** This pin is not connected to circuitry within the device.	→	**NC** **No Connection** This pin is not connected to circuitry within the device.
PRA **Probe A** The Probe A pin is used for FPGA diagnostics. Function is controlled by the MODE pin.	→	**I/O** This pin is used as an I/O only. It is not used for diagnostic probe or device programming functions on an MPGA.
PRB **Probe B** The Probe B pin is used for FPGA diagnostics. Function is controlled by the MODE pin.	→	**I/O** This pin is used as an I/O only. It is not used for diagnostic probe or device programming functions on an MPGA.
SDI **Serial Data Input** Serial data input for diagnostic probe and device programming. Function is controlled by the MODE pin.	→	**I/O** This pin is used as an I/O only. It is not used for diagnostic probe or device programming functions on an MPGA.
V_{CC} **Supply Voltage** HIGH supply voltage.	→	**V_{CC}** HIGH supply voltage.
V_{KS} **Programming Voltage (ACT 2 and ACT 3 only)** Supply voltage used for device programming.	→	**Ground** LOW supply voltage.
V_{PP} **Programming Voltage (ACT 2 and ACT 3 only)** Supply voltage used for device programming.	→	**V_{CC}** HIGH supply voltage.
V_{SV} **Programming Voltage** Supply voltage used for device programming.	→	**V_{CC}** HIGH supply voltage.

MPGA Architecture

The Actel MPGA is built using a "sea-of-gates" architecture. A solid, regularly ordered array of transistors is overlaid with a multilevel metal interconnect. Surrounding this logic core is an array of programmable power and I/O pads. Separate grids provide power and ground supplies for the core logic and I/O cells.

The highly dense structure of Actel MPGAs provides for a cost-effective solution while maintaining the high performance of each particular design. This architecture reduces die size for low cost while minimizing gate length and shortening routing paths for excellent system performance. The robust power supply grids provide high I/O current drive without sacrificing high noise immunity. Since Actel FPGAs use a similar gate array architecture, design migration is a straightforward, simple process. Because of the advanced technology employed by the MPGA, the internal and external performance of each design is virtually assured to be preserved or improved after migration. To simplify migration further, the I/O pads are carefully arranged to allow FPGA pin assignments to be directly transferred to the full line of MPGA packages. For more information about the ease of design migration from Actel FPGAs to MPGAs, see the application note "Designing for Migration to Actel MPGAs."

Power Dissipation

The power dissipation for an Actel MPGA is composed of two parts: static power and active power. The static power is a product of the standby supply current (Icc) and the DC supply voltage (Vcc). Specifications for Icc and Vcc are located in the "Electrical Specifications" section of this data sheet. The active power is a product of equivalent capacitance, square of the DC supply voltage, and average switching frequency of the circuit. It is expressed in the formula

$$\text{Power } (\mu W) = C_{EQ} \bullet V_{CC}^2 \bullet f$$

where

C_{EQ} is the equivalent capacitance in picofarads (pF)

V_{CC} is the DC supply voltage in volts (V)

f is the switching frequency in megahertz (MHz)

Upon receipt of the "Design Start Checklist" and associated materials, Actel calculates the MPGA active power

dissipation for each design based on this formula. This calculation is immediately relayed to you so that you can update system power specifications accordingly. Typically, power dissipation of an Actel design is significantly lower for the MPGA version versus the FPGA version.

Timing Characteristics

The timing characteristics for Actel MPGA devices are consistent across family and device types. Typical I/O buffer, internal logic cell, and internal routing delays are common to all MPGA devices. The advanced technology of the devices ensures converted designs meet or exceed FPGA performance. Refer to the MPGA Timing Model diagram and Timing Characteristics chart for detailed timing and delay estimates.

Timing Derating

Timing derating factors due to temperature, voltage, and process variations are summarized in the following tables and graphs. Use these derating factors to determine device performance at any particular condition within the electrical and environmental specifications.

MPGA devices are manufactured in a CMOS process. Therefore, device performance varies according to temperature, voltage, and process variations. Minimum timing parameters reflect maximum operating voltage, minimum operating temperature, and best-case processing. Maximum timing parameters reflect minimum operating voltage, maximum operating temperature, and worst-case processing.

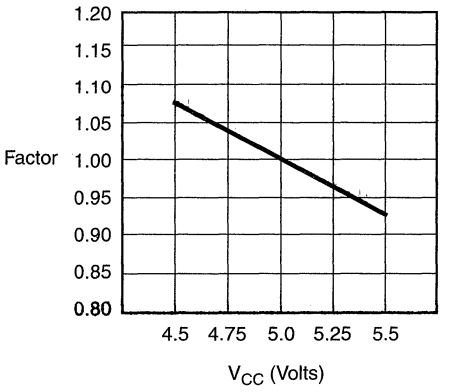
Timing Derating Factor, Temperature and Voltage

	Industrial	
	Minimum	Maximum
(Commercial Minimum/Maximum Specification) x	0.85	1.07

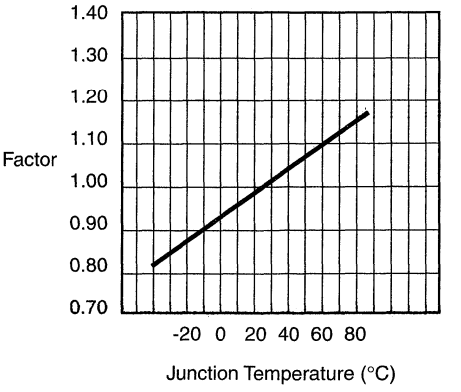
Timing Derating Factor for Designs at Typical Temperature (T_J = 25°C) and Voltage (V_{CC} = 5.0 V)

(Commercial Maximum Specification) x	0.86

Voltage Derating Curve

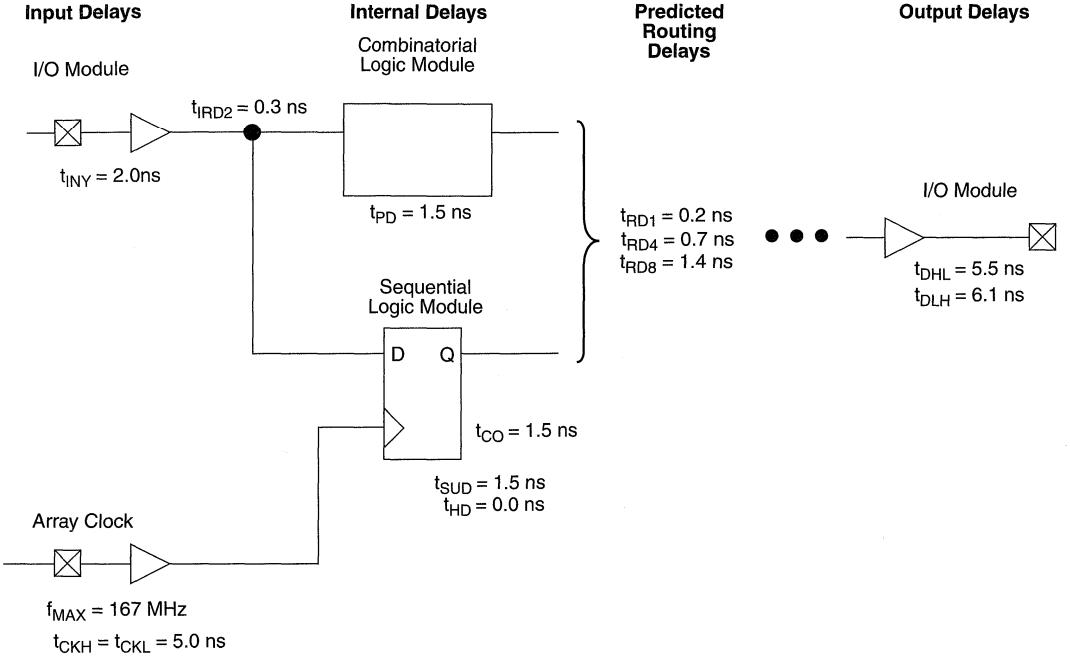


Temperature Derating Curve

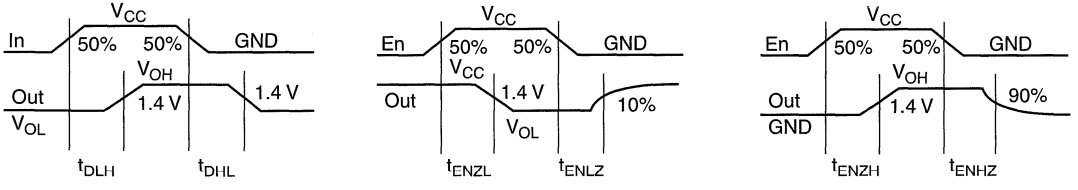


Note: This derating factor applies to all routing and propagation delays.

MPGA Timing Model

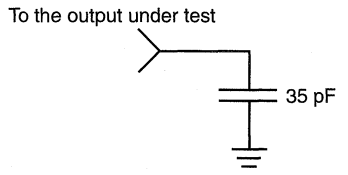


Output Buffer Delays

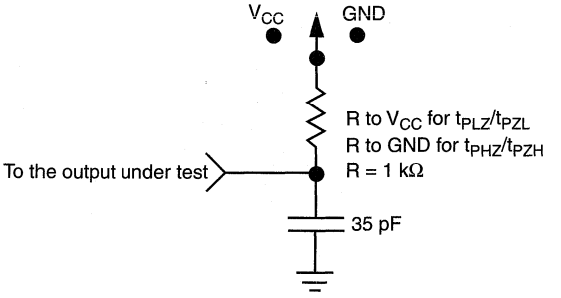


AC Test Loads

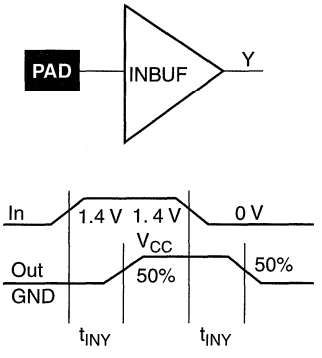
Load 1
(Used to measure propagation delay)



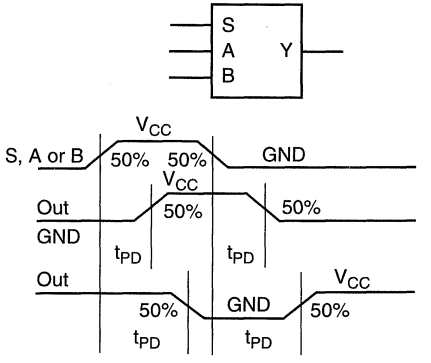
Load 2
(Used to measure rising/falling edges)



Input Buffer Delays

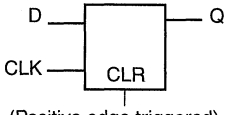


Module Delays

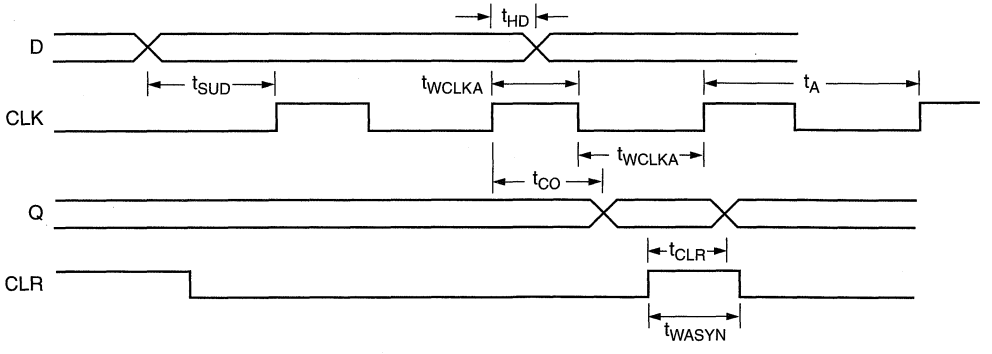


Sequential Module Timing Characteristics

Flip-Flops



(Positive edge triggered)



MPGA Timing Characteristics

(Worst-Case Commercial Conditions, V_{CC} = 4.75 V, T_J = 70°C)

		Preliminary Information		
Logic Module Propagation Delays				
Parameter	**Description**	**Min.**	**Max.**	**Units**
t_{PD}	Internal Array Module		1.5	ns
t_{CO}	Sequential Clock to Q		1.5	ns
t_{CLR}	Asynchronous Clear to Q		1.5	ns
Predicted Routing Delays[1]				
t_{RD1}	FO=1 Routing Delay		0.2	ns
t_{RD2}	FO=2 Routing Delay		0.3	ns
t_{RD3}	FO=3 Routing Delay		0.5	ns
t_{RD4}	FO=4 Routing Delay		0.7	ns
t_{RD8}	FO=8 Routing Delay		1.4	ns
Logic Module Sequential Timing				
t_{SUD}	Flip-Flop Data Input Setup	1.5		ns
t_{HD}	Flip-Flop Data Input Hold	0.0		ns
t_{SUD}	Latch Data Input Setup	1.5		ns
t_{HD}	Latch Data Input Hold	0.0		ns
t_{WASYN}	Asynchronous Pulse Width	2.0		ns
t_{WCLKA}	Flip-Flop Clock Pulse Width	2.0		ns
t_A	Flip-Flop Clock Input Period	8.0		ns
f_{MAX}	Flip-Flop Clock Frequency		125	MHz
I/O Module Input Propagation Delay				
t_{INY}	Input Data Pad to Y		2.0	ns
Predicted Input Routing Delays[1]				
t_{IRD1}	FO=1 Routing Delay		0.2	ns
t_{IRD2}	FO=2 Routing Delay		0.3	ns
t_{IRD3}	FO=3 Routing Delay		0.5	ns
t_{IRD4}	FO=4 Routing Delay		0.7	ns
t_{IRD8}	FO=8 Routing Delay		1.4	ns

Note:

1. *Routing delays are for typical designs across worst-case operating conditions. These parameters should be used for estimating device performance. Postroute timing analysis or simulation is required to determine actual worst-case performance. Postroute timing is based on actual routing delay measurements performed on the device prior to shipment.*

MPGA Timing Characteristics (continued)

(Worst-Case Commercial Conditions)

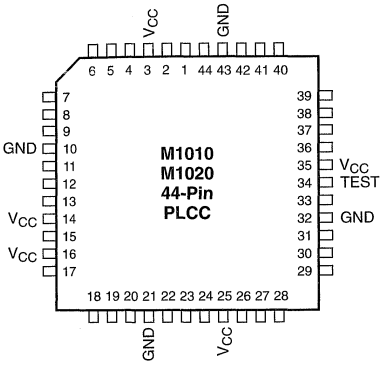
		Preliminary Information		
I/O Module – TTL Output Timing[1]				
Parameter	**Description**	**Min.**	**Max.**	**Units**
t_{DHL}	Data to Pad, High to Low		6.8	ns
t_{DLH}	Data to Pad, Low to High		3.9	ns
t_{ENZH}	Enable to Pad, Z to High		4.5	ns
t_{ENZL}	Enable to Pad, Z to Low		6.8	ns
t_{ENHZ}	Enable to Pad, High to Z		3.8	ns
t_{ENLZ}	Enable to Pad, Low to Z		2.0	ns
d_{TLH}	Delta Low to High		0.05	ns/pF
d_{THL}	Delta High to Low		0.09	ns/pF
I/O Module – CMOS Output Timing[1]				
t_{DHL}	Data to Pad, High to Low		5.5	ns
t_{DLH}	Data to Pad, Low to High		6.1	ns
t_{ENZH}	Enable to Pad, Z to High		6.7	ns
t_{ENZL}	Enable to Pad, Z to Low		5.6	ns
t_{ENHZ}	Enable to Pad, High to Z		3.8	ns
t_{ENLZ}	Enable to Pad, Low to Z		2.0	ns
d_{TLH}	Delta Low to High		0.09	ns/pF
d_{THL}	Delta High to Low		0.07	ns/pF
Global Clock Networks (for Fanout = 1000)				
t_{CKH}	Input Low to High		5.0	ns
t_{CKL}	Input High to Low		5.0	ns
t_{PWH}	Min. Pulse Width High	2.9		ns
t_{PWL}	Min. Pulse Width Low	2.9		ns
t_{CKSW}	Maximum Skew		0.4	ns
t_P	Minimum Period	6.0		ns
f_{MAX}	Maximum Frequency		167	MHz

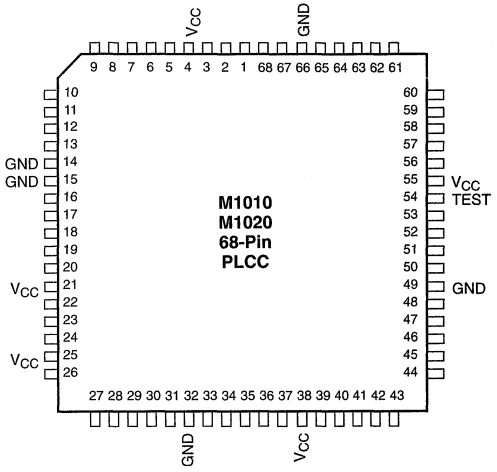
Note:

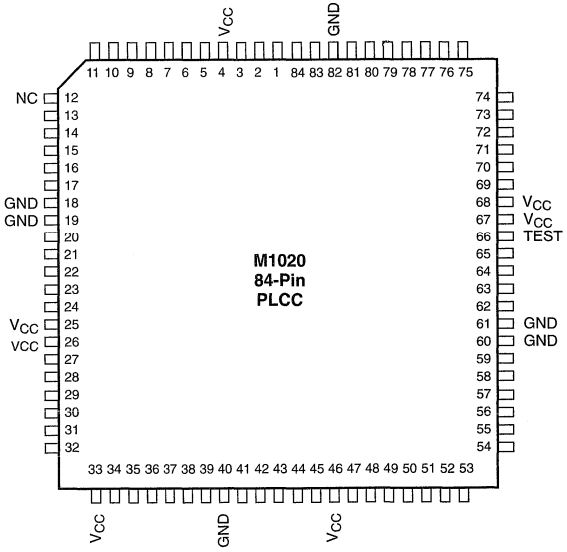
1. Delays based on 35pF loading.

Package Pin Assignments—ACT 1

PLCC (Top View)





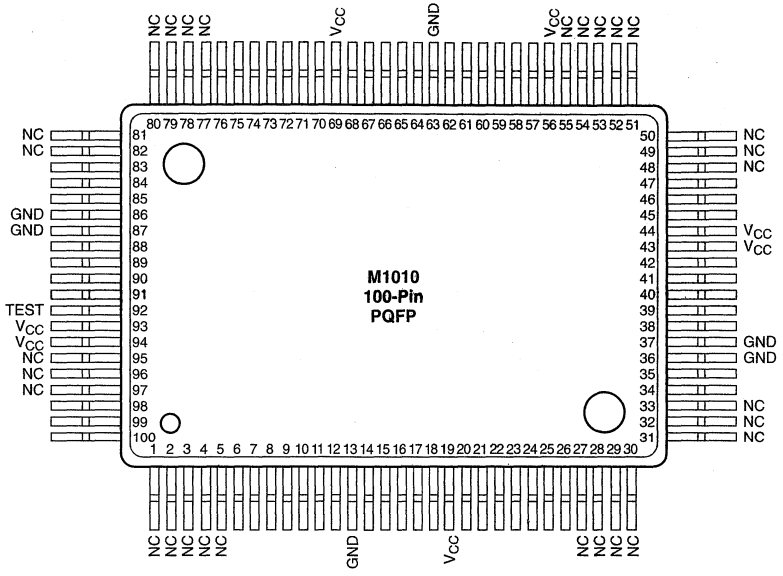


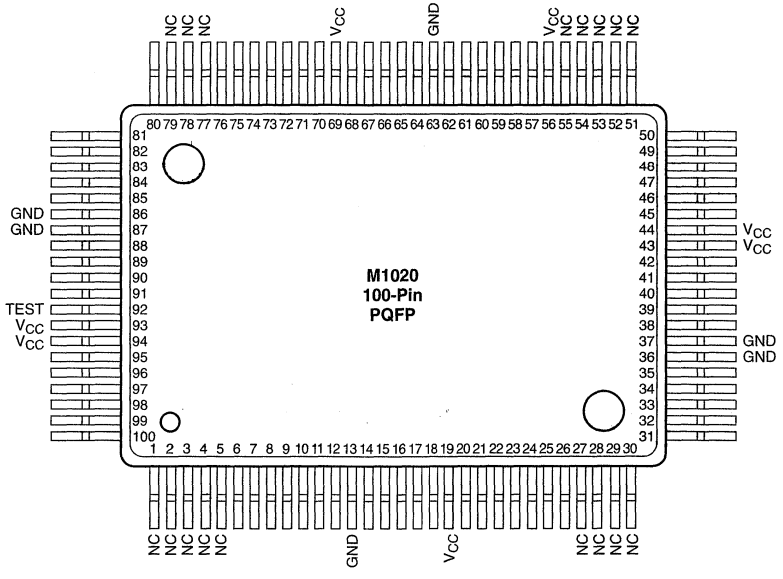
Notes:

1. *All unassigned pins are available for use as I/Os.*
2. *TEST pin used for device testing only. See Pin Description for details.*

Package Pin Assignments—ACT 1 (continued)

PQFP (Top View)



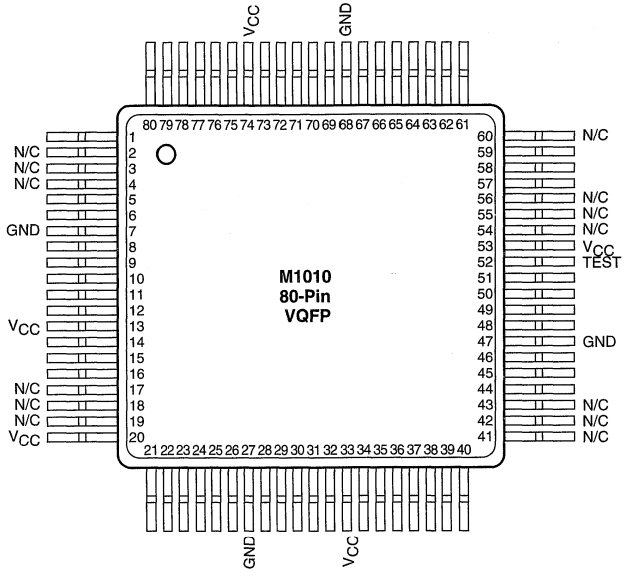


Notes:

1. *All unassigned pins are available for use as I/Os.*
2. *TEST pin used for device testing only. See Pin Description for details.*

Package Pin Assignments—ACT 1 (continued)

VQFP (Top View)

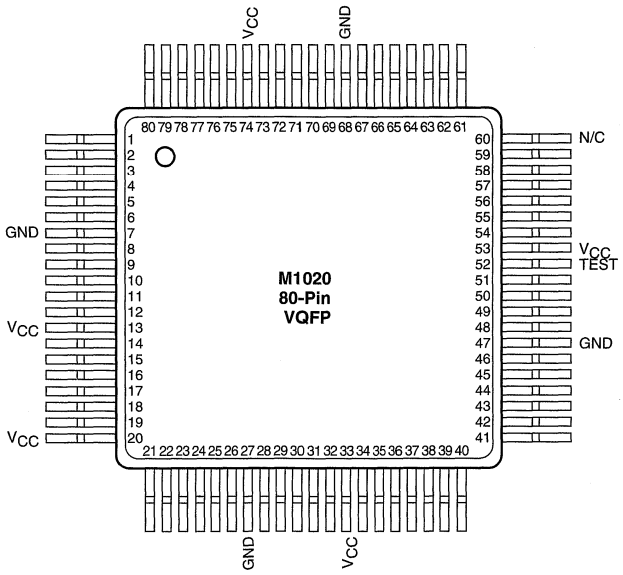


Notes:

1. *All unassigned pins are available for use as I/Os.*
2. *TEST pin used for device testing only. See Pin Description for details.*

Package Pin Assignments—ACT 1 (continued)

VQFP (Top View)

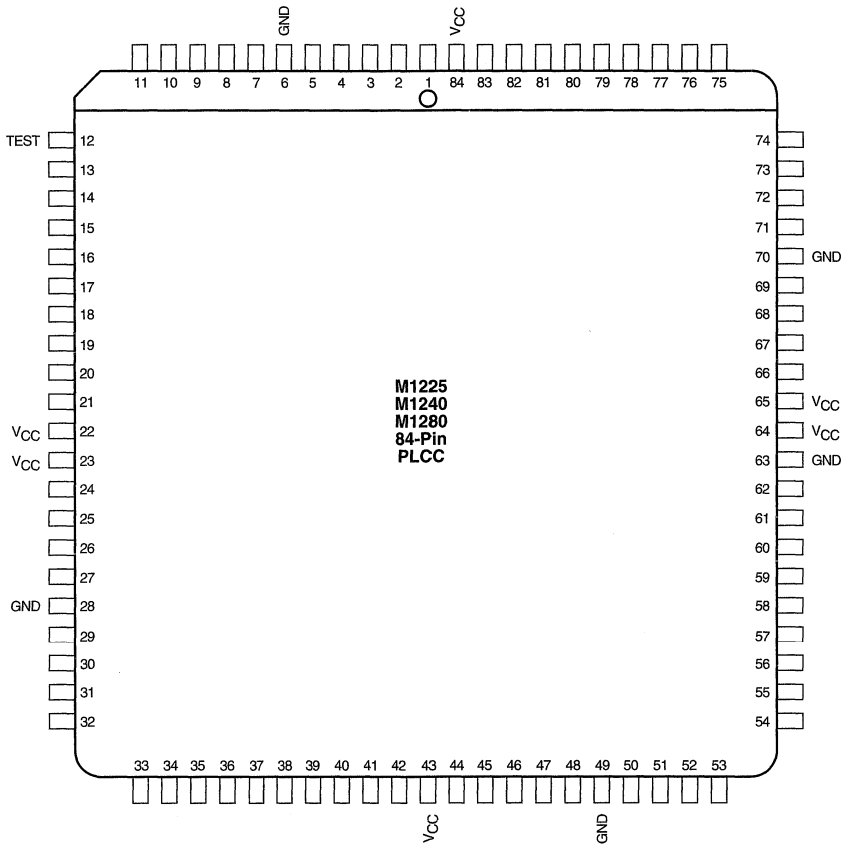


Notes:

1. *All unassigned pins are available for use as I/Os.*
2. *TEST pin used for device testing only. See Pin Description for details.*

Package Pin Assignments—ACT 2

PLCC (Top View)

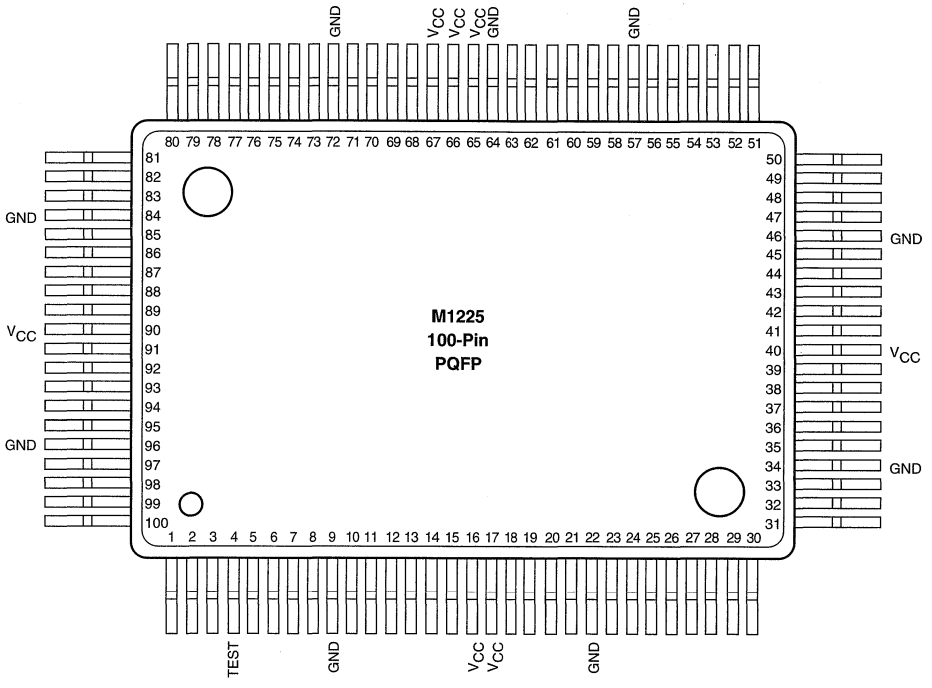


Notes:

1. *All unassigned pins are available for use as I/Os.*
2. *TEST pin used for device testing only. See Pin Description for details.*

Package Pin Assignments—ACT 2 (continued)

PQFP (Top View)

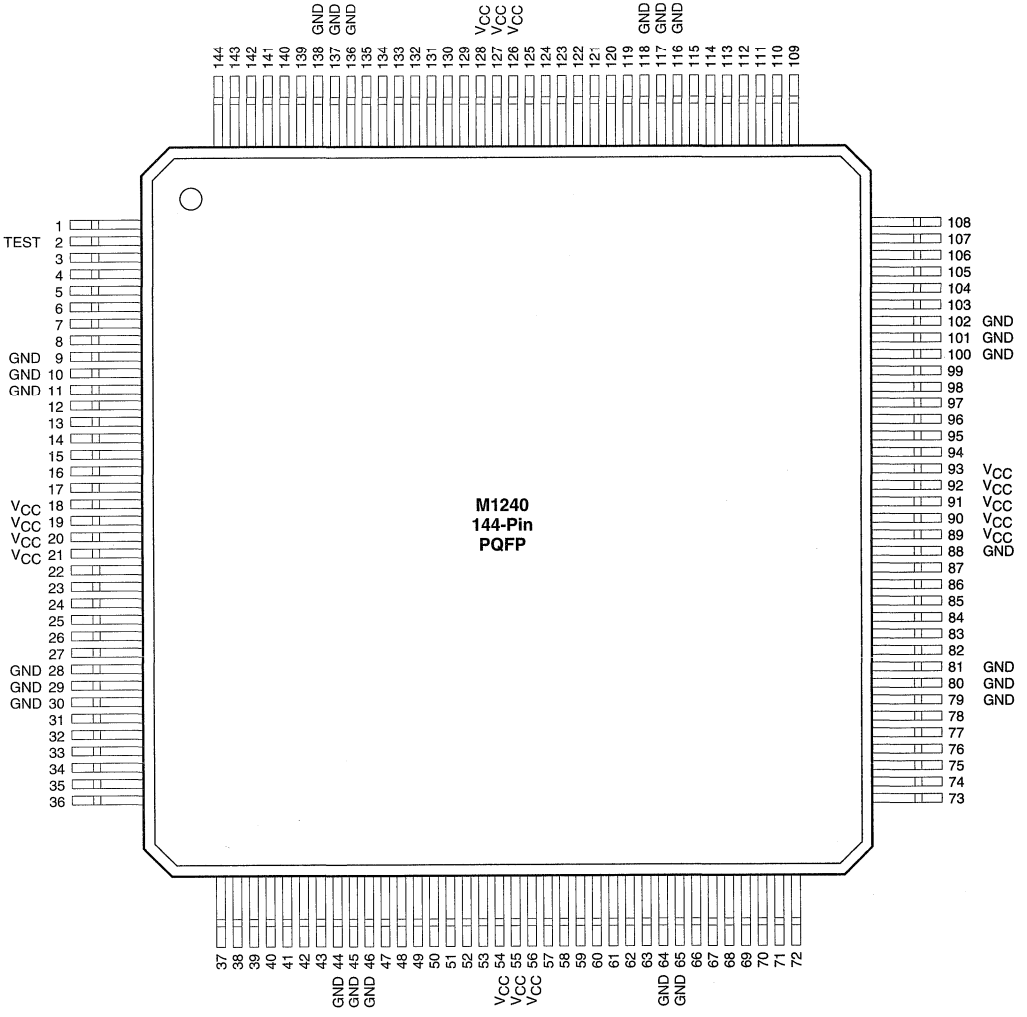


Notes:

1. *All unassigned pins are available for use as I/Os.*
2. *TEST pin used for device testing only. See Pin Description for details.*

Package Pin Assignments—ACT 2 (continued)

PQFP (Top View)

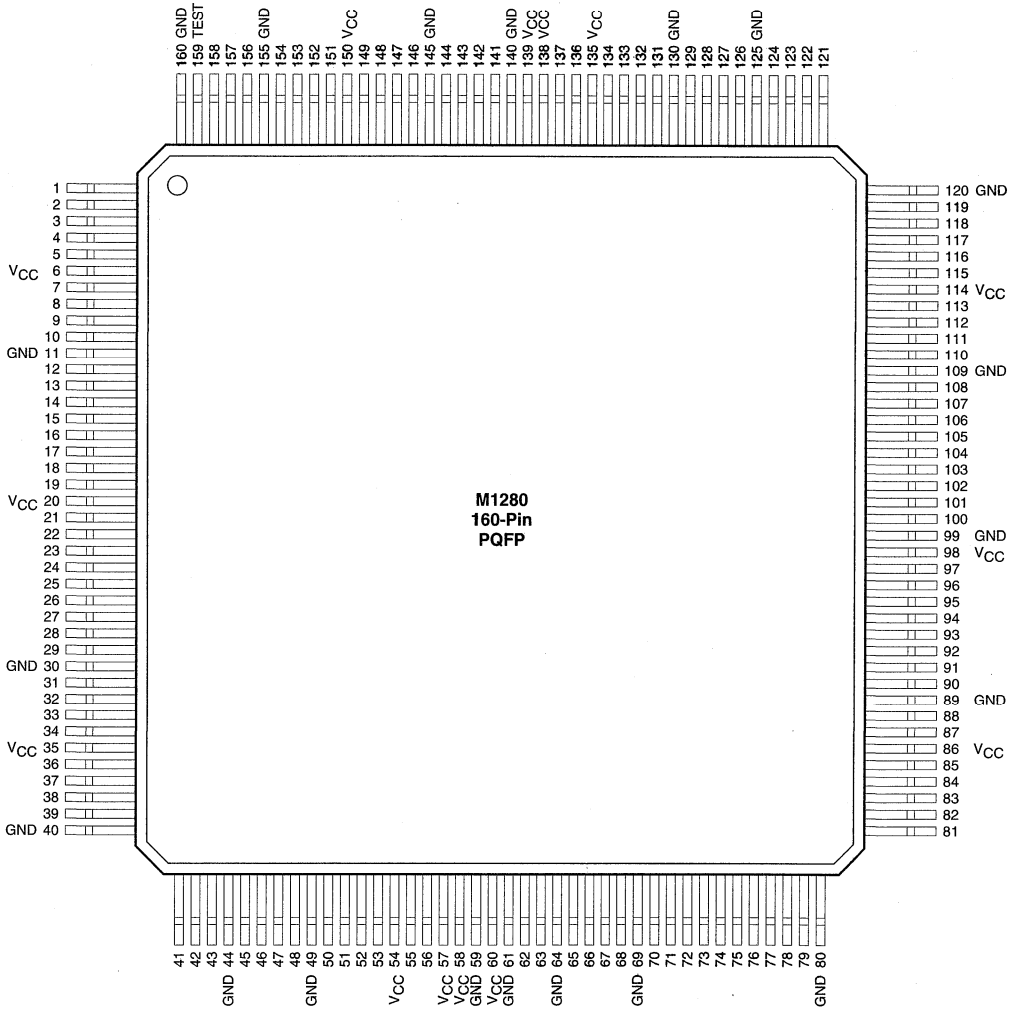


Notes:

1. *All unassigned pins are available for use as I/Os.*
2. *TEST pin used for device testing only. See Pin Description for details.*

Package Pin Assignments—ACT 2 (continued)

PQFP (Top View)

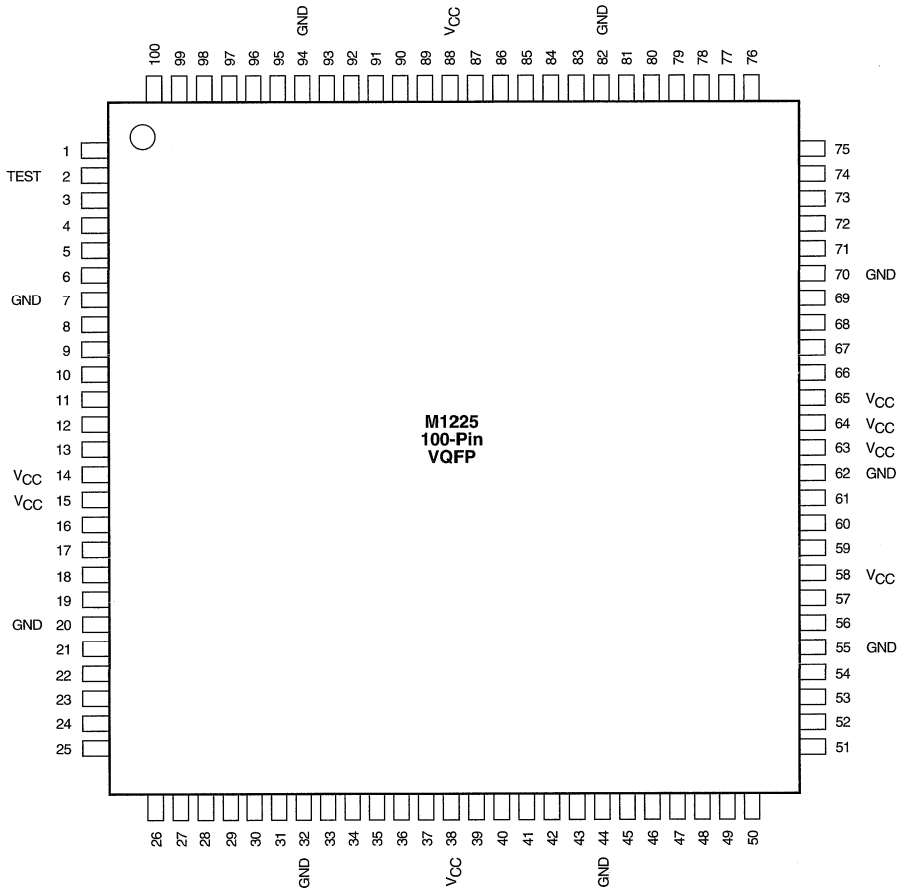


Notes:

1. *All unassigned pins are available for use as I/Os.*
2. *TEST pin used for device testing only. See Pin Description for details.*

Package Pin Assignments—ACT 2 (continued)

VQFP (Top View)

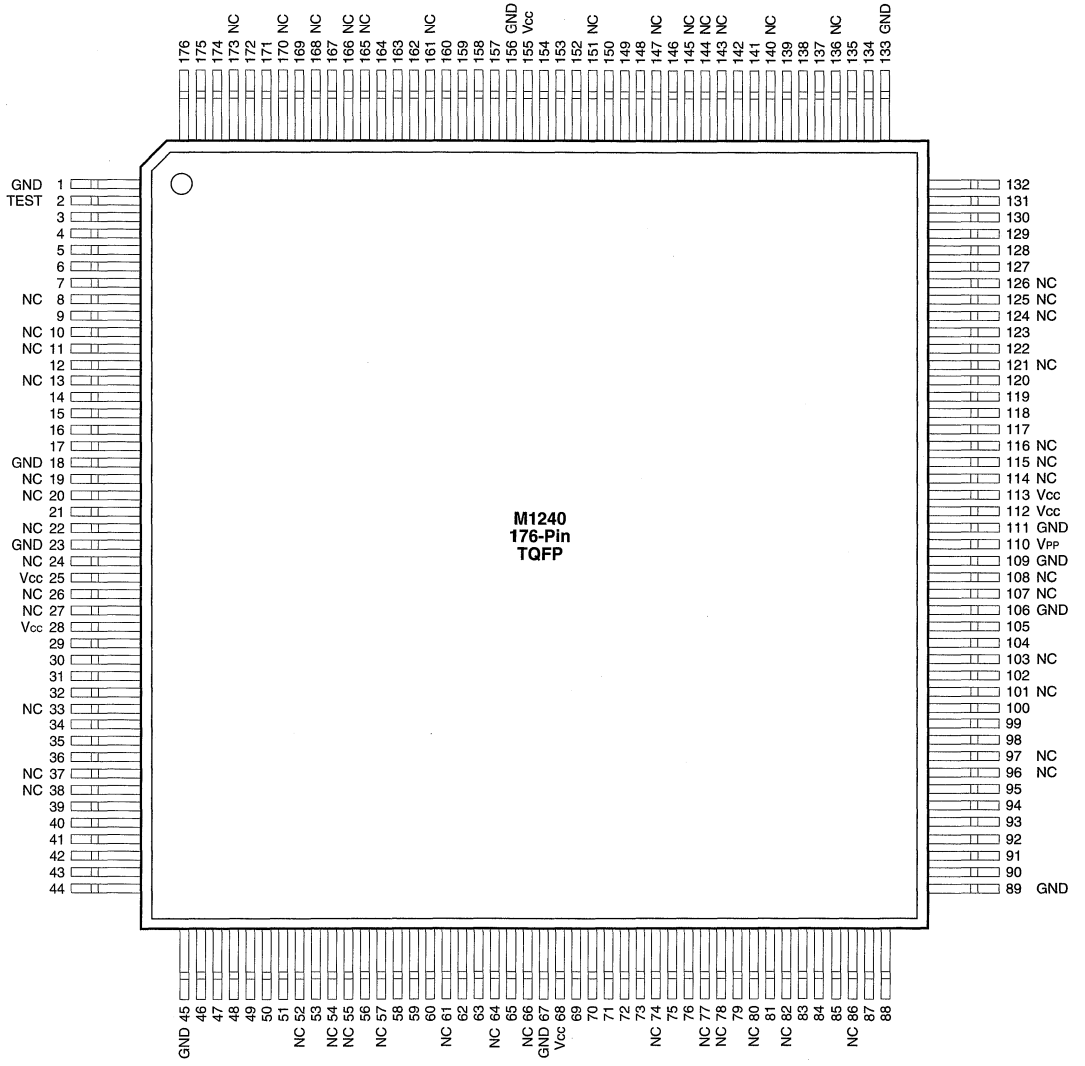


Notes:

1. *All unassigned pins are available for use as I/Os.*
2. *TEST pin used for device testing only. See Pin Description for details.*

Package Pin Assignments—ACT 2 (continued)

TQFP (Top View)

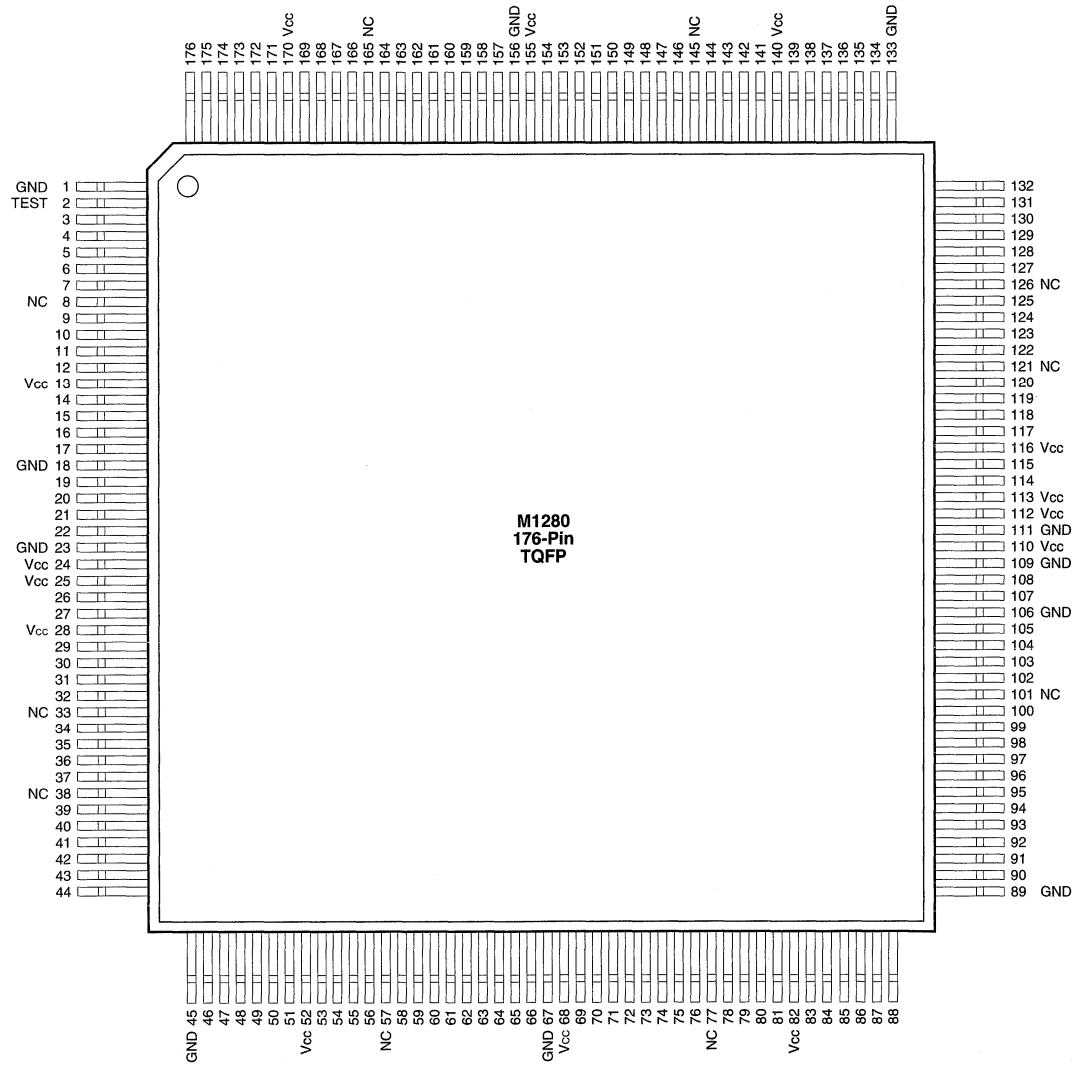


Notes:

1. *All unassigned pins are available for use as I/Os.*
2. *TEST pin used for device testing only. See Pin Description for details.*

Package Pin Assignments—ACT 2 (continued)

TQFP (Top View)

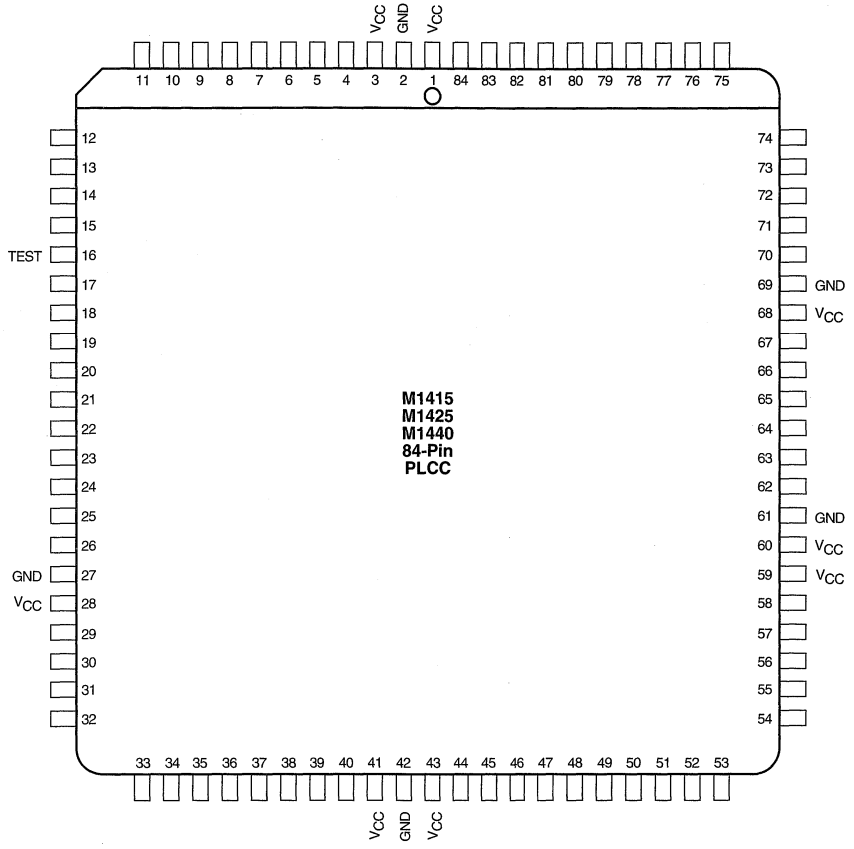


Notes:

1. *All unassigned pins are available for use as I/Os.*
2. TEST pin used for device testing only. See Pin Description for details.

Package Pin Assignments—ACT 3

PLCC (Top View)

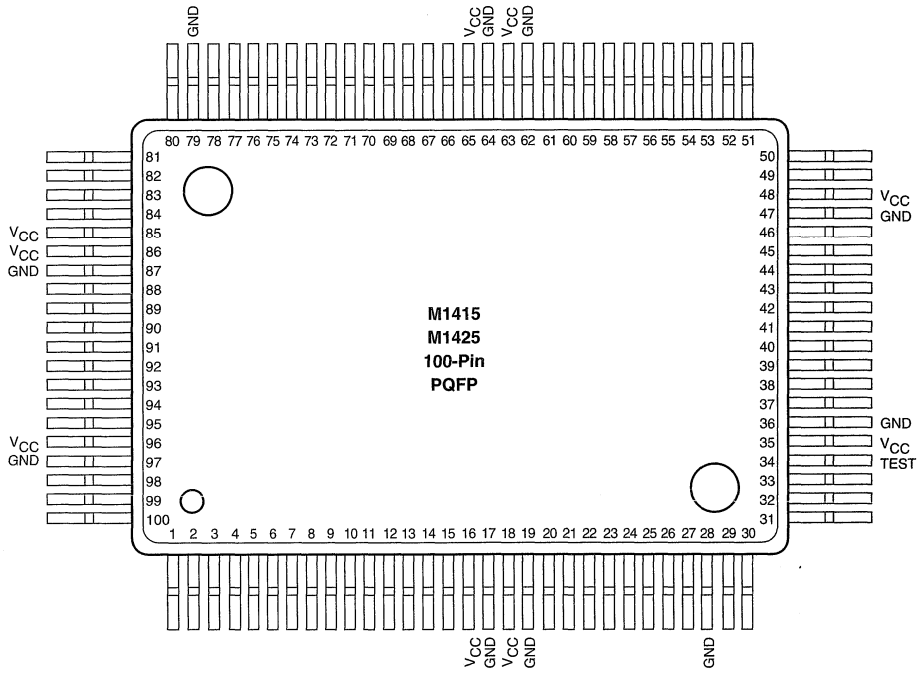


Notes:

1. *All unassigned pins are available for use as I/Os.*
2. *TEST pin used for device testing only. See Pin Description for details.*

Package Pin Assignments—ACT 3 (continued)

PQFP (Top View)

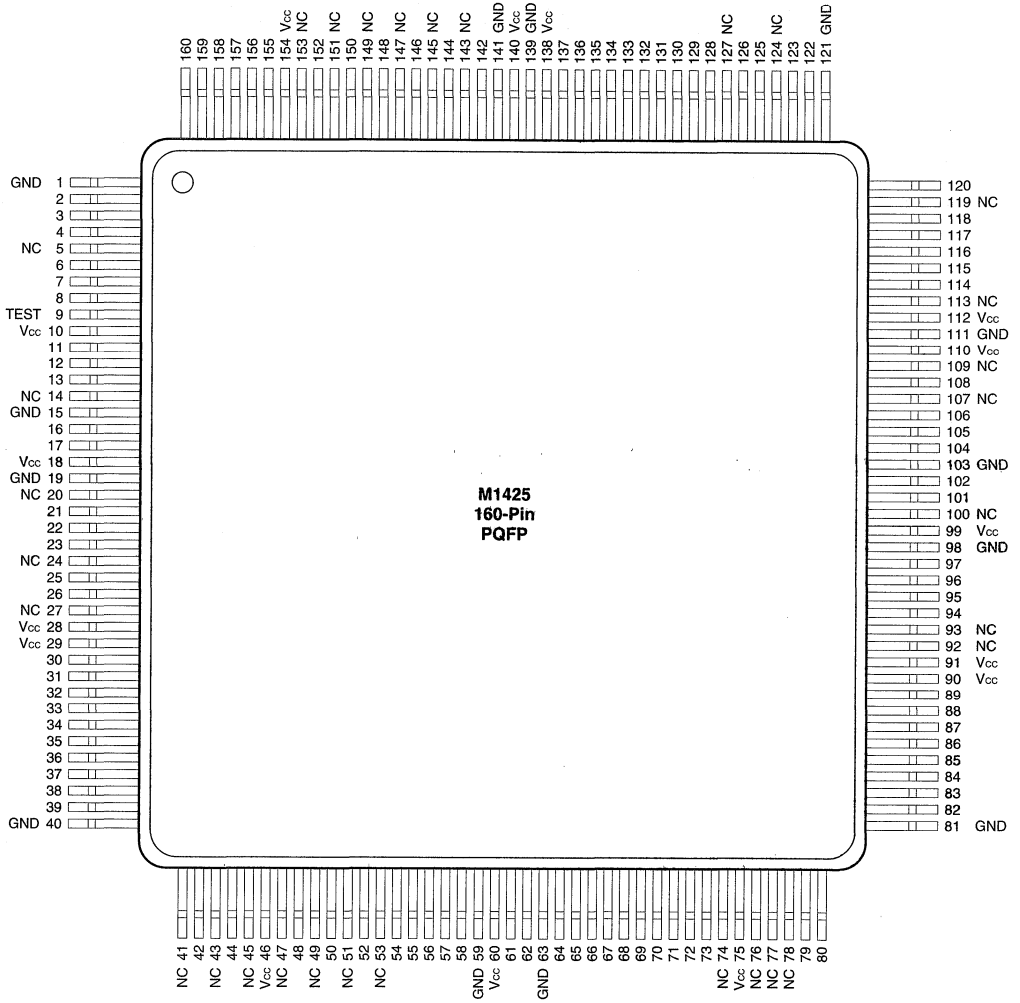


Notes:

1. *All unassigned pins are available for use as I/Os.*
2. *TEST pin used for device testing only. See Pin Description for details.*

Package Pin Assignments—ACT 3 (continued)

PQFP (Top View)

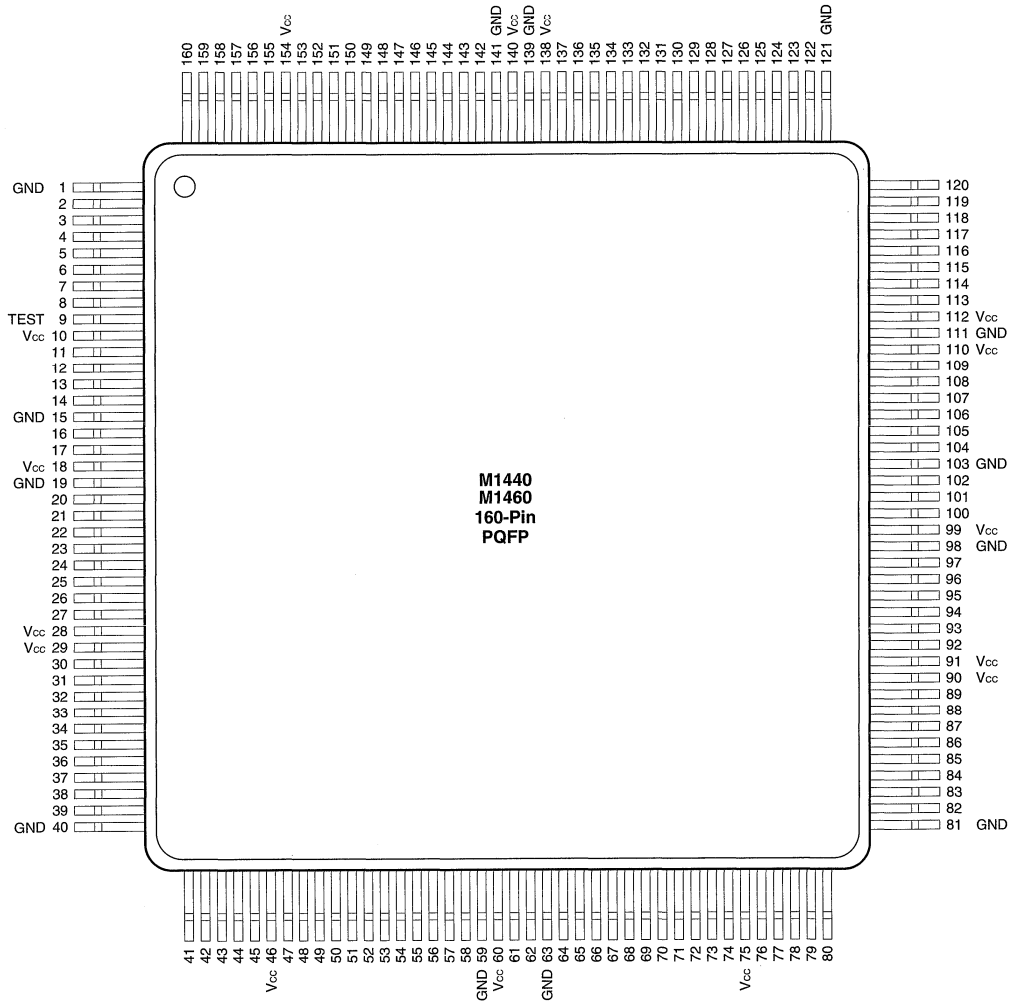


Notes:

1. *All unassigned pins are available for use as I/Os.*
2. *TEST pin used for device testing only. See Pin Description for details.*

Package Pin Assignments—ACT 3 (continued)

PQFP (Top View)

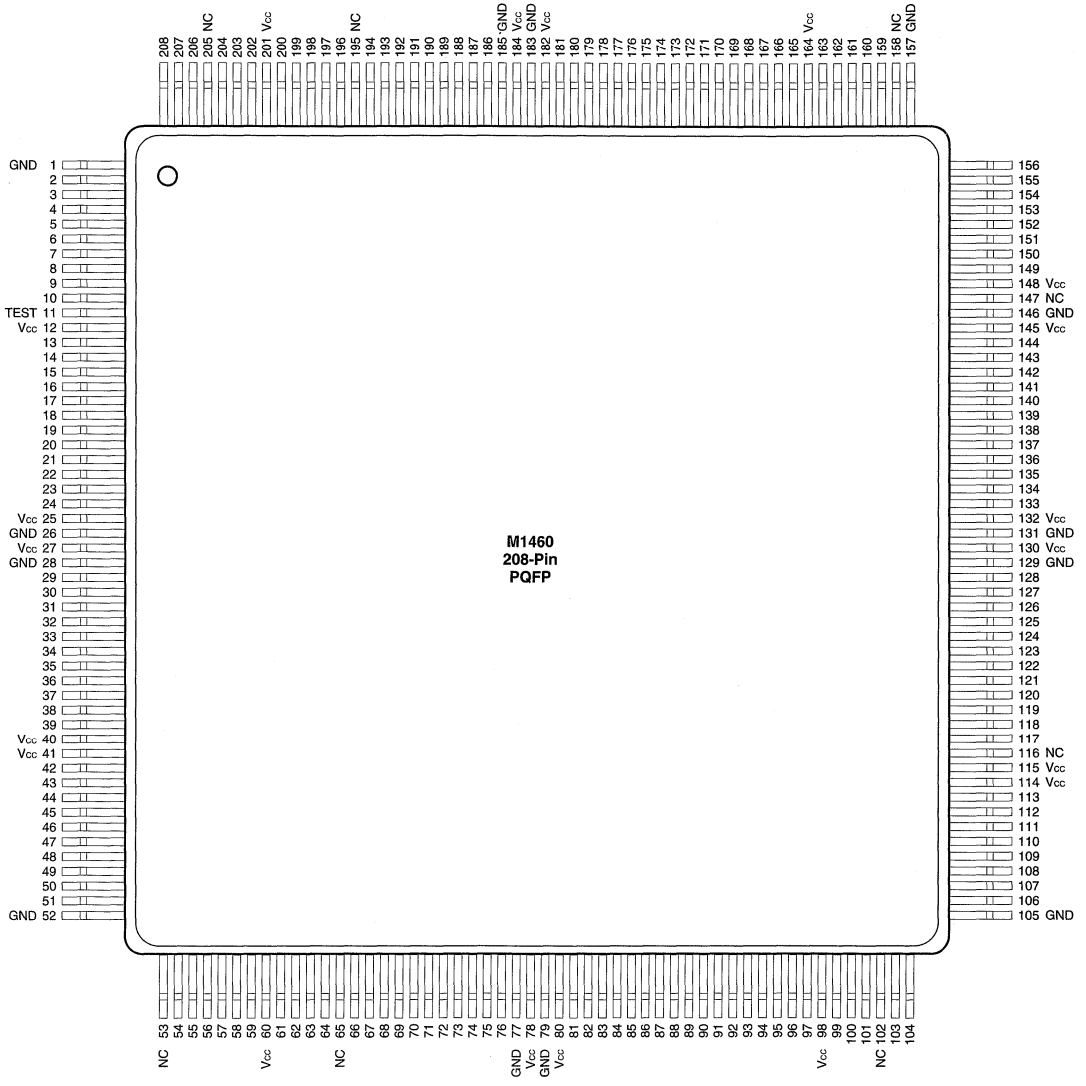


Notes:

1. *All unassigned pins are available for use as I/Os.*
2. *TEST pin used for device testing only. See Pin Description for details.*

Package Pin Assignments—ACT 3 (continued)

PQFP (Top View)

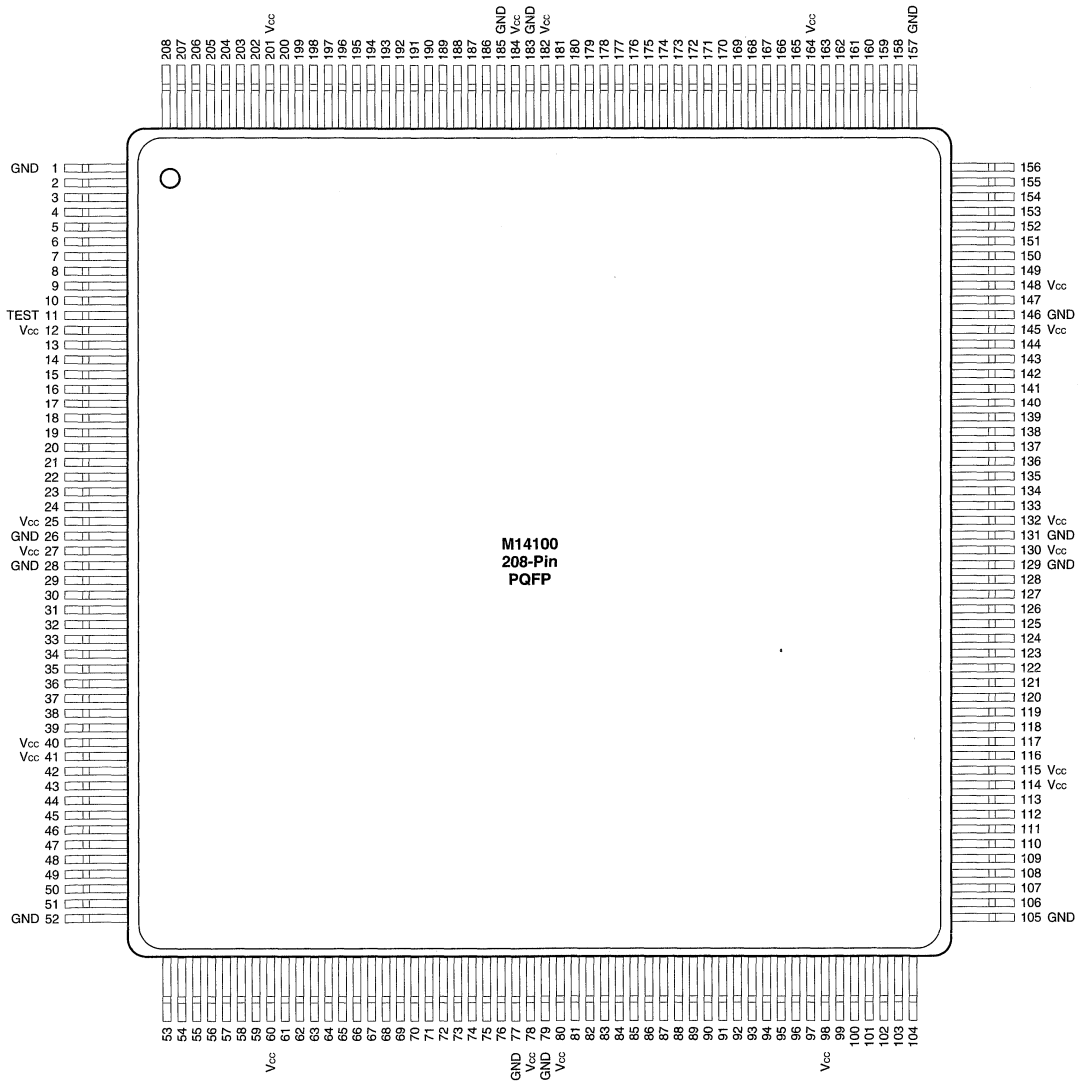


Notes:

1. *All unassigned pins are available for use as I/Os.*
2. *TEST pin used for device testing only. See Pin Description for details.*

Package Pin Assignments—ACT 3 (continued)

PQFP (Top View)

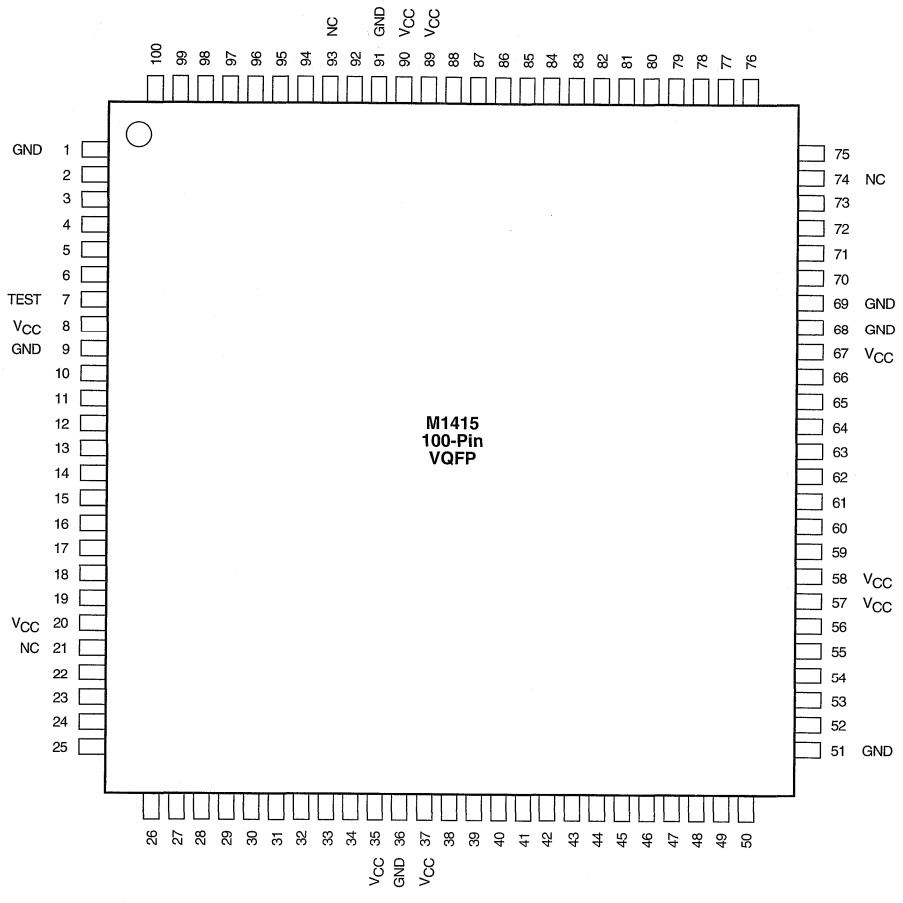


Notes:

1. *All unassigned pins are available for use as I/Os.*
2. *TEST pin used for device testing only. See Pin Description for details.*

Package Pin Assignments—ACT 3 (continued)

VQFP (Top View)

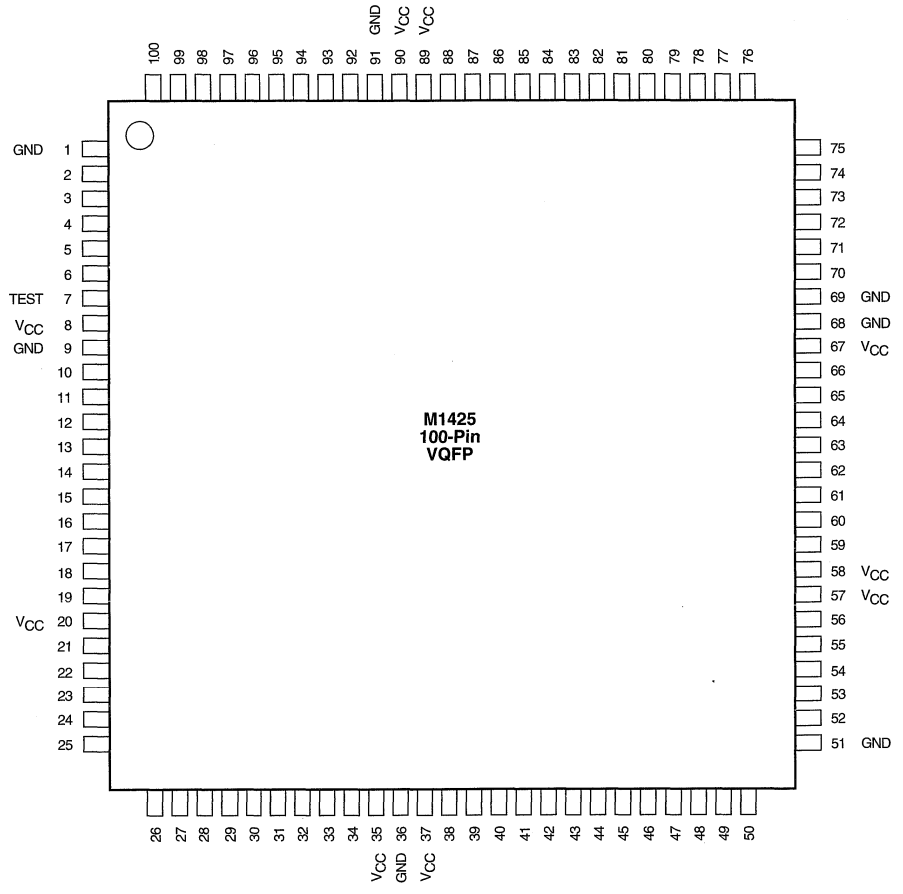


Notes:

1. *All unassigned pins are available for use as I/Os.*
2. *TEST pin used for device testing only. See Pin Description for details.*

Package Pin Assignments—ACT 3 (continued)

VQFP (Top View)

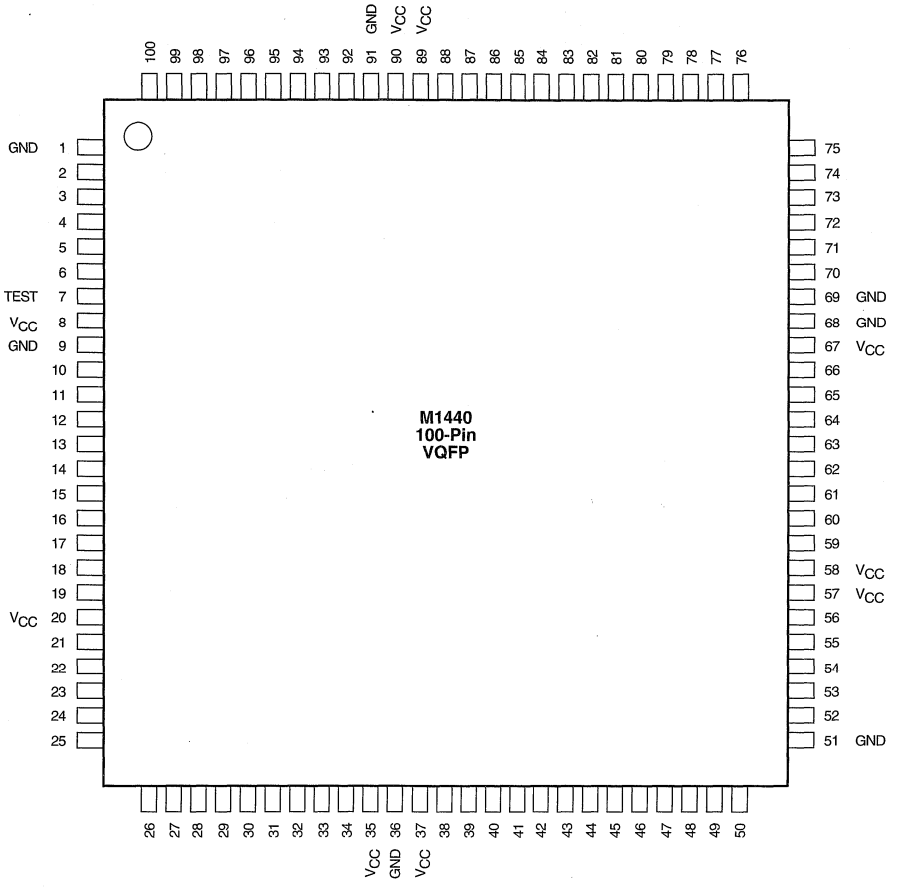


Notes:

1. *All unassigned pins are available for use as I/Os.*
2. *TEST pin used for device testing only. See Pin Description for details.*

Package Pin Assignments—ACT 3 (continued)

VQFP (Top View)

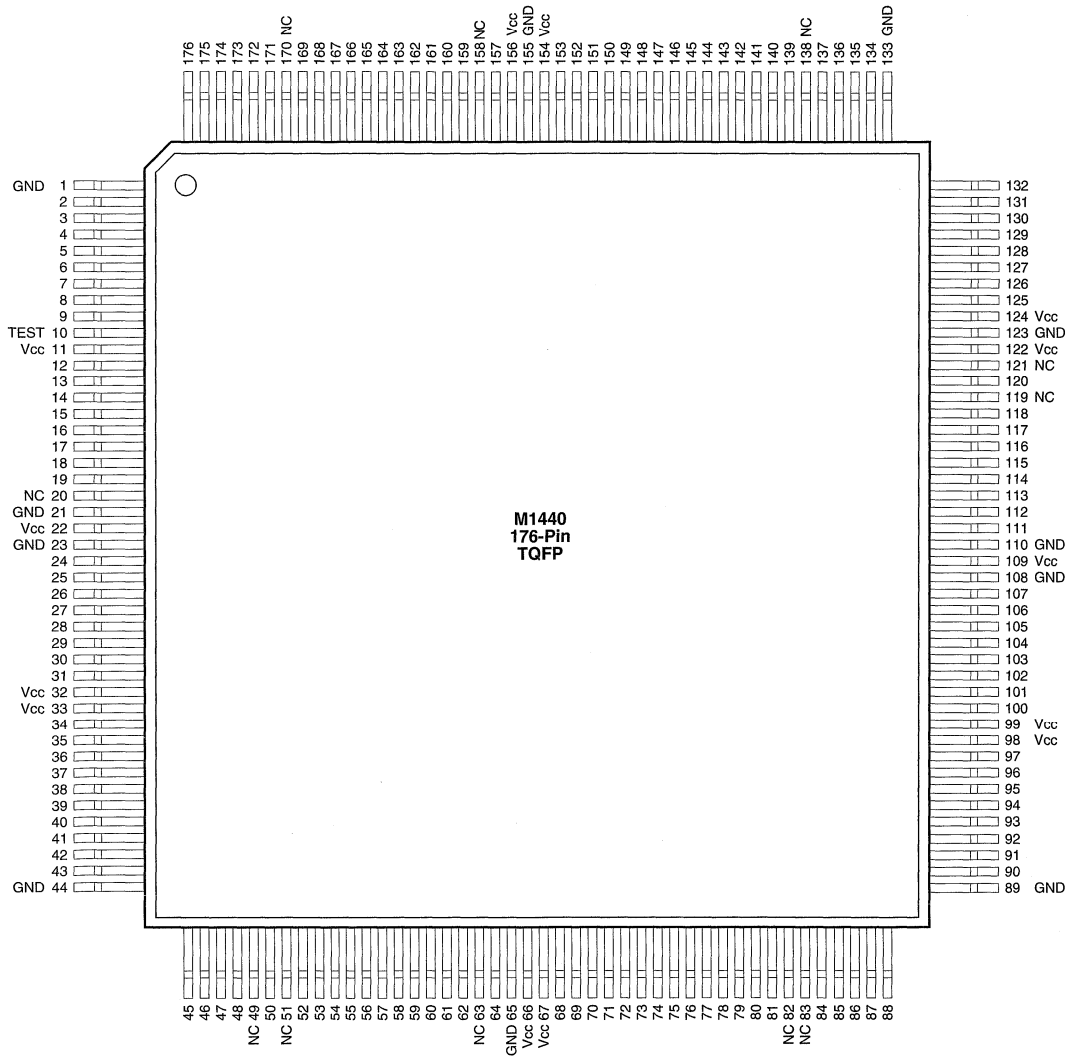


Notes:

1. *All unassigned pins are available for use as I/Os.*
2. *TEST pin used for device testing only. See Pin Description for details.*

Package Pin Assignments—ACT 3 (continued)

TQFP (Top View)

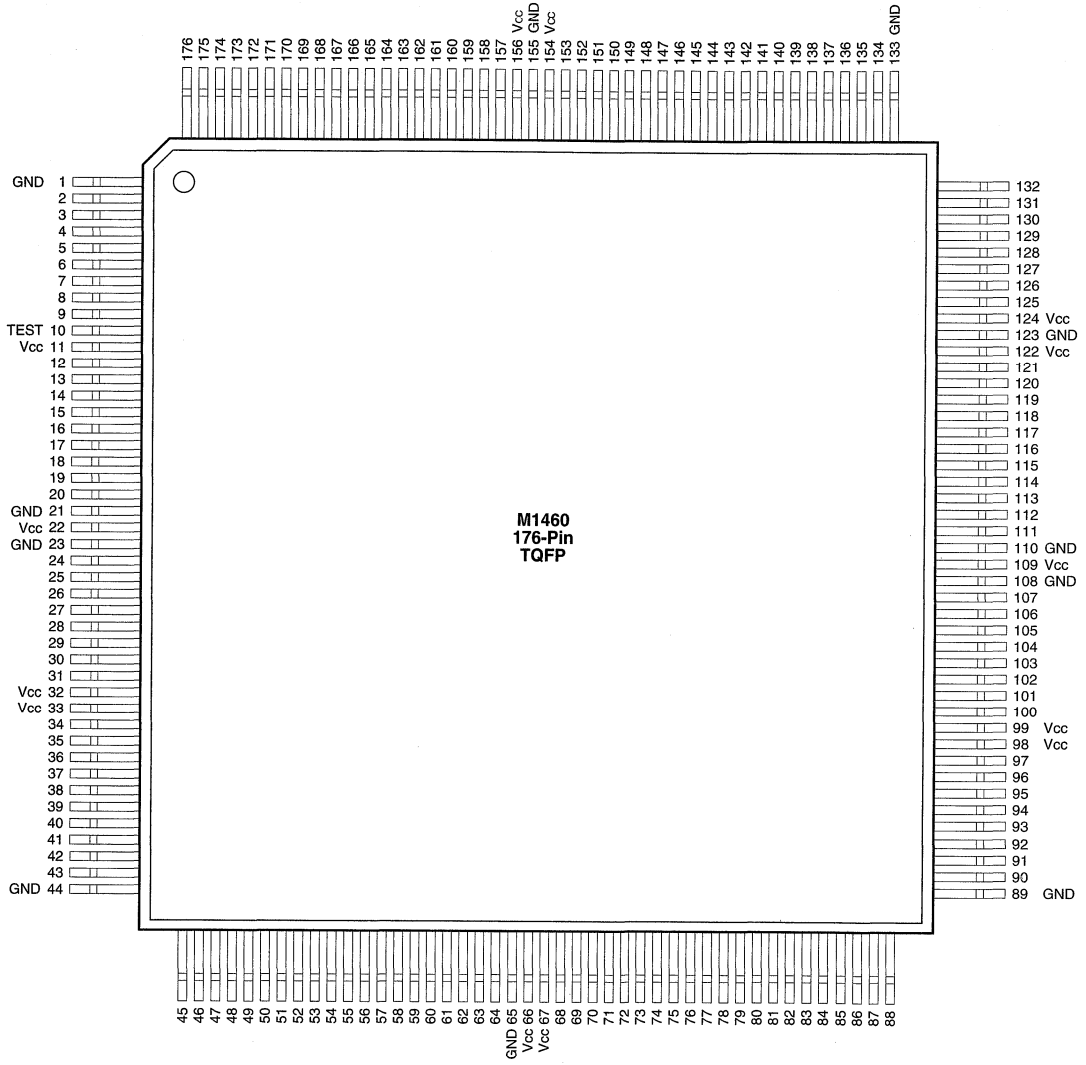


Notes:

1. *All unassigned pins are available for use as I/Os.*
2. *TEST pin used for device testing only. See Pin Description for details.*

Package Pin Assignments—ACT 3 (continued)

TQFP (Top View)



Notes:

1. *All unassigned pins are available for use as I/Os.*
2. *TEST pin used for device testing only. See Pin Description for details.*

Package and Mechanical Drawings

Section 2: Package and Mechanical Drawings

Package Options: User I/Os per Package

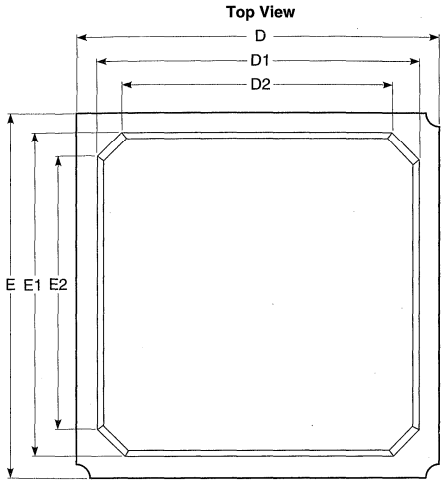
Package	Pins	ACT 1		ACT 2			1200XL			ACT 3				
		A1010B	A1020B	A1225A	A1240A	A1280A	A1225A	A1240A	A1280A	A1415A	A1425A	A1440A	A1460A	A14100A
PLCC	44	34	34											
	68	57	57											
	84		69	72	72	72	72	72	72	70	70	70		
PQFP	100	57	69	83			83			80	80			
	144				104			104						
	160					125			125		100	131	131	
	208												167	
RQFP	208													175
VQFP	80	57	69											
	100			83			83			80	83	83		
TQFP	176				104	140		104	140			140	151	
BGA	225												168	
	313													228
CPGA	84	57	69											
	100			83			83			80				
	132/133				104			104			100			
	175/176					140			140			140		
	207												168	
	257													228
CQFP	84		69											
	132										100			
	172					140			140					
	196												168	
	256													228

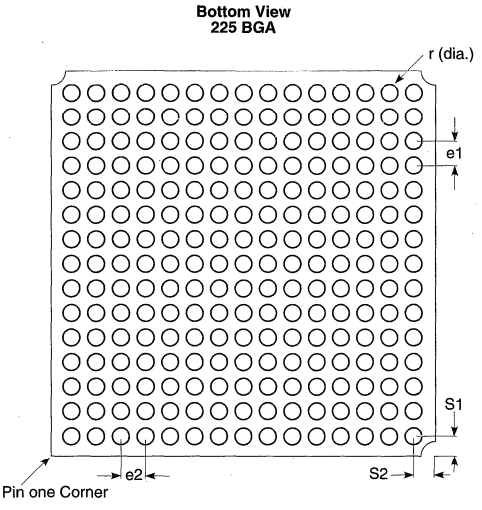
Package Thermal Characteristics

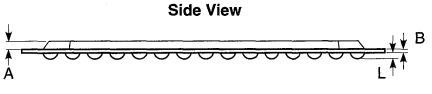
Package Type		Pin Count	Θ_{JA} Still Air	Θ_{JA} 300 ft/min	Units
Plastic Leaded Chip Carrier	PLCC	44	52	40	°C/W
		68	45	35	°C/W
		84	44	38	°C/W
Plastic Quad Flatpack	PQFP	100	55	47	°C/W
		144	35	26	°C/W
		160	33	26	°C/W
		208	33	26	°C/W
Plastic Power Quad Flatpack	RQFP	208	17	13	°C/W
Very Thin (1.0mm) Quad Flatpack	VQFP	80	68	55	°C/W
		100	43	35	°C/W
Thin (1.4mm) Quad Flatpack	TQFP	176	32	25	°C/W
Ball Grid Array	BGA	225	25	21	°C/W
		313	23	18	°C/W
Ceramic Pin Grid Array	CPGA	84	33	20	°C/W
		100	35	17	°C/W
		132	30	15	°C/W
		133	30	15	°C/W
		175	25	14	°C/W
		176	23	12	°C/W
		207	22	13	°C/W
		256	15	8	°C/W
Ceramic Quad Flatpack	CQFP	84	40	30	°C/W
		132	55	30	°C/W
		172	25	15	°C/W
		196	36	24	°C/W
		256	30	18	°C/W

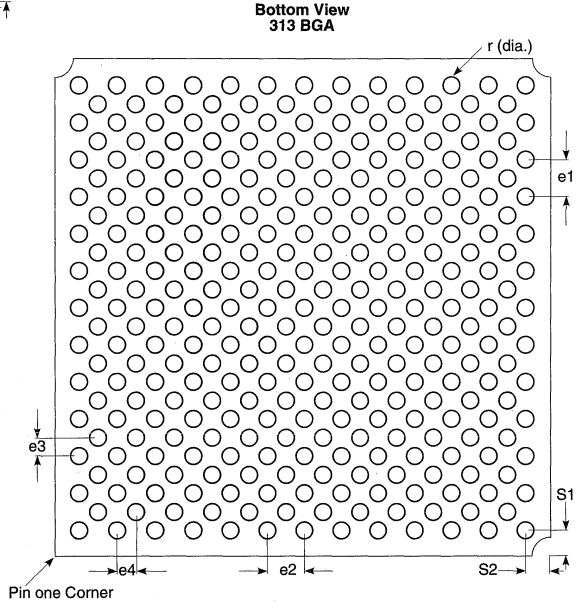
Package Mechanical Drawings

Ball Grid Array









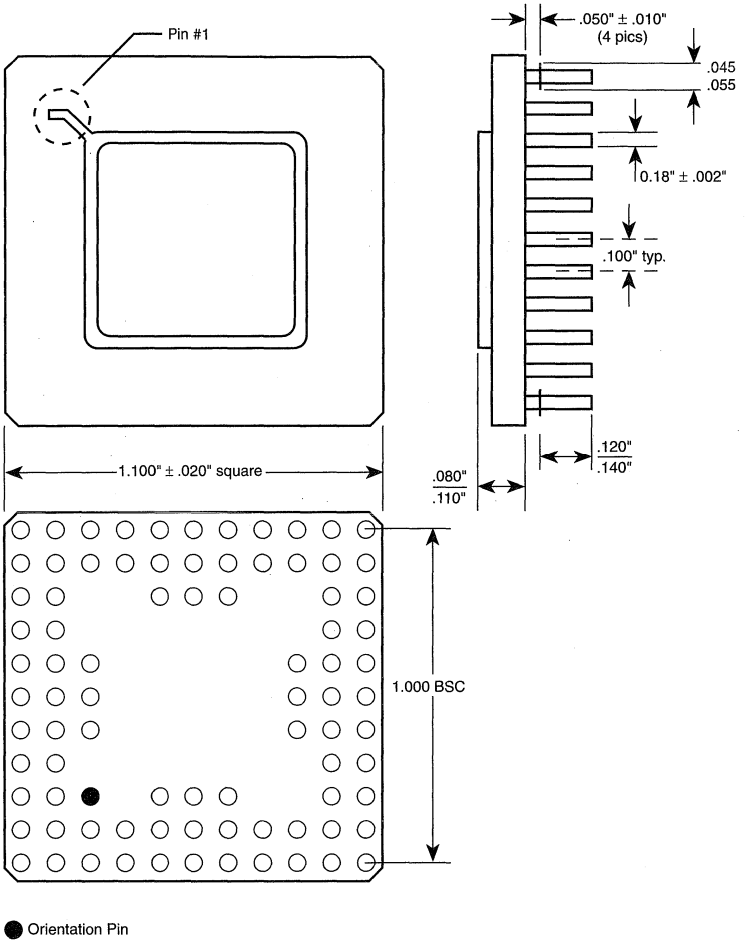
Package Name	BGA 225	BGA 313
Number of Pins	225	313
Dimension (mm)	TYP.	TYP.
A	1.17	1.17
B	0.36	0.56
D/E	27.00	35.00
D1/E1	24.00	30.00
D2/E2	16.10	22.00
L	0.60	0.60
e1 & e2	1.50 BCS	2.54 BCS
e3 & e4	0.75 BCS	1.27 BCS
S1 & S2	3.00	2.25
r (diameter)	0.72 (nom.)	0.72 (nom.)

Note:

1. All dimensions are in millimeters, unless otherwise specified.

Ceramic Pin Grid Array

84-Pin CPGA

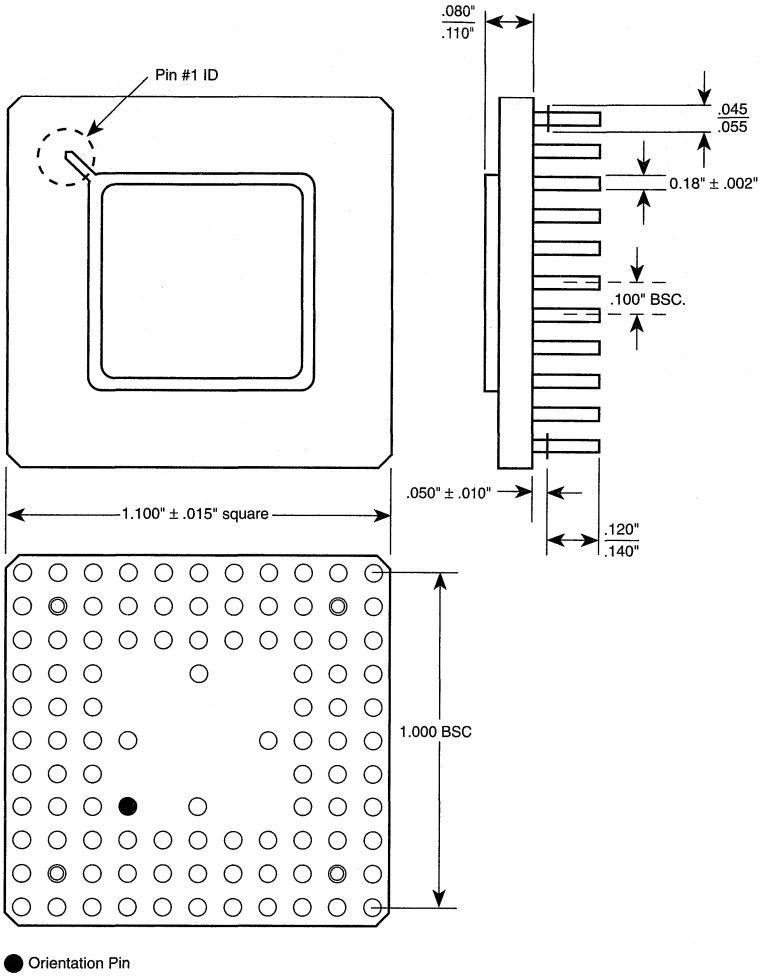


Notes:

1. *All dimensions are in inches unless otherwise stated.*
2. *BSC—Basic Spacing between centers.*

Ceramic Pin Grid Array (continued)

100-Pin CPGA

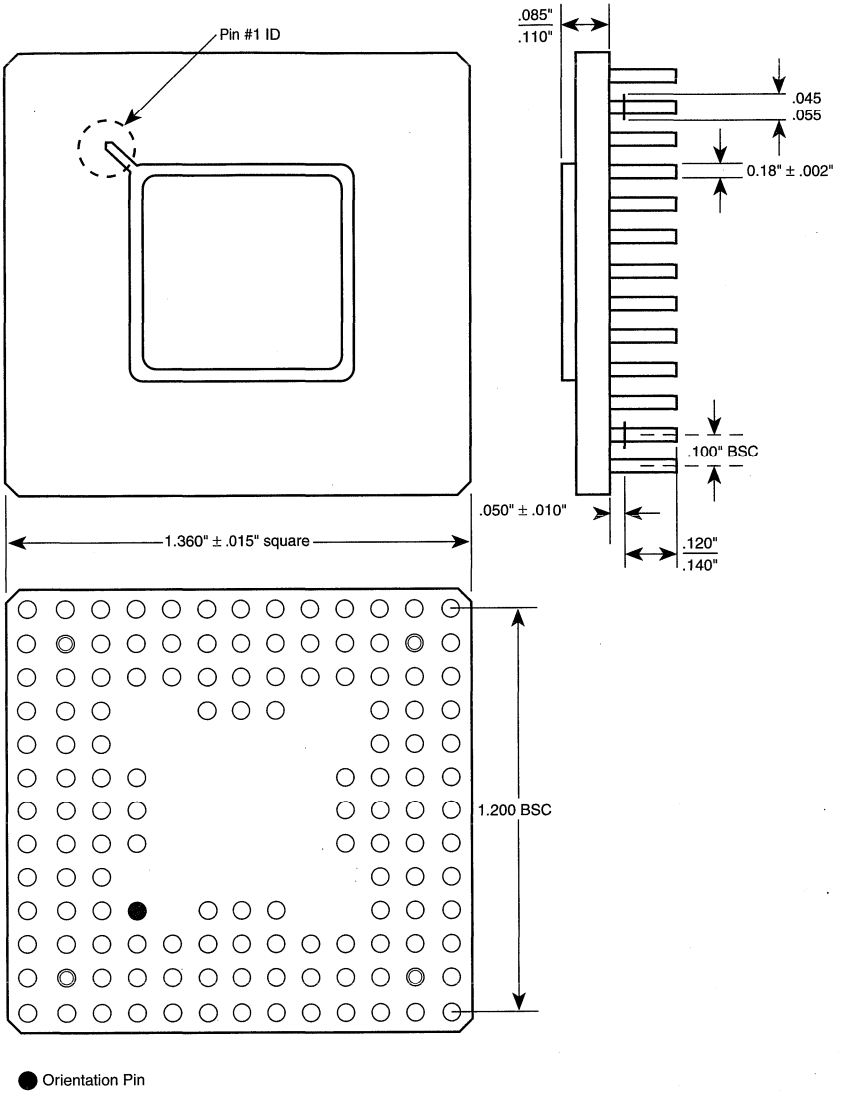


Notes:

1. *All dimensions are in inches unless otherwise stated.*
2. *BSC—Basic Spacing between centers.*

Ceramic Pin Grid Array (continued)

132-Pin CPGA

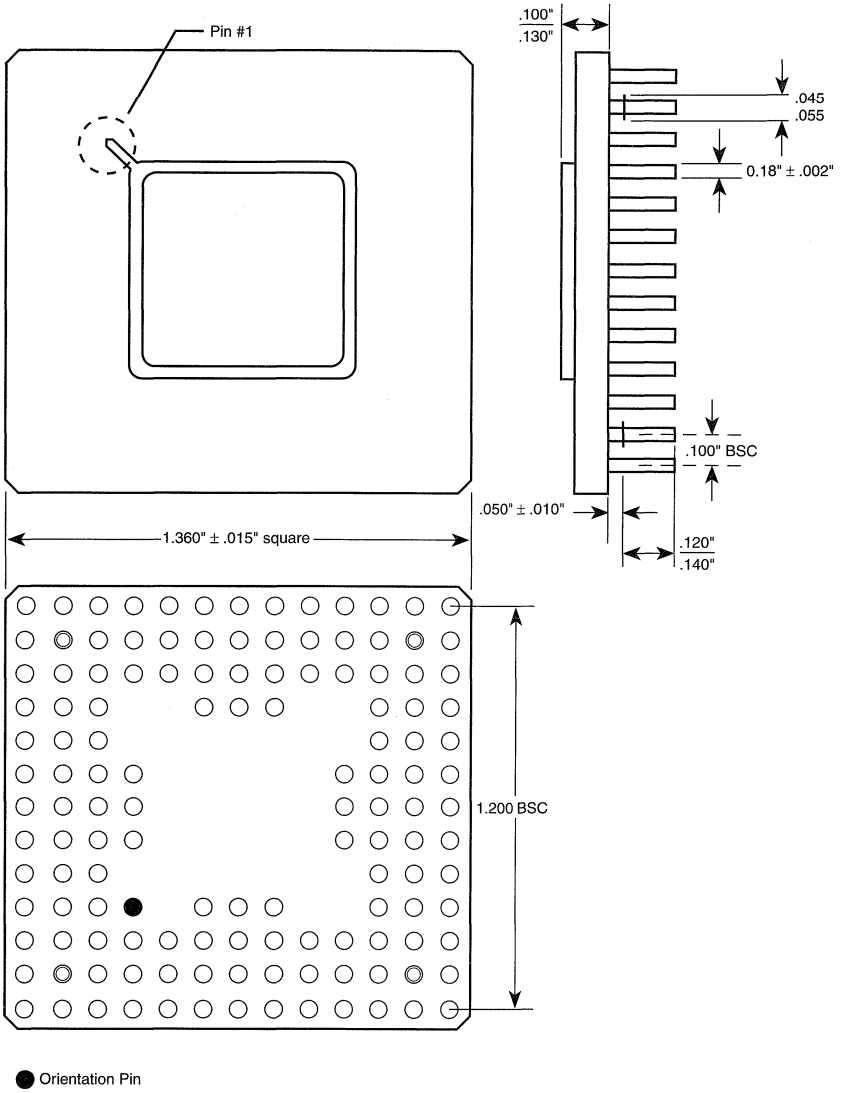


Notes:

1. *All dimensions are in inches unless otherwise stated.*
2. *BSC—Basic Spacing between centers.*

Ceramic Pin Grid Array (continued)

133-Pin CPGA

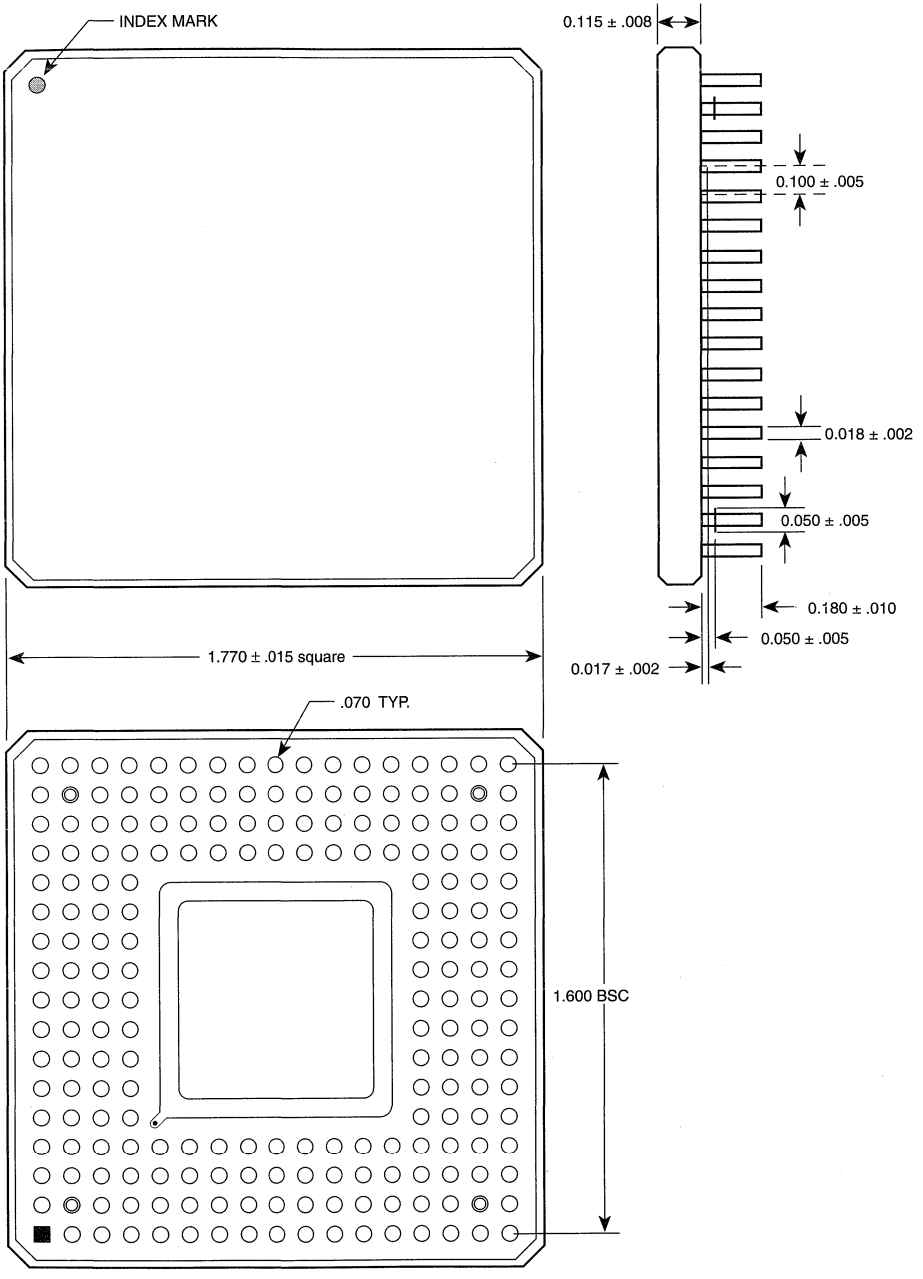


Notes:

1. *All dimensions are in inches unless otherwise stated.*
2. *BSC—Basic Spacing between centers.*

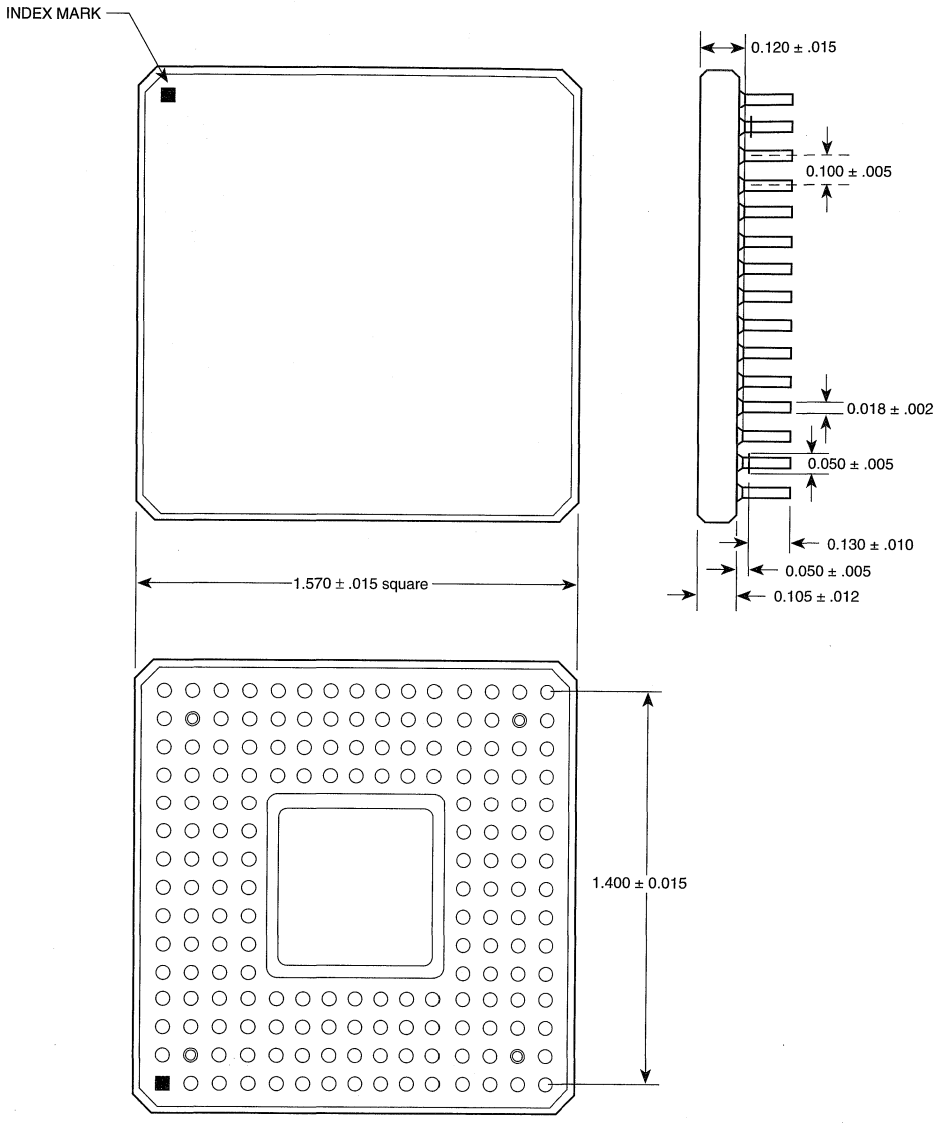
Ceramic Pin Grid Array (continued)

207-Pin CPGA



Ceramic Pin Grid Array (continued)

175-Pin CPGA

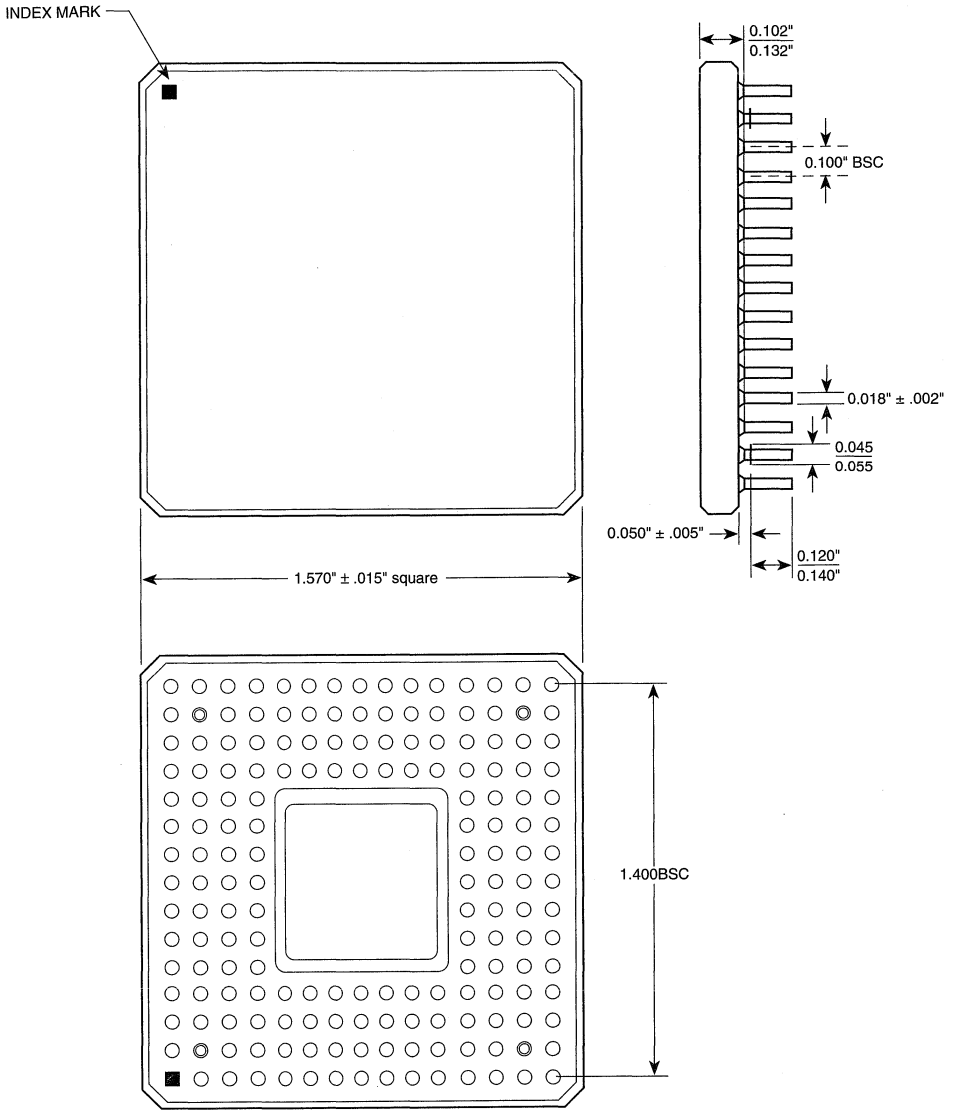


Notes:

1. *All dimensions are in inches unless otherwise stated.*
2. *BSC—Basic Spacing between centers.*

Ceramic Pin Grid Array (continued)

176-Pin CPGA

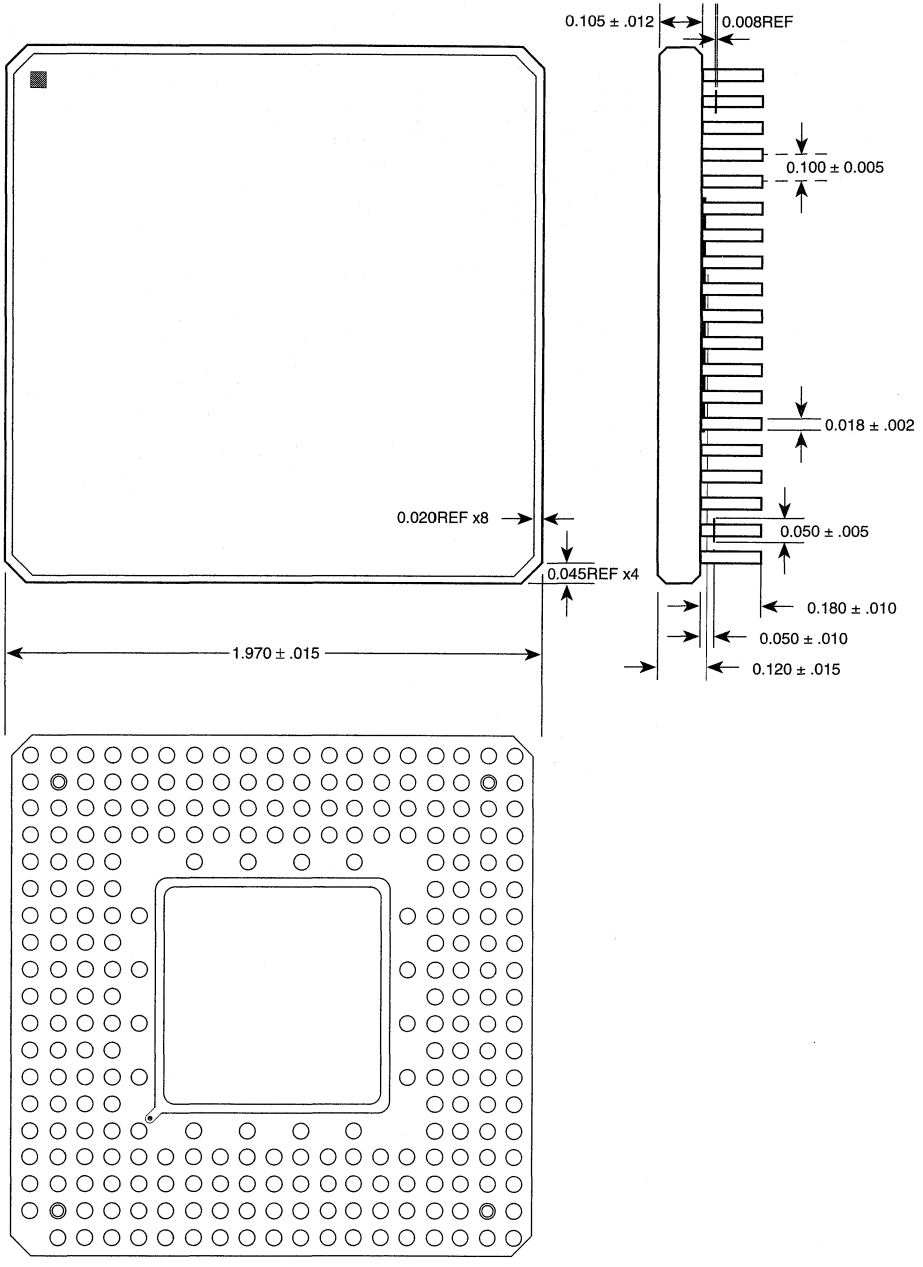


Notes:

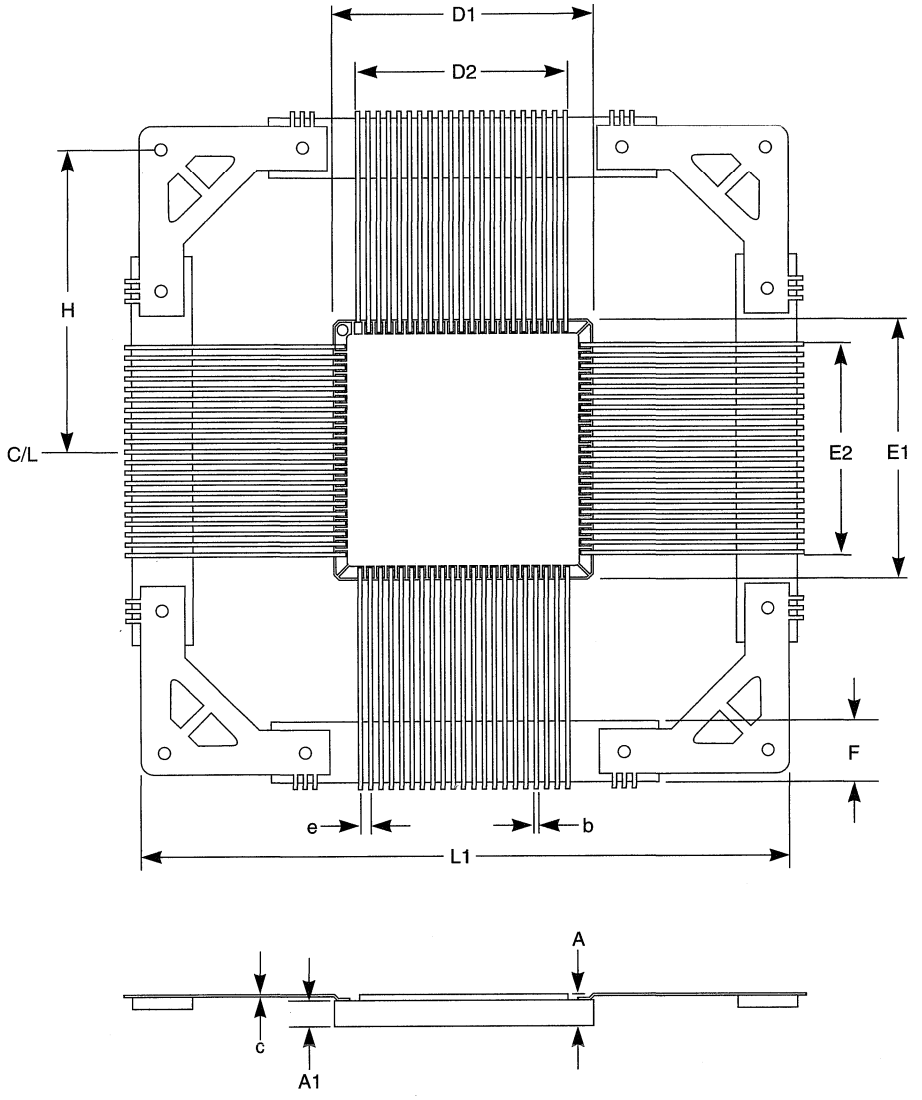
1. *All dimensions are in inches unless otherwise stated.*
2. *BSC—Basic Spacing between centers.*

Ceramic Pin Grid Array (continued)

257-Pin CPGA



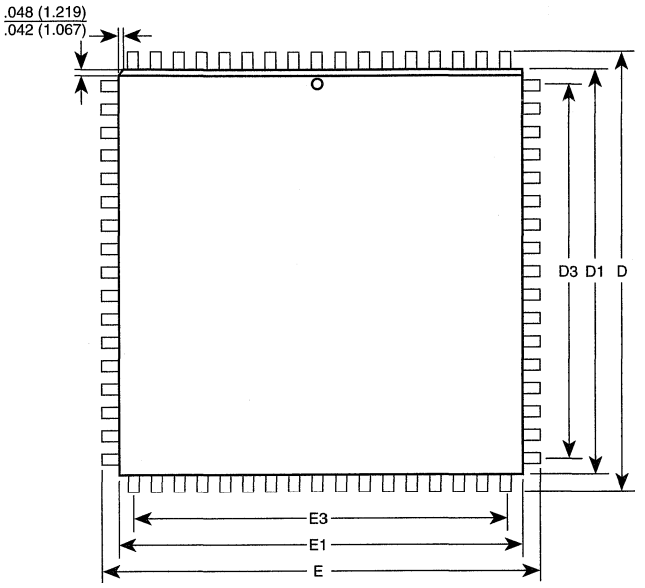
Ceramic Quad Flatpack (CQFP)

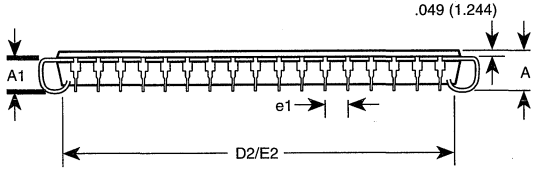


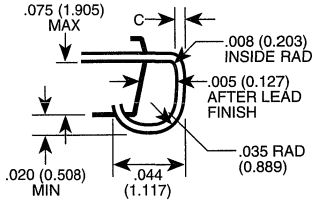
Cermamic Quad Flatpack (CQFP)

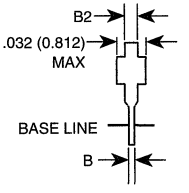
JEDEC	CQ84 MO-90		CQ132 MO-113		CQ172 MO-113		CQ196 MO-113		CQ256 MO-134	
Symbol	**Min**	**Max**	**Min**	**Max**	**Min**	**Max**	**Min**	**Max**	**Min**	**Max**
A	.085	0.130	0.086	0.140	0.086	0.140	0.086	0.140	2.162	3.178
A1	.060	0.105	0.078	0.125	0.078	0.125	0.078	0.125	2.036	2.544
b	0.006	0.012	0.007	0.013	0.007	0.013	0.007	0.013	0.19	0.25
c	0.004	0.008	0.004	0.008	0.004	0.008	0.004	0.008	0.11	0.20
D1/E1	0.635	0.660	0.935	0.965	1.165	1.195	1.325	1.365	35.64	36.36
D2/E2	.500 BSC		.800 BSC		1.050 BSC		1.200 BSC		31.5 BSC	
e	.025 BSC		.025 BSC		.025 BSC		.025 BSC		0.50 BSC	
F	0.130	0.150	0.325	0.375	0.175	0.225	0.175	0.200	7.05	8.45
H	.730 BSC		1.150 BSC		1.150 BSC		1.150 BSC		35.00 BSC	
L1	1.595	1.615	2.485	2.505	2.485	2.505	2.485	2.505	74.60	75.40

Plastic Leaded Chip Carrier (PLCC)





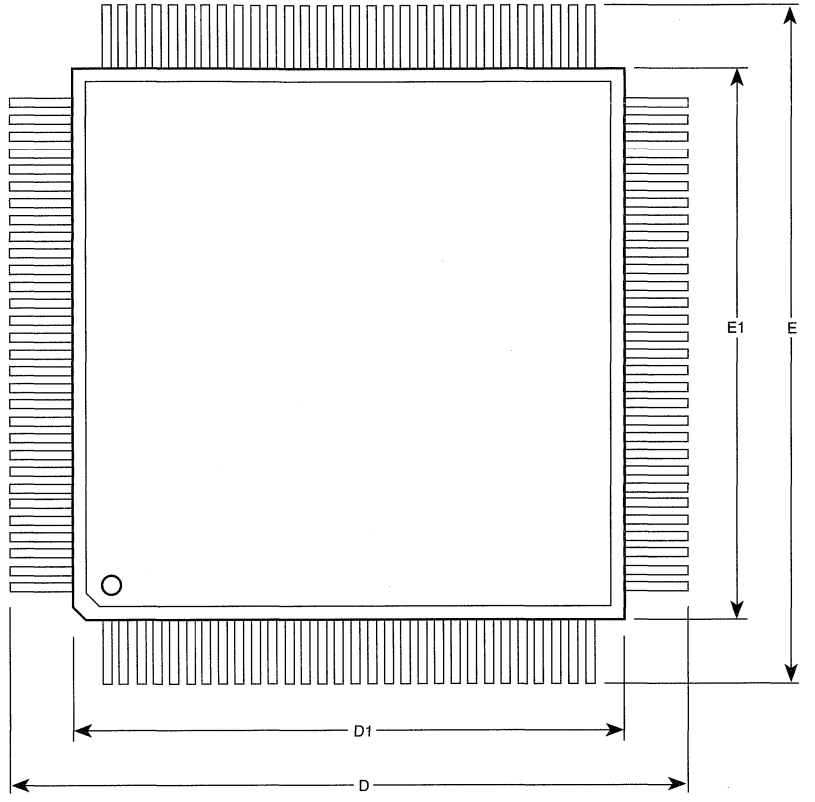


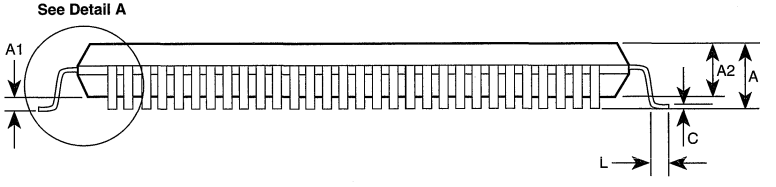


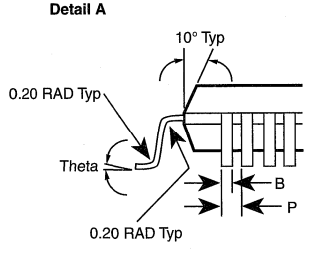
Plastic Leaded Chip Carrier Package (PLCC)

Jedec Equiv	PLCC 44 MS007 AB VAR		PLCC 68 MS007 AD VAR		PLCC 84 MS007 AE VAR	
Dimension	**Min**	**Max**	**Min**	**Max**	**Min**	**Max**
A	0.155	0.175	0.155	0.175	0.155	0.175
A1	0.09	0.13	0.09	0.13	0.09	0.13
B	0.13	0.27	0.13	0.27	0.13	0.27
B2	0.26	0.32	0.26	0.32	0.26	0.32
C	0.007	0.013	0.005	0.011	0.005	0.011
D	0.67	0.71	0.97	1.01	1.17	1.21
D1	0.64	0.66	0.94	0.96	1.14	1.16
D2	0.59	0.63	0.89	0.93	1.09	1.13
D3	0.50 nom		0.80 nom		1.00 nom	
E	0.67	0.71	0.97	1.01	1.17	1.21
E1	0.64	0.66	0.94	0.96	1.14	1.16
e1	0.05 BSC		0.05 BSC		0.05 BSC	
E2	0.59	0.63	0.89	0.93	1.09	1.13
E3	0.50 nom		0.80 nom		1.00 nom	

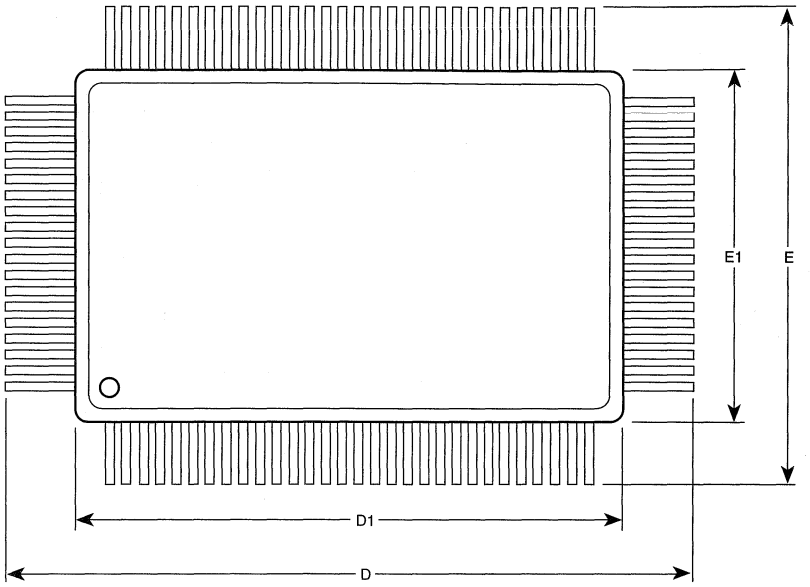
Plastic Quad Flatpack (PQFP and RQFP)

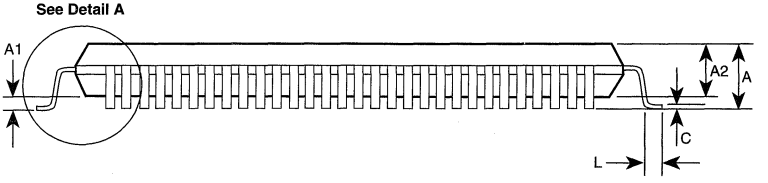




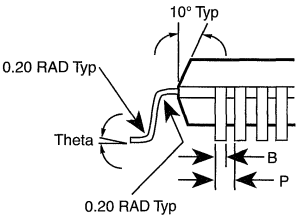


Plastic Quad Flatpack Rectangular Package(PQ100)





Detail A

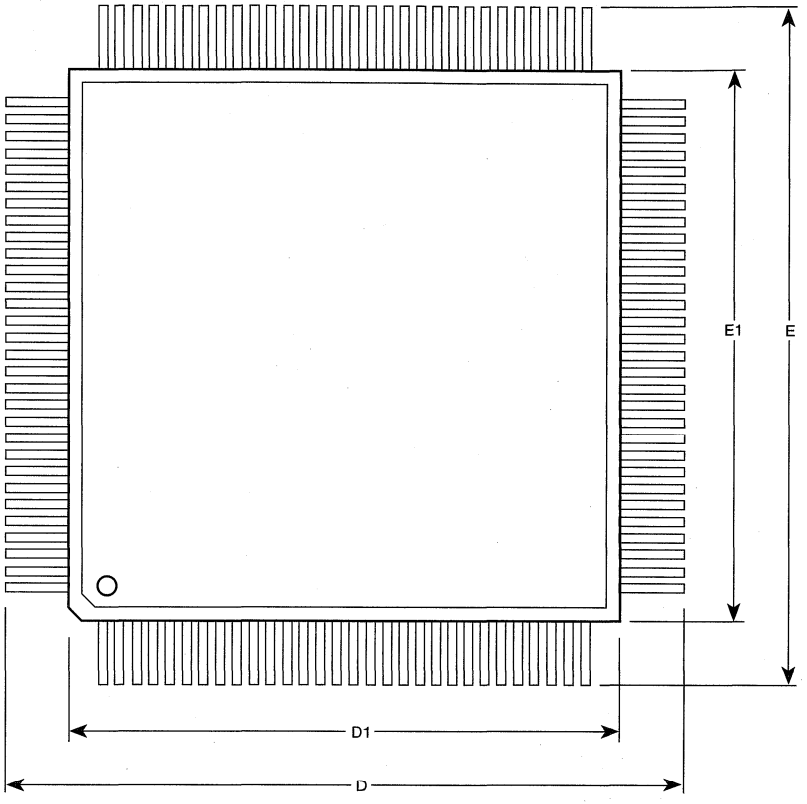


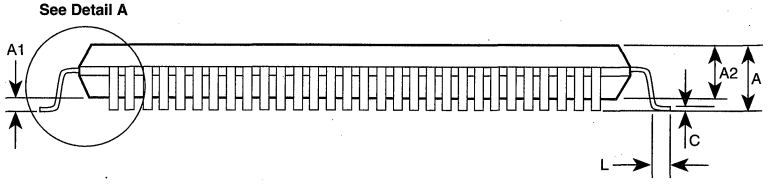
Plastic Quad Flat Packages (PQFP, RQFP)

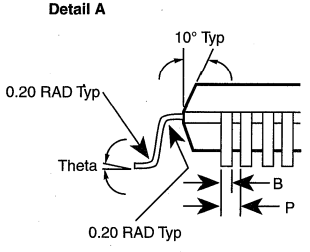
Jedec Equiv	PQFP 100 MO-108		PQFP 144 MO-108		PQFP 160 MO-108/MO-112*		PQFP 208, MO-143		RQ208	
Dimension	**Min**	**Max**	**Min**	**Max**	**Min**	**Max**	**Min**	**Max**	**Min**	**Max**
A	2.8	3.10	3.52	3.84		4.07		4.07		4.07
A1	0.11	0.37	0.25	0.51	0.25	0.51	0.25	0.51	0.25	
A2	2.65	2.77	3.17	3.43	3.17	3.67	3.17	3.67	3.39	3.59
B	0.22	0.38	0.30 TYP		0.30 TYP		0.17	0.27	0.17	0.27
c	0.13	0.23	0.13	0.23	0.13	0.23	0.13	0.23		0.17
D	23.1	23.30	30.35	30.85	30.35	30.85	30.35	30.85	30.35	30.85
D1	19.95	20.05	27.90	28.10	27.90	28.10	27.90	28.10	27.90	28.10
E	17.10	17.30	30.35	30.85	30.35	30.85	30.35	30.85	30.35	30.85
E1	13.95	14.05	27.90	28.10	27.90	28.10	27.90	28.10	27.90	28.10
L	0.65	0.95	0.67	0.93	0.65	0.95	0.50	0.75	0.50	0.75
P	0.65 BSC		0.65 BSC		0.65 BSC		0.50 BSC		0.50 BSC	
Theta	0 deg	7 deg	0 deg	7 deg	0 deg	7 deg	0 deg	7 deg	0 deg	7 deg

*Only applicable to device A1280 f_p = 3.9 mm.

Thin Quad Flatpack (TQFP)
and
Very Thin Quad Flatpack (VQFP)







Thin Quad Flatpacks (TQFP, VQFP)

Jedec Equiv	TQFP 176 MO-136		VQFP 80 MO-136		VQFP 100 MO-136	
Dimension	**Min**	**Max**	**Min**	**Max**	**Min**	**Max**
A		1.60		1.20		1.20
A1	0.05	0.15	0.05	0.15	0.05	0.15
A2	1.39	1.41	0.95	1.05	0.95	1.05
B	0.17	0.27	0.22	0.38	0.22	0.38
c	0.09	0.20	0.09	0.20	0.09	0.20
D	25.75	26.25	15.75	16.25	15.75	16.25
D1	23.95	24.05	13.95	14.05	13.95	14.05
E	25.75	26.25	15.75	16.25	15.75	16.25
E1	23.95	24.05	13.95	14.05	13.95	14.05
L	0.45	0.75	0.45	0.75	0.45	0.75
P	0.50 BSC		0.65 BSC		0.50 BSC	
Theta	0 deg	7 deg	0 deg	7 deg	0 deg	7 deg

Socket Recommendation for Actel FPGA Packages

Prototyping Sockets Resold by Actel

These sockets have been engineered and manufactured by the respective original manufacturer. Actel resells small quantities of these sockets to Actel customers in order to provide a more complete prototyping solution. Actel also offers quick-turn design software and Actionprobe diagnostic tools to speed designs to market.

Package	Lead Count	Actel Part Number	Original Manufacturer	Original Manufacturer Part Number
PQFP	100	SA-PQ100	AMP	P94-1970-016-2
		SY-PQ100	Yamaichi	IC149-100-014-S5
	144	SA-PQ144	AMP	P93-1970-075-1
		SY-PQ144	Yamaichi	IC149-144K11453-0
	160	SA-PQ160	AMP	824176-1
		SY-PQ160	Yamaichi	IC140-160-023-S5
	208	SA-PQ208	AMP	824160-2
		SY-PQ208	Yamaichi	IC198-208-2101
RQFP	208	SA-PQ208	AMP	824160-2
		SY-PQ208	Yamaichi	IC198-208-2101
VQFP	80	SA-VQ80	AMP	824073-1
		SY-VQ80	Yamaichi	IC198-080-2001
	100	SA-VQ100	AMP	P93-1970-030-2
CQFP	132	SY-CQ196	Yamaichi	Custom
	172	SY-CQ196	Yamaichi	Custom
	196	SY-CQ196	Yamaichi	Custom

Socket Sources for Actel FPGA Packages

Actel has compiled this list of know suppliers for the convenience of our customers. This is simply a list of suppliers that we are aware of rather than a list of recommended sockets, as we have not tested them for reliability. For information on these sockets, contact the manufacturers directly.

CONTACTS:

AMP	(408) 725-4975
Enplas	(415) 572-1683
Methode	(408) 262-3812
Mil-Max	(516) 922-6000
McKenze	(510) 651-2700
Nepenthe	(415) 496-6666
Wells	(408) 559-8118
Yamaichi	(408) 456-0797

Surface Mount Sockets

		Prototype/Production	
Package	**Lead Count**	**Source**	**Part Number**
PLCC	44	Amp	821979-3 or 822035-3
		Methode	213-044-602
	68	Amp	822029-3 or 822073-3
		Methode	213-068-602
	84	Amp	821151 or 822180-1
		Methode	213-084-602
PQFP	100	Amp	P94-1970-016-2
		Yamaichi	IC149-100-014-S5
	144	Amp	P93-1970-075-1
		Yamaichi	IC149-144K11453-0
	160	Amp	824176-1
		Yamaichi	IC149-160-023-S5
	208	Amp	824160-2
		Yamaichi	IC198-208-2101
RQFP	208	Amp	824160-2
		Yamaichi	IC198-208-2101
VQFP	80	Amp	824073-1
		Yamaichi	IC198-080-2001
	100	Amp	P94-1970-030-2

Through Hole Sockets

		Prototype/Production		Zero Insertion	
Package	**Lead Count**	**Source**	**Part Number**	**Source**	**Part Number**
PLCC	44	Amp	821575-3	Nepenthe	PC1-044050-016
		Methode	213-044-401	Yamaichi	IC51-0444-400
	68	Amp	821574-3	Nepenthe	PC1-068050-001
		Methode	213-068-401	Yamaichi	IC51-0684-390-1
	84	Amp	821573-3	Nepenthe	PC1-084050-003
		Methode	213-084-401	Yamaichi	IC51-0844-401-1
PQFP	100			Yamaichi	IC51-1004-814-2
	144	Amp	822114-3 and 822115-3	Yamaichi	IC51-1014-KS10418
	160	Amp	822114-4 and 822115-4	Yamaichi	IC51-1604-845-1
	208			Yamaichi	IC51-1052-KS12897
RQFP	208			Yamaichi	IC51-1052-KS12897
TQFP	176			Yamaichi	IC51-1764-1505
VQFP	80			Yamaichi	IC51-0804-795
BGA	225			Enplas	BGA-225-1.5-01
	313			Enplas	BGA-313-841-1.27-01
CQFP	84 w/o tie bar			Wells	619-1000311-001
	84 w/ tie bar			Wells	619-1001611-001
	132			Actel	Custom
	172			Actel	Custom
	196			Actel	Custom
	256			TBD	TBD
PGA	85	Mil-Max	510-91-085-11-041	Yamaichi	NP35-11207-KS8108
		McKenze	PGA-85H-012B-1-1107		
	100/101	McKenze	PGA-101M-012B-1-11B5	Yamaichi	NP89-12110-KS11922
	132/133	McKenze	PGA-133H-003B-1-13GOR		
	175/176	Mil-Max	510-91-176-15-061	Yamaichi	NP89-22508-KS11957
		McKenze	PGA-177M-003B-1-1552		
	207			Amp	916227-6
	257			Amp	916229-6

PQFP Handling Instructions

Humidity Control

Follow the instructions on the packaging bag to keep devices dry prior to board assembly.

Handling Trays

The tray stack should be kept fastened with straps during transportation to prevent devices from coming out of their sockets and bending their leads. If a single loaded tray is to be moved, an empty tray should be placed over it as a cover and both trays should be taped to secure the devices in position (see Figure 1).

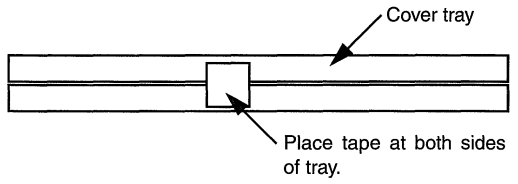


Figure 1 • Temporary Tray Holding

Device Programming

Always use the vacuum wand provided to transfer devices.

Removing Devices from Tray

1. Position the vacuum needle at the proper angle so that your index finger can rest on the vacuum switch (see Figure 2).

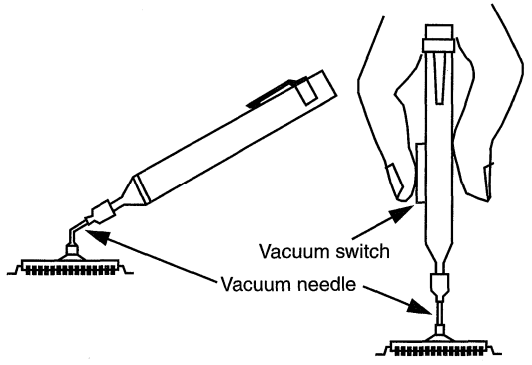


Figure 2 • Vacuum Wand

2. Slightly depress the vacuum switch and then place the vacuum cup on the surface of the device.
3. Release the vacuum switch while holding the cup down to generate the vacuum that causes the cup to adhere to the device body.
4. Make certain to transfer a device within 5 to 8 seconds.

Loading the Device into the Programming Socket

1. Carefully align the device with the socket and gently lower the device into the socket (see Figure 3).

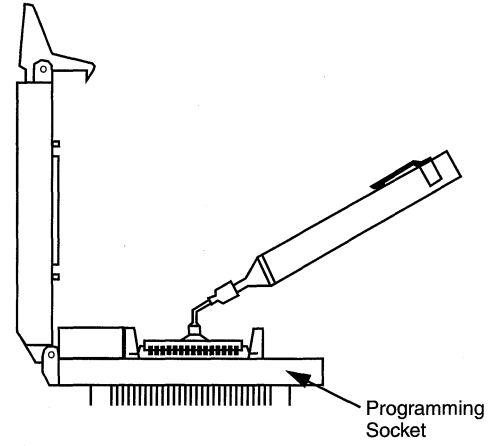


Figure 3 • Inserting the Device in the Socket

2. Do not force the device into the socket; otherwise, the corner leads may be bent if the alignment is out.
3. Release the vacuum by depressing the vacuum switch. The device will be self-aligned to the socket.

Note: To completely release the vacuum, the switch should be pressed down further than when pressed to generate the vacuum. Otherwise, a partial vacuum will pull the device back and may cause bent leads.

4. Close the lid. The device is ready for programming.

Removing the Device from the Programming Socket

When the programming of the device is complete, use the vacuum wand to remove the device from the socket and to place it in a tray labeled "programmed devices."

Cleaning the Vacuum Wand

The suction cup of the vacuum wand should be cleaned periodically using IPA to minimize possible leakage in vacuum.

Application Notes—Design with Actel Devices

Section 3: Application Notes—Design with Actel Devices

This section discusses Actel device specific design issues. The information presented here provides more insight into the architecture of the Actel Device families as well as further elaboration on using specific structures such as clock networks not covered in sufficient detail in Section 1. All the documents covered in this section are listed below and cross-referenced by family.

Switching Elements, The Key to FGPA Architecture

Switching Elements, the Key to FPGA Architecture

Yousef Khalilollahi
Actel Corporation
955 E. Arques Ave.
Sunnyvale, CA 95136

Introduction

In the competitive electronics industry, it is vital that new products reach the market as soon as possible. Furthermore, the financial risks incurred in developing new products must be minimized so that new ideas can be easily implemented. FPGA companies provide their users with means of implementing their logic designs on devices that are programmable by the circuit designers. An FPGA can be programmed in only minutes.

Designers describe their circuits through a schematic capture tool or a high level design language. The circuit is then mapped into the FPGA. The mapping process is dependent on the FPGA architecture and directly influences the results and the performance obtained from the device. This makes FPGA architecture, the most important factor in determining the end results of its implementation. Different elements characterize FPGA architecture. However, FPGA switching elements construct the fundamental characteristics of an FPGA and are the key to its architecture.

What is an FPGA?

An FPGA consists of an array of logically uncommitted elements that can be interconnected to perform a specific application.

There are three important elements that characterize any FPGA architecture:

- Internal building blocks (logic modules)
- Interconnect between the logic modules
- Switching elements

Logic Modules

The internal building blocks are the basic elements of an FPGA. Each building block is configurable to perform a logic function. The complexity of the logic function that can be implemented depends on the capacity of these building blocks. When necessary, two or more of these logic blocks may be connected to perform the desired logic function. In the Actel architecture, these blocks are called Logic Modules. There are two kinds of Logic Modules in the ACT 3 architecture: combinatorial and sequential. Figure 1a and 1b show an ACT 3 combinatorial and sequential Logic Module.
Using these logic modules as building blocks,

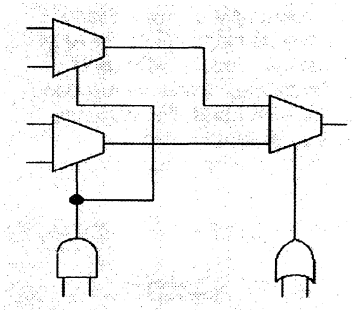


Figure 1a) ACT 3 Combinatorial Module

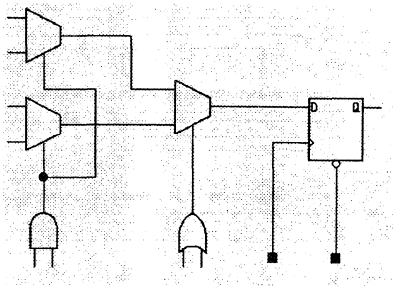


Figure 1b) ACT 3 Sequential Module

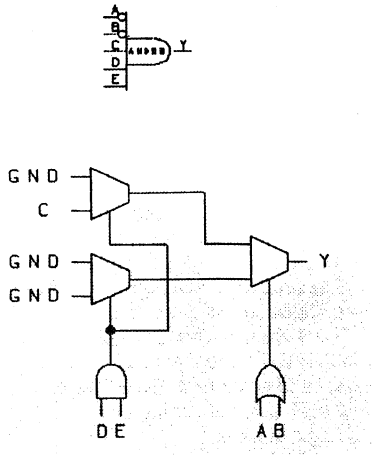


Figure 2) 5-input AND function implementation

Actel software maps logic functions into the Actel device. An example of how a 5 input AND function may be mapped into the Actel ACT 3 combinatorial module is shown in figure 2. Flip Flops and latches are mapped into the sequential modules. For efficiency, in many cases a combinatorial and a sequential function may be mapped into one sequential module.

Interconnects

The interconnection between the Logic Modules is provided by channels of wiring segments between the rows and columns of the Logic Modules (figure 3). These wiring segments are connectable at different points through switching elements (discussed in the next section). While sufficient wiring segments for good routability must be provided, excess wiring segments waste chip area. One of the important architectural considerations is the ability to route an application automatically with no manual intervention.

Switching Elements

The average designer might be familiar with the functionality of the logic modules and the interconnects between them. But what often gets ignored in some design environments is the choice of the switching elements in their FPGAs. Switching elements are actually the driving force in determining the choice of logic modules and their interconnect for FPGA companies.

This paper will examine the switching elements in the Actel and Xilinx FPGAs. Xilinx devices use an SRAM technology to implement the switching elements in the FPGA. An SRAM-programmable FPGA is programmed by loading configuration

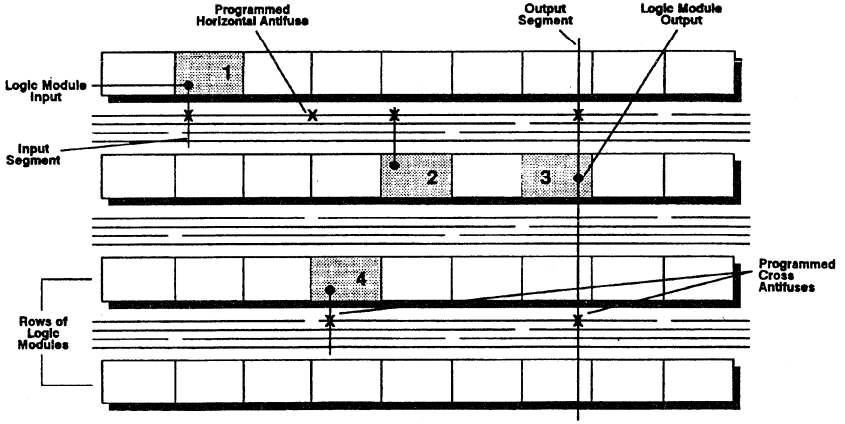


Figure 3) Logic Modules and their interconnects

memory cells from an external source. These RAM cells, control the logic blocks and their interconnects that perform the application function of the FPGA.

Each SRAM switching element consists of a 5 transistor RAM cell and a pass transistor called *pip (programmable interconnect point).* Pips control the connection of wiring segments in the programmable interconnect. Pips are controlled by the configuration RAM cell. Wire segments on each side of the Pip transistors are connected or not depending on the value in the RAM cell. This SRAM switching element occupies an area in the order of 50-225 square micron depending on the technology. SRAM switching elements resistance ranges between 800 and 1900 Ohms and its capacitive load falls between 8 to 17 FF (see figures 4).

Actel uses antifuses as switching elements in its FPGAs. An antifuse is a device that irreversibly changes from a high to low resistance state when a programming voltage is applied across its terminals. Antifuses fall into two categories: amorphous silicon and dielectric. The dielectric antifuse, used by Actel, consists of a multi layer of dielectric material placed between N+ diffusion and polysilicon. Upon application of sufficient voltage, the programming pulse melts the dielectric, creating a conductive link between the electrodes. This multilayer dielectric antifuse is called PLICE (Programmable Low Impedance Circuit Element).

The PLICE antifuse has a resistance over 100 giga-ohms in its unprogrammed state. Once programmed, its resistance falls between 100 to 600 ohms depending on the programming current. In addition, the size of the PLICE antifuse is so small (~ 9 sq. micron) that the area of a switch array is limited by the pitch of the metal wires rather than the size of the antifuse itself. In addition, the capacitive load presented by the PLICE antifuse is between 5 to 13 FF depending on the technology. Figure 4 shows the three important characteristics (area, resistance and capacitance) of different switching elements vs. the process technology used in the FPGA.

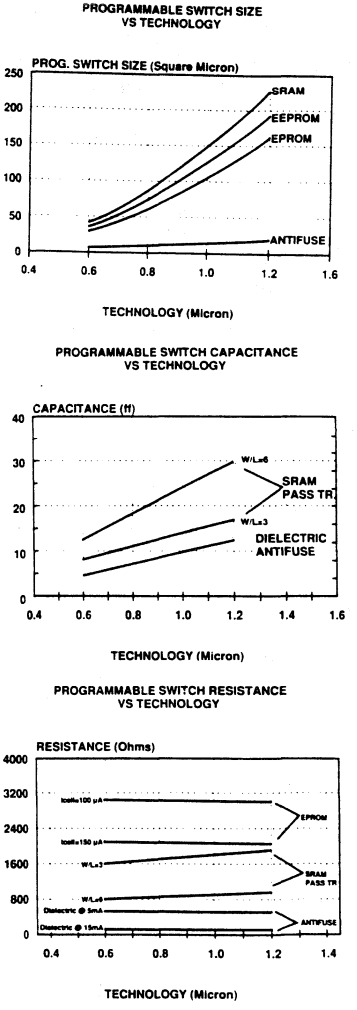


Figure 4) Switching elements characteristics

As the figure above shows the antifuse programming element offers the smallest cell size, lowest resistance and lowest capacitance. But why are these three factors important? Why should a designer be concerned about the details of the switching

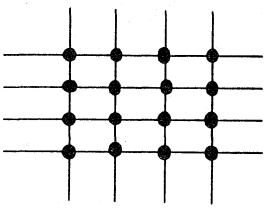
element in the FPGA? The answer is very simple. What determines the speed, performance and cost of an FPGA is actually buried in the characteristics of its switching element. The lower the resistance and capacitance of the switching element the faster the interconnect speed. Furthermore it is obvious that the smaller the switching element, the larger the total die area left for logic capacity. The area dedicated to switching elements does not contribute to logic capacity of the device and hence it wastes the die area. The effects of the switching elements characteristics are even more extensive. One of the main impacts of the size, resistance and capacitance of the switching elements is their effect on granularity of the logic blocks in the FPGA. If the switching element in an FPGA is large and slow, it effects the strategy in choosing the ideal Logic Block in the following way:
Because of the large size and the delay introduced by the switching element, it is best to avoid using too many of them in the chip. This dictates the choice of the Logic Block directly. The Logic Block used in that FPGA must be large enough to be able to pack in as much logic as possible and consequently avoid needing interconnects to another logic block. If it happens that the application's logic cannot be packed into one Logic Block, it must use the interconnect resources and switching elements and hence pay a penalty in performance.

The above scenario is the case in Xilinx's family of FPGAs. The SRAM switching elements' large area is too costly to allow its free usage in Xilinx devices. An architecture with more switching elements provides more options for routing, simplifying routing and possibly enhancing performance. However, in SRAM FPGAs, more switching elements means considerable more die area and it is simply not possible.
Furthermore, the SRAM switching elements are slow enough to warrant a large Logic Block. The XC4000 CLBs (Configurable Logic Block) consist of two 4-input Lookup Tables (LUT4) driving a 3-input Lookup Table multiplexed with a Flip Flop element. This large CLB can pack in large chunks of logic. In fact, the CLB used in XC4000 seems to be the right choice for the large and slow SRAM switches. However, using the large CLBs, the device is faced with possible inefficiencies. This can be illustrated with a consideration of the CLB and its lookup table width. The four-input lookup table has sixteen memory cells. If the logic being implemented in 4-input lookup table breaks naturally into two or three input gates, half or more of every 4-input lookup table area will be wasted, leading to an inefficient implementation on the FPGA. Likewise, if the logic being implemented naturally breaks into 6-input gates, again more than half of the second lookup table area will be wasted.

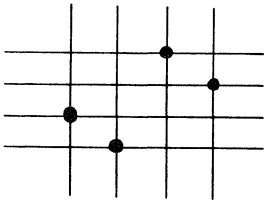
In Actel's FPGAs, since antifuse switches occupy negligible die area, a much smaller Logic Module is used. Less logic may be packed in these modules than the SRAM based CLBs. However, when needed, two or more of them may be connected to each other at considerable lower cost in terms of die area and speed degradation. Since antifuse switches will provide fast and low capacitance connections between wire segments, there is no need to choose a large Logic Module. The smaller Logic Module, providing a higher level of granularity, enables Actel's software to pack in logic with minimal wasted die area.

Another significant effect of the size of switching elements is the limitations that it introduces in connecting wire segments. As mentioned earlier, the routing architecture of an FPGA includes channels of wiring segments. Where the wiring channels intersect, their connections are made by the switching elements. Maximum routability is achieved if every horizontal wiring segment is connectable to every vertical segment (figure 5a). A full crossbar switch connecting n horizontal wires to m vertical wires requires nxm switching elements. In practice, if these switching elements are large, only a few of the possible connections between segments are allowed. The routing architecture of an SRAM FPGA falls victim to this limitation. nxm pips occupy a prohibitively large area. A much

smaller subset of all possible connections is provided using the SRAM switches (figure 5b).



5a) Fully connected crossbar switch



5b) partially connected segments

This routing limitation can have serious side effects. One of the more aggravating consequences of this shortage of switching elements is the problems experienced during design iterations. Consider the following likely scenario:
In many circuit design situations, the board is configured before the FPGA design is complete. Consequently, the I/O assignment of the FPGA is determined before the design has been implemented in the FPGA. In many cases, an existing FPGA design needs to be modified slightly, and re-programmed into the FPGA. In all of these situations, the I/O assignment of the FPGA is fixed before the FPGA has been placed and routed. The flexibility provided by an abundant of switching elements is essential for the routing algorithms' success. If the there are limited routing choices due to lack of switching elements (not lack of routing segments), the routing algorithm will fail to route the design. This will create a helpless situation for designers.

In an antifuse FPGA, the small size of the antifuse allows its free usage wherever routing segments cross each other. This provides the flexibility necessary for the routing algorithms to provide a solution regardless of number of design iterations and modifications. An antifuse FPGA is simply more routable than an SRAM FPGA.

An important effect of routability is its influence on the segmentation of the routing tracks and consequently the performance of the FPGA. Segmentation is defined as the way the routing tracks are configured in an FPGA. As mentioned earlier, the wire segments span along the FPGA in routing channels between the Logic modules. However, the length of their span can be varied. A wire segment may span between only two adjacent Logic Modules while others may span between three or more Logic Modules. In the segmentation process, there is a distinct trade-off between routability and performance (delay performance of the FPGA). However, the scale always tips toward routability first and performance second. Once the FPGA companies are confident that their FPGAs are easily routable under any circumstances, then they can think about improving the performance achieved by their routing algorithms. If a design can not be routed inside an FPGA, its performance is irrelevant.

Figures 6 and 7 show two different approaches in FPGA segmentation. The Unit length segmentation used in Xilinx's SRAM FPGAs simply consists of wire segments of equal size spanning between two adjacent Logic Blocks. This provides the much needed ensurance that routability is possible in most cases. However, the premium is paid by reduced speed performance. Since every wire segment spans only between two adjacent blocks, if a block needs to be wired to another block N columns away, N SRAM switches must be used for this connection (unless the full length segments are available). This reduces the speed performance of the FPGA considerably.

In the varied length segmentation approach, used in Actel's antifuse FPGAs, the wire segments span in different lengths. The length

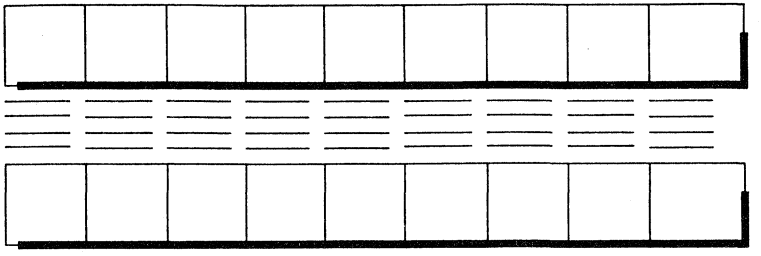


Figure 6) Unit length segmentation

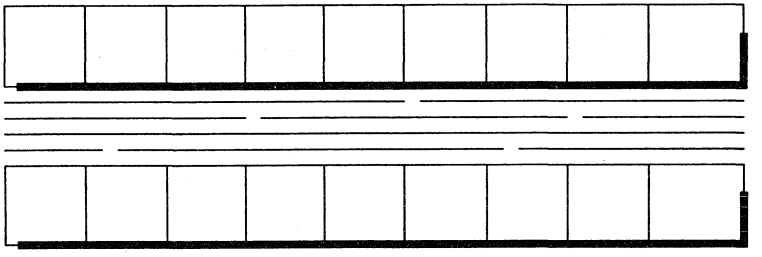


Figure 7) Varied length segmentation

of the wire segments varies from short (two adjacent Logic Modules), to long, which spans up to the whole die. The lengths of these segments is chosen statistically during the development period. The connection between two far away blocks no longer needs many switching elements. In fact the maximum number of antifuses needed to connect any block to any other block is four. The average number is much lower at 2.07 antifuses for each connection. The varied length segmentation approach provides less routing solutions than if the wire segments only spanned between two Logic Modules. However, the speed performance is improved considerably. Since its routability confidence is high enough, the antifuse based FPGA can afford to trade off routability for performance. Consequently, the varied length segmentation scheme is used in Actel's FPGAs to achieve better performance. This is not the case in the SRAM FPGAs, since routability is so marginal and precious, it is not traded for performance gain.

Conclusion

The architectural characteristics of FPGAs lie deep in their switching elements. Everything from the choice of logic blocks to the routing segmentation is influenced by these switches. SRAM switches provide the possibility of reprogramming the device. However, this feature comes at a prohibitive price: large area and reduced performance. Unless the reprogrammabilty of an FPGA is an absolute requirement, the antifuse based FPGAs provide a better solution to the switching requirements inside an FPGA. In most design environments, the cost of implementing a design on an FPGA is not dominated by the unit cost of the FPGAs programmed in the development phase. The dominant cost is the engineering time spent on achieving performance goals and the risks involved in not reaching the market on time.

References

S.M. Trimberger et. al., Field-Programmable Gate Array Technology.

Dave Lee et. al., "A high performance I/O intensive synthesis-friendly sub-micron field programmable gate array family," COMPCON Spring 1993.

Actel Logic Modules

Introduction

The ACT™ 1, ACT 2, 1200XL, and ACT 3 families of Actel FPGAs are based on channeled array architecture consisting of rows of modules interspersed with routing channels. There are two types of modules: logic modules and I/O modules. The logic modules are blocks that implement logic functions. What follows describes in detail the structure of logic modules for all the Actel families.

Discussion

ACT 1 Logic Modules

A logic module for ACT 1 devices is an eight-input, one-output logic circuit that can implement logic functions (NAND, AND, OR, NOR, and so on) in gates of two, three, or four inputs. One logic module has three 2-to-1 multiplexers with different select lines (Figure 1). The select lines of the two multiplexers (M1 and M2 in Figure 1) are individual inputs. The select line of the third multiplexer (M3 in Figure 1) is an OR function of the other two inputs. One logic module can implement all functions of two-input variables, most functions of three-input variables, and some functions of four-input variables. Figure 2 shows one of several ways to implement a three-input AND gate using one ACT 1 logic module. Actel's layout software automatically selects the best implementation for each instance of a logical function. All latches in the ACT 1 architecture are implemented with one logic module, while all flip-flops are implemented with two adjacent logic modules in a master/slave configuration. The full ACT 1 logic module is available for use as the CM8A hard macro.

ACT 2 and 1200XL Logic Modules

Logic modules for ACT 2 and 1200XL devices are classified into two types: combinatorial modules (C-modules) and sequential modules (S-modules) (Figure 3 and Figure 4). The C-module is an enhanced version of the ACT 1 family logic module optimized to implement high fan-in combinatorial logic functions. One C-module consists of a 4-to-1 multiplexer with one select line (S1 in Figure 3) acting as an OR function of the two inputs, A1 and B1, and the second select line (S0 in Figure 3) acting as an AND function of two other inputs, A0 and B0. All three-input gates, most four-input gates, and some five-input gates can be generated with one C-module. The S-module is designed to implement a high-speed flip-flop (or latch) with additional combinatorial logic in a single module

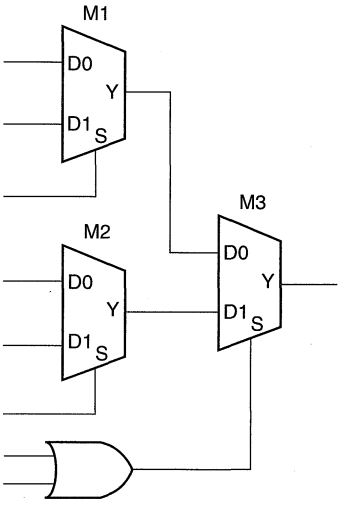


Figure 1 • Logic Module of ACT 1 Devices

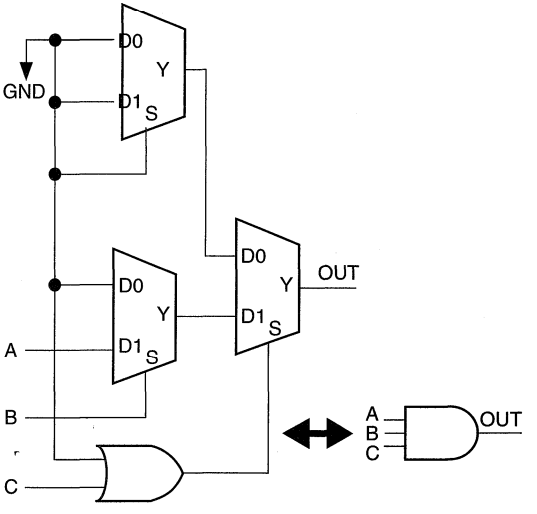


Figure 2 • An Example of Implementing a Three-input AND Gate Using One ACT 1 Logic Module

without additional delay. For example, hard macro DFM6A fits into one S-module, but it does not fit into one ACT 1 logic module. The S-module includes a C-module and a sequential element. An S-module can be configured as a seven-input combinatorial function plus a D-type flip-flop with clear (Figure 4a) or as a seven-input combinatorial function plus a latch without clear (Figure 4b). However, one S-module implements only a four-input logic function plus a latch with clear (Figure 4c). Thus, the combinatorial function used in the S-module is not a full implementation of the C-module. The S-module can also be configured strictly as a C-module for maximum combinatorial logic utilization (Figure 4d).

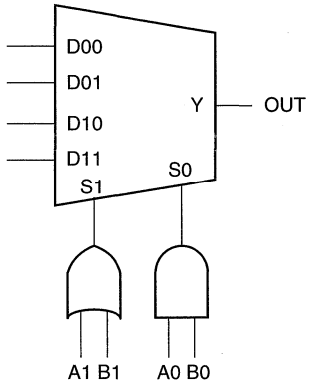
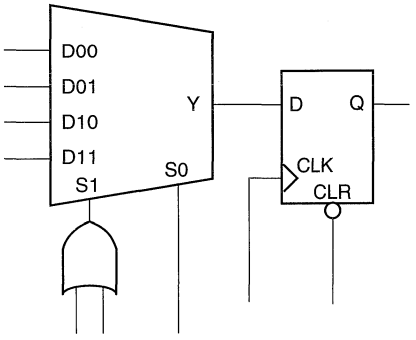
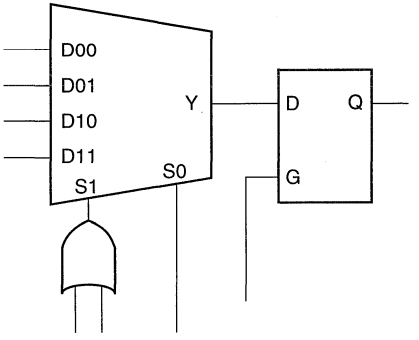


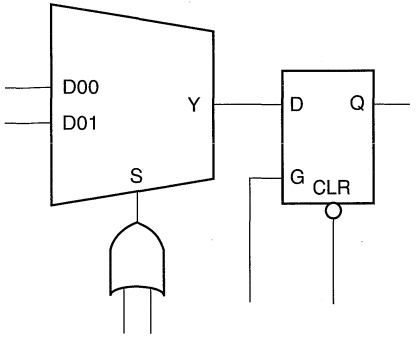
Figure 3 • ACT 2 C-module Implementation



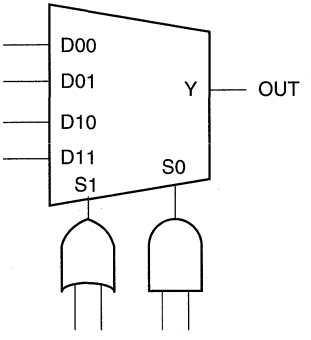
a. Up to Seven-input Function Plus D-type Flip-Flop with Clear (ACT 2, 1200XL)



b. Up to Seven-input Function Plus Latch (ACT 2, 1200XL)



c. Up to Four-input Function Plus Latch with Clear (ACT 2, 1200XL)



d. Up to Eight-input Function— Same as C-module (ACT 2, 1200XL)

Figure 4 • ACT 2 and 1200XL S-Module Implementation

ACT 3 Logic Modules

Logic modules of ACT 3 devices are enhanced versions of the ACT 2 and 1200XL family logic modules. The C-module is equivalent to the previous one discussed (Figure 5). The S-module contains a full implementation of the C-module with eight inputs plus a clearable element that can be configured as a latch or a D-type flip-flop (Figure 6). A single S-module can implement any function implemented by a C-module plus a flip-flop or latch with clear. The clock input CLOCK may be connected to one of three clock networks, CLKA, CLKB, HCLK or any other internal macro. As in the ACT 2 and 1200XL devices, ALS can configure an S-module as a C-module. The ACT 3 logic module is available for use as the CM8 and DFM8A/B hard macros.

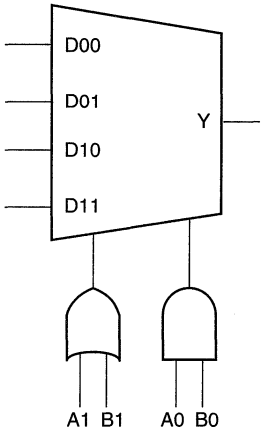


Figure 5 • ACT 3 C-module Diagram

Table 1 lists the number of logic modules in all Actel devices.

Designing for Combinability

Combinable flip-flops and latches are so called because the combinatorial resource is free to be used to implement the combinatorial functions in the library. The resource is not used by the flip-flop itself, leaving it available for combinatorial functions. If a combinatorial function in the schematic is driving a single data input to a combinable flip-flop or latch, ALS will put the combinatorial function and

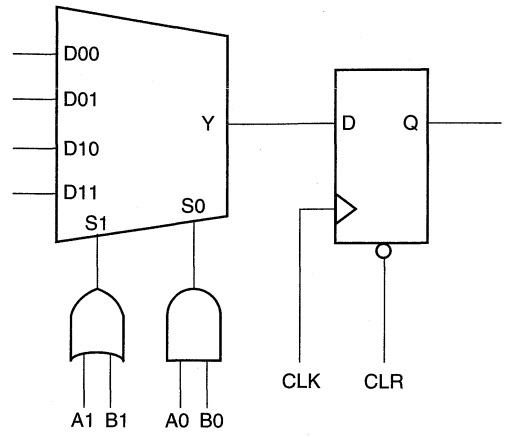


Figure 6 • ACT 3 S-module Diagram

the flip-flop into the same S-module. When the two functions are combined into one S-module, there is only one module delay for both macros. Use this technique of connecting combinable combinatorial macros to combinable sequential elements for the fastest, most compact designs.

The ACT Family Macro Library Guide provides information about the combinability of each combinatorial function for ACT devices. Of the 111 combinatorial macros available in the ACT 2 and 1200XL libraries, 86 are combinable. Some examples of combining are shown in Figure 4. *For ACT 3 devices, all combinatorial macros that use a single C-module are combinable.*

Following these three rules will ensure that the architecture will be used effectively most of the time for either flip-flops or latches:

1. Use combinable flip-flops whenever possible, these include:

   DF1, DF1B, DFCIB, DFID (See Figure 7.)

2. Use combinable latches whenever possible, these include:

   DL1, DL1B, DLC, DLCA (See Figure 8.)

3. Limit fanin of the flip-flop or batch to one.

Remember that all ACT 3 single C-module macros are combinable.

Table 1 • Logic Modules in Actel Devices

	A1010	A1020	A1225	A1225XL	A1240	A1240XL	A1280	A1280XL	A1415	A1425	A1440	A1460	A14100
Logic Modules	295	547	451	451	684	684	1232	1232	200	310	564	848	1377
S-modules			231	231	348	348	624	624	104	160	288	432	697
C-modules			220	220	336	336	608	608	96	150	276	416	680

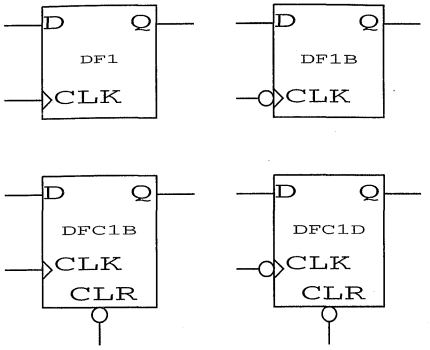


Figure 7 • Combined Flip-flops

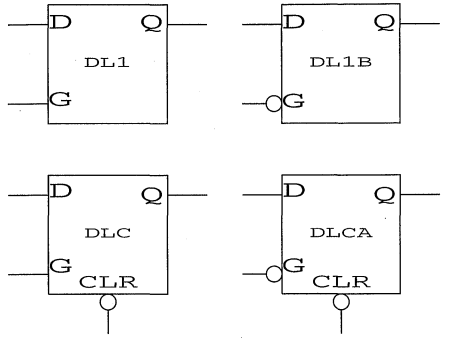


Figure 8 • Combinable Latches

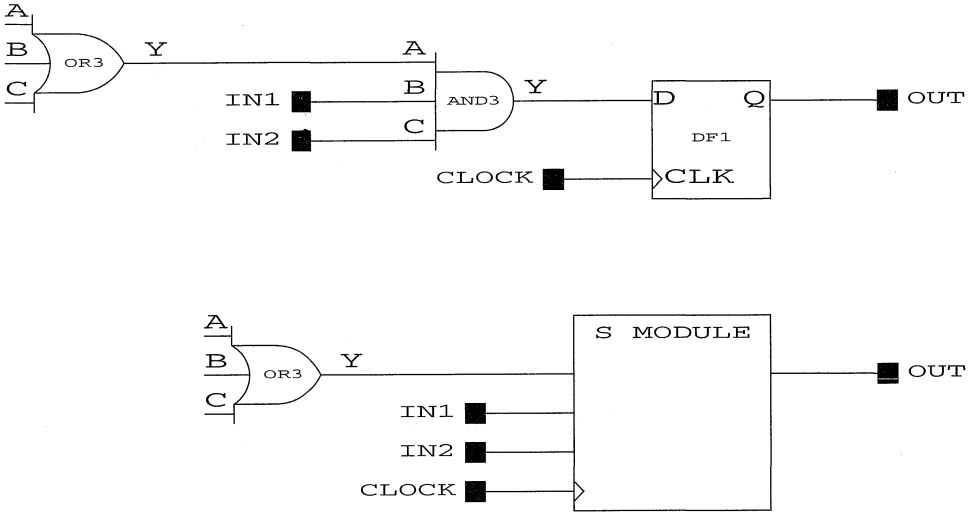


Figure 9 • Combining of AND3 and DF1

Benefits of Combining

Combining a sequential and combinatorial function into a single function increases the gate capacity of the device, since it makes more efficient use of the available gates in the S-module. It also improves the performance of the design because the delay through the combinatorial function is included with the setup time for the flip-flop data input. In effect, the combined logic level has zero propagation delay. Figure 7 shows combining of an ANDS gate and DF1 flip-flop into an S-module.

For designs that require pipelining, the pipeline stages do not incur additional cost if they can be done using combinable logic. An example can be found in the conversion of an adder to an accumulator. Adders may be designed using the ACT 2, 1200XL, and ACT 3 libraries so that every output is combinable. Converting such a design to an accumulator can be done without cost simply by adding a register to the adder output bus in the schematic as shown in Figure 10. ALS will automatically combine each output function with its respective flip-flop in the register into one S-module. The ALS timing analyzer and backannotated simulations will list the propagation delay of combined functions as zero.

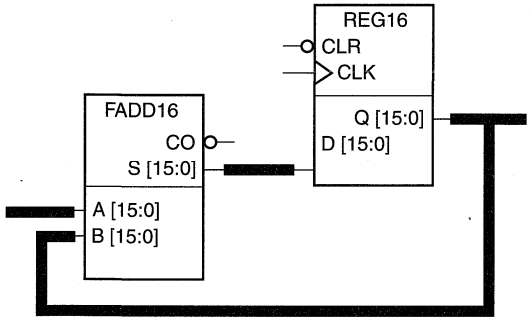


Figure 10 • 16-Bit Accumulator

Summary

Logic modules of ACT 1 devices are eight-input, one-output combinatorial modules. Logic modules of ACT 2, 1200XL, and ACT 3 devices are classified into S-modules and C-modules. Some of the flip-flop functions can be implemented within a single S-module without causing extra delay. Each logic function may have many versions with different combinations of active-low/active-high inputs and outputs.

Speed Improvements of the 1200XL Family

The Actel 1200XL family adds another option for designers requiring high gatecount and mid-to-high-range performance. The 1200XL family is best suited for designs that require 2500 to 8000 equivalent gates and system performance in the range of 40 to 100 MHz—and that are price sensitive. For other applications, the other Actel families may be better suited.

The improvements offered in this family are made possible by a reengineering of the I/O cells and the clock distribution networks and a migration to 0.6 micron process. Like the other Actel families, the 1200XL devices use a two-level metal CMOS process and the patented PLICE antifuse technology. The new streamlined I/O cell allows faster pin-to-pin and clock-to-out delays. The two improved clock distribution networks, which can be driven by an I/O pin or from an internal element, have dramatically reduced propagation delays. For the A1225XL-1 device, clock buffer delay is only 5.8 ns for a fanout of 32 and increases to only 6.5 ns for a load of 256. These are almost 50 percent improvements, compared with the ACT 2 standard devices. Clock skew is also improved and has a maximum value of 1.0 ns with a fanout of 256. All of these improvements translate directly to faster system speeds for your application. Table 1 shows 1200XL family specifications.

To get a better perspective on expected performance with the 1200XL family, the test circuit of Figure 1 was implemented in an Actel 1225A device, with the timing obtained from the static timing tool available in the Designer Series tools, using actual net delays. Since designs can be readily migrated between Actel families, it was a simple task to obtain timing for the 1225XL device as well. In fact, the 1200XL devices are fully die compatible with their ACT 2 counterparts, so it is not necessary to rerun place and route. The test circuit implements eight output pads with logic levels progressing from one to four to measure 1200XL pin-to-pin delays. In addition, there are two paths used to measure the clock-to-out path with a clock load of 32. (Note the use of the GAND macro, which allows connection of the global clock network to a combinatorial macro.) Two paths are included to determine predictability (i.e., minimal variations between delay paths).

The timing results are shown in Table 2 for the standard A1225A, A1225XL parts, as well as for the speed-graded devices A1225A-2 and A1225XL-1. All timing is given for commercial worst-case conditions (i.e., worst-case processing, 70c and 4.75 volts). N shown in the table refers to the number of logic levels in the given path. The table shows, for example, that a pad-to-pad delay with one logic level for the 1225XL-1 is between 13.2 and 13.6 ns, or approximately 75 MHz. The "REG" columns represent the clock-to-out path from clock pad to the latched output buffer with TTL thresholds and 35 pF loading. (An internal latch with inverted gate combined with the latched output forms a register.) Clock-to-out for the 1225XL-1 is between 12.7 and 13.0 ns, or approximately 78 MHz.

Comparing the 1225XL-2 devices with the standard 1225A part shows substantial speed improvements over these paths. These range from 43 percent for a four-logic-level path to

	A1225XL-1	A1240XL-1	A1280XL-1
Gate Capacity	2500	4000	8000
Equivalent 32 macro-cell CPLDs	5	8	16
Equivalent TTL devices	63	100	200
Logic Modules	451	684	1232
Dedicated Flip-flops	231	348	624
User I/Os	83	104	140
Packages	84PLCC 100 PQFP 100 VQFP 100 CPGA	84 PLCC 144 PQFP 176 TQFP 132 CPGA	84 PLCC 160 PQFP 176 TQFP 176 CPGA
Clock rates (counters, datapath)	135 MHz	130 MHz	110 MHz
Clock-out (pad-pad)	10.0 ns	10.5 ns	11.0 ns
Input setup	0.4 ns	0.4 ns	0.4 ns

Table 1 • 1200XL Family Specifications

56 percent for the clock-to-out path. The final analysis of this data shows the high predictability of the Actel devices—the variation between similar paths, that directly reflects the extent to which the designer can rely on the routing to provide consistent timing results, which are estimated early in the design via the device datasheet. Predictability for the design is above 97 percent for all devices. This is, of course, a very small design, so this predictability data is a little suspect. However, in other larger design analysis, including PREP benchmarks, Actel has a predictability value of approximately 90 percent.

Two of the 1200XL devices were benchmarked with respect to the PREP Benchmark Suite #1. (Please consult Section 5 of this data book for more information on PREP benchmarks.)

Table 3 shows the results for all nine PREP circuits for the 1225XL-1 device, and the 1225A-2 for comparisons. All numbers indicate frequency in MHz. Table 4 shows similar results for the 1280XL-1 device.

The 1200XL family is the Actel value family for designs that require mid-to-high performance with high logic density and that are cost sensitive. The design flow to take advantage of these new devices is fast and easy using the Actel Designer Series tools if your methodology is traditional schematic capture, Boolean entry, or behavioral HDL. And the process is even further simplified if you already have an Actel design, since migration between families uses the same tools, and most of the library elements between families are common.

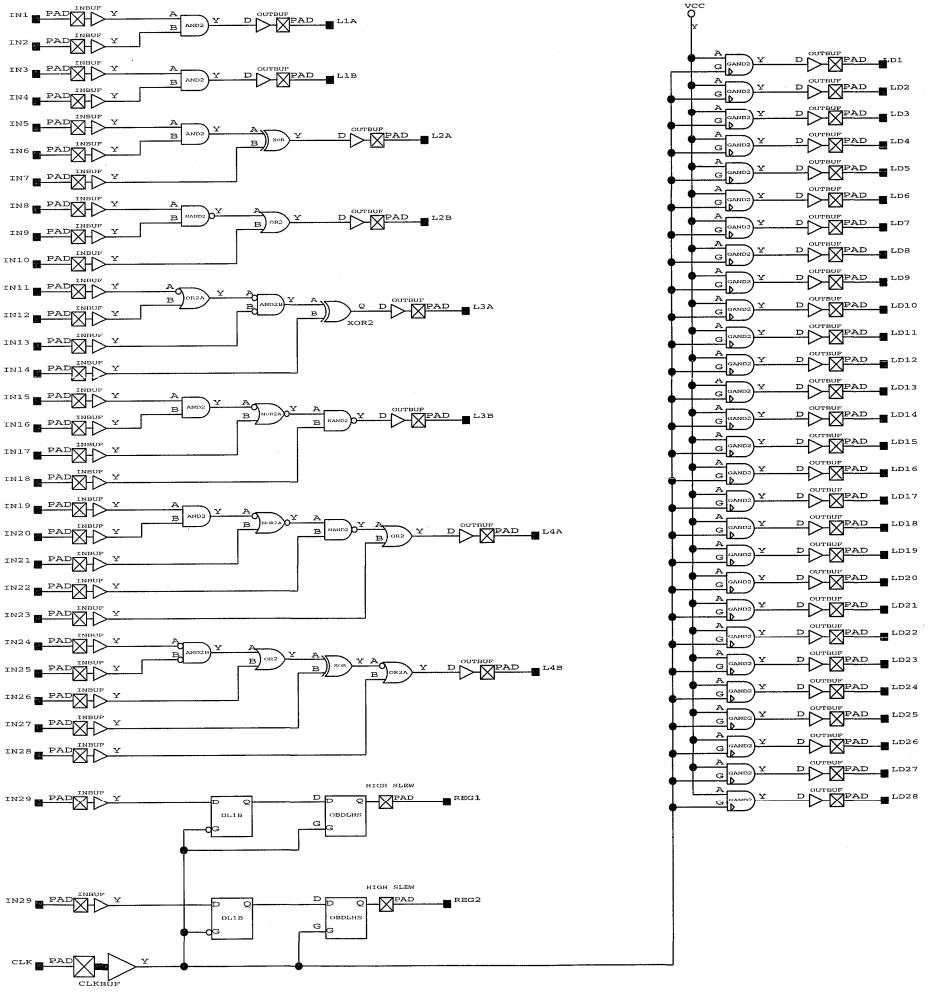


Figure 1 • A1225A/A1225XL Timing Test Circuit

Table 2 • 1200XL Timing Comparison

Each path contains INBUF, N Logic Levels and an OUTBUF, or INBUF, REG, and OUTBUF.

Columns N=1 to N=4 below are pad-to-pad propagation delays.

Columns REG below are clock-to-out propagation delays.

Device	N=1	N=1	N=2	N=2	N=3	N=3	N=4	N=4	REG	REG
1225A	25.8	25.9	32.0	33.6	39.6	39.8	46.4	46.9	29.1	29.5
1225A-2	19.4	19.5	24.0	25.2	29.7	29.9	34.8	35.2	21.9	22.1
1225XL	15.5	16.0	20.3	21.2	25.3	25.5	31.4	31.4	11.0	11.8
1225XL-1	13.2	13.6	17.2	18.0	21.5	21.6	26.7	26.7	9.4	10.0
Improvement Compared with 1225A										
1225A-2	25%		25%		25%		25%		25%	
1225XL	39%		37%		36%		33%		60%	
1225XL-1	48%		46%		46%		43%		67%	
Predictability = 1– (max – min)/average for similar paths										
1225A	98.4%									
1225A-2	98.4%									
1225XL	97.8%									
1225XL-1	97.9%									

Table 3 • 1225XL-1 PREP Timing Results

Device	Param	PREP1	PREP2	PREP3	PREP4	PREP5	PREP6	PREP7	PREP8	PREP9
1225A-2	Fmax	105	41	45	26	21	39	70	105	46
	Fmin	105	37	41	25	20	34	63	105	41
	Fmean	105	39	43	25	21	37	66	105	43
	Fext	49	40	26	18	16	23	34	42	27
1225XL-1	Fmax	135	47	54	29	22	42	76	135	53
	Fmin	112	41	47	28	21	37	71	122	48
	Fmean	126	45	51	29	21	40	74	132	50
	Fext	60	47	34	21	18	29	37	47	30
delta	Fmax	29%	15%	20%	12%	5%	8%	9%	29%	15%
	Fmin	7%	11%	15%	12%	5%	9%	13%	16%	17%
	Fmean	20%	15%	19%	16%	0%	8%	12%	26%	16%
	Fext	22%	18%	31%	17%	13%	26%	9%	12%	11%

Table 4 • 1280XL PREP Timing Results

Device	Param	PREP1	PREP2	PREP3	PREP4	PREP5	PREP6	PREP7	PREP8	PREP9
1280A-2	Fmax	85	41	44	25	18	36	67	85	44
	Fmin	85	34	33	23	16	30	53	71	36
	Fmean	85	38	40	24	17	33	58	84	41
	Fext	48	40	25	17	13	24	28	41	26
1280XL-1	Fmax	115	48	53	30	21	39	72	100	53
	Fmin	98	38	39	26	18	30	56	77	40
	Fmean	110	44	47	28	20	35	62	96	48
	Fext	68	47	33	21	18	30	31	48	34
delta	Fmax	35%	17%	20%	20%	17%	8%	7%	18%	20%
	Fmin	15%	12%	18%	13%	13%	0%	6%	8%	11%
	Fmean	29%	16%	18%	17%	18%	6%	7%	14%	17%
	Fext	42%	18%	32%	24%	38%	25%	11%	17%	31%

Using ACT 3 Family I/O Macros

The ACT 3 Family from Actel has a complex I/O macro which allows users to implement high speed complex functions entirely in the I/O. Common functions like shift registers operating at 160Mhz, state machines and counters with 7.5 ns clock to output (pin to pin) and 160 Mhz pipeline registers are all implementable in ACT 3 I/Os. This significantly increases capacity and I/O performance of ACT 3 over previous generations of FPGAs. This application note covers several examples of how designers can use these powerful IO macros in real world applications.

ACT 3 I/O Architecture

The ACT 3 family architecture is shown in Figure 1. The logic array consists of rows of Sequential and Combinatorial logic cells with routing channels in between, much like a channeled gate array. I/O modules are located at the periphery of the array. Each I/O module is used to interconnect device signal pins to array inputs and outputs. The I/O module contains specialized circuitry to assist the designer in getting signal on and off chip. Figure 2 shows the detailed block diagram of the ACT 3 I/O module.

I/O Module

The main function of the ACT 3 I/O module is to provide a high-speed high-functionality interface between the pins of the device and the logic array. Data to/from the logic array arrives on the left side of Figure 2. U01 and U02 are the data from the array and 'Y' is the data to the array. The pad is shown on the far right side of Figure 2 and is the input/output pin of the device. Data from the array may be registered in the output register or bypassed using the OTB control signal. If registered, the data may be selectively loaded or held via the synchronous enable signal ODE and asynchronously initialized to a logic Low or High via the IOPCL input. The output drivers have both an individual slew control and individual output enable. The Preset/Clear, OTB and Slew features are usually automatically determined when the user selects the desired I/O macro. For example, the I/O macro ORECTH shown in Figure 3A uses a Clearable output register with High Slew. Correct selections of OTB, Preset/Clear and Slew are made automatically. Another feature of the I/O macro is the ability to feedback the contents of the output register back to the logic array. This allows embedded functions like state machines and shift registers to be easily implemented in the IO macro. Figure 3B shows the FECTML macro and the feedback multiplexor used to select between the output register and the device pad.

When data is sources from the device pad, the input register can be used to capture incoming data. The input register has a synchronous data enable and a Preset/Clear feature similar to the output register. The IDE and IEN signals are used to control the Y multiplexer and the data enable multiplexer and provide the most commonly needed input functions. Again, these signals are usually determined automatically when the user selected the desired macro. For example, the IREC input register macro is shown in Figure 3C and has IDE, IEN and Preset/Clear predetermined to implement the desired functions.

The complex I/O functions available with ACT 3 devices have many common applications. The following sections show several ways in which these applications can make full use of the ACT 3 I/O capabilities in 'real world' systems designs. The three sections are:

- Input Oriented Functions
- Output Oriented Functions
- Bi-Directional Oriented Functions

Table 1 • ACT 3 I/O Macro Performance

Input Features	
Input Timing Register	167 MHz
Input Pipeline Register	167 MHz
Pulse Stretching Input Register	120 MHz
Input Synchronization	167 MHz
Input Address Holding Register	164 MHz
Input Control Register	167 MHz
Output Functions	
State Machine Output Register	135 MHz
Output Shift Register	167 MHz
Output Pseudo Random Counter	131 MHz
Output Datapath	131 MHz
Bi-Directional Functions	
Bi-Directional Registers	167 MHz

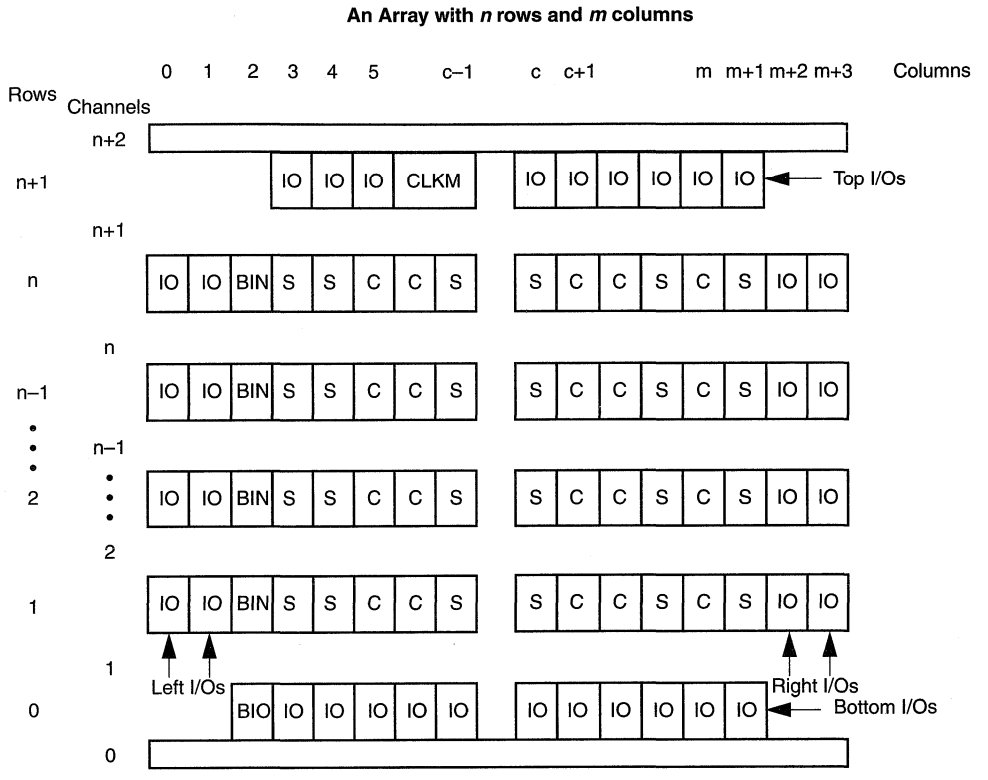


Figure 1 • Generalized Floor Plan of ACT 3 Device

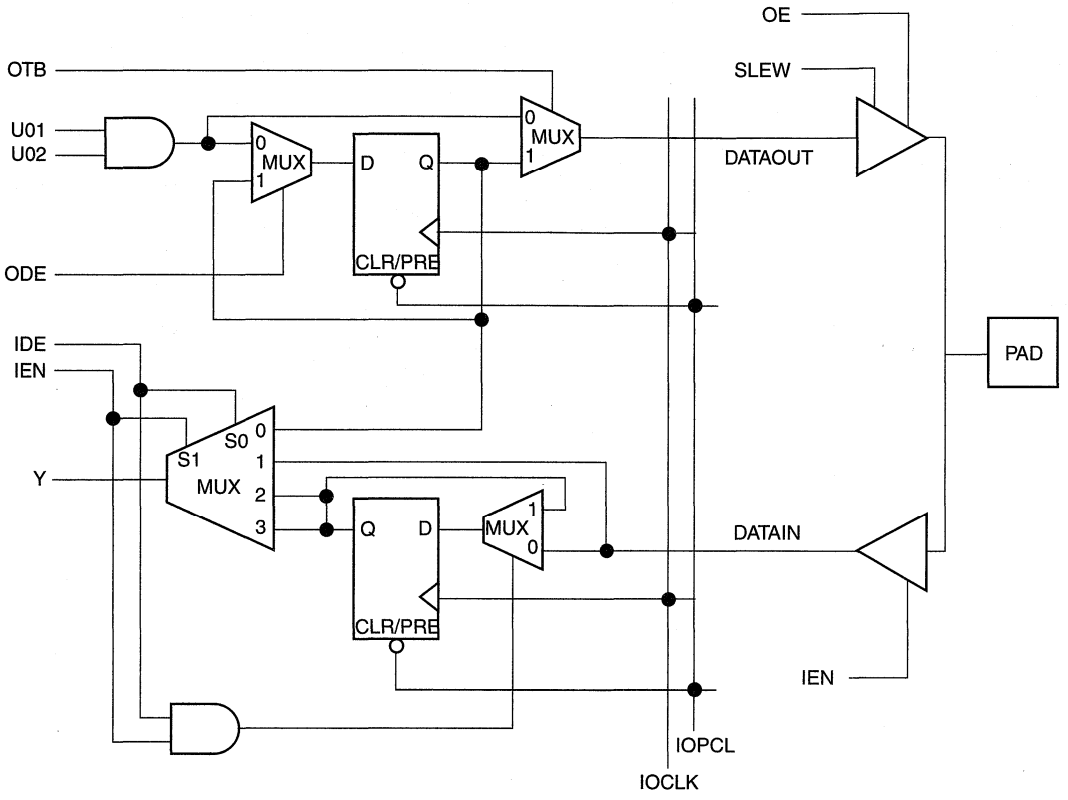


Figure 2 • I/O Macro Block Diagram

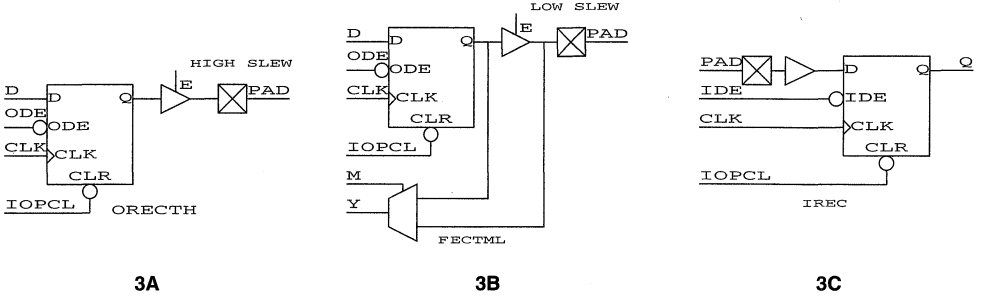


Figure 3 • I/O Macro Implementations

Every registered ACT 3 I/O macro (for example, DECETH, IREC, and OREPTL) includes an asynchronous preset or clear input IOPCL. This input asynchronously sets the output of I/O flip-flops to 0 or 1. The IOPCL pin of a registered I/O macro must be driven by the dedicated I/O macro, IOPCLBUF. The IOPCLBUF is externally driven in turn by the dedicated IOPCL package pin as shown in Figure 4.

Similarly, the clock (CLK) inputs of registered I/O macros are driven by a dedicated circuit. The dedicated I/O clock buffer, IOCLKBUF, is used to drive the clock pin of each I/O macro. The IOCLKBUF is driven in turn by the dedicated external package pin, IOCLK, as shown in Figure 5. See the package pin assignment diagrams in the ACT 3 datasheet for specific locations of the IOPCL and IOCLK package pins.

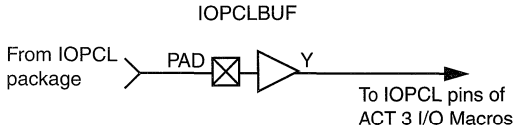


Figure 4 • An IOPCLBUF Driven by a Dedicated IOPCL Package Pin

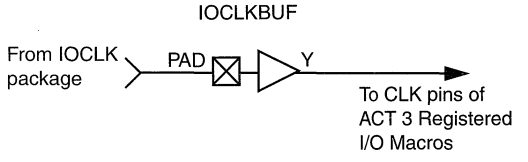


Figure 5 • An IOCLKBUF Driven by a Dedicated IOCLK Package Pin

If no registered I/O macros are used in an ACT 3 design, then the dedicated IOCLK and IOPCL package pins can be used as normal I/O pins. In that case, these pins can be connected to any other type of I/O buffer except IOCLKBUF, IOPCLBUF, and HCLKBUF. For example, undedicated normal I/O buffers such as INBUF and OUTBUF may be connected to these pins. If any registered I/O macros are used in an ACT 3 design, then the IOCLKBUF and IOPCLBUF macros must be included in the design, and the pin assignment must include the IOPCL and IOCLK pins. The IOPCL and IOCLK pins must be explicitly specified because ALS will not automatically create pin assignments for IOCLKBUF and IOPCLBUF.

In an ACT 3 design, all types of I/O macros can be used in a single design. For example, I/O macros without registers, I/O macros with preset, and I/O macros with clear can be used together in one design. However, IOCLKBUF must drive all the clock inputs of the registered I/O macros and the IOPCL must drive all the clear or preset inputs of the registered I/O macros. CLK and IOPCL pins of registered I/O macros can not be connected to GND or V_{CC}. All ACT 3 I/O registered buffers share the same dedicated clock and asynchronous clear or preset network. Once the built-in dedicated IOCLK and IOPCL networks are used, all registered I/O macros will be cleared and preset at the same time by the same signal.

Input Oriented Functions

The ACT 3 IO macro has several functions useful when bringing data onto the device. In particular the input data enable control (IDE), the input register bypass multiplexer (and its control signal IMSEL) and the asynchronous PRESET or CLEAR functions can all be used to implement important functions in the IO.

Common Input Oriented macros are shown in Figure 6. Application examples and tips in using these types of macros are shown in more detail starting on page XX.

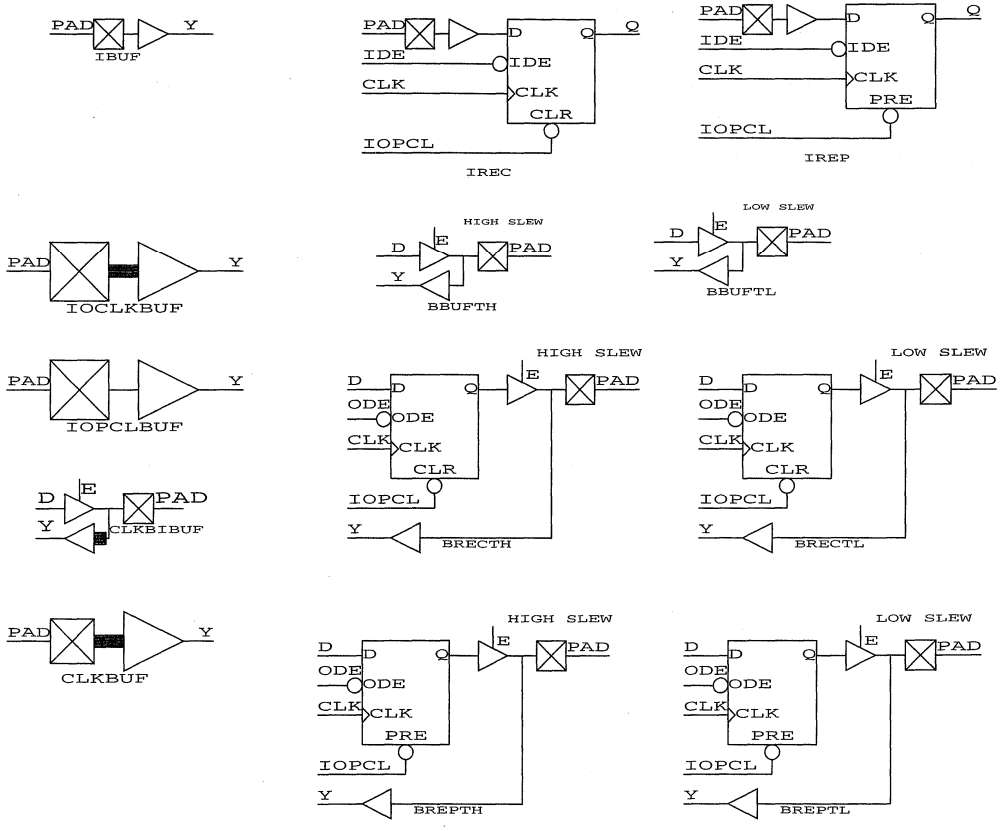


Figure 6 • Common Input Oriented macros

Input Timing Register

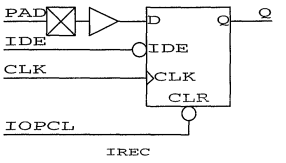
Description

ACT 3 I/O macros can be used to implement registers used to capture input signals with precise timing using the IO Clock (IOclk) in conjunction with the High Speed Internal Clock (Hclk). Because they are separate hardwired clock networks, with very low skew (1.3 ns max) IOclk may be skewed with respect to Hclk to adjust timing precisely.

Examples and Tips

- Use the IREC and IREP macros to implement timing registers on input paths. Example applications are asynchronous input metastability hardening, synchronous data bus timing, address sampling, and control signal capturing.
- Use the data enable and the data input on the register to implement complex timing functions.
- Use the Preset or Clear version of the macro in order to determine the state of the associated input on initialization. This is insures that all required control inputs are inactive on initialization.

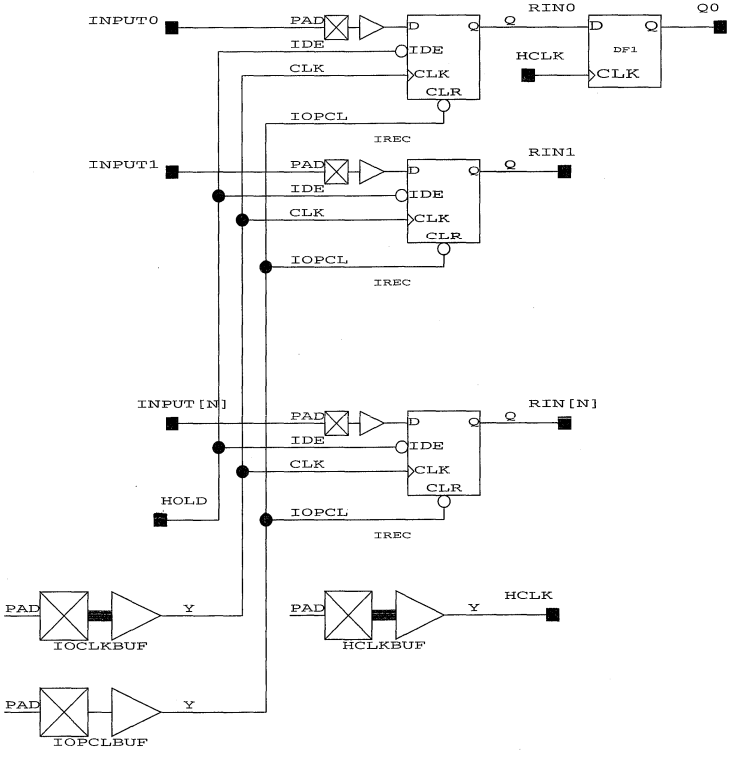
Macro Diagram—IREC



Timing Analysis

A1425A-2	Worst Case Commercial Delays
Input Set-up Time	**1.5 ns**
Clock to Output	3.0 ns
Routing Delay(FO=2)	1.4 ns
Total	**4.4 ns**
Fmax (Clock Limited)	**167 MHz**

Application Diagram—Input Timing Register



Input Pipeline Register

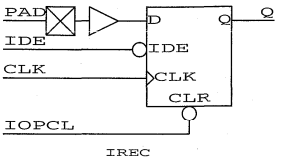
Description

ACT 3 I/O macros can be used to implement pipeline registers in high performance systems. Pipelined processing inserts register stages between processing elements to increase processing clock rate. This technique increases signal latency, but in latency friendly applications this is a powerful technique. The ACT 3 I/O registers are ideal for registering data going on or off chip in synchronous pipeline processing applications.

Examples and Tips

- Use the IREC or IREP macro to pipeline data on input pins. Typical applications are digital signal processing, high speed communications datapaths, graphics processing and networking.
- Use the data enable and the data input on the register to hold data for multiple clock cycles.
- Use the Preset or Clear version of the macro in order to determine the state of the associated output on initialization. This insures that all required control outputs are inactive on initialization.
- Use the input bypass multiplexer on the IRECM macro to bypass the pipeline register when flushing the pipe.

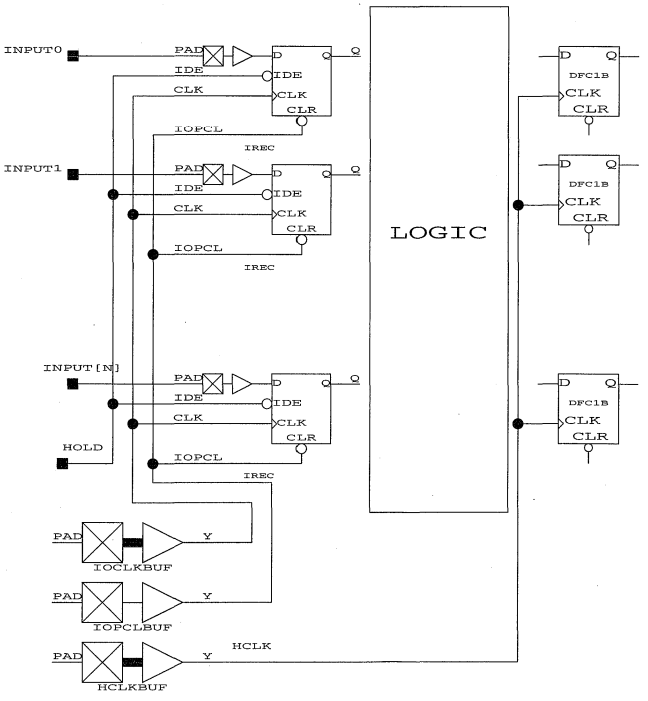
Macro Diagram—IREC



Timing Analysis

A1425A-2	Worst Case Commercial Delays
Input Set-up Time	**1.5 ns**
Clock to Output	3.0 ns
Routing Delay (FO=2)	1.4 ns
S-module Set-up time	0.8 ns
Total	**5.2 ns**
Fmax (Clock Limited)	**167 MHz**

Application Diagram—Input Pipeline Register



Pulse Stretching Input Register

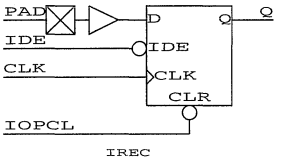
Description

ACT 3 I/O macros can be used to stretch timing pulses when control signals dissappear too soon. The enable on the input register can be used to convieniently hold the captured data until processing has been completed..

Examples and Tips

- Use IREC and IREP input registers to stretch input pulses. Typical applications are in peripheral interfaces, asynchronous bus control and networking.
- Use the Preset or Clear version of the macro in order to determine the state of the associated output on initialization. This is insures that all required control outputs are inactive on initialization.
- Use the IO Clock and the High Speed Clock to adjust timing as needed.

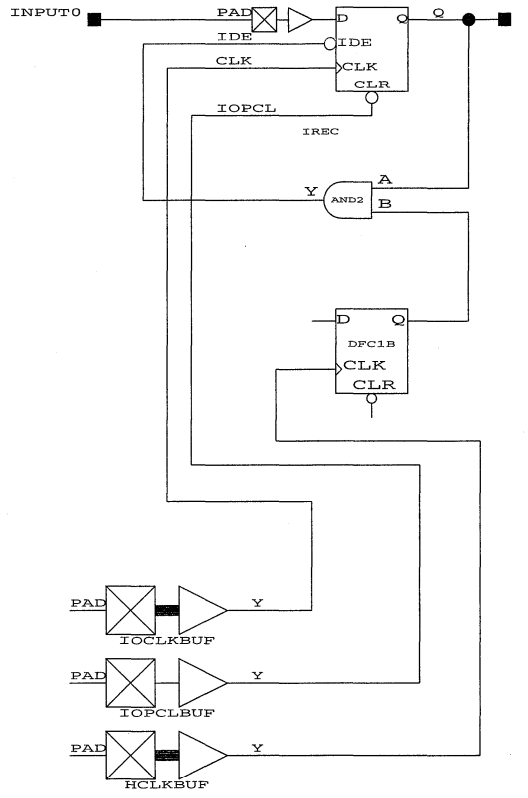
Macro Diagram—IREC



Timing Analysis

A1425A-2	Worst Case Commercial Delays
Input Set-up Time	**1.5 ns**
Clock to Output	3.0 ns
Routing Delay (FO=2)	1.4 ns
Logic Module delay	2.3 ns
Routing Delay (FO=1)	1.0 ns
Enable Set-up	0.6 ns
Total	**8.3 ns**
Fmax	**120 MHz**

Application Diagram—Pulse Stretching Input Register



Input Synchronization

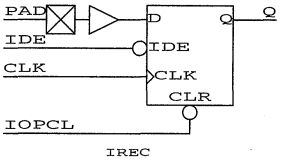
Description

ACT 3 I/O macros can be used to help synchronize asynchronous inputs and increase metastability hardness. The IO macro register is used as the first stage of the synchronizer with additional stages provided by the logic array.

Examples and Tips

- Use the IREC and IREP input registers to synchronize asynchronous inputs. Typical applications include peripheral interface, industrial controls and data acquisition.
- Use the Preset or Clear version of the macro in order to determine the state of the associated output on initialization. This is insures that all required control outputs are inactive on initialization.

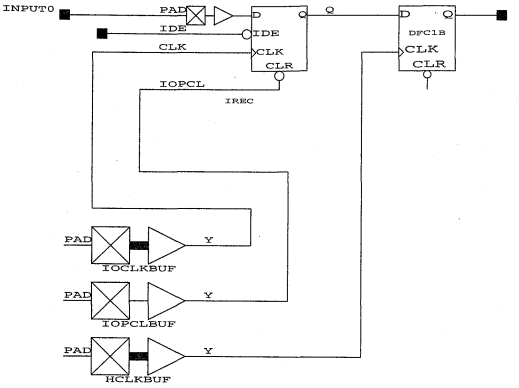
Macro Diagram—IREC



Timing Analysis

A1425A-2	Worst Case Commercial Delays
Input Set-up Time	**1.5 ns**
Clock to Output	3.0 ns
Routing Delay (FO=2)	1.4 ns
S-module Set-up time	0.8 ns
Total	**5.2 ns**
Fmax (Clock Limited)	**167 MHz**

Application Diagram—Input Synchronization



Input Address Holding Register

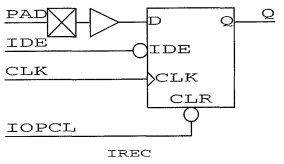
Description

ACT 3 I/O macros can be used to implement address holding registers on processor interface applications. The data enable on the input register can be used to capture addresses and hold them during the full processing cycle. The address register output can also be multiplexed with the input pin using the input register bypass multiplexer. This is useful when the address needs to be saved for later use, but the information on the bus (perhaps data on a multiplexed address/data bus) needs to be available also.

Examples and Tips

- Use the IREC and IREP macros to implement address holding registers. Typical applications include synchronous bus interfaces, multiplexed address/data busses and memory control.
- Use the Preset or Clear version of the macro in order to determine the state of the associated output on initialization. This is insures that all required control outputs are inactive on initialization.
- Use IO Clock and input data enable in conjunction with the High Speed Clock to control timing of address capture.

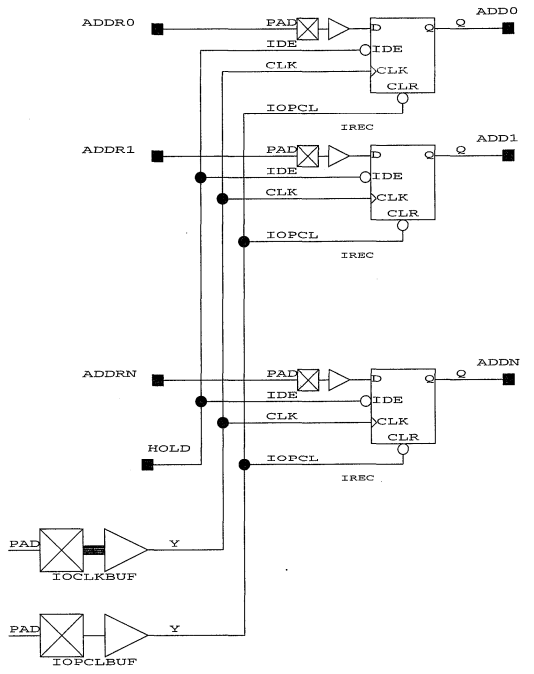
Macro Diagram—IREC



Timing Analysis

A1425A-2	Worst Case Commercial Delays
Input Set-up Time	**1.5 ns**
Logic Module	2.3 ns
Routing Delay (FO=8)	3.2 ns
IO Module enable set-up	0.6 ns
Total	**6.1 ns**
Fmax (Clock Limited)	**164 MHz**

Application Diagram—Input Address Holding Register



Input Control Register

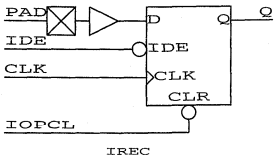
Description

ACT 3 I/O macros can be used to implement input control registers for capturing control signals for use inside the device. A simple request acknowledge handshake can be implemented completely in the IO macros in ACT 3 Devices.

Examples and Tips

- Use the IREC and IREP IO macros to implement simple control register functions. Typical applications are request acknowledge handshakes, interrupt requests, chip select, data strobe enables and master slave bus arbitration.
- Use the data enable and the data input on the register to implement complex transition functions.
- Use the Preset or Clear version of the macro in order to determine the state of the associated output on initialization. This is insures that all required control outputs are inactive on initialization.
- Use IO Clock in conjunction with the High Speed Clock to solve perverse timing problems when interfacing to unsynchronous busses.

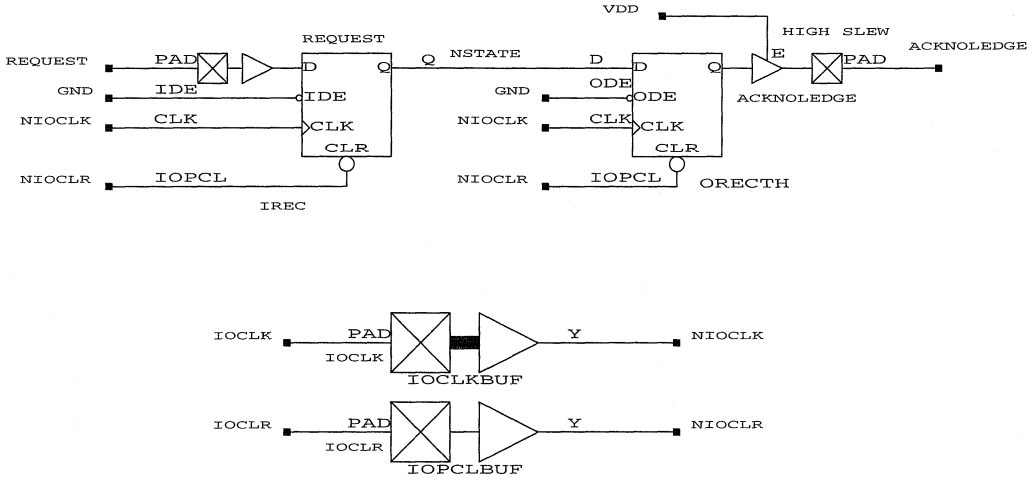
Macro Diagram—IREC



Timing Analysis

A1425A-2	Worst Case Commercial Delays
Input Set-up Time	**1.5 ns**
Clock to Out (Pad to Pad)	7.5 ns
Clock to Output	3.0 ns
Routing Delay (FO=2)	1.4 ns
S-module Set-up time	0.8 ns
Total	**5.2 ns**
Fmax (Clock Limited)	**167 MHz**

Application Diagram—Input Control Register



Output Oriented Functions

The ACT 3 IO macro has several functions useful when bringing data off the device. In particular the output data enable control (IDE), the output register bypass multiplexer (and its control signal OMSEL), the feedback multiplexer and the asynchronous PRESET or CLEAR functions can all be used to implement important functions in the IO.

Common Output Oriented macros are shown Figure 7. Application examples and tips in using these types of macros are shown in more detail starting on page XX.

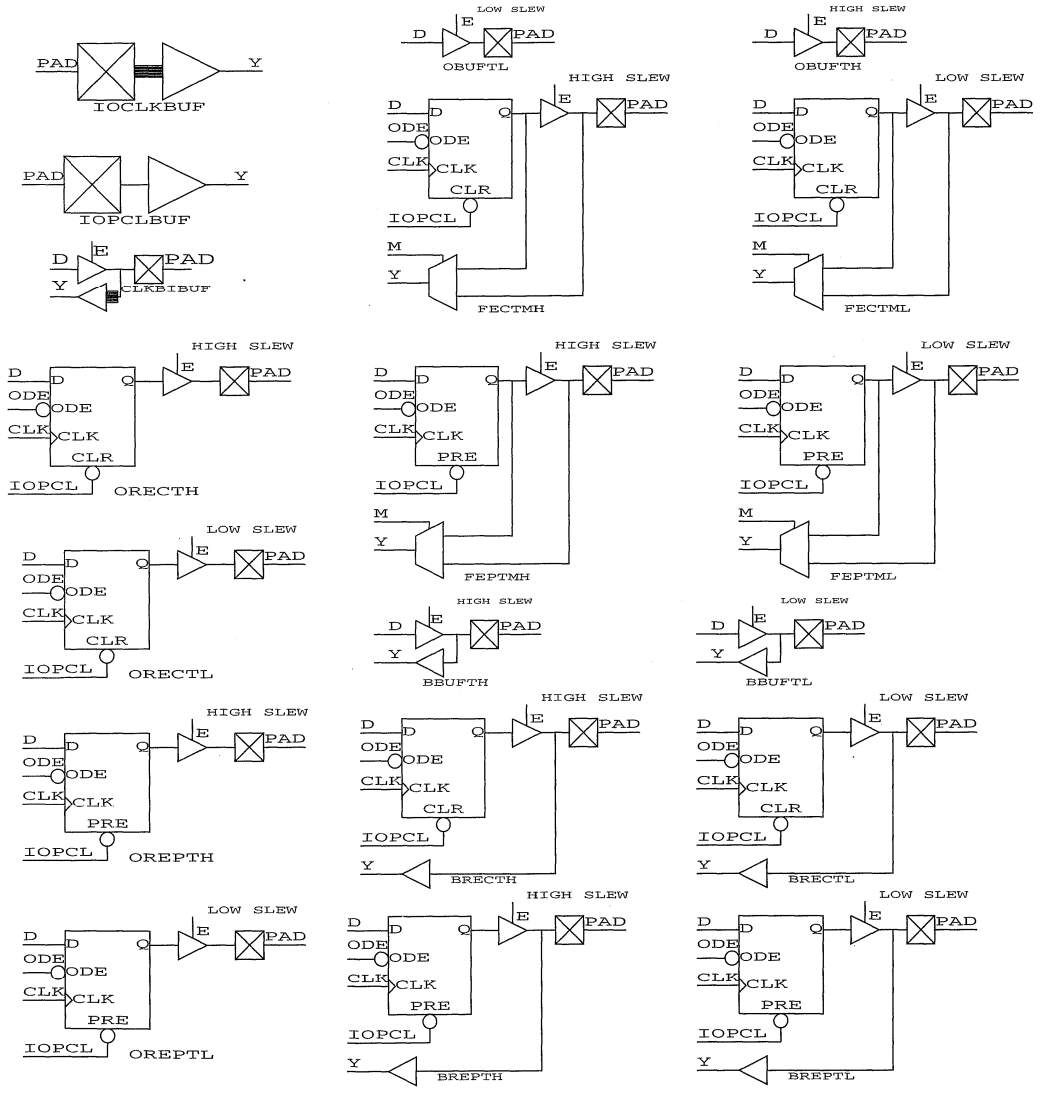


Figure 7 • Common Output Oriented macros

State Machine Output Register

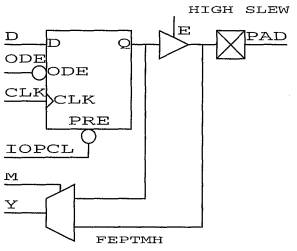
Description

ACT 3 I/O macros can be used to implement state registers for control signals which need fast clock to output. Typical applications are DMA controllers, DRAM interface, RISC Processor Interface, fast interchip signaling, timing generation and control, or interleaved memory control.

Examples and Tips

- Use the FEPTMH macro with the feedback multiplexor selected for the direct register output (Low on the select). This cuts the delay by several ns since it need not go through both the output buffer and input buffer. Remember to use a stop set in the timer with the PADs of each register included so this longer path won't be used (The timer can't tell that the mux has selected the internal path).
- Use the data enable and the data input on the register to implement complex transition functions.
- Use the Preset or Clear version of the macro in order to determine the state of the associated output on initialization. This is insures that all required control outputs are inactive on initialization.

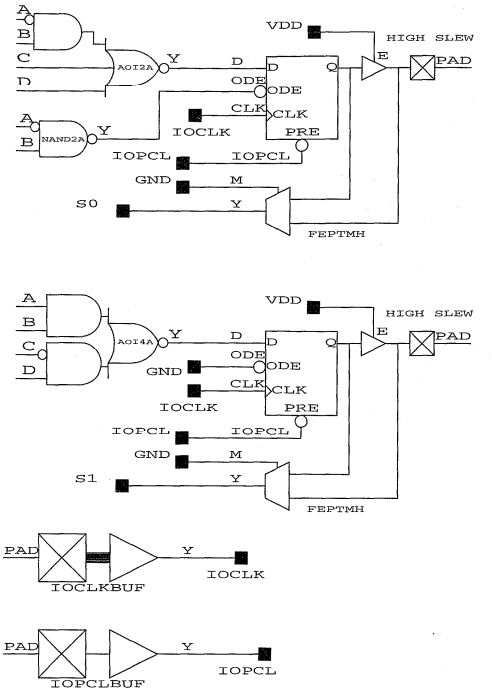
Macro Diagram—FEPTMH



Timing Analysis

A1425A-2	Worst Case Commercial Delays
Clock to Out	3.5 ns
Module Delay (AOI2A)	2.3 ns
Routing Delay (FO=1)	1.0 ns
IO Reg set-up	0.6 ns
Total Delay	**7.4 ns**
Fmax	**135 MHz**
Clock to Out (Pad to Pad)	**7.5 ns**

Application Diagram—State Machine Output Register



Output Shift Register

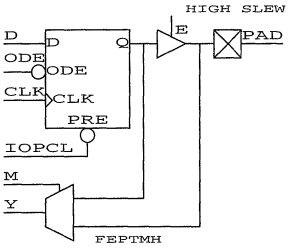
Description

ACT 3 I/O macros can be used to implement output shift registers utilizing only IO macros. Typical applications are serial communications, network interfaces, bus bandwidth matching and timing generation.

Examples and Tips

- Use the FEPTMH macro with the feedback multiplexor selected for the direct register output (Low on the select). This cuts the delay by several ns since it need not go through both the output buffer and input buffer. Remember to use a stop set in the timer with the PADs of each register included so this longer path won't be used (The timer can't tell that the mux has selected the internal path).
- Use the data enable and the data input on the register to control the transfer of data off chip.
- Use the Preset or Clear version of the macro in order to determine the state of the associated output on initialization. This is insures that all required control outputs are inactive on initialization.

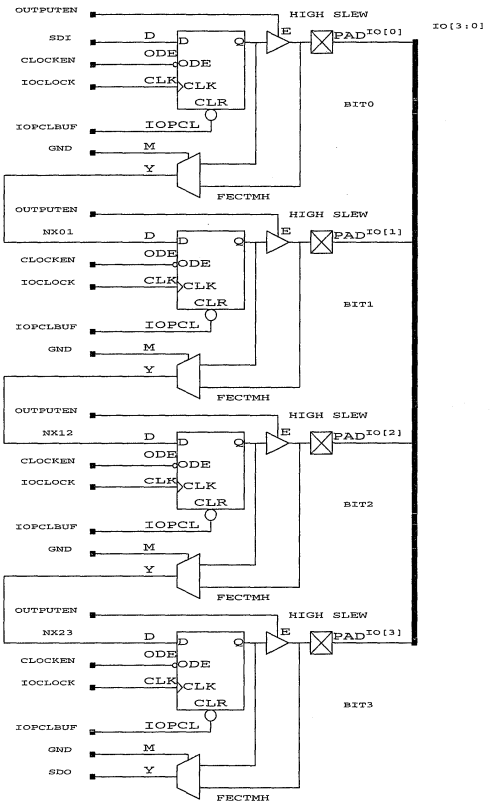
Macro Diagram—FEPTMH



Timing Analysis

A1425A-2	Worst Case Commercial Delays
Clock to Out	3.5 ns
IO Reg set-up	0.6 ns
Total Delay	**4.1 ns**
Fmax (Clock Limited)	**167 MHz**
Clock to Out (Pad to Pad)	**7.5 ns**

Application Diagram—Output Shift Register



Output Pseudo Random Counter

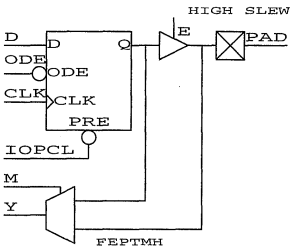
Description

ACT 3 I/O macros can be used to implement pseudo random counters for high speed counting and timing functions. This technique results in faster and smaller implementations than traditional binary counters and only costs 1 combination in N^2. Typical applications are FIFO memories, large scratch pad storage, interprocess data transfers and high precision delay generation. Remember to use timing tricks with the IOclock if needed to satisfy perverse system timing constraints.

Examples and Tips

- Use the FEPTMH macro with the feedback multiplexor selected for the direct register output (Low on the select). This cuts the delay by several ns since it need not go through both the output buffer and input buffer. Remember to use a stop set in the timer with the PADs of each register included so this longer path won't be used (The timer can't tell that the mux has selected the internal path).
- Use the data enable and the data input on the register to control timing.
- Use the Preset or Clear version of the macro in order to determine the state of the associated output on initialization. This is insures that all required control outputs are inactive on initialization.

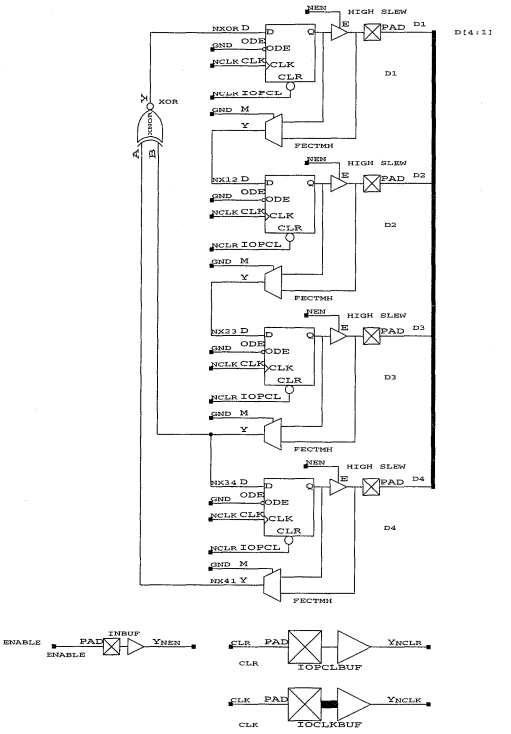
Macro Diagram—FEPTMH



Timing Analysis

A1425A-2	Worst Case Commercial Delays
Clock to Out	3.5 ns
Module Delay (XNOR)	2.3 ns
Routing Delay (FO=1)	1.0 ns
IO Reg set-up	0.8 ns
Total Delay	**7.6 ns**
Fmax	**131 MHz**
Clock to Out (Pad to Pad)	**7.5 ns**

Application Diagram—Output Pseudo Random Counter



Output Datapath

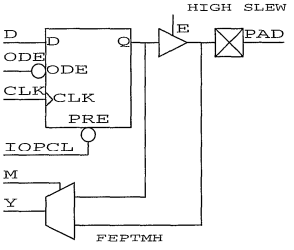
Description

ACT 3 I/O macros can be used to implement datapath output registers for common datapath functions like counters, adders, multiplexers, and pipeline registers. Typical applications are digital signal processing, microprocessor peripherals, communications datapaths, networking controllers and algorithm enhancement coprocessors.

Examples and Tips

- Use the FEPTMH macro with the feedback multiplexor selected for the direct register output (Low on the select). This cuts the delay by several ns since it need not go through both the output buffer and input buffer. Remember to use a stop set in the timer with the PADs of each register included so this longer path won't be used (The timer can't tell that the mux has selected the internal path).
- Use the data enable and the data input on the register to control data transfers off chip.
- Use the Preset or Clear version of the macro in order to determine the state of the associated output on initialization. This is insures that all required control outputs are inactive on initialization.

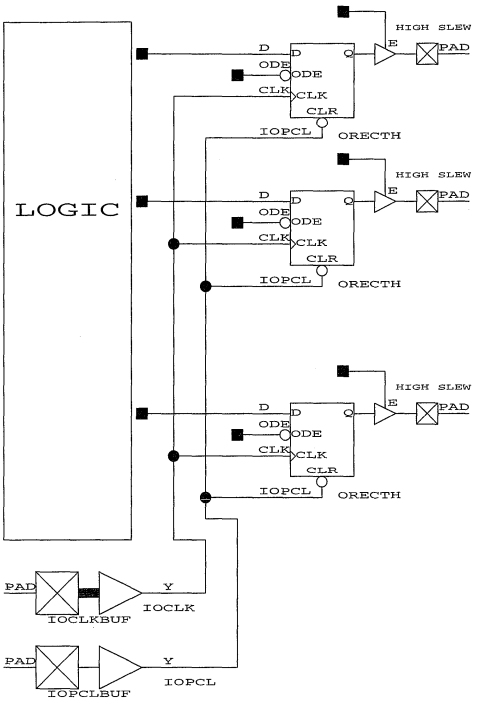
Macro Diagram—FEPTMH



Timing Analysis

A1425A-2	Worst Case Commercial Delays
Clock to Out	3.5 ns
Module Delay	2.3 ns
Routing Delay (FO=1)	1.0 ns
IO Reg set-up	0.8 ns
Total Delay	**7.6 ns**
Fmax	**131 MHz**
Clock to Output (Pad to Pad)	**7.5 ns**

Application Diagram—Output Datapath



Bi-Directional Oriented Functions

The ACT 3 IO macro has several functions useful when implementing bi-directional functions. In particular, the input register and output register can be used together to implement synchronous bus interfaces. Preset and Clear as well as separate data enables on each register provide additional functionality in synchronous bus applications.

Common Bi-directional Oriented macros are shown in Figure 8. Application examples and tips in using these types of macros are shown in more detail starting on page XX.

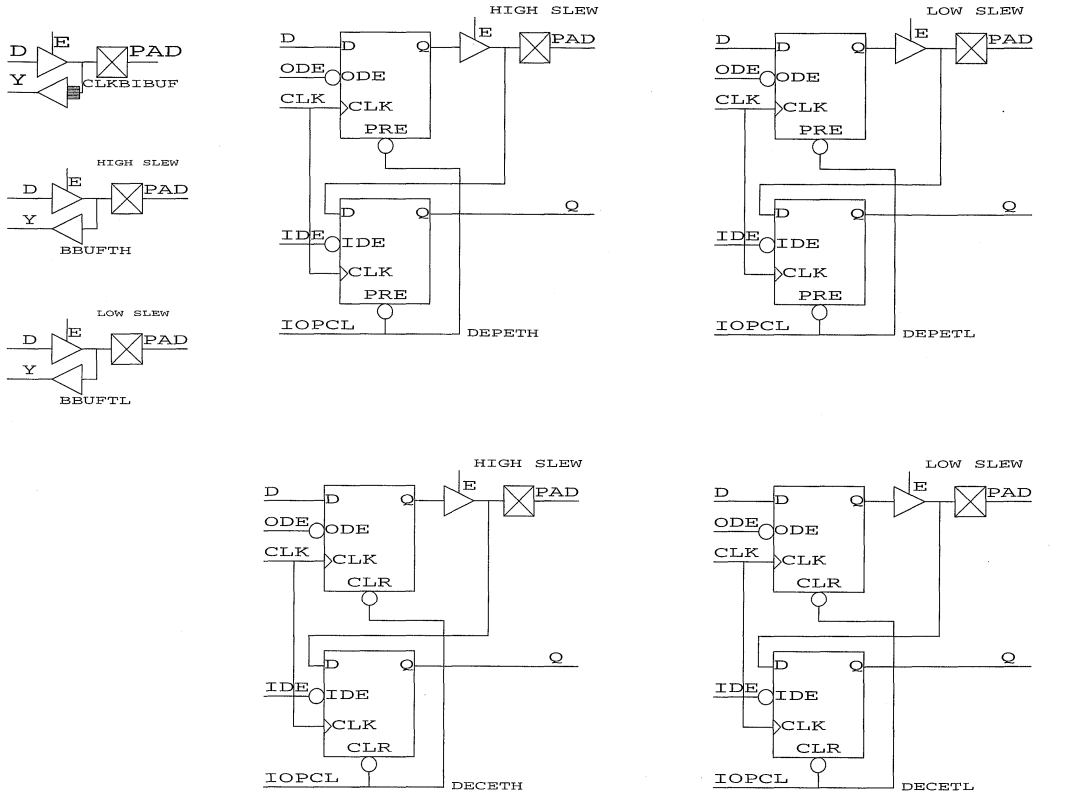


Figure 8 • Common Bi-Directional Oriented macros

Bi-Directional Registers

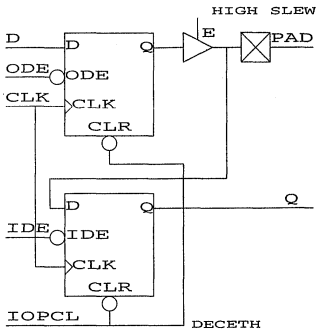
Description

ACT 3 I/O macros can be used to implement bi-directional registers for synchronous bus interfaces. Typical applications are synchronous processor bus interfaces, synchronous memory interfaces, interchip bus interfaces, high speed communications datapath switches, and high speed peripheral interfaces. The separate IO clock network can be used in conjunction with the high speed internal clock to adjust timing of data transfers to meet difficult system timing requirements.

Examples and Tips

- Use the DECETH macro to implement synchronous registered bi-directional data busses in ACT 3 devices.
- Use the data enables to control data flow to/from the bus.
- Use the Preset or Clear version of the macro in order to determine the state of the associated output on initialization. This is insures that all required control outputs are inactive on initialization. (Remember that preset and clear apply to both registers. Registers can only be both preset or both cleared. Other combinations are not allowed.

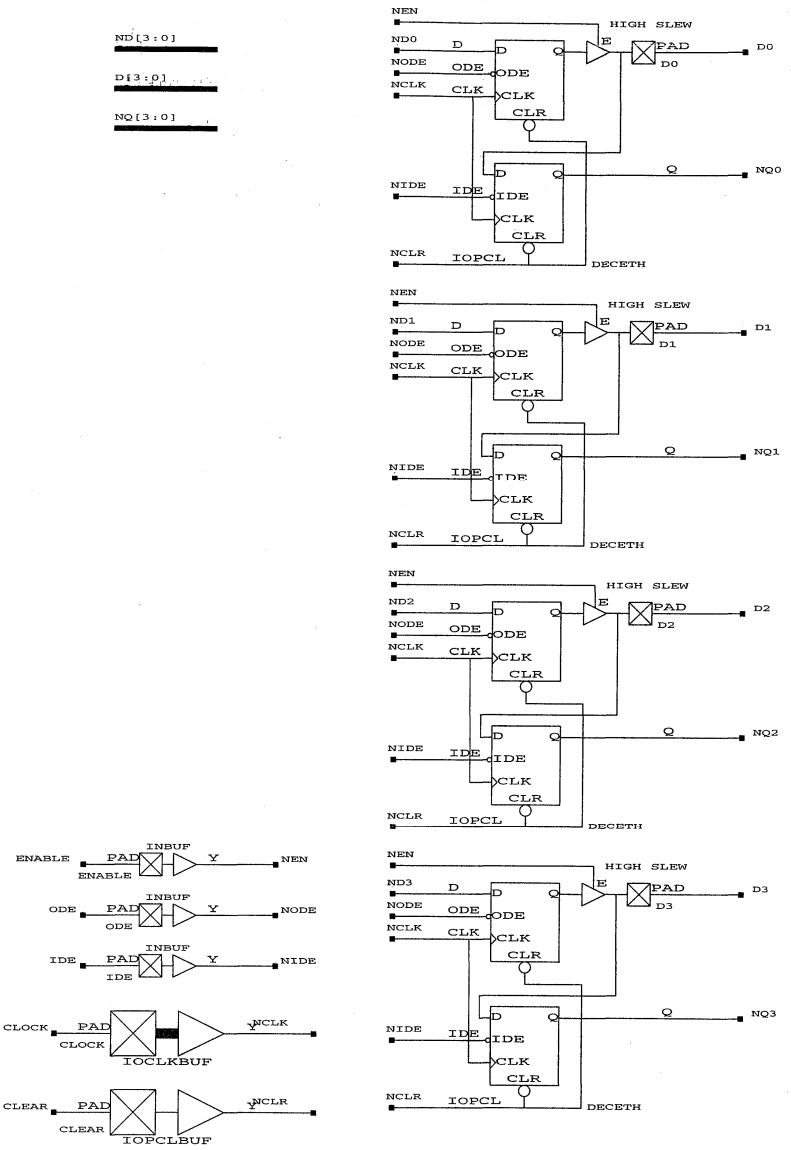
Macro Diagram—DECETH



Timing Analysis

A1425A-2	Worst Case Commercial Delays
Clock to Out	3.5 ns
IO Reg set-up	0.8 ns
Clock to Output (Pad to Pad)	7.5 ns

Application Diagram—Bi-Directional Registers



Conclusions

ACT 3 devices allow users to implement many complex functions entirely in the IO macros at the periphery of the array. These functions provide improved clock to output and setup times as well as significantly increasing the real capacity of the device.

System-Level Timing Considerations of ACT 3 I/O Macros

The purpose of this application note is to describe the strengths and limitations of the ACT 3 I/O and how to use these resources effectively in system applications.

The ACT 3 I/O module was designed to provide a powerful interface for system-level applications. There are two main factors that limit performance when interfacing to other devices. The first factor is the output delay from a driving device, which determines the required input setup/hold times for the receiving device. The second factor is the setup/hold requirement for receiving devices, which establishes the maximum and minimum output delays allowed for the driving device. Figure 1 illustrates the system-level interface with the constraints that would be placed on an FPGA solution.

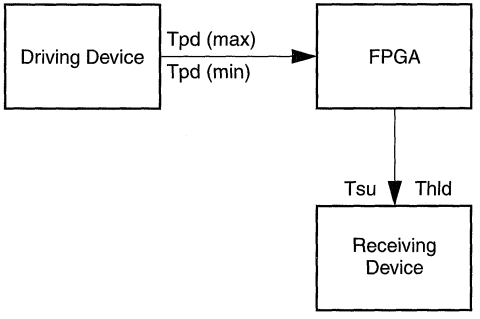


Figure 1 • System-Level Interface

Most devices specify outputs with respect to clock edges and define the minimum and maximum delays. The minimum delay is often specified to be near 0 ns, which requires any receiving device clocking the signal to have an external hold time near 0 ns as well. The maximum delay often approaches the clock period of the device, especially in high-performance applications. For this case, a setup time near 0 ns is important to meet timing requirements and not force the designer to lengthen the clock period.

Receiving devices have defined setup requirements on inputs that are typically based on clock edges. The minimum period (or maximum frequency) can potentially be limited by external interfaces if the delay from the driving device plus the required setup of the receiving device is longer than the minimum internal period of the two devices. This situation occurs typically in high-speed applications in which the period is very short (< 20 ns). To minimize performance impacts, the driving device should have clock-to-off-chip delays as short as possible.

The ACT 3 I/O module is designed to meet the needs for high-performance applications described in the previous section. The actual ACT 3 I/O characteristics are defined in Table 1. The external setup and hold times of 3.0 and 0.0 ns, respectively, should meet the needs of most systems. The external pin-pin delays of 12.0 ns or less make ACT 3 an acceptable solution for high-speed applications.

Table 1 • I/O Specification for ACT 3 Devices Under Worst-Case Commercial Conditions and Output Loading of 35 pF

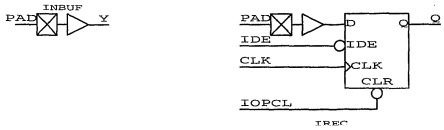
Device	1415–1440	1460–14100
External Setup	3.0 ns	3.0 ns
External Hold	0.0 ns	0.0 ns
External Clk-Q	10.0 ns	12.0 ns

To achieve these numbers, the flexibility of the I/O module is limited. First of all, both the input and output registers are active only on the rising edge and can be driven only by the IOCLK pin. The separate pin is used to reduce loading on the I/O clock network. Limiting the clock to only rising edge active prevents an additional level of logic and thus more delay.

The ACT 3 I/O module can be configured in any of the ways shown in Figure 2, giving a total of nine possible basic combinations. Only options using a flip-flop will be considered here.

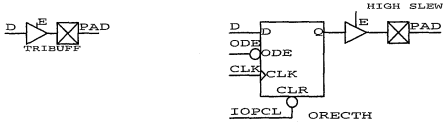
The IOCLK pin drives a fast internal clock net on ACT 3 devices, and problems can be encountered when the fast I/O logic interfaces to internal logic are controlled by slower clocks. To alleviate this problem, the ACT 3 devices also contain an HCLK, which is hard-wired to internal registers and is nearly as fast as the IOCLK. Two other routed clocks (RCLK) exist and can be accessed either externally or internally; however, these clocks are significantly slower than the IOCLK and HCLK. Problems with setup and hold can be caused in sequential logic when using dual clocks that have significant skew between them. The designer must be aware of these potential problems and how they can impact a

a) Input



b) Output

c) Tristate Output



d) Bidirectional Output

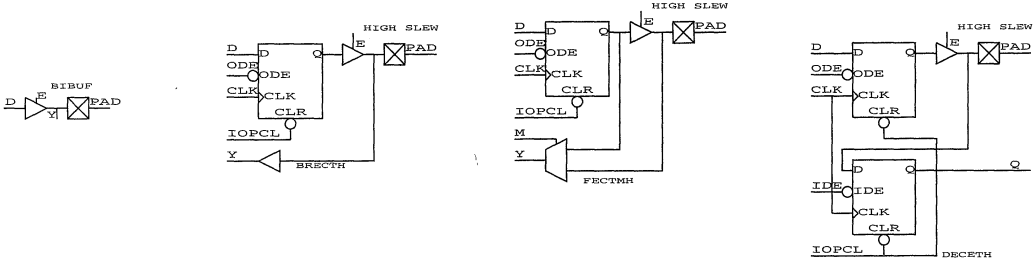


Figure 2 • ACT 3 I/O Module Configurations

design. Hold-time violations can be encountered when the faster I/O register logic is feeding slower internal register logic. The chance of setup time violations increases when an internal register driven by a slower internal clock feeds a faster I/O register.

The input register in the ACT 3 I/O module provides excellent setup and hold characteristics for interfacing to other devices. The clock that samples the data must be driven by the IOCLK pin. Sampling of the data is controlled by the IDE input. If data must be sampled every cycle, then the IDE pin should be tied to ground (active). If the input needs to be sampled only on certain cycles and then retain the present value, internal enable logic can be designed to control the IDE pin. The circuitry controlling the enable pin can be generated by any clock; however, to maximize the available cycle time, the controlling logic should be generated from HCLK, as shown in Figure 3.

If data must be subsequently sampled by additional internal logic (see Figure 4), then try to use HCLK to minimize the chance of hold-time problems. If an RCLK is substituted for the HCLK, then the chance of violating hold times is very high and must be checked either with a simulator or with the Actel Timer tool. Figure 5 details the potential problem.

Data in the output register is also sampled on the rising edge by the IOCLK pin. This path provides excellent off-chip CLK-Q delay times. Again, the output register can be controlled with the IDE pin either to sample or to hold data. Again, use the HCLK to maximize cycle time for IDE and DATA control, as shown earlier in Figure 3. (Using RCLK will cause an addition loss of about 5.0 ns of available cycle time.)

Figure 6 illustrates a method to maximize the external device performance, which runs at a subrate of the IOCLK input frequency. This example uses a T flip-flop to create a divide

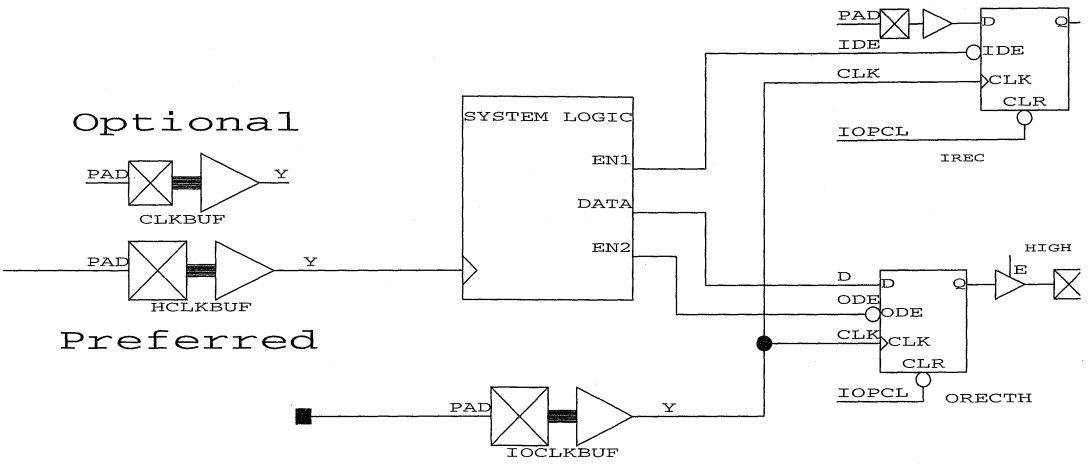


Figure 3 • Controlling Logic Generated from HCLK

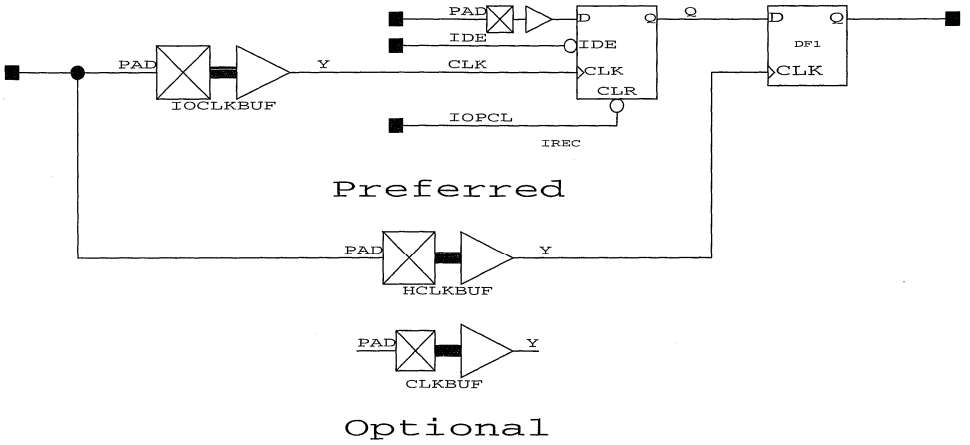


Figure 4 • Data Sampled by Additional Internal Logic

Data arrival (load of 3) = 7.0 + 2.1 = 9.1 ns
Required setup into DF1 = 1.0 ns
Required clock arrival Tpd – Tsu = 9.1 – 1.0 = 8.1 ns

Actual HCLK arrival (max) = 5.5 ns (no violation)
Actual RCLK arrival (max) = 9.0 ns (hold violation)

Figure 5 • Required Clock Arrival Time (Timings for the 1460 Under Worst-Case Commercial Conditions)

by two circuits causing data to be passed every other cycle. The internal circuitry providing data to the output register can then run at half the IOCLK frequency while still getting excellent off-chip times.

The bidirectional ACT 3 module can be configured to have both an input and an output register or to have an output-only register. For the bidirectional output with both, the registers should be controlled individually, as described in the previous sections.

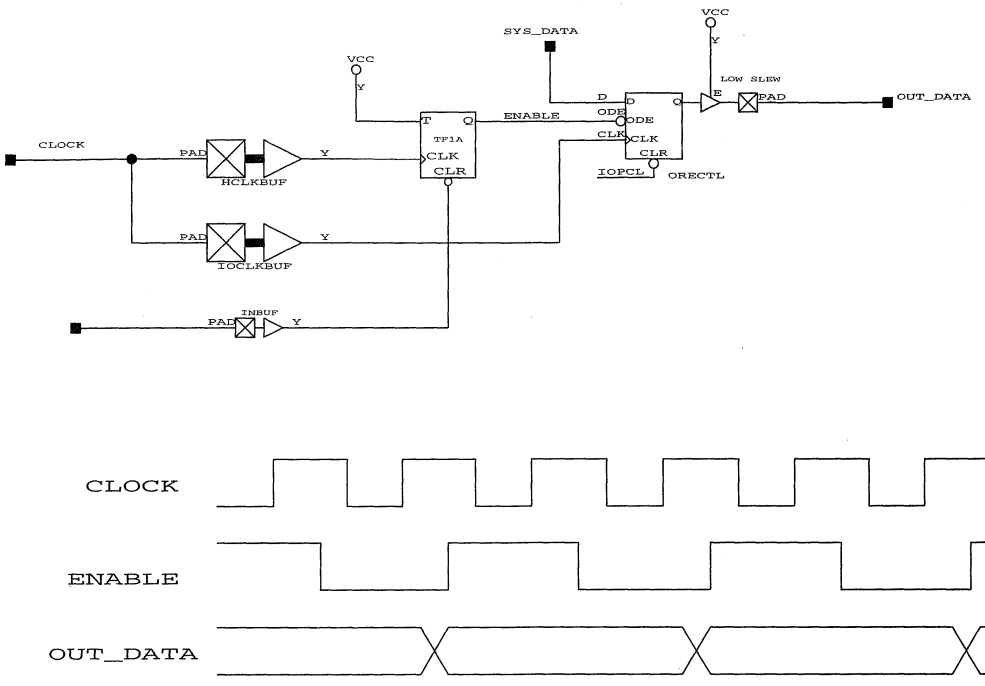


Figure 6 • Maximum External Device Performance

For the case of the bidirectional output with only an output register, two cases exist for feeding data back into the internal array. A "fast path" can pass the data from the output of the register back to the internal array without having to be driven onto the pad. This function can be dynamically controlled using the FECTMH macro, shown in Figure 7. When the M pin is low, the register output is passed. When high, the pad data is passed. When using the BRECTH macro, data must be driven onto the pad before the information is fed back into the internal array. The fast internal feedback of the FECTMH macro improves the delay by about 9.0 ns over the external path.

The I/O registers in unbonded pins currently cannot be accessed; however, if a device has additional unused pins, then the registers in these I/Os can be used as storage for internal logic. Figure 8 shows how the I/O can be configured to generate 2-bit SISO shift register and a 1-bit storage register. The first case requires the data to be driven to the off-chip pad; however, the second case can use the internal path and does not need to be driven off-chip.

The ACT 3 I/Os provide a powerful system-level interface for low- and high-speed applications. The ACT 3 I/Os have good

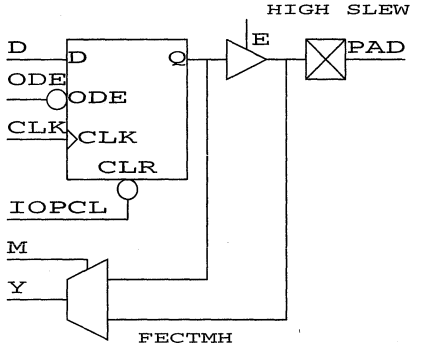


Figure 7 • FECTMH Macro

input setup and hold requirements and good off-chip clock to Q delays, making them ideal for a large number of applications. Some care must be used when interfacing the I/O registers to internal logic; however, the HCLK is provided to minimize any timing problems when using the ACT 3 I/O registers.

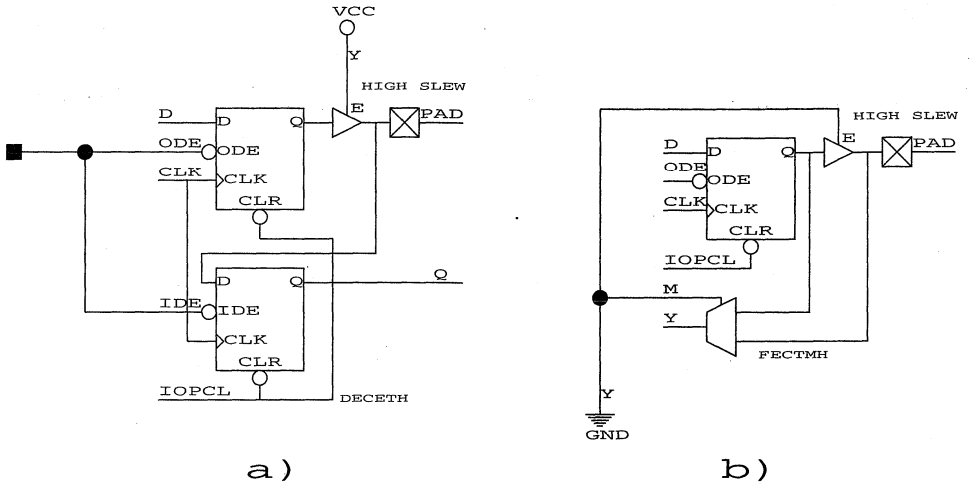


Figure 8 • I/O Configurations to Generate a 2-Bit SISO Shift Register and a 1-Bit Storage Register

Board Level Considerations for Actel FPGAs

Introduction

Simulating and debugging individual components is the first step towards verifying a system design. In some cases, the devices no longer behave as expected after they are integrated. Many factors, such as power, airflow, and transmission lines can introduce undesirable results in a system and ultimately impair system performance. This application note will explain how many of these factors should be treated when integrating ACT™ family field programmable gate arrays (FPGAs) in board-level designs.

Power Up

Actel FPGAs are nonvolatile and therefore require no external configuration circuitry on power up. However, at power up it does take a finite amount of time for the device to become stable and operate normally. For a V_{CC} slew rate of ~30 ns/V, it takes approximately 250 µs for the device to become fully operational. Power up time varies with temperature, where cold is worst case. At power up, the state of all flip-flops is undefined. Refer to *A Power On Reset Circuit for Actel Devices* for more information.

Operating Environment

ACT family FPGAs must not be operated outside of the recommended operating conditions as described in the Absolute Maximum Ratings section of the ACT data sheets. Exposure to maximum rated conditions for prolonged periods may result in irreparable damage to the device.

As can be seen in the timing derating section of the ACT datasheets, variations in voltage, temperature, and process will affect device performance. You must take care to design systems so that the effects of these variables, from best to worst case, will not cause timing problems during interchip communications.

Thermal Considerations

An often overlooked test case examines ambient temperature effects on FPGA performance. The Action Logic® System reports timing as derated by thermal junction temperature. This is the temperature found at the junction of the die and the package casing. Additional derating must be applied for ambient to junction temperature effects (θja). The value for θja will vary from package to package. The units for θja are °C/W. This implies that the power dissipation must first be calculated to determine the value for θja. (Please refer to the Power Dissipation section of the ACT datasheets.) Once θja is calculated, this temperature should be added at the junction to case temperature (θjc). This is the temperature value to be used for the temperature derating portion of the timing analysis. Because θja is a function of ambient temperature, the use of a fan can decrease the θja value. This can be seen by observing the difference between the θja value in still air and at 300 ft./min., as shown in the Package Thermal Characteristic section of the datasheets.

Ground Bounce and Switching Characteristics

Actel defines simultaneously switching outputs (SSO) as any outputs that transition in phase within a 10 ns window. The resultant ground bounce produced is a function of the number of outputs switching simultaneously and the capacitive loading of the outputs.

Table 1 shows the recommended SSO limit for ACT 1 and ACT 2 FPGAs. To measure worst-case ground bounce, the I/Os are placed adjacently and are driving a 50 pF load. The observed ground bounce is less than 1 V, with a pulse width of less than 2 ns.

Exceeding the recommended limits results in a larger ground bounce. However, the width of the ground bounce pulse prevents it from being recognized by most discrete digital receivers. Refer to the specifications of the external receivers for verification.[1]

Table 1 • Recommended SSO Limits for Actel FPGA

Device	Package	Maximum Recommended SSOs (50 pF load)
A1010A/A1020A	44 PLCC	16
A1010A/A1020A	68 PLCC	24
A1020A	84 PLCC	32
A1010A/A1020A	84 PGA	32
A1010A/A1020A	100 PQFP	32
A1280	PG176, PQ160	64
A1240	PG132, PQ144	48
A1240/A1225	84 PLCC	32
A1225	100 PGA, PQFP	32

Should you have the need to switch more signals than recommended, the SSOs should be distributed evenly around device. This will reduce the amount of mutual inductance produced between adjacent I/Os and thus reduce the ground bounce.

Buffers may also be inserted in the path before the output buffer. This will reduce the possibility of adjacent signals switching within 10 ns of each other. The ALS Timer should be used to verify the delays.

Power and Ground Pins

Actel FPGAs are designed with ample power and ground pins on the device resulting in minimal ground bounce characteristics during I/O switching. A ground pin is provided for approximately every eight I/Os (please refer to package drawings found in the *Actel FPGA Data Book and Design Guide* for further detail). Unused I/Os cannot be programmed to act as ground pins.

Table 2 describes the function of each of the power and ground pins found on the devices.

Table 2 • Power and Ground Pin Functions

V_{CC}	Device operating power.
V_{PP}	Device programming voltage. Should be tied to V_{CC} during normal operation.
V_{SV}	Super voltage supply. $V_{SV} = V_{PP} + 3$. V_{SV} provides a precise lower limit on the voltage that is applied to a fuse during programming. During normal operation $V_{SV} = V_{CC}$.
V_{KS}	V_{KS} provides voltage supply for keepers (keepers maintain floating tracks at $V_{PP}/2$ during programming and prevent them from leaking down to GND). During programming, $V_{KS} = V_{PP}/2$. During normal operation, V_{KS} = GND.
GND	Supplies GND to the array (all channels).

Slew Rate

ACT 1 and ACT 2 outputs operate at high slew rate only. Figures 1, 2, and 3 illustrate the slew rate characteristics for high slew rate outputs, both loaded and unloaded, for an A1280 PG176 device.

Device I/O

The I/Os on each Actel FPGA are nonrestrictive. Any pin shown as an I/O in the pin description list can be assigned to be either input, output, bidirectional, or tri-state. ACT 2

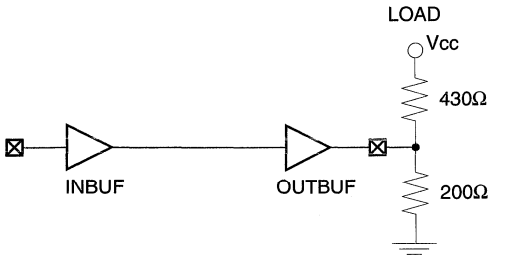


Figure 1 • The Test Circuit

FPGAs add latch capabilities to each of these options. Actel FPGAs do not incorporate any type of internal pullup on the I/Os. Inputs must not be left floating, since this may cause irreparable damage to the device. Any unused I/Os will be configured as active low outputs by the Action Logic System. The sink and source capabilities of individual outputs for ACT 1 and ACT 2 FPGAs have been extrapolated from V-I curves and are shown in Table 3. The V-I curves for ACT 1 and ACT 2 are shown in Figures 4 through 9.

Table 3. Worst-Case Sink and Source Current for Actel FPGAs

	ACT 1	ACT 2
TTL	8 mA	10 mA
CMOS	4 mA	6 mA

I/Os may be tied together externally to increase drive capability. The outputs must switch within 4 ns of each other, thus they should be placed as close to each other as possible. (Please refer to device die bonding diagrams for pad locations.) The switching times can be verified with the ALS Timer.

Decoupling Capacitors

Actel recommends the use of decoupling capacitors with all FPGA devices. We suggest a minimum of one capacitor per side, located next to power and ground. A 0.1 µF monolithic ceramic type is sufficient.

References

1. David Shear, "EDN's Advanced CMOS Logic Ground-Bounce Tests," *EDN*, March 2, 1989, p. 88.

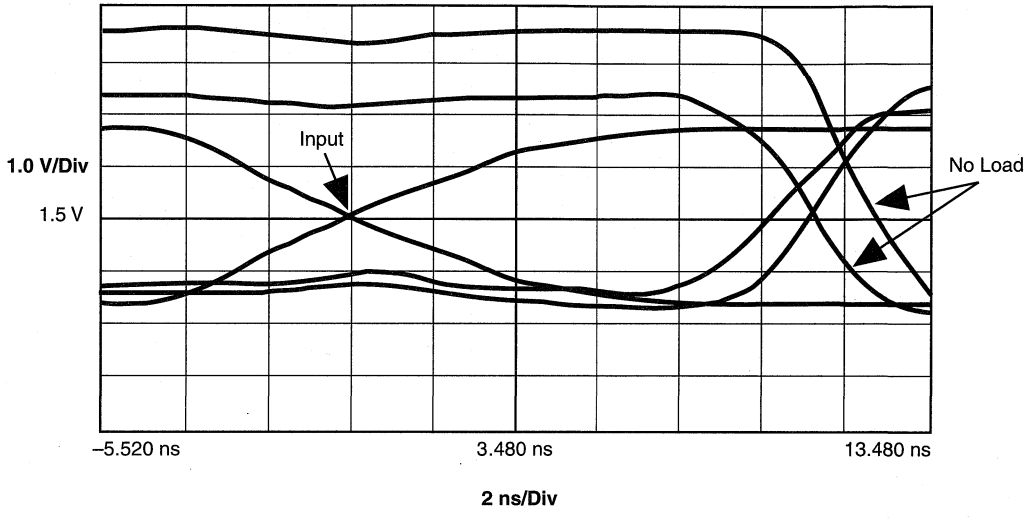
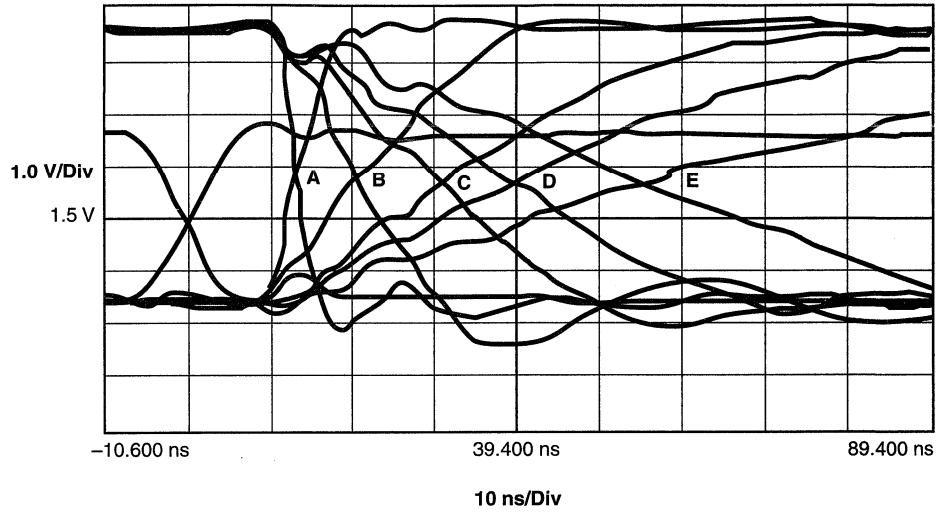
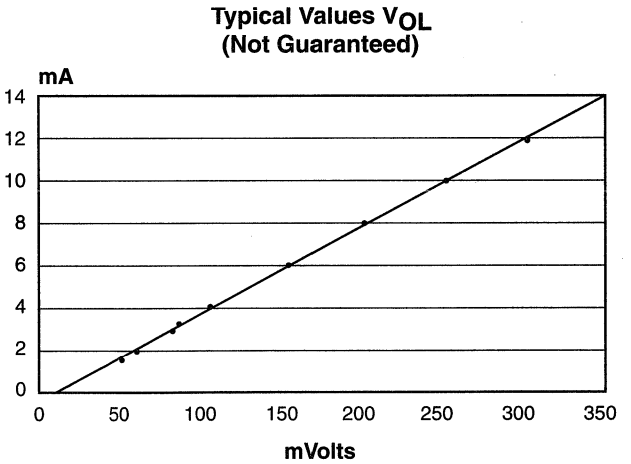


Figure 2 • No Load Versus Load—Fast Slew



A = "No Capacitive Load" (~25 pF jig + 9 pF scope)
B = 82 pF
C = 200 pF
D = 300 pF
E = 560 pF

Figure 3 • *Capacitive Loading Versus Slew*



V_{CC} = 4.75 V, 25°C, Typical Process

Figure 4 • *A1020A Typical Sink Current*

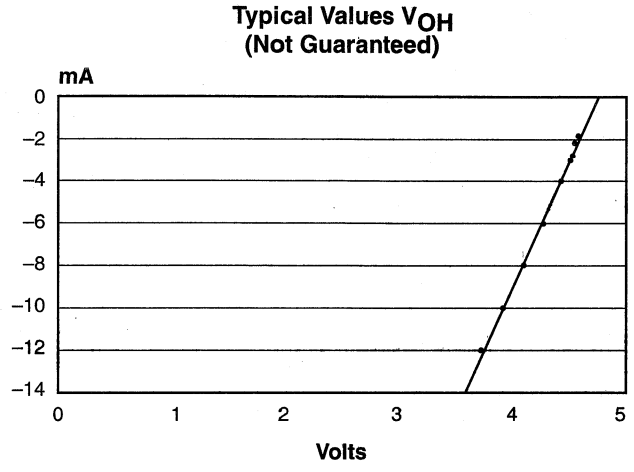


Figure 5 • A1020A Typical Source Current

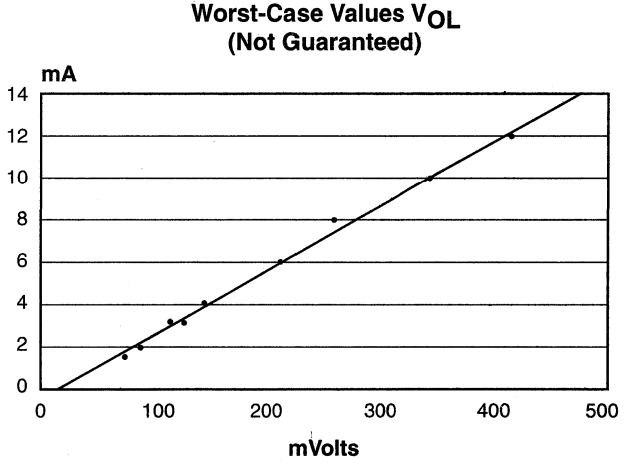


Figure 6 • A1020A Worst-Case Sink Current

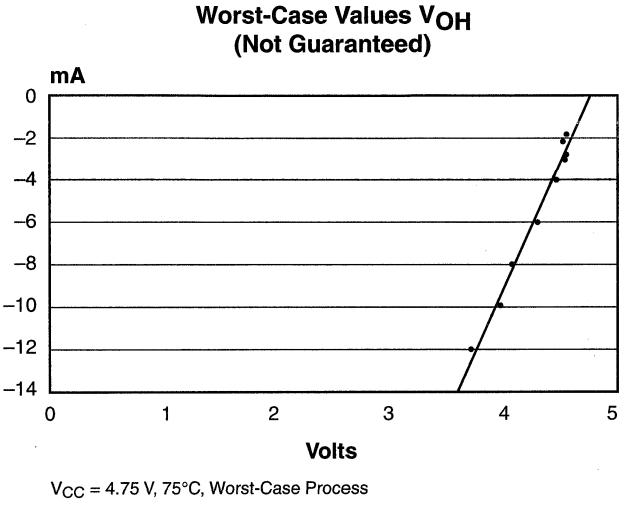


Figure 7 • A1020A Worst-Case Source Current

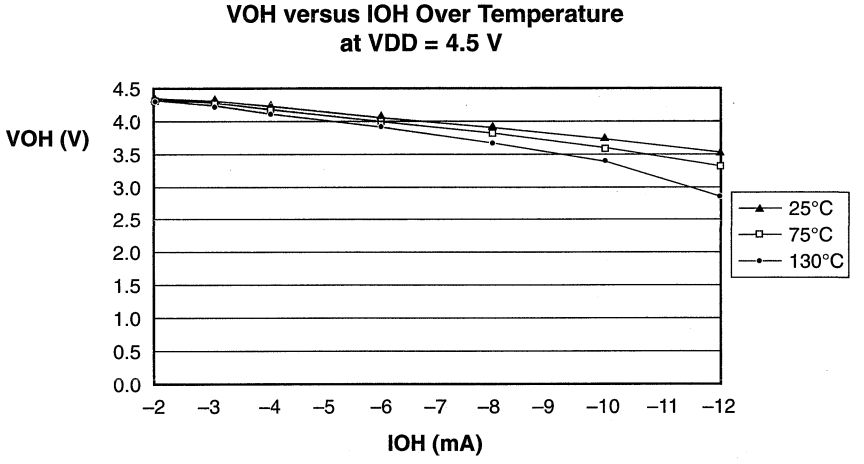


Figure 8 • A1280 Typical Source Current

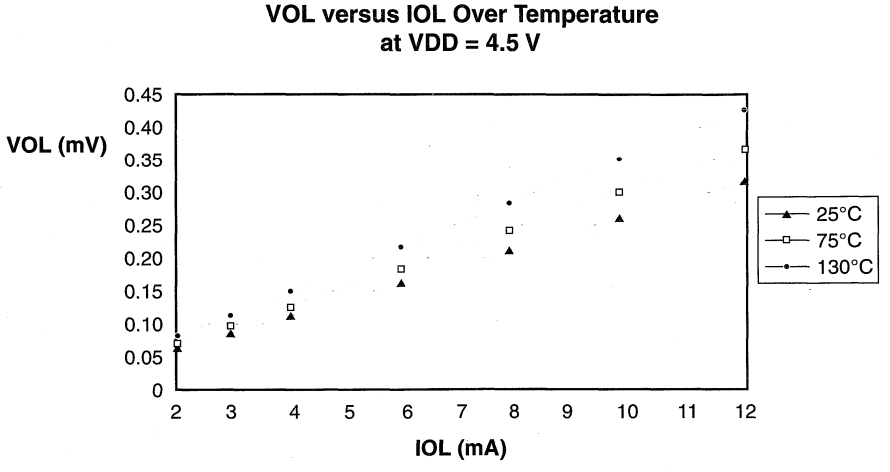


Figure 9 • A1280 Typical Sink Current

Fast On and Off Chip Delays with ACT 2 and 1200XL I/O Latches

Introduction

Using ACT™ 2 and 1200XL devices, latched I/O buffer macros can improve clock input-to-registered-output performance. Latched I/O buffer macros also increase the speed of latching signals into the FPGA. Flip-flops created from these I/O latch macros improve performance by up to 22 percent compared with traditional approaches. This application note compares the use of traditional approaches with the use of I/O latch macros in ACT 2 and 1200XL designs.

Master/Slave Flip-Flops

As shown in Figure 1, two level-sensitive latches can be combined to create a positive edge-sensitive flip-flop.

Where:

1. $T_{CO} = t_{CO2}$
2. $T_{SU} = t_{CKL} - t_{CO1} - t_{RD1} > t_{SU2}$
3. t_{SU2} = minimum setup time for slave

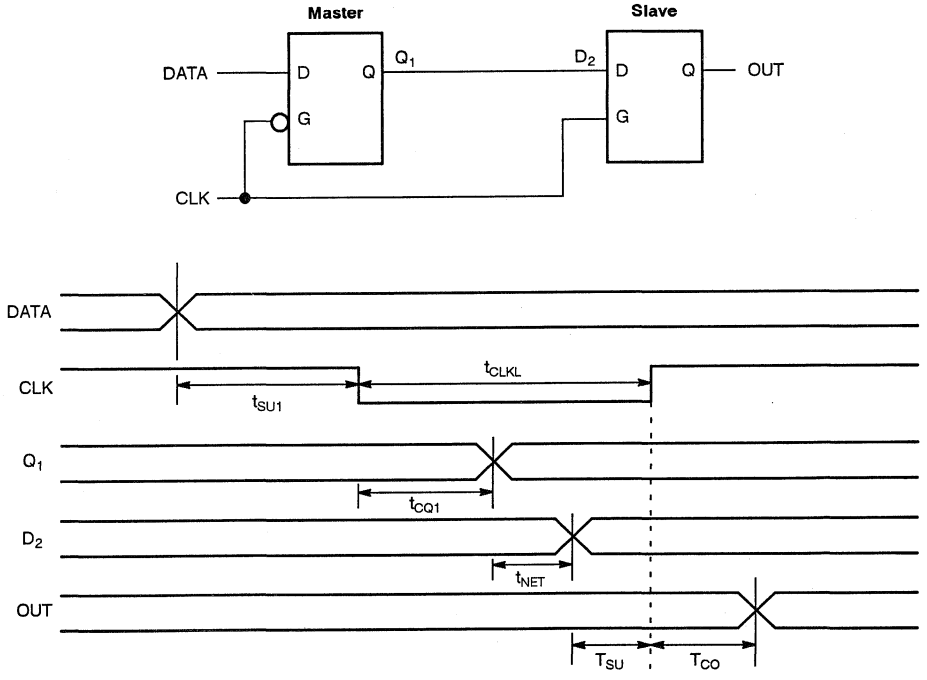
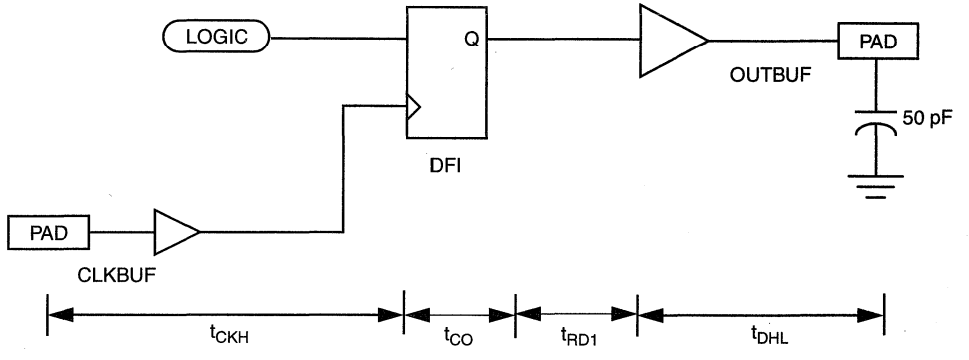


Figure 1 • Master/Slave Flip-Flop

The clock- to-output delay of the resulting flip-flop is determined by the clock-to-output delay of the slave latch. The clock period low time (t_{CKL}) must be greater than the clock-to-output delay of the master latch (t_{CO1}) plus the net delay from the master latch output (t_{RD1}) plus the setup time of the slave latch (t_{SU2}).

Constructing Registered Outputs

You can construct a registered output by combining a flip-flop macro with an output buffer as depicted in Figure 2. The clock-to-out delay for an A1225A-2 under worst-case commercial conditions is 25.2 ns. For the A1225XL-1 under worst-case commercial conditions, clock-to-out delay is 14.3 ns.



A1225A-2 $T_{CO} = t_{CKL} + t_{CO} + t_{RD1} + t_{DHL} = 10.2 + 3.8 + 1.1 + 10.1$ ns $= 25.2$ ns

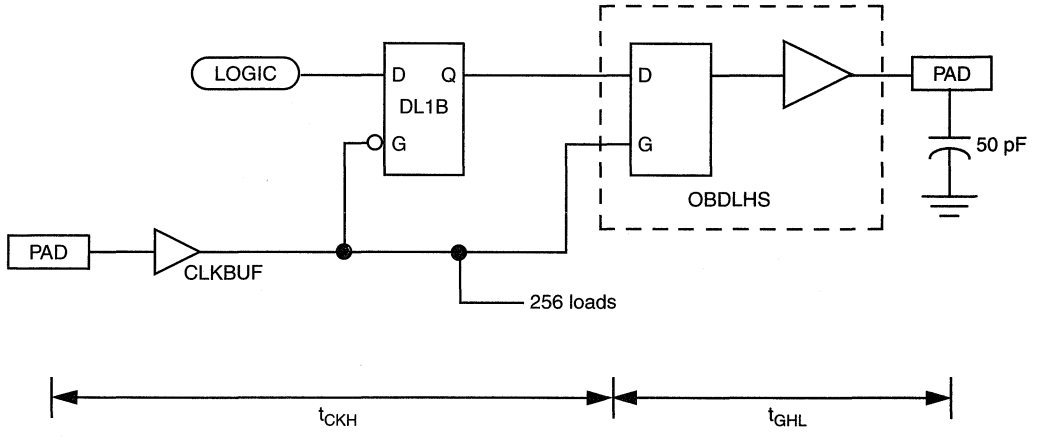
A1225XL-1 $T_{CO} = t_{CKL} + t_{CO} + t_{RD1} + t_{DHL} = 5.8 + 3.0 + 0.9 + 4.6$ ns $= 14.3$ ns

Figure 2 • Conventional Registered Output

I/O Latch Flip-Flops

You can also constuct a registered output as a master/slave flip-flop using a logic module latch (DL1B) and a latched output buffer (OBDLHS) as shown in Figure 3.

In this case, the clock-to-out delay for an A1225A-2 device under commercial worst-case conditions is 23.0 ns. In this case, the loading on the global clock network is assumed to be 256 loads. For similar conditions, the A1225XL-1 has a clock-to-out delay of only 11.3 ns.



A1225A-2 $T_{CO} = t_{CKH} + t_{GHL} = 11.8 + 11.2$ ns $= 23.0$ ns

A1225XL-1 $T_{CO} = t_{CKH} + t_{GHL} = 6.5 + 4.8$ ns $= 11.3$ ns

Figure 3 • Master/Slave Registered Output

Registered inputs can also be constructed using a logic module latch (DL1) with a latched input buffer macro (IBDL) as shown in Figure 4. This is equivalent to the I/O macro, IR. Data can be latched into the FPGA most efficiently in this configuration. Because of the unique architecture of the I/O buffer latches in ACT 2, the input latch has zero external setup time. This means that the data may change just before the rising edge of the clock signal. Data must be held at the input of the latch for the time t_{HEXT}.

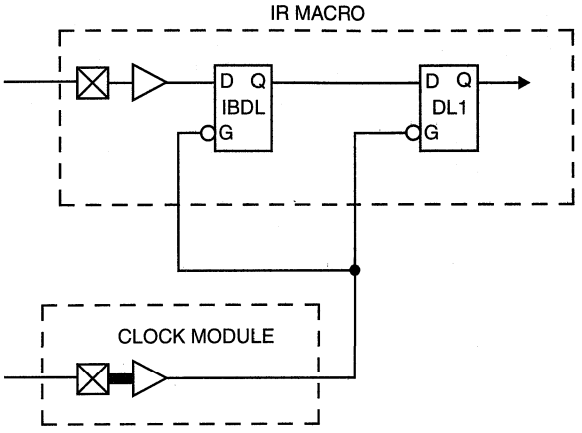
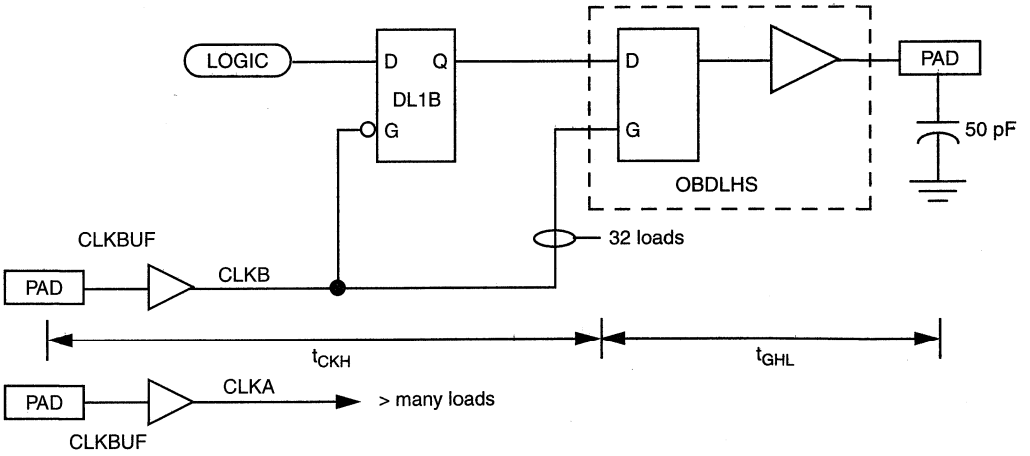


Figure 4 • Master/Slave Registered Input

Dual Clock Approach

Faster clock-to-out can be achieved by utilizing the second global clock network as an I/O clock. In this configuration, shown in Figure 5, CLKA drives the synchronous circuitry on the device and CLKB drives the I/O master/slave latches. Both clock networks are operating at the same frequency and must be connected together outside of the devices. In this case, a 21.4 ns clock-to-out delay can be achieved for the A1225A-2 and 10.6 ns for the A1225XL-1.

Alternately, an INBUF can be used to drive the I/O latch, if fanout on the INBUF is kept less than four to achieve similar performance.



A1225A-2 $T_{CO} = t_{CKH} + t_{GHL} = 10.2 + 11.2$ ns $= 21.4$ ns

A1225XL-1 $T_{CO} = t_{CKH} + t_{GHL} = 5.8 + 4.8$ ns $= 10.6$ ns

Figure 5 • Dual Clock Master/Slave Registered Output

Implementation Rules

Actel strongly suggests following these rules when constructing flip-flops with the I/O latches:

1. Do not put combinatorial macros in the data path between master and slave latches. Added delay may prevent the flip-flop from operating properly. For inputs, combinatorial macros are allowed in the data path between the master and slave latches *only* if the logic module latch and the combinatorial latch are both combinable hard macros.
2. For outputs, do not connect the master latch output to any loads except the I/O latch D input.
3. Use a latch made from sequential logic modules for the master stage for outputs on the slave stage for inputs. Sequential module latches have better timing characteristics. They also allow combining to take place, which can improve the performance of the data being registered. The transparent-low sequential module latches are DL1B and DL1C.
4. Use net criticality to ensure that the net delay does not violate the setup requirements of the slave latch (as defined in Figure 1). Verify the timing conditions after place and route is complete. Note that an asymmetrical duty cycle on the clock signal (less than 50% low time) will provide more tolerance on the allowable net delay between latches.
5. Design to combine. The Action Logic® System (ALS) will automatically combine combinatorial logic into the D input of the DL1B latch if the combiner rules are met.

The Hidden Cost of Reprogrammability

On the surface, reprogrammability may appear to be a no-lose feature in a field programmable fate array (FPGA) design environment. However, this issue is more complex in the real engineering world. Unless reprogrammability is required for the system-level design, its purported benefits can be substantially outweighed by hidden costs and pitfalls. This application note outlines the areas an engineer must examine before selecting a programmable digital logic technology.

Background

Programmable logic device (PLD) and enhanced programmable logic device (EPLD) designers have evolved a trial and error design philosophy—they enter a design, compile and program it, then "see if it works." If the design doesn't work, they are forced to analyze output states based on input states because internal signals are not available for analysis. Since the behavior of the internal signals cannot be observed, the solutions to design problems are difficult to uncover. Each possible solution must be entered, compiled, and programmed. Then, the engineer must "see if it works" again. For many designs, this cycle may be repeated several times.

For asynchronous designs, reprogramming is even more difficult and time consuming, especially because changes may affect the functionality of the overall design as well as portions that previously worked! An alternative to this approach is to utilize incremental logic design: implementing small functional blocks, debugging them, and then combining all the blocks. This approach also consumes many iterations and still requires much educated guessing from the engineer.

With a reprogrammable PLD/EPLD, an argument can be made that this trial and error approach is acceptable because one device can be utilized over and over again. However, the hidden cost is the time spent debugging, modifying, compiling, and repeating the process. Obviously, for a simple 22V10 design, little time is involved. But as the design increases in gate count, speed, and circuit complexity, the design time goes up exponentially. Specifically, a 2000 or 4000 gate design might have many errors that take weeks to find and debug with a trial and error design approach. Reprogrammability is worth very little if the errors in the design cannot be discovered easily. This same phenomenon was experienced by the early users of microprocessors and microcoded bit slice machines. This predicament drove the development of in-circuit emulators (ICE) by the microprocessor and development tool suppliers such as Intel and Motorola. Few engineers today design microprocessor-based systems without utilizing an ICE.

The Real Cost of a Design

The cost of design time has always affected the engineering budget, but in today's competitive marketplace, design time even more dramatically affects the product's success and profitability. Successful companies such as 3Com and Hewlett-Packard have determined that each week of delay translates into a loss of $30,000. Even worse, a delayed product might miss the market window entirely. Together these costs force engineering departments to consider the total cost of the design, including time and device costs, in deciding a design process.

Cost-Effective Design Alternative

To afford the engineer a more cost-effective total solution, Actel has developed an architecture and tools that enable an engineer to utilize not only the most cost-effective programmable devices but also the most cost-effective design methodology. At first glance, Actel does not appear to offer the most cost-effective design solution because the Actel devices are one-time programmable. In fact, given the design methodology described above, Actel's devices might appear to be the most expensive, potentially requiring an engineer to program dozens of devices before a system design is complete.

However, Actel's architecture enables engineers to approach logic design in an entirely different manner, saving days if not weeks of design time as well as reducing the number of devices needed to achieve a fully functional design. Historically, addressing the logic design problem meant using simulation. While simulation enables the observation of an unlimited number of internal functions and all timing aspects of a device, it usually requires that a new tool and design methodology be learned. To avoid the investment in the time and money learning new methods, Actel offers a tool that achieves practically all the advantages of simulation without a lengthy, expensive learning period.

Actionprobe® Diagnostic Tools

Actel's architecture offers a unique feature that supports the probing of any internal node in an Actel device running in system. Conceptually, Actel's Actionprobes work just like a microprocessor in-circuit emulator, enabling any internal points to be examined "on-the-fly" in real time. Architecturally, the Actionprobes are controlled and observed by four special probe I/O pins on each device. These pins function as user-defined I/Os during normal device operation. Note that probes can be utilized in conjunction with simulation to catch undefined or ill-defined environmental conditions because the chip is now running in system at real time.

When the Actionprobes are enabled, two probe pins are used to select two internal nodes for observation. These signals are sent to the two Actionprobe output pins. There they can be viewed with an oscilloscope or logic analyzer. The Actionprobe signals may be changed dynamically in circuit, in real time.

Because the Actionprobe network is overlaid across the entire chip, there is no incremental load applied to the point being tested. Thus, the use of the Actionprobes does not change the dynamic characteristics of the circuit. Also, since the choice of probe signals can be changed at any time, any piece of the design can be debugged in real time without any additional compiling or programming. A chip's inputs can be traced through the entire device up to the output pins. If desired, after the design is tested and released to manufacturing, the probes can be used in circuit for the in-system testing of pin continuity as part of the manufacturing test process.

The Actionprobes allow an improved design methodology that is quicker, more controllable, and substantially less frustrating than the trial and error process. Achieving a working design requires very few iterations, saving time and money. In practice, anywhere from a few days to weeks can be saved depending on the design complexity and problem/solution iteration cycle. Additionally, very few devices are actually used to achieve a working design. A survey of Actel's users have shown that on average fewer than five devices are needed to achieve a working design. Designers attain working designs sooner, with fewer iterations, and at the cost of less than five devices.

The total cost of designing with Actel FPGAs is substantially more cost-effective than with reprogrammable approaches. The Actel FPGA approach typically saves one week of design time per chip. The cost savings (one week of engineering design time minus the cost of a handful of devices) may be conservatively estimated to be several thousand dollars per design. The real cost savings can be much greater given considerations such as the complexity of the design. The Actel solution will still be more cost-effective in the long run even if the number of devices used in development becomes large. In production, Actel devices are substantially more cost-effective than reprogrammable devices.

The Real Cost of Reprogrammability

To understand some of the less obvious costs associated with reprogrammable parts, consider the fundamental technology used to program them. The programming elements in reprogrammable devices are much larger than the Actel antifuse. The antifuse is less than 1.0 micron in diameter, it is a passive device, and it lies within the metal routing tracks of the device. No overhead is required other than that for the programming and testing circuitry, which is roughly the same amount for all programmable devices. Other programming elements require substantially more room because they are active devices. The larger programming elements increase the area overhead required for sufficient routing resources and ultimately result in a larger die size, a less flexible architecture, or both. Because of the antifuse advantage, the Actel die sizes are substantially smaller and as a result offer a more cost-effective production device.

While the unit cost tradeoffs are obvious, the architecture tradeoffs are not. The abundant, virtually free routing resources afforded by the antifuse switching element enable logic changes to be made with little or no impact on the performance or utilization of the device. No other programmable element-based architecture offers this flexibility. The smaller antifuse not only makes abundant internal routing resources available, but it also enables abundant routing resources between I/O and the internal device logic. This enables design changes to be made with little or no impact on the I/O locations. I/O locations can be finalized and the PCB artwork sent out for manufacturing without worrying about design changes affecting the pinout of the chip, which requires the PCB to be revised prior to manufacturing. Obviously, turning a PCB is not only very expensive, but it also requires a schedule delay.

The impact of routing resources on I/O locations is directly related to the design's gate count, complexity, and speed. The higher the performance, complexity, or gate count as a percentage of the device's available gates, the worse the problem can be. Actel's antifuse-based devices are the most flexible programmable devices in this area. The designer is virtually guaranteed that I/O locations won't change because of logic changes to the design. No other programmable technology can make this claim.

Conclusion

Reprogrammable architectures appear at first glance to offer the most cost-effective design solution, but in reality they are the most expensive. Not only are reprogrammable architectures more expensive on a per unit basis, they don't offer a probing capability that enables users to find and debug errors. Furthermore, they don't offer enough routing resources to make design changes without I/O changes that could require expensive PCB turns or extra nonbudgeted design time. When selecting a programmable architecture, design engineers should weigh all these factors and their associated costs, as well as the opportunity cost of being late to market. Given these, the logical choice is Actel FPGAs for the most cost-effective design solution.

Designing for Migration to Actel MPGAs

The Actel Mask Programmed Gate Array (MPGA) family of high-density, high-performance devices comprises mask programmed versions of the ACT 1, ACT 2, and ACT 3 Field Programmable Gate Array (FPGA) families. The MPGA family provides a low-risk conversion path from programmable gate arrays to production quantity devices. By significantly reducing the production costs of a product without technical or time-to-market risks, MPGAs prolong the life cycle of a finished design.

The MPGA products are specifically designed to make conversion from programmable devices virtually transparent to all members of a product team, from initial system designers to final production manufacturing personnel. Virtually any successful FPGA design can be successfully converted to an MPGA device without difficulty. Figure 1. shows the design process for Actel FPGA and MPGA devices. The design flow is tailored to ensure that the conversion process is as seamless as possible, but adhering to some simple guidelines will further ease the transformation. Use the following suggestions and design hints during the FPGA development process to make each design as convertible as possible.

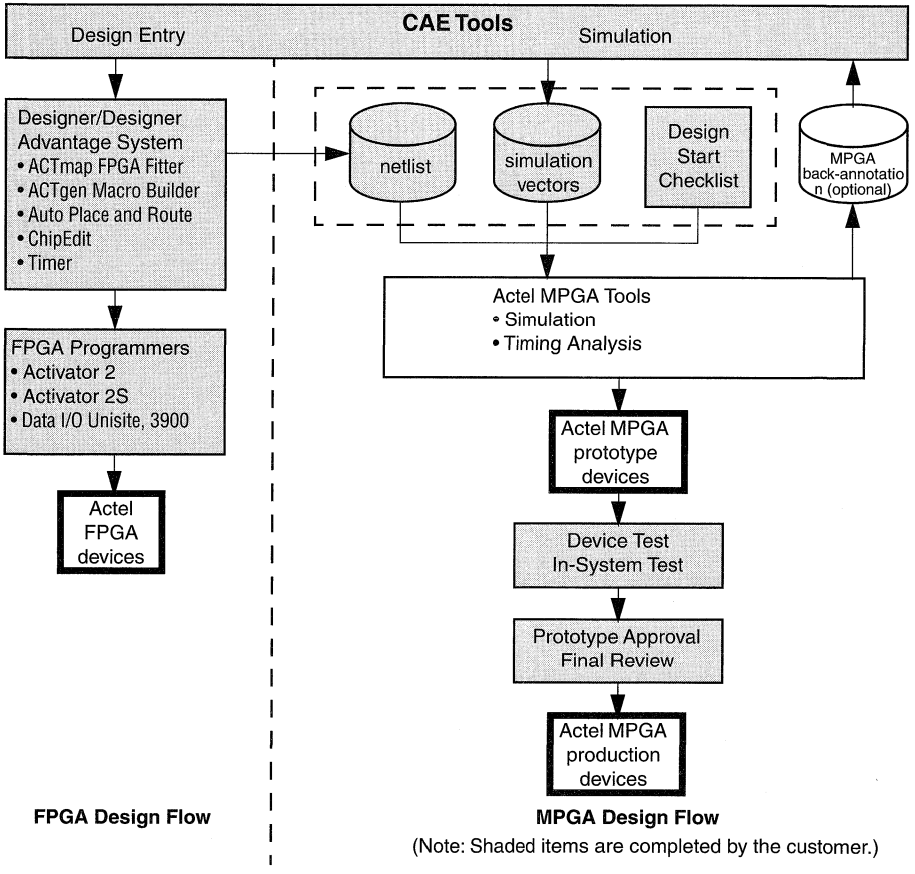


Figure 1. Actel Device Design Flow

General Design Guidelines

Fundamental principles should be applied to all digital designs regardless of the intended target devices. Actel FPGA designs and their corresponding MPGA versions are no different; reliable systems result from solid circuit designs. Design practices such as reducing unnecessary logic (for example, inverters) using DeMorgan's Law to combine logic, and balancing logic levels (and their corresponding delays) between registers will all help to produce stable circuits. More suggestions come from the application notes "Hints and Tips for Better Actel Designs I" and "Hints and Tips for Better Actel Designs II." The *Actel FPGA Data Book* includes many other application notes to assist in the design process. Refer to the Data Book chapters "Designing with Actel Devices" and "Application Examples and Design Techniques" for more information.

MPGA versus FPGA Performance

Actel MPGAs are built using a "sea-of-gate" architecture with 0.8 and 1.0 micron processes. This architecture is similar to the FPGA architecture but has more compact spacing between logic modules and tighter metal-only interconnections. It enables MPGAs to provide performance equal or superior to their FPGA counterparts for almost every design. On average, both logic module and routing speeds are significantly faster for MPGAs. This usually translates into superior system performance for the same design.

Some designs that use ACT 3 devices to satisfy very high on-chip and off-chip performance requirements may experience a small decrease in speed. The ACT 3 I/O modules are especially fast and are not implemented identically in the MPGA versions. However, in most cases, the MPGA is still fast enough as a drop-in replacement for any Actel FPGA. Check the MPGA data sheet to verify that its I/O performance satisfies your particular system requirement.

Power

Due to its relatively smaller die size and metal-only interconnections, MPGA power dissipation is usually significantly lower than its FPGA counterpart. The active power dissipation of a CMOS device is a function of its equivalent capacitance. On average, the removal of programmable elements from the device reduces its equivalent capacitance and, in turn, its active power dissipation. Upon receipt of the Design Start Checklist and associated materials, Actel calculates the MPGA active power dissipation for each design based on this formula. This calculation is immediately relayed to you so that you can update system power specifications accordingly.

Use Synchronous Design

Asynchronous designs are often the cause of many problems, especially when a change is made among target devices to implement a design. An asynchronous design that works with SSI/MSI devices may translate poorly to other technologies such as ASICs because of significant differences in timing between devices. Designs using the same technology from a single manufacturer still may fail because of process differences from die to die. Even similar devices such as Actel FPGAs and MPGAs may not automatically implement an asynchronous design without problems. Generally, most of the problems come from small timing differences that lead to functional errors and days of debugging.

Synchronous design significantly increases the reliability of a circuit. It also makes systems easier to test and debug. The odds of successfully migrating to Actel MPGAs increase directly with the elimination of asynchronous elements in the design. In particular, try to avoid the following types of potential problem circuits:

1. **glitch generators**—Avoid circuits that may create pulses or glitches whose widths depend on variations in propagation delay. For example, an asynchronously generated clock signal generated by decoding the output of a counter with combinatorial logic may work in one technology but may fail to generate a proper rising edge in another. The same decoder may create extra, unwanted clock pulses if the decoder has uneven levels of logic from input to output.
2. **gated clocks**—Do not gate clock inputs to edge triggered storage elements. Use flip-flops with enable inputs instead. Not only is this more reliable, but it also saves logic resources.
3. **asynchronously generated reset signals**—Flip-flops or latches implemented to create asynchronous one-shot signals are unreliable and untestable. Whenever the output of a storage element is fed back to its asynchronous input, the resulting pulse varies widely in behavior depending on process and environmental variations.
4. **nternal delay lines**—Strings of buffers or inverters used to create ring oscillators or internal delay lines have unpredictable delay and frequency. They are also extremely difficult to test. Use stable, external oscillators and synchronous pulse generators as alternatives.

Sometimes asynchronous elements must be included in a design. In that case, be sure to understand the effects that technology, process, and environmental variations have on each circuit. Refer to industry standard ASIC logic design manuals for more complete information about creating reliable, testable designs.

Testability

Actel FPGAs are fully testable after programming using two Actionprobe circuits built into every device. Any node may be observed in-circuit, in real time to determine its behavior under any stimulus. The Actionprobes help to verify the results of static timing analysis along with functional and backannotated simulation. Actel MPGAs do not include Actionprobe circuits, so it is important to design testable circuits into the FPGA from the beginning before converting to an MPGA. A complete simulation procedure for thorough testing is essential to ensure the success of FPGA and MPGA designs. Make sure that the simulation procedures test each section of the chip thoroughly for proper functionality under a wide variety of inputs. Refer to the application note "Actel MPGA Testing Guidelines" for more information.

Some common methods to make each design more testable include the following:

- Add global reset inputs to all sequential logic.
- Use global, minimal skew clock signals.
- Use unused I/O pins to observe important internal nodes.
- Use loadable synchronous counters.
- Remove asynchronous logic loops.
- Remove asynchronous one-shot pulse generators.

Use ACTgen or Actel Soft Macro Library

The Actel Designer and Designer Advantage systems include ACTgen Macro Builder software to create fast, reliable soft macros automatically. You can ACTgen to generate customized macros such as counters, registers, decoders, adders, and multiplexers according to precise specifications. For other types of high-level functions, use the Actel soft macro libraries. Whether coming from ACTgen or the Actel libraries, Actel soft macros are easily migrated from FPGA to MPGA devices because of their built-in reliability and testablility.

I/O Pin Assignment

The performance and routability of Actel FPGA designs are largely determined by the effectiveness of the I/O pin assignments. Likewise, corresponding MPGA conversions have an equally important dependence upon pin placements. Actel's Designer and Designer Advantage development systems include I/O placement software to determine automatically excellent locations for the I/Os, depending on circuit topology and device resources. Since automatic pin placement generally produces superior results versus a design with manual placements, it is important to limit the number of manually placed pin locations as much as possible. The Actel application note "Selecting and Modifying I/O Assignments" thoroughly discusses the issues concerning this subject.

Since MPGA devices are specifically designed to be drop-in replacements for FPGAs, there is no need to change the I/O pin assignments as part of the conversion process. If at all possible, it is best to retain an identical pinout for all versions of a design. The performance of the MPGA may be adversely affected if many changes are made to the pin placements. However, Actel will modify the MPGA pin placements if any changes are necessary.

Summary

By following these design hints and guidelines, the conversion from programmable devices is virtually transparent to all members of a product team. Virtually any successful FPGA design can be replaced by a drop-in replacement without difficulty.

Predicting the Power Dissipation of Actel FPGAs

Introduction

Calculating the power dissipation of field programmable gate arrays (FPGAs) is similar to using the calculations for other CMOS ASIC devices, such as gate arrays and standard cells. The power dissipation depends on such factors as utilization, average operating frequency, and load conditions. In contrast, most PALs and PLDs have a fixed power consumption.

This application note discusses power dissipation and the concept of equivalent power capacitance. The general approach to calculating power in an ACT™ device will be described using equivalent power capacitance values for ACT devices. This general equation is useful if internal switching frequencies can be accurately determined. Since this is often difficult to do, a set of approximation curves based on average frequency rules of thumb are provided. The graphs provide an upper limit estimate for active power sufficient for most designs.

General Power Equation

$$P = [I_{CC}\text{standby} + I_{CC}\text{active}] * V_{CC} + I_{OL} * V_{OL} * N + I_{OH} * (V_{CC} - V_{OH}) * M \quad (1)$$

Where:

I_{CC}standby is the current flowing when no inputs or outputs are changing.

I_{CC}active is the current flowing due to CMOS switching.

I_{OL} and I_{OH} are TTL sink/source currents.

V_{OL} and V_{OH} are TTL level output voltages.

N equals the number of outputs driving TTL loads to V_{OL}.

M equals the number of outputs driving TTL loads to V_{OH}.

Determining N and M depends on the design and the system I/O. An accurate determination of power dissipation comes from two components, static and active, which are considered separately.

Static Power Component

Actel FPGAs have small static power components that result in lower power dissipation than PALs or PLDs. By integrating multiple PALs/PLDs into one FPGA, an even greater reduction in board-level power dissipation can be achieved.

The power due to standby current is typically a small component of the overall power. For an ACT 3 device, the standby power is specified as 5 mWatts, worst case.

The static power dissipated by TTL loads depends on the number of outputs driving high or low and the DC load current. Again, this value is typically small. For instance, a 32-bit bus sinking 4 mA at 0.33 V will generate 42 mWatts with all outputs driving low and 140 mWatts with all outputs driving high. The actual dissipation will average somewhere between as I/Os switch states with time.

Active Power Component

Power dissipation in CMOS devices is usually dominated by the active (dynamic) power dissipation. This component is frequency dependent, a function of the logic and the external I/O. Active power dissipation results from charging internal chip capacitances of the interconnect, unprogrammed antifuses, module inputs, and module outputs, plus external capacitance because to PC board traces and load device inputs. An additional component of the active power dissipation is the totem-pole current in CMOS transistor pairs. The net effect can be associated with an equivalent capacitance that can be combined with frequency and voltage to represent active power dissipation.

Equivalent Capacitance

The power dissipated by a CMOS circuit can be expressed by the equation:

$$\text{Power } (\mu\text{Watts}) = C_{EQ} * V_{CC}^2 * F \quad (2)$$

Where:

C_{EQ} is the equivalent capacitance expressed in pF.

V_{CC} is the power supply in volts.

F is the switching frequency in MHz.

Equivalent capacitance is calculated by measuring I_{CC}active at a specified frequency and voltage for each circuit component of interest. Measurements have been made over a range of frequencies at a fixed value of V_{CC}. Equivalent capacitance is frequency independent so that the results may be used over a wide range of operating conditions. The results for ACT 1, ACT 2, and ACT 3 devices are given in Table 1.

Table 1 • CEQ Values for ACT FPGAs

	ACT 1	ACT 2	1200XL	ACT 3
Modules	6.3	7.7	7.7	8.2
Input Buffers	16.0	18.0	18.0	1.5
Output Buffers	25.0	25.0	25.0	2.3
Clock Buffer Loads	5.3	2.5	2.5	N/A
I/O Clock Buffer Loads	N/A	N/A	N/A	0.4
Dedicated Array Clock Buffer Loads	N/A	N/A	N/A	0.5
Routed Array Clock Buffer Loads	N/A	N/A	N/A	0.5 + fixed/ device

Finding the active power dissipated from the complete design requires solving Equation 2 for each component type. This requires the switching frequency of each part of the logic. The exact equation is a piecewise linear summation over all components as shown in Equation 3. For ACT 1 and ACT 2 devices:

$$\text{Power} = [(m * C_{EQ} * f_m)_{modules} + (n * C_{EQ} * f_n)_{Inputs} + (p * (C_{EQ} + C_L) * f_p)_{Outputs} + (q * C_{EQ} * f_q)_{clk\_loads}] * V_{CC}^2 \quad (3)$$

Where:

m = Number of logic modules switching at frequency f_m

n = Number of input buffers switching at frequency f_n

p = Number of output buffers switching at frequency f_p

q = Number of clock loads on the global clock network

f_m = Average logic module switching rate in MHz

f_n = Average input buffer switching rate in MHz

f_p = Average output buffer switching rate in MHz

f_q = Frequency of global clock

C_L = Output load capacitance

For ACT 3 devices:

$$\text{Power } (\mu W) = [(m \times 8.2 \times f_1) + (n \times 1.5 \times f_2) + (p \times (2.3 + C_L) \times f_3) + (q \times 0.5 \times f_4) + ((r_1 + 0.5\, r_2) \times f_5) + (s \times 0.4 \times f_6)] \times V_{CC}^2 \quad (4)$$

Where:

m = Number of logic modules switching at f_1

n = Number of input buffers switching at f_2

p = Number of output buffers switching at f_3

q = Number of clock loads on the dedicated array clock network

A1415: $q = 104$

A1425: $q = 160$

A1440: $q = 288$

A1460: $q = 432$

A14100: $q = 697$

r_1 = Fixed capacitance due to routed array clock network

A1415: $r_1 = 60$

A1425: $r_1 = 75$

A1440: $r_1 = 105$

A1460: $r_1 = 145$

A14100: $r_1 = 195$

r_2 = Number of clock loads on the routed array clock network

s = Number of clock loads on the dedicated I/O clock network

A1415: $s = 80$

A1425: $s = 100$

A1440: $s = 140$

A1460: $s = 168$

A14100: $s = 228$

f_1 = Average logic module switching rate in MHz

f_2 = Average input buffer switching rate in MHz

f_3 = Average output buffer switching rate in MHz

f_4 = Average dedicated array clock rate in MHz

f_5 = Average routed array clock rate in MHz

f_6 = Average dedicated I/O clock rate in MHz

C_L = Output load capacitance in pF

Since all of the modules or inputs or outputs do not switch at the same frequency, a weighted average can be used. For example, a design consisting of 100 modules switching at 10 MHz and 200 modules switching at 5 MHz would have a weighted average frequency of:

$$f_{ave} = [(100 * 10) + (200 * 5)]/(100 + 200) = 6.67 \text{ MHz}$$

Determining Average Frequency

Determining the exact average frequency for a design requires a detailed understanding of the data input values to the circuit. Logic simulation can provide insight into average frequency, although simulation is limited by the percentage of real-time stimulus that can be applied. Fortunately, studies based on large numbers of ASIC designs have been made to determine rules of thumb for average switching frequency in logic circuits. These rules are meant to represent worst-case scenarios, hence their use for predicting the upper limits of power dissipation is generally acceptable. The rules given in Tables 2 and 3 for ACT 1 and ACT 2 devices. Table 4 gives the rules for ACT 3 devices. Using these rules, we can develop power estimates.

Table 2 • Rules for Determining Average Frequency for ACT 1

1.	Module Utilization = 90%
2.	Average Module Frequency = F/10
3.	1/3 of I/Os are Inputs
4.	Average Input Frequency = F/5
5.	2/3 of I/Os are Outputs
6.	Average Output Frequency = F/10
7.	Clock Net Loading = 45%
8.	Clock Net Frequency = F

Table 3 • Rules for Determining Average Frequency for ACT 2 and 1200XL

1.	Module Utilization = 80% of combinatorial modules
2.	Average Module Frequency = F/10
3.	1/3 of I/Os are Inputs
4.	Average Input Frequency = F/5
5.	2/3 of I/Os are Outputs
6.	Average Output Frequency = F/10
7.	Clock Net 1 Loading = 40% of sequential modules
8.	Clock Net 1 Frequency = F
9.	Clock Net 2 Loading = 40% of sequential modules
10.	Clock Net 2 Frequency = F/2

Table 4 • Rules for Determining Average Frequency for ACT 3

1.	Logic Modules (m)	= 80% of modules
2.	Average module switching rate (f_1)	= F/10
3.	Inputs switching (n)	= # I/Os used/12
4.	Average input switching rate (f_2)	= F
5.	Outputs switching (p)	= # I/Os used/15
6.	Output loading (C_L)	= 35
7.	Average output switching rate (f_3)	= F/2
8.	Dedicated array clock loads (q)	= fixed by device
9.	Average dedicated array switching rate (f_4)	= F
10.	Routed array fixed capacitance (r_1)	= fixed by device
11.	Routed array clock loads (r_2)	= 40% of sequential modules
12.	Average routed array switching rate (f_5)	= F/2
13.	I/O clock loads (s)	= # I/Os used
14.	Average I/O switching rate (f_6)	= F

Average Frequency Example

While some portions of a logic design switch at the system frequency, F, most of the logic switches at a reduced (or divided) frequency. Consider a 16-bit synchronous counter with a system input clock equal to F as shown in Figure 1.

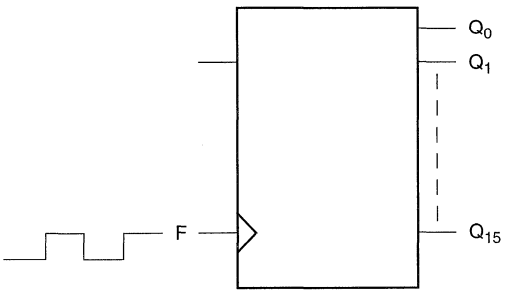


Figure 1 • 16-Bit Synchronous Counter

Where:

The Q0 output is switching at F/2 (or $1/2^1$),

The Q1 output is switching at F/4 (or $1/2^2$),

The Q15 output is switching at F/65536 (or $1/2^{16}$).

The average frequency is:

$F_{ave} = 1/16 * (1/2^1 + 1/2^2 + ...1/2^{16}) \cong F/16$

Thus, the average frequency of an n-bit synchronous counter switching at F MHz is F/n.

Estimated Power

The rules in Tables 2 and 3 are applied to ACT 1, ACT 2, and 1200XL devices. The resulting power components are detailed in Tables 5 and 6, and the total device power is shown in Figures 2 and 3. The graphs provide a simple guideline for estimating power. The tables may be interpolated when your application has different resource utilizations or frequencies. Table 4 details the rules applied to ACT 3 devices.

Table 5 • Power Components for ACT 1 (Watts)

F (MHz)	Module Power	Input Power	Output Power	Clock Power	Total Power (watts)
A1010					
1	0.005	0.002	0.009	0.019	0.035
2	0.010	0.004	0.017	0.038	0.069
5	0.025	0.009	0.043	0.096	0.173
10	0.051	0.018	0.085	0.192	0.347
15	0.076	0.027	0.128	0.289	0.520
20	0.101	0.036	0.171	0.385	0.693
25	0.126	0.046	0.213	0.481	0.866
A1020					
1	0.009	0.002	0.010	0.035	0.057
2	0.019	0.004	0.021	0.071	0.114
5	0.047	0.011	0.052	0.176	0.286
10	0.094	0.022	0.103	0.353	0.572
15	0.141	0.033	0.155	0.529	0.858
20	0.188	0.044	0.207	0.705	1.144
25	0.235	0.055	0.258	0.882	1.430

Table 6 • Power Components for ACT 2 (Watts)

F (MHz)	Module Power	Input Power	Output Power	Clock Power	Total Power (watts)
A1280/1280XL					
1	0.011	0.013	0.021	0.027	0.072
2	0.023	0.025	0.042	0.054	0.144
5	0.057	0.063	0.105	0.136	0.360
10	0.113	0.125	0.210	0.272	0.720
20	0.227	0.250	0.419	0.544	1.440
30	0.340	0.375	0.629	0.815	2.159
40	0.453	0.500	0.839	1.087	2.879
A1240/1240XL					
1	0.006	0.009	0.016	0.015	0.046
2	0.013	0.019	0.031	0.030	0.092
5	0.032	0.046	0.078	0.074	0.231
10	0.064	0.093	0156	0.149	0.461
20	0.127	0.186	0.311	0.298	0.923
30	0.191	0.279	0.467	0.447	1.384
40	0.255	0.372	0.623	0.596	1.845
A1225/1225XL					
1	0.004	0.007	0.012	0.009	0.033
2	0.008	0.015	0.025	0.019	0.066
5	0.020	0.037	0.062	0.047	0.166
10	0.040	0.074	0.124	0.094	0.332
20	0.080	0.148	0.249	0.187	0.664
30	0.120	0.222	0.373	0.281	0.996
40	0.160	0.297	0.497	0.375	1.329
50	0.200	0.371	0.621	0.468	1.661

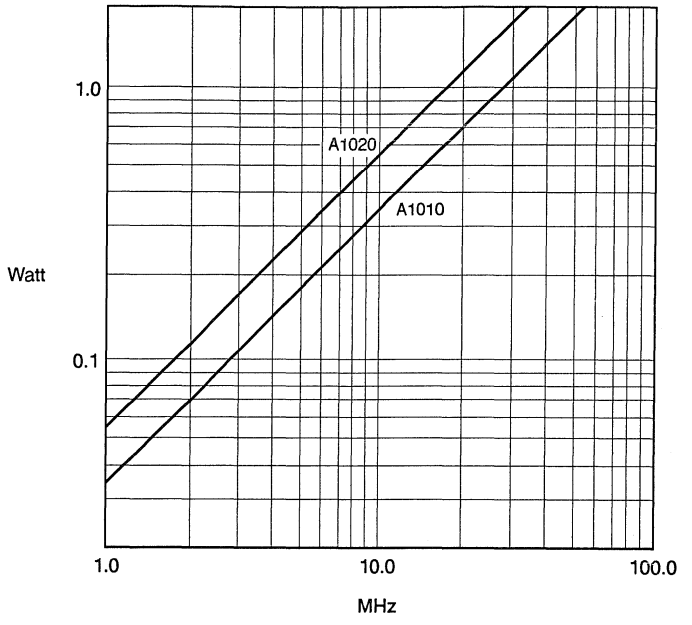


Figure 2 • ACT 1 Power Estimates

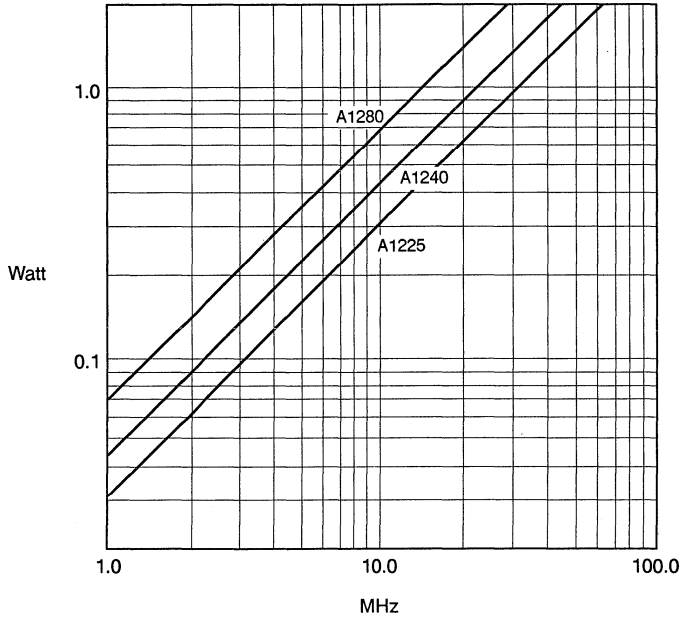


Figure 3 • ACT 2 Power Estimates of Total Power (watts)

Binning Circuit of Actel FPGAs

Introduction

All Actel FPGA devices are available in different speed grades to meet varying system design requirements. As part of the production process, devices are separated, or "binned," into groups by their measured performance. The performance of each device is measured by a dedicated binning circuit that closely characterizes its speed. The Actel binning circuit measures the AC performance of a device prior to programming. Since the binning circuit shares the same process as the rest of the die, the binning circuit speed reflects the speed of the device after programming. Actel guarantees the performance of every device for all speed grades. Therefore, the following descriptions of ACT™ device binning circuits are mostly for informational purpose only.

ACT 1 Binning Circuit

The ACT 1 binning circuit consists of one input buffer, n logic modules (n = 16 for A1010/A1010A/A1010B and n = 28 for A1020/A1020A/A1020B), and one output buffer. The binning circuit delay is obtained by setting the device to the test mode (that is, MODE pin = HIGH), and setting up the appropriate mode register bits. Using the SDI pin as the mode register input and DCLK as the clock, seven mode register bits are clocked into the device in the sequence, 1011101. After this setup process, the propagation delay of the binning circuit is measured from the input pin BININ to the output pin PRA. Figure 1 shows the delay of binning circuit T_{PDB}. Table 1 shows the binning circuit pin assignments for ACT 1 family devices.

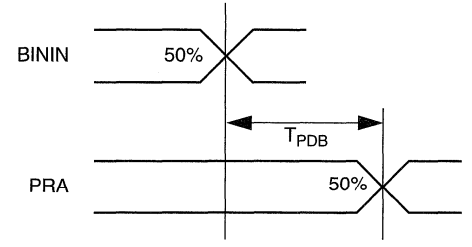


Figure 1 • T_{PDB} Measurement Waveform

ACT 2 and 1200XL Binning Circuit

The delay path of the ACT 2 and 1200XL binning circuits includes an input pad through an input module, n combinatorial logic modules (n = 16 for A1280/A1280A/A1280XL, 12 for A1240/A1240A/A1240XL, and 10 for A1225/A1225A/A1225XL), an output module, and a pad driver. Binning circuit delays are obtained by setting the MODE pin high, then shifting data into the internal mode, control, and binning registers through the SDI pin for the appropriate mode of operation. To set up the binning circuit, several data patterns are shifted into SDI with each rising edge of DCLK. The first ten data bits are counter bits used to set up the internal counter length. The next 21 bits are Mode bits used to set up the mode of operation, and the remaining bits are used to fill the registers in the shift register chain. There is one stop bit at the end. Then, the binning circuit delay can be measured from input pin, BININ, to output pin, BINOUT. Tables 2 and 3 list the binning circuit pin assignments and the data patterns for ACT 2 binning circuits.

ACT 3 Binning Circuit

The ACT 2, 1200XL, and ACT 3 binning circuits are identical except for the data patterns. Tables 4 and 5 list the binning circuit pin assignments and the data patterns for ACT 3 binning circuits.

Table 1 • ACT 1 Binning Circuit I/O Pins

	PLCC/JQCC			CPGA	CQFP	PQFP
	44-pin	68-pin	84-pin	84-pin	84-pin	100-pin
MODE	34	54	66	E11	55	92
SDI	36	56	72	B11	61	98
DCLK	37	57	73	C10	62	99
BININ 1010	9	12	NA	D2	NA	34
1020	7	10	13	C2	2	32
PRA	38	58	74	A11	63	100

Table 2 • ACT 2 and 1200XL Binning Circuit I/O Pins

	A1280/A1280A/A1280XL			A1240/A1240A/A1240XL		
	176 PGA	172 CQFP	160 PQFP	144 PQFP	132 PGA	84 PLCC
MODE	C3	1	159	2	A1	12
SDI	B14	131	38	110	B12	76
DCLK	B3	171	2	144	C3	10
BININ	N3	44	119	37	L3	34
BINOUT	P2	45	118	38	M2	35

	A1225/A1225A/A1225XL		
	100 PGA	100 PQFP	84 PLCC
MODE	C2	4	12
SDI	C8	79	76
DCLK	C3	2	10
BININ	K2	28	34
BINOUT	L2	29	35

Table 3 • ACT 2 and 1200XL Binning Circuit Data Patterns

Device	#Bits	Counter Bits	Mode Bits	Shift Register Bits and Stop Bit
A1280/A1280A/A1280XL	276	0001101010	0000 0101 1100 0101 0000 0	one 1, followed by 244 0s
A1240/A1240A/A1240A	218	0110010011	0000 0101 1100 0101 0000 0	one 1, followed by 186 0s
A1225/A1225A/A1225XL	176	1010101011	0000 0101 1100 0101 0000 0	one 1, followed by 144 0s

Table 4 • ACT 3 Binning Circuit I/O Pins

	A1425/A1425A		A1460/A1460A	
	133 CPGA	160 PQFP	208 PQFP	207 CPGA
MODE	E3	9	11	D7
SDI	C2	2	2	C3
DCLK	D4	160	208	E4
BININ	K3	38	50	D13
BINOUT	L4	39	51	E14

Table 5 • ACT 3 Binning Circuit Data Patterns

Device	#Bits	Counter Bits	Mode Bits	Shift Register Bits and Stop Bit
A1425/A1425A	410	1101100001	0000 0101 1100 1101 0000 0	one 1, followed by 378 0s
A1460/A1460A	553	0101101100	0000 0101 1100 1101 0000 0	one 1, followed by 521 0s

Global Clock Networks

The Actel architecture provides global clock networks that allow high fanout drive for flip-flops and latches with minimal skew. Table 1 shows the available global networks and their characteristics, which are defined as follows.

- *Routed clocks* are clock networks which can be used by selecting CLKBUF/CLKBIBUF or CLKINT macros or both.
- *Dedicated clocks* are clock networks that are directly wired to sequential and I/O modules. They contain no programming elements in the path from the I/O pad driver to the input of S-modules or I/O modules; they provide sub-nanosecond skew and guaranteed performance.
- A *special* network refers to a special hard-wired input for I/O modules that can only drive preset/clear pins of I/O modules.

Note that the *Internal Drive Option* for ACT™ 2, 1200 XL, and ACT 3 can only be utilized by selecting the CLKINT macro to drive an internal clock network.

ACT 1 Clock Network

A single clock distribution network is provided on ACT 1 arrays. The clock network provides unlimited fanout (the ability to drive all logic modules in the array) with minimal delay and skew time. Figure 1 illustrates the clock distribution network for an ACT 1010 array. It is arranged similarly in the ACT 1020 array.

The network is driven by a specific I/O pin that drives a dedicated on-chip buffer tree. Each row of logic modules (8 on the A1010 and 14 on the A1020) has a dedicated buffered clock track. The clock distribution network is selected automatically when the CLKBUF macro is used in the schematic and assigned to its dedicated package pin. The CLK I/O pin can also be used for normal I/Os by assigning INBUF, OUTBUF, TRIBUFF, or BIBUF to the CLK pin location.

Clock Balancing Scheme

Clock balancing equalizes the clock loads on each branch of the global clock network, thereby minimizing clock skew. Clock skew can cause setup and hold time problems. The clock balancing strength sets the level of clock balancing for ACT 1 designs only. It is not used for ACT 2 and ACT 3 designs because the global clock networks for those families have been designed for minimal clock skew. The results are design dependent. If more than 50 percent of the logic modules are driven by the global clock, the effect is minimal. Also, strong clock balancing may result in increased delays and reduced routability.

Table 1 • Global Clock Attributes

Input Pad Name	Type	Family	Number	Internal Drive Option	Macro	Note
CLK	routed	ACT 1	1	No	CLKBIBUF, CLKBUF	Can adjust skew with clock balancing
CLKA, CLKB	routed	ACT 2, 1200XL	2	Yes	CLKBIBUF, CLKBUF, CLKINT	Use CLKINT for Internal Drive Option
CLKA, CLKB	routed	ACT 3	2	Yes	CLKBIBUF, CLKBUF, CLKINT	Use CLKINT for Internal Drive Option
HCLK	dedicated	ACT 3	1	No	HCLKBUF	Connected to all S-modules
IOCLK	dedicated	ACT 3	1	No	IOCLKBUF	Connected to all I/O modules
IOPCL	special	ACT 3	1	No	IOPCLBUF	Connected to I/O module set and reset pins

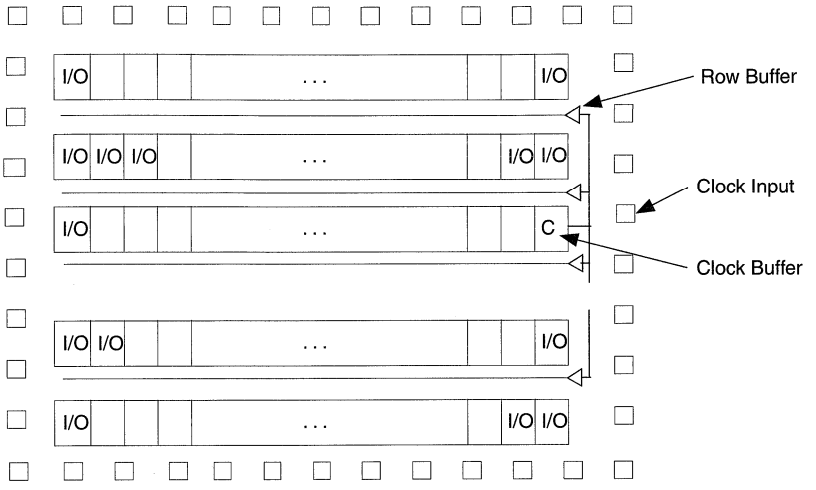


Figure 1 • Clock Distribution Network for an ACT 1010

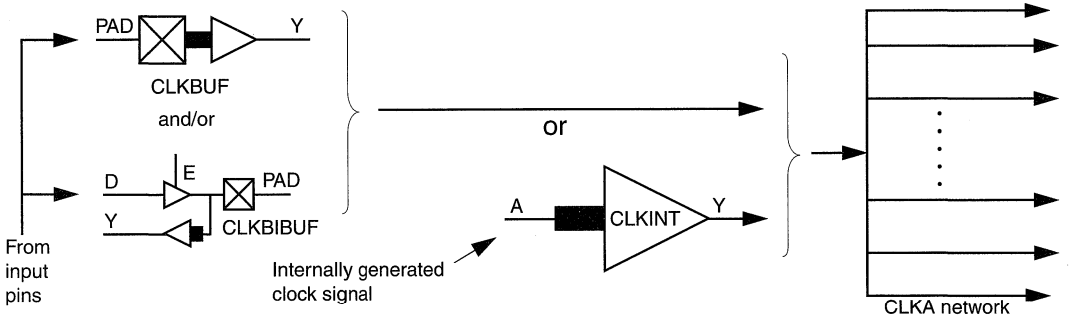


Figure 2 • One of Two ACT 2 Clock Networks (CLK0)

ACT 2 and 1200XL Clock Networks

Two low-skew, high fanout clock distribution networks are provided in the ACT 2 and 1200XL architectures. Figure 2 illustrates the implementation of one of two clock networks using CLKBUF/CLKBIBUF or CLKINT macro or both. ACT 2 and 1200XL devices offer two identical clock networks. The clock modules are located in the top row of I/O modules. Clock drivers and a dedicated horizontal clock track are located in each horizontal routing channel. The clock input pads may also be used as normal I/Os, bypassing the clock networks. These networks are referred to as CLKA and CLKB. Each network has a clock module that selects the source of the clock signal and may be driven as follows:

- externally from the CLKBUF macro
- externally from the CLKBIBUF macro
- internally from the CLKINT macro

As mentioned above, the macro CLKBUF is used to connect one of the two external clock pins to a clock network, and the macro CLKINT is used to connect an internally generated clock signal to a clock network. Figure 3 illustrates the implementation of internal clock network using CLKINT macro. In the figure, the input signals are driven by regular I/O buffers, not CLKBUF or CLKBIBUF. These input signals drive a user-created Clock Conditioning Module, which generates the internal clock signal. This signal is subsequently driven by the CLKINT macro.

ACT 3 Clock Networks

The ACT 3 architecture contains four clock networks: two high performance dedicated clock networks and two general purpose routed networks. The high performance networks function at up to 150 MHz, while the general purpose routed networks function at up to 75 MHz. Figure 4 illustrates the ACT 3 clock networks.

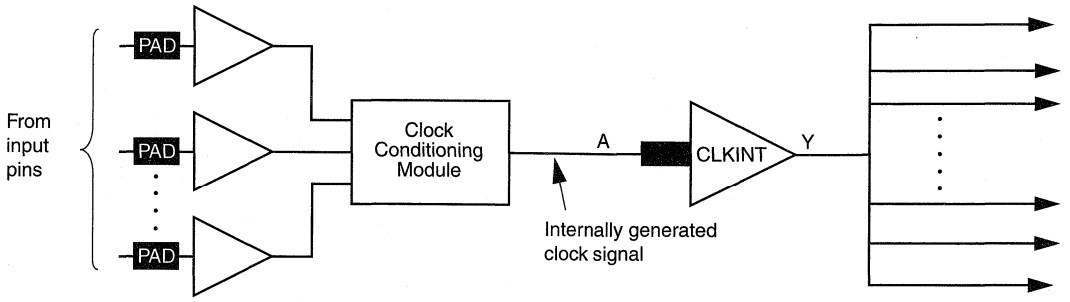


Figure 3 • Implementation of Internal Clock Network Using CLKINT Macro

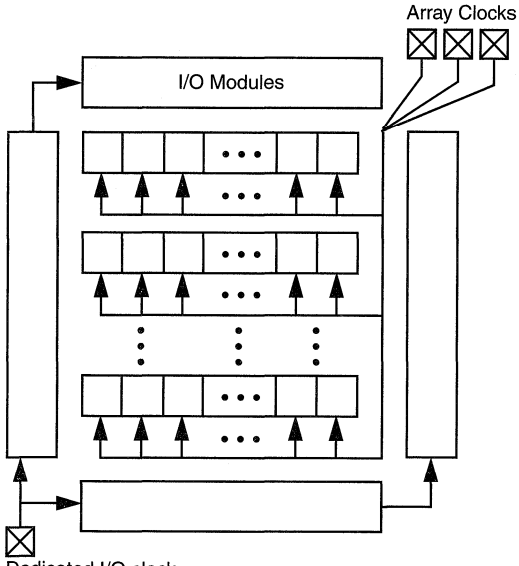


Figure 4 • ACT 3 Clock Networks

Dedicated Clocks

There are two dedicated clock networks: one for the array registers known as Dedicated Array Clock (HCLK) and one for the I/O registers known as Dedicated I/O Clock (IOCLK) The clock networks are accessed by special I/Os. Figure 5 shows the macros that drive the Dedicated Array and I/O clock networks in ACT 3.

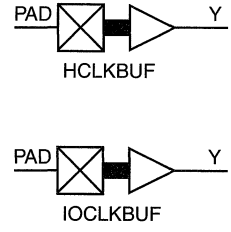


Figure 5 • Dedicated Array and I/O Clock Buffers

HCLK is a dedicated hard-wired clock input for sequential modules. HCLK is directly wired to each S-module and offers guaranteed clock speeds independent of the number of S-modules being driven. IOCLK is a dedicated hard-wired clock input for I/O modules. IOCLK is directly wired to each I/O module and offers guaranteed clock speeds independent of the number of I/O modules being driven. These dedicated clock networks support high performance by providing sub-nanosecond skew and guaranteed performance. Dedicated clock networks contain no programming elements in the path from the I/O pad driver to the input of S-modules or I/O modules.

Routed Clocks

The routed clock networks for ACT 3 have the same characteristics as the ACT 2 and 1200XL networks as shown in Figure 2 and Figure 3. The macros that drive routed clock networks in ACT 1, ACT 2, 1200XL, and ACT 3 are shown in Figure 6.

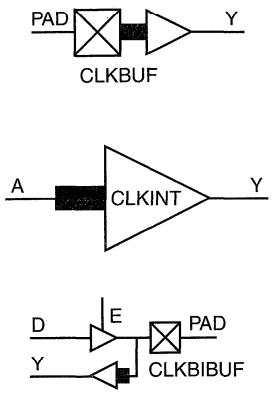


Figure 6 • Dedicated Routed Clock Buffers

CLKA and CLKB are global signals with unlimited fanout. Refer to ACT 2 clock networks described above for more detail on ACT 3 routed clocks.

Special Hard-wired Preset/Clear Network

IOPCL is a dedicated special hard-wired input for I/O modules. It is directly wired to the Preset and Clear inputs of all I/O registers. IOPCL functions as an I/O when no I/O preset or clear macros are used. Figure 7 shows the IOPCLBUF macro that drives the dedicated hard-wired Preset/Clear network. IOPCLBUF can only be connected to the preset/clear pins of I/O macros.

Figure 7 • Dedicated Preset/Clear Network Buffer

The routed clocks CLKA and CLKB can also be used to drive high fanout nets like resets, output enables, or data enables. This saves logic modules and results in performance increases in some cases.

Clock Connections for ACT 2, 1200XL, and ACT 3

To minimize loading on the clock networks, only a subset of module inputs has antifuses on the clock tracks. Therefore, only a few of the C-module and S-module inputs can be connected to the clock networks. To further reduce loading on the clock network, only a subset of the horizontal routing tracks can connect to the clock inputs of the S-module. Figure 8 illustrates the connections to the clock networks.

Creating a User-Defined Clock Distribution Network

Some applications require many internal clock networks. For these type of designs or for clock networks with a small number of loads, it may be better to create a user-defined clock network. Because this clock network is not a built-in, dedicated circuit, the skews and delays cannot be guaranteed.

However, by intelligently placing the I/O pins and declaring the nets associated with the clock network as critical, the automatic placement of these macros create a network that controls clock skew and delay. The results can be verified by running a simulation or using the ALS Timer. It may be necessary to place and route again to meet timing requirements. To minimize the skew between paths, try to equalize the loading in each leg of the clock distribution network.

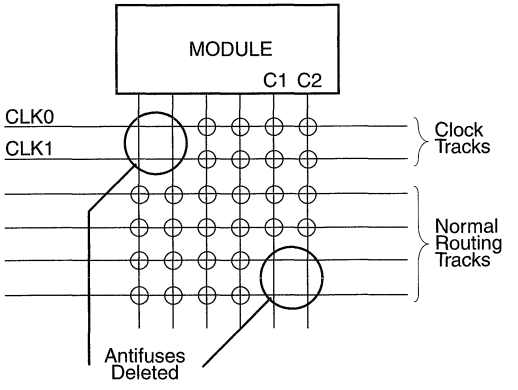


Figure 8 • Fuse Deletion on Clock Networks

ACT 1 Global Clock Network Enhancement

The global clock network for the ACT 1 family of Actel devices prior to 1010B and 1020B (1.0µ process) is very useful yet it possesses limitations in certain system applications. If a design is placed such that one row is heavily loaded with flip-flops, while an adjacent row is lightly loaded, there will be considerable skew between clock signals in each row.

Improvements

The improved lithography offered by the 1.0µ process, and greater understanding of clock network usage in FPGAs, offered the opportunity to implement circuit enhancements. Refer to figure 1 and figure 2 in reference to the following discussion. Improvements were made in the following areas:

- TTL Translator at Pad. The B die revision has separate paths at the Pad for the clock network and the user input buffer. The path separation of clock network and input buffer increases the capacitance at the pin but such an increase is trivial in comparison with the package and/or board capacitances. To allow increased drive and speed, the clock network uses an independent TTL driver that is double the size of the A die revision.
- Improved Buffers. Following the TTL translator is a buffer which drives the inputs to the drivers in each row. In the B die revision the buffer is a high-efficiency six-transistor driver. This driver allows the use of large devices without any increase in power dissipation because its design supresses totempole current spikes. The same concept is applied to the row drivers.
- Shorting the Row Drivers. The row drivers are shorted together in the B die revision behind the programming isolation device. This allows a lightly loaded row to offer charging current to a heavily loaded neighboring row. The result is a markedly reduced clock skew between rows for a given placement. In addition, shorting the row drivers for a given tolerable skew allows more flexible placement of clocked macros.
- Widening the Isolation Device. The isolation device is a high voltage NMOS transistor. During programming this device is OFF, isolating the high programming voltage (17 V) of the channel from the 5 V row drivers. In normal mode the device is ON, with a pumped gate voltage of around 13 V, and passes full CMOS levels. Improved process lithography allows an increased width for this device, lowering series resistance at the top of the clock RC tree. Reducing this resistance allows smaller propagation delays and further reduced clock skew.

Design Example

The following is a design example extracted from the ALS Timer:

The design occupies 93% of the 1020B device, with 100% pin utilization (69 I/O out of 69 I/O modules). The clock load is 196 (196 registers are driven by the same clock network). Pin skew or PSK is a measurement of the difference in delay between the common clock pins of two registers driven by the same clock. Note there must be a logical path between the two registers, i.e. the output of the first register drives the data input of the second register. The clock to register skew or CSK is a measurement of the difference in delay between the clock pad and all registers driven by that clock signal (CSK is measured manually). Clock buffer delays, or Tckh and Tckl, are the clock pad to Y delay input, low to high, and high to low respectively.

Device	*1020B*	*1020B*
Speed	*-2*	*Standard*
Load *(Fanout)*	*196*	*196*
PSK	*0.4ns*	*0.5ns*
CSK	*0.8ns*	*0.9ns*
Tckh	*6.6ns*	*8.8ns*
Tckl	*8.3ns*	*11.2ns*
Max *clock pad to register* ***delay***	*10.7ns*	*14.2ns*
Min *clock pad to register* ***delay***	*9.9ns*	*13.3ns*

A Power-On Reset (POR) Circuit for Actel Devices

The state of a system at start-up is an important consideration in designing a circuit. It is usually desirable to provide an input signal at start-up to reset synchronous circuitry. Otherwise, the system may initially operate in an unpredictable fashion because flip-flops are not designed to power-on in any particular state. Figure 1 shows a typical power-on reset (POR) circuit from which the series resistor, R1, is omitted for TTL circuits.

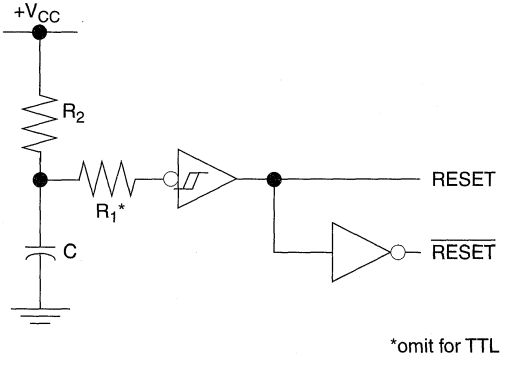


Figure 1 • Power-On Reset Circuit

This resistor is necessary with CMOS implementations to prevent damage to the device when power is removed from the circuit. Otherwise, the capacitor would try to power the system via the CMOS input gate protection circuit. A Schmitt trigger (40106, 74LS14) maybe advantageous in making the RESET signal switch off cleanly. The hysteresis symbol shown in Figure 1 indicates an inverter with a Schmitt trigger input such as the CMOS 40106 hex inverter. The following sections describe the power-up conditions of an Actel device and a recommended POR circuit.

Behavior of ACT™ 1 FPGA Inputs

During power-on, the +5 V logic supply rail of a system typically rises from 0 to +5 V in 50 ms or less. Because regulator outputs are usually current limited during this transition, the rise time is more or less linear, with a slope in the range of 0.1 V/ms to 5 V/ms. Each Actel FPGA has a universal pad driver design that may be configured as an input, output, three-state output, or bidirectional input/output. This configuration of the pad driver is accomplished by programming antifuses in the pad driver circuitry.

As the +5 V logic supply rail passes through the region from approximately +2.2 V through +2.5 V, pad drivers that have been programmed as inputs may behave temporarily as outputs that are in the logical '1' state. Thus, these input pins will temporarily source current (approximately 8 to 10 mA, if not otherwise limited) into whatever driver is connected to them. They will be sourcing this current from the +5 V logic rail, which at this time is at +2.2 to +2.5 V. This duration is a function of the power rail rise time. For +5 V rails that come up quickly, at 5 V/ms, the duration of the behavior will be approximately 60 µs. For supply rails that rise slowly, at 0.1 V/ms, the duration of the current sourcing behavior will be 3 ms. In the former case, the Actel input can deliver as much as 0.6 µC to the circuit that drives it; in the latter case, the charge is as much as 30 µC. For many driver circuits, this amount of charge is insignificant; however, for others it may be unacceptable.

Inserting a series resistance of sufficient size into the Actel input line can limit the effect of this behavior. In the case of the POR circuit in Figure 2, the series resistance must be chosen to keep $\Delta V \leq 1V$. This guarantees that the POR remains at a logic '0' following the irregularity when the logic supply rail is at approximately 2.5 V, where $\Delta V/\Delta t = i/C$ is the voltage rise time of the capacitor. The capacitor is charged through a resistor to the 5 V logic supply rail, and the diode across the resistor is used to discharge the capacitor at power-off. For a power rail rise time of 0.1 V/ms, the duration of this behavior will be approximately 3 ms. This means that for a POR capacitance of 0.1 µF, the current out of the Actel device input must be limited to

$$i = C\Delta v / \Delta t = (0.1\ \mu F * 1\ V) / 3\ ms = 33\ \mu A$$

which can be achieved using a resistance of 2.24 V/33 mA = 68 kΩ.

Furthermore, for drivers that cannot accept a source of current at their outputs or for a multiple-source data bus, it is strongly recommended that the bus drivers be three stated during POR.

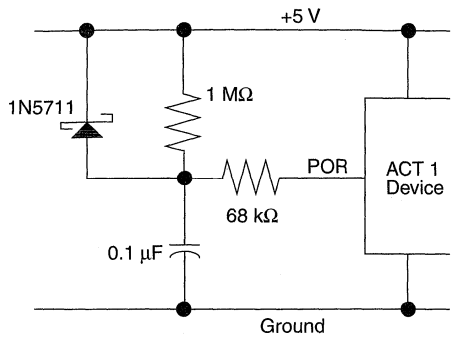


Figure 2 • Power-On Reset (POR) Circuit with Current-Limiting Resistor

Device Behavior during Power-Up

Actel devices have been characterized with two V_{CC} rise time conditions—fast power-up of approximately 0.5 V/µs and slow power up of approximately 0.2 V/ms. During power-up, the following I/O conditions were characterized: input characteristics, tristate outputs, output driving low, output driving high, and output toggling. Normal input behavior is defined as the input being high impedance (<10 µA leakage). Normal output behavior means the output will either be tristate or be in a predictable state as defined by the logic of the user's design. The description that follows summarizes what happens to the Actel devices during power-up.

ACT1

Inputs and outputs will behave normally within 100 µs after V_{CC} reaches 4.75 V with fast power-up and 3.5 V with slow power-up. With fast power-up, inputs remain in a high impedance state. With slow power-up, prior to reaching 3.5 V, inputs will momentarily behave as outputs driving high or low. This happens in about a 0.3 V window between 1 V and 2.5 V. The length of time this high or low level is exerted depends on the rise time of V_{CC}. At 0.2 V/ms, the input went high for about 1 ms and twice went low for about 150 µs each time. Tristated outputs will behave about the same as inputs. Outputs driving high or low will sometimes go to the opposite state (or tristate). Toggling outputs may not perform as expected until Vcc reaches 3.5 V. When Vcc is between 1.5 V and 3.5 V, I_{CC} will increase to 10–60 mA and then return to normal. The duration is dependant on the Vcc rise time, slow power-up being the worst case.

The inputs and outputs of the 3.3 V devices will behave normally within 200 µs after Vcc reaches 3 V for fast power-up and within 100 µs after Vcc reaches 2 V for slow power-up.

ACT2

Normal operation will occur within 100 µs after V_{CC} reaches 3.5 V for both fast and slow power-up. With slow power-up, inputs and tristated outputs will behave as outputs driving high for approximately 200 µs during a 0.3 V window where V_{CC} is between 1.5 V and 3.0 V. Inputs and tristated outputs stay at high impedance with fast power-up. I_{CC} will rise to 10–50 mA when V_{CC} is between 2 V and 4.5 V and then return to normal.

The ACT 2XL family is expected to reach normal operation within 100 µs after V_{CC} reaches 3 V.

ACT3

Normal operation for inputs and outputs will occur within 100 µs after V_{CC} reaches 2.75 V. Before reaching the point of normal operation, all inputs and outputs are in a high impedance state (tristate) regardless of V_{CC} rise time. I_{CC} rises to 10–60 mA when Vcc is between 2 V and 3 V and then returns to normal.

Summary

Use these methods for avoiding POR problems. The transistors for these devices are turned on at approximately 0.7 V, their threshold voltage, while the circuit is functionally operational at a voltage level of approximately 3.3 V. The global routed clocks in Actel devices can also be used as resets for synchronous circuits when connected to either CLEAR or PRESET inputs of synchronous macros for the ACT 1 family, and to the CLEAR input for ACT 2 and ACT 3 families.

Simultaneously Switching Output Limits for Actel FPGAs

Introduction

For high performance field programmable gate arrays (FPGAs) with many I/Os, the allowable number of Simultaneously Switching Outputs (SSOs) for each device is an important issue for system designers. The limits for SSOs depend on factors such as package type and die size. Use the following guidelines to help ensure reliable system designs.

System Noise and Transients

Noise generated by off-chip drivers is a major concern in FPGA design for high-performance systems. Noise is closely related to interconnections and increases as a function of fast rise times, large total chip currents and die dimensions, and small spacing between components on-chip and onboard. As clock frequencies and die dimensions increase, signal wavelengths become comparable to wire lengths, thereby making better antennas. Reduction in the spacing between circuits leads to an increase in the capacitive, inductive, and resistive couplings. The voltage (IR) drop across the power lines becomes more significant as the current densities increase and the feature size decreases. With larger amounts of current switching, the inductive noise associated with the power lines also increases. These current transients may generate large potential drops because of the inductance of the power distribution network and is referred to as simultaneous switching (ΔI) noise.

When several circuits switch simultaneously, the current supplied by the power lines can change at a very fast rate and the inductive voltage drop along the line can cause the power supply level to fall. This voltage drop is proportional to the switching speed, the number of drivers switching simultaneously, and the effective inductance of the power lines. This effect can be summarized by Faraday's Law, which states that any change in the magnetic flux will be confronted by an opposition, a self induced EMF, determined by the rate of change in the total magnetic flux, represented by this equation:

$$EMF = d\Phi/dt = LdI/dt$$

This reduction in the supply level diminishes the current drive of a circuit, increases the delay, and may lead to the spurious switching of the receiver.

SSO Recommendations

Actel defines SSOs as any outputs that transition in phase within a 10-ns window. The output current of these drivers is shown in the *Actel FPGA Data Book and Design Guide*. The amplitude and duration of the ground bounce is a function of the number of outputs switching simultaneously and the capacitive loading of the outputs.

Table 1 shows the recommended SSO limits for the Actel FPGA family. These may vary because the amount of ground bounce that an application can tolerate is difficult to determine. Worst-case conditions are simulated by placing the I/Os adjacent to each other while driving 20 pF, 35 pF, and 50 pF loads. The observed ground bounce is less than 1.5 V, with a pulse width of less than 2.0 ns. This is within the acceptable limits of the dynamic threshold of 74F, 74LS, and ACT families. Results from the EDN Special Report on Ground Bounce Tests by David Shear[1] show a plot of the ground bounce pulse amplitude versus pulse width duration (ns) for different logic families. This article shows that as the input pulse width gets shorter, the voltage must be higher to affect the output of the device. Exceeding the recommended limits may result in larger ground bounce. The output drivers should be placed separately if more outputs must be switched simultaneously. This arrangement reduces the mutual inductance produced between adjacent I/Os resulting in a lower ground bounce. If necessary, buffers may also be inserted in the path before the output buffer. This will reduce the probability of adjacent drivers switching within 10 ns of each other.

References:

1. David Shear, "EDN Special Report on Ground Bounce Tests," *EDN*, April 15, 1993, p. 120–134,.

Table 1. Recommended SSO Limits for Actel FPGAs

Device	Package	Maximum Recommended SSOs for Loads		
		20 pf	35 pf	50 pf
A1010A/A1020A	44 PLCC	40	22	16
A1010A/1020A	68 PLCC	60	34	24
A1020A	84 PLCC	80	45	32
A1010A/1020A	84 PGA	80	45	32
A1010A/A1020A	100 PQFP	80	45	32
A1280/A1280XL	PG 176, PQ 160	160	90	64
A1240/A1240XL	PG 132, PQ 144	120	68	48
A1240/A1225/A1225XL	84 PLCC	80	45	32
A1225/A1225XL	100 PGA, PQFP	80	45	32

Implementing Three-State and Bidirectional Buses with Multiplexers in Actel FPGAs

Three-state logic is used in conventional MSI logic devices to allow buses where multiple drivers are directly connected to one or more loads. Figure 1 shows a typical bus configuration with TTL three-state bus drivers and registers. Each driving device has a control input that places all outputs in a high impedance state when asserted. (For the register, the control pin is OC; for the bus driver, there are two control pins, 1G and 2G). To prevent data collisions, only one driver can be active at a time; the other drivers must be in a high impedance state. The four NAND gates perform a logic decoding function to ensure that only one driver is active at a time. If the first bus driver is selected to be active via SELA and SELB, then the data bits W0 to W7 will drive the bus (BUS0 to BUS7). Similarly, the other data bits will be selected when the respective register is active. The loads on the bus are not shown in Figure 1.

To make effective use of routing resources for many different applications, the Actel FPGA implements internal multiple drivers on a net with multiplexers instead of three-state logic. Figure 2 depicts the Actel implementation of the three-state bus discussed above. In this case, a 4 to 1 multiplexer is used for each bit of the bus to redirect the desired signal from one of four sources (W, X, Y, or Z). In addition to replacing the three-state nature of the MSI devices, the multiplexer eliminates the need for the LS241 bus drivers (inputs W and X). The desired source is selected by the two multiplexer select lines, which eliminate the need for the decoding logic used in the MSI implementation. For greater than four sources, the MX8 8 to 1 multiplexer can be used in a similar fashion.

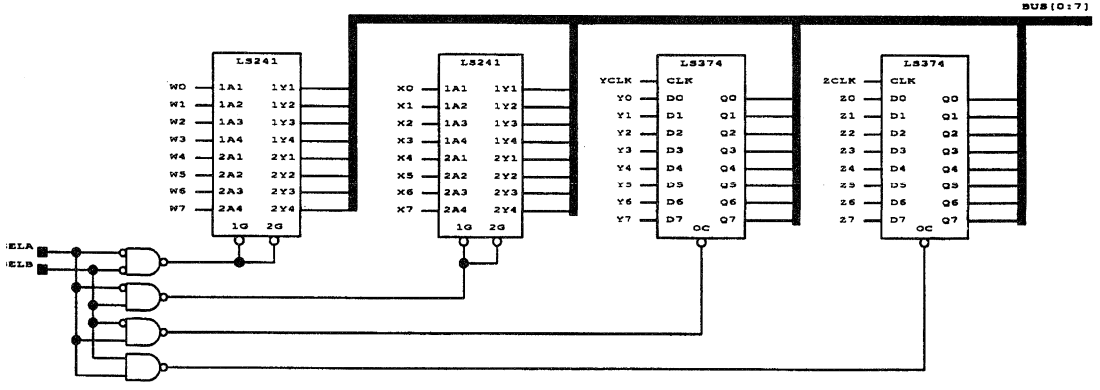


Figure 1 • Three-State Bus Implementation with MSI Logic

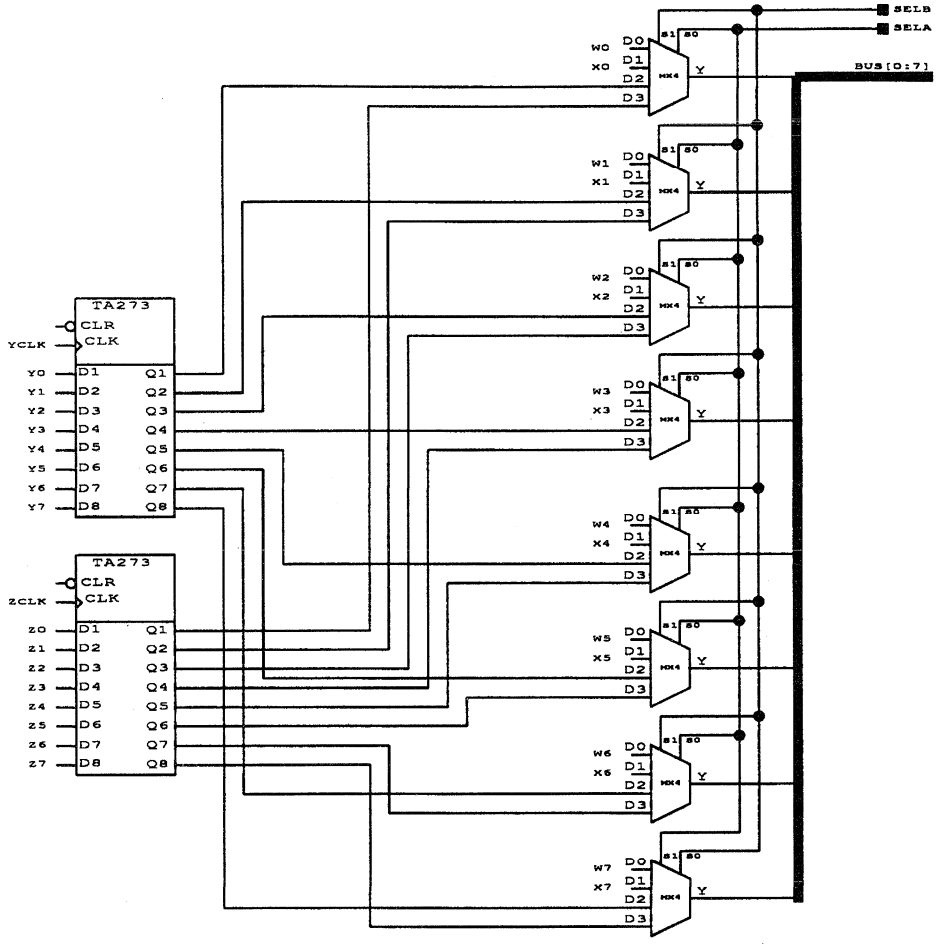


Figure 2 • Three-State Bus with Actel FPGA Using Multiplexers

Bidirectional signals can also be replaced readily with multiplexers in the Actel architecture. Figure 3 shows the conversion of a popular transceiver to a 2 to 1 multiplexer.

Figure 4 illustrates the conversion to the multiplexer implementation of three bidirectional transceivers driving a bus.

The multiplexer circuit assumes that A's driver also drives any inputs on the A sub-bus, B drives B's inputs, and so on. Two additional sub-buses can be made available by controlling the select lines accordingly.

If the bidirectional element is located at the FPGA pad, the BIBUF macro can be used directly. Figure 5 shows a simple circuit using the LS245 transceiver with flip-flops driving and leading the bus. Figure 6 shows the Actel implementation of the BIBUF macro and the DFM multiplexed flip-flop. Note that when enable is low the pad drives the B flip-flop and that when enable is high, A drives both the PAD and B.

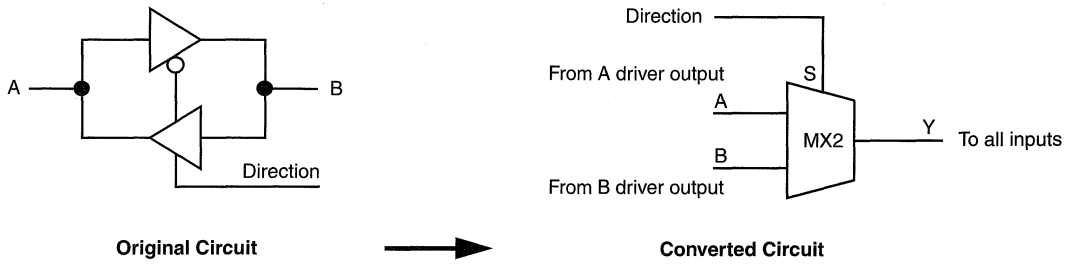


Figure 3 • Transceiver Conversion to a 2 to 1 Multiplexer

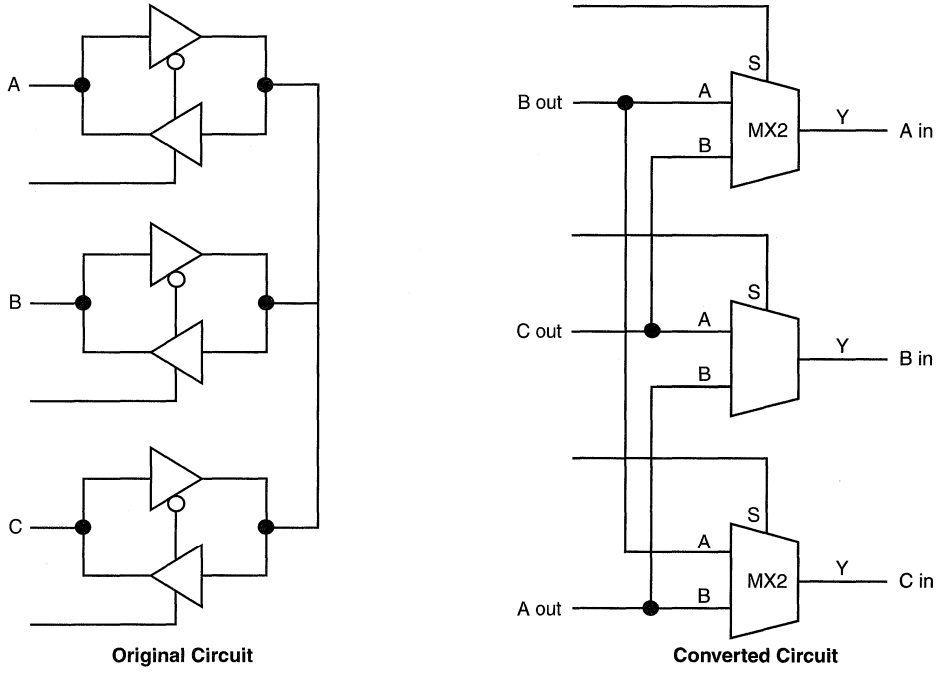


Figure 4 • Three Bit Transceiver Conversion

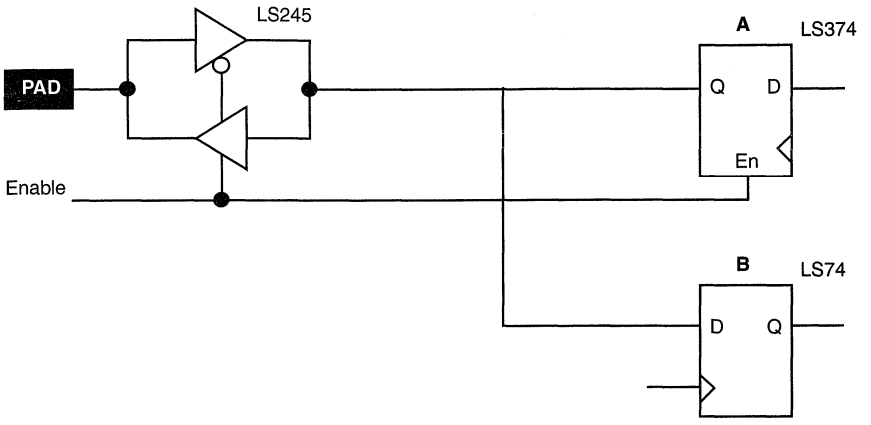


Figure 5 • Bidirectional Bus with LS245

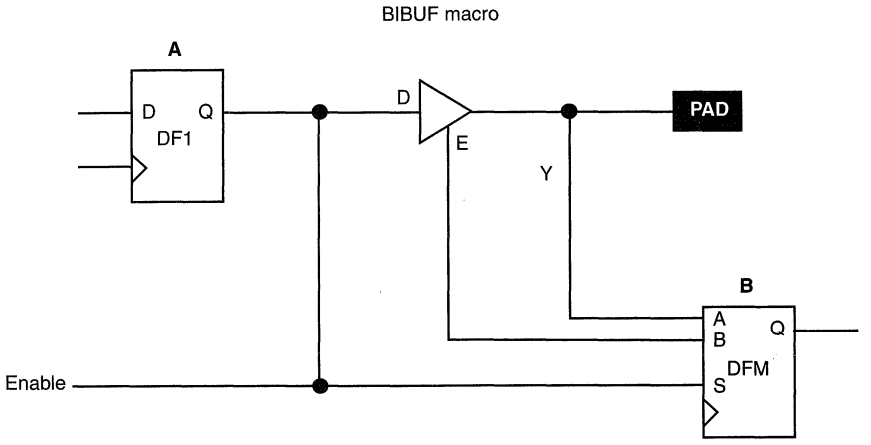


Figure 6 • Bidirectional Bus with BIBUF Macro

Oscillators for Actel FPGAs

Crystal Oscillator

Oscillators are fundamental design circuits used to provide a reference clock signal essential for digital designs. Crystal oscillators provide a simple solution for precise, stable, and calibration-free clocks. An on-chip crystal oscillator can be implemented with Actel devices using the traditional configuration shown in Figure 1. This oscillator has been tested up to 20 MHz and is used in Actel programmers. The 10 MΩ resistor provides a negative feedback path for the inverter, which makes it behave like a high-gain amplifier. The remaining passive elements, including the crystal, form a pi network that provides a 180 degree phase inversion. The inverter will lock on to the parallel resonant frequency of the crystal, thus providing a very stable output. The RC network also acts as a low-pass filter to ensure that the crystal operates at the fundamental frequency and not a harmonic frequency. The capacitor values range from 5 to 30 picofarads and depend on the crystal frequency. Some experimentation is suggested to get an optimal value for a specific design. Generally, the two capacitors will have the same value, although the capacitor connected to the input of the inverter can be varied independently to alter the output frequency by ± 0.1 percent.

A second OUTBUF or CLKBIBUF output buffer is used to provide a sharper clock signal at full amplitude when used outside the FPGA. Alternatively, the output buffer can be replaced with an internal buffer, which will allow a direct connection to internal macros. For ACT™ 2 and ACT 3

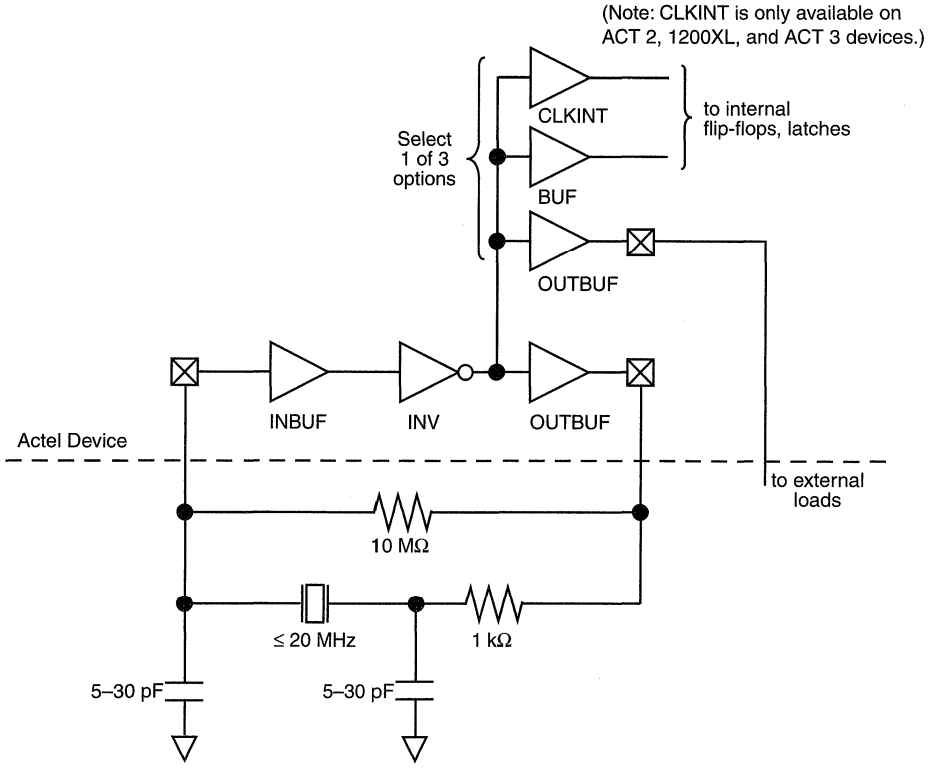


Figure 1 • Crystal Oscillator Circuit

devices, the CLKINT macro can be used allowing high fanout drive capability of internal macros with minimal skew.

Fix the placement of the oscillator I/O macros to adjacent package pins to minimize internal delays. Consult the *ALS User Guide* for more information on fixing pins. Also, fixing the I/O macros near ground pins and far from other high-speed switching I/O pads minimizes noise effects.

RC Oscillator

For applications not requiring the accuracy of a crystal, there is an RC oscillator that can be used in an Actel device as shown in Figure 2. As a strong word of caution, this circuit is not recommended for system clocks, since it is heavily dependent on resistor and capacitor tolerances, process variation, and temperature. Some applications for this lower cost oscillator include LCD backplane and debounce circuits.

The circuit reaches alternate switching thresholds by charging and discharging the capacitor with resistor R2. The R1 resistor provides a better square wave output by minimizing effects of input protection diodes of the input buffer. The approximate formula for output frequency is:

$$frequency \cong \frac{1}{2.2\ R2\ C}$$

The formula is most accurate if parameters have the following limitations:

R1 > 10R2

10K > R2 > 1 M

1000 pF > C > 10 µF

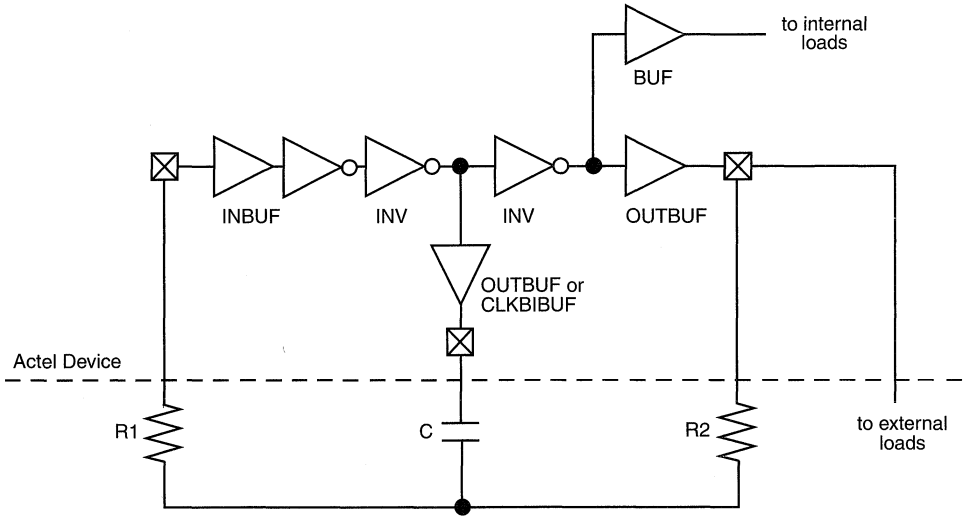


Figure 2 • RC Oscillator

Three-Stating ACT Device I/O Pins for Board Level Testing

Introduction

During board testing and debugging, it is frequently desired or necessary to place all device I/O pins into a three-state, high-impedance condition. This isolates the device from other devices that have common signal paths on a printed circuit board. The three-state condition also allows board testing for trace integrity or insertion damage to pins. It is usually more convenient to simultaneously three-state all I/O pins of a device with a built-in procedure rather than to create test vectors to force this condition by circuit function.

All Actel field programmable gate arrays (FPGAs) include a special operating mode to place all I/O pins in a temporary, three-state condition. Thus, each ACT™ device may be easily isolated from I/O signal paths of other devices on a circuit board, enabling a convenient board testing procedure. This capability, combined with Actionprobe® diagnostic tools, allows both single device and system board testing with a power and ease previously unavailable in programmable devices.

ACT 1 Three-State Procedure

Three-stating an ACT 1 device is easy using the unique debugging features of the ACT architecture. Three special pins on ACT devices are used for this function: MODE, SDI, and DCLK. To maintain the three-stating feature on ACT 1 devices, no user-defined output or bidirectional pins may be assigned to the SDI and DCLK locations. These pins should remain unassigned or should be defined as input-only locations. All other user-defined pins have no restrictions for their function; they may be assigned as input-only, output-only, or bidirectional pins. These pins can be temporarily three-stated for testing and debugging.

Figure 1 shows the sequence for three-stating an ACT 1 device. Seven data bits are clocked into the device using the SDI pin as data input and DCLK as the clock signal. The MODE pin distinguishes "test" mode from "normal" mode. The data sequence is {0001011}. After clocking the seventh bit, all user-defined pins are placed in a three-state condition until MODE is returned to logic low.

ACT 2 and ACT 3 Three-State Procedure

The same special purpose pins are used to three-state ACT 2 and ACT 3 devices: MODE, SDI, and DCLK. In contrast to the ACT 1 family, there are no restrictions on the assignment of user-defined functions to these pins. All user-defined pins have no restrictions for their function; they may be assigned as input-only, output-only, or bidirectional pins.

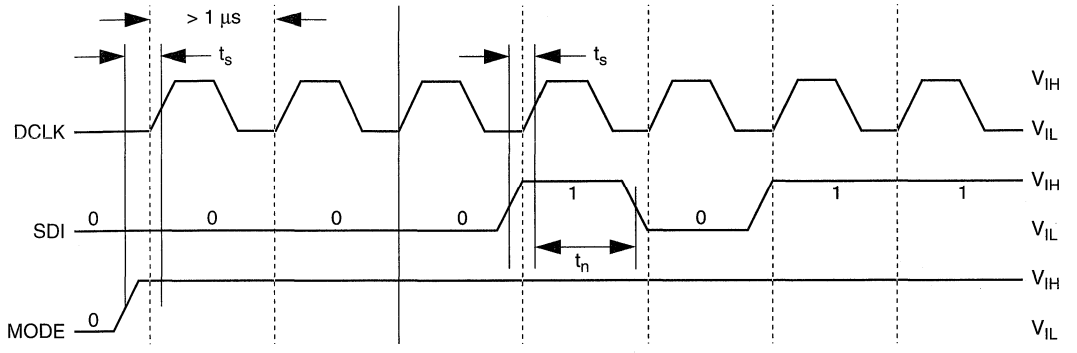
Thirty-two data bits are clocked into the device using the SDI pin as data input and DCLK as the clock signal. The MODE pin distinguishes "test" mode from "normal" mode. The data sequence for ACT 2 and ACT 3 devices is determined by the programmed state of the device. For unprogrammed, blank devices, the data sequence is

(LSB) 0000000000 0000000000 0000000111 10 (MSB)

For programmed devices, the data sequence is

(LSB) 0000000000 0000000000 0000000110 00 (MSB)

Load the data beginning with the LSB. After clocking the 32nd bit, all user-defined pins are placed in a three-state condition until MODE is returned to logic low. Use the same relative timing for the MODE, SDI, and DCLK inputs as shown in Figure 1.



Notes:

1. $0\ V \le V_{IL} \le 0.5\ V$; $3.0\ V \le V_{IH} \le VCC$
2. Test mode configuration is a low frequency (< 1.0 MHz)

Figure 1 • ACT 1 Device Three-State Timing Diagram

Using Actel Devices in Hot Socketing Applications

Hot socketing refers to the practice of inserting and removing a module from a system board while system power is active. Hot socketing is a requirement for systems for which turning off the machine would be unacceptably disruptive. Telecommunications switches and routers are two examples.

The biggest concern regarding hot socketing devices is potential latch up—a well-known cause of failure in CMOS devices such as FPGAs. When a subsystem is plugged into active hardware a low impedance path occurs between Vcc and ground. A large amount of current flows through this path and can temporarily make the device nonfunctional. Although latch up with Actel devices is not likely to occur, due to the robust internal protection diodes, there is a recommended technique to improve circuit integrity.

Actel recommends that the connector of hot socketing modules be modified to guarantee that Vcc and the Gnd connection be established prior to all other signals. This can be achieved by making the Vcc and Gnd contacts (or fingers) longer than all others. In this way, power will be applied to the FPGA prior to any logic signals, thereby preventing latch up.

The circuit shown in Figure 1 is an added precaution for hot socketing. This added circuit is recommended for FPGA input signals that are connected directly to the connector. It essentially duplicates the internal protection circuit with the input pad of the FPGA. The diodes prevent the input voltage from straying beyond one diode drop of Gnd or Vcc ,which is within maximum device specifications. The protection resistor is used to limit safely peak current flowing in the diodes.

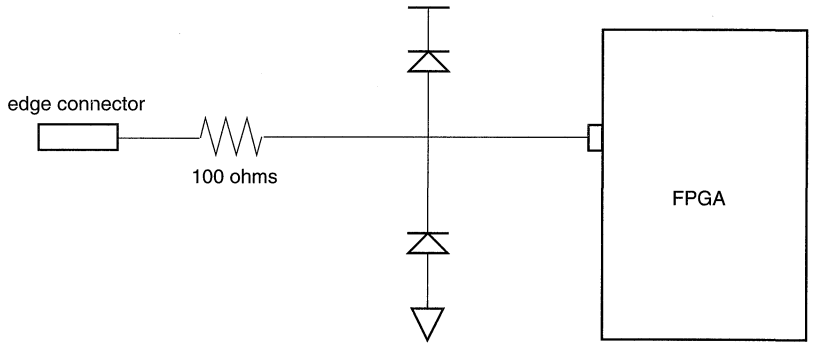


***Figure 1** • Added Circuit Recommended As Precuation for Hot Socketing*

Testing and Reliability

Section 4: Testing and Reliability

Testing and Programming Actel Field Programmable Gate Arrays (FPGAs)

Introduction

Testing has long been a struggle for users of masked gate arrays. To avoid board level, system level, or even possible field failures, the system designer must extend great effort in developing test vectors for gate array designs. Even after the vectors are developed, fault coverage for typical designs may be only about 70 percent, with 95 percent coverage about the best possible. With a 70 percent fault coverage, typical masked gate array designs are likely to have 2 to 5 percent devices defective[1].

In general, field programmable logic devices have allowed the user to avoid the need to develop test vectors. These devices allow the tests to be performed by the semiconductor vendor prior to programming. However, most one-time programmable logic devices have not yet achieved the functional quality levels of other semiconductor devices because they don't allow the chip manufacturer to access and test all internal gates. Early one-time programmable devices had poor test coverage, and users were often disappointed to see functional failure rates of more than 10 percent on parts that had passed programming. Over time, on-chip test circuits and testing techniques have greatly improved and now one-time programmable devices have functional defect rates in the range of 0.1 to 1 percent[2]. Although this failure rate is low for individual chips, putting 10 such chips on a single board can still mean a board failure rate of 5 to 10 percent.

Reprogrammable logic devices that use EPROM, EEPROM, or RAM technology however, have improved functional quality levels to nearly 100 percent. Since the semiconductor manufacturer can program these chips to any desired configuration, it is possible to test all internal gates. This can result in functional failure rates equivalent to most other semiconductor devices.

The Actel FPGA Product Family

There are three Actel FPGA product families. The ACT™ 1 family offers 1200 (A1010) and 2000 (A1020) gate products. The ACT 2 family consists of three products (A1225, A1240, and A1280) with 2500, 4000, and 8000 gate array equivalent gates. This family offers improved performance and number of I/O's compared to ACT 1. The ACT 3 family contains five products (A1415A, A1425A, A1440A, A1460A, and A14100A) with 1500, 2500, 4000, 6000, and 10,000 equivalent gates. The ACT 3 products offer the highest performance and number of I/Os of the three families.

Testability of Actel FPGAs

Although Actel's FPGA Family arrays use a one-time programmable technology, the device's unique architecture permits a degree of testability comparable to reprogrammable devices. Special test modes allow functional testing of unprogrammed devices at essentially 100 percent fault coverage. This testability is independent of the large number of equivalent gates in the A1010 (1200 gates) through the A14100 (10,000 gates). To show how this accomplished, we will first review the architecture of the Actel FPGAs and describe how they are programmed.

Architecture

The basic building block of all Actel FPGAs is the logic module. Each logic module is programmable and capable of implementing all two-input logic functions, most three-input functions, and many other functions up to eight inputs. With an architecture similar to a channeled gate array, logic modules are organized in rows and columns across the chip (Figure 1). Adjacent to each row of logic modules are routing channels. Horizontal routing channels are shown in the figure, but vertical channels also run through the logic modules. These are used to configure a logic module and connect inputs and outputs of logic modules together to implement a design. Surrounding the array of logic modules and routing channels are I/O buffers and test circuits. .

Within the routing channels are programmable antifuse (PLICE™) elements. The antifuse is normally open and is programmed to form an electrical connection between routing elements. An antifuse that connects a horizontal routing track to a vertical track is called a cross antifuse. An example of a logic module interconnection (or a net) is shown in Figure 2. Here the output from Module 3 is connected to a horizontal routing track by programming a cross antifuse. Another cross antifuse is programmed to connect an input to Module 4. In a similar manner, the output of Module 3 is connected to the input of Module 2. Notice that not all horizontal tracks are continuous across the chip. Often tracks are broken into a series of smaller tracks called "segments." Segments are useful because it is often desirable to connect logic modules that are close to each other, and using a full horizontal track would waste routing resources and slow down circuit performance. Sometimes, however, it is necessary to connect two segments together to form a longer segment. This can be done by programming a special type of

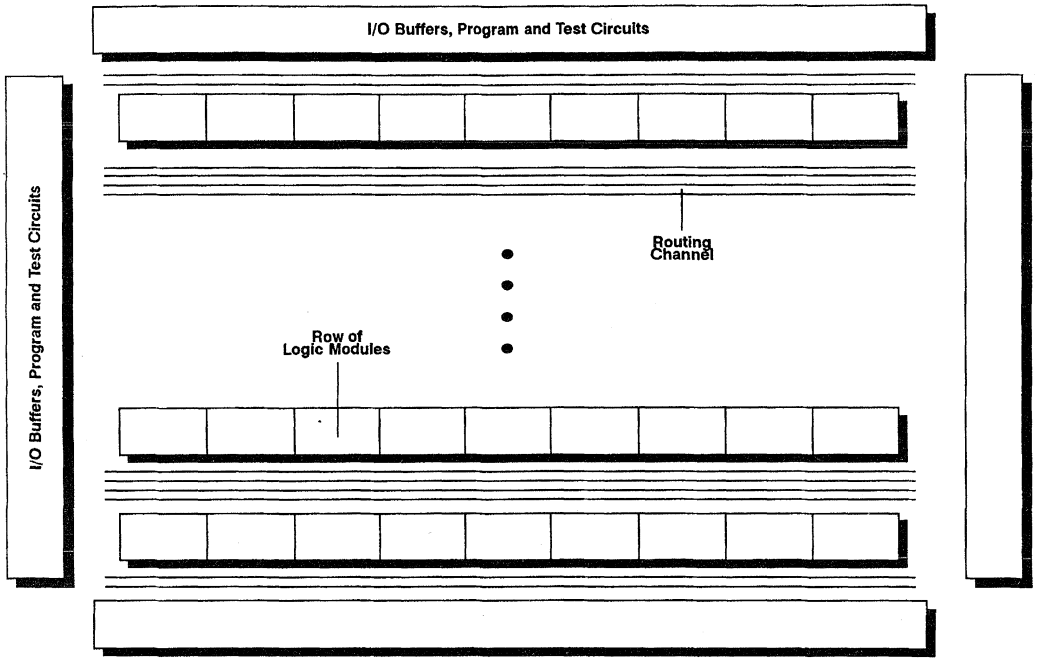


Figure 1 • Architecture

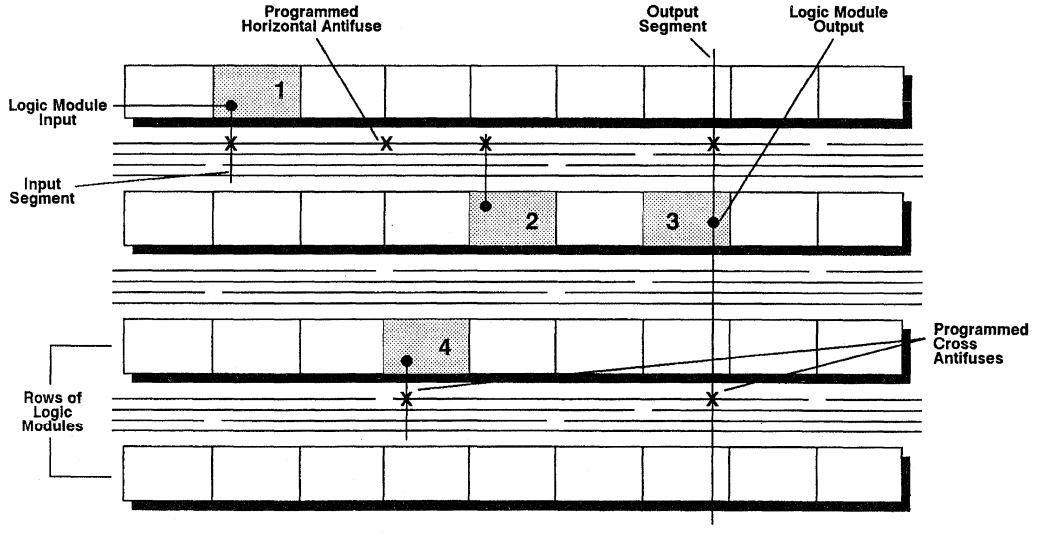


Figure 2 • Routing

antifuse referred to as a horizontal antifuse. As an example, the output of Module 3 is also connected to the input of Module 1 by programming two cross antifuses and one horizontal antifuse. Vertical antifuses are used to connect two vertical segments (not shown)

A more detailed example of the Actel FPGA architecture is shown in Figure 3. Six logic modules (two rows, three columns) are shown. Between the two rows are six horizontal tracks. Down each column are five vertical tracks. Note that the products actually have 25 to 36 horizontal and 13 to 15 vertical tracks. The circles at the intersection of vertical and horizontal tracks represent cross antifuses. There are also circles at certain points on the horizontal tracks that are horizontal antifuses. No vertical antifuses are shown. Notice the transistors that connect both horizontal and vertical tracks. By turning on selected transistors, various horizontal or vertical tracks can be connected together even though an antifuse has not been programmed. This ability to connect tracks in unprogrammed devices is used extensively during antifuse programming and is one of the key elements responsible for the excellent testability of the Actel FPGAs.

Logic configuration of modules is interesting because there are no dedicated antifuses in the module to accomplish this. Instead, the inputs (and outputs) of logic modules extend into the cross antifuse array. Each logic module has eight to ten inputs and one output. By programming appropriate antifuses, an input can be connected to a dedicated horizontal ground line, a Vcc line, or a horizontal routing track. The logic module implements a particular logic function by tying appropriate unused inputs to ground or Vcc

Programming

The following discussions about programming and testing modes are specific to the ACT 1 family of FPGAs. However, basic concepts also apply to the ACT 2 and ACT 3 families.

An antifuse is programmed by applying a sufficiently high voltage across it. This voltage is referred to as Vpp. To access an antifuse deep inside the chip, it is necessary to create electrical paths from Vpp and ground to the antifuse. This is done by turning on the appropriate horizontal and vertical pass transistors (in normal chip operation, these transistors are always off). The transistors are turned on by applying Vpp

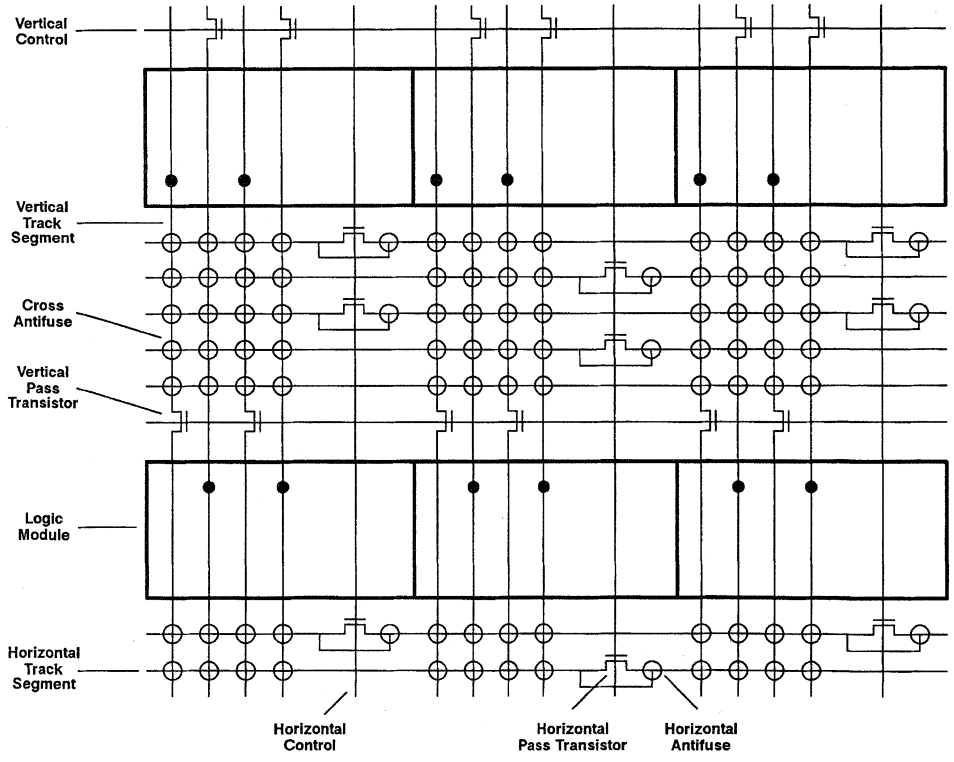


Figure 3 • Programmable Interconnect

to their gates. In Figure 4, we see an example of programming a typical cross antifuse. Vpp is applied to a vertical track at the top of the chip and ground is applied to a horizontal track on the right side. The design of the A1010/A1020 actually allows Vpp or ground to be applied from the top, bottom, left, or right as is appropriate to best access a particular antifuse. Notice that Vpp is also applied to the gates of the horizontal and vertical pass transistors on the tracks accessing the cross antifuse. The circled cross antifuse now has Vpp applied to it on one side and ground on the other. This voltage breaks down the antifuse's dielectric and creates an electrical connection between the horizontal and vertical routing tracks.

There is one other important consideration when programming an antifuse. Notice that the cross antifuses in the same vertical track as the antifuse to be programmed also have Vpp applied to them on one side. This is true until the track is broken by a vertical pass transistor below it that is turned off. However, the potential on the other side of the antifuses is not being driven. Should this potential be at ground, the other cross antifuses on the vertical segment could be accidentally programmed.

The same logic applies to other antifuses on the same horizontal track. Here, one side of the antifuse is being driven to ground and if the other side were at Vpp, extra antifuses could program.This problem is solved by first applying what is referred to as a "precharge cycle." During the precharge cycle, all horizontal and vertical tracks are charged to Vpp/2. As a result, there is no voltage across the antifuses. The appropriate vertical track is then driven to Vpp and a horizontal track to ground (Figure 5). At this point, other antifuses on the vertical track have a potential of Vpp/2 across them (Vpp on one side and Vpp/2 on the other). This Vpp/2 voltage is not sufficient to program the antifuses. Other antifuses on the same horizontal track also have Vpp/2 across them (Vpp/2 on one side and ground on the other). Most other antifuses in the chip still have Vpp/2 on both sides and will not program.

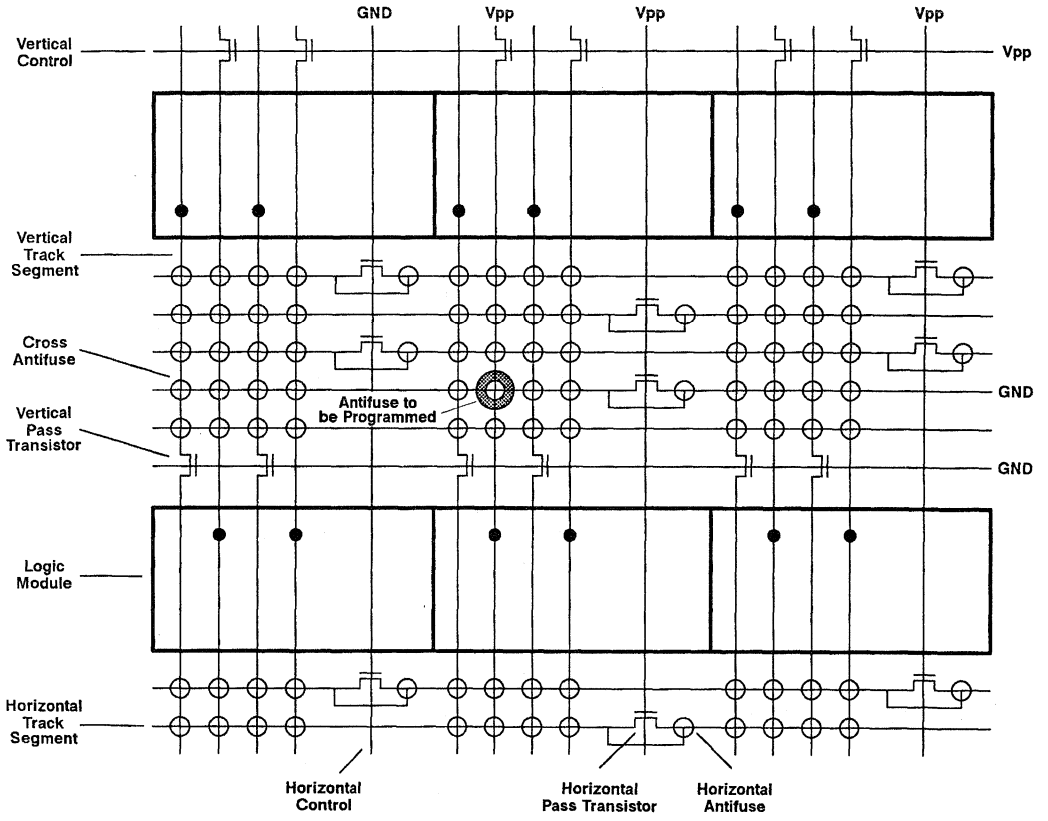


Figure 4 • Programmable Interconnect

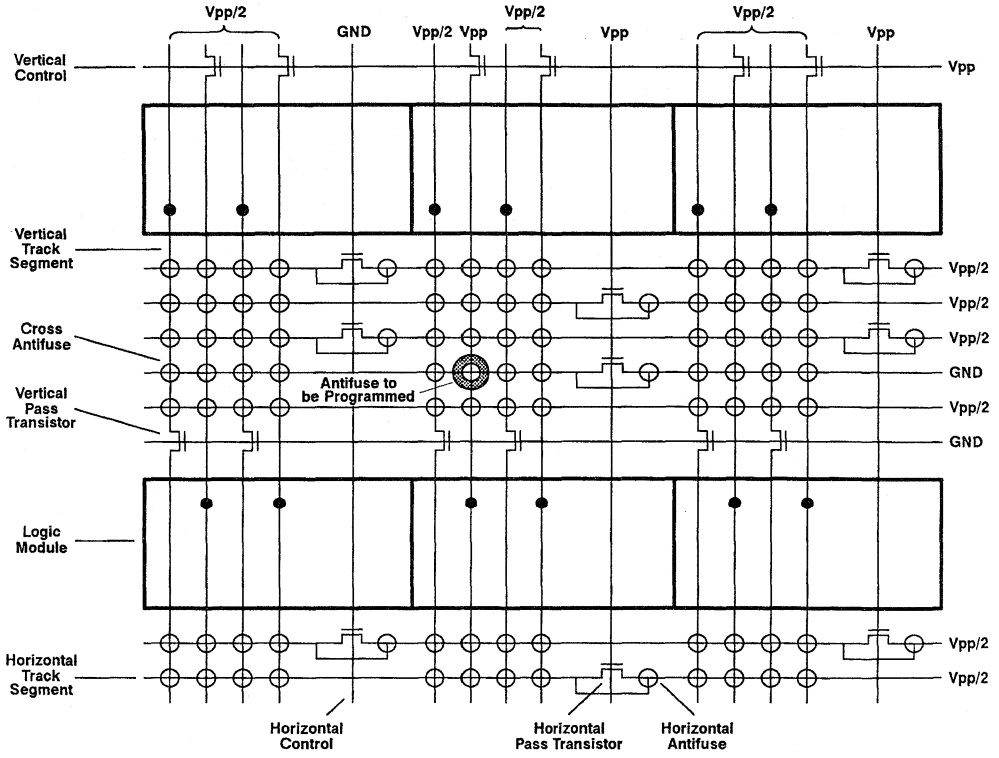


Figure 5 • Programmable Interconnect

Programming Algorithm

In concept, the Actel FPGAs are programmed in a manner very similar to many other programmable logic devices as well as memories such as EPROMs. The programming algorithm consists of the following steps:

1. An addressing sequence to select the antifuse to be programmed.
2. A programming sequence where Vpp is applied in pulses until the antifuse programs.
3. A soak or "overprogram" step to ensure uniform, low antifuse resistance.
4. A verify step to make sure the antifuse was properly programmed.

Unlike a memory where an antifuse is addressed by applying a parallel address, the FPGAs are addressed in a serial manner by using the special DCLK (Data Clock) and SDI (Serial Data In) pins. There is a large shift register that travels around the periphery of the chip. Bits in this shift register can be used to drive tracks to ground, Vcc, Vpp, or float. It is also possible to sense the level on the track (high or low) and load this information into the shift register. By shifting in the correct address, any antifuse can be selected for programming. The shift register also plays a key role in testing the chip. This will be discussed later.

The programming sequence starts with the precharge pulse where Vpp/2 is applied to the Vpp pin. This is followed by a programming pulse where Vpp is applied to the pin. Following the program pulse, the voltage on the Vpp pin is returned to a nominal value (about 6 V). See Figure 6 for a typical Vpp waveform. The precharge/program pulse sequence is repeated until either the selected antifuse programs or a maximum number of pulses is exceeded (in which case the antifuse is considered nonprogrammable and the device is rejected).

Confirmation that an antifuse has programmed is determined by monitoring the current on the Vpp pin. This current is very low (typically < 10 µa) until an antifuse programs. Once an antifuse is programmed, an electrical connection is made between Vpp and ground in which case currents in the range of 3 to 15 mA may be observed on Vpp. Once this current is

observed, the antifuse is considered programmed and enters the soak or "overprogram" cycle. Here, extra pulses are applied to the antifuse to achieve minimum antifuse resistance.

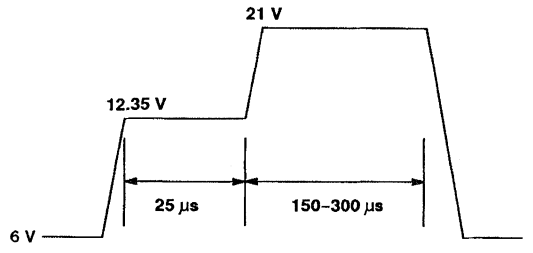


Figure 6 • Vpp Waveform

ACT 1 Programming Algorithm

Current Parameters		
V Program	=	21 V
V Precharge	=	12.35 V
V Verify	=	6.0 V
t Program	=	150–300 µs
t Precharge	=	25 µs
I Threshold	=	~2.5 mA (to detect programmed antifuse)
I Max	=	15 mA (clamp current)
# Soak	=	30–800 pulses
Maxpulses	=	60,000

Test Modes of Actel FPGAs

The unique architecture described above allows outstanding testability of unprogrammed devices at the factory. Details of the various test modes available are as follows:

1. The shift register circling the periphery of the chip can be both downloaded and uploaded. This allows the use of various test patterns to ensure that the shift register is fully functional.
2. All vertical and horizontal tracks can be tested for continuity and shorts. There are several ways to implement these tests. One way of doing continuity testing is to precharge the array, turn on all vertical or horizontal pass transistors on a track, drive the track low from one side of the chip, and read a low on the other side. Shorts can be detected by driving every other track low after precharge and reading back on the other side. Note that these tests also confirm that the vertical and horizontal pass transistors will turn on.
3. It is important for programming to make sure that all tracks can hold the precharge level. By charging a track, floating it, and waiting a predetermined amount of time, the track can be read back and confirmed to be still high.
4. Leakage of vertical and horizontal pass transistors can be tested for by driving one side of a track to a voltage via the Vpp pin and grounding the other side. All pass transistors except the one being tested are turned on. If excess current is detected on the Vpp pin, the pass transistor is considered defective.
5. There are one or two dedicated clock buffers that travel across all horizontal channels. This buffer can be tested by driving with the clock pin and reading for the proper levels at the sides of the array.
6. There are two special pins referred to as Probe A and Probe B (Actionprobes™). By entering a test mode, the shift register can be made to address the internal output of any logic module. This output is then directed to one of two dedicated vertical tracks, which in turn can be observed externally on the Probe A or B pins. This ability to observe internal signals (even on unprogrammed parts) allows Actel to perform a large number of functional tests. The first such test is the Input Buffer Test. Input buffers on all I/O pins can be tested for functionality by driving at the input pad and reading the internal I/O output node through the probe pins.
7. Test modes exist to drive all output buffers low, high, or tristate. This allows testing of Vol, Voh, Iol, Ioh, and leakage on all I/Os.
8. One of the key tests is the ability to functionally test all internal logic modules. By turning on various vertical pass transistors and driving from the top or bottom of the chip, any of the eight to ten module inputs can be forced to a high or low. The output of the module can then be read through the Actionprobe pins. The logic module test allows 100 percent fault coverage of each module. In addition, the architecture allows modules to be tested in parallel for reduced test time.
9. Actel FPGAs have one or two dedicated columns on the chip that are transparent to the user and used by the factory for speed selection. These columns are referred to as the "Binning Circuit." Modules in the columns are connected to each other by programming antifuses. The speed of the completed test circuit can then be tested. The Binning Circuit allows the separation of units into different speed categories. It also allows the speed distribution within each category to be minimized.

10. There are several tests to confirm that the programming circuitry is working. The first such test is a basic junction stress/leakage test. The program mode is enabled and Vpp voltage plus a guardband is applied to the Vpp pin. All vertical and horizontal tracks are driven to Vpp; thus, no voltage is applied across the antifuses. The Ipp current is then measured. If it exceeds its normal value, the device is rejected.
11. There is a test to ensure that all antifuses are not programmed. This is referred to as the Antifuse Shorts Test (or Blank Test). The array is precharged and then the vertical tracks are driven to ground. The horizontal tracks are then read to confirm that they are still high (a programmed or leaky antifuse would drive a horizontal track low). The test is repeated by driving horizontal tracks low and reading vertical tracks.
12. The functionality of the programming circuitry can be verified by programming various extra antifuses on the chip that are transparent to the user. Some of these antifuses were already described earlier when the Binning Circuit was discussed. Actel FPGAs also have a Silicon Signature™. In the ACT 1 family, the Silicon Signature consists of four words of data, each word 23 bits in length. The first word is hard wired (no antifuses) and contains a manufacturer ID number as well as a device ID number. These numbers can be read by a programmer and the proper programming algorithm would be automatically selected. The other words contain antifuses and are programmable. Actel is currently using bits in these words to store information such as the chip's run number and wafer number. Thus, each Actel FPGA has traceability down to the wafer level. By programming this information, the functionality of the programming circuitry is also tested. Actel software also allows the user to program a design ID and checksum into the Silicon Signature. By later reading this back, the user can verify that the chip is correctly programmed to a given design.
13. The most important antifuse test is the stress test. When this test is enabled, a voltage applied to the Vpp pin can be applied across all antifuses on the chip (the other side is grounded). The voltage applied is the precharge voltage plus a significant guardband. After the voltage is applied, the Antifuse Shorts Test is again used to make sure no antifuses have programmed. The antifuse stress test is effective at catching antifuse defects. Becasue the reliability of the antifuse is much more voltage dependent than it is temperature dependent, this test is also an effective antifuse infant mortality screen. See the Actel Reliability Report for details.

Burn-In of Actel FPGAs

As mentioned above, Actel has found that antifuse infant mortality failures can be effectively screened out during electrical testing, and it is thus unnecessary to do any kind of burn-in for standard commercial production units to screen out antifuse infant mortality failures. However, burn-in is still an effective screen for standard CMOS infant mortality failure mechanisms, and it is required for all military 883D products. MIL-883D Method 1005 allows several types of burn-in screens. These can be divided into two categories: steady-state (static) and dynamic. Static burn-in applies DC voltage levels to the pins of the device under test. The device may or may not be powered up. Dynamic burn-in applies AC signals to device inputs. These signals are selected so that the device receives internal and external stresses similar to what it may see in a typical application.

Static burn-in is by far the simplest to implement. By choosing appropriate biasing conditions and load resistors, it is possible to design a single burn-in circuit that could be used for both unprogrammed and programmed devices. It would not matter what pattern is programmed into the device. Static burn-in can be an effective screen for some types of failure modes, particularly those that may happen at device inputs or outputs (such as screening for mobile ionic contamination). It is not, however, very effective at stressing internal device circuits. Many internal nodes may be biased at ground without receiving any voltage or current stress. Signal lines will not toggle, and it may not be possible to screen failure modes such as metal electromigration.

A properly designed dynamic burn-in can effectively stress inputs, outputs, and internal circuits. However, dynamic burn-in of ASIC products can be very expensive because customer-specific burn-in circuits and burn-in boards must be designed and built to properly stress each design implemented in the ASIC. This results in large NRE costs and long lead times to design and build these boards. From the standpoint of burn-in, a programmed FPGA is essentially the same as a mask-programmed ASIC, and it would require similar custom burn-in circuits to do a dynamic burn-in. However, Actel has been able to use the testability features of its FPGA products to allow effective dynamic burn-in of unprogrammed devices. This dynamic burn-in allows us to stress circuits in a way that static burn-in would be unable to duplicate.

During burn-in of unprogrammed units, test commands are serially shifted into each device using the SDI pin and

clocked using the DCLK pin. There are three test modes shifted into each device. The first test stresses each cross antifuse with a voltage of Vpp–2 V (Vpp is normally set at 7.5 V so that each antifuse gets 5.5 V across it). This voltage is applied to all vertical tracks while the horizontal tracks are grounded. Once enabled, the stress mode is held for 10 ms.

The second test mode is identical to the first except that the horizontal tracks are driven to Vpp–2 V while the vertical tracks are grounded. Note that both of these modes are similar to the antifuse stress test described earlier (although the stress voltage is lower during burn-in). Not only do these tests stress the antifuses, but they also toggle all routing tracks in the chip to Vpp–2 V and ground. All input and output tracks to the logic modules are also toggled.

The third test drives several I/O pins on the chip to a low state. Prior to this, they are at high impedance state and held at Vcc through pull-up resistors. This test confirms that the burn-in is being properly implemented by looking at these I/O pins to see if they display the proper waveform. It also passes current through each I/O as it toggles low.

Although the chip is unprogrammed, these tests allow us to apply stresses to the inputs, outputs, and internal nodes that are similar to what a programmed device may see in normal operation. Once burn-in is completed, post burn-in testing as specified by MIL-883D is performed (including PDA) to ensure that fully compliant devices are shipped to the customer.

Conclusion

The description of the Actel FPGA architecture and the numerous test modes attest to the outstanding testability of these devices. All internal logic gates can be tested without programming antifuses other than the few for the Binning Circuit and Silicon Signature. Because Actel FPGAs are one-time programmable, the only item that is not fully tested at the factory is the programmability of all the individual antifuses. However, this is done on the programmer while the units are being programmed. Being able to test all internal gates allows Actel to achieve functional yields superior to other one-time programmable devices and equivalent to reprogrammable parts.

References:

1. Henshaw, "User Requirements for Fault Coverage," Wescon Proceedings, 1990, P. 179
2. AMD PAL Device Data Book, 1988, P. 3–106

ACT Family Reliability Report

Actel's field programmable gate arrays (FPGAs) are currently available in three product families—ACT™ 1, ACT 2, and ACT 3. The ACT 1 family consists of the A1010 and A1020, which are 1200- and 2000-gate FPGAs respectively. The ACT 2 family includes the A1225, A1240, and A1280 FPGAs, which offer 2500, 4000, and 8000 gates respectively. The ACT 3 family is the newest family of FPGAs and contains products ranging from 1500 to 10,000 gates (A1415, A1425, A1440, A1460, and A14100).

The programming element for all Actel FPGAs is an Actel-invented PLICE® (Programmable Low-Impedance Circuit Element) antifuse. An antifuse is a device that is normally open and in which an electrical connection is established by applying programming voltage. Although Actel FPGAs are one-time programmable devices, their unique architecture includes complete functional testability.

ACT 1 products were originally manufactured using 2 µm design rules (A1010/A1020). Later they were linearly shrunk in size by 20% to 1.2 µm (A1010A/A1020A). Currently, the family is being manufactured using to 1.0 µm design rules (A1010B/A1020B). ACT 2 products were first introduced with 1.2 µm design rules (A1225/A1240/A1280) and as a result of a 20% linear shrink are now available in 1.0 µm versions (A1225A/A1240A/A1280A). ACT 3 products are manufactured using a 0.8 µm process. In all cases, the technology is a standard double metal, twin well, CMOS process in which three additional masking steps have been added to implement the PLICE antifuse. The main process parameters are shown in Table 1. Because Actel FPGAs are manufactured with a conventional CMOS process, normal CMOS failure modes are observed. However, the addition of the antifuse adds another structure that could affect the device's reliability.

Table 1 • ACT 1 and ACT 2 Process Description

Dimensions	2.0 µm Process		1.2 µm Process		1.0 µm Process		0.8 µm Process	
	Width (µm)	Space (µm)	Width (µm)	Space (µm)	Width (µm)	Space (µm)	Width (µm)	Space (µm)
N +	4.0	2.0	3.2	1.6	2.3	1.2	1.8	1.4
p +	4.0	2.0	3.2	1.6	2.3	1.2	1.8	1.4
Cell PolySilicon	2.0	3.6	2.1	2.4	1.5	0.9	1.2	1.2
Gate PolySilicon	1.6	2.4	1.6	1.6	1.2	1.2	1.0	1.2
Metal I	4.0	2.0	3.2	1.6	2.2	1.2	1.6	1.2
Metal II	4.8	2.2	3.8	1.8	2.1	1.6	1.6	1.4
Contact	1.8 x 1.8	2.0	1.2 x 1.2	1.8	1.0 x 1.0	1.0	1.0 x 1.0	1.2
Via	2.0 x 2.0	2.0	1.3 x 1.3	1.9	1.0 x 1.0	1.0	1.0 x 1.0	1.2
Thickness	**2.0 µm Process**		**1.2 µm Process**		**1.0 µm Process**		**0.8 µm Process**	
Normal Gate Oxide	25 nm		25 nm		20 nm		18 nm	
High Voltage Gate Oxide	40 nm		40 nm		32.5 nm		35 nm	
Cell PolySilicon	450 nm		450 nm		400 nm		400 nm	
Gate PolySilicon	450 nm		450 nm		400 nm		400 nm	
Metal I	900 nm		900 nm		720 nm		750 nm	
Metal II	1000 nm		1000 nm		1050 nm		1050 nm	
Passivation	1100 nm		1100 nm		1100 nm		1100 nm	
Compositions								
Metal I	Al–SI (1%)–Cu (0.5%)		Al–SI (1%)–Cu (0.5%)		Ti/TiN/Ai–Si (1%)–Cu (0.5%)		Ti/TiN/Ai–Si (1%)–Cu (0.5%)	
Metal II	Al–SI (1%)–Cu (0.5%)		Al–SI (1%)–Cu (0.5%)		Ti/Al–Si (1%)–Cu (0.5%)		Ti/Al–Si (1%)–Cu (0.5%)	
Passivation	300 nm SiO_2, 800 nm SiN		300 nm SiO_2, 800 nm SiN		300 nm SiO_2, 800 nm SiN		300 nm SiO_2, 800 nm SiN	

To quantify the reliability of the antifuse, Actel has completed numerous studies. These studies lead to the conclusion that the time-to-failure of the antifuse is substantially more than 40 years under normal operating conditions and that the combined contribution of all antifuses to the gate array product's hard failure rate is less than 10 FITs (Failures-in-Time or 0.001% failures per 1000 hours).

The PLICE Antifuse

The antifuse is a vertical, two-terminal structure. It consists of a polysilicon layer on top, N+ doped silicon on the bottom, and an ONO (oxide-nitride-oxide) dielectric layer in between. A Scanning Electron Microscope (SEM) cross section of the antifuse is shown in Figure 1. The size of the antifuse is 3.2 μm^2, 1.4 μm^2, 1.0 μm^2, and 0.8 μm^2 for the 2.0 µm, 1.2 µm, 1.0 µm, and 0.8 µm processes respectively. This small size, along with a low programmed on-resistance (typically 150–200 ohms in speed sensative paths) makes the PLICE antifuse an attractive alternative to EPROM, EEPROM, or RAM as a programming element in a large programmable gate array. In the unprogrammed state, the resistance of the antifuse is more than 100 megohms. Actel FPGAs may contain anywhere from 112,000 antifuses (A1010) to 940,000 (A14100). However, typical applications that utilize 85% of the available gates require you to program only 2% to 3% of the available antifuses.

The Unprogrammed Antifuse

To evaluate antifuse reliability, Actel has developed models and collected data for both unprogrammed and programmed antifuses.[1,2] We'll consider the unprogrammed antifuse first. Since the antifuse is a dielectric sandwiched between polysilicon and silicon, the model to evaluate its reliability in the unprogrammed condition is the same as that used for MOS transistor gate oxides.[3] The parameter to evaluate is the dielectric time-to-breakdown (t_{bd}). This parameter is a function of the electric field across the dielectric, as well as temperature. It has the following relationship.[3]

$$t_{bd} = t_0 * \exp(G/E)(1)$$

Where t_{bd} is the time-to-breakdown in seconds, t_0 is a constant in seconds, E is the electric field in Mv/cm, and G is the field acceleration factor in Mv/cm. (G is temperature dependent and will be discussed later.)

By taking the log of both sides of equation 1, we have:

$$\ln(t_{bd}) = G * (1/E) + 1n(t_0)(2)$$

From experimental data, we can plot the log of the time-to-breakdown of the antifuse at various temperatures versus the reciprocal of the electric field across it and derive G from the slope and t_0 from the y-intercept. Actel has done this on single antifuses, large antifuse capacitors, test arrays of 28,000 antifuses, and actual FPGA products. These antifuse areas range from 3.2 μm^2 to 0.35 mm^2. Figure 2 shows plots of data collected on antifuses of different sizes.

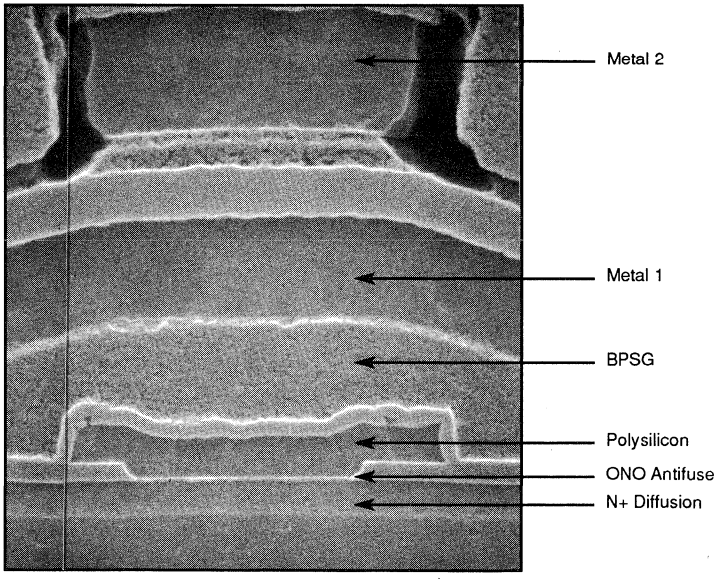


Figure 1 • Antifuse SEM Cross Section

There is some discussion in the literature regarding whether time-to-breakdown depends on E or 1/E. To verify the validity of equation 2, we conducted the following experiment. Large 200 μm by 200 μm (0.04 mm^2) area capacitors were packaged and then stressed at more than 11 different voltages. Capacitors with thicknesses ranging from a low of 8.0 nm to a high of 9.5 nm were chosen from two different wafer runs. A total of 610 capacitors were used in the experiment. The test splits and sample sizes are summarized in Table 2. The distribution of time-to-breakdown of the dielectric at each voltage is shown in Figure 3. In Figure 4, the median of the cumulative failure percentage rates (t_{50}) from Figure 3 is plotted versus 1/E. In Figure 5 the median failure percentage is plotted versus E. By comparing the two figures, the validity of the 1/E model is clearly established. A more detailed statistical verification of the 1/E model for the ONO antifuse is given by S. Chiang, etal.[2]

***Table 2** • Field Accelerated Test Data for Two Lots with Thicknesses Ranging from 8 nm to 9.5 nm. (The test was done on a 0.04 mm^2 area capacitor.)*

Lot A					Lot B				
Voltage (V)	Tox (nm)	E-Field (MV/cm)	# of Capacitors	t_{50} (sec)	Voltage (V)	Tox (nm)	E-Field (MV/cm)	# of Capacitors	t_{50} (sec)
13.5	8.3	16.2	22	4.2 e-3	14.0	8.7	15.9	25	9.8 e-3
12.5	8.3	15.1	22	3.7 e-2	13.0	8.7	14.9	25	5.0 e-2
12.0	8.3	14.4	22	1.5 e-1	12.5	8.7	14.3	25	2.4 e-1
11.5	8.3	13.8	22	8.6 e-1	12.0	8.7	13.7	25	1.3 e0
11.0	8.4	13.1	22	4.7 e0	11.4	8.7	13.1	25	9.0 e0
10.5	8.4	12.5	9	5.8 e1	11.2	8.7	13.1	45	8.0 e1
10.0	8.3	12.0	6	3.2 e2	10.8	8.7	12.5	45	3.52 e2
9.5	8.3	11.4	6	2.5 e3	10.2	9.0	12.0	45	2.88 e3
9.0	8.3	10.7	36	2.5 e5	9.7	9.0	11.3	45	2.88 e3
8.5	8.3	10.2	15	2.3 35	9.0	8.7	10.3	32	3.35 e5
8.0	8.3	9.6	59	1.5 e6	9.0	9.3	9.7	32	2.22 e6
Subtotal of tested capacitors:			241					369	
Total of tested capacitors:								610	

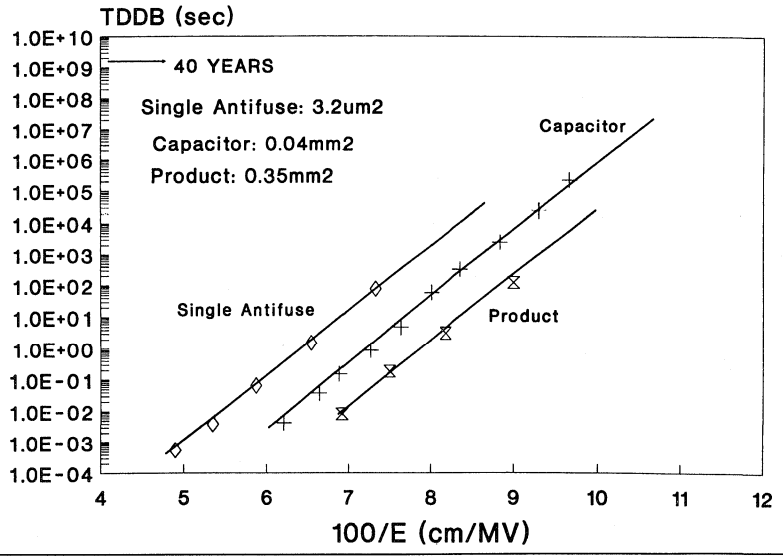


***Figure 2** • Field Accelerated Test*

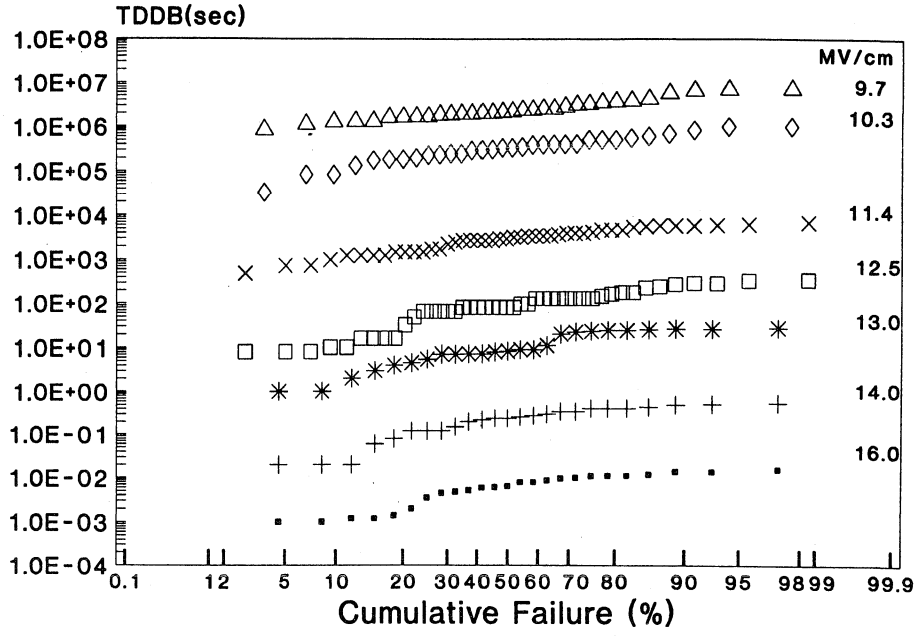


Figure 3 • TDDB Distribution

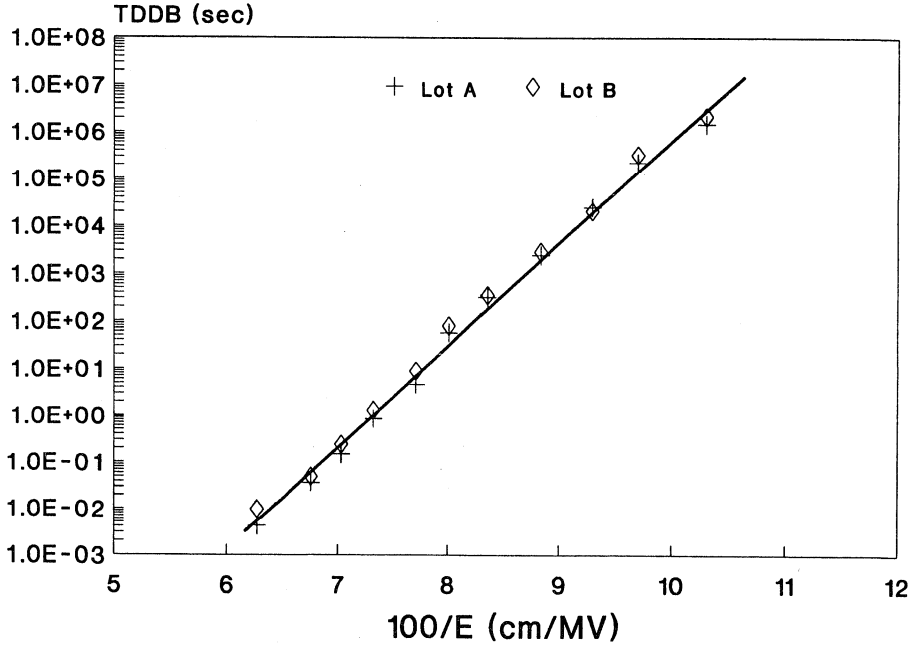


Figure 4 • ONO Reliability (1/E Model)

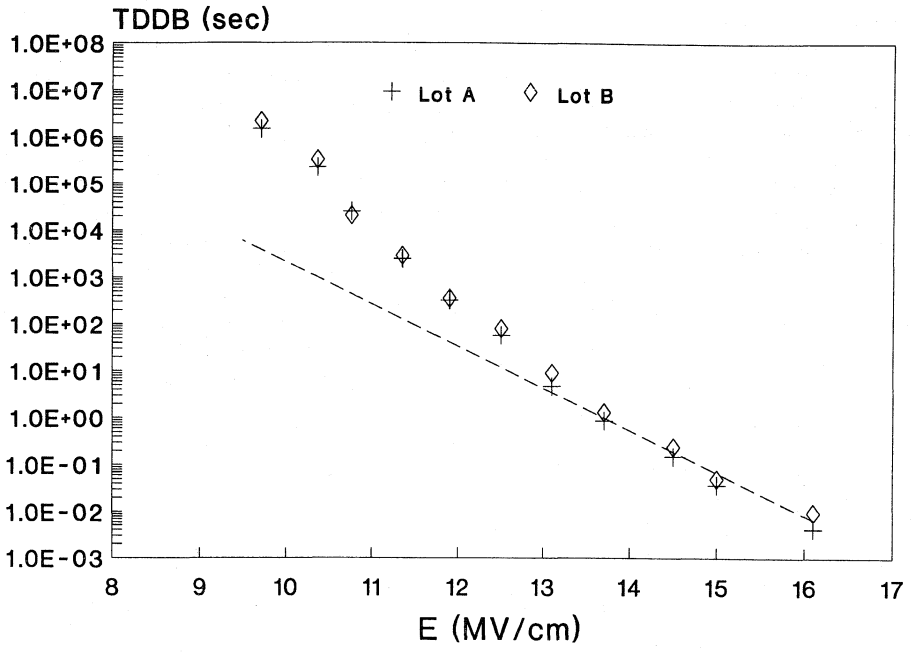


Figure 5 • ONO Reliability (E Model)

To quantify the temperature dependence of the time-to-breakdown, we use the Arrhenius equation to determine the semiconductor failure rate of a given process (failure mode) over temperature:

$$R = R_0 * \exp(E_a/kT) \quad (3)$$

where R is the failure rate, R_0 is a constant for a particular process, T is the absolute temperature in degrees Kelvin, k is Boltzmann's constant (8.62×10^{-5} eV/°K) and E_a is the activation energy for the process in electron volts. To determine the acceleration factor for a given failure mode at temperature T_2 as compared with temperature T_1 we use equation 3 to derive:

$$A(T_1, T_2) = \exp[(E_a/k) * \{(1/T_1)-(1/T_2)\}] \quad (4)$$

where A is the acceleration factor.

In Figure 6 we plot t_{50} at different temperatures and electric fields. This plot shows that time-to-breakdown is dependent on temperature as well as electric field. For a given time-to-breakdown of a dielectric, the expression,

$$E_a = k * d\ln(tbd)/d(1/T) \quad (5)$$

gives us the activation energy[2]. The slope in Figure 6 represents the activation energy E_a. E_a also shows a linear correlation with 1/E as shown in Figure 7. The field acceleration factor, G, is also temperature dependent, that is,

$$G(T) = G(298) * [1 + \delta/k * \{1/T - 1/298\}] \quad (6)$$

where G(298) is the field acceleration factor at room temperature (25°C = 298°K) and δ (in eV) characterizes the temperature dependence of G.

By combining equations 1, 5, and 6, E_a can be related to G(T) by:

$$E_a = G(298) * \delta/E - E_b \quad (7)$$

where δ and E_b are treated as fitting parameters between E_a and G.

From the data shown in Figures 3, 4, 6, and 7, we can obtain values of G(298), δ, E_b, and E_a regardless of antifuse area. Typical values of G(298), δ, E_b, and E_a at 5.5 V are 480 MV/cm, 0.014 eV, 0.43 eV, and 1×10^{-16}. By combining equations 1, 6, and 7, we obtain an overall equation for the time-to-breakdown for a given temperature and E field:

$$t_{bd} = t_0 * \exp\{(G(298)/E)\,[1+(\delta/k) * (1/T-1/298)] - (E_b/k) * (1/T-1/298)\} \quad (8)$$

By applying the values for the constants as defined above, the time-to-breakdown for the antifuse dielectric can be derived for a given temperature and electric field. In Table 3, we have used equation 8 to solve for the acceleration factors associated with powering up a device at high voltage or temperature or both. Then we compare the failure rate with more typical voltages or temperatures. We can see the effect of temperature by comparing 125°C at 5.5 V with 55°C at 5.5 V. The Actel model (equation 8) gives us an acceleration

factor of 55.3, or 6.3 equivalent years for a 1000 hour burn-in at 125°C. Note that this acceleration factor of 55.3 is close to the value of 41.8 derived from the Arrhenius equation (equation 4) using an activation energy of 0.6eV and the same temperatures. We use 0.6 eV (and 0.9 eV) as the activation energy of a general semiconductor failure mode when calculating failure rates based on high temperature operating life (HTOL) later in this report.

We can also see from Table 3 that a small change in voltage is a much more effective reliability screen than is a change in temperature. For example, if we compare 25°C at 5.75 V to 25°C at 5.25 V, we see that a change of just a half volt yields an acceleration factor of 1092.6, or 124.7 equivalent years per 1000 hours at 5.75 V. This strong dependence on voltage allows Actel to screen antifuse infant mortality failures during normal wafer sort testing simply by applying a higher than normal voltage across all antifuses. Because antifuse infant mortality failures can be detected and effectively screened, Actel FPGAs have as high a level of reliability as standard CMOS-processed products.

To establish that the antifuse contributes less than 10 FITs (at 5.5 V, 125°C) to the overall product reliability, Actel has calculated the product failure rates due to the antifuse using three different techniques. In the first case, we evaluated the t_{bd} distribution of 125 A1010 units in which the antifuses received an 11 V stress. Using E_a = 0.9 eV and extrapolating to 5.5 V, we found that the 1% antifuse failure lifetime at 5.5 V is well over 40 years and less than 10 FITs.

The second method of determining product reliability was to look at the results of production wafer sorts. As mentioned earlier, all antifuses receive a high voltage stress at wafer sort to screen out infant mortality failures. Specifically, all antifuses receive the equivalent of two 10 V stresses for one second each. The first stress is to screen out clearly defective antifuses. The second stress is to catch weaker antifuses that could cause problems with product programming yield or infant mortality failures. Actual failure rates observed on the A1010 over ten runs for these two stresses (FS-1 and FS-2) demonstrate average yield loss at the second stress screen of less than 0.3%. By extrapolating this yield loss to a normal 5.5 V operating voltage, we thus conclude that the contribution of the antifuse to the product's lifetime is less than 10 FITs.

The third technique of determining the product's antifuse failure rate is by performing an accelerated burn-in of Actel FPGA products. The acceleration is accomplished by using both higher voltage (5.75 V to 6.0 V) and higher temperatures (125°C to 150°C). Units are programmed to a specific design and exercised in a manner similar to what may occur in a real application. For a detailed description of the test and the results, see "High Temperature Operating Life" later in this report. The results indicate that the antifuse contributes less than 10 FITs to the product's overall failure rate; it is thus an insignificant factor in a product lifetime of 40 years at 5.5 V and 125°C.

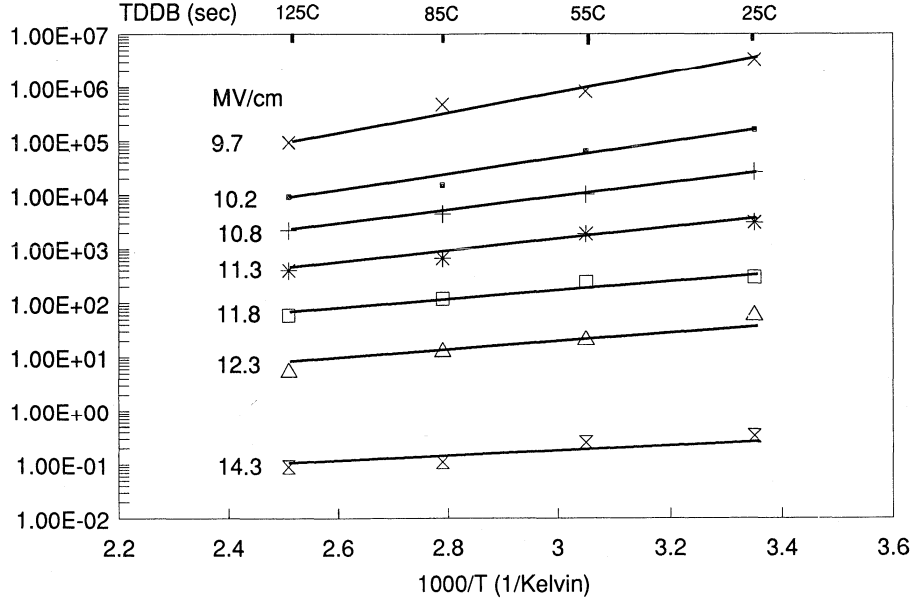


***Figure 6** • Dependence of E-Field Acceleration on Temperature*

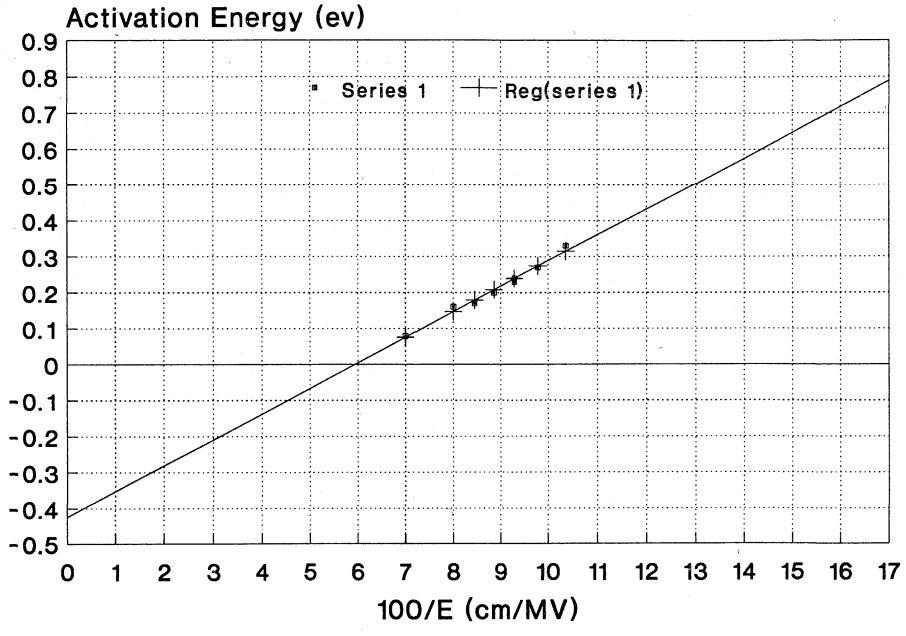


Figure 7 • Activation Energy versus 1/E

Table 3 • Acceleration Factor versus Operating Conditions (Unprogrammed Antifuse)

$t_0 = 1 \times 10^{-16}$ sec., G = 480 MV/cm, f = 0.014 eV, $E_b = 0.43$ eV.

Model	Temperature/Voltage High	Temperature/Voltage Typical	Acceleration Factor	Equivalent Years for 1000 Hour 125°C Burn-In
Fixed Voltage	125°C/5.5 V	55°C/5.5 V	55.3	6.3
	125°C/5.5 V	90°C/5.5 V	6.1	0.7
Fixed Temperature	25°C/5.5 V	25°C/5.25 V	38.8	4.4
	25°C/5.75 V	25°C/5.25 V	1092.6	124.7
	25°C/5.75 V	25°C/5.5 V	28.2	3.2
	25°C/6.0 V	25°C/5.25 V	23321	2662
	25°C/6.0 V	25°C/5.5 V	601.8	68.7
Varied Temperature and Voltage	125°C/5.5 V	55°C/5.25 V	1787.2	204.0
	125°C/5.75 V	55°C/5.5 V	987.8	112.8
	125°C/5.75 V	90°C/5.5 V	109.4	12.5
	125°C/6.0 V	55°C/5.5 V	13865	1583
	125°C/6.0 V	90°C/5.5 V	1535.9	175.3
Fixed 0.6 eV Activation Energy Voltage—Independent	150°C/5.5 V	55°C/5.5 V	117.6	13.4
	150°C/5.5 V	90°C/5.5 V	15.2	1.7
	125°C/5.5 V	55°C/5.5 V	41.8	4.8
	125°C/5.5 V	90°C/5.5 V	5.4	0.6

The Programmed Antifuse

A Kelvin test structure as shown in Figure 8 was used to evaluate the reliability of a programmed antifuse. Here, a strip of polysilicon crosses an N+ diffusion. The antifuse is located at the intersection of the two. There are metal-to-poly contacts at nodes 1 and 3, as well as metal-to-N+ contacts at nodes 2 and 4. A four-terminal Kelvin structure is useful should a failure occur, because antifuse opens can be separated from other problems (such as polysilicon or contact opens) simply by checking for continuity on appropriate pairs of nodes.

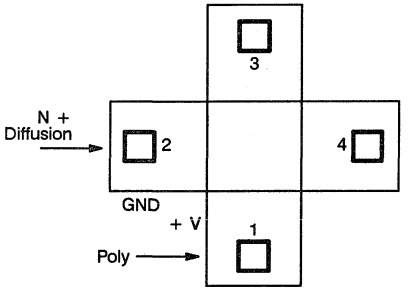


Figure 8 • Antifuse Kelvin Structure

Test devices were stressed by forcing a constant 5 mA current from polysilicon to N+ through the antifuse at 250°C. Note that this stress is far greater than a programmed antifuse would be subject to in a device under normal operating conditions. Because the antifuse is used to connect two networks, there is usually no voltage across it; hence, no current passes through. A voltage will appear across the antifuse only momentarily while a network switches from low-to-high or high-to-low.

During the 5 mA, 250°C stress, the voltage across the antifuse was monitored. Figure 9 is a plot of the voltage as a function of stress time. A sudden increase in voltage indicates that an open occurred. As can be seen from the figure, failures occurred at about 300 hours of stress. However, by probing on nodes 3 and 4 of the Kelvin structure, we were able to measure continuity and determine that the cause of failure was not the antifuse. The failed units were then examined on an SEM, where the cause of failure was revealed as metal-to-poly contact electromigration. This is a well-known failure mode in CMOS, which has been determined to have an activation energy of 0.9 eV. Using equation 4, we can predict a lifetime under normal operating conditions in excess of 40 years for this failure mode. The lifetime of the programmed antifuse is even longer.

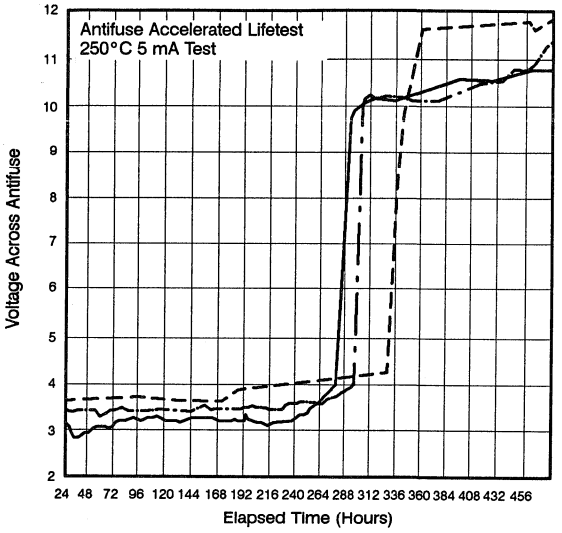


Figure 9 • Voltage Across Antifuse versus Stress Time

Actel FPGA Product Reliability

Product reliability was evaluated on seven Actel products: a 64K PROM (PROM64); a 300-gate FPGA (1003); 1,200-gate FPGAs (A1010, A1010A, A1010B); 2,000-gate FPGAs (A1020, A1020A, A1240B); 2,500-gate FPGAs (A1225, A1225A, A1425, A1425A); 4,000-gate FPGAs (A1240, A1240A); 6,000-gate FPGA (A1460A); and 8,000-gate FPGAs (A1280, A1280A). The PROM64 product uses the same process and antifuse as the FPGAs. The 1003 is a test device that is a smaller version of the A1010/A1020; it was used for early characterization and qualification. As mentioned earlier, the "A" and "B" versions of the FPGAs are linear shrinks of the original 2.0 µm (ACT 1) and 1.2 µm (ACT 2) products. Reliability tests were conducted on units assembled in many different package types as listed in Tables 6 through 11. Package characteristics for Actel FPGAs are shown in Table 4.

High Temperature Operating Life (HTOL)

The intent of an HTOL test is to operate a device dynamically at a high temperature (usually 125°C or 150°C) and extrapolate the failure rate to typical operating conditions. This test is defined by Military Standard-883 in the Group C Quality Conformance Tests. The Arrhenius relationship in equations 3 and 4 is used to do the extrapolation. To use the Arrhenius equation, we need to know the activation energy of the failure mode. Activation energies of antifuse failure modes were discussed previously. Table 5 gives the activation energies of general semiconductor failure modes.

Table 4 • Actel FPGA Package Characteristics

Characteristics	PLCC	PQFP
Molding Compound	Sumitomo 6300 H	Sumitomo 6300 H
Filler Material	Fused silicon 70% by weight	Fused silicon 70% by weight
Lead Frame Material	Copper (Olin 150 or equiv.)	Copper (Olin 150 or equiv.)
Lead Plating Composition	Solder, 300–800 micro inches (µin)	Solder, 300–800 micro inches (µin)
Die Attach Material	Silver Epoxy	Silver Epoxy
Flame Retardance	UL-94, V-0	UL-94, V-0
Bond Wire	Gold, 1.3 mil diameter	Gold, 1.3 mil diameter
Bond Attach Method	Thermosonic	Thermosonic
Characteristics	**JQCC**	**PGA/CQFP**
Body Material	Ceramic	Alumina
Lid Material	Ceramic	Kovar, 50 µin gold plated
Sealant	Glass	Au/Sn Solder
Lead Frame Material	Alloy 42 (40% Nickel, 60% Iron)	N/A
Bond Wire	99% Aluminum, 1% Si, 1.25 mil dia.	99% Aluminum, 1% Si, 1.25 mil. dia.
Bond Attach Method	Ultrasonic	Ultrasonic
Lead Finish	Solder dip, 200 µin. min.	A42 or Kovar
Die Attach Material	Silver loaded glass	Silver loaded glass

Thermal Resistance (°C/Watt)									
Package	**44 PLCC**	**44 JQCC**	**68 PLCC**	**68 JQCC**	**80 VQFP**	**84 PLCC**	**84 JQCC**	**84 PGA**	**84 CQFP**
θ_{JC}	15	8	13	8	12	12	8	8	5
θ_{JA} (still air)	52	38	45	35	68	44	34	33	40
Package	**100 PQFP**	**100 PGA**	**132 PGA**	**144 PQFP**	**160 PQFP**	**172 CQFP**	**176 PGA**	**207 PGA**	**208 PQFP**
θ_{JC}	13	5	5	15	15	8	8	8	15
θ_{JA} (still air)	55	35	30	35	33	25	23	22	33

Six different data patterns are programmed into the 64K PROMs for HTOL testing: a diagonal of zeros (98% programmed); a diagonal of ones (2% programmed); a topological checkerboard pattern (50%); all zeros (100%); all ones (0%); and an incrementing pattern (50%). During burn-in, all addresses are sequenced through at a 1 MHz clock rate. The outputs are enabled and loaded with a 100 ohm resistor to a 2 V supply. This results in an output loading of equal to or greater than the specified limits of Ioh = –4 mA and Iol = 16mA. In most cases, the PROMs are burned in at Vcc = 5.5 V and at 125°C. However, voltage acceleration experiments are also done at 7 V, 125°C and 8 V, 25°C.

The PROM is useful for antifuse reliability studies for several reasons. First, it allows us to program anywhere from 0% to 100% of the antifuses, although we program only 2% to 3% of the antifuses for a given design in an FPGA device. Also, an antifuse failure on the PROM is very noticeable because the antifuse is directly addressed. A weak fuse would show an AC speed drift, and a failed antifuse would read the wrong data.

For evaluating the Actel families of FPGAs, we programmed an actual design application into most devices (some units were burned-in unprogrammed) and performed a dynamic burn-in by toggling the clock pins at 1 MHz. The designs selected utilized 85% to 97% of the available logic modules and 85% to 94% of the I/Os. Outputs were usually loaded with 1.2–2.8 K ohm resistors to Vcc, which results in up to 4 mA of sink current as each I/O toggled low. Under these conditions, each unit typically draws a minimum of 100 mA during dynamic burn-in. Most of this current comes from the output loading while about 5 mA is from the device supply current. For a 125°C burn-in, this results in junction temperatures of at least 150°C for plastic packages and 145°C for ceramic packages (depending on package type). Most burn-in was done at 5.75 V or 6.0 V (for voltage acceleration of the antifuse) and 125°C or 150°C.

Table 5 • CMOS Failure Modes

Failure Mechanism	Activation Energy
Ionic Contamination	1.0 eV
Oxide Defects	0.3 eV
Hot Carrier Trapping in Oxide (Short Channels)	–0.06 eV
Silicon Defects	0.5 eV
Aluminum-Silicon-Copper Electromigration	0.6 eV
Contact Electromigration	0.9 eV
Electrolytic Corrosion	0.54 eV

As mentioned previously, some units are burned in unprogrammed. To accomplish this, we use a special burn-in circuit that allows us to use the product's test features to serially shift in commands to the chip during burn-in. All internal routing tracks are toggled between Vss and Vcc. When vertical tracks are at Vcc, horizontal tracks are held at Vss and vice versa. Thus, all antifuses that can connect vertical and horizontal tracks receive a full Vcc stress in both directions. Since vertical tracks connect to logic module inputs and outputs, these are also toggled between Vss and Vcc. Finally, a command is sent to the chip to toggle some external I/O pins between Vss and Vcc. This special dynamic burn-in circuit is the same one used by Actel to screen unprogrammed products to MIL-STD-883 requirements. Since virtually all antifuses receive a full Vcc stress, this screen is much more effective at catching unprogrammed antifuse infant mortality failures than is burning in programmed devices where only a fraction of the antifuses are stressed.

A summary of the HTOL data collected by Actel is shown in Table 6. A failure is defined as any device that shows a functional failure, exceeds datasheet DC limits, or exhibits AC speed drift. Among the parts tested, no speed drift, faster or slower, was observed within the accuracy of the test set-up. Failure rates at 55°C, 70°C, and 90°C were extrapolated by using the Arrhenius equation and general activation energies of 0.6 eV and 0.9 eV. Poisson statistics were used to derive a calculated failure rate with a 60% confidence level. Using Poisson statistics is valid for a failure rate that is low and a failure mode that occurs randomly with time. At 55°C, the calculated failure rate with 60% confidence level (0.6 eV) was found to be 20 FITs or 0.0020% failures per 1000 hours. This number was derived from over 9.8 million device hours (125°C) of data. There were eight total failures out of 7856 tested units. Only one occurred in the first 80 hours of burn-in (of a 1003 product). Seven of the eight failures observed were due to common CMOS failure modes (gate oxide failure, silicon defects, or open via). Only one unit failed due to the antifuse. This unit was burned in in the unprogrammed state to stress all antifuses. It was stressed at 6 V, 150°C. It passed at 168 hours and failed at 650 hours because an antifuse became programmed. By passing at 168 hours, the unit received a total stress well in excess of 100 years of operation at 5.5 V, 125°C (using equation 8). With only one antifuse related failure in 9.8 million device hours at 125°C, we use equation 8 to derive that this one antifuse failure at 6 V is significantly less than 10 FITs at 5.5 V.

Unbiased Pressure Pot Test

This test is used to qualify products in plastic packages. Units are placed in an autoclave (pressure pot) and exposed to a saturated steam atmosphere at 121°C and 15 psi. Problems with bonding, molding compounds, or wafer passivation can cause metal corrosion to occur in this atmosphere. The existence of metal corrosion is detected during a full electrical test of the device following exposure in the autoclave.

A total of 1771 units from 44 wafer runs were evaluated. Read points were taken at 96, 168, 240, and 336 hours. There were a total of seven failures (Table 7). Five failures were found to be due to bond wires lifting off bond pads. These were caused by assembly problems that occurred only in our first lots of plastic units. The failures were caused by temperature and not by metal corrosion. The assembly problems have been corrected, with no further failures observed. Two A1010A units failed at 336 hours because of metal corrosion at the bond pads, but both units had passed at 168 hours.

Biased Moisture Life Test (85/85)

In this test, the units are placed in a chamber at a temperature of 85°C and a relative humidity of 85%. A voltage of 5.5 V is applied to every other device pin while other pins are grounded. 5.5 V is applied to Vcc while Vss is grounded. This test effectively detects die-related and plastic package-related problems.

As shown in Table 8, a total of 1277 units have been stressed. There have been three failures. Two failures were due to lifted bond wires; these units came from the same lot in which we saw failures in the steam pressure pot. The 1000-hour failure is nonfunctional due to an open Metal 1 line.

Highly Accelerated Stress Test (HAST) at 131°C/85% Humidity

As in the 85/85 test, units receive an alternate pin bias (5.5 V and 0 V) but are exposed to a higher pressure and higher temperature environment. Fifty hours of HAST is generally considered to be equivalent to 1000 hours of 85/85. As summarized in Table 9, 1277 units from 28 wafer runs have been tested with no failures.

Temperature Cycling/Thermal Shock

These tests check for package integrity by cycling units through temperature extremes. Data was taken for cycles of 0°C to 125°C, –40°C to 125°C, and –65°C to 150°C. Both programmed and unprogrammed units were placed on temperature cycle. As shown in Table 10, of 4110 units tested to date, there have been no failures. We have also performed –65°C to 150°C Thermal Shock test on 272 units (Table 11), and have seen no failures.

Table 6 • High Temperature Operating Life (HTOL) Summary

Product	Total Units	Total Wafer Runs	Device Hours at 125°C (0.6 eV)	Failures	Equiv. Dev. Hrs. at 55°C (0.6 eV)	Equiv. Dev. Hrs. at 70°C (0.6 eV)	Equiv. Dev. Hrs. at 90°C (0.6 eV)
64K PROM	295	4	6.18E+05	0	2.59E+07	1.02E+07	3.34E+06
1003	238	3	3.60E+05	1	1.50E+07	5.94E+06	1.94E+06
A1010	703	10	1.17E+06	1	4.87E+07	1.93E+07	6.29E+06
A1010A	618	13	8.20E+05	1	3.43E+07	1.36E+07	4.43E+06
A1010B	536	5	5.08E+05	1	2.12E+07	8.39E+06	2.74E+06
A1020	334	5	2.16E+05	0	9.04E+06	3.57E+06	1.17E+06
A1020A	1720	29	2.61E+06	3	1.09E+08	4.32E+07	1.41E+07
A1020B	737	12	6.75E+05	0	2.82E+07	1.12E+07	3.64E+06
A1225	207	1	2.07E+05	0	8.66E+06	3.42E+06	1.12E+06
A1225A	80	2	8.00E+04	0	3.35E+06	1.32E+06	4.32E+05
A1240	485	7	6.11E+05	0	2.55E+07	1.01E+07	3.30E+06
A1240A	210	4	2.40E+05	0	1.00E+07	3.97E+06	1.30E+06
A1280	586	12	7.03E+05	1	2.94E+07	1.16E+07	3.80E+06
A1280A	480	9	3.62E+05	0	1.52E+07	5.99E+06	1.96E+06
A1425	275	3	2.75E+05	0	1.15E+07	4.55E+06	1.49E+06
A1425A	130	3	1.30E+05	0	5.44E+06	2.15E+06	7.02E+05
A1460	222	2	2.22E+05	0	9.29E+06	3.67E+06	1.20E+06
Totals	**7856**	**124**	**9.80E+06**	**8**	**4.10E+08**	**1.62E+08**	**5.30E+07**

HTOL Summary

Total Units:	7856
Total Runs:	113
Total Device Hours:	8.93E+06
Total Failures:	8

Equivalent Device Hours:

Ambient Temperature, Activation Energy	Device Hours
90°C, 0.6EV	5.30E+07
70°C, 0.6EV	1.62E+08
55°C, 0.6EV	4.10E+08
90°C, 0.6EV	1.35E+08
70°C, 0.6EV	7.24E+08
55°C, 0.6EV	2.91E+09

Fits:	Observed	60% Confidence
90°C, 0.6EV	151	178
70°C, 0.6EV	49	58
55°C, 0.6EV	20	23
90°C, 0.6EV	59	70
70°C, 0.6EV	11	13
55°C, 0.6EV	3	3

Table 6 • High Temperature Operating Life (HTOL) Summary (Continued)

High Temperature Operating Life

Product	Run #	Package	# Units	Pattern	# Hours	# Failures	Temp.	V_{CC}
64KPR	DG1060	24 SB	59	6PATS	2000	0	125	5.50
64KPR	DG1064	24 SB	36	6PATS	2500	0	125	5.50
64KPR	JB13	24 SB	40	6PATS	2000	0	125	5.50
64KPR	JB13	24 SB	40	6PATS	2000	0	125	7.00
64KPR	JB13	24 SB	10	6PATS	2000	0	25	8.00
64KPR	JB14	24 SB	50	6PATS	2500	0	125	5.50
64KPR	JB14	24 SB	50	6PATS	2500	0	125	7.00
64KPR	JB14	24 SB	10	6PATS	2500	0	25	8.00
1003	DG1063	84 JLCC	25	AL7C	2000	0	125	5.75
1003	DG1065	84 JLCC	32	AL7C	2000	0	125	5.75
1003	DG1065	84 JLCC	159	AL7C	500	1	150	7.00
1003	DG1067	84 JLCC	22	AL7C	1000	0	125	5.75
1010	DG1042	84 JLCC	3	4BCNTR	1000	0	125	5.75
1010	DG1047	84 JLCC	2	4BCNTR	1000	0	125	5.75
1010	DG1073	84 JLCC	10	AL9B	2000	0	125	5.75
1010	DG1077	84 JLCC	15	AL9B	2000	0	125	5.75
1010	JB13	84 PLCC	32	AL9B	2000	0	125	5.75
1010	JB13	68 PLCC	126	AL9B	2000	1	125	5.75
1010	JB13	84 JLCC	64	AL9B	2000	0	125	5.75
1010	JB14	68 PLCC	100	AL9B	2000	0	125	5.75
1010	JB14	68 JLCC	50	AL9B	2000	0	125	5.75
1010	JB22	68 PLCC	100	AL9B	1000	0	125	5.75
1010	JB42	68 PLCC	47	VFUSE	1000	0	125	5.75
1010	JB44	68 PLCC	40	VFUSE	1000	0	125	5.75
1010	JB46	68 PLCC	49	VFUSE	1000	0	125	5.75
1010	TI24	68 PLCC	65	VFUSE	2000	0	125	5.75
1010A	JG03	68 PLCC	59	VFUSE	2000	0	125	5.75
1010A	JG03	68 PLCC	117	VFUSE	1000	0	125	5.75
1010A	TI24	68 PLCC	74	VFUSE	2000	0	125	5.75
1010A	TI29	68 PLCC	53	AL9B	1000	1	125	5.75
1010A	TI31	68 PLCC	26	AL9B	2000	0	125	5.75
1010A	TI32	68 PLCC	9	AL9B	2000	0	125	5.75
1010A	TI33	68 PLCC	19	AL9B	2000	0	125	5.75
1010A	TI35	68 PLCC	15	AL9B	2000	0	125	5.75
1010A	TI1104 TI1243 TI1263 TI1297	68 PLCC	107	AL9B	1000	0	125	5.75
1010A	E01-1	68 PLCC	69	AL9B	1000	0	125	5.75
1010A	E02-1	68 PLCC	70	AL9B	1000	0	125	5.75

Table 6 • High Temperature Operating Life (HTOL) Summary (Continued)

High Temperature Operating Life								
Product	**Run #**	**Package**	**# Units**	**Pattern**	**# Hours**	**# Failures**	**Temp.**	**V_{CC}**
1010B	TI2072857 TI2072858 TI2072860	68 PLCC	400	AL9B	1000	1	125	5.75
1010B	U1G-01	68 PLCC	79	AL9B	1000	0	125	5.75
1010B	U1G-02	68 PLCC	57	AL9B	500	0	125	5.75
1020	DG1133	84 JLCC	29	AL11	2000	0	125	5.75
1020	JB22	84 PLCC	16	AL11	1000	0	125	5.75
1020	JB22	84 JLCC	32	AL11	1000	0	125	5.75
1020	JB25	84 PLCC	16	AL11	1000	0	125	5.75
1020	JB26	84 PLCC	32	AL11	500	0	125	5.75
1020	JE03	84 JQCC	104	AL11	186	0	150	5.75
1020	JB33	84 JLCC	105	AL11	80	0	150	5.75
1020A	JF01	84 JLCC	25	AL11	2000	0	125	5.75
1020A	JF01	84 PLCC	15	VFUSE	2000	0	125	5.75
1020A	JF02	84 JLCC	44	AL11	2000	0	125	5.75
1020A	JF02	84 JLCC	41	VFUSE	2000	0	125	5.75
1020A	JF04	84 PLCC	77	VFUSE	1000	0	125	5.75
1020A	JF04	84 PLCC	20	VFUSE	500	0	125	5.75
1020A	JF14	84 PLCC	58	AL11	417	0	150	6.00
1020A	JF14	84 PLCC	100	BLANK	417	1	150	6.00
1020A	JF37	84 PLCC	14	AL11	300	1	150	6.00
1020A	JF37	84 PLCC	20	BLANK	300	0	150	6.00
1020A	JF39	84 PLCC	32	AL11	300	0	150	6.00
1020A	JF39	84 PLCC	29	BLANK	300	0	150	6.00
1020A	JF42	84 PLCC	49	BLANK	650	0	150	6.00
1020A	JF42	84 PLCC	30	VFUSE	650	1	150	6.00
1020A	JF66	84 PLCC	33	AL11	1000	0	125	5.75
1020A	JF66	84 PLCC	39	BLANK	1000	0	125	5.75
1020A	JF67	84 PLCC	49	AL11	1000	0	125	5.75
1020A	JF67	84 PLCC	45	BLANK	1000	0	125	5.75
1020A	TI S#1	84 PLCC	79	AL11	1000	0	125	5.75
1020A	E-14	84 PLCC	45	AL11	2000	0	125	5.75
1020A	E-15	84 PLCC	44	AL11	2000	0	125	5.75
1020A	E-17	84 PLCC	45	AL11	2000	0	125	5.75
1020A	JF-207	100 PQFP	129	AL11	1000	0	125	5.75
1020A	D1J1815 D2B2704	84 PGA	51	ICE20	1000	0	125	5.75

Table 6 • High Temperature Operating Life (HTOL) Summary (Continued)

High Temperature Operating Life								
Product	**Run #**	**Package**	**# Units**	**Pattern**	**# Hours**	**# Failures**	**Temp.**	**V_{CC}**
1020A	E-01	84 PLCC	45	AL11	1000	0	125	5.75
1020A	E-02	84 PLCC	45	AL11	1000	0	125	5.75
1020A	E-03	84 PLCC	45	AL11	1000	0	125	5.75
1020A	ADK29X	84 PLCC	45	AL11	2000	0	125	5.75
1020A	ADA72X	84 PLCC	45	AL11	2000	0	125	5.75
1020A	ADC21X	84 PLCC	45	AL11	2000	0	125	5.75
1020A	TI1130	84 PLCC	223	AL11	1000	0	150	6.00
1020A	TI1139							
1020A	TI1210							
1020A	TI1800	84 PLCC	34	AL11	648	0	150	6.00
1020A	TI1803							
1020A	UP-04	84 PLCC	40	AL11	1000	0	125	5.75
	UP-05	84 PLCC	40	AL11	1000	0	125	5.75
1020B	JJ-14	84 PLCC	45	AL11	1000	0	125	5.75
1020B	JJ-15	84 PLCC	45	AL11	1000	0	125	5.75
1020B	JJ-17	84 PLCC	45	AL11	1000	0	125	5.75
1020B	JJ-13	84 PGA	30	ICE20	1000	0	125	5.75
1020B	JJ-13	84 PGA	80	AL11	500	0	125	5.75
1020B	JJ-16	84 PLCC	80	AL11	1000	0	125	5.75
1020B	U1P-01	84 PLCC	40	AL11	1000	0	125	5.75
1020B	U1P-02	84 PLCC	40	AL11	1000	0	125	5.75
1020B	JJ-24	84 PLCC	87	AL11	1000	0	125	5.75
1020B	EBFJ004	84 PLCC	80	AL11	1000	0	125	5.75
1020B	EBFI004	84 PLCC	40	AL11	1000	0	125	5.75
1020B	U1P41HM	100 PQFP	80	AL11	1000	0	125	5.75
1020B	U1P25	80 VQFP	45	AL11	500	0	125	5.75
1225	UJ-01	100 PGA	80	SPEED26	1000	0	125	5.75
1225	UJ-01	100 PQFP	127	PQFP26	1000	0	125	5.75
1225A	U1J-02	100 PQFP	80	PQFP26	1000	0	125	5.75
1240	TI3257	132 PGA	7	CHI1240	500	0	125	5.75
1240	TI3257	144 PQFP	129	CHI1240	1000	0	125	5.75
1240	TI1045571	132 PGA	38	CHI1240	2000	0	125	5.75
1240	TI1053933	132 PGA	55	CHI1240	2000	0	125	5.75
1240	TI1053932	132 PGA	36	CHI1240	2000	0	125	5.75
1240	TI1220494	132 PGA	90	CHI1240	1000	0	125	5.75
1240	UI-01	132 PGA	50	CHI1240	1000	0	125	5.75
1240	UI-03	84 PLCC	80	ACT40	1000	0	125	5.75

Table 6 • High Temperature Operating Life (HTOL) Summary (Continued)

High Temperature Operating Life								
Product	**Run #**	**Package**	**# Units**	**Pattern**	**# Hours**	**# Failures**	**Temp.**	**V_{CC}**
1240A	E-02	144 PQFP	100	CHI1240	1000	0	125	5.75
	E-03							
1240A	E-04	84 PLCC	30	ACT40	2000	0	125	5.75
1240A	U1L-26	144 PQFP	80	CHI1240	1000	0	125	5.75
1280	JH05	176 PGA	15	BETA12	2000	0	125	5.75
1280	JH06	176 PGA	15	BETA12	2000	0	125	5.75
1280	JH03(K)	176 PGA	25	BETA12	2000	0	125	5.75
1280	JH03(SB)	176 PGA	25	BETA12	2000	0	125	5.75
1280	TI1143649	176 PGA	44	BETA12	1000	1	125	5.75
1280	TI1143650	176 PGA	44	BETA12	1000	0	125	5.75
1280	TI1136307	176 PGA	42	BETA12	1000	0	125	5.75
1280	UH-01	176 PGA	26	BETA12	1000	0	125	5.75
1280	UH-02	176 PGA	26	BETA12	1000	0	125	5.75
1280	UH-05	176 PGA	40	SPEED9	1000	0	125	5.75
1280	UH-04	160 PQFP	79	SPEED12	1000	0	125	5.75
1280	UH-10	176 PGA	75	SPEED13	1487	0	125	5.75
	UH-14							
1280	ADC18X	160 PQFP	130	SPEED12	1000	0	125	5.75
1280A	EBFJ002	160 PQFP	30	SPEED12	168	0	125	5.75
1280A	EBFJ003	160 PQFP	30	SPEED12	168	0	125	5.75
1280A	EBFJ004	160 PQFP	20	SPEED12	168	0	125	5.75
1280A	UHI-01	160 PQFP	27	SPEED12	1000	0	125	5.75
1280A	UHI-02	160 PQFP	27	SPEED12	2000	0	125	5.75
1280A	UIH-35	176 PGA	132	HUTMP80	72	0	150	5.75
1280A	UIH-18	160 PQFP	80	SPEED12	1000	0	125	5.75
1280A	EWAJ03,04	160 PQFP	134	HUTMP80	1000	0	125	5.75
1425	JK08,09,10	133 PGA	140	C25H	1000	0	125	5.75
1425	JK08,09,10	84 PLCC	135	C25L	1000	0	125	5.75
1425A	UCJ01,2,3	133 PGA	130	C25L	1000	0	125	5.75
1460A	JL-01	208 PQFP	80	C60H	1000	0	125	5.75
1460A	JL-01	207 PGA	80	C60H	1000	0	125	5.75
1460A	JL-03	208 PQFP	62	C60H	1000	0	125	5.75

Table 6 • High Temperature Operating Life (HTOL) Summary (Continued)

HTOL Failure Analysis			
Product	**Run #**	**Hours**	**Cause**
1003	DG1065	80	I_{CC} = 40 mA. Fails at High V_{CC}. Functional at Low V_{CC}. Not antifuse related
1010	JB13	500	I_{CC} = 30 to 40 mA. Not Functional. Liquid crystal shows hot spot in chip periphery. Silicon defect. Not antifuse related.
1010A	TI29	500	Functional at V_{CC} > 4.5 V. All nets slow. Damaged isolation transistor oxide. Not antifuse related. Unit was marginal at T_0.
1010B	TI2072857	144	I_{DDH} failure at 144 hours. Gate poly to drain short due to gate oxide breakdown in I/O.
1020A	JF14	417	Open Metal I to Metal II via.
1020A	JF37	300	Gross functional failure due to shorted gate oxide in the logic module. Unit was good at 112 hours.
1020A	JF42	650	Failed fuse shorts. Cause is shorted antifuse. Unit was good at 168 hours. Blank pattern at 6 V is much stronger stress than normal application.
1280	TI1143649	500	Tristate leakage on pin 28. No cause determined due to bond wire damage during decapsulation.

Table 7 • 121°C, 15 PSI Steam Pressure Pot (Unbiased Autoclave)

				Number of Failures			
Product	**Run #**	**Package**	**# Units**	**96 Hours**	**168 Hours**	**240 Hours**	**336 Hours**
1010	JB13	84 PLCC	34	0	3	—	0
1010	JB13	68 PLCC	71	1	0	—	0
1010	JB14	68 PLCC	71	0	0	—	0
1010	JB22	68 PLCC	50	0	0	—	0
1010	JB27	68 PLCC	50	0	0	—	0
1010	JB28	68 PLCC	42	1	0	—	0
1010A	TI15	68 PLCC	77	0	0	—	0
1010A	TI24	68 PLCC	129	0	0	—	0
1010A	TI1104 TI1243 TI1263 TI1297	68 PLCC	77	0	—	0	0
1010A	E01-1	68 PLCC	30	0	0	—	0
1010A	E02-1	68 PLCC	30	0	0	—	0
1010A	E03-1	68 PLCC	30	0	0	—	2
1010B	TI2072857 TI2072858 TI2072860	68 PLCC	45	0	0	0	—
1010B	U1G-01	68 PLCC	40	—	0	—	—
1020A	TI1800 TI1859 TI2156	84 PLCC	77	0	—	0	—
1020A	E-01	84 PLCC	26	0	0	—	0
	E-02	84 PLCC	28	0	0	—	0
	E-03	84 PLCC	26	0	0	—	0

Table 7 • 121°C, 15 PSI Steam Pressure Pot (Unbiased Autoclave) (Continued)

				Number of Failures			
Product	**Run #**	**Package**	**# Units**	**96 Hours**	**168 Hours**	**240 Hours**	**336 Hours**
1020A	ADK29	84 PLCC	27	—	0	—	—
	ADC21X	84 PLCC	27	—	0	—	—
	ADA72X	84 PLCC	27	—	0	—	—
1020A	JF-207	100 PQFP	80	—	0	—	—
1020A	S-1702A	84 PLCC	25	—	0	—	0
	S-1702B		26	—	0	—	0
	S-1702C		25	—	0	—	0
1020B	JJ14-17	84 PLCC	81	—	0	—	—
1020B	U1P-01	84 PLCC	40	—	0	—	—
1225	UJ-01	100 PQFP	80	—	0	—	0
1240	TI10301	144 PQFP	79	—	0	—	0
1240	UI-03	84 PLCC	79	—	0	—	0
1240A	E-02	144 PQFP	25	—	0	—	—
1240A	E-03	144 PQFP	30	—	0	—	—
1240A	E-04	84 PLCC	25	—	0	—	—
1280	JH-14	160 PQFP	80	—	0	—	0
1280	ADC18X	160 PQFP	82	—	0	—	—

Failure Analysis:

Three 1010 JB13 failures at 168 hours due to lifted bond wires.
Corrective action implemented at assembly vendor.

1010 JB13 failure at 96 hours; same problem as above.

1010 JB28 failure at 96 hours due to open bond wire caused by lifted die paddle. This was the first qual lot from a new assembly vendor and corrective action was implemented.

Two 1010A E03-1 failed cont. at 336 hours. Both units had corroded pads.

Table 8 • 85°C/85% Humidity with DC Alternate Pin Bias of 0 V to 5.5 V

				Number of Failures			
Product	**Run #**	**Package**	**# Units**	**168 Hours**	**500 Hours**	**1000 Hours**	**2000 Hours**
1010	JB13	68 PLCC	80	—	2	1	0
1010	JB14	68 PLCC	81	—	0	0	0
1010	JB22	68 PLCC	54	—	0	0	—
1010	JB26	68 PLCC	54	—	0	0	—
1010	JB27	68 PLCC	19	—	0	0	—
1010A	E01-1	68 PLCC	79	—	0	—	—
	E02-2						
	E03-1						
1010A	TI1104	68 PLCC	80	0	0	0	—
	TI1243						
	TI1263						
	TI1297						
1010B	TI2072857	68 PLCC	201	0	0	0	—
	TI2072858						
	TI2072860						
1020	JF48	84 PLCC	34	0	0	—	—
	JF49	84 PLCC	49	0	0	—	—
1020A	E-01	84 PLCC	64	—	0	0	—
	E-02						
	E-03						
1020A	ADK29	84 PLCC	27	—	0	0	—
	ADC21X	84 PLCC	27	—	0	0	—
	ADA72X	84 PLCC	27	—	0	0	—
1020A	E14	84 PLCC	24	—	0	0	—
	E15	84 PLCC	29	—	0	0	—
	E17	84 PLCC	32	—	0	0	—
1020A	UP-06	100 PQFP	77	—	0	0	—
1020B	JJ-14	84 PLCC	27	—	—	0	—
	JJ-15	84 PLCC	27	—	—	0	—
	JJ-17	84 PLCC	27	—	—	0	—
1240	TI3256	144 PQFP	78	—	0	0	—
1280	UH-04	160 PQFP	80	—	0	0	—

Failure Analysis:

1010	JB13	Two failures at 500 hours. Open pins due to bond lifting. Corrective action implemented at assembly vendor.
		One failure at 1000 hours. Horizontal track open (Metal I).

Table 9 • Biased Humidity (HAST) 131°C/85% Humidity

Product	Run #	Package	# Units	Number of Failures					
				50 Hours	100 Hours	200 Hours	240 Hours	300 Hours	400 Hours
1020	E18, 22	84 PLCC	29		0	0	—	0	0
1020A	TI1130	84 PLCC	77	—	0	0	0	—	—
	TI1139								
	TI1210								
1020B	EBFJ001	84 PLCC	80	—	0	—	—	—	—
1020B	U1P41HM	100 PQFP	81	—	0	—	—	—	—
1225	UJ-03	100 PQFP	80	0	0	—	—	—	—
1240	UI-03	84 PLCC	80	0	0	—	—	—	—
1240A	E-04	84 PLCC	77	0	—	—	—	—	—
1240A	E-02,03	144 PQFP	53	0	—	—	—	—	—
1240A	U1I26	144 PQFP	80	—	0	—	—	—	—
1280	ADC20X	160 PQFP	80	0	—	—	—	—	—
1280A	U1H-01	160 PQFP	40	0	—	—	—	—	—
1280A	U1H-02	160 PQFP	40	0	—	—	—	—	—
1280A	EBFJ002	160 PQFP	30	0	—	—	—	—	—
1280A	EBFJ003	160 PQFP	30	0	—	—	—	—	—
1280A	EBFJ004	160 PQFP	20	0	—	—	—	—	—
1280A	U1H18	160 PQFP	80	—	0	—	—	—	—
1280A	EWAJ003	160 PQFP	79	—	0	—	—	—	—
1425	JK8,9,10	84 PLCC	81	—	0	—	—	—	—
1425A	UCJ01,2,3	100 PQFP	80	—	0	—	—	—	—
1460A	JL04A	208 PQFP	80	—	0	—	—	—	—

Table 10 • Temperature Cycle

Product	Run #	Package	# Units	Number of Failures			
				100 Cycles	500 Cycles	1000 Cycles	2000 Cycles
0°C to 125°C Cycles							
1010	DG1077	84 JLCC	20	0	—	—	—
1010	JB13	68 PLCC	158	0	—	0	—
1010	JB14	68 PLCC	28	0	—	0	—
1010	JB26	68 PLCC	21	0	—	0	—
1010	JB28	68 PLCC	31	0	—	0	—
1010A	TI15	68 PLCC	125	0	—	0	—
1010A	TI24	68 PLCC	176	0	—	0	—
1010A	TI1104	68 PLCC	129	0	0	0	0
	TI1243						
	TI1263						
	TI1297						
1020	JB22	84 JLCC	17	0	—	0	—
1020A	TI1800	84 PLCC	129	0	0	0	0
	TI1859						
	TI2156						

Table 10 • Temperature Cycle (Continued)

Product	Run #	Package	# Units	Number of Failures			
				100 Cycles	500 Cycles	1000 Cycles	2000 Cycles
–40°C to 125°C Cycles							
1010A	TI1104	68 PLCC	129	0	0	0	0
	TI1243						
	TI1263						
	TI1297						
1020A	TI1800	84 PLCC	129	0	0	0	0
	TI1859						
	TI2156						
–65°C to 1505°C Cycles							
1010A	TI1104	68 PLCC	129	0	0	0	0
	TI1243						
	TI1263						
	TI1297						
1010B	TI2072857	68 PLCC	201	0	0	0	—
	TI2072858						
	TI2072860						
1010B	U1G-01,02	68 PLCC	40	—	—	0	—
1020A	TI1800	84 PLCC	129	0	0	0	0
	TI1859						
	TI2156						
1020A	JF-71	100 PQFP	129	—	—	0	—
1020A	E01	84 PLCC	85	—	—	0	—
	E02						
	E03						
1020A	S-1702A,B,C	84 PLCC	144	0	—	0	—
1020B	JJ14-17	84 PLCC	81	—	—	0	—
1020B	U1P-01,02	84 PLCC	40	—	—	0	—
1020B	EBFJ001	84 PLCC	80	—	—	0	—
1020B	U1P41HM	100 PQFP	80	—	—	0	—
1020B	EWAI003	84 PLCC	80	—	—	0	—
1225	UJ-01	100 PGA	80	0	—	0	—
1225	UJ-01	100 PQFP	80	—	—	0	—
1240	TI1053932	132 PGA	15	0	0	0	—
	TI1045571		20				
	TI1053933		42				
1240	TI 3256	144 PQFP	80	—	0	0	—
1240	UI-01	132 PGA	50	0	—	0	—
1240	UI-02	132 PGA	50	0	—	0	—
1240	UI-03	84 PLCC	80	—	—	0	—
1240A	E-02	144 PQFP	25	—	—	0	—
1240A	E-03	144 PQFP	28	—	—	0	—
1240A	E-04	84 PLCC	77	—	—	0	—
1240A	U1I-26	144 PQFP	80	—	—	0	—
1280	TI1136307	176 PGA	71	0	—	0	—
	TI1143650	176 PGA	5	0	—	0	—
	TI1143649	176 PGA	1	0	—	0	—

Table 10 • Temperature Cycle (Continued)

Product	Run #	Package	# Units	Number of Failures			
				100 Cycles	500 Cycles	1000 Cycles	2000 Cycles
1280	UH-01	176 PGA	25	0	—	0	—
1280	UH-02	176 PGA	26	0	—	0	—
1280	UH-04	160 PQFP	34	0	—	0	—
1280	JH-14	160 PQFP	48	0	—	0	—
1280	ADE16X	160 PQFP	27	—	—	0	—
1280	ADC18X	160 PQFP	30	—	—	0	—
1280	ADC20X	160 PQFP	27	—	—	0	—
1280A	UH1-01	160 PQFP	40	—	—	0	—
1280A	UH1-02	160 PQFP	40	—	—	0	—
1280A	UIH18	160 PQFP	80	—	—	0	—
1280A	EWAF003	160 PQFP	75	—	—	0	—
1280A	EWAJ03,04	160 PQFP	80	—	—	0	—
1425	JK8,9,10	133 PGA	81	—	—	0	—
1425	JK8,9,10	84 PLCC	83	—	—	0	—
1425A	UCJ01,2,3	100 PQFP	80	—	—	0	—
1440A	JN-02	160 PQFP	80	—	—	0	—
1460A	JL-01	208 PQFP	80	—	—	0	—
1460A	JL-01	207 PGA	80	—	—	0	—

Table 11 • Thermal Shock –65°C to 150°C

Product	Run #	Package	# Units	Number of Failures			
				100 Cycles	500 Cycles	1000 Cycles	2000 Cycles
1010A	TI1104	68 PLCC	77	0	—	0	0
	TI1243						
	TI1263						
	TI1297						
1010B	TI2072857	68 PLCC	45	0	0	0	—
	TI2072858						
	TI2072860						
1020A	TI1800	84 PLCC	77	0	—	—	—
	TI1859						
	TI2156						
1240	TI149935	132 PGA	43	—	0	—	—
1280	UH-01, 02	176 PGA	30	0	—	—	—

Other Tests

Electrostatic Discharge (ESD)

All Actel products contain static electricity protection circuitry and are tested for sensitivity to static electricity by using the human body model as described in MIL-STD-883D (100 pf discharged through 1.5 Kohms). Three positive and three negative pulses are discharged into each pin tested at each voltage level. For inputs and I/Os, these six pulses are applied with three different grounding conditions: Vss only grounded, Vcc only grounded, and all other I/Os grounded. Thus, each pin receives a total of eighteen pulses for each test voltage. After pulsing, the units are then tested on a VLSI tester. Leakage currents are measured at 0 V and 5.5 V. Any pin showing more than 1 µA of leakage is considered to be a failure. To date, all Actel products pass ESD testing at 1500 V or better. Newer 1.0µm and 0.8µm products exceed 2000V on all I/Os. For further information about specific products and packages, please contact Actel.

Latch-Up

Latch-up is a well known cause of failure in CMOS circuits. Parasitic bipolar transistors are created by the P-Channel transistors, the N-channel transistors, the N-Well, and the P-Substrate. These transistors are connected in a manner that effectively creates an SCR. If a voltage on an external pin were to forward bias to the substrate, the parasitic SCR can be latched to the on state, thereby creating a low impedance path between Vcc and ground. A large amount of current then flows through this path. This current can, at best, temporarily make the device nonfunctional and, at worst, cause permanent damage.

There are several techniques used by CMOS designers to reduce the chance of latch-up. One of the most common techniques is using guard rings to isolate P-Channel and N-Channel transistors. The disadvantage of this method is that it requires additional silicon die area. Another method is to use a substrate bias generator. Creating a negative substrate bias means that an input must go even more negative to cause latch-up. A third technique is to use EPI wafers to achieve low substrate resistance, which lowers the chance of triggering latch-up. Actel uses both guard ring and EPI wafer techniques for all FPGA devices.

The latch-up test method is defined by JEDEC Standard No. 17. Each I/O pin on a tested device was forward biased in both directions (to Vss and Vcc) by forcing negative and positive currents ranging from ±50 mA to ±250 mA in 50 mA increments. Following each stress, the device Icc current was measured. If the current exceeded the data sheet limit of 10 mA, the unit was rejected. The device was also functionally tested.

Six units from three different wafer lots are tested to qualify each Actel product. Testing is done at room temperature as well as at a worst-case temperature of 135°C. All device I/Os and power supplies are tested. To date, no failures have been detected up through 250 mA. Additional testing has been performed up to 400 mA without any failures.

Radiation Hardness

A programmed antifuse makes a connection between an upper layer of polysilicon and an N+ diffusion on the bottom. This connection is very similar to a "buried contact" used in some MOS processes. Many other programmable logic products rely on a stored charge to make their connections (for example, RAM, EPROM, or EEPROM). This stored charge can be susceptible to degradation from radiation exposure. The Actel antifuse makes a hard contact and does not rely on a stored charge. As a result, one would expect the Actel products to have superior radiation tolerance compared to products that use a stored charge.

Actel does not perform any radiation testing on devices, however, several customers and independent contractors have performed tests and shared their data. While the results depend on device type and process technology, Actel devices have shown the capability to withstand total dose and single event upset effects. For detailed information on radiation results, please contact your local Actel sales representative.

Summary

The data presented in this report establishes the excellent reliability of Actel FPGAs. Both Actel models and actual device testing show that the antifuse is highly reliable and that the combined contribution of all antifuses to the gate array product's hard failure rate is less than 10 FITs or 0.001% failures per 1000 hours.

References:

1. E. Hamdy, et al, "Dielectric Based Antifuse for Logic and Memory ICs," IEDM paper, p. 786–789, 1988
2. S. Chiang, et al, "Oxide-Nitride-Oxide Antifuse Reliability," Proc. Int. Rel. Phys. Symp., 1990
3. J. Lee, I. Chen, and C. Hu, "Modeling and Characterization of Gate Oxide Reliability," IEEE Trans. of Elec. Dev., Dec. 1988.

Antifuse Field Programmable Gate Arrays

JONATHAN GREENE, MEMBER, IEEE, ESMAT HAMDY, SENIOR MEMBER, IEEE, AND SAM BEAL

Invited Paper

An antifuse is an electrically programmable two-terminal device with small area and low parasitic resistance and capacitance. Field programmable gate arrays (FPGA's) using antifuses in a segmented channel routing architecture now offer the digital logic capabilities of an 8000-gate conventional gate array and system speeds of 40–60 MHz. A brief survey of antifuse technologies is provided. The antifuse technology, routing architecture, logic module, design automation, programming, testing and use of ACT™ antifuse FPGA's are described. Some inherent tradeoffs involving the antifuse characteristics, routing architecture and logic module are illustrated.

I. Introduction

A decade ago the designer of an application specific integrated circuit (ASIC) was constrained chiefly by the performance, logic complexity, and fabrication cost attainable with state-of-the-art semiconductor process technology. Today, the capabilities of leading edge silicon have outstripped the requirements of typical applications. Gate arrays with over 100 000 usable gates and propagation delays well below 1 nsec are now available. Yet estimates show that over half the volume of the gate array market is in designs below 10 000 gates [1], and that over half the systems using gate arrays run at clock frequencies below 25 MHz [2]. At the same time, pressures to reduce product development time have become intense. Gate arrays require a customized mask for the metal wiring and hence take weeks to months to produce.

This environment motivated the development of field programmable gate arrays (FPGA's). For the many users who do not require the speed and gate capacity of conventional arrays, it is preferable to program an off-the-shelf FPGA and have chips available within minutes after their design is completed.

A key characteristic of an FPGA is the programmable switch technology used to configure it. Many such technologies have been considered for use in FPGA's, including laser programming [3], [4], pass transistors controlled by either SRAM [5] or EPROM [6] memory cells and antifuses [7]–[10]. (See [11] for a list of recent references.)

An antifuse is an electrically programmable two-terminal device. It irreversibly changes from a high to a low resistance when a programming voltage (in excess of normal signal levels) is applied across its terminals. Antifuses offer several unique features for FPGA's, most notably a relatively low "on" resistance of 100–600 ohms and a small size. The layout area of an antifuse cell is generally smaller than the pitch of the metal lines it connects; it is about the same size as a via connecting metal lines in a mask programmed array. Nearly 750 000 antifuses can now be integrated on a single FPGA chip, facilitating the development of routing architectures approaching the flexibility and scaling potential of conventional gate arrays. Current antifuse FPGA's offer complexity equivalent to an 8000-gate conventional gate array and typical system clock speeds of 40–60 MHz.

This paper describes the technology, architecture and use of the ACT FPGA families, which are based on the PLICE(R) antifuse [8]. Similar principles will most likely apply to other antifuse-based FPGA's now emerging as the field undergoes rapid development of both antifuse process technology and FPGA architecture. The ACT 1 family [12] [13] ranges from 1200 to 2000 gates and the ACT 2 family [14] from 2500 to 8000 gates.[1] An ACT 3 family supporting higher speeds over a range of 1000–10 000 gates is planned.

Figure 1 shows the basic structure of an ACT FPGA. Rows of logic modules are interspersed with horizontal routing channels containing predefined wiring segments of various lengths and horizontal positions. Other wiring segments run vertically through the modules and across the channels. (The figure shows only a few suggestive vertical

Manuscript received December 30, 1990; revised January 30, 1992.

J. Greene is with the BioCAD Corporation, Mountain View, CA 94043.
E. Hamdy and S. Beal are with the Actel Corporation, Sunnyvale, CA 94086.

IEEE Log Number 9210746.

[1] Logic capacities in this paper are based on the stated capacity in "gates" (cells of four transistors) of an equivalent conventional channeled gate array. The estimates are derived as follows using various benchmark circuits and realistic application designs. First one determines the number of replications of a circuit that fit in a given FPGA; placement and routing are done automatically and must complete. The number of replications is then multiplied by the number of gates each instance of the circuit would require in a conventional channeled gate array, assuming the 80% gate utilization typical of such arrays. The benchmarks include data path, counter, state machine and arithmetic circuits. See [15] for details.

0018–9219/93$03.00 © 1993 IEEE

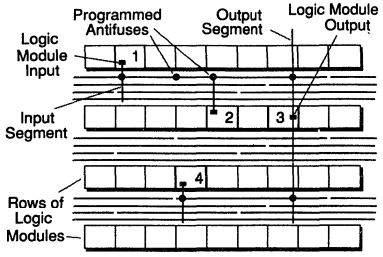


Fig. 1. Basic structure of an ACT FPGA.

segments.) Each logic module computes a single-output function of several inputs. Each module input is connected to a dedicated vertical wiring segment spanning either the channel just above or below the module. Each output signal appears on a dedicated vertical wiring segment of somewhat longer length. An antifuse is provided at each intersection of a horizontal and vertical segment, permitting them to be connected. The output of the driver, module 3, is connected by programmed antifuses to horizontal segments, which in turn are connected to input segments. In the top channel, an antifuse is used to link two adjacent horizontal segments end-to-end, making it possible to reach an input of module 1 as well as 2.

The central array of modules and channels is surrounded by input/output pads and buffers. Each I/O buffer can be connected to the internal logic through a special module near the edge of the array.

The remainder of this paper is organized as follows. Section II begins with a brief survey of the history and current status of antifuse technology in general, and then describes the physical structure, manufacture and reliability of the PLICE antifuse. Section III describes some basic principles of programmable routing applicable to any programmable switch technology, and the "segmented routing channel" model. Section IV describes the routing architecture of ACT FPGA's, which employ antifuses to implement segmented channels. Section V discusses the factors influencing the choice of the basic logic module and gives a full description of the modules used in the ACT families. Section VI briefly describes how input/output and clock distribution are handled. Section VII gives an overview of programming and testing methods, while Section VIII describes how a design is created and the design automation tools used. The design examples given in Section IX convey the speed and integration capabilities of antifuse FPGA's. By way of summary, Section X discusses how antifuse FPGA's have been used in practice and are expected to evolve in the future.

II. Antifuse Technology

A high-performance FPGA requires a programmable interconnect switch having small area and low parasitic resistance and capacitance. The switch technology must be manufacturable and reliable.

Laser-programmed switches [3], [4] can offer good performance, but require costly equipment having direct access to the unpackaged die for programming. Although the switch itself may be small, some approaches require a buffer zone around it to protect adjacent structures from being damaged by the laser.

Early PROM's and PLD's employed electrically blown fuses made of various types of material, e.g., polysilicon, platinum silicide, tungsten-titanium, and nickel-chromium. These have proven difficult to manufacture and program with sufficient reliability for state-of-the-art integrated circuits. The most common difficulty is that an inadequately blown fuse can "grow back" (reconnect) over time.

A pass transistor can also be used as a programmable switch [5]. Although this approach is widely used (see [11] for references), the significant resistance and capacitance of the switch transistor and the large area of the SRAM or EPROM cell controlling it constrain the design of the routing architecture and the performance of the circuit.

An antifuse switch technology offers these advantages for FPGA's:

- Antifuses have a significantly lower "on" resistance and parasitic capacitance than switch transistors, reducing RC delays in the routing.
- Antifuses are small, typically fitting within the minimum pitch of metal wires. This allows the use of regular and flexible routing architectures.
- Antifuses are "normally off" devices. Only the small fraction of the total that need to be turned on must be programmed (about 2% for a typical application). So other things being equal, programming is faster with antifuses than with "normally on" devices.

The concept of antifuses dates back at least to 1957, when they were considered for use in memories [16]. Antifuses developed since then fall into two categories: amorphous silicon and dielectric.

A layer of amorphous silicon placed between two metal layers undergoes a phase change when current is passed through it, becoming conductive. Devices based on this principle have been the subject of research for many years [7], [9], [10], [17]–[20], and were considered for an early FPGA design [21]. Their use has been hampered by two difficulties. First, application of a reverse current can return the amorphous silicon in a programmed antifuse to a nonconductive state. Second, even unprogrammed devices pass a small but significant current, termed *leakage current*. In a memory, where only a few bits must be active simultaneously, these problems can be avoided by careful design. They are more significant in an FPGA since the supply voltage is present across about half the antifuses at any given time.

Recent efforts at developing an amorphous silicon antifuse for FPGA's report the following results. Resistance is inversely proportional to the programming current, and is 50–110 ohms with a mode of 80 ohms at a programming current above 10 mA [22]. The capacitance contributed by each antifuse is 1.3 femtofarads in a 1.0 micron CMOS

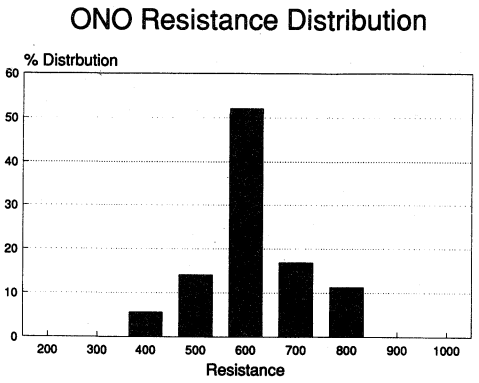


Fig. 2. Resistance distribution of PLICE antifuses programmed with 5mA.

process [22]. (This figure does not account for the capacitance of the metal lines themselves, which contribute several times this amount per antifuse.) Preprogramming leakage current is under 10 nanoamperes at 5.5 volts [10].

Dielectric antifuses consist of a layer of dielectric material placed between N+ diffusion and polysilicon. Upon application of sufficient voltage, the dielectric breaks down. Early dielectric antifuses used a single-layer oxide dielectric. The remainder of this section focuses on the "programmable low impedance circuit element" (PLICE), a multilayer oxide-nitride-oxide (ONO) dielectric antifuse developed for use in FPGA's [8]. FPGA's integrating as many as 750 000 PLICE antifuses on a single chip are currently in volume production.

Application of a 16 V programming pulse across the PLICE melts the dielectric, creating a conductive link of polycrystalline silicon between the electrodes. Typically, a single link is observed. The radius of the link increases with programming current, hence lowering the resistance. As the programming current flows, dopant atoms flow from both electrodes into the link, providing a controllable low resistance. A pulse of under one msec suffices.

For a minimum area PLICE programmed with 5 mA, the resistance is distributed as shown in Fig. 2, with a mode of 600 ohms. With a programming current of 15 mA, the mode of the distribution is shifted down to about 100 ohms. Capacitance is 10 femtofarads per antifuse in a 1.2 micron CMOS process; this includes the contribution of the polysilicon and diffusion electrodes and the metal lines used to connect them. An unprogrammed PLICE has a leakage current of about one femtoampere, so even for the largest FPGA's the total leakage is negligible.

The PLICE structure is shown in Fig. 3. A thin layer of oxide is thermally grown on top of the N+ surface, followed by low pressure chemical vapor deposition (LPCVD) nitride and the reoxidized top oxide. Use of the ONO structure tightens the resistance distribution and also improves both the yield and the reliability compared to simple oxide antifuses [23]. The PLICE adds three masks

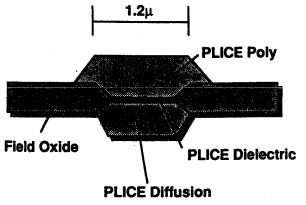


Fig. 3. PLICE antifuse structure.

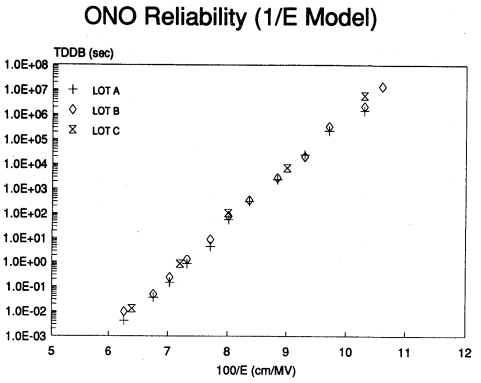


Fig. 4. Time to dielectric breakdown versus reciprocal of electric field.

to a conventional double-metal CMOS process. It can be fabricated in a typical CMOS facility using standard material, processing equipment and techniques.

Antifuse reliability must be considered for both the unprogrammed and programmed states. For an unprogrammed antifuse, with ONO less than 10 nm thick, time dependent dielectric breakdown (TDDB) reliability over 40 years is an important consideration. Ordinary accelerated testing using electric field and temperature stress has been done in order to extrapolate the dielectric's lifetime under normal operating conditions. Figure 4 shows a plot of the time-to-breakdown vs. the reciprocal of the electric field applied to the dielectric, which has been shown to be the proper model [23]. Based on this data, one can extrapolate a lifetime for an ONO antifuse of well over 40 years of normal operation at 5.5 V and 125 C.

It is equally important that the resistance of a programmed antifuse remain low during the life of the part. Single-layer oxide dielectrics are known to be susceptible to "self healing," which would increase the resistance with time. This is not the case for ONO dielectrics. Temperature-accelerated measurements reveal no intrinsic failure mechanism; the programmed antifuse resistance remains unchanged in all cases. The true lifetime of a programmed antifuse has yet to be determined since normal CMOS electromigration failures destroy the test structure first.

Of course, the reliability of the FPGA is affected by the reliability of the base CMOS process as well as the PLICE. PLICE-based FPGA product reliability studies show failure levels encountered with normal CMOS circuits. [24]

Fortuitously, the ONO dielectric is highly radiation resistant. Initial results show that products containing ONO antifuses can withstand 1 500 000 rads [25].

III. Principles of Programmable Routing

A routing architecture for an FPGA must meet two criteria: routability and speed. Routability refers to the capability of an FPGA to accommodate all the nets of a typical application, despite the fact that the wiring segments must be defined at the time the blank FPGA is made. Only the switches connecting the wiring segments can be customized (by programming) for a specific application, not the numbers, lengths or locations of the wiring segments themselves. While sufficient wiring segments for good routability must be provided, excess wiring segments will waste chip area. It is also important that the routing of an application can be determined by an automatic program with little or no manual intervention.

Propagation delay through the routing is a major factor in FPGA performance. In any efficient gate array architecture, whether mask or field programmable, it is inevitable that some nets require longer routings than others. After routing, the exact segments and switches used to establish the net are known and the delay from the driving output to each input can be computed accordingly. The resulting post-layout delays will deviate from prelayout estimates according to some statistical distribution. If the distribution of the delays for nets of a particular fanout is too broad, a user will have difficulty estimating delays when designing an application. And, of course, if the average of the distribution is too high the chip will be slow.

The delay distribution for mask-programmed arrays is sufficiently tight that variation isn't a major difficulty for the designer, since the resistance and capacitance of the metal wires are low.[2] This problem is more challenging for FPGA architectures. Any programmable switch (EPROM, MOS pass device, or antifuse) has a significant resistance and capacitance. Each time a signal passes through a programmable switch, another RC stage is added to the propagation path. For a fixed R and C, the propagation delay mounts quadratically with the number of series RC stages. This tends to increase the average of the net delays and to broaden the post-layout delay distribution.

The use of a low resistance switch, such as an antifuse, helps to keep the delay low and its distribution tight. Of equal significance is optimization of the routing architecture. Some tradeoffs involving the length of wiring segments in a channel, the area required and the the resistance and capacitance of the switch are illustrated in Fig. 5.

[2] As feature size shrinks, however, metal resistance increases faster than capacitance decreases. Even with current masked gate arrays, it is becoming important to model the distributed RC delay in the final routing to get accurate delays.

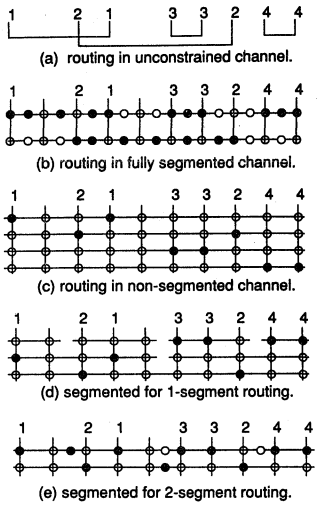


Fig. 5. Segmented routing tradeoffs. Dark circles denote programmed antifuse, open circles unprogrammed antifuses.

Figure 5(a) shows a set of nets routed in a conventional channel. With the complete freedom to configure the wiring afforded by mask programming, the positions and lengths of the horizontal wires can be customized for the particular set of nets. The "left edge algorithm" [26] shows how to do this using a number of tracks equal to the *channel density*, the maximum number of nets passing through any cut across the channel [27]. (This assumes there are no "vertical constraints" [27]. Vertical constraints do not occur in FPGA's since each signal enters or leaves the channel on its own vertical segment.)

In an FPGA, achieving this complete freedom would require switches at every cross point. More switches would be required between each two cross points along a track to allow the track to be subdivided into segments of arbitrary length [Fig. 5(b)]. Since the number of RC stages encountered by a net is proportional to its length, the delay of long nets becomes unacceptable.

Another alternative would be to provide continuous tracks in sufficient number to accommodate all nets [Fig. 5(c)]. This is the approach used in many types of programmable logic arrays and in the interconnect array portion of certain programmable logic devices (e.g., [6], [28]). Advantages are that only two RC stages are encountered and that the delay of each net is identical and predictable. However, even short nets incur the capacitance of a full track length. Furthermore the area is excessive, growing quadratically with the number of nets.

A segmented routing channel offers an intermediate approach. The tracks are divided into segments of varying lengths [Fig. 5(d)], allowing each net to be routed using a single segment of the appropriate size. Greater routing flexibility is obtained by allowing multiple adjacent seg-

ments in the same track to be joined end-to-end by switches [Fig. 5(e)]. Enforcement of simple limits on the number of segments joined or their total length guarantees that the delay will not be unduly increased.

The problem of routing a segmented channel, i.e., determining which segment(s) to assign to each net, is solvable in linear time for single-segment routing [the model of Fig. 5(d)]. When a net is allowed to use multiple segments [Fig. 5(e)], the general routing problem becomes more difficult (in particular, NP-complete [29]). However many important special cases can be solved in polynomial time, and practical problems can be routed in a few minutes on a personal computer using heuristic methods. Reference [29] gives a more thorough review of algorithms for segmented channel routing.

How does one design a segmented channel? In a conventional channel the number of tracks, or *channel width*, must be chosen to accommodate most applications. Statistical models have been developed to estimate the required channel width (e.g., [30]). In a segmented channel, both the channel width and the segment lengths and positions must be chosen carefully to suit the statistics of anticipated applications. Analytical solutions to this problem are as yet unknown, but experience indicates that even channels designed in an ad hoc manner can be quite efficient (with limited use of multiple-segment routing).

A well-designed segmented channel does not require many more tracks than would be needed in a conventional channel. This is a surprising finding given the considerable restrictions segmented routing imposes, but it can be supported both experimentally and analytically. A distribution giving the probability of occurrence of a connection as a function of length and starting point was derived from placements of 510 channels from 34 applications. A segmentation was designed for a channel with 32 tracks, taking this distribution into account and assuming that at most two segments will be used for each net. Then sets of nets with various densities were chosen randomly from the distribution, and an attempt was made to route each set in the channel. Figure 6 shows the results. In a conventional channel, any problem with density 32 or less can always be routed. A high probability of routing in the segmented channel is observed when the density is only three or four below the number of tracks. (Further details of this study are given in [31].)

An asymptotic analysis [29] has confirmed that the number of tracks required in a segmented channel grows linearly with the expected channel density, just as with conventional channels. This is true even for single-segment routing.

IV. Routing Architecture of ACT FPGAs

We now describe how segmented routing is applied in the routing architecture of the ACT 2 and 3 FPGA's. Figure 7 shows a simplified view of a four row by two column section of the array of modules. Segmented channels extend horizontally between the rows.

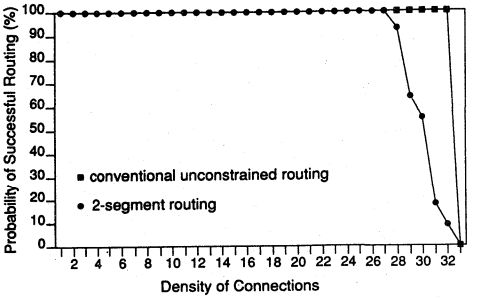


Fig. 6. Probability of successful routing in a 32-track channel vs. density of connections.

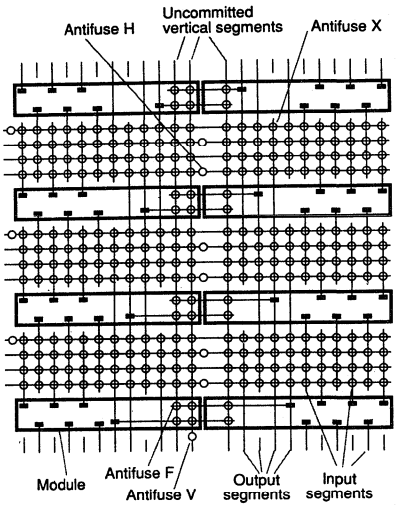


Fig. 7. Detailed view of routing architecture and modules. (Only a representative sample of input and uncommitted segments are shown.)

A module has several inputs, each of which appears on a dedicated vertical input segment. (The figure shows six inputs per module, but the actual number varies according to the function of the module; it is typically eight for a basic logic module.) The input segments each span only one channel in order to minimize their capacitance and the total wiring area.

Each module has a single output, which appears on a dedicated vertical output segment. Output segments span two channels above and two channels below the module. (Those reaching the top or bottom channel may be slightly longer or shorter). In this way a module can distribute its output to more than one nearby channel without the need for inserting more antifuses in the signal path. The added capacitance of the output segment is a small price to pay since the segment is driven directly by the module, not through any antifuses.

Each input or output segment can connect to any of the uncommitted horizontal segments in the channels it crosses through an antifuse such as the one marked X. Antifuses such as the one marked H allow connection of the horizontal segments end-to-end to support multiple-segment routing. The number of tracks per channel varies with the size of the array, according to the need for routing resources. Each channel also contains one full length horizontal segment that is grounded and another tied high so that any input can be programmed to logical 0 or 1.

Uncommitted segments are also provided in the vertical direction. These span several rows and channels, in many cases the full height of the array. Typically either one or two tracks of uncommitted vertical segments are provided per column of modules (as shown in the figure), but again this varies with array size. The uncommitted vertical segments can be joined end-to-end by antifuses such as the one marked V, and thus constitute a vertical segmented channel. However we do not refer to them as a channel since in the actual layout the vertical segments pass over the modules rather than between them (like the "feed-throughs" in a conventional channeled gate array). Each uncommitted vertical segment can connect to the horizontal segments it crosses. Also, special antifuses such as the one marked F connect a module's output and each uncommitted vertical segment passing over the module or its neighbor; two horizontally adjacent modules thus share one set of uncommitted vertical segments. These antifuses are located within the area of the module itself, but are considered part of the routing.

(The earlier ACT 1 architecture is similar to the above description except in the following respects. Additional antifuses permit certain pairs of inputs of a module to be joined; this increases routing flexibility. Also, antifuses such as the one marked F are not provided in ACT 1.)

The routing shown in Fig. 1 is typical of many nets. Note that in order to minimize the number of series RC stages, each channel's horizontal segments are driven directly by the dedicated output segment. This style of global routing is classified as a "Steiner Tree with Trunk" [32]. The chosen output segment length is sufficient to permit this routing to be used for most nets.

Although this favorable routing can be assured for all speed critical nets, another 5–10% of the nets usually must be placed with an input in some channel beyond the reach of the dedicated output segment. In this case, a suitable uncommitted vertical segment is selected to provide an alternate trunk for the channels not reached by the output. The vertical segment is driven by the output through one "F" antifuse, as shown in Fig. 8. Since this antifuse may be called upon to drive nearly the whole capacitive load of a widely dispersed net, the delay is very sensitive to its resistance. By programming these antifuses with higher currents and providing them with additional strapping contacts, the resistance is reduced to about 100 ohms, compared to 600 ohms for the other antifuses. The "F" antifuses thus require greater layout area but since there are few of them the cost is negligible.

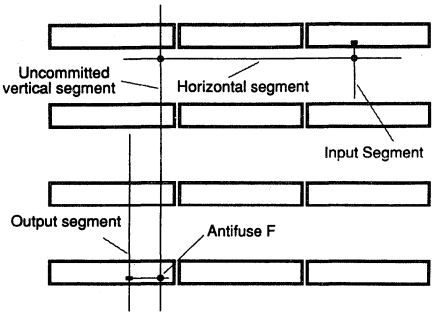


Fig. 8. Routing using directly driven uncommitted vertical segment.

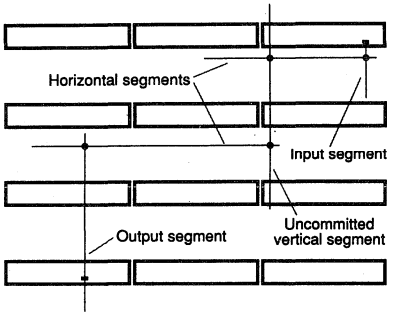


Fig. 9. Routing using indirectly driven uncommitted vertical segment.

In rare instances an uncommitted vertical segment is needed but there is none available that is accessible to the driving module's column and that extends through the required channels. A segment from some other column must be used instead. In this case the uncommitted segment is driven through one of the horizontal channels spanned both by it and the module's output, as shown in Fig. 9. This route involves an extra RC stage, and so it is reserved for nets whose speed is not critical.

Assignment of uncommitted segments in each horizontal and vertical channel to those nets that need them constitutes a segmented channel routing problem. Algorithms for solving these are discussed in [29].

V. Logic Module

FPGA architectures are often categorized by the complexity of their basic logic module, termed *granularity*. A simpler module will have less internal delay and, since each module takes less area, more of them can be provided on the chip. Furthermore, a fine-grained architecture tends to be more flexible. For example, a wide variety of functions can be built with equal efficiency out of two-input NAND gates, but eight-input NAND gates are much better at some functions than others. With simple modules there is also often more than one way to implement a function, allowing beneficial tradeoffs between area and delay.

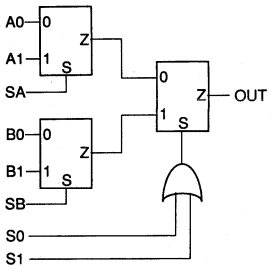


Fig. 10. ACT 1 logic module.

On the other hand, using a module that is too simple can overburden the routing network. If a function that could be packed into a few complex modules must instead be distributed among many simple modules, more connections must be made through the programmable routing network.

As a rule of thumb, an FPGA should be as fine-grained as possible while maintaining good routability and routing delay for the given switch technology. The module should be chosen to implement a wide variety of functions efficiently, yet have minimum layout area and delay.

Our approach to selecting a module began by collecting statistics on usage of various logic functions, or *macros*, in actual gate array applications. These were then used to evaluate candidates for the function of the module itself. The idea is to choose a module into which the most commonly used macros can be efficiently embedded.[3]

The ACT 1 family uses one general-purpose logic module [12], shown in Fig. 10. The module is composed of three two-to-one multiplexors and a gate. Various macrofunctions (AND, NOR, flip-flops, etc.) can be implemented by applying each input signal to the appropriate module input(s) and tying other module inputs to 0 or 1. The module can implement all combinational functions of two inputs, all functions of three inputs with at least one positive unate input, many functions of four inputs, and others ranging up to eight inputs. AND gates and OR gates up to four inputs wide are accommodated (for one or more combination of inversions at the inputs). In all, 702 distinct combinational macros are possible. Any sequential macro can be configured from one or more modules using appropriate feedback routings. Over a range of designs, an ACT 1 FPGA accommodates just over three gates of logic per module. This value is fairly consistent independent of the ratio of combinational to sequential macros in the design.

For larger chips, we can rely more strongly on the law of averages to keep the fraction of sequential macros in a design within a given range. Furthermore, statistics indicate a significant proportion of the nets driving the data input of a flip-flop have no other fanout. This motivated the use of a mixture of two specialized modules for the ACT 2 and 3 families.

[3] Given the parameters of antifuse routing, our efforts focused on modules not more complex than a typical gate array macro. Thus statistics from a design targeted for implementation in a gate array were taken to reflect designs targeted at any candidate module. The possibility of simple local optimizations (such as relocating inversions or combining a macro with a subsequent D flip-flop) was accounted for, but nothing beyond that. Ideally one would optimally reorganize the logic of each sample design for the module under consideration using logic synthesis and technology mapping, as has been done in more recent work [33], [34]. Such reoptimization becomes more important for more complex modules.

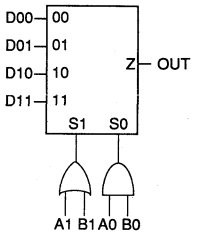


Fig. 11. "C" module.

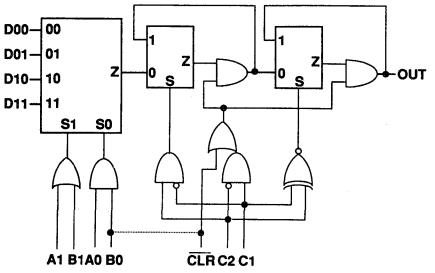


Fig. 12. "S" module. In ACT 2 chips, the B0 and $\overline{CLR}$ inputs are joined.

The *C module* (Fig. 11) is a modified version of the ACT 1 module reoptimized to better accommodate high-fan-in combinational macros, notably some five-wide AND and OR gates, though with some loss in ability to build sequential functions. It is composed of a four-to-one multiplexor and two gates, and can implement a total of 766 distinct combinational macros.

The *S module* consists of a front end equivalent to the C module followed by a sequential block built around two latches; Fig. 12 gives a functional diagram. The sequential block can be used as a rising- or falling-edge D flip-flop, or a transparent-high or transparent-low latch, by tying the C1 and C2 inputs to a clock signal, logical zero or logical one in various combinations. For example, tying C1 to 0 and clocking C2 implements a rising-edge D flip-flop. The block can also be set permanently transparent (by tying C1 to 1 and C2 to 0), making the S module equivalent to a simple C module with a small additional delay. Figure 13 shows the function categories that can be implemented using the S module. Toggle or enabled flip-flops can be built using the combinational front end in addition to the D flip-flop. Other less commonly used flip-flops (such as JK or set/reset) not supported by the sequential block can be configured from one or more C or S modules using external feedback connections.

In ACT 2 chips, the clear input is joined to the B0 input of the front end to limit the total number of inputs at some

minor cost in logic flexibility. ACT 3 FPGA's will provide independent clear and B0 inputs for greater regularity, primarily for the benefit of logic synthesis programs.

A chip with an equal mixture of C and S modules provides sufficient flip flops for most designs, plus an extra margin to assure flexibility in placement. Over a range of designs, the ACT 2 mixture provides about 1.4–2.0 times the logic capacity per module of the ACT 1 module.[4]

Figure 14 shows a typical critical path in a state machine implemented using four C modules and one S module. The ability to fit a five-wide gate in one C module saves one routed net compared to an ACT 1 module implementation. The use of the S module eliminates one more routed net (which is now internal to the module in the second stage).

The choice of module also greatly influences the routability of the chip. Because each input is accessible from only one of the two channels adjacent to a module, it would appear that routability is degraded compared to a conventional *double-entry* gate array cell, which a signal can enter from either channel at the convenience of the router. However, there is nearly always more than one way to implement a macro. For an N-input macro, there may be as many as 2^N different assignments of the input signals to the two channels. This corresponds to full double-entry symmetry. Even if not all of the 2^N assignments are possible, a sufficiently sophisticated router can take advantage of whatever flexibility exists.

For the C module, a given signal can be routed from either side of the module an average of 70% of the time (weighted by macro usage), quite sufficient for good routability. This figure is affected by both the module function and the assignment of the module inputs to sides. It is another important criterion in selecting a module function.

Finally, we note that the user is insulated from the details of the module function. Logic is entered as a schematic using a familiar gate-array-style macro library or a logic synthesis program. Software automatically determines how each module should be connected to realize the desired macro. (Methods of design entry are described more fully in Section VIII.)

VI. Input/Output and Clock Distribution

Each I/O pad has an adjacent bidirectional buffer, which connects to the array through an associated I/O module. These modules fit in the outer columns and rows of the array next to the logic modules, and interface to the routing channels in the same way. Each I/O module has inputs for outgoing data and enable, which are sent to the buffer, and an output for incoming data which is driven from the pad.

[4]It is interesting to note the related work of Rose *et al.* [33]. They found that the addition of a D flip-flop to a module increased the logic capacity by a factor of 1.4–2.3. The results are not exactly comparable, however, since their model differs from ours in several respects: their module is a three- or four-input look-up table; they add flip-flops to all modules rather than to half of them; our results include a slight change in the combinational capabilities of the module as well.

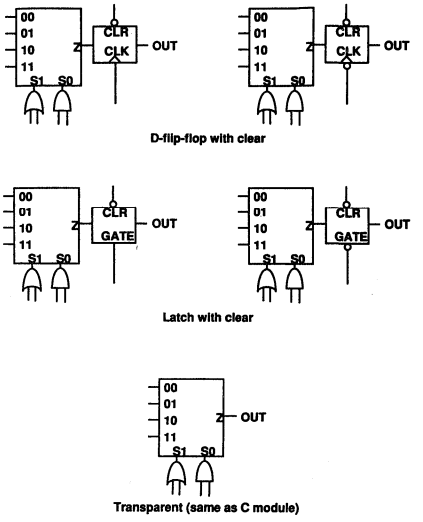


Fig. 13. "S" module function categories.

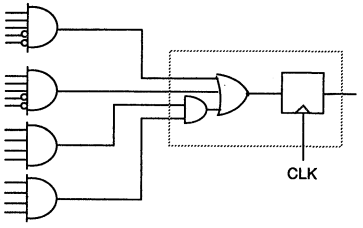


Fig. 14. Sample critical path in an ACT 2 FPGA.

Thus the I/O module can be programmed to provide input, output, tristate or bidirectional capability.

In order to keep clock-to-output delay to a minimum, the ACT 2 family has a dedicated transparent-high latch for each pad, which can serve as the slave stage of a flip-flop. The latch is controlled by a gate input to the I/O module. If flow-through operation is desired, the gate is simply tied high to make the latch transparent. A dedicated transparent-low latch is similarly provided on each input path. The polarities of the input and output latches are chosen so they can be combined with each other, and possibly with other internal latches or flip-flops, to form a path that is functionally equivalent to a chain of rising-edge flip-flops.

Clock signals present unusual requirements: they have high fanout and yet allow only minimal delay and skew. Rather than attempt to route clocks like any other signal, dedicated clock distribution networks are provided. Each network is driven directly from a special input pad for high speed. The signal passes through a buffer tree, and appears on a dedicated full-length horizontal track in each channel. Thus the network reaches every logic and IO module in the

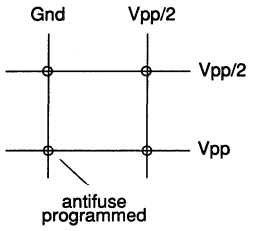


Fig. 15. Application of programming voltage.

array. The ACT 2 family provides two clock distribution networks. Each network can be driven from user-defined internal logic as well as from an input pad.

To minimize the capacitive load on the clock network, antifuses are provided to connect the clock track to only certain module inputs, specifically the clock inputs of the S and IO modules and a subset of the combinational inputs on all the logic modules. Skew is further reduced by having the automatic placement program attempt to balance the loading on each branch of the distribution tree.

The clock inputs to the S module, like other inputs, can also be connected to the uncommitted horizontal segments. This accommodates designs with several local asynchronous clocks routed in the normal fashion.

VII. Programming and Testing

One of the great puzzles in the development of antifuse FPGA's was to find an efficient way to address each of the two-terminal antifuses uniquely. Diodes in series with each antifuse would allow unique addressing, but block signal flow. Early schemes required the use of an individual control line for each routing track, doubling the number of lines required in each direction [21]. However, methods for programming and testing antifuse FPGA's that use only a few control lines for an entire channel have been developed [12], [14]. Although a full description is beyond the scope of this paper, the following explanation conveys the basic concepts.

Consider an array of antifuses at the intersection of some horizontal and vertical segments, as shown in Fig. 15. An antifuse is programmed by applying a programming voltage, Vpp, across it. This is done by precharging all segments to an intermediate voltage of about Vpp/2. Then a selected vertical segment is grounded and a selected horizontal segment is driven to Vpp. Other segments are left floating at Vpp/2. Only the single antifuse at the intersection of the selected segments sees the full Vpp.

We now show how the segments can be selectively driven. Figure 16 illustrates the *series pass transistor addressing* method. A sample of horizontal and vertical segments are shown. Each pair of adjacent segments in the same track is connected by a pass transistor. (In tracks containing uncommitted segments there is already an antifuse joining adjacent segments, and so the transistor is hooked in parallel with the antifuse.) These transistors are

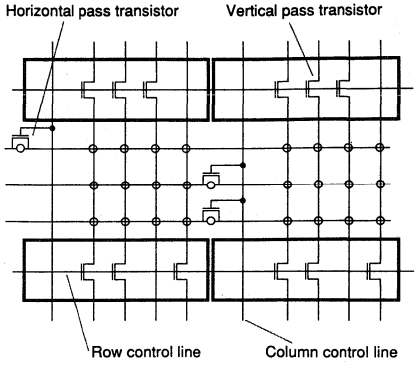


Fig. 16. Pass transistor addressing.

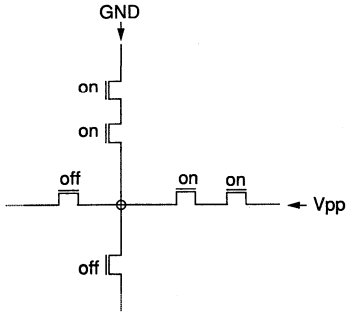


Fig. 17. Programming an antifuse between a horizontal and a vertical segment.

used only for programming and testing and are all shut off during normal operation of the programmed part. The transistors in each row of modules are gated by a control line running along the row, and the transistors in each column of modules are gated by a control line running along the column. These lines in turn are driven by special logic at the periphery of the array. Note that only one control line per row or column is required, regardless of the number of tracks.

Figure 17 shows how this circuitry is used to program an antifuse at the intersection of a horizontal and vertical segment. The vertical track containing the antifuse is driven to ground from one end by special logic at the periphery of the array. The pass transistors in all rows from the driving periphery to the antifuse are turned on. The horizontal track is driven to Vpp in a similar manner. Figure 18 shows how an antifuse between two adjacent segments in the same track is programmed. All columns of pass transistors except the one bypassing the antifuse are turned on, and the track is grounded at one end and driven to Vpp at the opposite end.

In certain cases *direct addressing* circuitry offers a favorable alternative to series pass transistors. Figure 19 shows how this method can be used to address horizontal seg-

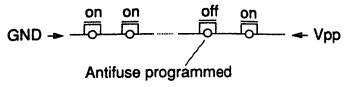


Fig. 18. Programming an antifuse between two adjacent segments in the same track.

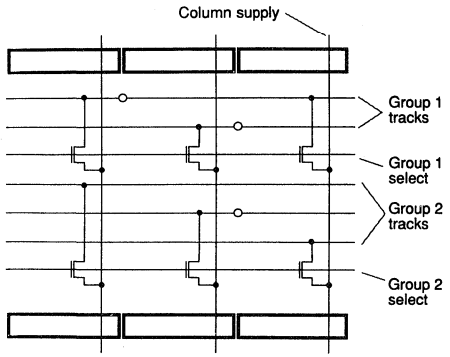


Fig. 19. Direct addressing circuitry.

ments. Column supply lines run vertically through the array. Each segment along a track can be connected through its own addressing transistor to a supply line. These transistors are gated by a horizontal select line. Each select line serves all the segments in a group of one or more adjacent tracks in the same channel. Activating one supply line and one select line uniquely addresses a particular horizontal segment. The number of segments in a channel that can be addressed is limited by the number of supply lines times the number of group select lines. It follows from this that the ratio of tracks to select lines is at most the average segment length. Also note that the supply path from the periphery to the segment includes only one transistor, rendering the programming current independent of the position of the segment and the segmentation of its track. Thus the direct addressing method is most efficient for irregularly segmented channels with long segments.

ACT 1 chips use only the pass transistor method. ACT 2 and 3 chips use the pass transistor method for the vertical tracks, which contain many short input segments, and the direct address method for the horizontal tracks.

In either scheme, some care is required to assure that a unique antifuse is addressed once other antifuses have already been programmed. Figure 20 gives an example of how improper addressing might allow programming current to divert along a *sneak path* through previously programmed antifuses 3 and 4, programming antifuse 1 instead of the intended antifuse 2. Fortunately, we are not interested in programming an arbitrary pattern of antifuses (an FPGA is not a PROM!). For example, we are not called upon to program a pattern connecting two outputs together since this does not form a useful net. For the relevant patterns, it can be proved that there is always an order

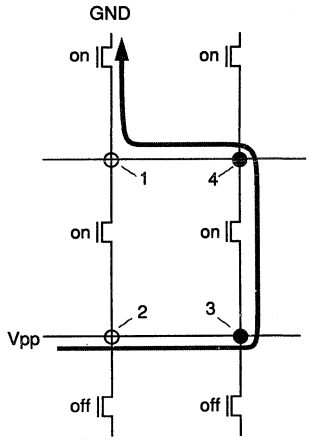


Fig. 20. A sneak path. The arrow shows the path of programming current through antifuse 1, bypassing antifuse 2.

in which the antifuses can be programmed such that sneak paths *never* occur.

Special care is also required to protect the module circuitry from the voltages present on the segments during programming. Transistors that are in contact with routing segments must be specially designed to withstand the programming voltage.

Next we turn to testing. This takes place in three phases: before, during and after programming. Preprogramming tests check for shorted or open segments, shorted or even weak antifuses, and proper module and I/O operation. Continuity of the segments is easily verified by turning on all pass transistors, using the peripheral circuits to drive the segment at one end of a track and read the segment at the other end. Testing for the absence of shorts between segments in adjacent tracks is done in a similar way by applying a pattern of alternating zeros and ones. Weak antifuses can be screened out by applying the proper stress voltage (higher than normal operating voltage but lower than Vpp) across groups of antifuses in parallel using the programming circuits. Breakdown of an antifuse is detected by passage of excessive current.

To verify the functionality of the modules, we need to apply test vectors to their inputs and read their outputs. A vector can be applied simultaneously to all modules by turning on all vertical pass transistors and applying the vector to the vertical tracks from the peripheral logic. A simple row-select and column-sense scheme conveys the output of each module in turn to an output pad for monitoring. (This same circuitry allows users to probe internal nodes during normal operation, as described in the next section).

Proper closure of the programmed antifuses is verified during programming by sensing the passage of programming current. A complete test for unintended connections between any two segments can be done after programming

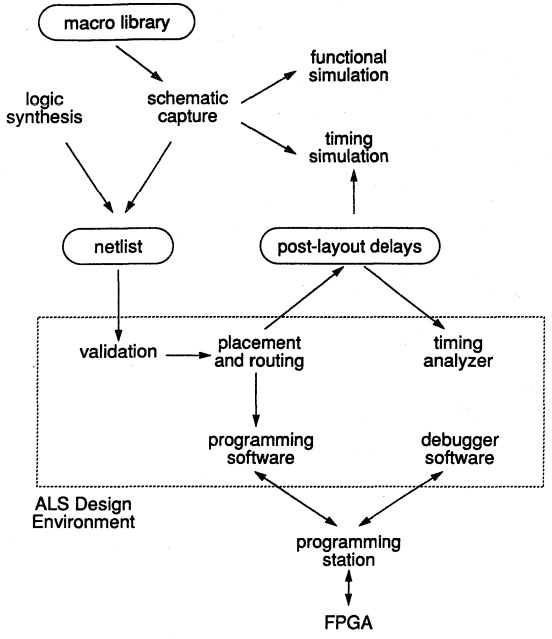


Fig. 21. Design process.

using the programming circuitry to precharge, drive and read the segments. This is true despite the fact that it is no longer possible to address each individual antifuse uniquely once programming commences. The solution to this paradox is that detection (though not location) of shorts can be accomplished simultaneously for many antifuses in parallel.

Taken together, these tests insure correct functioning of the programmed part. (Further details of testing in ACT 1 FPGA's are found in [35].)

VIII. Design Process and Software

Figure 21 diagrams the process of embedding a design in an ACT FPGA. This process begins with capture of the design in a computer readable format. Currently, most users enter their designs as schematics built of macros from a library. Any of several standard schematic capture programs can be used. Since the ACT 1 and C modules have about the same complexity as a typical gate array library macro, the problem of selecting which groups of macros should share one module is avoided; each macro is usually assigned its own module. The larger capabilities of the S module are handled by software that automatically combines a flip-flop and a preceding combinational macro into one S module where possible. This process is transparent to the user and does not require modifying the schematic.

Another increasingly popular alternative is to enter designs in terms of Boolean equations, state machine descriptions, or functional (rather than structural) schematics. Different portions of a design can be described in different ways, compiled separately, and the results merged according to a top-level hierarchical schematic. Various industry standard formats are supported.

High-level synthesis tools such as MIS-II [36] or Synopsys [37] can be used with ACT FPGA's by providing a suitable library. This can encompass only the standard schematic library macros, or the complete set of all 700 or so functions embeddable in the module can be included. Recent research has investigated other more efficient ways to allow synthesis tools access to the full flexibility of the module. A method for rapid searching of large libraries using Boolean matching is given in [38]. Other approaches take advantage of the multiplexer structure of the module, using either binary decision diagrams [39] or "if-then-else DAG's" [40].

Several guidelines are suggested for reliable logic design with FPGA's. The general goal is to make proper circuit function independent of shifts in timing from one part to the next.

- Use synchronous logic design where possible.
- Avoid gated clocks, using enabled flip-flops instead.
- Avoid race conditions.
- Limit fanout by buffering.

These guidelines are the same as those for users of mask programmed gate arrays [41]. In addition, designs for ACT

	logic module utilization	pins per logic module
8000 gate FPGA		
a) design done in an 8K TI gate array	99.5%	4.28
b) 32 bit data path, 16x16 mult, state machine	99.4	4.30
c) 2901 ALU (x4)	98.1	4.57
d) DMA controller (x3)	97.1	3.99
e) asynchronous serial ECC	97.0	4.78
f) pipelined fixed point mult, div, sqrt	94.5	3.37
g) state mach., mult/add, datapath, counter	92.7	4.34
h) color crt controller (x3)	87.3	3.83
i) 32 bit data path with sum, compare (x3)	86.8	5.25
j) 40 bit floating point adder/subtractor	86.7	4.33
4000 gate FPGA		
k) 16 bit datapath, 16x16 mult, state machine	98.1	4.86
l) 2901 (x2)	93.2	4.68
m) DRAM, DMA & SCSI controllers, UART	92.6	4.73

Fig. 22. Statistics for some placed and routed designs.

FPGA's must use multiplexers rather than tristate drivers since internal tristate is not supported.

Once the design has been entered, it can be simulated either functionally or using prelayout delay estimates. The netlist is checked by a validation program for problems such as undriven nets, outputs shorted together, excessive fanout, etc.

The next step is to map the netlist into the ACT architecture. The placement problem (selecting a module for each macro) is similar to that for a conventional gate array, with a few additional considerations. I/O pins not constrained by the user must be assigned to locations that minimize delay and routing congestion inside the chip. Macros must be placed in modules of the appropriate type (C or S). Macros hooked to a clock network should be distributed so as to balance the load on the network's branches. Routing congestion within each horizontal channel must be limited. Whenever possible, the dedicated output segments should be used in preference to the uncommitted vertical segments, especially for nets identified by the user as speed critical. Uncommitted vertical segments must be assigned to any remaining nets. Routing of the segmented channels is done as previously described. The automatic placement and routing software takes 45–60 min to complete an 8000-gate FPGA on a 68030 microprocessor workstation.

Once placement and routing are completed, the propagation delay to each input pin is estimated. The calculation accounts for the module internal delay as well as the delay through each RC stage in the routed net. The estimates are about as accurate as what would be obtained with a SPICE circuit simulation. The delays can be back-annotated to a simulator, or checked with a static timing analyzer program.

A list of antifuses to be programmed is generated and downloaded to a programming station into which the FPGA is plugged. Programming time is about 5–10 min, depending on the size of the FPGA and the design. The station allows up to four chips to be programmed simultaneously in this time. The standard tests applied before, during and after programming, described in Section VII, assure correct function of the programmed part. No design-specific test vectors are required from the user.

The programming system can also access the previously described test circuitry inside the programmed FPGA, causing the FPGA to sense any selected module output and present it on an external pin in real time. This virtual probe can be used and its address changed even as the chip is operating in the user's system. The probe provides a useful back-up when simulation is difficult or impossible, such as for highly asynchronous designs or when the design must accommodate a poorly documented interface. Debugger software is also provided to enable the programmer to be used as a functional tester, presenting stimuli at the FPGA's pins and reading data back via the pins and the probe.

IX. Density and Performance

In this section we report results with various design examples to convey the density and performance of the ACT 2 architecture.

The logic capability of an FPGA depends on more than the number and capacity of it modules. The fraction of the modules that can be used without running out of routing capacity must also be considered. A simple design such as one long shift register can use all modules with very little routing, but realistic designs are more demanding.

Figure 22 summarizes results for a variety of designs in 8000- and 4000-gate FPGA's having 1232 and 649 logic modules, respectively. The designs were placed and routed automatically with no manual intervention. The table includes the number of routed pins per logic module to demonstrate the realistic burdens the designs impose on the routing. Nearly all designs with module utilization under 85%, most designs with utilization under 95%, and many with utilization up to 100%, route completely.

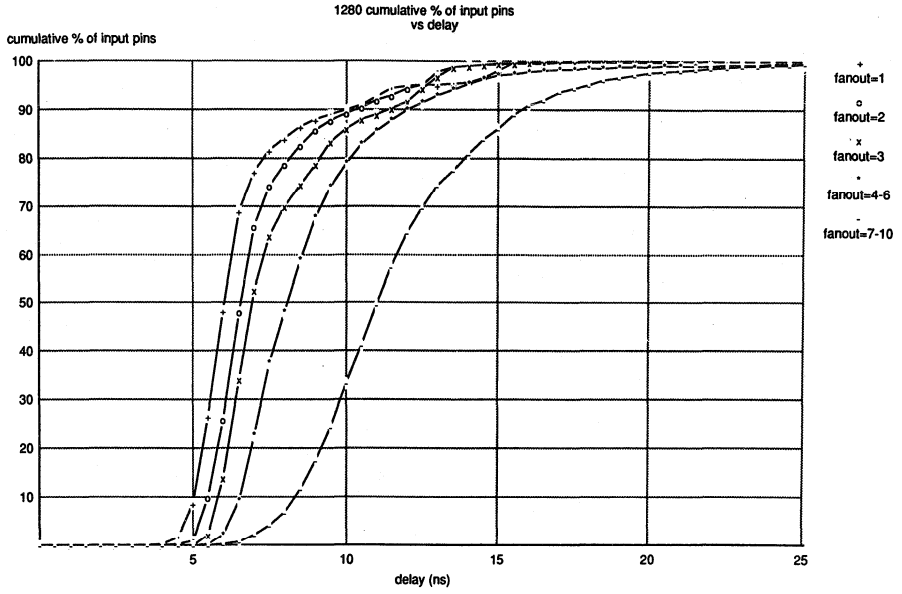


Fig. 23. Cumulative distribution of pin delays grouped by the fanout of their net. Delays were computed by the postlayout extraction program for conditions of 5V, 25C, and worst-case processing. Data based on 29 designs in an 8000-gate FPGA (A1280–1). The designs were automatically placed and routed without manual intervention.

How successful is the architecture in controlling routing delay? Figure 23 shows the distribution for the routing delay to each input pin for various ranges of net fanout. The figure includes all nets in the design. Pins on nets identified by the user as speed critical would generally be among the fastest 30%, and nearly always among the fastest 90%. Especially for low fanouts and critical nets, the routing delay distributions are tight, offering the user predictable delays.

Finally, we give some results regarding some basic circuit blocks. Table 1 shows the number of C and S modules used, number of levels of modules in the critical path, and nominal speed at 5V and 25C including routing delay. The counters are binary, with synchronous parallel load and asynchronous clear. The accumulator consists of an adder feeding into a register with asynchronous clear. Area-delay tradeoffs are possible using different types of adders. The state machine handles arbitration for a DRAM controller in a 68030 microprocessor system. It has 3 state bits.

Table 1. Implementations of Some Typical Subcircuits

Example	Modules	Levels	Clock period (nsec)
8-bit counter, 1-cycle load	16	2	17
16-bit counter, 1-cycle load	50	2	18
8-bit counter, 2-cycle load	40	1	12
16-bit counter, 2-cycle load	80	1	12
8-bit ripple-carry accumulator	17	8	60
8-bit carry-select accumulator	40	3	28
16-bit carry-select accumulator	89	3	33
state machine	16	2	16

X. The Practical Impact of Antifuse FPGA's

The major impact of FPGA's is the ability to get a design into production quickly and cheaply, even if iterations are required. This is demonstrated in the following case history from commercial practice.

Example 1: A digital video signal processing system was designed for the consumer market. Four 2000-gate FPGA's were used to integrate registers, multipliers, adders and digital phase-locked-loops; the pipelined data paths are 12–16 bits wide, and operate at a 16MHz clock rate. It is estimated that the use of FPGA's saved two to three months in product development time. Subsequently, it became necessary to add

features to the product. A single 8000-gate FPGA replaced the four 2000-gate FPGA's and provided the new circuitry.

A surprising fraction of conventional gate array designs (estimates indicate about half) never reach high-volume production. This may be because either the lifetime of the product is short or because new features or other changes are required. In such cases use of FPGA's for all production saves the nonrecurring engineering costs associated with developing a mask programmed gate array.

For the lucky few whose products are required in high volume, the FPGA design can be transferred to a conventional gate array. Due to the small granularity of the module, an antifuse FPGA netlist can be efficiently converted to a gate array library by substituting the appropriate realization for each macro in the FPGA library. Test vectors can often be generated automatically. Several vendors support this conversion path. The design in Example 1 was eventually converted to an 8000-gate conventional array.

Even when designs require the greater speed or density of conventional arrays, use of FPGA's for functional prototyping accelerates product development. This was true in the next example.

Example 2: A multiprocessor file server for a PC network was under development. The system employed multiple 80486 microprocessors, banks of DRAM, and multiple 64- and 32-bit busses. Several 8000-gate FPGA's implemented the random logic in a functional prototype using synchronous logic design. Since the logic was composed of clearly defined subblocks, principally state machines and simple datapaths, partitioning of the logic among the FPGA's was easy. For this complex system, the design process was incremental and somewhat experimental. Because it was anticipated that the design would be modified or extended several times, a fast design flow was necessary. The logic for each FPGA was synthesized with a Synopsys hardware description language (HDL) compiler using an Actel macro library, and then checked with an HDL simulator. Each design was automatically placed and routed, and resimulated with postlayout delays. This methodology allowed software to be developed and tested concurrently with the completion of the logic design.

We conclude with some comments on the future evolution of antifuse FPGA's. Research into antifuse technologies is gaining momentum due to the commercial importance of FPGA's. Reductions in the parasitic resistance or capacitance will help improve speeds. Reductions in the programming voltage or current will reduce the overhead area required for pass transistors and other programming circuitry, and hence cost. On the architecture front, better logic modules and improved methods of designing segmented routing channels may be developed. FPGA architectures could be specialized for certain classes of applications. Of course, antifuse FPGA's will also directly benefit from anticipated improvements in the underlying CMOS technology. Thus one would expect antifuse FPGA's to evolve more rapidly than conventional arrays, reducing the current gap between them. Ultimately it is possible that FPGA's will supplant conventional gate arrays for application-specific logic to the same extent that EPROM's have supplanted ROM's for application-specific memory.

ACKNOWLEDGMENT

The authors would like to thank C.-L. Chan, R. Gopisetty, S. Kaptanoglu, D. McCarty, W. Miller, W. Shu, and T. Whitney for their help in preparing data for this paper; J. Birkner for his kind discussions with the authors regarding the programming currents for the resistances reported in [22]; and A. Haines, D. How, J. Schlageter, and the reviewers for their suggested improvements to the manuscript.

REFERENCES

[1] G. Worchel, "Analysis of the CMOS gate array merchant market," *In-stat Services,* Mar. 1991.
[2] "Programmable logic user study," *Electronic Eng. Times*, May 1990.
[3] J. F. Smith, *et al.*, "Laser-induced personalization and alterations of LSI and VLSI circuits," in *Proc. 1st Int. Laser Processing Conf.*, Anaheim, Calif., Laser Institute of America, Nov. 16, 1981.
[4] D. Allen and R. Goldenberg, "Design aids and test results for laser-programmable logic arrays," in *Proc. Int. Conf. Computer Design*, pp. 386–390, 1990.
[5] W. Carter *et al.*, "A user programmable reconfigurable gate array," in *Proc. Custom Integrated Circuits Conf.*, pp. 233–235, 1986.
[6] S. Wong, H. So, J. Ou, and J. Costello, "A 5000-gate CMOS EPLD with multiple logic and interconnect arrays," in *Proc. Custom Integrated Circuits Conf.*, pp. 5.8.1–5.8.4, 1989.
[7] L. Gerzberg. U.S. Patent 4,590,589, 1986.
[8] E. Hamdy, J. McCollum, S. Chen, S. Chiang, S. Eltoukhy, J. Chang, T. Speers, and A. Mohsen, "Dielectric based antifuses for logic and memory ICs," *IEDM Tech. Digest*, pp. 786–789, 1988.
[9] "ICC takes the antifuse one step further," *Electronics Mag.*, p. 87, Mar. 1989.
[10] R. Whitten, R. Bechtel, M. Thomas, H.T. Chua, A. Chan, and J. Birkner, European Patent Application No. 90309731.9, May 9, 1990.
[11] J. Rose, A. El Gamal, and A. Sangiovanni-Vincentelli, "A classification and survey of field-programmable gate array architectures," *Proc. IEEE*, vol. 81, no. 7, July 1993..
[12] A. El Gamal, J. Greene, J. Reyneri, E. Rogoyski, K. El-Ayat, and A. Mohsen. "An architecture for electrically configurable gate arrays." *IEEE J. Solid-State Circuits*, vol. 24, no. 2, pp. 394–398, Apr. 1989.
[13] K. El Ayat, *et al.*, "A CMOS Electrically Configurable Gate Array," *IEEE J. Solid-State Circuits*, Vol. 24, No. 3, pp. 752–762, June 1989.
[14] M. Ahrens, *et al.*, "An FPGA family optimized for high densities and reduced routing delay," in *Proc. Custom Integrated Circuits Conf.*, 1990.
[15] B. Osann and A. El Gamal. "Compare ASIC capacities with gate array benchmarks," *Electronic Design*, pp. 93–98, Oct. 13, 1988.
[16] U.S. Patent 2,784,389, 1957.
[17] B. Roesner, U.S. Patents 4,424,579 and 4,442,507, 1984.
[18] Holmberg, *et al.*, U.S. Patents 4,499,557, 1985 and 4,599,705, 1986.
[19] Lim *et al.* ,U.S. Patent 4,569,121, 1986. Stacy et al. U.S. Patent 4,569,120, 1986.
[20] H. Stopper, *et al.* ,U.S. Patent 4,847,732, 1989.
[21] H. Graham and D. Seltz. "Electronically programmable gate array having programmable interconnect lines," U.S. Patent 4,786,904, Nov. 22, 1988.
[22] J. Birkner, A. Chan, H.T. Chua, A. Chao, K. Gordon, B. Kleinman, P. Kolze, and R. Wong, "A very high-speed field programmable gate array using metal-to-metal antifuse programmable elements," in 1991 IEEE Custom Integrated Circuits Conf., San Diego, CA, May 12, 1991.

[23] S. Chiang, R. Wang, J. Chen, K. Hayes, J. McCollum, E. Hamdy, and C. Hu, "Oxide-nitride-oxide antifuse reliability," *Int. Reliability Physics Symp.*, pp. 186–192, Mar. 1990.
[24] S. Chiang and K. Hayes, Act 1010/1020 Reliability Report, Actel Corporation, Sunnyvale, CA, April 1990.
[25] Preliminary data from a military system manufacturer. Test done on total dose Gamma Irradiation Survivability Test.
[26] A. Hashimoto, J. Stevens, "Wire routing by optimizing channel assignment within large apertures," in *Proc. 8th IEEE Design Automation Workshop*, 1971.
[27] M. Lorenzetti and D. Baeder, "Routing," in *Physical Design Automation of VLSI Systems*,B. Preas and M. Lorenzetti, Eds. Benjamin Cummings, Chapter 5, 1988.
[28] C. Marr, "Logic array beats development time blues," *Electronic System Design Mag.*, pp. 38–42, Nov. 1989.
[29] A. El Gamal, J. Greene, and V. Roychowdhury, "Segmented channel routing is nearly as efficient as channel routing (and just as hard)," in *Proc. Conf. Advanced Research in VLSI*, Santa Cruz, CA, Mar. 1991.
[30] A. El Gamal, "Two dimensional stochastic model for interconnections in master slice integrated circuits," *IEEE Trans. Circuits Syst.*, CAS-28, pp. 127–138, Feb. 1981.
[31] J. Greene, V. Roychowdhury, S. Kaptanoglu, and A. El Gamal, "Segmented channel routing," in *Proc. ACM/IEEE Design Automation Conf.*, June 1990.
[32] B. Preas and P. Karger, "Placement, assignment and floorplanning," in *Physical Design Automation of VLSI Systems*, B. Preas and M. Lorenzetti, Eds. Benjamin Cummings, Chapter 4, 1988.
[33] J. Rose, R. Francis, D. Lewis, and P. Chow, "Architecture of programmable gate arrays: the effect of logic block functionality on area efficiency," *IEEE J. Solid State Circuits*, vol. 25, no. 5, pp. 1217–1225, Oct. 1990.
[34] J. Kouloheris and A. El Gamal, "FPGA performance versus cell granularity," in *Proc. Custom Integrated Circuits Conf.*., pp. 6.2.1–6.2.4, May 1991.
[35] K. El Ayat, A. Haines, and K. Hayes, "Testing antifuse-based FPGAs," *Electronic Eng. Times*, May 15, 1989.
[36] R. Brayton, R. Rudell, A. Sangiovanni-Vincentelli, and A. Wang, "MIS: A multiple-level logic optimization system," *IEEE Trans. CAD*, Nov. 1987.
[37] R. Rudell and R. Segal, "Logic synthesis can help in exploring design choices," *1989 Semicustom Design Guide*, CMP Publications, Manhasset, NY.
[38] S. Ercolani and G. de Micheli, "Technology mapping for electrically programmable gate arrays," in *Proc. 28th ACM/IEEE Design Automation Conf.*, pp. 234–239, 1991.
[39] R. Murgai, Y. Nishizaki, N. Shenoy, R. Brayton, and A. Sangiovanni-Vincentelli. "Logic synthesis for programmable gate arrays," in *Proc. 27th ACM/IEEE Design Automation Conf.*, 1990.
[40] K. Karplus, "Amap: a technology mapper for selector-based field-programmable gate arrays," in *Proc. 28th ACM/IEEE Design Automation Conf.*, pp. 244–247, 1991.
[41] *HCMOS Gate Array Databook and Design Manual*, LSI Logic Corp., Oct. 1986.

Jonathan Greene (Member, IEEE) received the Sc.B. degree in biology from Brown University in 1979, and the Ph.D. degree in electrical engineering from Stanford University in 1983.

He then worked on various aspects of computer-aided IC design at Hewlett-Packard Laboratories and LSI Logic Systems Research Lab. From 1986 to 1990 he was with Actel Corporation, where he helped develop the architecture and software for their field-programmable gate arrays and became Director of System Architecture. He is currently with BioCAD Corp., Mountain View, Calif., applying computer-aided design and information theoretic techniques to the process of drug discovery.

Esmat Hamdy (Senior Member, IEEE) was born in Cairo, Egypt, of February 2, 1950. He received the B.Sc. and M.Sc. degrees from Cairo University in 1971 and 1975, respectively, and the M.A.Sc. and Ph.D. degrees from the University of Waterloo, Ontario, Canada, in 1977 and 1980, respectively, all in electrical engineering.

From 1971 to 1975 he served an Instructor in the Department of Electrical Engineering, Helwan University, Helwan, Egypt. From 1975 to 1980 he was a Research and Teaching Assistant in the Department of Electrical Engineering, University of Waterloo, where he was engaged in the analysis, modeling, and design of high-density bipolar and MOS integrated structures for LSI/VLSI technologies. He joined Intel Corp., Aloha, Oreg., in 1981, where he was a Senior Staff Engineer/Project Manager in the Technology Department involved in device physics and circuit design of the world first 256K CMOS DRAM and submicrometer CMOS for Microprocessors. In Actel he managed and coinverted the PLICE antifuse technology used in the world's first antifuse FPGA. He has authored or coauthored over 20 papers on LSI/VLSI circuits including contributions to IIL, single-device-well (SDW) MOSFET's CMOS latch-up and PLICE antifuse. He is the winner of the Best Student paper at the 1979 IEDM meeting. He also has 10 pending or granted patents.

Sam Beal received B.S., M.S., and Ph.D. degrees in electrical engineering from Southern Methodist University, Dallas, Tex., in 1975, 1977, and 1979, respectively, having done research in AlGaAs solar cells.

He joined Texas Instruments in 1980, working in the CMOS Technology Department, where his responsibilities included CMOS process development and ASIC design development. In 1984 he assumed responsibility for Texas Instruments' ASIC design center in Santa Clara, Ca. He joined Actel in 1988 as Applications Engineering Manager. He is currently Applications and Product Planning Manager for Development Systems.

Oxide-Nitride-Oxide Antifuse Reliability

Oxide-Nitride-Oxide Antifuse Reliability

Steve Chiang, Roger Wang, Jacob Chen, Ken Hayes, John McCollum, Esmat Hamdy, Chenming Hu*

Actel Corp. 955 E. Arques Ave. Sunnyvale, CA 94086
Phone:(408)739-1010

*Department of Electrical Engineering and Computer Science U.C. Berkeley, Berkeley, CA 94720

Abstract

Compact, low-resistance oxide-nitride-oxide antifuses are studied for TDDB, program disturb, programmed antifuse resistance stability, and effective screen. ONO antifuse is superior to oxide antifuse. No ONO antifuse failures were observed in 1.8 million accelerated burn-in device-hours accumulated on 1108 product units. This is in agreement with the 1/E field acceleration model.

Introduction

Field programmable gate arrays has been a fast growing field only recently [1]. The key to their configurability is the development of a programmable interconnect element. This element should have small area, low post programming resistance, and be reliable. Many known interconnect elements have been used, including SRAMs, EPROMs, and EEPROMs. Problems encountered using these elements are: large area, high resistance, or inefficient utilization due to circuit complexity. The antifuse approach, however, has some unique and attractive features. Since it is only a two terminal device, the area required is small, and the simple two terminal resistor structure allows simple and efficient routing schemes [2]. The programmed antifuse has very low resistance. It was found that oxide-nitride-oxide (ONO) antifuses have a lower and tighter resistance distribution than that of oxide antifuses (Fig. 1). The choice of antifuse material has further improved both the yield and the reliability over that of oxide antifuses. In addition, ONO is highly radiation resistant. Initial evaluation results indicate that products containing ONO antifuses can withstand 1.5 million rads [3]. The technology and performance characteristics of the ONO antifuse has been previously described [4]. In this paper, we will report the reliability characterization of the ONO antifuse.

We will discuss three different types of antifuse reliability. The first is that the unprogrammed antifuse has to survive a 5.5V 40 year operating condition. The second is that during programming, all unprogrammed antifuses are subject to a momentary stress of half the programming voltage (Vpp/2). The programming yield is required to match or exceed PAL yields which are in excess of 99%. The third is that programmed antifuses should have a very low resistance, which will not increase in value over the life of the part. As will be shown below, the unprogrammed antifuse is reliable, well in excess of the 40 year lifespan, the programming yield is excellent, and the programmed antifuse is not subject to any measurable electromigration. The weakest link in the technology is not the antifuse, but typical CMOS process limitations.

Antifuse Structure

The ONO antifuse is sandwiched between N+ diffusion and N+ poly-silicon gate to form a very dense array with density limited by metal pitches (Fig. 2). A thin layer of oxide is thermally grown on top of the N+ surface, followed by LPCVD nitride, and the re-oxidized top oxide. The target electrical thickness of the combined layer is equivalent to 9nm of silicon dioxide.

TDDB of Unprogrammed Antifuses at 5.5V

For the sub 10 nm ONO thickness, time-dependent-dielectic-breakdown (TDDB) reliability over 40 years is an important consideration. The very first task in determining the feasibility of the antifuse was to examine its TDDB reliability. Typical electrical field and temperature accelerated tests were done in order to extrapolate the dielectric lifetime under normal operating conditions. Based on the oxide study [5], it was reported that there may be different field dependencies of lifetime in high field (>6MV/cm) and in low field (<5MV/cm) regimes. In the case of ONO antifuses, the 5.5V operating field is already over 6MV/cm. The extrapolated data from the high field regime was therefore assumed accurate. This assumption was later confirmed with device burn-in data.

Field Acceleration (E vs 1/E model for ONO)

200X200 um^2 (0.04mm^2) area capacitors were packaged and then stressed at different voltages. ONO thickness ranging from 8nm to 9.5nm were studied. The test splits and sample sizes are summarized in Table 1. The TDDB distribution at each voltage condition is shown in Fig. 3.

In the literature, the oxide intrinsic lifetime has been observed to have an exp(1/E) dependence, which is explained mainly with the Fowler-Nordheim tunneling mechanism [6]. Oxide Log(I) curves and lifetime Log(t_{50}) curves exhibit a linear function of 1/E behavior. On the other hand, nitride Log(I) has been shown to follow the Frenkel Poole behavior ($\sqrt{E}$) [7]. Log(I) of ONO is not a linear function of 1/E (Fig. 4a). Rather, it more closely follows E

(Fig. 4b). Also, several studies have fitted lifetime of ONO to an E model [8,9].

Nevertheless a careful examination of our data revealed that TDDB lifetime of ONO follows the 1/E model (Fig. 5) better than the E model (Fig. 6). This is in agreement with conclusions from one study [10], but in contradiction with others which did not examine the fit between data and the 1/E model [8,9]. Based on our observation, we found that the E model can fit the data well over 4 to 5 orders of magnitude of time span. However, as time span increases to 7 orders of magnitude, the E model is clearly inadequate (Fig. 6).

Since there is no theoretical basis for ONO to follow the 1/E model or the E model, we tried a statistical approach to find out which model can best fit the data. First, the data is fitted to different field dependent models of $\exp(E^n)$ with n ranging from -1.5 to 1 at 0.5 intervals. Then the correlation coefficient is compared for different models in Fig. 7. The residual comparison is shown in Fig. 8. Again, the E model (n=1) turns out not to be a good fit for the data. The best fit appears to be n = -.5 or -1. This seems to suggest that ONO behavior is similar to oxide (n = -1). But, the addition of nitride (n=0.5) has changed the n to between -0.5 to -1. Which of the two exponents, n=-0.5 or -1, should be used may depend on the ONO processing conditions. The difference between extrapolated lifetime based on these two models is not nearly as dramatic as the choice between E and 1/E. At 5.5V, the difference in the extrapolated lifetime between n= -0.5 and -1 is one order of magnitude in time. On the other hand, the difference between n=1 and -1 is 5 orders of magnitude. In the subsequent analysis, we will use 1/E model exclusively for simplicity. The conclusion reached will not change much if the $1/\sqrt{E}$ model were to be used.

Besides the $0.04mm^2$ area capacitor data, we also did a TDDB study on single antifuses ($3.2um^2$) and ACT 1010 product antifuse arrays ($0.36mm^2$). Results are shown in Fig. 9. Again, the data follows the 1/E model well for all different area sizes.

Temperature Acceleration

The temperature effect on the $0.04mm^2$ ONO area capacitor lifetime is shown in Fig. 10. The activation energy as a function of the electrical field is shown in Fig. 11. A field dependent activation energy has been reported for oxide TDDB lifetime, as well.

For the 5.5 volt lifetime estimate, the activation energy is close to 0.9eV (1/E model). Using this estimate and the product TDDB defect distribution (Fig. 12), the 1% failure lifetime at 5.5V is well over 40 years. The projected product antifuse failure rate (containing 100K to 200K antifuses) is less than 50 FITS at 125°C.

Program Disturb and Screen

During programming, all antifuse electrodes are precharged at a given voltage, Vpre. To program the antifuse, its poly-silicon electrode is raised to Vpp while its N+ diffusion is grounded. The unselected antifuses are subjected to the stress of either Vpre to ground or Vpp to Vpre for an average of 100 times the single antifuse programing time. Usually the Vpre is set such that stress is approximately Vpp/2. If defective unselected antifuses fail (become programmed) due to this stress, they will show up as programming failures. These defective antifuses can be screened out at wafer sort by a 1 second stress at 10 volts (10V/1s). This screen is done twice during sort. The first 10v/1s (FS-1) screens out the defective dielectric distribution. The second 10V/1s stress (FS-2) simulates the percent yield loss during programming. In Fig. 13, it shows a typical wafer trend on the failure rate of both first and second stress. The 10 run average of FS-2 is 0.3 %. This suggests that the programming failure loss due to antifuse defects after the screen should be less than 0.3%.

Unlike floating gate EPROMs and EEPROMs, latent ONO defects can be easily screened out with a voltage stress as described above. This is one more advantage of the antifuse structure as a programming element. Once an antifuse has passed the voltage screen at sort, it is very reliable. Based on either 1/E or $1/\sqrt{E}$ model, the 10V/1s stress is equivalent to a stress time at 5.5V well over 40 years. We have calculated the equivalent product failure rate at 5.5V as a function of the FS-2 screen yield loss. It shows that for an FS-2 of 0.5%, the equivalent FITs at 5.5V 125°C is less than 50, which is consistent with the results mentioned at the end of the previous section.

Programmed Antifuse Reliability

Once the antifuse is programmed and forms a low resistance path, the resistance should remain low. In the case of oxide, it is a known fact that they are susceptible to self healing [11] or an increase in resistance with time. This is not the case for ONO as will be shown in the following section.

A four terminal Kelvin structure was used for the reliability study (Fig. 14). A constant 5mA current, which is much larger than the operating current, was passed through the antifuse at 250°C (through terminals A,B) while the voltage across the antifuse was monitored between terminals A and B. A typical voltage vs time graph is shown in Fig. 15. A sudden increase in voltage indicates that an open circuit has formed. Prior to that, there is no significant change in the voltage across the antifuse indicating that the resistance remained low.

Next, electric continuity measurement and scanning electron microscopy (SEM) were done on the Kelvin structure. It was found that

the antifuse resistance still remained low when measured from the other two unstressed terminals C and D. This is the case for all samples tested under this condition. SEM analysis showed that the open circuit was related to the metal to poly contact electromigration failure (Fig. 16). The activation energy (based on 250°C and 200°C data) for the contact electromigration is 1.1eV, which is in agreement with typical values obtained from contact electromigration failures [12]. The extrapolated lifetime of contacts in these circuits under normal operating conditions is well in excess of 40 years. The real lifetime of a programmed antifuse itself is yet to be determined.

High Temperature Product Burn-in Life Data

In previous sections, it was demonstrated that the extrapolated ONO antifuse lifetime follows the 1/E model instead of the E model. Product burn-in data supports this conclusion. 1108 units (including PROMs, ACT1010, and ACT1020) containing an average of about 100K antifuses per unit, about 5% of which were programmed, underwent dynamic burn-in at 125°C and 5.5V with roughly an accumulated 1.8 million device hours. No antifuse failure has been observed while two CMOS circuit failures have been observed and identified in the peripheral circuitry. This data is consistent with the failure rate projection based on 1/E extrapolation, while the E model extrapolation based on TDDB test data would have projected 90 unit failures (out of 1108 units) due to ONO antifuses.

Conclusions

We have investigated three reliability aspects of the ONO antifuses. During operation, the lifetime of the ONO antifuse is well in excess of 40 years at elevated temperatures. It has been further demonstrated that the E model is not adequate for lifetime extrapolation. Results indicate that 1/E is a better choice. The key to successful extrapolation is that data should span over seven orders of magnitude in time. Based on the 1/E model, the extrapolated lifetime is well over 40 years at 5.5V. To screen out the programming yield loss due to breakdowns of defective unselected antifuses, a screen was developed. This is not a yield limiting factor in the typical process as the yield loss due to the screen on the average is 1%. After the screen, the programming yield is higher than 99%. The reliability of programmed ONO antifuses was also studied. It was found that the lifetime is limited by the contact electromigration, not by the ONO antifuse. In addition, the resistance remains low throughout the test indicating the antifuse resistance does not increase. Finally, more than 1100 product units and over 1.8 million unit hours of burn-in data have shown no failure at all that can be attributed to the ONO antifuses. This is in agreement with the prediction based on wafer-level tests and the 1/E model.

References

[1] A. Haines , "Field-Programmable Gate Array with Non-Volatile Configuration", Microprocessors and Microsystems, Vol. 13, No. 5, pp. 305-312, June, 1989.

[2] A. El Gamal, J. Greene, J. Reyneri, E. Rogoyski, K. El-ayat, and A. Mohsen, "An Architecture for Electrically Configurable Arrays", IEEE J. Solid-State Circuits. Vol 24, No. 2 pp. 394-398, Apr. 1989.

[3] Preliminary data from a military system manufacturer. Test done on total dose Gamma Irradiation Survivability Test.

[4] E. Hamdy, J. McCollum, S. Chen, S. Chiang, S. ELtoukhy, J. Chang, T. Speers, A. Mohsen, " Dielectric Based Antifuses for Logic and Memory ICs", IEDM Tech. Digest, pp. 786-789, 1988.

[5] K. Boyko, D. Gerlach, "Time Dependent Dielectric Breakdown of 210 A Oxides", Proc. Int. Rel. Phys. Symp. pp. 1-8, 1989.

[6] I. Chen, S. Holland, C. Hu, "Electric Breakdown in Thin Gate and Tunneling Oxides", IEEE Trans. Electron Devices, ED-32, No. 2, pp. 413-422, Feb. 1985.

[7] S. Sze, "Physics of Semiconductor devices", 2nd Edition, John Wiley and Sons,Inc. pp 402-407, 1981.

[8] A. Nishimura, S. Murata, S. Kuroda, O. Enomoto, H. Kitagawa, and S. Hasegawa, "Long Term Reliability of SiO2/SiN/SiO2 Thin Layer Insulator Formed in 9 um Deep Trench on High Boron Concentrated Silicon", Proc. Int. Rel. Phys. Symp. pp. 158-162, 1989.

[9] Y. Ohji, T. Kusaka, I. Yoshida, A. Hiraiwa, K. Yagi, and K. Mukai, and O. Kasahara, "Reliability of Nano-Meter Thick Multi-Layer Dielectric Films on Poly-Crystalline Silicon", Proc. Int. Rel. Phys. Symp. pp. 55-59, 1987.

[10] P. Hiergeist, A. Spitzer, and S. Rohl, "Lifetime of Thin Oxide-Nitride-Oxide Dielectrics within Trench Capacitors for DRAM's", Trans. Electron devices, Vol. 36, No. 5, pp. 913-919, May, 1989.

[11] D. Walters, J. van der School, "Dielectric Breakdown in MOS Devices, Part I,II, III", Phillips J. Res. Vol. 40, pp. 115-192, 1985.

[12] D.S. Peck and O.D. Trapp, "Accelerated Testing Handbook", pp. 5-36 to 5-37, 1987.

Table 1 Field accelerated test data for two lots with thickness ranging from 8nm to 9.5nm. The test was done on 0.04mm2 area capacitor.

Lot A					Lot B				
Voltage (V)	Tox (nm)	E-field (MV/cm)	# of cap	t_{50} (sec)	Voltage (V)	Tox (nm)	E-field (MV/cm)	# of cap.	t_{50} (sec)
13.5	8.3	16.2	22	4.2e-3	14.0	8.7	15.9	25	9.8e-3
12.5	8.3	15.1	22	3.7e-2	13.0	8.7	14.9	25	5.0e-2
12.0	8.3	14.4	22	1.5e-1	12.5	8.7	14.3	25	2.4e-1
11.5	8.3	13.8	22	8.6e-1	12.0	8.7	13.7	25	1.3e0
11.0	8.4	13.1	22	4.7e0	11.4	8.7	13.1	25	9.0e0
10.5	8.4	12.5	9	5.8e1	11.2	8.7	12.5	45	8.0e1
10.0	8.3	12.0	6	3.2e2	10.8	9.0	12.0	45	3.52e2
9.5	8.3	11.4	6	2.5e3	10.2	9.0	11.3	45	2.88e3
9.0	8.3	10.7	36	2.5e4	9.7	9.0	10.8	45	2.07e4
8.5	8.3	10.2	15	2.3e5	9.0	8.7	10.3	32	3.35e5
8.0	8.3	9.6	59	1.5e6	9.0	9.3	9.7	32	2.22e6
Sub-total of tested cap.			241					401	
Total of tested cap.			642						

Table 2 High temperature operating life test data (HTOL).

Device	# of units	# of fuse per unit	Device Hours @ 125°C/5.5V*	# fuse Fail	Equivalent Device Hours @ 55°C
PROM64	275	65,536	450,000	0	18.8 Million
1003JLCC	238	40,000	359,400	0	15.0
1010JLCC	144	112,000	283,000	0	11.8
1020JLCC	61	186,000	90,000	0	3.8
1010PLCC	358	112,000	616,000	0	25.8
1020PLCC	32	186,000	5,300	0	0.2
Total	1108	701,536	1,804,100	0	75.5 Million

* All PLCC, 114/144 of 1010 JLCC and 32/61 of 1020 JLCC have 5.75V.

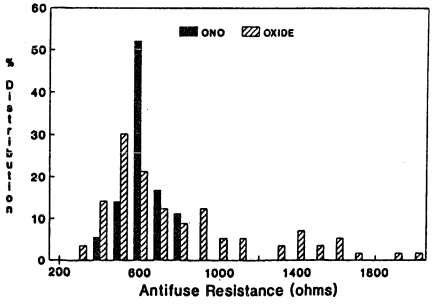


Fig. 1 ONO antifuse has a tighter resistance distribution than oxide antifuse.

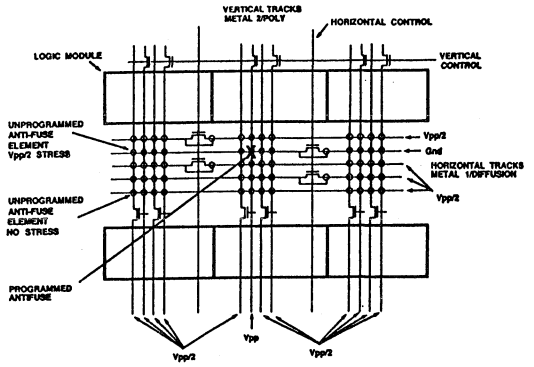


Fig. 2 Simplified product architecture showing logic modules, routing tracks, and antifuse arrays. Vpp is applied to program a selected antifuse. Unselected antifuses have Vpp/2 or 0V stress.

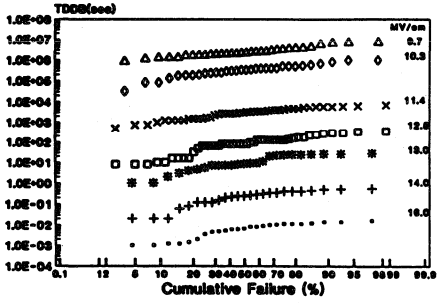


Fig. 3 Cumulative percentage failure verses time on a log-normal scale.

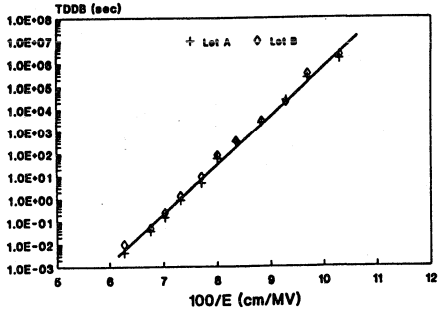
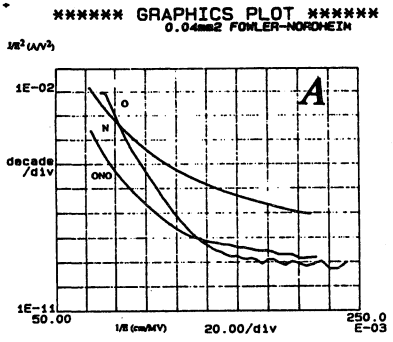


Fig. 5 Log t_{50} vs 1/E for two lots. 0.04mm^2 area capacitor was used.



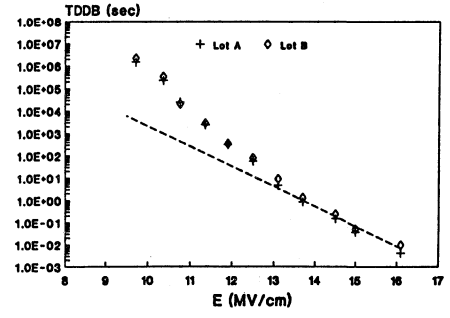


Fig. 6 Log t_{50} vs E for two lots. Log t_{10} has a similar E dependence.

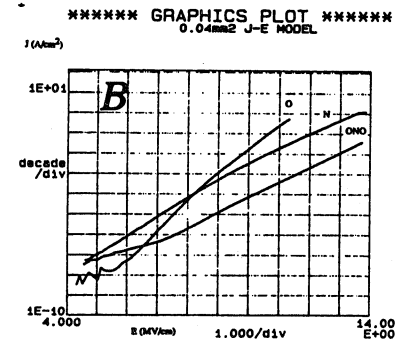


Fig. 4 I-V characteristics of oxide, nitride, and ONO. (a) Fowler-Nordheim tunneling plot. (b) J vs E plot. Log(J) of ONO is not a linear function of 1/E I-V.

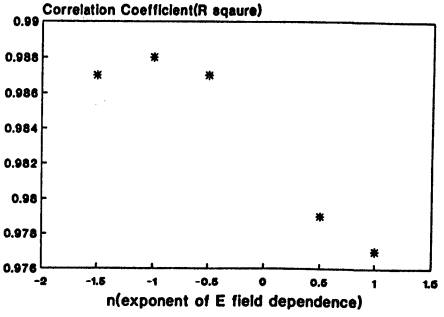


Fig. 7 t_{50} was fitted to 5 different distributions. The fitting correlation coefficient (R^2) is plotted against the field exponent n in [exp(E^n)]. 1/E(n= -1) has the best fit with the largest correlation coefficient.

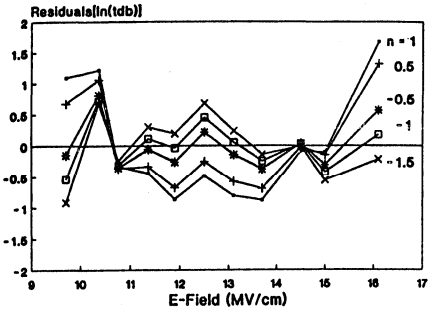


Fig. 8 Residuals at each data point are plotted for 5 different n's. E field dependence has the largest residuals. 1/E and $1/\sqrt{E}$ have the smallest residuals

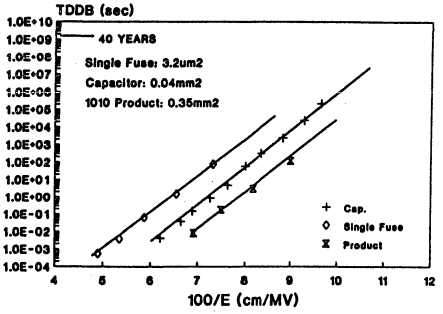


Fig. 9 Log t_{50} can be fit as a linear function of 1/E with the same slope for three different structures: single fuse ($3.2um^2$), area capacitor ($0.04mm^2$), and product array ($0.35mm^2$).

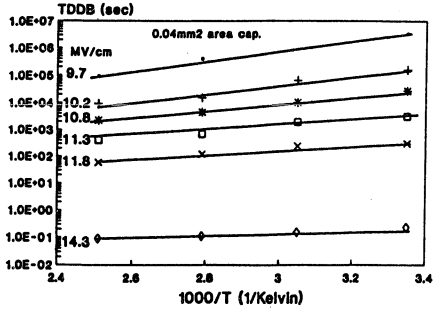


Fig. 10. Field effect on t_{50} at different temperatures ranging from 25°C to 150°C.

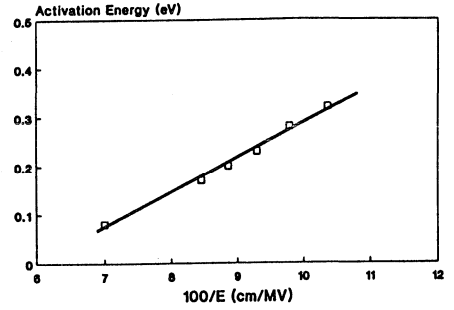


Fig. 11. Field dependence of ONO activation energy.

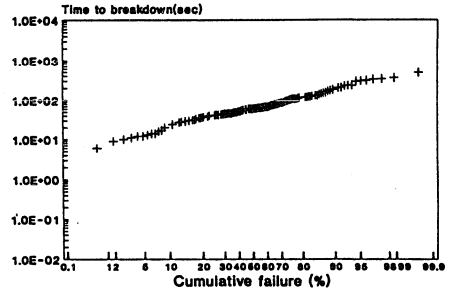


Fig. 12. Cumulative percentage failure of product antifuse array ($0.35mm^2$) vs breakdown time at 11V stress.

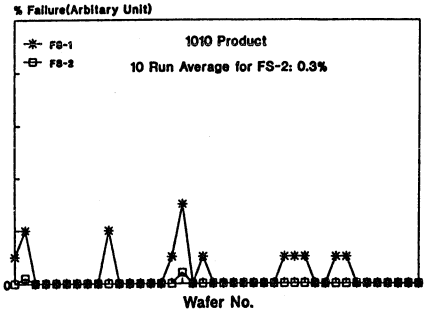


Fig. 13 A typical wafer sort yield loss plot for one lot. After the screen, the 10 run average yield loss is less than 0.3%. Since the screen is more severe than 5.5V/40 years, the product will be very reliable throughout the operating lifetime.

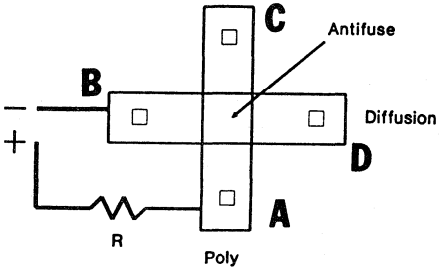


Fig. 14 A four terminal Kelvin structure is used for programmed antifuse reliability test. When an open failure is detected through two stressed terminals, A and B, the antifuse resistance remains low as measured through two unstressed terminals, C and D.

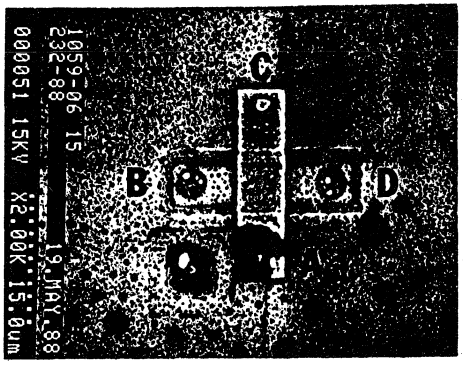


Fig. 16 SEM photograph of the Kelvin structure after showing an open circuit. The open is identified to be at poly to metal contact due to contact electromigration.

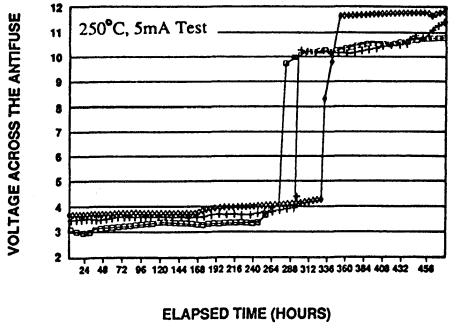


Fig. 15 Voltage across antifuse versus stress time, with 5mA current. Antifuse resistances remains little changed prior to contact failure.

Conductive Channel in ONO Formed by Controlled Dielectric Breakdown

Conductive Channel in ONO Formed by Controlled Dielectric Breakdown

*S. Chiang, R. Wang, T. Speers, J. McCollum, E. Hamdy and C. Hu**

Actel Corporation
955 East Arques Avenue, Sunnyvale, California 94086

*University of California
Berkeley, California 94720

Abstract – For the first time, cross section TEM photos capture the conductive channel of Oxide-Nitride-Oxide (ONO) films after electric breakdown. It reveals a single crystal or polycrystal channel with a dome-shaped cap depending on the breakdown current. The implication of this structure on electric characteristics is analyzed with a spherical thermal-electric model. The use of ONO as antifuses in FPGA's is also discussed.

INTRODUCTION

ONO as well as NO or ON has been well developed for DRAM capacitor and FPGA antifuse applications[1][2]. Its low defect density, scalability, and reliability make it very attractive[3]. One main difference between DRAM capacitor and FPGA antifuse application is that besides good dielectric integrity for both applications, understanding how the conductive channel is formed during breakdown is of considerable importance in controlling the resistance of the conductive channel in antifuse applications. Since the breakdown spot is very small and randomly located, capturing this spot in cross section TEM with high degree of repeatability has not been reported before.

STRUCTURE OF THE CONDUCTIVE PATH

Fig.1 Shows the TEM cross section of an ONO dielectric with an equivalent of 9nm oxide thickness. ONO is sandwiched between N+ poly and N+ diffusion. Programming, or breakdown of the ONO film, was achieved with controlled high voltage pulses. External resistor is used to limit the current going through the breakdown spot.

Fig.2 shows the SEM top view of the breakdown spot after the polysilicon gate is removed by wet etching. Etching stops on ONO film. Fig.2 revealed a dome-shaped loose network of the dielectric over the breakdown spot. The many channels through the loose network are apparently the conductive paths of the breakdown dielectric. The diameter of the dome is roughly 3000Å for 16mA programming current and the link resistance is 100Ω.

This dome-shaped structure is confirmed by cross sectional TEM as shown in Fig.3. Fig.3(a) shows a 200nm diameter dome produced by programming current of 12mA. In addition, the material inside the breakdown spot is single crystal. Fig.3(b) shows a small dome produced by a programming current of 3.5mA. Close examination of the photo shows that there are defects inside the N+ silicon near the breakdown channel and the material inside the conductive channel now is polycrystalline.

Apparently, during breakdown (programming), programming current produces sufficiently high temperature to melt the silicon and ONO film over a small volume centered around the point of breakdown. ONO film breaks up into a loose network and protrudes toward anode, probably due to the drift momentum of the electrons in the molten silicon. Upon cooling at the end of programming, nitride (melting point 1900°C) and oxide (melting point 1700°C) solidify into the dome first, then followed by silicon (melting point 1400°C) solidification. High quality epitaxial growth of silicon takes place when the melt volume is large, eg. at 16mA current. When the melt volume is small, such as at 3.5mA current, the temperature gradient and cooling rate are large – resulting in defects in the breakdown spot below ONO film. This prevents formation of epitaxial single crystal inside the channel. In addition, a small-grain polysilicon crest is formed over the dome. It shows that molten silicon region has a large diameter than the molten ONO because of silicon's lower melting point.

RESISTANCE CHARACTERISTICS AND MODEL

It was mentioned above that when the programming current is 3.5mA, the conductive channel is constituted with polysilicon. Temperature dependence of the resistance of a 5mA programmed antifuse, shown in Fig.4 is indeed similar to that of polysilicon resistor. Fig.5 shows that the conductive channel resistance decreased with increasing programming current.

Without the dome-cap (ONO part) in Fig.3(a), the resistance through the opening of 100nm radius, r, in the ONO film would have been $R=\rho/2r$, where ρ is the resistivity (2mΩ-cm) of the surrounding N+ silicon, i.e. R=100Ω. The measured resistance is about 200Ω, from which the resistivity of the dome material can be estimated to be 15mΩ-cm. Empirically, the dome cap contributes about half of the resistance over the range of programming current studied, i.e. it raised the effective ρ by 2X. To understand what determines r and R, assume that the temperature is spherically symmetrical and equal to 1900°C, ONO melting point, at radius r during programming and that most of the heat I^2R is dissipated within r. The result is

$$r = I \cdot \left(\frac{\rho}{8\pi \cdot 1900\,\kappa}\right)^{1/2} \quad (1)$$

$$R = \frac{\rho}{2r} = \frac{(1900 \cdot 2\pi\kappa\rho)^{1/2}}{I} \simeq \frac{2.5\,(V)}{I} \quad (2)$$

where ρ is resistivity of the surrounding N+ silicon (2mΩ-cm), κ is the silicon thermal conductivity (0.25W/cm°C), and I is the programming current.

Fig.6 shows the antifuse resistance versus measurement current after programming at 5mA. Resistance read at 100uA is very similar to that read at 5mA. As the read current increases, the resistance also increases due to heating and increase in resistivity. The trend reverses when silicon begins to melt. The melting occurs before the read current reaches 5mA since the melting of silicon can take place at lower current than melting of the initial dielectric. Beyond 5mA measurement current, the radius of the opening, r, in ONO increases and resistance, R, continues to decrease. On the second scan, R would start at the new lower value corresponding to the larger r.

When ONO films are used as antifuse in FPGA product, the resistance of the antifuse can be controlled by choosing a sufficiently large programming current level and the resistance remains stable during 1000 hours of burn-in at 125°C and 5.75V. Negligible change in delay time along many different data paths were observed as shown in Fig.7.

REFERENCES

[1] A. Nishimura, S. Murata, S. Kuroda, O. Enomoto, H. Kitagawa, and S. Hasegawa, "Long term reliability of SiO_2/SiN/SiO_2 thin layer insulator formed in 9μm deep trench on high boron concentrated silicon," *IEEE IRPS Proc.*, pp.158-162, 1989.

[2] A. El Gamal, J. Greene, J. Reynerl, E. Rogoyski, K. El-Ayat, and A. Mohsen, "An architecture for electrically configurable gate arrays," *IEEE J. Solid-State Circuits*, Vol.SC-24, no.2, pp.394-398, Apr. 1989.

[3] S. Chiang, R. Wang, J. Chen, K. Hayes, J. McCollum, E. Handy, and C. Hu, "Oxide-nitride-oxide antifuse reliability," *IEEE IRPS Proc.*, pp.186-192, 1990.

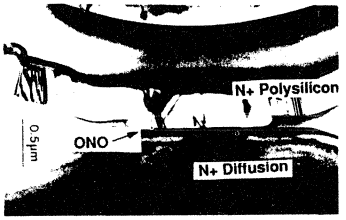


Fig. 1. TEM cross-section of ONO antifuse.

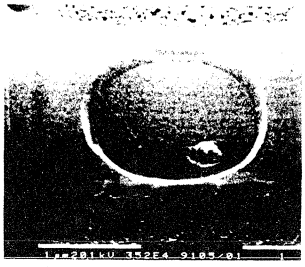


Fig. 2. SEM shows that 16mA breakdown pulse produced a 3000Å opening in the ONO film with a dome cap of a loose network of dielectrics.

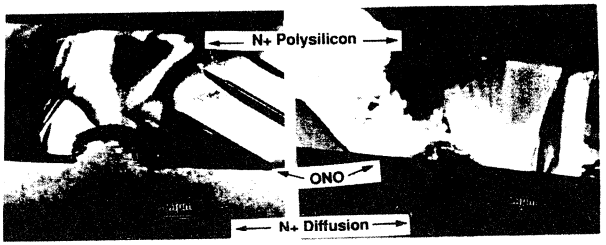


Fig. 3. TEM photos of (a) 2000Å diameter opening in the ONO produced by 12mA breakdown and (b) 700Å diameter opening after 3.5mA breakdown.

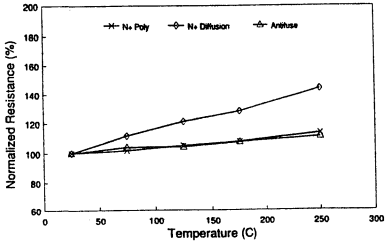


Fig. 4. Temperature dependence of antifuse, N+ diffusion, and polysilicon resistance.

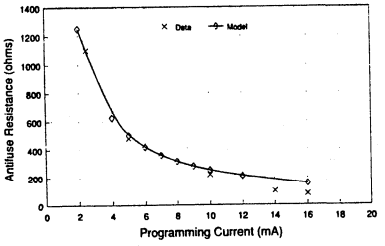


Fig. 5. Antifuse resistance decreases with programming current.

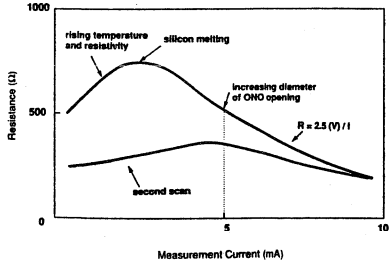


Fig. 6. Resistance versus measurement current after programming with 5mA of breakdown current.

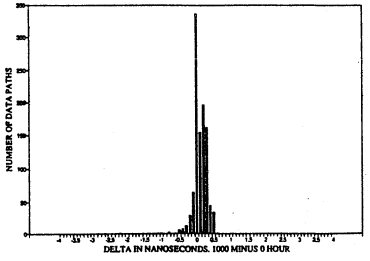


Fig. 7. Change in delay time from a 1000 gate programmable gate array after 1000 hours dynamic burn-in at 125°C and 5.75V.

PREP Data

Section 5: PREP Data

Design Improvement with the PREP Benchmarks

Introduction

The PREP Benchmarks provide designers with a wealth of data with which they can improve their FPGA designs. This application note will describe in detail several ways in which the PREP data can be used to quickly and easily improve the selection of FPGA devices, and the estimation of FPGA capacity and performance during the design. Additionally, some of the limitations in using the PREP Benchmarks for these purposes will be described. Key points are:

- Use the PREP Benchmarks to identify key characteristics like speed, performance and predictability of device families and technologies. Antifuse advantages are clear when compared to SRAM and EEPROM technologies.
- Use Average PREP Capacity (APC) and Average Internal Performance (AIP), as defined below, when comparing device families. In particular, avoid unrealistic averages (like averaging internal and external performance) which may make a particular vendors products look better, but are not representative of actual applications requirements.
- Use the PREP Benchmarks data to estimate performance and capacity of target applications in specific devices. This technique is far superior to manufacturers 'rules of thumb' like 'Toggle Rates' or 'PLD Gates'.

Understand the limitations of the PREP Benchmarks and take into account special features of devices like fast chip-to-chip speeds and complex I/O macros which are not directly measured by the PREP Benchmarks. In particular, ACT 3 devices have fast clock to output (7.5ns pin to pin) and input set-up times (1.5ns), which are very useful in real world applications, but not directly measured in the PREP Benchmarks.

PREP Benchmarks

The full PREP Benchmarks will not be reviewed here. Anyone unfamiliar with the PREP Benchmarks should read Appendix 1 at the end of this document for a quick review. The full benchmark suite is available from PREP Corporation for a modest fee. Contact PREP Corporation at 504 Nino Avenue, Los Gatos, CA 95032, USA 408-356-2169.

Evaluating Key Device Characteristics

One of the main uses for the PREP Benchmark data is in evaluating the key device characteristics of performance, capacity and cost. Prior to the PREP Benchmarks each vendor measured performance differently (Remember Toggle Rates, System Clock Rates and Fmax). Now each vendor reports the actual worst case commercial performance and capacity for fully placed and routed designs of common functions like adders and counters. These metrics can then be compared between different devices with more concrete results.

Classification of Programmable Devices

Determining the right programmable device for a design can be a difficult task. Information regarding the usable device capacity, performance and I/O count, as well as the price, is vital to choosing the most appropriate programmable device. Graphically displaying the devices by the PREP Benchmark measurements of performance and capacity, as shown in Figure 1, can expedite this decision process. The Average Internal Performance (AIP), expressed in MHz, is calculated as the average of the mean internal operating frequency over the set of benchmark circuits while using 100% automatic placement and routing software. AIP is the best single metric available for comparing FPGA performance. AIP is used because in most FPGA applications, device performance is

AIP—Average Internal PREP Performance = avg. of mean internal frequencies	Large FPGAs hold most of the logic in a single device. AIP represents real world functional performance.
APC—Average PREP Capacity = avg. number of benchmark repetitions	APC standardizes capacity claims. An easy "gate" calculation is to multiply APC by 250.
APP—Average PREP Predictability = 1 – [(AVGmax – AVGmin) / AVGmean]	High APP means best and worst paths are very similar. Thus, design changes are low risk, offering ease of use.
Chip-to-Chip (MHz) = 1000 / [Clock-out (ns) + Input Setup (ns)]	Most large designs are synchronous. Chip-to-chip represents real world system performance.

usually limited by the clock to clock rate of the most complex logic function implemented in the device. External performance is usually only limited by clock to output and input set-up times (9 ns total on ACT 3 Devices), usually much faster than internal rates. Additionally, for high capacity devices (over 1000 gates), complete functions are incorporated in a single device, and there are few cases where functions need to be split between devices. In low capacity PLDs, on the other hand, logic functions are more likely to be split between devices or used to simply 'glue' two separate functions together. External operating frequency may be a better metric for PLDs in these cases. In most FPGA applications, the PREP External Performance (Fext) benchmark is not a good indicator of real world external performance. (For more details on this point refer to the "Limitations to the PREP Benchmarks" section of this document). Beware of averaging methodologies which include both external and internal performance. These measurements are completely different and mixing them does not make sense.

The Average PREP Capacity (APC), is expressed as the average number of instances of unit repetitions for the nine benchmarks. (An average 'PREP Rep' is about equivalent to 250 gate array gates.). For example, if Device XYZ had a total of 90 repetitions of instances for all the benchmarks, its APC would be 10 (90 reps divided by 9). This measure of capacity assumes all benchmarks are of equal importance. Designers may wish to create their own capacity measure by weighting the benchmarks differently. For example, a communications application might be more accurately represented by excluding the accumulator, large state machine and memory map benchmarks or weighting the datapath and small state machine more heavily. As a representation of the general applications space, the unweighted average over all the benchmarks is probably the best metric.

Figure 1 shows distinctive device classifications. The high-speed devices congregate in the upper left hand corner of the chart, the high-speed, high-capacity devices group in the upper middle to right side, the high-capacity devices group in the middle of the right side, and the remaining mid-range devices fill the middle of the chart. These four classifications offer designers a quick means of determining which devices will satisfy their performance and capacity requirements. For instance, if a design needs both high speed and high capacity, the designer would quickly realize that only the Actel A1460A and A14100A devices offer both characteristics. When comparing devices based on PREP Benchmark data, it is wise to only make performance and value comparisons between devices within the same classification. For example, comparing the benchmark performance of two high-capacity devices would have more merit than comparing a high-capacity device with a high-speed device. Likewise, when comparing device cost, it is important to compare devices of similar capacity.

Figure 1 can also be used to draw several conclusions about key device family characteristics. Notice that E/EEPROM-based devices, like the EPM7xxx, have high performance at low capacity, but performance drops off as capacity increases. This is a direct result of the fixed interconnect nature of the E/EEPROM architectures'. As more PLD-type blocks are added to increase capacity the interconnect between blocks expands, slowing interconnect speed. Additionally, the power budget available for each PLD block drops as more blocks are added, reducing performance of each block. These effects combine making performance drop dramatically for E/EEPROM-based devices. Figure 2 shows this trend more clearly.

The RAM-based devices, like the XC40xx and FLEX80xx, do not drop as much in performance as capacity increases since the interconnect increases only linearly. Notice, however, that performance levels start low and stay low for RAM-based devices. This is due to the slow interconnect available with RAM technology and the lack of sufficient routing resources to efficiently interconnect logic modules. Increasing interconnect is not effective with RAM-based devices since the large size of the RAM switch would increase die size dramatically, making the devices unmanufacturable.

Antifuse-based devices start out with performance levels equivalent to E/EEPROM-based devices at the low capacity levels. As capacities increase antifuse-based devices only fall off in performance slightly, while E/EEPROM-based device performance falls dramatically. Because of the small size and low impedance of antifuse, antifuse-based devices offer abundant high-speed routing resources even at high capacities. This results in high-performance, high-capacity devices. Only antifuse-based technologies like the Actel ACT 3 family have these combined characteristics.

Performance Predictability

With time-to-market issues becoming a significant factor, the ability of a device to meet required performance goals without using manual optimization is a vital need. Design architecture's which offer both design flexibility and highly predictable performance, while using 100% automatic place and route software, can ensure a designer that the design will be completed on time.

One measure of performance predictability is Average Performance Predictability (APP). This measure expresses performance predictability as the ratio of the slowest instance to the fastest instance in a particular benchmark. The average for all 9 benchmarks is then computed to arrive at APP. Table 1 shows the APC, AIP, APP and Chip-to-Chip performance for the high performance devices shown in Figure 1.

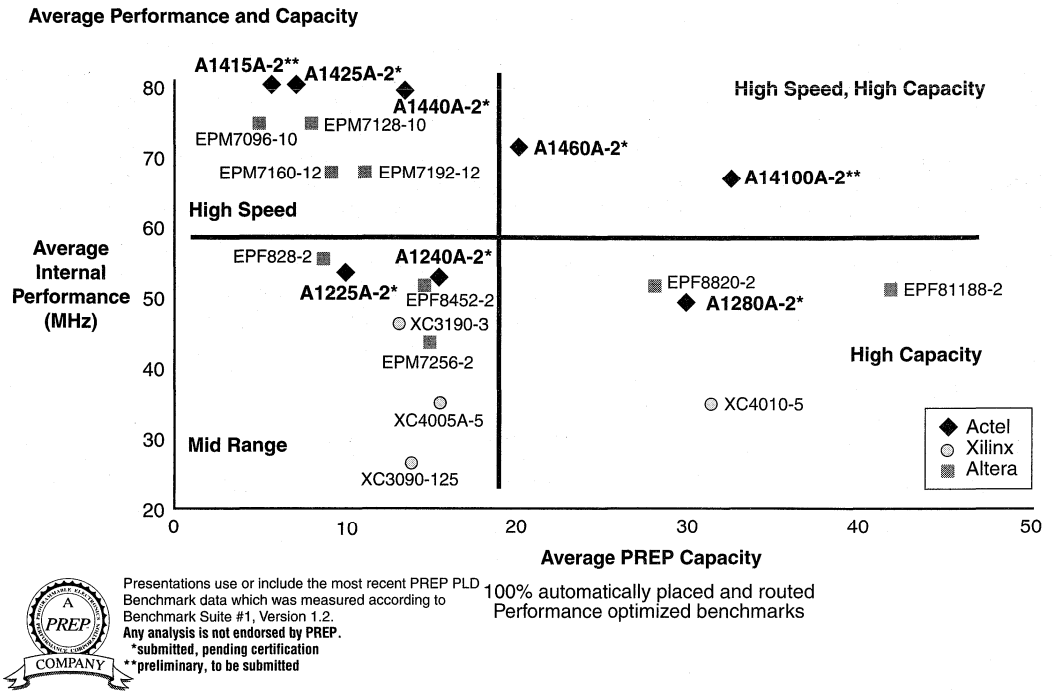


Figure 1 • Average Internal Performance versus Average PREP Capacity

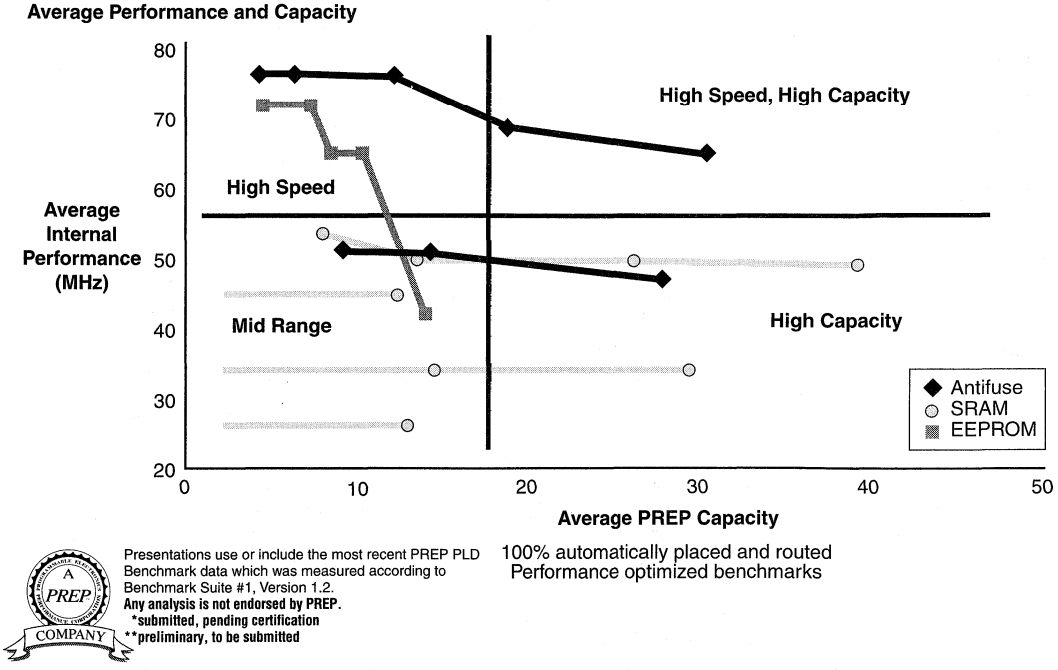


Figure 2 • Average Internal Performance versus Average PREP Capacity Trend

PLDs, known for their predictability, do not offer the design flexibility required in many time-to-market applications. FPGAs, known for their flexible design architecture, can also offer very predictable performance, as shown in Table 1. The PREP Benchmark data shows antifuse-based FPGAs to be much more predictable than SRAM-based FPGAs. This means that fast time-to-market will be maintained by antifuse devices, where as RAM-based devices might require tedious and time consuming manual effort to hit what appeared to be moderate speed goals. Only antifuse-based FPGAs are classified as predictable, high-speed, and high-density devices.

Specific Applications and PREP Benchmark Data

The PREP Benchmarks are useful in drawing conclusions about device family and technology characteristics when overall averages are used, but a particular designer is probably more interested in the PREP Benchmark results which apply to a specific real world design. Table 2 shows several real world designs using the ACT 3 Family. (Several of these designs have detailed applications notes, and macros available to help further speed time to market. See the references at the end of this section). The performance and capacity for the key functions in each of these designs can be identified by the designer early and the PREP Benchmark data used to compare different devices.

Table 1 • APC, AIP, APP, Chip-to-Chip

Device	Capacity APC	Performance AIP	Chip-to-Chip	Predictability APP
A1415A-2	4.9	81 MHz	111 MHz	89%
EPM7096-10	4.9	75 MHz	77 MHz	100%
XC4002A-5	5.0	35 MHz	69 MHz	60%
A1425A-2	7.0	81 MHz	111 MHz	89%
EPM7128-10	7.1	74 MHz	77 MHz	100%
XC4003A-5	7.9	35 MHz	69 MHz	60%
EPF8282-2	8.6	56 MHz	80 MHz	83%
EPM7192-12	11.0	67 MHz	63 MHz	100%
A1440A-2	13.0	80 MHz	100 MHz	89%
EPF8452-2	14.4	52 MHz	77 MHz	79%
EPM7256-12	14.7	45 MHz	42 MHz	99%
XC4005A-5	15.4	35 MHz	69 MHz	60%
A1460A-2	19.7	72 MHz	90 MHz	88%
XC4006-5	20.1	35 MHz	69 MHz	60%
EPF8820-2	28.1	52 MHz	77 MHz	76%

Chip-to-chip for all devices is calculated from datasheet specifications.

Table 2 • Real World Applications

Market Segment	Application	Performance with '2' Speed Grades
Networking	100 MBit Ethernet Receiver	
	100 MHz Fiber Optic Concentrator	
Telecom	64 MHz RISC Coprocessor	(80 MHz for '–2' Speed Grade)
Computing	40 MHz DRAM Controller	(50 MHz for '–2' Speed Grade)
	40 MHz DMA Controller	(50 MHz for '–2' Speed Grade)
	40 MHz Protocol Controller	(50 MHz for '–2' Speed Grade)
	40 MHz Register Banks	(50 MHz for '–2' Speed Grade)
Peripheral	40 MHz SCSI DMA Controller	(50 MHz for '–2' Speed Grade)
DSP	40 MHz FIR Filter	(50 MHz for '–2' Speed Grade)
Graphics	100 MHz Memory Control	(133 MHz for '–2' Speed Grade)
	100 MHz I/O State Machines	(133 MHz for '–2' Speed Grade)
	50 MHz BIT BLT	(67 MHz for '–2' Speed Grade)

Table 3 and Table 4 compare the performance of the High and Mid-Range Capacity devices from Figure 1. The individual benchmark result is shown for each device and it is easy to determine which device is faster for a particular benchmark function. If the performance of a particular benchmark function is critical to the designers application (perhaps a counter in a DMA controller) this performance could be used to determine which devices could realistically be expected to meet the performance goal. In Table 3 the highest performance device for an individual benchmark is indicated by a single star. In Table 4, the fastest has two starts and the runner up has one star. Notice that some devices are fastest at one type of function, but another device is fastest for another type. For example, in Table 4, ACT 3 devices are faster in datapath and counting applications while the more PLD oriented MAX devices are faster in state machine applications. For some functions, parts have quite similar performance characteristics.

Once a designer has identified the important PREP Benchmarks for a particular application, estimates can be made for performance and capacity prior to the detailed design. This is particularly important during the decision making process prior to the start of a design. Committing to a particular device or family is in many cases an irrevocable decision and the designer needs to be as certain as possible that the selected device will support the application requirements.

***Table 3** • Performance at High Capacity (15 or more PREP Reps)*

	Actel	Xilinx	Altera
PREP Benchmark	**A1460A-2**	**XC4010-5**	**EPF8820-2**
Datapath	133 MHz ★	58 MHz	88 MHz
Timer/Counter	55 MHz ★	26 MHz	27 MHz
State Machine	65 MHz ★	39 MHz	45 MHz
Large State Machine	37 MHz ★	27 MHz	18 MHz
Arithmetic	28 MHz ★	19 MHz	20 MHz
16-bit Accumulator	48 MHz	31 MHz	57 MHz ★
16-bit Counter	88 MHz ★	42 MHz	71 MHz
16-bit Prescaled Counter	132 MHz ★	34 MHz	88 MHz
Memory Map	61 MHz ★	47 MHz	35 MHz
AIP (Average Internal Performance)	72 MHz ★	35 MHz	52 MHz

PREP performance based on A1460A-2 (submitted, pending certification), XC4010-5 and EPF8820-2 devices.

***Table 4** • Performance at Mid-Range Capacity (5 to 15 PREP Reps)*

	Actel	Xilinx		Altera	
PREP Benchmark	**A1425A-2**	**XC3190-3**	**XC4003-5**	**EPM7096-10**	**EPF8282-2**
Datapath	164 MHz ★★	101 MHz ★	58 MHz	100 MHz	97 MHz
Timer/Counter	56 MHz ★★	43 MHz	26 MHz	52 MHz ★	49 MHz
State Machine	69 MHz ★	49 MHz	39 MHz	100 MHz ★★	49 MHz
Large State Machine	41 MHz ★	23 MHz	27 MHz	63 MHz ★★	20 MHz
Arithmetic	29 MHz ★★	24 MHz	19 MHz	26 MHz ★	21 MHz
16-bit Accumulator	54 MHz ★	28 MHz	31 MHz	32 MHz	60 MHz ★★
16-bit Counter	93 MHz ★	40 MHz	42 MHz	100 MHz ★★	71 MHz
16-bit Prescale Counter	160 MHz ★★	68 MHz	34 MHz	100 MHz ★	97 MHz
Memory Map	63 MHz ★	54 MHz	47 MHz	100 MHz ★★	38 MHz
AIP (Average Internal Performance)	81 MHz ★★	46 MHz	35 MHz	75 MHz	56 MHz

PREP performance based on A1425A-2 (preliminary, to be submitted), XC3190-3, XC4010-5, EPM7096-10 and EPF8282-2 devices.

If the designer can identify the key functions needed in the target application, and these functions are similar to some of the PREP Benchmarks, then the benchmark performance and capacity numbers can be used to estimate the speed and capacity of the application in a particular device. Devices which can't hit the speed requirements of the application can be excluded from further consideration and devices which can meet the requirement can then be examined for capacity fit. Devices which hit both speed and capacity requirements can then be compared with respect to cost, power, package, and any other important concern. A more detailed description of this process as well as an example is given in Appendix 2.

Even though the PREP Benchmark data allows realistic estimates for capacity and performance (certainly better than previous estimates based on 'Toggle Rate', 'Gate Count' and other rules of thumb), there are several important limitations on using the benchmarks and designers must keep these in mind as they go through the evaluation process.

Limitations of PREP Benchmark Data

As good as the PREP Benchmarks are at identifying key family and technology characteristics and at estimating performance and capacity applications fit some of the artificiality of the benchmarks can introduce errors, primarily when looking at real world applications. A key requirement of the PREP Benchmarks is that each instance of the benchmark is implemented exactly the same. This limits the ability of devices to take full advantage of special hardware capabilities which might only be used on the inputs or outputs of the device. For example, the ACT 3 Family has a complex I/O logic module which gives designers additional functionality and performance when signal go on or off chip. In real world applications, designers make use of these resources and the design of a function is considerably different if it is inside the device versus on the inputs or outputs of the device.

Synthesis software which automatically makes us of the powerful IO macros is now becoming available. The use of this software will allow designers to more easily reap the benefits of these architectural features without needing to get involved with the details of the design implementation. This software are expected to improve the clock to output of real world designs by as much as 50%, and PREP Benchmark results for Fext by up to 20%.

To illustrate this concept a simple example is given in Figure 3. A 4-bit shift register is implemented in 2 different locations: inside the device and on the outputs of the device. Each implementation produces the same logic function, but each used different capabilities of the device. Limiting the designer to only a single implementation results in an unrealistic estimate of the functions performance and capacity. Notice the difference in the reported performance of this simple benchmark when the PREP rules are used versus the 'real world' design. In particular, notice the improvement in the External Operating Frequency when the IO macros are used. This is one the main reasons why the Fext performance numbers of the PREP Benchmarks for ACT 3 Devices are unrealistically low.

If a more 'real world' approach is taken to implementing some of the PREP Benchmarks better estimates for application performance and capacity would result. As an example, the Datapath benchmark has been redesigned without the unrealistic PREP constraint on the use of IO macros. Both designs of this benchmark are shown in Figure 4 along with the internal and external performance measurements. Notice the significant increase in Fext when the IO macros are used.

Another important constraint imposed by the PREP methodology is the requirement that the output of the last benchmark needs to be connected to the input of the first benchmark to measure Fext. Usually in high capacity FPGA designs inputs and outputs are registered. In these cases the external operating frequency is simply the sum of the clock to output delay and the input register set-up time. For ACT 3 devices the clock to output and input set-up time are significantly improved when using IO macros. These times were shown in Table 1 for each ACT 3 device and are a more realistic estimate for Fext performance in 'real world' applications.

Conclusion

Selecting the right device for a particular application is a complicated decision. The PREP Benchmarks provide the designer with a wealth of data with which to improve and speed this decision process. Key steps in this decision process are:

1. Determine the appropriate product family and technology by looking at the key family characteristics of capacity, cost, predictability and performance.
2. Determine the appropriate device within the product family by estimating the capacity and performance requirements of the application.
3. Include any important aspects of the device not directly measured by the PREP Benchmarks, such as IO macro capability, clock to output delays, input set-up times, and power dissipation to refine the previous estimates.

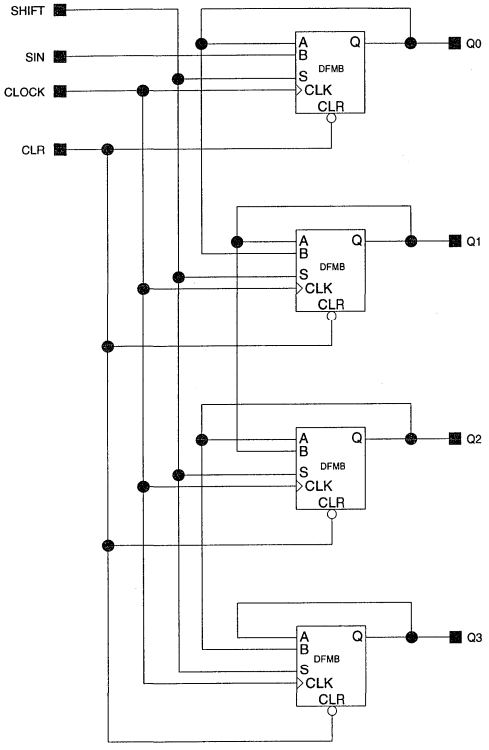
References:

1. Coble, Dave, *A Finite Impulse Response Filter for E3 Using Field Programmable Gate Arrays*. PLD Conference, 1994.
2. Linton, Ken, *The Design of a Numerical Accelerator Using RISC Processors and PLD's*. PLD Conference, 1994.
3. Miller, Warren, *A 60 MHz RISC Coprocessor Using the A1460 and VHDL Entry*. Data I/O Application Quarterly Q1 '94.
4. Miller, Warren, *Using Actel ACT 3 FGPA's in Emerging 100 MBit Ethernet Applications*. Silicon Valley Networking Conference 1994.
5. Symanski, Peter, *An Image Compression Application Using Programmable Logic*. PLD Conference, 1994.

PREP Benchmark Implementation

Fext performance* 70 MHz

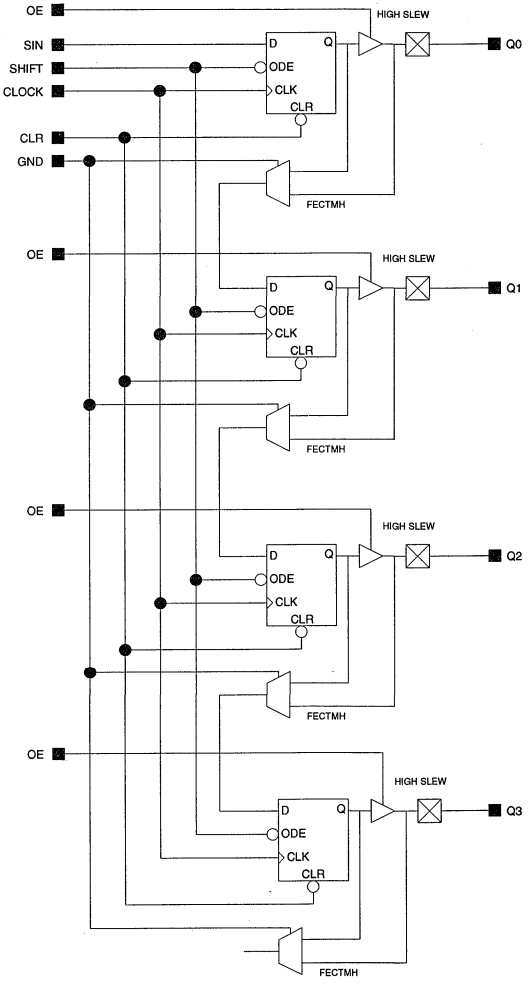
Fint performance* 125 MHz



Real World Implementation

Fext performance* 110 MHz

Fint performance* 125 MHz



*Based on A1425A-2 device

Figure 3 • Shift Register Implemented in 2 ways, with performance numbers

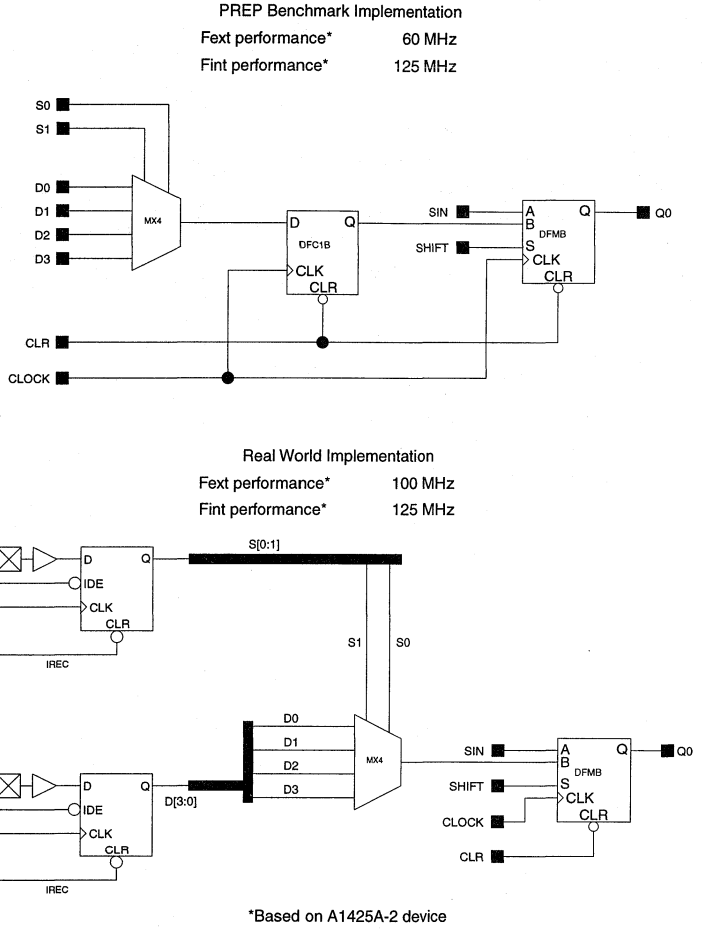


Figure 4 • PREP Datapath Design and Real World Design

Appendix 1

PREP Benchmark Review

There are 9 major benchmark functions and they are listed in Table 5. (The full suite will not be described in detail.) The PREP Benchmark methodology requires that a benchmark circuit is repeated to completely fill the measured device in a step and repeat manner. Each circuits input is connected to the output of the previous circuit. The first instances inputs are connected to Input pins and the last instances outputs are connected to Output pins.

Capacity is measured by reporting the number of instances of the benchmark circuit that fit in a particular device. Nine different benchmark circuits are used and thus capacity measurements are reported for all nine of the benchmarks for an individual device. For example, the first benchmark circuit is a datapath circuit and is shown in Figure 5. The circuits are repeated as shown in Figure 6, with inputs and outputs interconnected. Notice that some signals are global and go to all instances, while other signals are only used to interconnect instances. The first and last instances are connected to package inputs and outputs. For example, in the A1425A 10 instances of the datapath circuit can fit, thus the capacity for benchmark 1 is reported as 10 repetitions (Reps).

Performance is reported in the benchmarks by measuring the worst case commercial maximum operating frequency of each benchmark instance. The maximum operating frequency of a single benchmark instance is determined by the slowest of:

- the longest path between a flip-flop in the benchmark instance and a flip-flop in the previous benchmark instance, including any combinatorial delays between the flip-flops.
- the longest path between any two flip-flops in the same benchmark instance, including any combinatorial delays between the flip-flops.

The first benchmark instance performance is not reported, since it has no previous instance.

The slowest, fastest and mean frequency of the benchmark instances in a given device are than reported for each benchmark. For example, if 1/2 of the of the datapath benchmark instances in Device1 run at a worst case frequency of 80Mhz, and 1/2 run at 100MHz, the minimum frequency is 80Mhz, the maximum is 100Mhz and the mean is 90Mhz.

External frequency of operation is also reported by taking the outputs of the last benchmark instance and connecting it to the input of the first benchmark instance through the input and output pins of the device. A single number is reported since only a single instance exists for this measurement.

The PREP Benchmark reporting format for the datapath benchmark is shown in Table 6. The key columns are: Capacity/Reps, which indicates the number of benchmark instances which fit in a particular device; Performance/Worst/Best/Mean, which give the commercial Fmax for the slowest (Worst) instance, fastest (Best) instance and the average (Mean) of all the instances; Performance/Ext, which gives the commercial Fmax for the external (Ext) operating frequency. Notice that the designs can be optimized for capacity (Optimized/Cap) or performance (Optimized/Perf) so some parts have more than one entry.

Table 5 • *PREP Benchmark Functions*

Benchmark	Function
Datapath	8-bit 4:1 Multiplexer, 8-bit register and 8-bit shift register
Timer/Counter	Counter, Load Register, Compare Register, Comparitor (all 8-bits)
Small State Machine	8-State State Machine with simple transition terms
Large State Machine	16-State State Machine with complex transition terms
Arithmetic	4x4 Multiplier, 8-bit Adder and 8-bit Register
Accumulator	Registered 16-bit Adder
Counter	16-bit Loadable Counter
Pre-scaled Counter	16-bit Counter optimized for counting (Loading takes multiple cycles)
Memory Mapper	Multiple address Decode with error detection

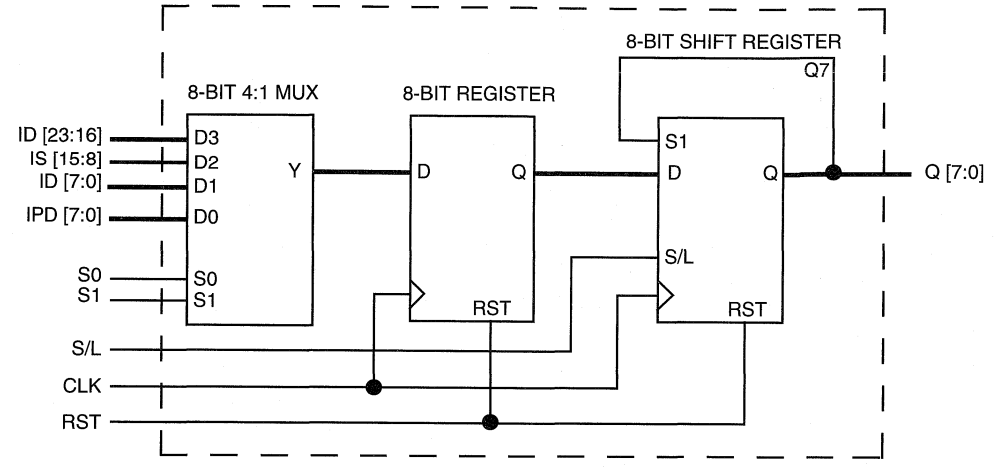


Figure 5 • A Datapath Benchmark Circuit

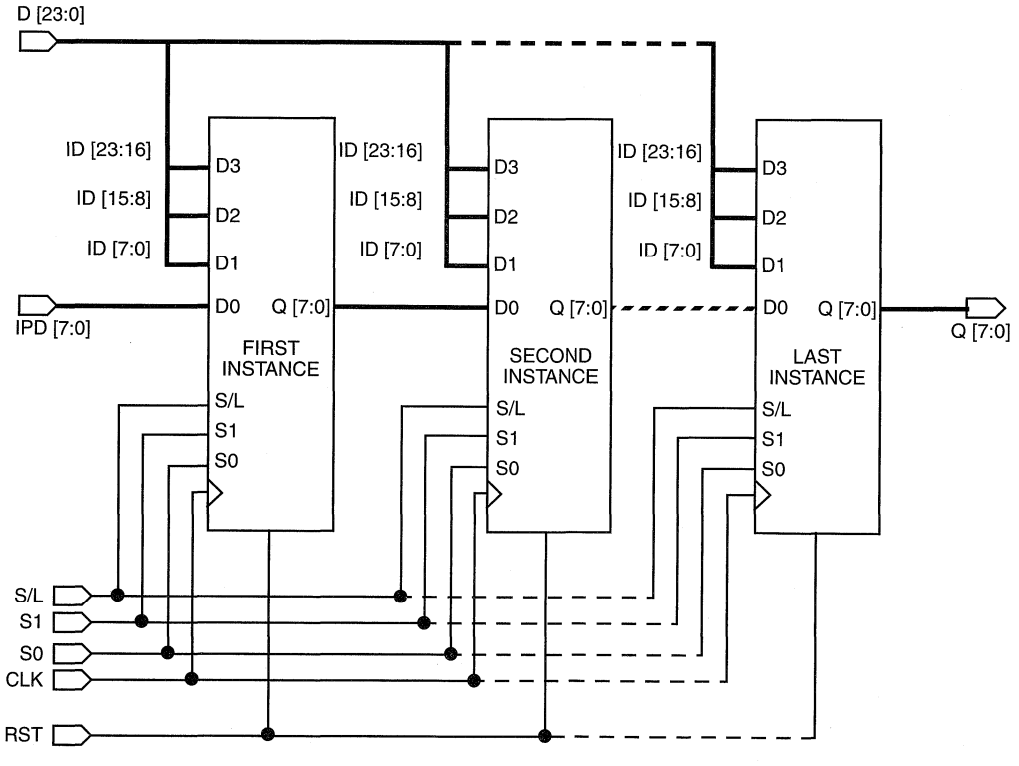


Figure 6 • A Datapath Benchmark Circuit with Interconnected Inputs and Outputs

Table 6 • PREP Reporting Format

			Capacity			Performance (MHz)				Optimized			Tool	Tools Used						
Company	Device/Grade	Reps	Fill	Rep%	Fill%	Worst	Best	Mean	Ext	Cap	Perf	Vendor	Name	Rev.	APR	Crit.	HM	Manp	Manr	Notes
Actel	A1280-1	48	0	97%	0%	29	41	34	12	X		Actel	ALS	2.2	X	X				
Actel	A1280-1	39	0	54%	0%	55	65	61	30		X	Actel	ALS	2.2	X	X				
Actel	A1240	21	0	54%	0%	66	66	66	30		X	Actel	ALS	2.2	X	X				
Actel	A1240A-2	27	0	99%	0%	49	61	57	24	X		Actel	ALS	2.2	X	X				
Actel	A1240	27	0	99%	0%	37	46	43	20	X		Actel	ALS	2.2	X	X				
Actel	A1240A-2	21	0	54%	0%	95	95	95	35		X	Actel	ALS	2.2	X	X				
Actel	A1225-1	18	0	100%	0%	44	53	49	25	X		Actel	ALS	2.2	X	X				
Actel	A1225-1	14	0	54%	0%	85	85	85	35		X	Actel	ALS	2.2	X	X				
Actel	A1425	10	2	52%	26%	125	125	125	38			Actel	ALS	2.2	X	X				
Actel	A1425	7	0	43%	0%	64	74	70	50			Actel	ALS	2.2		X		X		

Appendix 2

Using PREP Data to Estimate Application Performance and Capacity

Most of the PREP Benchmark functions are familiar to designers, and usually the designers application will be made up of a variety of similar functions. If the designer can identify the major functions in the target application and compare these performance and capacity requirements to the PREP Benchmarks for the various devices, the field of potential devices can be narrowed considerably. Once the designer knows the two or three possible candidates other factors such as cost, ease of use, tool familiarity and other application requirements can also be factored into the decision. The first step though, is to identify the key functions in the target application.

Identify Key Function Blocks

A simple block diagram is usually required to identify the key functions in a target application. Usually a "back of the envelope" drawing will suffice. Figure 7 shows a block diagram for an example application. It is a graphics controller for an LCD display. The display is bit mapped, with the image data stored in RAM. The controller interleaves simple operations on the bits (SET, RESET, XOR, XOR with MASK, SHIFT, etc.) with reading the data from memory and sending it to the display.

The address counter is used to address the memory during display operations. A multiplexer is used to select between display addresses and host addresses register. The desired address is loaded into the memory address register. The addressed value is read from memory and loaded into the serial shift register which provides the serial bit stream to the display. Bit operations are executed in the ALU, with data sources from the memory register, the host data register and the bit mask register. Notice that all registers can be read by the host processor over the host data bus (via the multiplexer) to simplify coding and debugging. A small state machine controls the host/controller interface and provides the required timing for the display.

Determine Performance Requirements

The required performance of the critical blocks for each section is next estimated by the designer. Estimates for the host bus interface state machines can be made based on the

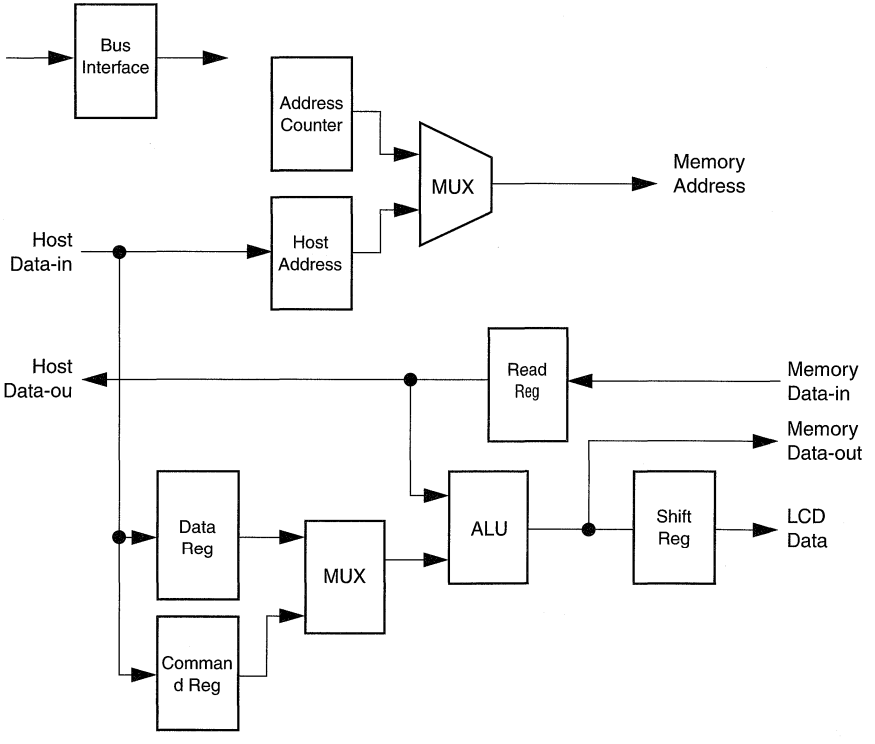


Figure 7 • LCD Controller Block Diagram

processor bus speed. The host bus is a 32 MHz synchronous bus, so a 32 Mhz bus interface state machine speed is required. The shift register needs to operate at the clock rate of the display (64 Mhz). The address counters need to operate at the memory access speed during burst read or write cycles. A 32 MHz rate for the counters will be needed to meet the burst rate of the bus. The ALU operates at the same speed as the state machine (32MHz), to insure synchronization. The other paths in the design can operate at slower speeds and are not speed critical. All these key speed requirements are tabulated in Table 7.

Table 7 • Performance Requirements

Block	Performance
Address Counter	32 MHz
ALU	32 MHz
Shift Register	64 MHz
Bus State Machine	32 MHz

Identify Application Function Blocks Match With PREP Benchmark Functions

Several of these blocks are similar to the PREP Benchmark units. Performance and capacity requirements for each of these functions can be compared to the benchmark results for candidate devices to determine the best device for the application. The designer needs to identify the benchmark circuit the application block is most similar to, so that the benchmark capacity and performance can results can be used. Table 8 shows an example match of the application blocks to the benchmark functions.

The counters for the address generator are very similar to the loadable counter in the benchmark suite. If they needed to be loaded and compared to arbitrary values the timer counter benchmarks would be more appropriate. The state machine is reasonably simple in nature, so the small state machine benchmark can be used to predict performance. The bit stream register is exactly the same as the shift register portion of the datapath benchmark, making it a good candidate for estimation. The ALU is fairly simple, implementing only logical operations, no arithmetics. The accumulator benchmark would provide a good conservative estimate for the performance of the ALU. The rightmost columns in Table 8 indicate the benchmark functions to be used for each of the major sections of the target applications and the required performance of each benchmark. These target performance numbers can now be compared to the benchmark results for a variety of devices to determine which devices meet the performance requirements.

Table 8 • Corresponding Benchmark Blocks

Application Block	Benchmark Block	Target Speed
Address Counter	Counter	32 MHz
ALU	Accumulator	32 MHz
Shift Register	Datapath	64 MHz
Bus State Machine	Small State Machine	32 MHz

Compare Device Speed to Application Performance Requirements

The benchmark results for several Actel devices are given in Table 9. Notice that some benchmarks have two results. Since designs can be optimized for either speed (using more logic) or capacity (using less logic) both numbers are given where appropriate. Using the higher speed for the performance comparison is best. The desired performance can now be compared to the actual benchmark performance of each device and this is done in Table 10. Devices which meet or exceed the performance requirement can be easily determined. The A1280A, for example, misses the desired performance of the accumulator benchmark. The A1225A, A1240A and A1425A each meet all of the benchmark performance requirements, and thus are candidates for further evaluation.

Table 9 • Mean Benchmark Performance of Actel Devices

Benchmark	A1225-1	A1240-2	A1280-1	A1425
Datapath (Capacity)	49 MHz	57 MHz	34 MHz	125 MHz
Datapath (Speed)	85 MHz	95 MHz	61 MHz	125 MHz
Timer/Counter	31 MHz	35 MHz	21 MHz	41 MHz
Small State Machine	34 MHz	39 MHz	29 MHz	41 MHz
Large State Machine	21 MHz	24 MHz	17 MHz	28 MHz
Arithmetic	13 MHz	15 MHz	11 MHz	18 MHz
Accumulator (Capacity)	24 MHz	29 MHz	21 MHz	36 MHz
Accumulator (Speed)	32 MHz	36 MHz	28 MHz	40 MHz
Counter	50 MHz	58 MHz	41 MHz	68 MHz
Pre-scaled Counter	85 MHz	95 MHz	75 MHz	114 MHz
Memory Mapper	34 MHz	39 MHz	30 MHz	49 MHz

Table 10 • Application Performance Requirements and Device Benchmark Performance

Application Block	Benchmark Block	Target Speed	A1225-1 Speed	A1240-2 Speed	A1280-1 Speed	A1245 Speed
Address Counter	Counter	32 MHz	50 MHz	58 MHz	41 MHz	68 MHz
ALU	Accumulator (Speed)	32 MHz	32 MHz	36 MHz	28 MHz	40 MHz
Shift Register	Datapath (Speed)	64 MHz	85 MHz	95 MHz	61 MHz	125 MHz
Bus State Machine	Small State Machine	32 MHz	34 MHz	39 MHz	29 MHz	46 MHz

Capacity Estimation

Once candidate devices have passed the performance requirement, they next need to be examined to see if they offer sufficient capacity to fully implement the desired functions. The methodology for capacity is very similar to the performance methodology. Each block in the target application can again be identified with a corresponding benchmark function and the portion of a target device required to implement the block computed. Any device needing less than 100% to implement the target application is a potential candidate for implementing the target application.

Table 11 shows the percentage of the device the benchmark functions require in several Actel devices. Remember to use the benchmarks optimized for speed if that performance is required to meet the applications requirements since faster implementations require more logic modules and thus a larger percentage of the device capacity. This simple tabular format will make it easier to estimate capacity for the target application.

Estimating Capacity of the Example Application

Using the example application in Figure 7, remembering the correspondence between application blocks and benchmark functions, and utilizing the Actel device capacity given in Table 11, Table 12 can be constructed. Notice that the non-speed critical logic needs to be identified with a benchmark since the capacity estimate must include all logic, not just the speed critical portions. The multiplexers and registers are most similar to the datapath benchmark and will each be counted as equivalent to one datapath benchmark in capacity. Since these paths are not speed critical they can use the capacity optimized datapath numbers. The percentage of the target device capacity required for each block is listed below the appropriate device name. The individual capacities are then summed to determine the total capacity required. Any device which requires more than 100% to implement the target application should not be considered for implementation. The estimate shows the example application should fit in the A1225A, with 6% of the device to spare, in the A1240A with 39% to spare, but not in the A1425A. The A1225A would be the best fit for the application when both performance and capacity are taken into account.

The PREP Benchmark data is an invaluable tool for estimating which target devices are the best fit for a particular application. In many cases, devices can be excluded from consideration well before the detailed design of the application has been completed. This should help reduce the risk that a target device will not meet performance or capacity requirements.

There are some limitations on the effectiveness of this technique which experienced designers will notice. The next section describes these limitations and suggests ways to augment the PREP Benchmarks with additional data to make these estimations even more accurate.

Table 11 • Benchmark Capacity of Actel Devices

Benchmark	A1225-1	A1240-2	A1280-1	A1425
Datapath (Capacity)	5.6%	3.7%	2%	10%
Datapath (Speed)	7%	5%	2.5%	10%
Timer/Counter	11%	7%	5%	16%
Small State Machine	6.7%	4.3%	2.3%	10%
Large State Machine	17%	10%	5.6%	25%
Arithmetic	17%	11%	6.3	25%
Accumulator (Capacity)	20%	12.5%	6.7%	33%
Accumulator (Speed)	25%	14%	7.7%	33%
Counter	11%	7.7%	4%	17%
Pre-scaled Counter	17%	10%	5.6%	25%
Memory Mapper	5%	3.2%	1.8%	7%

Table 12 • Capacity Fit of the Example Application in Actel Devices

Application Block	Benchmark Block	A1225-1	A1240-2	A1425
Address Counter	Counter	11%	7.7%	17%
ALU	Accumulator (Speed)	25%	14%	33%
Shift Register	Datapath (Speed)	7%	5%	10%
Bus State Machine	Small State Machine	6.7%	4.3%	10%
Data Registers (4)	Datapath (Capacity) X 4	22%	14.8%	40%
Multiplexers (4)	Datapath (Capacity) X 4	22%	14.8%	40%
Total Capacity		**94%**	**61%**	**150%**

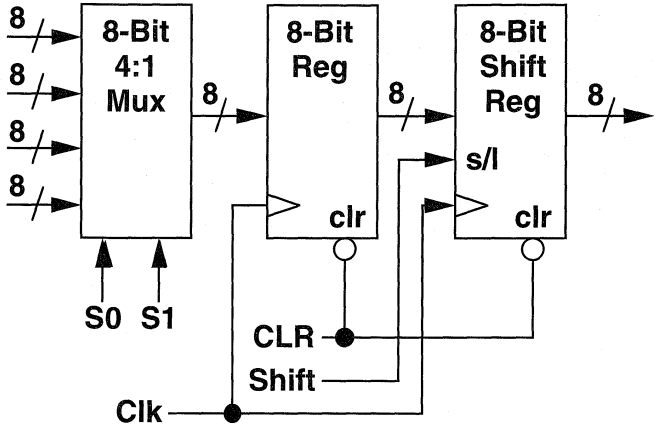
Actel PREP™ Benchmark Results

Datapath

(Benchmark #1)

Functions
• Data Selecting/Holding
• Data Shifting
• 24-Bit Global Data Bus (Routing Intensive)

Actel Characteristics
• Single Logic Level
• Runs at Clock Rate
• Ample Routing Resources



Device/Grade	Capacity	Performance			
	Reps	Worst	Best	Mean	Ext
A1225A-2	18	105	105	105	49
A1240A-2	21	95	95	95	35
A1280A-2	48	85	85	85	48
A1425A-3	10	190	170	184	82
A1460A-3	27	152	151	152	76
A14100A-3	59	104	69	94	48

100% Automatically placed and routed in Designer Series software.

Presentations use or include the most recent PREP PLD Benchmark data which was measured according to Benchmark Suite #1, Version 1.2, dated 3/28/93.

Any analysis is not endorsed by PREP.

5

Timer Counter

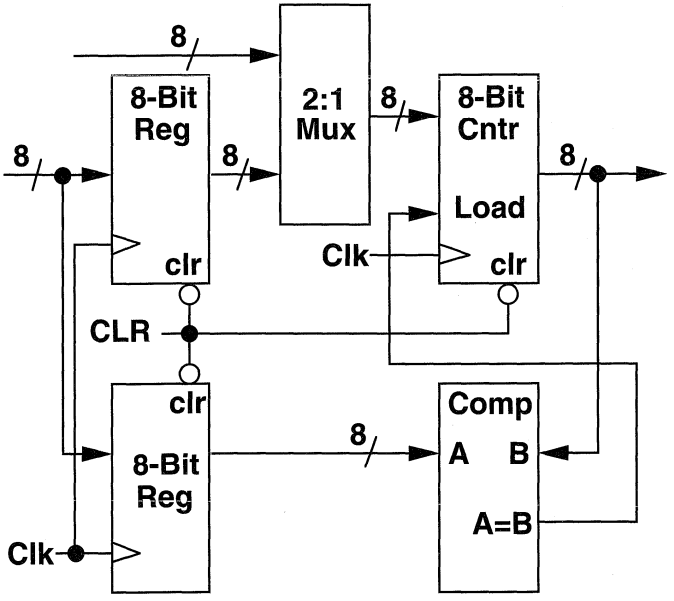
(Benchmark #2)

Functions

- Counting
- Data Comparing/Holding
- Bit Global Data Bus (Routing Intensive

Actel Characteristics

- Register Intensive
- Four Logic Level Compare
- High-Speed Counter
- Ample Routing Resources



Device/Grade	Capacity	Performance			
	Reps	Worst	Best	Mean	Ext
A1225A-2	9	41	37	39	40
A1240A-2	14	35	36	35	24
A1280A-2	26	41	34	38	40
A1425A-3	6	66	57	63	64
A1460A-3	17	66	58	62	58
A14100A-3	29	64	54	58	37

100% Automatically placed and routed in Designer Series software.

Presentations use or include the most recent PREP PLD Benchmark data which was measured according to Benchmark Suite #1, Version 1.2, dated 3/28/93.

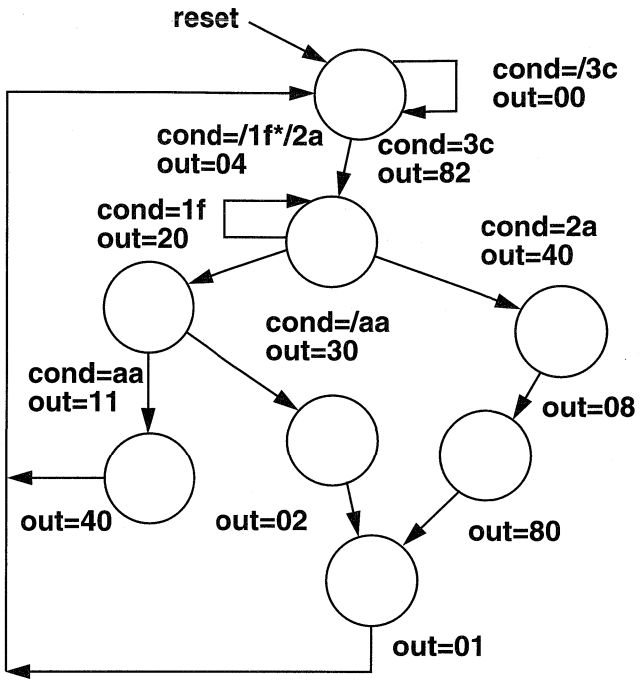
Any analysis is not endorsed by PREP.

State Machine

(Benchmark #3)

Functions
• Simple State Machine
• Eight States, Eight Inputs, Eight Outputs
• Mostly Single Transitions

Actel Characteristics
• Three Logic Levels
• Use Bit Per State



Device/Grade	Capacity	Performance			
	Reps	Worst	Best	Mean	Ext
A1225A-2	16	45	41	43	26
A1240A-2	23	37	41	39	23
A1280A-2	43	44	33	40	25
A1425A-3	10	83	75	77	48
A1460A-3	35	77	67	73	46
A14100A-3	53	78	64	72	45

100% Automatically placed and routed in Designer Series software.

Presentations use or include the most recent PREP PLD Benchmark data which was measured according to Benchmark Suite #1, Version 1.2, dated 3/28/93.

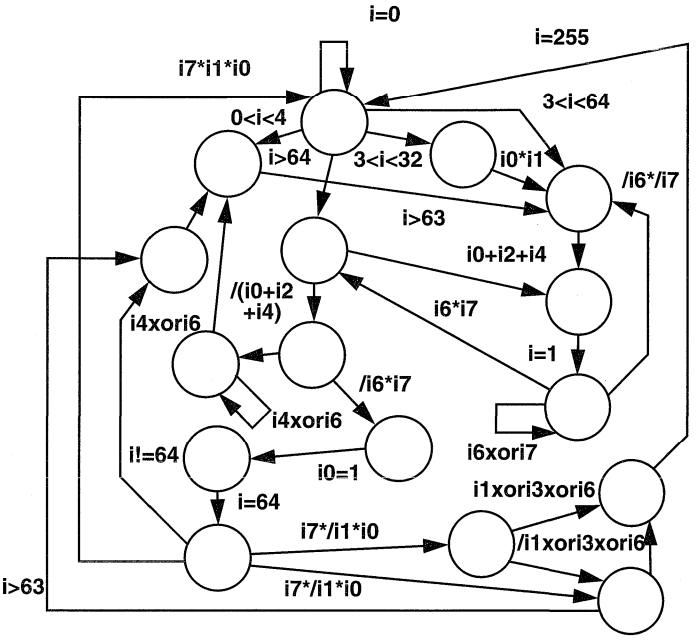
Any analysis is not endorsed by PREP.

Large State Machine

(Benchmark #4)

Functions
State Machine
16 States, 8 Inputs, 8 Outputs
Complex Transactions and Outputs

Actel Characteristics
Five Logic Levels
Use Bit Per State



Device/Grade	Capacity	Performance			
	Reps	Worst	Best	Mean	Ext
A1225A-2	6	26	25	25	18
A1240A-2	10	22	25	24	17
A1280A-2	16	25	23	24	17
A1425A-3	4	47	45	46	30
A1460A-3	12	45	39	41	31
A14100A-3	21	43	31	39	28

100% Automatically placed and routed in Designer Series software.

Presentations use or include the most recent PREP PLD Benchmark data which was measured according to Benchmark Suite #1, Version 1.2, dated 3/28/93.

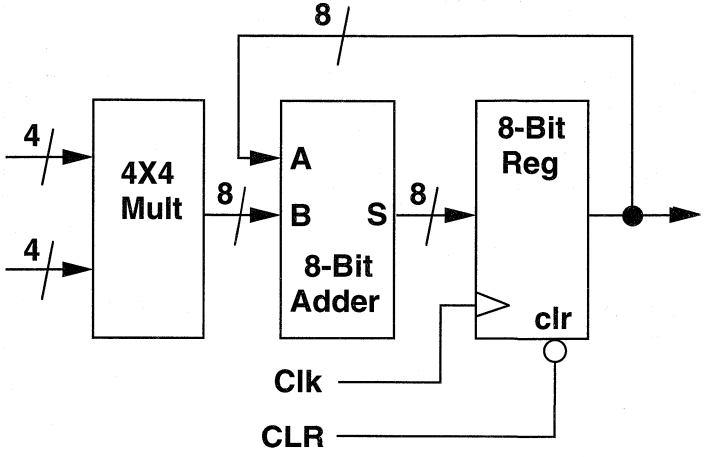
Any analysis is not endorsed by PREP.

Arithmetic

(Benchmark #5)

Functions
• Multiplying
• Adding
• Data Holding

Actel Characteristics
• 10 Logic Levels
• Module Speed Trade-Offs



Device/Grade	Capacity	Performance			
	Reps	Worst	Best	Mean	Ext
A1225A-2	6	21	20	21	16
A1240A-2	9	14	15	15	12
A1280A-2	16	18	16	17	13
A1425A-3	4	32	31	32	25
A1460A-3	9	32	30	31	24
A14100A-3	17	25	22	24	19

100% Automatically placed and routed in Designer Series software.

Presentations use or include the most recent PREP PLD Benchmark data which was measured according to Benchmark Suite #1, Version 1.2, dated 3/28/93.

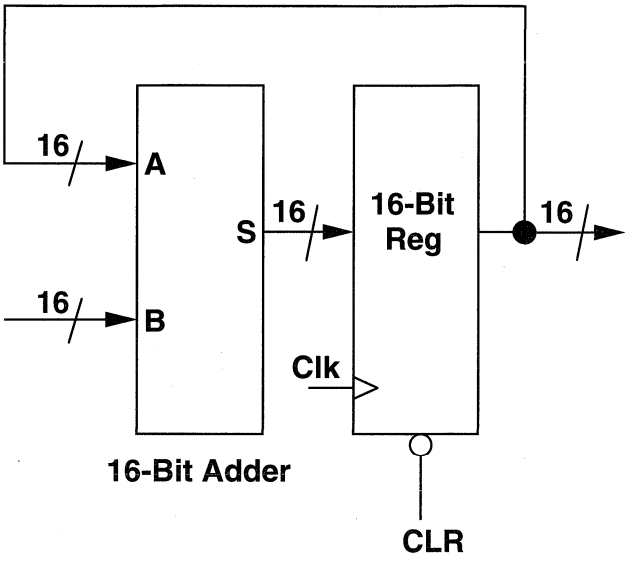
Any analysis is not endorsed by PREP.

Accumulator

(Benchmark #6)

Functions
• 16-Bit Adder
• 16-Bit Register

Actel Characteristics
• Three Logic Levels
• Parallel Implementation Faster than Competitor's Serial Approaches
• Module Speed Trade-Offs



Device/Grade	Capacity	Performance			
	Reps	Worst	Best	Mean	Ext
A1225A-2	5	39	34	37	23
A1240A-2	8	27	30	29	20
A1280A-2	15	36	30	33	24
A1425A-3	3	62	58	60	39
A1460A-3	10	63	53	58	39
A14100A-3	17	52	43	48	34

100% Automatically placed and routed in Designer Series software.

Presentations use or include the most recent PREP PLD Benchmark data which was measured according to Benchmark Suite #1, Version 1.2, dated 3/28/93.

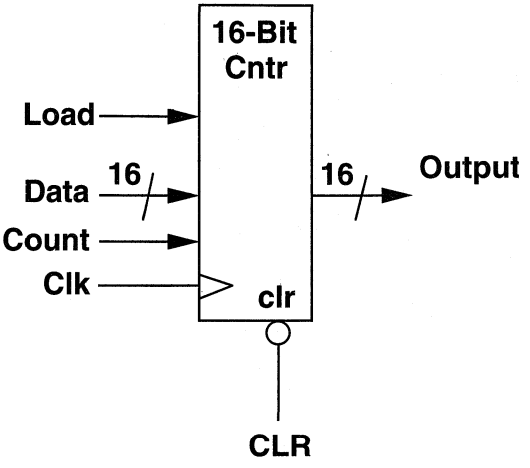
Any analysis is not endorsed by PREP.

16-Bit Counter

(Benchmark #7)

Functions
• Counting
• Loading
• Count After Load

Actel Characteristics
• Two Logic Levels
• Uses Look Ahead to Cut Logic Levels



Device/Grade	Capacity	Performance			
	Reps	Worst	Best	Mean	Ext
A1225A-2	9	70	63	66	34
A1240A-2	13	52	64	58	32
A1280A-2	25	67	53	58	28
A1425A-3	6	110	95	104	58
A1460A-3	17	111	84	99	51
A14100A-3	28	101	76	90	49

100% Automatically placed and routed in Designer Series software.

Presentations use or include the most recent PREP PLD Benchmark data which was measured according to Benchmark Suite #1, Version 1.2, dated 3/28/93.

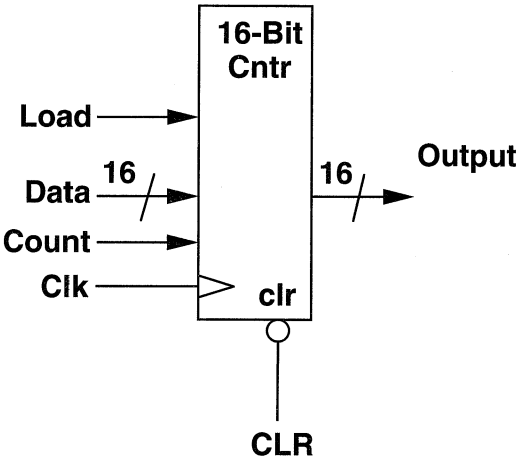
Any analysis is not endorsed by PREP.

16-Bit Prescaled Counter

(Benchmark #8)

Functions
• Counting
• Multicycle
• Uses Prescalar for LSBs

Actel Characteristics
• Single Logic Level
• Runs at Full Clock Rate



Device/Grade	Capacity	Performance			
	Reps	Worst	Best	Mean	Ext
A1225A-2	6	105	105	105	42
A1240A-2	10	95	95	95	20
A1280A-2	19	85	71	84	41
A1425A-3	4	198	165	179	71
A1460A-3	12	152	133	146	63
A14100A-3	21	149	104	145	67

100% Automatically placed and routed in Designer Series software.

Presentations use or include the most recent PREP PLD Benchmark data which was measured according to Benchmark Suite #1, Version 1.2, dated 3/28/93.

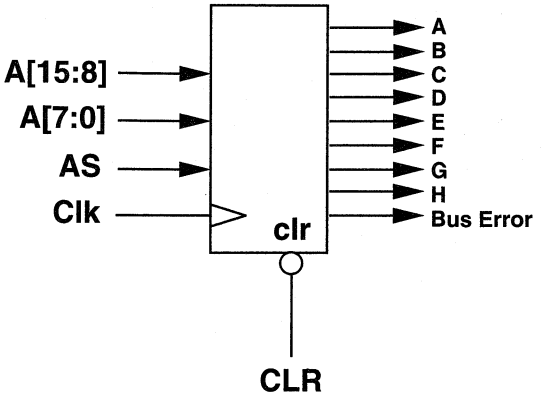
Any analysis is not endorsed by PREP.

Memory Map

(Benchmark #9)

Functions
• Address Decode
• Address Hold
• Wide Gating
• 8-Bit Global Address Bus (Routing Intensive

Actel Characteristics
• Three Logic Levels
• Abundant Routing



Device/Grade	Capacity	Performance			
	Reps	Worst	Best	Mean	Ext
A1225A-2	20	46	41	43	27
A1240A-2	31	37	42	39	23
A1280A-2	56	44	36	41	26
A1425A-3	14	77	66	71	44
A1460A-3	38	73	64	68	44
A14100A-3	62	73	58	67	42

100% Automatically placed and routed in Designer Series software.

Presentations use or include the most recent PREP PLD Benchmark data which was measured according to Benchmark Suite #1, Version 1.2, dated 3/28/93.

Any analysis is not endorsed by PREP.

Macro Libraries

Section 6: Macro Libraries

ACT 1 Hard Macro Library Overview

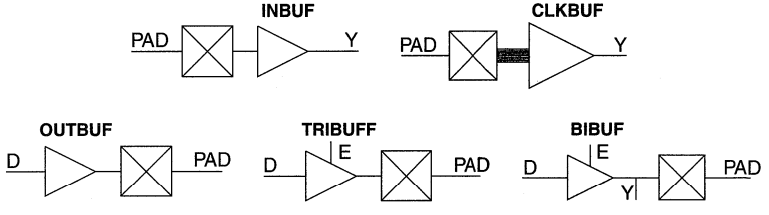
The following illustrations show all the available hard macros. The *ACT Family Macro Library Guide* contains module count, combinability, and pin loading information of each hard macro. It also includes a complete truth table for each macro.

Most ACT 1 hard macros are implemented by a single logic module. The following ACT 1 hard macros require two logic modules to implement:

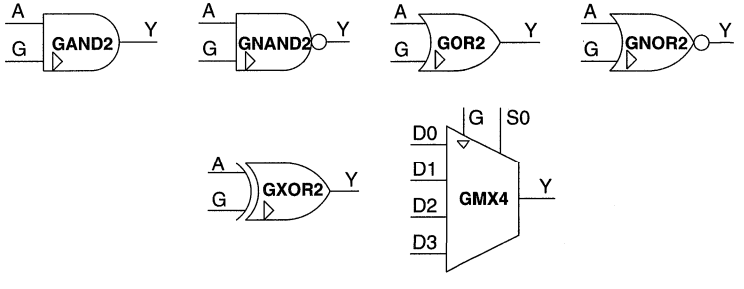
- All flip-flops
- All adders
- Two-module hard macro gates: AND4, AND4A, AND4D, AOI1, AOI4, NAND3, NAND4, NAND4A, NAND4B, NOR4, NOR4C, NOR4D, OAI3, OR3C, OR4B, OR4C, OR4D

Two-module hard macro gates have a "2" displayed on some input pins. This indicates that the input to output path has two levels of logic delay for these input pins only. Also, the full adders have a "2" on the "S" output pins. This indicates that there are two levels of logic delay from the input pins to the "S" output pins. Refer to the ACT 1 Timing Characteristics for detailed timing information of all ACT 1 macros.

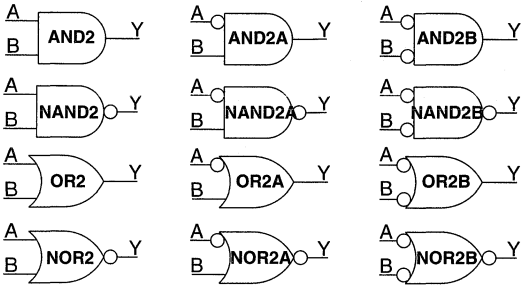
Input and Output Buffers



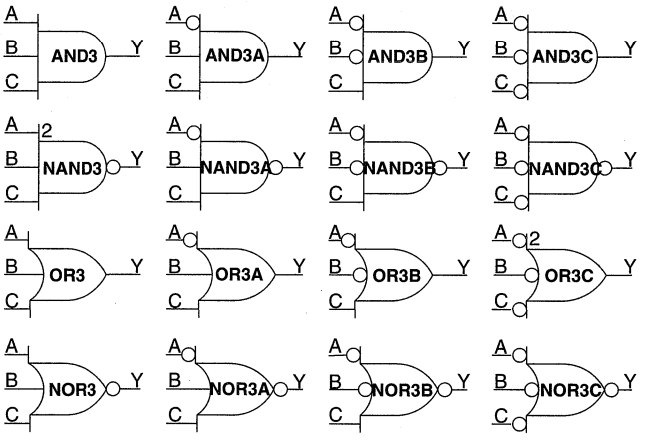
CLKBUF Interface Macros



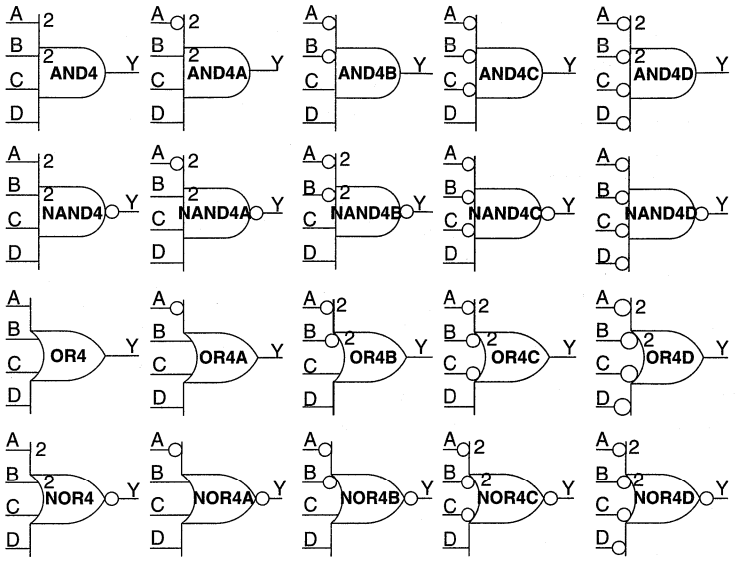
2-Input Gates



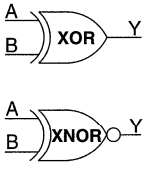
3-Input Gates



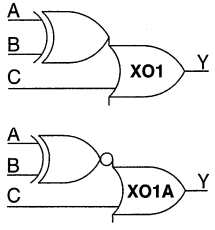
4-Input Gates



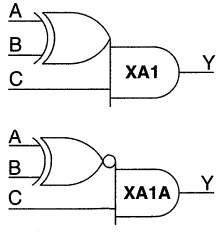
XOR Gates



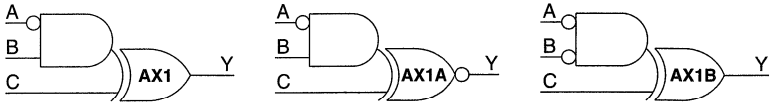
XOR-OR Gates



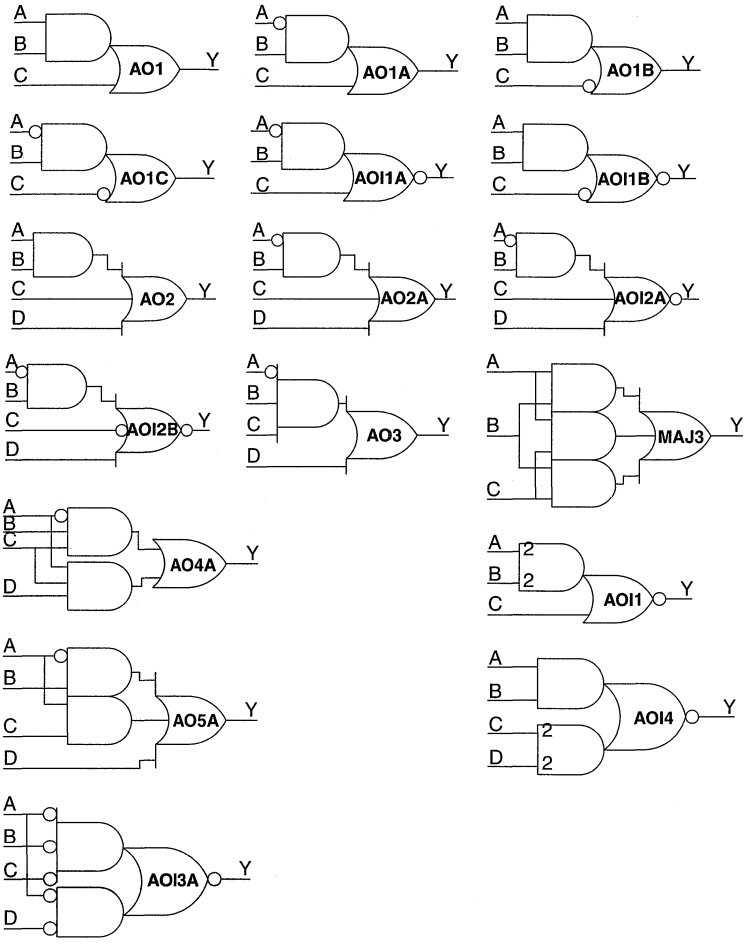
XOR-AND Gates



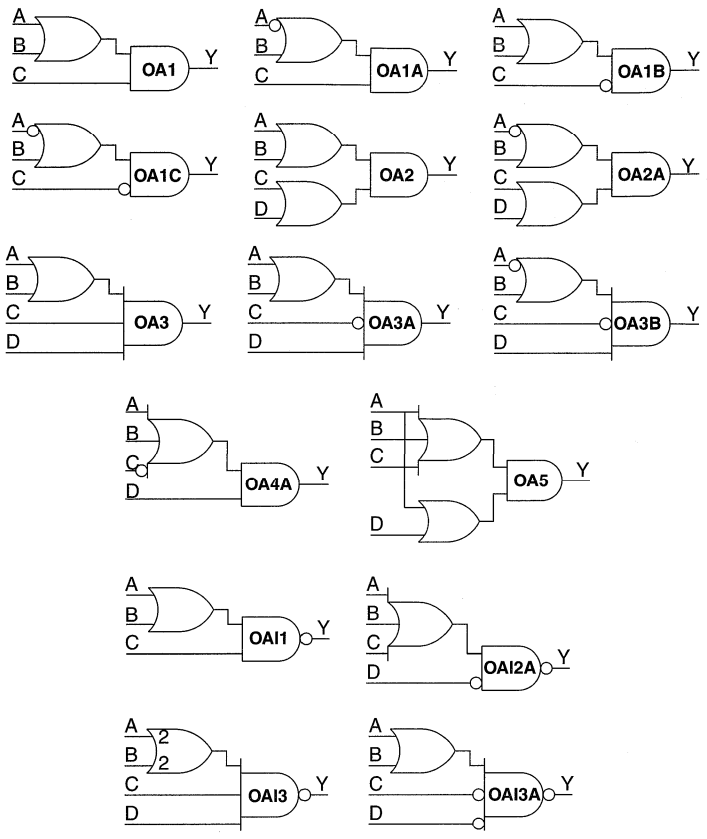
AND-XOR Gates



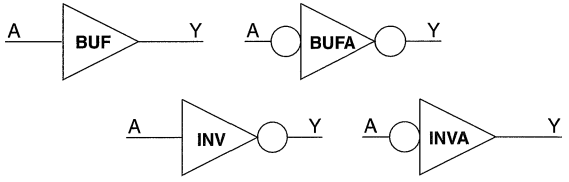
AND-OR Gates


A
B
C
AO1
Y
A
B
C
AO1A
Y
A
B
C
AO1B
Y
A
B
C
AO1C
Y
A
B
C
AOI1A
Y
A
B
C
AOI1B
Y
A
B
C
D
AO2
Y
A
B
C
D
AO2A
Y
A
B
C
D
AOI2A
Y
A
B
C
D
AOI2B
Y
A
B
C
D
AO3
Y
A
B
C
MAJ3
Y
A
B
C
D
AO4A
Y
A
B
C
2
2
AOI1
Y
A
B
C
D
AO5A
Y
A
B
C
2
D
2
AOI4
Y
A
B
C
D
AOI3A
Y

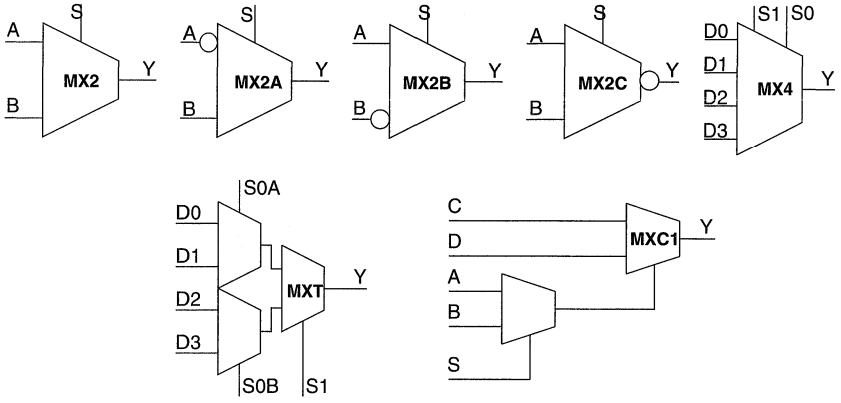
OR-AND Gates



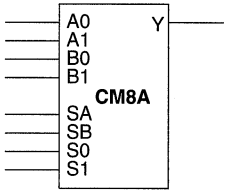
Buffers



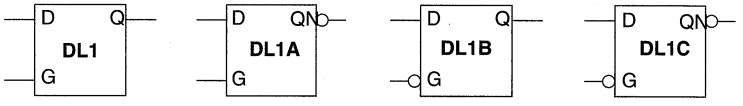
Multiplexors



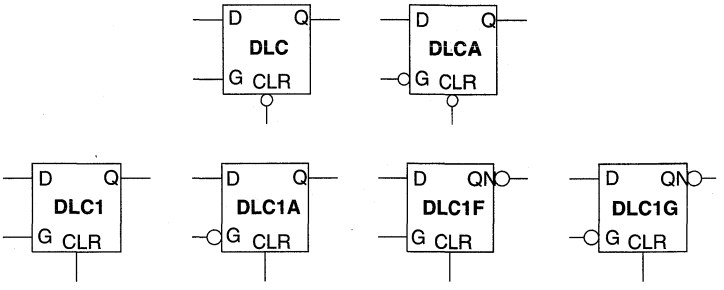
Combinatorial



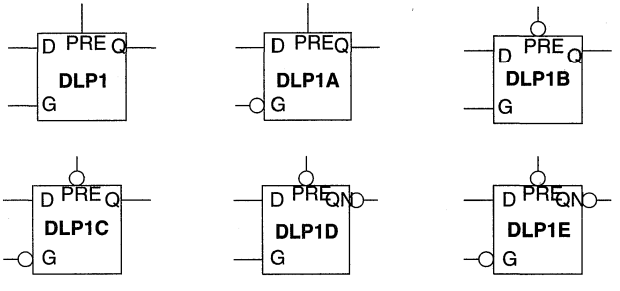
D-Latches



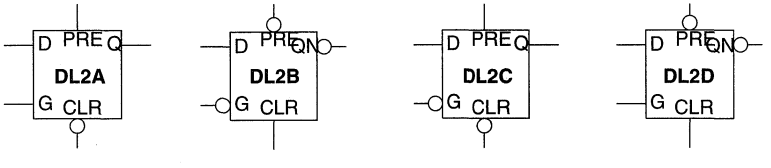
D-Latches with Clear



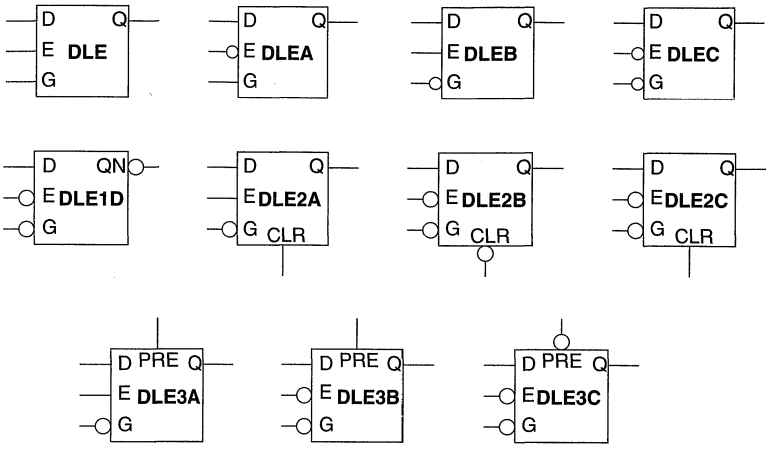
D-Latches wilth Preset



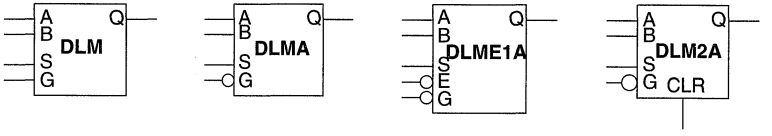
D-Latches with Preset and Clear



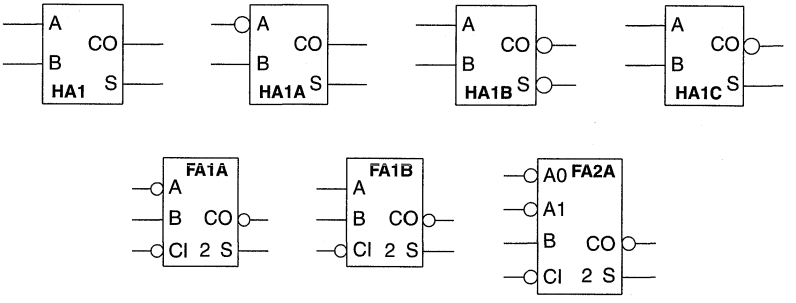
D-Latches with Enable



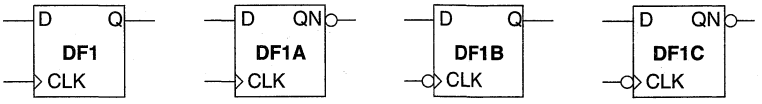
Mux Latches



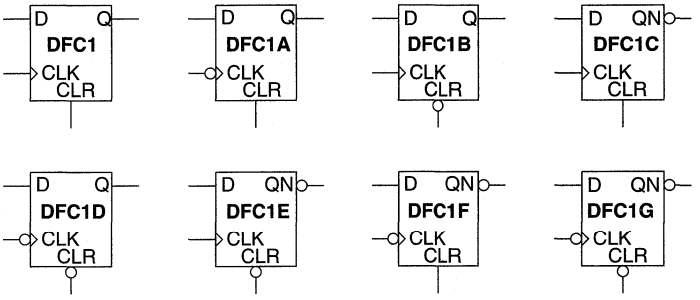
Adders



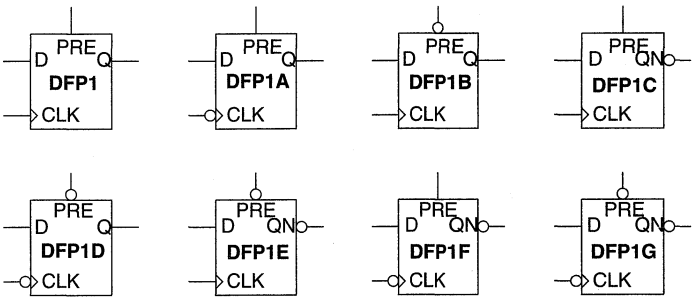
D-Type Flip-Flops



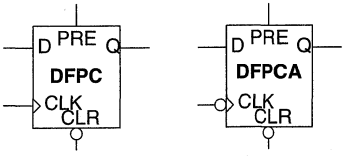
D-Type Flip-Flops with Clear



D-Type Flip-Flops with Preset

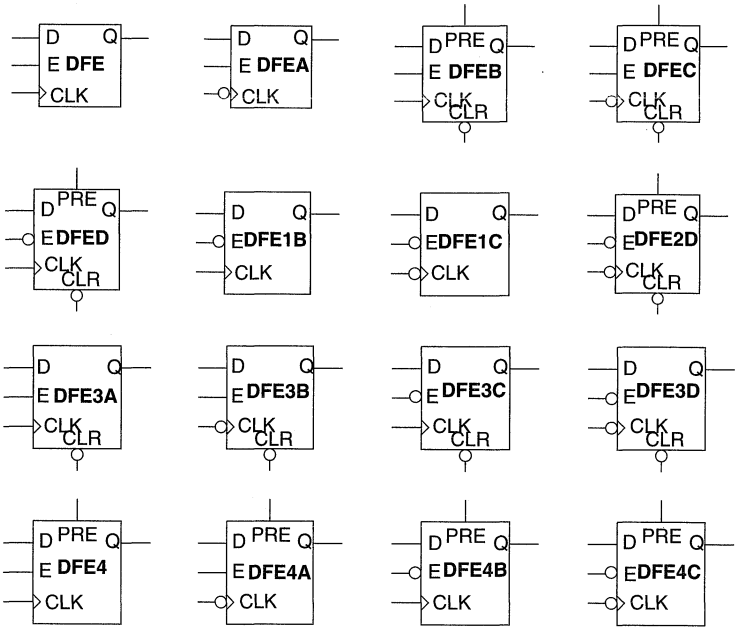


D-Type Flip-Flops with Preset and Clear

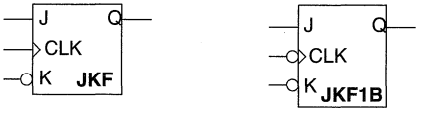


6

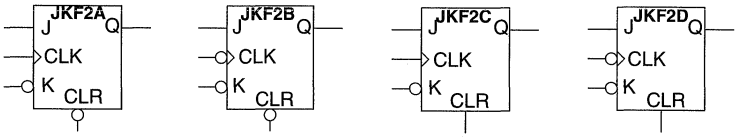
D-Type Flip-Flops with Enable



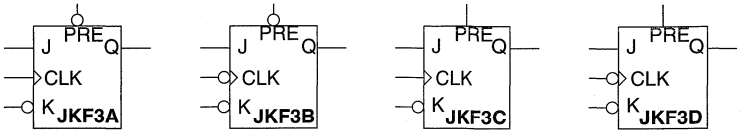
JK Flip-Flops



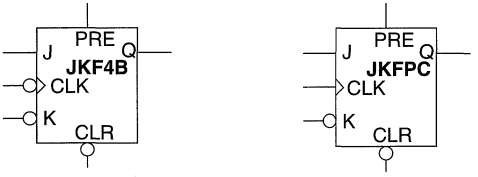
JK Flip-Flops with Clear



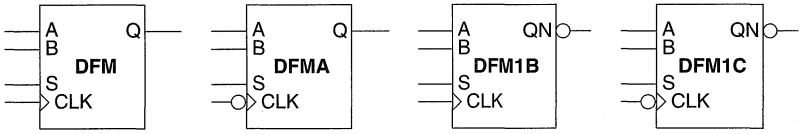
JK Flip-Flops with Preset



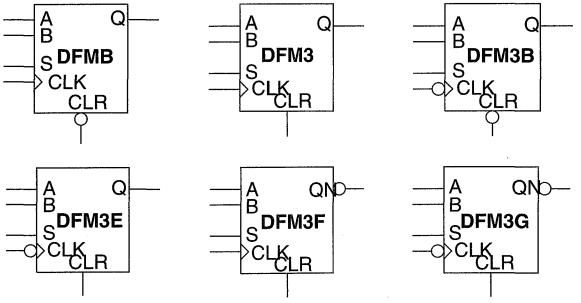
JK Flip-Flops with Preset and Clear



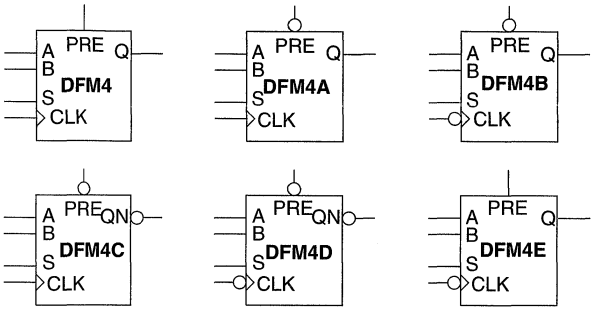
Mux Flip-Flops



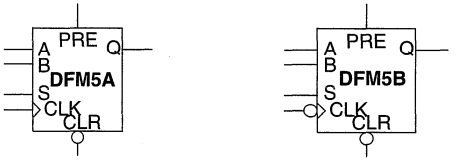
Mux Flip-Flops with Clear



Mux Flip-Flops with Preset



Mux Flip-Flops with Preset and Clear



ACT 2, 1200XL, and ACT 3 Hard Macro Library Overview

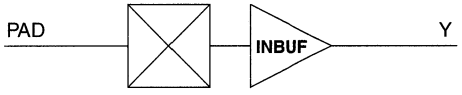
The following illustrations show all the available hard macros. The *ACT Family Macro Library Guide* contains module count, combinability, and pin loading information of each hard macro. It also includes a complete truth table for each macro.

Most ACT 2, 1200XL, and ACT 3 hard macros are implemented by a single logic module. The following ACT 2, 1200 XL, and ACT 3 hard macros require more than one logic module to implement:

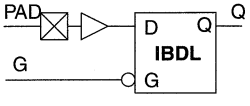
- Two-module hard macro gates: AND4D, AOI4, AX1A, MXC1, MXT, NAND4, NOR4, OAI3, OR4D
- Two-module adders: FA1A, FA1B, FA2A, HA1, HA1A, HA1B, HA1C
- Two-module latches: DL2A, DL2B, DL2C, DL2D, DLE2A, DLE3A, DLM2A
- Two-module flip-flops: DFC1, DFC1A, DFC1E, DFC1G, DFM3, DFM3E, DFP1, DFP1A, DFP1B, DFP1C, DFP1D, DFP1F, DFPC, DFPCA, JKF2C, JKF2D, JKF3A, JKF3B, JKF3C, JKF3D, JKF4B, JKFPC

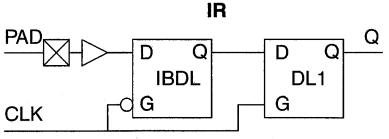
Two-module hard macro gates have a "2" displayed on some input pins. This indicates that the input to output path has two levels of logic delay for these input pins only. Also, the full adders have a "2" on the "S" output pins. This indicates that there are two levels of logic delay from the input pins to the "S" output pin. Refer to the ACT 2, 1200XL, and ACT 3 Timing Characteristics for detailed timing information of all ACT 2 and ACT 3 macros.

Input Buffers

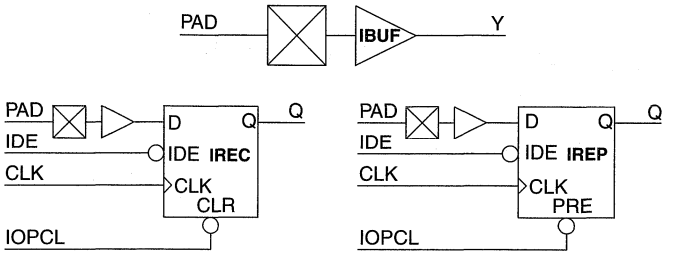


Input Buffers (ACT 2 only)

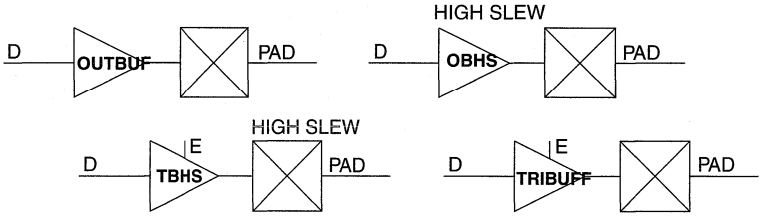




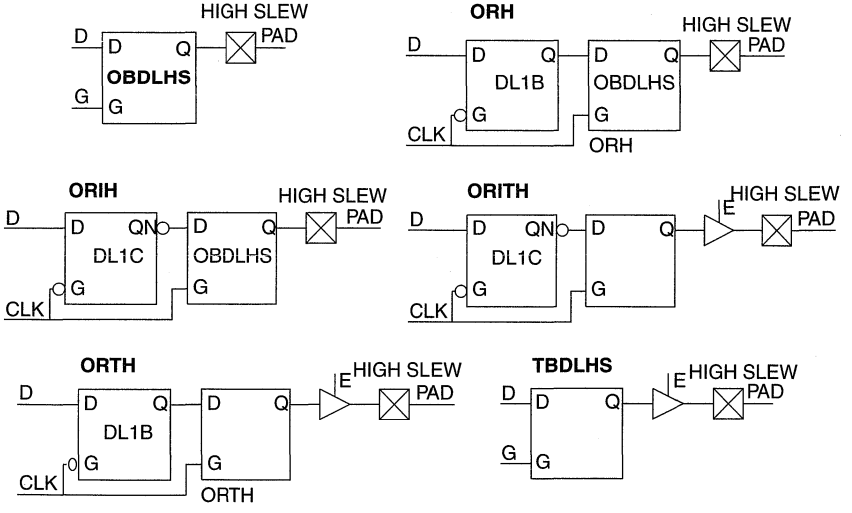
Input Buffers (ACT 3 only)



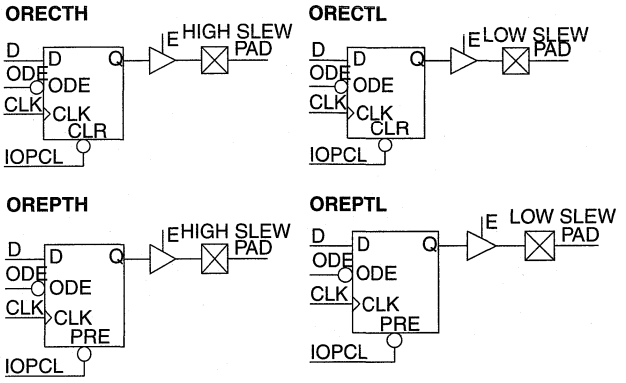
Output Buffers



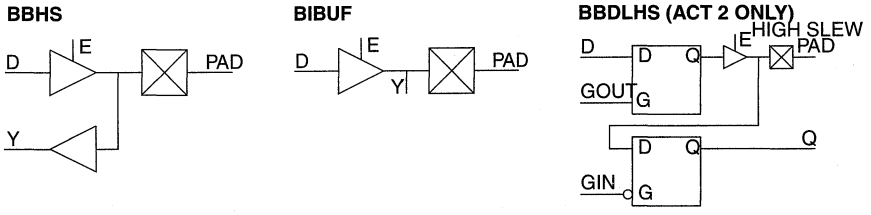
Output Buffers (ACT 2 only)



Output Buffers (ACT 3 only)

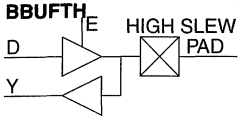


Bidirectional Buffers

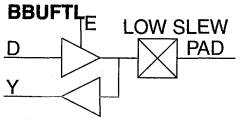


Bidirectional Buffers (ACT 3 only)

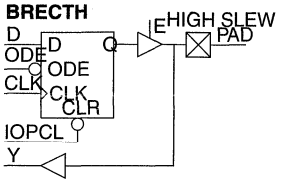
BBUFTH



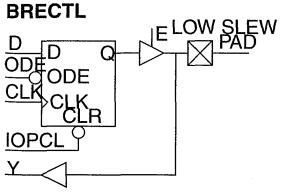
BBUFTL



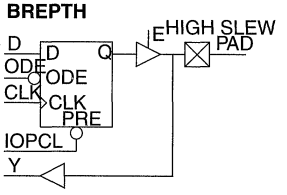
BRECTH



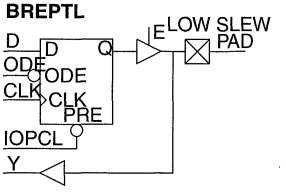
BRECTL



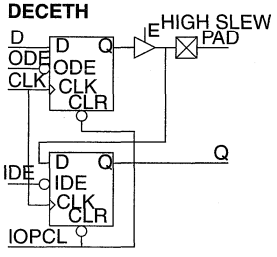
BREPTH



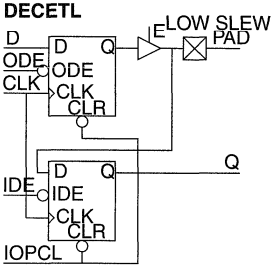
BREPTL



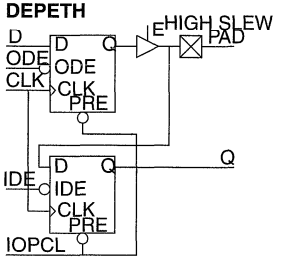
DECETH



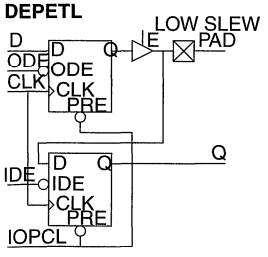
DECETL



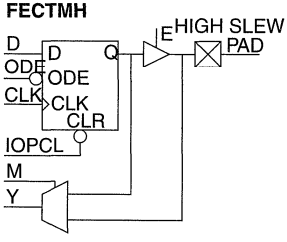
DEPETH



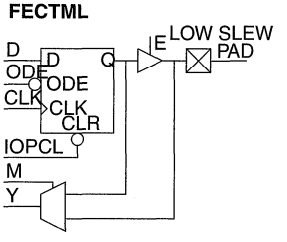
DEPETL



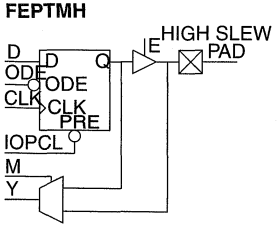
FECTMH



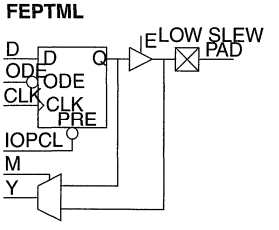
FECTML



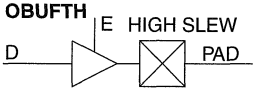
FEPTMH



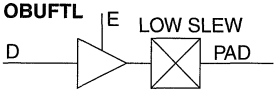
FEPTML



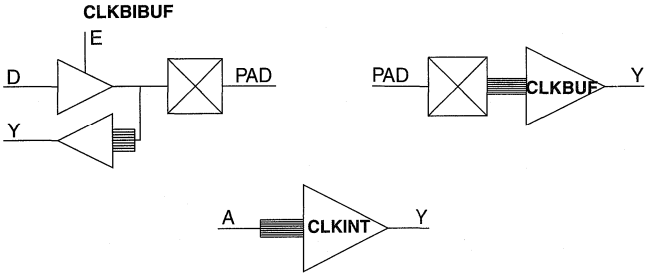
OBUFTH



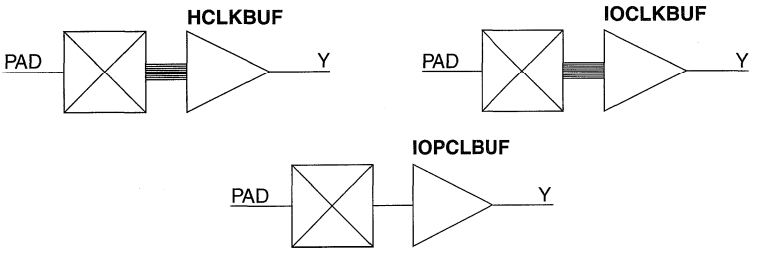
OBUFTL



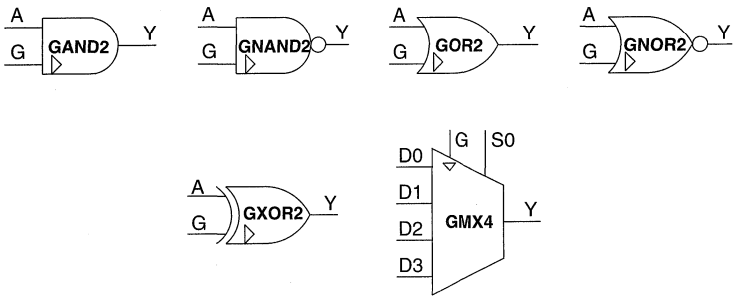
Clock (Dedicated Network) Buffers



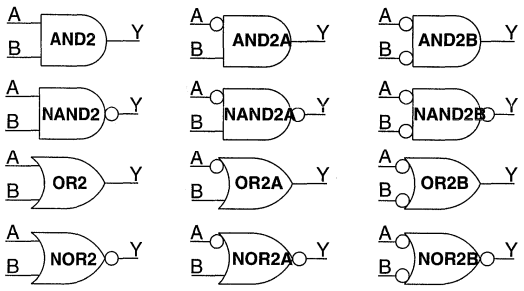
Clock (Dedicated Network) Buffers (ACT 3 only)



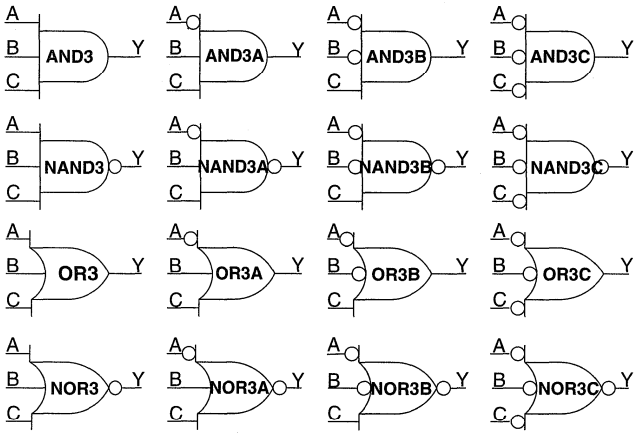
CLKBUF Interface Macros



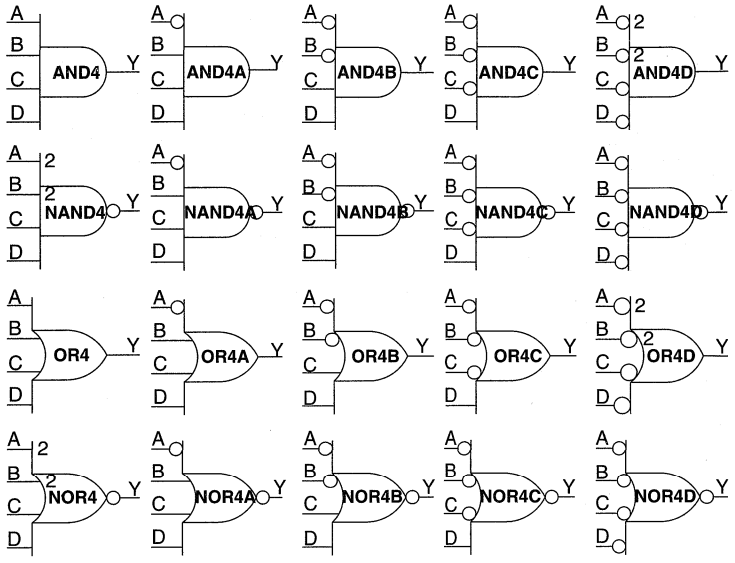
2-Input Gates



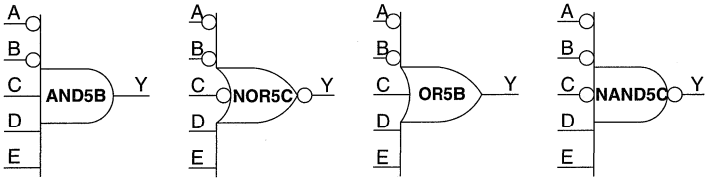
3-Input Gates



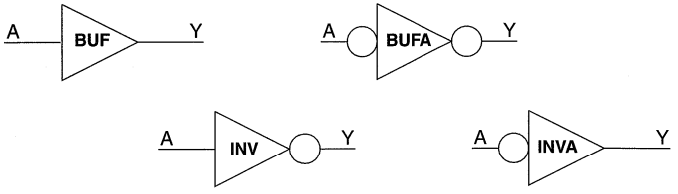
4-Input Gates



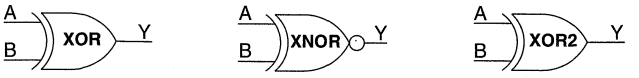
5-Input Gates



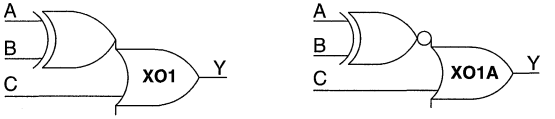
Buffers



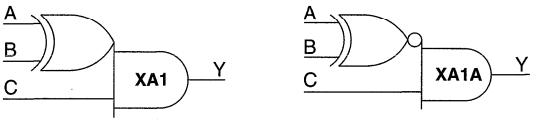
XOR Gates



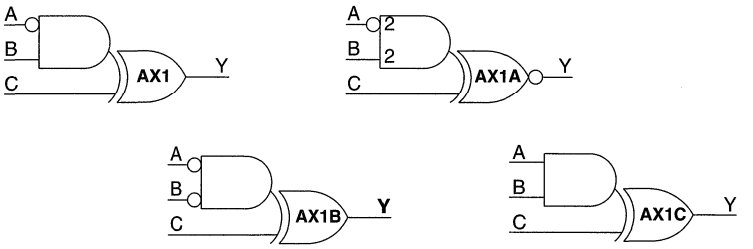
XOR-OR Gates



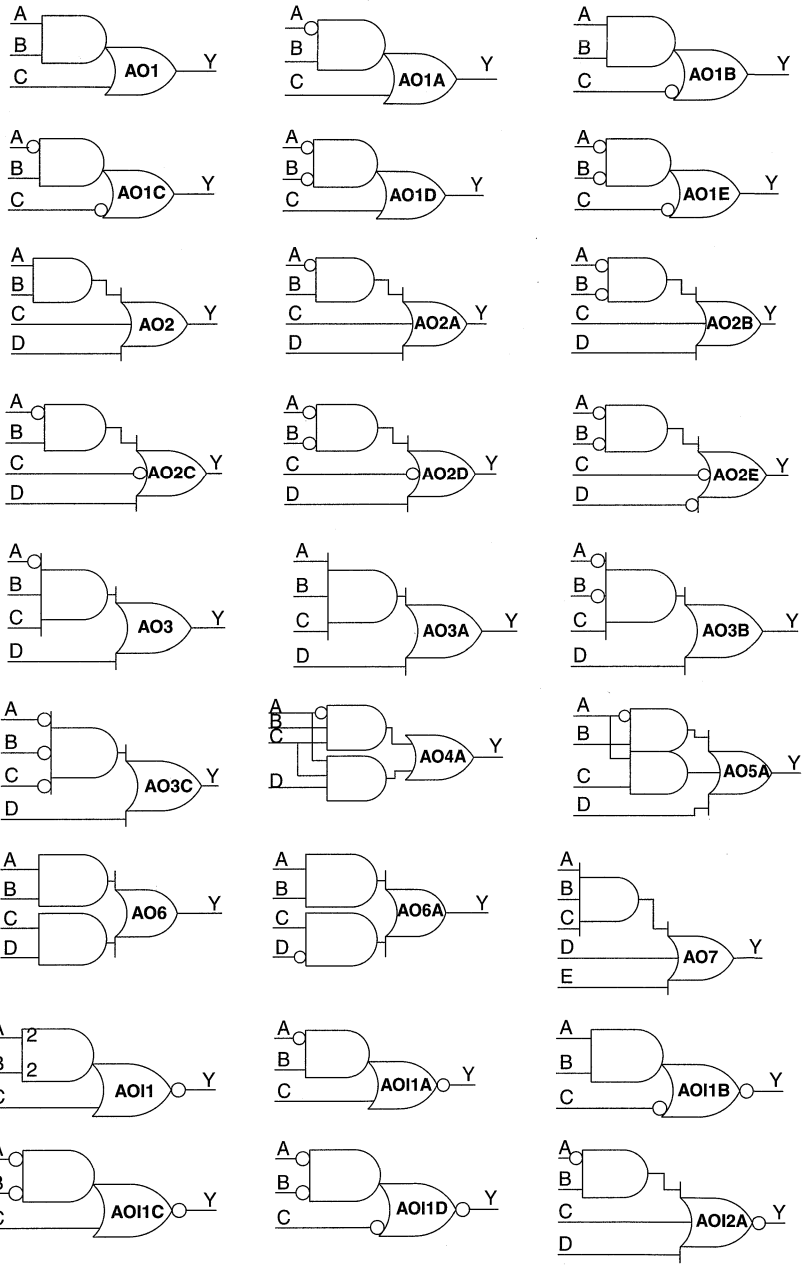
XOR-AND Gates



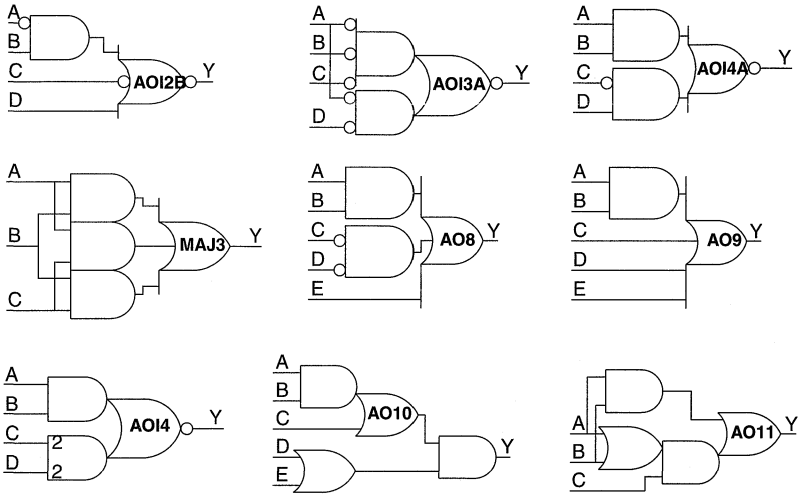
AND-XOR Gates



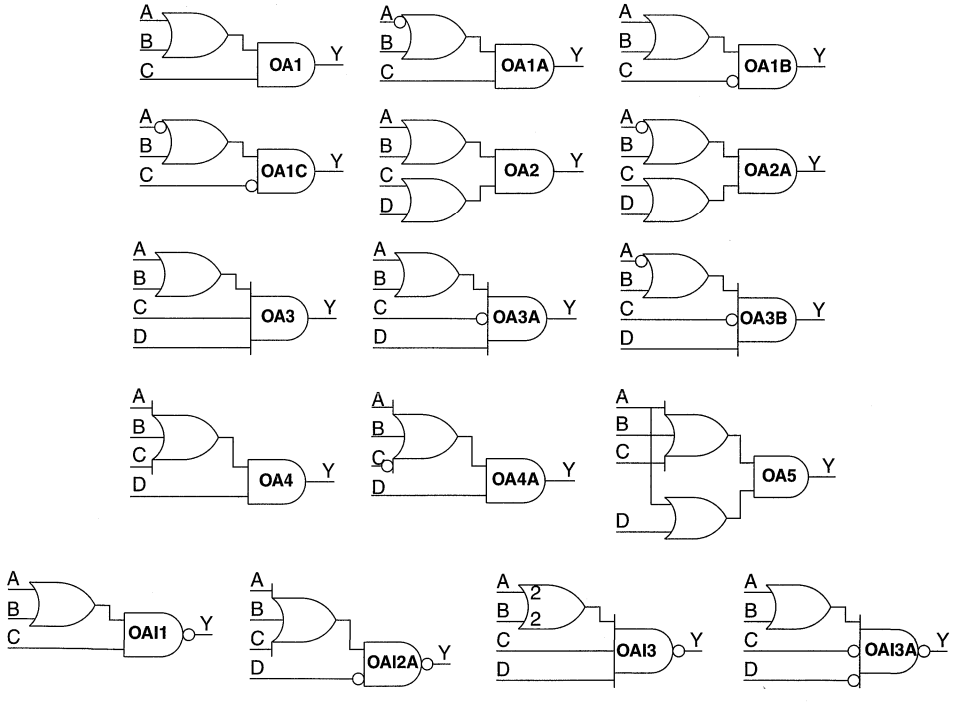
AND-OR Gates


A
B
C
AO1
Y
A
B
C
AO1A
Y
A
B
C
AO1B
Y
A
B
C
AO1C
Y
A
B
C
AO1D
Y
A
B
C
AO1E
Y
A
B
C
D
AO2
Y
A
B
C
D
AO2A
Y
A
B
C
D
AO2B
Y
A
B
C
D
AO2C
Y
A
B
C
D
AO2D
Y
A
B
C
D
AO2E
Y
A
B
C
D
AO3
Y
A
B
C
D
AO3A
Y
A
B
C
D
AO3B
Y
A
B
C
D
AO3C
Y
A
B
C
D
AO4A
Y
A
B
C
D
AO5A
Y
A
B
C
D
AO6
Y
A
B
C
D
AO6A
Y
A
B
C
D
E
AO7
Y
A
B
C
AOI1
Y
A
B
C
AOI1A
Y
A
B
C
AOI1B
Y
A
B
C
AOI1C
Y
A
B
C
AOI1D
Y
A
B
C
D
AOI2A
Y

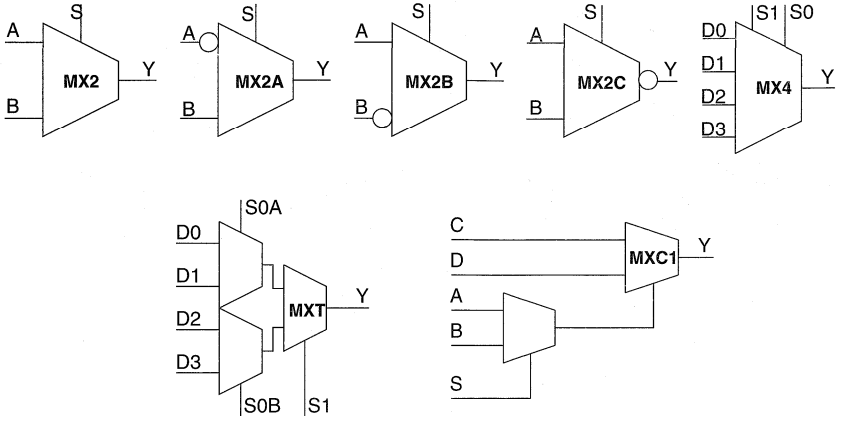
AND-OR Gates (continued)



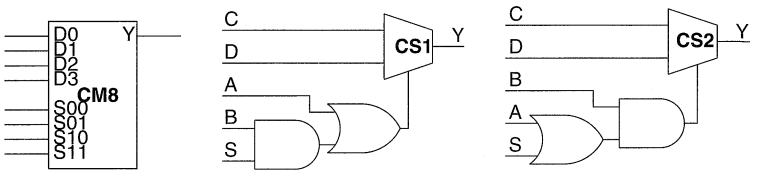
OR-AND Gates



Multiplexors

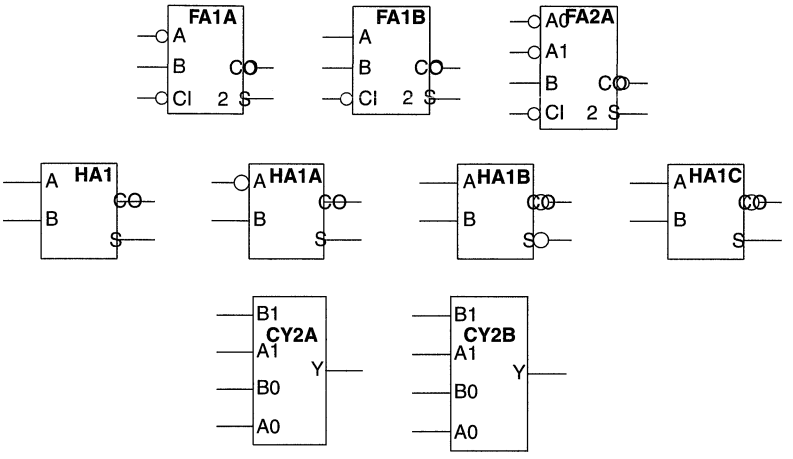


Combinatorial

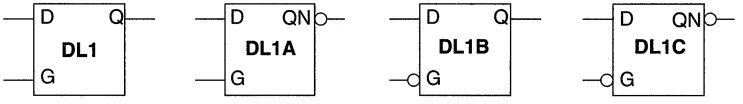


6

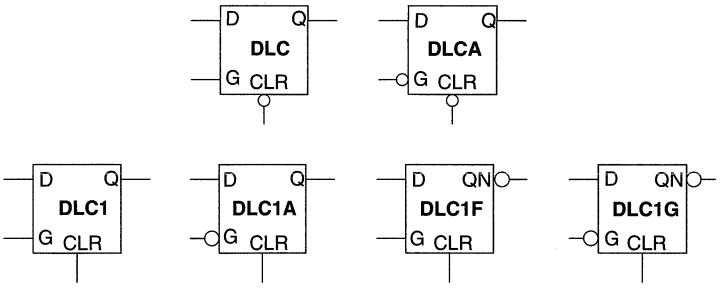
Adders



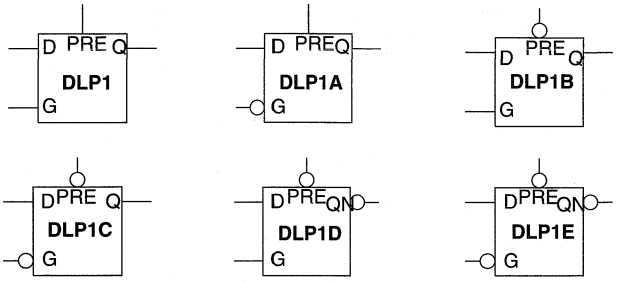
D-Latches



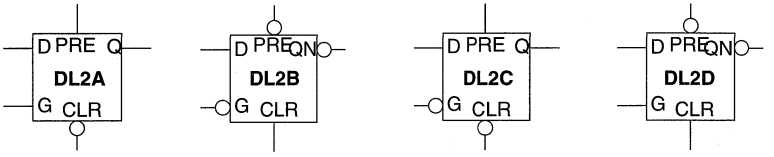
D-Latches with Clear



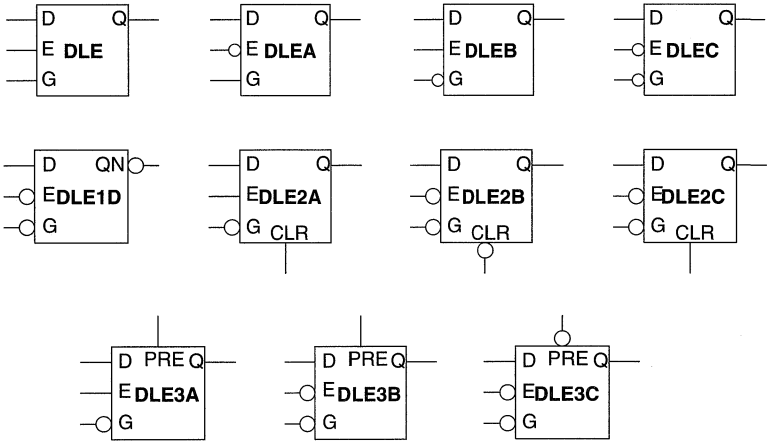
D-Latches with Preset



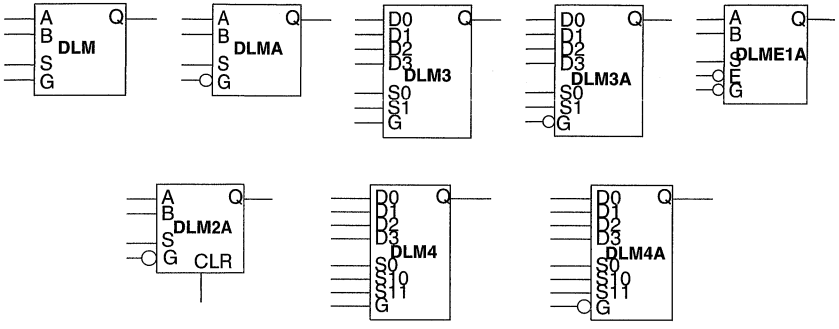
D-Latches with Preset and Clear



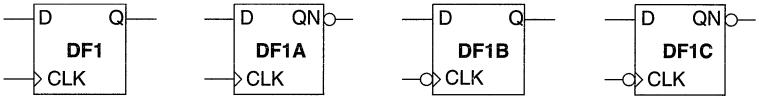
D-Latches with Enable



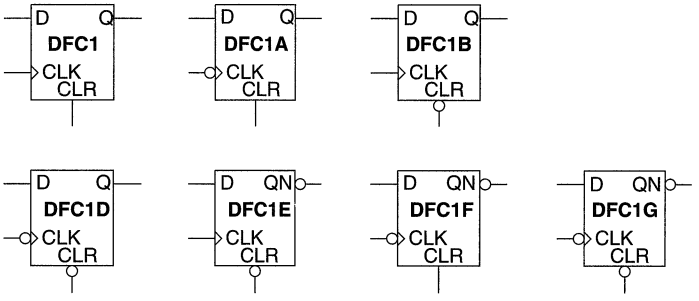
Mux Latches



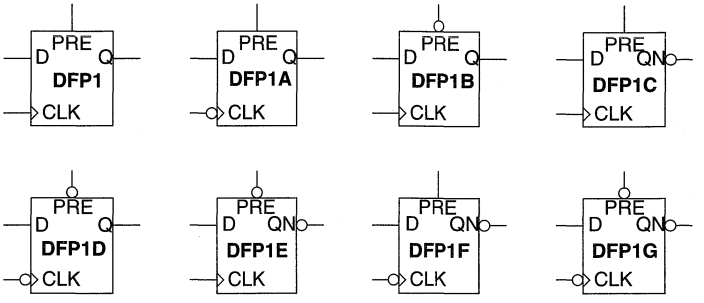
D-Type Flip-Flops



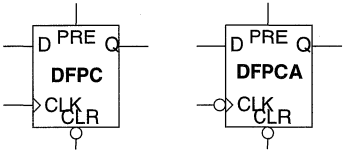
D-Type Flip-Flops with Clear



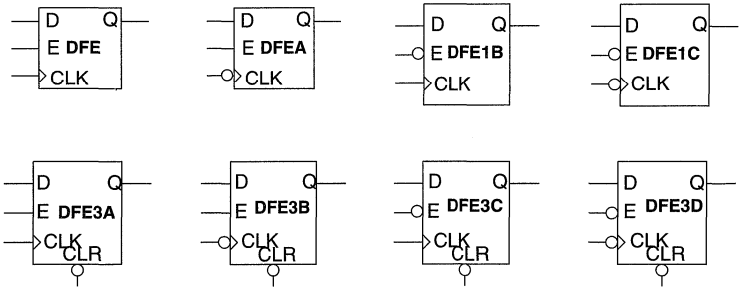
D-Type Flip-Flops with Preset



D-Type Flip-Flops with Preset and Clear

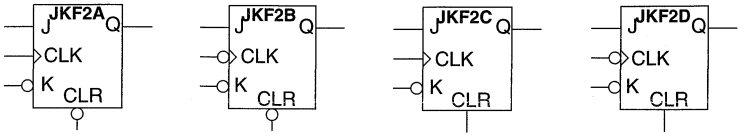


D-Type Flip-Flops with Enable

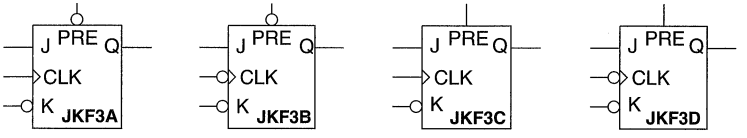


JK Flip-Flops

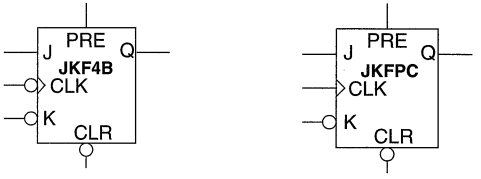
JK Flip-Flops with Clear



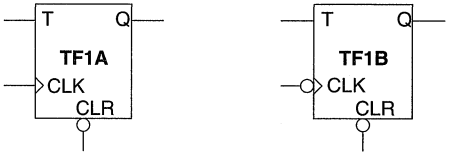
JK Flip-Flops with Preset



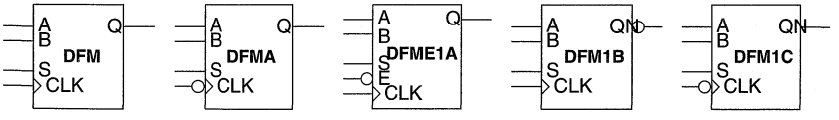
JK Flip-Flops with Preset and Clear



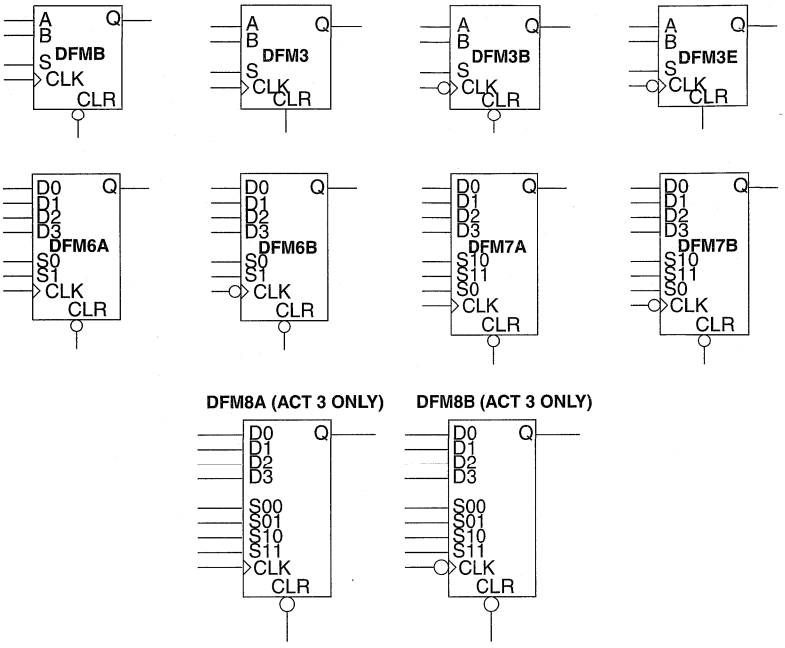
Toggle Flip-Flops



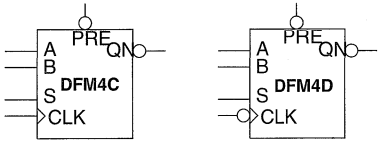
Mux Flip-Flops



Mux Flip-Flops with Clear



Mux Flip-Flops with Preset



Development Tools

Section 7: Development Tools

An Overview of the Actel Designer and Designer Advantage Systems

The Actel Designer and Designer Advantage™ systems are high-productivity, computer-aided engineering environments for the Actel family of field programmable gate array (FPGA) devices. The Designer System can handle all Actel FPGA families up to 2500-gates, while the Designer Advantage System handles every Actel device, from the smallest to the largest.

Environment

Depending on whether you work on a PC or a workstation, you can use either the Designer or Designer Advantage systems outfitted with our new, easy-to-use Microsoft Windows™ (PC) or X Window™ (workstation) graphical user interfaces. A key advantage of either system is that they let you take designs from concept to silicon in hours, without costly non recurring engineering (NRE) expenses.

Each Designer Series system gives you placement and routing, timing verification, and programming interface software, as well as interfaces to many popular CAE and electronic design automation platforms such as Cadence™, Mentor Graphics™, OrCAD™, Synopsys and Viewlogic™. In addition, both Designer Series systems support programming on Data I/O's Unisite and 3900 series programmers.

Another benefit is that both systems are very open. You can continue using your existing design environments to enter VHDL descriptions, capture schematics, enter PAL or Boolean equations, simulate, and perform timing analysis. That lets you design Actel FPGAs in your chosen environment.

Design Creation

Designer Series systems let you create Actel FPGA designs several different ways. No matter if you use schematic capture, synthesis or PAL design methodologies to implement your logic, the Designer Series supports you.

Schematic Capture

You can enter your design by drawing a schematic with symbols generated by the ACTgen Macro Builder, or from the Actel Macro Library. This Library has more than 500 different hard and soft macro functions, including everything from simple gates to complex functions. For added design flexibility, you can create your own application-specific macros using the ACTgen Macro Builder. Currently, the Macro Library and the ACTgen Macro Builder support most popular schematic capture and simulation systems such as Cadence, Mentor Graphics, and Viewlogic.

Logic Synthesis

Another way to describe Actel FPGA designs is through synthesis using hardware description languages such as VHDL or Verilog HDL. You'll find that Actel FPGAs give you excellent results with most popular logic synthesis tools, including Cadence's PIC Designer, Exemplar Logic's CORE, IST's ASYL+, Mentor Graphics' Autologic, Synopsys's Design Compiler and FPGA Compiler, and Viewlogic's VHDL Synthesis.

ACTmap VHDL, included in all Designer Series development systems, gives you a powerful, entry-level VHDL synthesis solution. Optimized for users familiar with PAL design entry methodologies, ACTmap VHDL synthesizes VHDL language descriptions into an Actel FPGA and supports the optimization of EDIF or Actel netlist formats, as well as PALASM. No matter which input format you choose, the synthesis process can optimize your design for maximum gate utilization or device performance.

Synopsys Libraries

If you'd like, we can also supply you with a technology library for the Synopsys logic synthesis environment, including versions for the Sun and HP workstation platforms. Our library lets you capture Actel designs by entering VHDL or Verilog HDL format source files into the Synopsys Design Compiler or FPGA Compiler. The compiler then creates an EDIF netlist, which is read by the Actel supplied EDIF reader in the Designer Advantage environment.

The Synopsys library fully supports DesignWare with ACTgen Macro Builder macros and lets you instantiate more than 50,000 different macro functions. The Synopsys compiler also automatically infers high-performance ACTgen macros for addition, subtraction and magnitude comparison operators.

Furthermore, the Synopsys library supports the Synopsys VHDL System Simulator (VSS), which lets you perform both pre- and post-route simulations of your VHDL design description.

Design Processing

The Actel Designer Series automatically adapts your design to Actel-specific design rules, so that it works in an Actel FPGA. The Designer Series software also calculates routability and performs design rule checks before routing. During processing, you'll get error information messages and system warnings if there are any electrical rule violations such as excessive fanout, shorted output, and unconnected inputs. When the process is finished, you can be confident that your design obeys all Actel FPGA electrical rules and that it's ready for placement and routing.

Automatic Place and Route

Our automatic place and route software minimizes design delay by assigning macros to optimal locations on the chip. The software uses the design netlist, critical net information, and I/O assignments to automatically place and route all the logic blocks within the circuit. Placement and routing is fully automatic, even at high device utilization.

To minimize delay, the route program assigns the shortest possible connections between logic modules, routing 100% of the nets automatically for up to 95% logic module utilization. With the highly efficient and well planned routing resources in the Actel architecture, nets will have tight delay distributions and more predictable design performance. For even greater control, you can create speed critical nets by assigning any of four levels of criticality to the net—and have the place and route software minimize the net delay for you.

DirectTime™ Layout (optional)

By the middle of 1995, we will add a powerful, new design capability to our Designer Series systems called DirectTime Layout. This new capability will give you an extraordinary degree of control over the timing in your design. With DirectTime, you will be able to specify the overall operating frequency and pin-to-pin timing requirements as you create your design. Then during place and route, DirectTime will use those requirements to create an FPGA that achieves the performance characteristics you specified.

Our place and route software also gives you data on the actual interconnect delay. You'll be able to use this information with the DirectTime Analyzer timing verification tool—or choose to back annotate to a simulation netlist for accurate post-route timing simulations.

Incremental Place and Route

This function lets you quickly iterate an existing design by making small logic changes without changing the entire placement and routing. After the first automatic placement and routing, you can specify how much of the design can be changed to accommodate design modifications. Then, you can lay out the design again to include the changes—and only the sections that have been modified will go through place and route.

Because it uses the initial automatic place and route file, Incremental Place and Route can save you significant design time. It also gives you greater design flexibility, since you have the freedom to make design changes while preserving most of the original performance characteristics.

Design Analysis

The Designer Series systems allow you to analyze the functional and timing characteristics of your design several different ways. For instance, you can perform static timing analysis, or use back annotation capabilities to give you links to timing simulation.

Timing Analysis

Actel's static timing analyzer gives you complete point-to-point timing information and allows you to specify which signals you want reported.

In mid-1995, we will release the DirectTime Analyzer—an interactive timing analysis tool that graphically displays the performance of all critical and non critical paths within a specified design. The DirectTime Analyzer will perform a static timing analysis, using the post-route delays extracted from the layout. Afterwards, it will give you a spreadsheet view of the timing results, which you can use to optimize your design to meet timing specifications. When used with DirectTime Layout, the Analyzer provides a system that implements the timing you specify and allows you to verify that you've achieved the timing you want.

Functional and Timing Simulation

The Designer Series systems work with most popular simulators, including those from Cadence Design System, Data I/O, Intergraph, Mentor Graphics, OrCAD, SusieCAD and Viewlogic. With the Designer Series, you can perform a functional analysis of your design before place and route—an important step in verifying logical correctness. You can also get a post-route back annotation file that can be used by these simulators to validate the timing of your design.

Programming

After you complete a final timing analysis, the Designer Series systems produce device programming files that can be used by either Data I/O or Actel Activator™ series programmers. Designer Series software generates a fuse map for the specific device that you're programming. This map specifies the programming of the PLICE™ antifuse and determines interconnections within the FPGA. If you have Data I/O Unisite or Series 3900 programmers, you can program any Actel-certified FPGA device.

Naturally, our own Activator 2 and Activator 2S programmers can program all Actel FPGAs. The Activator 2 can handle up to four devices of the same design simultaneously, while the Activator 2S programs one device at a time. Simply mount the appropriate programming adapter for your device and package selection—and you're ready to go.

In-Circuit Test and Debug

After a device is programmed and functionally verified, it's ready to operate in the system. To assure that the device performs "in system" according to specifications, you can test it using the optional Actionprobe™ diagnostic tool.

This tool works with either an Activator 2 or Activator 2S programming system and has FPGA diagnostic probe pins that connect to the target system. Under full control of the Actionprobe software, using the two diagnostic probe pins, you can select and analyze any two internal design signals during system operation. If you need timing and waveform analysis, you can also use an oscilloscope or a logic analyzer.

The Actel Annual Support Program

Join the Actel Annual Support Program, and we'll update your Designer and Designer Advantage software automatically to support the most recently released devices and packages. Typically, we release software updates at least three times a year, free of additional charges.

Technical Support Hotline

As an Annual Support Program member, you'll also receive technical assistance on both Actel software and hardware products via the Actel Technical Hotline. All hotline calls are answered by our Technical Message Center, which records your name, company name, phone number and question. The information is forwarded to an Actel applications engineer who will try to solve the problem and return your call within an hour. The Hotline phone hours are from 7:00 AM to 5:30 PM PST. Besides answering calls, our applications engineers can also assist our customers by writing application notes, users guides, creating design examples and evaluating software.

Electronic Bulletin Board Service (BBS)

We also give you information access and transfer through a 24-hour, worldwide bulletin board. You can download information such as new soft macros and software updates. In addition, our application engineers can help you solve design problems over the BBS. In fact, you can upload your design to a private BBS directory and have an Actel applications engineer check your design and make the necessary adjustments.

Our BBS telephone number is 408-739-6397. To connect to the BBS by modem, you'll need the following equipment: a modem with a 9600 BPS baud rate; a data format set at 8 data bits, 1 stop bit, no parity. In addition, we support Xmodem, Ymodem, Zmodem and ASCII.

You can establish your account with your first call. You'll be prompted for your name, company, phone number and other important information. Once your account is set up, call our Technical Support Hotline at 800-262-1060 and request a file transfer class. Out Technical Message Center then verifies your name and company and upgrades your account's security level.

Training

We offer an introductory 2-day course that covers all aspects of designing an Actel FPGA, including design methodologies, a brief look at Viewlogic schematic capture and simulation tools, and a thorough examination of Actel design software. Classes include a mix of lectures and lab exercises and require a minimum of three students. To register for classes, just call our Technical Support Hotline.

Action Facts

This is our 24-hour fax-back service through which you can get a list of current software improvements, application notes, design hints, package pinouts, and more. Just call the toll-free number and ask us to fax a catalogue to you. Action Facts is updated bimonthly and will give you up to five documents per call. The number is 800-262-1062.

International e-mail

Recently, we've introduced a new way for you to communicate with Actel: e-mail. Now you can simply e-mail your technical questions or design files to tech@actel.com and you'll get answers back via e-mail or phone. Typically, you'll get a response to your questions within two days.

Designer Advantage System with Cadence Composer/Verilog Design Kit

Development System Capabilities

Actel's Designer Advantage™ system for the Cadence™ design environment is a low-cost field programmable gate array (FPGA) design, programming, and verification software system for all the Actel FPGAs, including the 10K gate A14100. The system consists of Actel's design software and libraries for the Cadence Composer environment, which contains macro libraries and simulation models for all three families of Actel FPGAs. Included in the design software is ACTmap™ VHDL Synthesis, which will synthesize VHDL for Actel FPGAs, and the ACTgen Macro Builder, which creates complex logic functions according to parameters you select. Available options are the Activator® 2 and Activator 2S programmers and Actionprobe® diagnostic tools.

Cadence Design Flow (with Verilog)

Figure 1 shows how the Designer Advantage software works with Cadence's Composer schematic-capture and Verilog simulation tools. You create a schematic description by using Composer. The ACTgen Macro Builder enhances schematic or synthesis-based capture methods by creating macro functions that are customized for your application. You specify the macro type, bit width, control signals, and so on for your macro, and the Macro Builder does the rest—automatically. Custom, complex functions can be generated in minutes. Then your design files are exported from Composer directly into the Designer Advantage environment, which runs under the X-Window System on the Sun and HP700 workstations.

After the files are imported into the Designer Advantage system, the schematic design can be combined with VHDL descriptions by using ACTmap VHDL. PALs®, Boolean equation descriptions, or state machines can be merged into the design. ACTmap can be used both to synthesize the VHDL into an Actel FPGA-specific description and to optimize pieces of the design for better utilization or performance. A specific Actel device is selected, and the Design Validator then verifies design rule compliance by completing an electrical rules check and providing statistical information such as utilization percentage and average fanout. After validation, the automatic place and route software implements the engineer's design as an FPGA programming file. After automatic placement and routing, the design can be modified by using the incremental place and route tool to make localized changes to your design. After placement and routing, the Timer, a static timing analysis tool, verifies circuit timing and delays. Alternatively, postroute simulations can be done with a backannotated delay file to the Cadence Verilog simulator. Once the design is complete, the Activator programmer programs the proper antifuses to configure the FPGA. The Actionprobe diagnostic hardware and software allow you to explore the operation of a programmed FPGA by providing observability of internal nodes while the FPGA is in the target system.

Software Requirements

Sun Workstation

- Sun OS version 4.1.3 (or later)
- Sun OpenWindows version 2.0 (or later)
- Cadence Composer version 4.2.1, 4.3 (or later)
- Cadence Verilog version 1.7, 1.8, 1.9, 2.0 (or later)

HP700 Workstation

- HPUX version 9.01 (or greater)
- Cadence Composer version 4.2.1, 4.3 (or later)
- Cadence Verilog version 1.7, 1.8, 1.9, 2.0 (or later)

Hardware Requirements

Sun Workstation

- Sun SPARC or SPARC2
- 32 MB RAM (minimum)
- 60 MB hard disk space (minimum)
- CD-ROM drive

HP700 Workstation

- HP700 series workstation
- 32 MB RAM (minimum)
- 60 MB hard disk space (minimum)
- CD-ROM drive

Programmers for Actel FPGAs

- Activator 2S programmer—single device programmer requires at least one programming adapter (see Table 1)
- Activator 2 programmer—programs up to four devices simultaneously and requires at least one programming adapter
- Data I/Os Unisite and 3900 series programmers

Actionprobe Diagnostic Tools

Actionprobe diagnostic tools provide 100 percent real-time observability of internal nodes while the FPGA is running in the target system. Any node in your design can be probed while your FPGA is operating. This observability is a unique feature that reduces the time required for design verification and debugging.

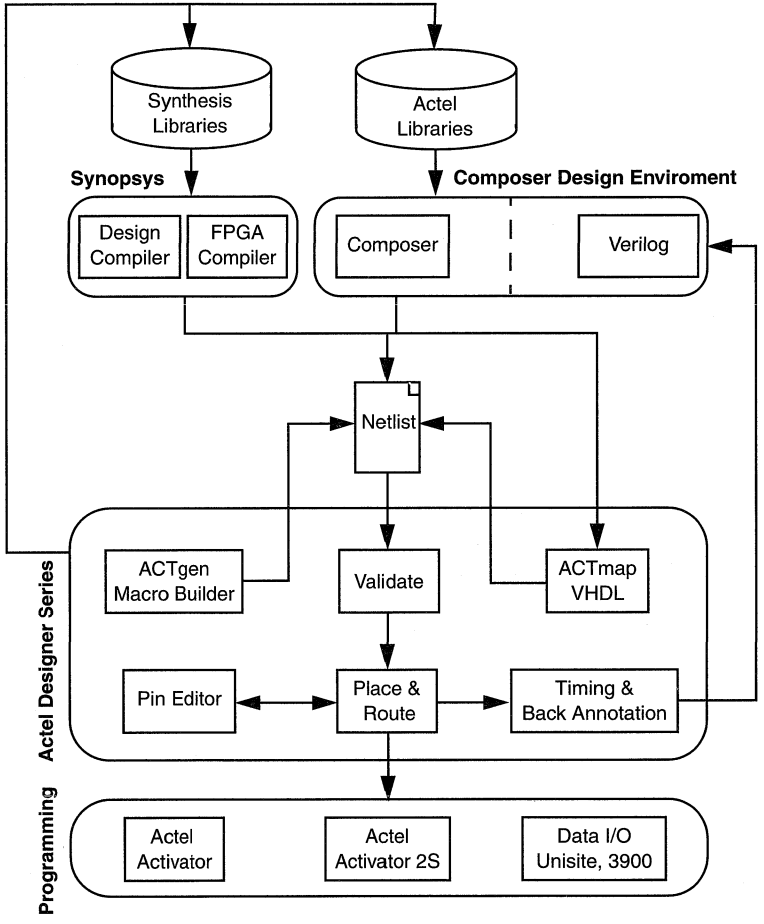


Figure 1 • Flow Diagram

Table 1 • Activator Programming Adapters

ACT 1 Programming Adapters	Package	ACT 2/1200XL Programming Adapters	Package	ACT 3 Programming Adapters	Package
ALS-280	100-PQFP	ALS-286	132-CPGA	MA3-PG100	100-CPGA
ALS-281	44-PLCC	ALS-287	176-CPGA	MA3-PG133	133-CPGA
ALS-282	68-PLCC	ALS-289	100-CPGA	MA3-PG175	175-CPGA
ALS-283	84-PLCC	ALS-290	100-PQFP	MA3-PG207	207-CPGA
ALS-284	84-CPGA	ALS-292	144-PQFP	MA3-PG257	257-CPGA
ALS-285	84-CQFP	ALS-293	160-PQFP	MA3-PL84	84-PLCC
MA1-VQ80	80-VQFP	ALS-294	172-CQFP	MA3-QF100	100-PQFP
		MA2-PL84	84-PLCC	MA3-QF160	160-PQFP
		MA2-TQ176	176-TQFP	MA3-QF208	208-PQFP
		MA2-VQ100	100-VQFP	MA3-QF208	208-RQFP
				MA3-TQ176	176-TQFP
				MA3-VQ100	100-VQFP
				MA3-BG225	225-BGA
				MA3-BG313	313-BGA

Table 2 • Designer Advantage System Selector Guide

System Part Number	Actel FPGA Design Software	Programming	Actionprobe Diagnostics
DA-SN-CDB	≤ 10,000 gates	Activator 2 or 2S	Optional
DA-HP7-CDB	≤ 10,000 gates	Activator 2 or 2S	Optional

Designer Advantage System with Cadence Concept/Verilog Design Kit

Development System Capabilities

Actel's Designer Advantage™ system for the Cadence™ design environment is a low-cost field programmable gate array (FPGA) design, programming, and verification software system for all Actel FPGA families, including the 10K gate A14100. The system consists of Actel's design software and libraries for the Cadence Concept environment, which contains macro libraries and simulation models for all three families of Actel FPGAs. Included in the design software is ACTmap™ VHDL Synthesis, which will synthesize VHDL for Actel FPGAs, and the ACTgen Macro Builder, which creates complex logic functions according to parameters you select. Available options are the Activator® 2 and Activator 2S programmers and Actionprobe® diagnostic tools.

Cadence Design Flow (with Verilog)

Figure 1 shows how the Designer Advantage software works with Cadence's Concept schematic-capture and Verilog simulation tools. You create a schematic description by using Concept. The ACTgen Macro Builder enhances schematic or synthesis-based capture methods by creating macro functions that are customized for your application. You specify the macro type, bit width, control signals, and so on for your macro, and the Macro Builder does the rest—automatically. Custom, complex functions can be generated in minutes. Then your design files are exported from Concept directly into the Designer Advantage environment, which runs under the X-Window System on the Sun Workstation®.

After the files are imported into the Designer Advantage system, the schematic design can be combined with VHDL descriptions by using ACTmap VHDL. PALs®, Boolean equation descriptions, or state machines can be merged into the design. ACTmap can be used both to synthesize the VHDL into an Actel FPGA-specific description and to optimize pieces of the design for better utilization or performance. A specific Actel device is selected, and the Design Validator then verifies design rule compliance by completing an electrical rules check and providing statistical information such as utilization percentage and average fanout. After validation, the automatic place and route software implements the engineer's design as an FPGA programming file. After automatic placement and routing, the design can be modified by using the incremental place and route tool to make localized changes to your design. After placement and routing, the Timer, a static timing analysis tool, verifies circuit timing and delays. Alternatively, postroute simulations can be done with a backannotated delay file to the Cadence Verilog simulator. Once the design is complete, the Activator programmer programs the proper antifuses to configure the FPGA. The Actionprobe diagnostic hardware and software allow you to explore the operation of a programmed FPGA by providing observability of internal nodes while the FPGA is in the target system.

Software Requirements

Sun Workstation

- Sun OS version 4.1.3 (or later)
- Sun OpenWindows version 2.0 (or later)
- Cadence Logic Workbench™ version 1.6, 1.7, 1.8 (or later)
- Cadence Verilog version 1.7, 1.8, 1.9, 2.0 (or later)

Hardware Requirements

Sun Workstation

- Sun SPARC or SPARC2
- 32 MB RAM (minimum)
- 60 MB hard disk space (minimum)
- CD-ROM drive

Programmers for Actel FPGAs

- Activator 2S programmer—single device programmer requires at least one programming adapter (see Table 1)
- Activator 2 programmer—programs up to four devices simultaneously and requires at least on programming adapter
- Data I/O Unisite and 3900 series programmers

Actionprobe Diagnostic Tools

Actionprobe diagnostic tools provide 100 percent real-time observability of internal nodes while the FPGA is running in the target system. Any node in your design can be probed while your FPGA is operating. This observability is a unique feature that reduces the time required for design verification and debugging.

Designer Options

Synopsys Libraries

The Synopsys Libraries support synthesis with Synopsys's Design Compiler and FPGA Compiler. Synopsys versions 3.1 and 3.2 are supported. The synthesis library is vastly improved, offering substantial performance benefits over previous versions. DesignWare is fully supported—optimized macros are automatically inferred for addition, subtraction, and magnitude comparison operators. You can also instantiate any of the high-performance ACTgen Macro Builder generated macros. A simulation library for Synopsys's VHDL System Simulator is also included.

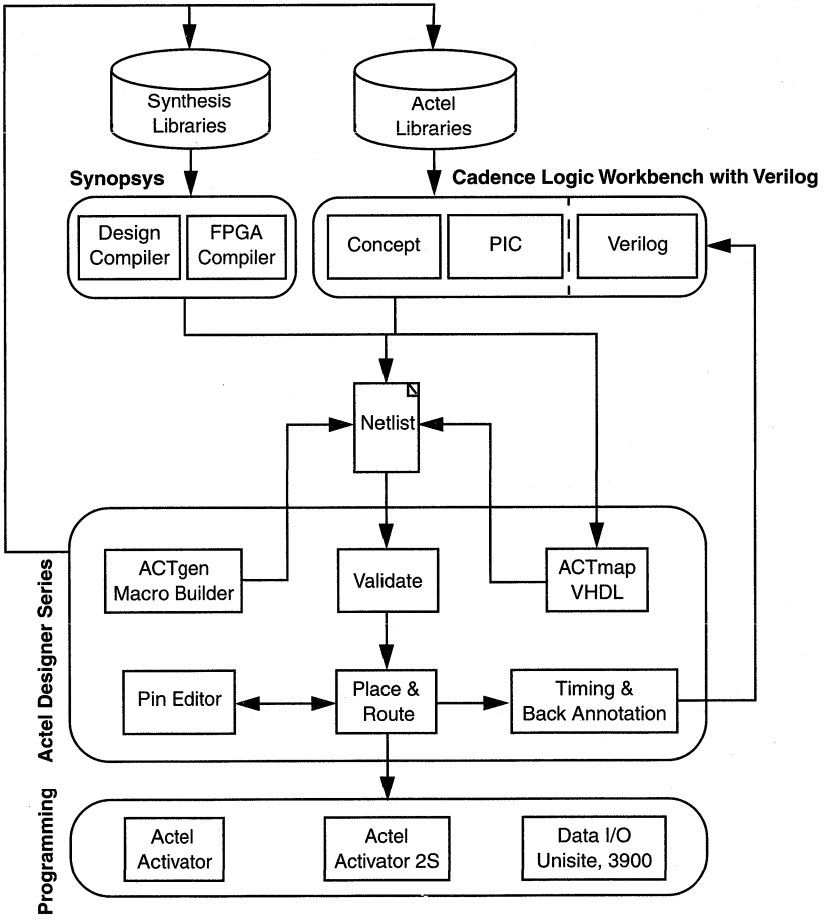


Figure 1 • Flow Diagram

Table 1 • Activator Programming Adapters

ACT 1 Programming Adapters	Package	ACT 2/1200XL Programming Adapters	Package	ACT 3 Programming Adapters	Package
ALS-280	100-PQFP	ALS-286	132-CPGA	MA3-PG100	100-CPGA
ALS-281	44-PLCC	ALS-287	176-CPGA	MA3-PG133	133-CPGA
ALS-282	68-PLCC	ALS-289	100-CPGA	MA3-PG175	175-CPGA
ALS-283	84-PLCC	ALS-290	100-PQFP	MA3-PG207	207-CPGA
ALS-284	84-CPGA	ALS-292	144-PQFP	MA3-PG257	257-CPGA
ALS-285	84-CQFP	ALS-293	160-PQFP	MA3-PL84	84-PLCC
MA1-VQ80	80-VQFP	ALS-294	172-CQFP	MA3-QF100	100-PQFP
		MA2-PL84	84-PLCC	MA3-QF160	160-PQFP
		MA2-TQ176	176-TQFP	MA3-QF208	208-PQFP
		MA2-VQ100	100-VQFP	MA3-QF208	208-RQFP
				MA3-TQ176	176-TQFP
				MA3-VQ100	100-VQFP
				MA3-BG225	225-BGA
				MA3-BG313	313-BGA

Table 2 • Designer Advantage System Selector Guide

System Part Number	Actel FPGA Design Software	Programming	Actionprobe Diagnostics
DA-SN-CVB	≤ 10,000 gates	Activator 2 or 2S	Optional

Designer Advantage System with Cadence Concept/RapidSIM Design Kit

Development System Capabilities

Actel's Designer Advantage™ system for the Cadence™ design environment is a low-cost field programmable gate array (FPGA) design, programming, and verification software system for all Actel FPGA families, including the 10K gate A14100. The system consists of Actel's design software and libraries for the Cadence Concept environment, which contains macro libraries and simulation models for all three families of Actel FPGAs. Included in the design software is ACTmap™ VHDL Synthesis, which will synthesize VHDL for Actel FPGAs, and the ACTgen Macro Builder, which creates complex logic functions according to parameters you select. Available options are the Activator® 2 and Activator 2S programmers and Actionprobe® diagnostic tools.

Concept Design Flow with RapidSIM™

Figure 1 shows how the Designer Advantage software works with Cadence's Concept schematic-capture and RapidSIM simulation tools. You create a schematic description by using Concept. The ACTgen Macro Builder enhances schematic or synthesis-based capture methods by creating macro functions that are customized for your application. You specify the macro type, bit width, control signals, and so on for your macro, and the Macro Builder does the rest— automatically. Custom, complex functions can be generated in minutes. Then your design files are exported from Concept directly into the Designer Advantage environment, which runs under the X-Window System on the Sun Workstation®.

After the files are imported into the Designer Advantage system, the schematic design can be combined with VHDL descriptions by using ACTmap VHDL. PALs®, Boolean equation descriptions, or state machines can be merged into the design. ACTmap can be used both to synthesize the VHDL into an Actel FPGA-specific description and to optimize pieces of the design for better utilization or performance. A specific Actel device is selected, and the Design Validator then verifies design rule compliance by completing an electrical rules check and providing statistical information such as utilization percentage and average fanout. After validation, the automatic place and route software implements the engineer's design as an FPGA programming file. After automatic placement and routing, the design can be modified by using the incremental place and route tool to make localized changes to your design. After placement and routing, the Timer, a static timing analysis tool, verifies circuit timing and delays. Alternatively, postroute simulations can be done with a backannotated delay file to the Cadence RapidSIM simulator. Once the design is complete, the Activator programmer programs the proper antifuses to configure the FPGA. The Actionprobe diagnostic hardware and software allow you to explore the operation of a programmed FPGA by providing observability of internal nodes while the FPGA is in the target system.

Software Requirements

Sun Workstation

- Sun OS version 4.1.3 (or later)
- Sun OpenWindows version 2.0 (or later)
- Cadence Logic Workbench™ version 1.6, 1.7, 1.8 (or later)
- Cadence RapidSIM version 1.0 (or later)

Hardware Requirements

Sun Workstation

- Sun SPARC or SPARC2
- 32 MB RAM (minimum)
- 60 MB hard disk space (minimum)
- CD-ROM drive

Programmers for Actel FPGAs

- Activator 2S programmer—single device programmer requires at least one programming adapter (see Table 1)
- Activator 2 programmer—programs up to four devices simultaneously and requires at least one programming adapter
- Data I/Os Unisite and 3900 series programmers

Actionprobe Diagnostic Tools

Actionprobe diagnostic tools provide 100 percent real-time observability of internal nodes while the FPGA is running in the target system. Any node in your design can be probed while your FPGA is operating. This observability is a unique feature that reduces the time required for design verification and debugging.

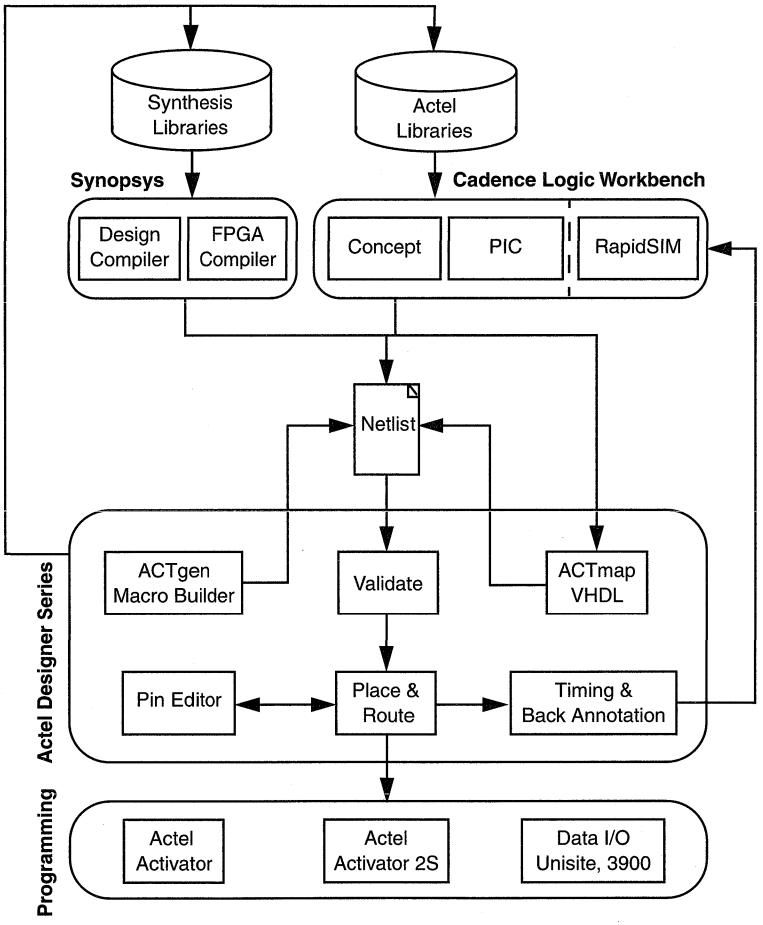


Figure 1 • Flow Diagram

Table 1 • Activator Programming Adapters

ACT 1 Programming Adapters	Package	ACT 2/1200XL Programming Adapters	Package	ACT 3 Programming Adapters	Package
ALS-280	100-PQFP	ALS-286	132-CPGA	MA3-PG100	100-CPGA
ALS-281	44-PLCC	ALS-287	176-CPGA	MA3-PG133	133-CPGA
ALS-282	68-PLCC	ALS-289	100-CPGA	MA3-PG175	175-CPGA
ALS-283	84-PLCC	ALS-290	100-PQFP	MA3-PG207	207-CPGA
ALS-284	84-CPGA	ALS-292	144-PQFP	MA3-PG257	257-CPGA
ALS-285	84-CQFP	ALS-293	160-PQFP	MA3-PL84	84-PLCC
MA1-VQ80	80-VQFP	ALS-294	172-CQFP	MA3-QF100	100-PQFP
		MA2-PL84	84-PLCC	MA3-QF160	160-PQFP
		MA2-TQ176	176-TQFP	MA3-QF208	208-PQFP
		MA2-VQ100	100-VQFP	MA3-QF208	208-RQFP
				MA3-TQ176	176-TQFP
				MA3-VQ100	100-VQFP
				MA3-BG225	225-BGA
				MA3-BG313	313-BGA

Table 2 • Designer Advantage System Selector Guide

System Part Number	Actel FPGA Design Software	Programming	Actionprobe Diagnostics
DA-SN-CRB	≤ 10,000 gates	Activator 2 or 2S	Optional

Designer Advantage System with Mentor Graphics Design Kit

Development System Capabilities

Actel's Designer Advantage™ system for the Mentor Graphics® design environment is a low-cost field programmable gate array (FPGA) design, programming, and verification software system for all Actel FPGA families, including the 10K gate A14100. The system consists of Actel's design software and libraries for the Mentor Graphics environment, which contains macro libraries and simulation models for all three families of Actel FPGAs. Included in the design software is ACTmap™ VHDL Synthesis, which will synthesize VHDL for Actel FPGAs, and the ACTgen Macro Builder, which creates complex logic functions according to parameters you select. Available options are the Activator® 2 and Activator 2S programmers and Actionprobe® diagnostic tools.

Mentor Graphics Design Flow

Figure 1 shows how the Designer Advantage software works with Mentor Graphics's Design Architect schematic-capture and QuickSim II™ simulation tools. The ACTgen Macro Builder enhances schematic or synthesis-based capture methods by creating macro functions that are customized for your application. You specify the macro type, bit width, control signals, and so on for your macro, and the Macro Builder does the rest—automatically. Custom, complex functions can be generated in minutes. Then your design files are exported from Design Architect directly into the Designer Advantage environment, which runs under the X-Window System on the Sun and HP700 workstations.

After the files are imported into the Designer Advantage system, the schematic design can be combined with VHDL descriptions by using ACTmap VHDL. PALs®, Boolean equation descriptions, or state machines can be merged into the design. ACTmap can be used both to synthesize the VHDL into an Actel FPGA-specific description and to optimize pieces of the design for better utilization or performance. A specific Actel device is selected, and the Design Validator then verifies design rule compliance by completing an electrical rules check and providing statistical information such as utilization percentage and average fanout. After validation, the automatic place and route software implements the engineer's design as an FPGA programming file. After automatic placement and routing, the design can be modified by using the incremental place and route tool to make localized changes to your design. After placement and routing, the Timer, a static timing analysis tool, verifies circuit timing and delays. Alternatively, postroute simulations can be done with a backannotated delay file to the Mentor Graphics QuickSim II simulator. Once the design is complete, the Activator programmer programs the proper antifuses to configure the FPGA. The Actionprobe diagnostic hardware and software allow you to explore the operation of a programmed FPGA by providing observability of internal nodes while the FPGA is in the target system.

Software Requirements

Sun Workstation

- Sun OS 4.1.3 (or later)
- Sun OpenWindows version 2.0 (or later)
- Mentor Graphics version 8.2.5

HP700 Workstation

- HPUX version 9.0.1 (or later)
- Mentor Graphics version 8.2.5

Hardware Requirements

Sun Workstation

- Sun SPARC or SPARC2
- 32 MB RAM (minimum)
- 60 MB hard disk space (minimum)
- CD-ROM drive

HP700 Workstation

- HP700 series workstation
- 32 MB RAM (minimum)
- 60 MB hard disk space (minimum)
- CD-ROM drive

Programmers for Actel FPGAs

- Activator 2S programmer—single device programmer requires at least one programming adapter (see Table 1)
- Activator 2 programmer—programs up to four devices simultaneously and requires at least one programming adapter
- Data I/Os Unisite and 3900 series programmers

Actionprobe Diagnostic Tools

Actionprobe diagnostic tools provide 100 percent real-time observability of internal nodes while the FPGA is running in the target system. Any node in your design can be probed while your FPGA is operating. This observability is a unique feature that reduces the time required for design verification and debugging.

Designer Options

Synopsys Libraries

The Synopsys Libraries support synthesis with Synopsys's Design Compiler and FPGA Compiler. Synopsys versions 3.1 and 3.2 are supported. The synthesis library is vastly improved, offering substantial performance benefits over previous versions. DesignWare is fully supported—optimized macros are automatically inferred for addition, subtraction, and magnitude comparison operators. You can also instantiate any of the high-performance ACTgen Macro Builder generated macros. A simulation library for Synopsys's VHDL System Simulator is also included.

Verilog Libraries

The Verilog Libraries option allows you to simulate an Actel FPGA, created with Design Architect, by using the Verilog simulator. Both functional and backannotated timing simulations are supported.

Table 1 • Activator Programming Adapters

ACT 1 Programming Adapters	Package	ACT 2/1200XL Programming Adapters	Package	ACT 3 Programming Adapters	Package
ALS-280	100-PQFP	ALS-286	132-CPGA	MA3-PG100	100-CPGA
ALS-281	44-PLCC	ALS-287	176-CPGA	MA3-PG133	133-CPGA
ALS-282	68-PLCC	ALS-289	100-CPGA	MA3-PG175	175-CPGA
ALS-283	84-PLCC	ALS-290	100-PQFP	MA3-PG207	207-CPGA
ALS-284	84-CPGA	ALS-292	144-PQFP	MA3-PG257	257-CPGA
ALS-285	84-CQFP	ALS-293	160-PQFP	MA3-PL84	84-PLCC
MA1-VQ80	80-VQFP	ALS-294	172-CQFP	MA3-QF100	100-PQFP
		MA2-PL84	84-PLCC	MA3-QF160	160-PQFP
		MA2-TQ176	176-TQFP	MA3-QF208	208-PQFP
		MA2-VQ100	100-VQFP	MA3-QF208	208-RQFP
				MA3-TQ176	176-TQFP
				MA3-VQ100	100-VQFP
				MA3-BG225	225-BGA
				MA3-BG313	313-BGA

Table 2 • Designer Advantage System Selector Guide

System Part Number	Actel FPGA Design Software	Programming	Actionprobe Diagnostics
DA-SN-MGB	≤ 10,000 gates	Activator 2 or 2S	Optional
DA-HP7-MGB	≤ 10,000 gates	Activator 2 or 2S	Optional

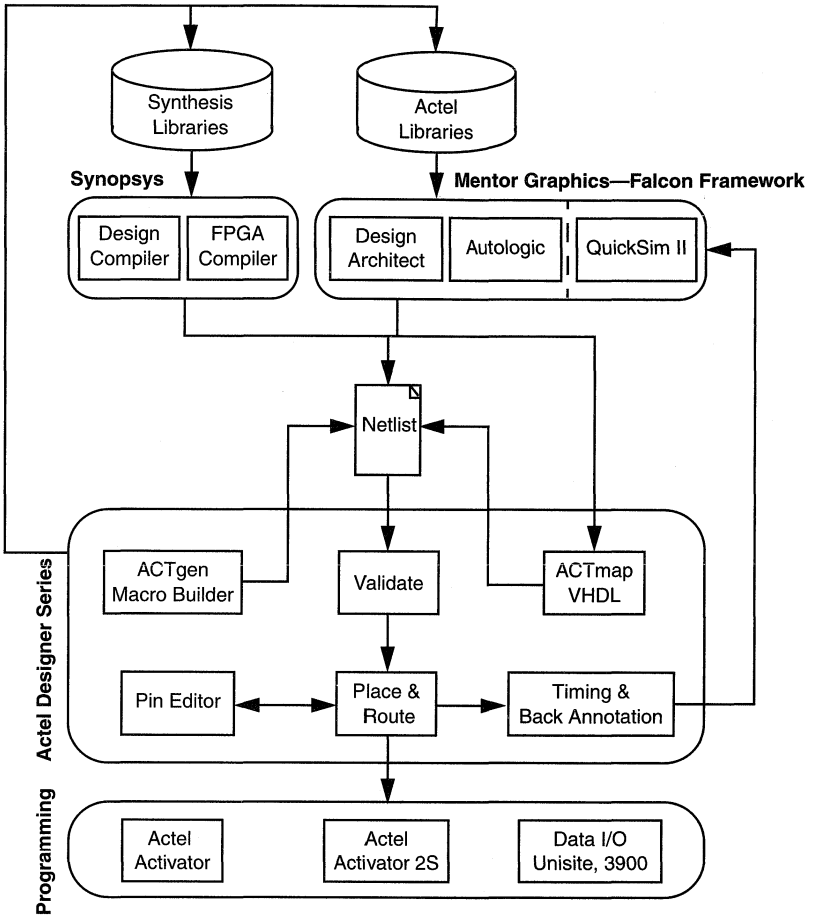


Figure 1 • Flow Diagram

Designer and Designer Advantage System with OrCAD Design Kit

Development System Capabilities

Actel's Designer and Designer Advantage™ system for the OrCAD™ design environment is a low-cost field programmable gate array (FPGA) design, programming, and verification software system for all Actel FPGA families, including the 10K gate A14100. The system consists of Actel's design software and libraries for the OrCAD environment, which contains macro libraries and simulation models for all three families of Actel FPGAs. Included in the design software is ACTmap™ VHDL Synthesis, which will synthesize VHDL for Actel FPGAs, and the ACTgen Macro Builder, which creates complex logic functions according to parameters you select. Available options are the Activator® 2 and Activator 2S programmers and Actionprobe® diagnostic tools.

Schematic Design Tools 386+ Design Flow

Figure 1 shows how the Designer Advantage software works with OrCAD's Schematic Design Tools 386+ schematic-capture and Design Simulation Tools 386+ simulation tools. You create a schematic description by using Schematic Design Tools 386+. The ACTgen Macro Builder enhances schematic or synthesis-based capture methods by creating macro functions that are customized for your application. You specify the macro type, bit width, control signals, and so on for your macro, and the Macro Builder does the rest—automatically. Custom, complex functions can be generated in minutes. Then your design files are exported from OrCAD directly into the Designer or Designer Advantage environment, which runs under Microsoft Windows™ on the PC.

After the files are imported into the Designer Advantage system, the schematic design can be combined with VHDL descriptions by using ACTmap VHDL. PALs®, Boolean equation descriptions, or state machines can be merged into the design. ACTmap can be used both to synthesize the VHDL into an Actel FPGA-specific description and to optimize pieces of the design for better utilization or performance. A specific Actel device is selected, and the Design Validator then verifies design rule compliance by completing an electrical rules check and providing statistical information such as utilization percentage and average fanout. After validation, the automatic place and route software implements the engineer's design as an FPGA programming file. After automatic placement and routing, the design can be modified by using the incremental place and route tool to make localized changes to your design. After placement and routing, the Timer, a static timing analysis tool, verifies circuit timing and delays. Alternatively, postroute simulations can be done with a backannotated delay file to the OrCAD Design Simulation Tools 386+ simulator. Once the design is complete, the Activator programmer programs the proper antifuses to configure the FPGA. The Actionprobe diagnostic hardware and software allow you to explore the operation of a programmed FPGA by providing observability of internal nodes while the FPGA is in the target system.

Software Requirements

486- or Pentium-based PC

- MS-DOS™ version 3.3 (or later)
- Windows version 3.1 (or later)—required
- OrCAD Schematic Design Tools 386+ version 1.1 (or later)
- OrCAD Design Simulation Tools 386+ version 1.1 (or later)

Hardware Requirements

486- or Pentium-based PC

- 24 MB RAM (minimum)
- 40 MB virtual disk
- additional 50 MB disk space for program and data files
- 1.44 MB floppy drive
- CD-ROM drive (recommended)
- VGA, EGA, or monochrome graphics card
- Microsoft compatible mouse

Programmers for Actel FPGAs

- Activator 2S programmer—single device programmer requires at least one programming adapter (see Table 1)
- Activator 2 programmer—programs up to four devices simultaneously and requires at least one programming adapter
- Data I/Os Unisite and 3900 series programmers

Actionprobe Diagnostic Tools

Actionprobe diagnostic tools provide 100 percent real-time observability of internal nodes while the FPGA is running in the target system. Any node in your design can be probed while your FPGA is operating. This observability is a unique feature that reduces the time required for design verification and debugging.

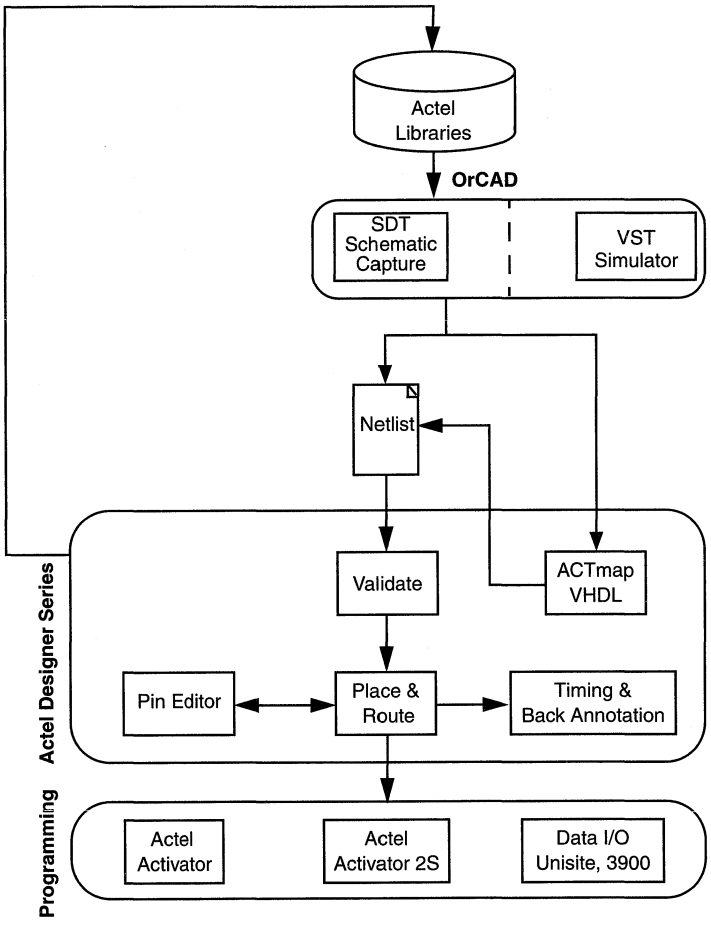


Figure 1 • Flow Diagram

Table 1 • Activator Programming Adapters

ACT 1 Programming Adapters	Package	ACT 2/1200XL Programming Adapters	Package	ACT 3 Programming Adapters	Package
ALS-280	100-PQFP	ALS-286	132-CPGA	MA3-PG100	100-CPGA
ALS-281	44-PLCC	ALS-287	176-CPGA	MA3-PG133	133-CPGA
ALS-282	68-PLCC	ALS-289	100-CPGA	MA3-PG175	175-CPGA
ALS-283	84-PLCC	ALS-290	100-PQFP	MA3-PG207	207-CPGA
ALS-284	84-CPGA	ALS-292	144-PQFP	MA3-PG257	257-CPGA
ALS-285	84-CQFP	ALS-293	160-PQFP	MA3-PL84	84-PLCC
MA1-VQ80	80-VQFP	ALS-294	172-CQFP	MA3-QF100	100-PQFP
		MA2-PL84	84-PLCC	MA3-QF160	160-PQFP
		MA2-TQ176	176-TQFP	MA3-QF208	208-PQFP
		MA2-VQ100	100-VQFP	MA3-QF208	208-RQFP
				MA3-TQ176	176-TQFP
				MA3-VQ100	100-VQFP
				MA3-BG225	225-BGA
				MA3-BG313	313-BGA

Table 2 • Designer and Designer Advantage System Selector Guide

System Part Number	Actel FPGA Design Software	Programming	Actionprobe Diagnostics
DS-PC-OR	≤ 2,500 gates	Activator 2 or 2S	Optional
DA-PC-OR	≤ 10,000 gates	Activator 2 or 2S	Optional

Designer and Designer Advantage System with Viewlogic Design Kit

Development System Capabilities

Actel's Designer and Designer Advantage™ system for the Viewlogic® design environment is a low-cost field programmable gate array (FPGA) design, programming, and verification software system for all Actel FPGA families, including the 10K gate A14100. The system consists of Actel's design software and libraries for the Viewlogic environment, which contains macro libraries and simulation models for all three families of Actel FPGAs. Included in the design software is ACTmap™ VHDL Synthesis, which will synthesize VHDL for Actel FPGAs, and the ACTgen Macro Builder, which creates complex logic functions according to parameters you select. Available options are the Activator® 2 and Activator 2S programmers and Actionprobe® diagnostic tools.

Viewlogic Design Flow

Figure 1 shows how the Designer Advantage software works with Viewlogic's ViewDraw® or PROcapture® schematic-capture and ViewSim® or PROsim® simulation tools. You create a schematic description by using ViewDraw or PROcapture. The ACTgen Macro Builder enhances schematic or synthesis-based capture methods by creating macro functions that are customized for your application. You specify the macro type, bit width, control signals, and so on for your macro, and the Macro Builder does the rest—automatically. Custom, complex functions can be generated in minutes. Then your design files are exported from Viewlogic directly into the Designer or Designer Advantage environment, which runs under Microsoft Windows™ on PCs and under the X-Window System on the Sun and HP700 workstations.

After the files are imported into the Designer Advantage system, the schematic design can be combined with VHDL descriptions by using ACTmap VHDL. PALs®, Boolean equation descriptions, or state machines can be merged into the design. ACTmap can be used both to synthesize the VHDL into an Actel FPGA-specific description and to optimize pieces of the design for better utilization or performance. A specific Actel device is selected, and the Design Validator then verifies design rule compliance by completing an electrical rules check and providing statistical information such as utilization percentage and average fanout. After validation, the automatic place and route software implements the engineer's design as an FPGA programming file. After automatic placement and routing, the design can be modified by using the incremental place and route tool to make localized changes to your design. After placement and routing, the Timer, a static timing analysis tool, verifies circuit timing and delays. Alternatively, postroute simulations can be done with a backannotated delay file to the Viewlogic ViewSim or PROsim simulator. Once the design is complete, the Activator programmer programs the proper antifuses to configure the FPGA. The Actionprobe diagnostic hardware and software allow you to explore the operation of a programmed FPGA by providing observability of internal nodes while the FPGA is in the target system.

Software Requirements

486- or Pentium-based PC

- MS-DOS™ version 3.3 (or later)
- Windows version 3.1 (or later) optional
- Viewlogic Workview ® PLUS or PROSeries 6.0
- Viewlogic Viewsim or PROsim

Sun Workstation

- Sun OS version 4.3 (or later)
- Sun OpenWindows version 2.0 (or later)
- Viewlogic Powerview ® version 5.1 (or later)

HP700 Workstation

- HPUX version 9.0.1 (or later)
- Viewlogic Powerview version 5.1 (or later)

Hardware Requirements

486- or Pentium-based PC

- 24 MB RAM (minimum)
- 40 MB virtual disk
- Additional 50 MB disk space for program and data files
- 1.44 MB floppy disk drive
- CD-ROM drive (recommended)
- VGA, EGA, or monochrome graphics card

Sun Workstation

- Sun SPARC or SPARC2
- 32 MB RAM (minimum)
- 60 MB hard disk space (minimum)
- CD-ROM drive

HP700 Workstation

- HP700 series workstation
- 32 MB RAM (minimum)
- 60 MB hard disk space (minimum)
- CD-ROM drive

Programmers for Actel FPGAs

- Activator 2S programmer—single device programmer requires at least one programming adapter (see Table 1)
- Activator 2 programmer—programs up to four devices simultaneously and requires at least one programming adapter
- Data I/Os Unisite and 3900 series programmers

Actionprobe Diagnostic Tools

Actionprobe diagnostic tools provide 100 percent real-time observability of internal nodes while the FPGA is running in the target system. Any node in your design can be probed while your FPGA is operating. This observability is a unique feature that reduces the time required for design verification and debugging.

Designer Options

Synopsys Libraries

The Synopsys Libraries support synthesis with Synopsys's Design Compiler and FPGA Compiler. Synopsys versions 3.1 and 3.2 are supported. The synthesis library is vastly improved, offering substantial performance benefits over previous versions. DesignWare is fully supported—optimized macros are automatically inferred for addition, subtraction, and magnitude comparison operators. You can also instantiate any of the high-performance ACTgen Macro Builder generated macros. A simulation library for Synopsys's VHDL System Simulator is also included.

Verilog Libraries

The Verilog Libraries option allows you to simulate an Actel FPGA, created with Design Architect, by using the Verilog simulator. Both functional and backannotated timing simulations are supported.

Table 1 • Activator Programming Adapters

ACT 1 Programming Adapters	Package	ACT 2/1200XL Programming Adapters	Package	ACT 3 Programming Adapters	Package
ALS-280	100-PQFP	ALS-286	132-CPGA	MA3-PG100	100-CPGA
ALS-281	44-PLCC	ALS-287	176-CPGA	MA3-PG133	133-CPGA
ALS-282	68-PLCC	ALS-289	100-CPGA	MA3-PG175	175-CPGA
ALS-283	84-PLCC	ALS-290	100-PQFP	MA3-PG207	207-CPGA
ALS-284	84-CPGA	ALS-292	144-PQFP	MA3-PG257	257-CPGA
ALS-285	84-CQFP	ALS-293	160-PQFP	MA3-PL84	84-PLCC
MA1-VQ80	80-VQFP	ALS-294	172-CQFP	MA3-QF100	100-PQFP
		MA2-PL84	84-PLCC	MA3-QF160	160-PQFP
		MA2-TQ176	176-TQFP	MA3-QF208	208-PQFP
		MA2-VQ100	100-VQFP	MA3-QF208	208-RQFP
				MA3-TQ176	176-TQFP
				MA3-VQ100	100-VQFP
				MA3-BG225	225-BGA
				MA3-BG313	313-BGA

Table 2 • Designer and Designer Advantage System Selector Guide

System Part Number	Actel FPGA Design Software	Programming	Actionprobe Diagnostics
DS-PC-VL	≤ 2,500 gates	Activator 2 or 2S	Optional
DA-PC-VL	≤ 10,000 gates	Activator 2 or 2S	Optional
DA-SN-VLB	≤ 10,000 gates	Activator 2 or 2S	Optional
DA-HP7-VLB	≤ 10,000 gates	Activator 2 or 2S	Optional

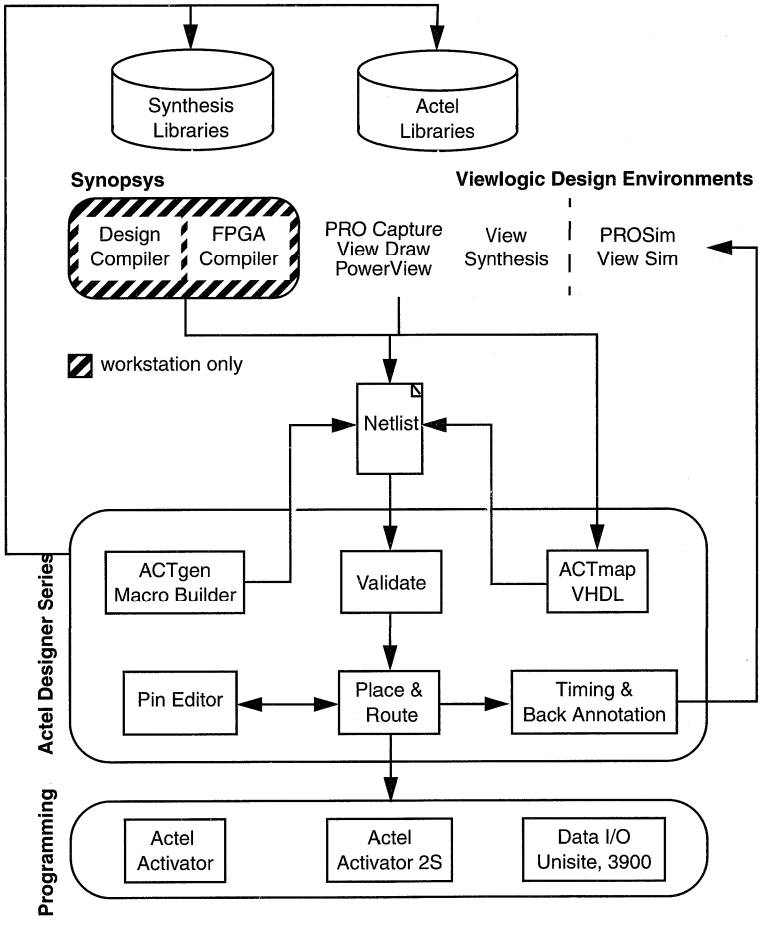


Figure 1 • Flow Diagram

Synopsys Libraries

Features

- Performance-oriented FPGA design kit for Synopsys® version 3.1 or greater high level design tools
- Supports Synopsys® Design Compiler™ and FPGA Compiler
- Provides Actel specific DesignWare™ synthetic libraries to access Actel optimized structured macros such as adders and counters for high performance designs
- Includes simulation models and backannotation utilities SDF for Synopsys' VHDL System Simulator™ (VSS)
- Tailored to take advantage of Actel's fine grained and synthesis friendly architecture
- Works in conjunction with Actel's Designer Series development systems to provide high level FPGA design

Introduction

Actel's Synopsys Libraries provide users the ability to utilize Synopsys High Level Design Automation (HLDA) tools for top down design. These libraries support Design Compiler and FPGA Compiler version 3.1, DesignWare, and Synopsys VHDL System Simulator (VSS). The improvements to the algorithms in the technology libraries results in significant improvements over previously synthesized designs. Designs are described in VHDL or Verilog® HDL and mapped to an Actel netlist using the Synopsys Design Compiler or FPGA Compiler. During design description, DesignWare elements such as Actel specific adders and counters can be instantiated into the design or inferred by the compiler to obtain optimal macro function performance. After describing the design in VHDL, it can be pre-route simulated in the VSS simulator before translated to Actel's Designer Advantage™ system for placement and routing. Place and route delays can be backannotated to the VSS or Verilog simulator to verify design performance before programming the FPGA.

High Performance Synthesis

The version 3.1 technology and DesignWare libraries provide a substantial improvement over previous libraries for performance and utilization in Actel FPGAs. This is accomplished by the combination of new algorithms in the FPGA and Design Compliers tailored for the Actel architectures and the addition of Design Ware libraries supporting optimized structured logic functions. Table 1 shows a comparison of the improved performance of adders, counters, comparators, and datapath oriented designs using the version 3.1 versus version 3.0 libraries.

Product Details

Technology Libraries for FPGA Compiler and Design Compiler version 3.1

The 3.1 Libraries allow designers to describe their design in VHDL or Verilog and translate them into an Actel ACT 1, ACT 2, or ACT 3 FPGAs. Synopsys has implemented new features that take advantage of the Actel architecture. These features include sequential mapping which takes advantage of Actel's multiplexer and flip-flop oriented architecture to reduce logic levels and improved support for the special features of the Actel ACT™ 2 and ACT 3 complex I/O cells.

7

Table 1 • Performance Comparison

Function	Version 3.0	Version 3.1*	Percent Improvement
16 Bit Adder	31.6 MHz	62.5 MHz	98%
24 Bit Adder	26.6 MHz	47.6 MHz	79%
16 Bit Counter	40.3 MHz	90.9 MHz	125%
24 Bit Counter	35.2 MHz	78.1 MHz	122%
16 Bit Comparator	37.3 MHz	55.6 MHz	49%
24 Bit Comparator	31.3 MHz	42.4 MHz	36%
Datapath Intensive	22.1 MHz	26.0 MHz	18%

* Performance figures are obtained using the version 3.1 and DesignWare Libraries.

VHDL System Simulator (VSS)

Actel designers now have the advantages of powerful debugging capability and fast turnaround time to verify their designs using the VSS simulator. Actel has provided a complete gate level VHDL library for all three FPGA families: ACT 1, ACT 2, and ACT 3. This allows the designer to complete pre-route delay estimations and verify post layout timing. Post layout timing verification is supported through a standard delay file (SDF) for backannotation of routing delays. The timing delays provided in the SDF file support multiple simulation scenarios based upon temperature, voltage, and device speed grade. The Actel VSS library models are full timing gate simulated (FTGS) and VITAL compliant. These provide simulation models that allow quick and accurate VHDL simulation.

Note: For Verilog sumulation, Actel provides separate Verilog simulation libraries. These libraries combined with the SDF post layout delays provide a complete Verilog simulation enviroment for Synopsys. See Figure 1 for the Actel Synopsys Verilog design flow.

DesignWare Libraries

DesignWare is a product which increases productivity through design reuse. Actel supports this product by providing a synthetic library for Synopsys DesignWare. The library elements provide commonly used functions such as adders, subtractors, and counters up to 32 bits and comparators with six comparison functions. By taking advantage of the synthetic library elements, the VHDL or Verilog code can be fine tuned for optimal Actel implementation. These libraries are based upon Actel's ACTgen™ parameterized macro generator product which allows the user to achieve performance comparable to schematic drawn macros. These library elements can be instantiated into the design or inferred by the compilers resulting in an extremely efficient design process.

Actel Designer Series System

After describing and verifying the top down design using Synopsys' tools, the netlist is exported into the Designer Series system for completion of the Actel FPGA. The Designer Series systems are high-productivity computer-aided engineering environments used with Actel's family of field programmable gate array devices. User friendly Microsoft® Windows™ and X Window graphical user interfaces allow completion of Actel designs from concept to silicon in hours without costly nonrecurring engineering expenses. The systems consist of placement and routing, timing verification, and programming software.

Placement and routing consists of 100% automatic, incremental, and PinEdit software functions. Fully automatic place and route software minimizes design delay by assigning macros to optimal locations on the chip. The system uses the design's netlist, critical net information, and I/O assignments to automatically place and route all the logic blocks within the circuit. No manual intervention is required, even at high device utilization. Incremental place and route software allows quick iterations of an existing design without changing the entire placement and routing. After the initial automatic placement and routing, the user can select varying levels of incremental strength and relayout the design to include the changes.

Timing verification is accomplished through backannotation of post route delays to VSS or by using the Actel timing analyzer, is an interactive tool that determines and analyzes the performance of all critical and noncritical paths within a specified design. It performs a static timing analysis using the post route delays extracted from the layout.

After final timing analysis is completed, the Designer Series system produces device programming files that can be used by either Data I/O® or optional Actel Activator® series programmers. Existing Data I/O customers can utilize the Unisite or Series 3900 programmers to program all Actel ACT 1, ACT 2, and ACT 3 devices when certified by Actel. Alternatively, Actel offers the Activator 2 and Activator 2S programmers, which program all Actel FPGAs. The Activator 2 programs up to four devices of the same design at a time, while the Activator 2S is a single-device programmer.

Hardware/Software Requirements

On a Sun® SPARC™ or HP700® workstation, the Technology Libraries require:

- Synopsys Design or FPGA complier version 3.1 or greater
- Sun OS 4.1.1 or greater or
- HPUX 9.01
- 64 MB of memory
- Hard disk with 10 MB (synthesis libraries) and 50 MB (VSS simulation libraries) free disk space
- Swap Space—100 MB

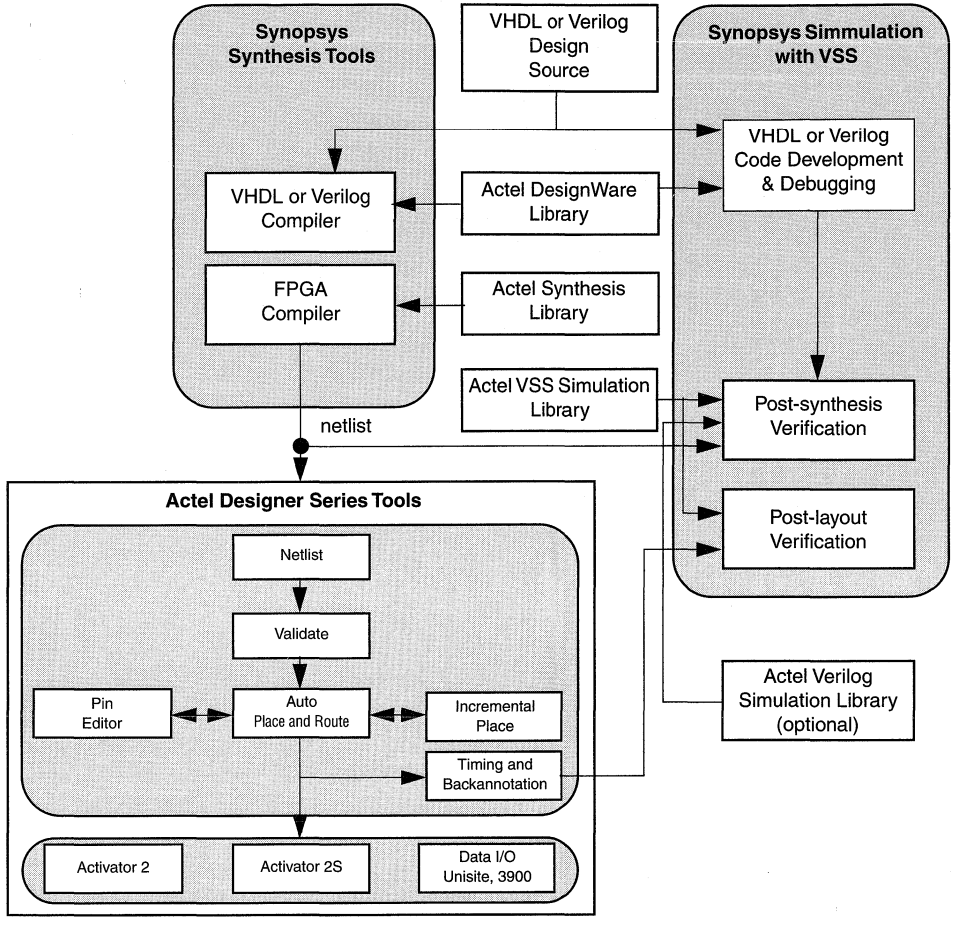


Figure 1 • *Actel Synthesis Design Flow for Synopsys*

Actel's Industry Alliance Program

Purpose

The Actel Industry Alliance program establishes a practical Technology Transparent Design (TTD) environment for Actel field programmable gate array (FPGA) users. Actel and Alliance members cooperate to provide Actel FPGA users complete design support using the EDA system of their choice. Actel design libraries available from Alliance members allow users to describe and simulate Actel designs using their existing EDA tools. After satisfactory design entry and simulation, these designs can be transferred to Actel's Designer Series development system to complete placement and routing, timing analysis, and programming.

Technical Cooperation

Actel provides technical cooperation and support to Alliance members, including:

- Access to Actel's FPGA development environment, the Designer Series, for placement and routing, timing analysis, and programming.
- CAE connection package, which provides the information necessary to connect the CAE system to Actel's development system.
- Standard verification circuits and test vectors for self certification of the design libraries and interfaces by the Alliance member.

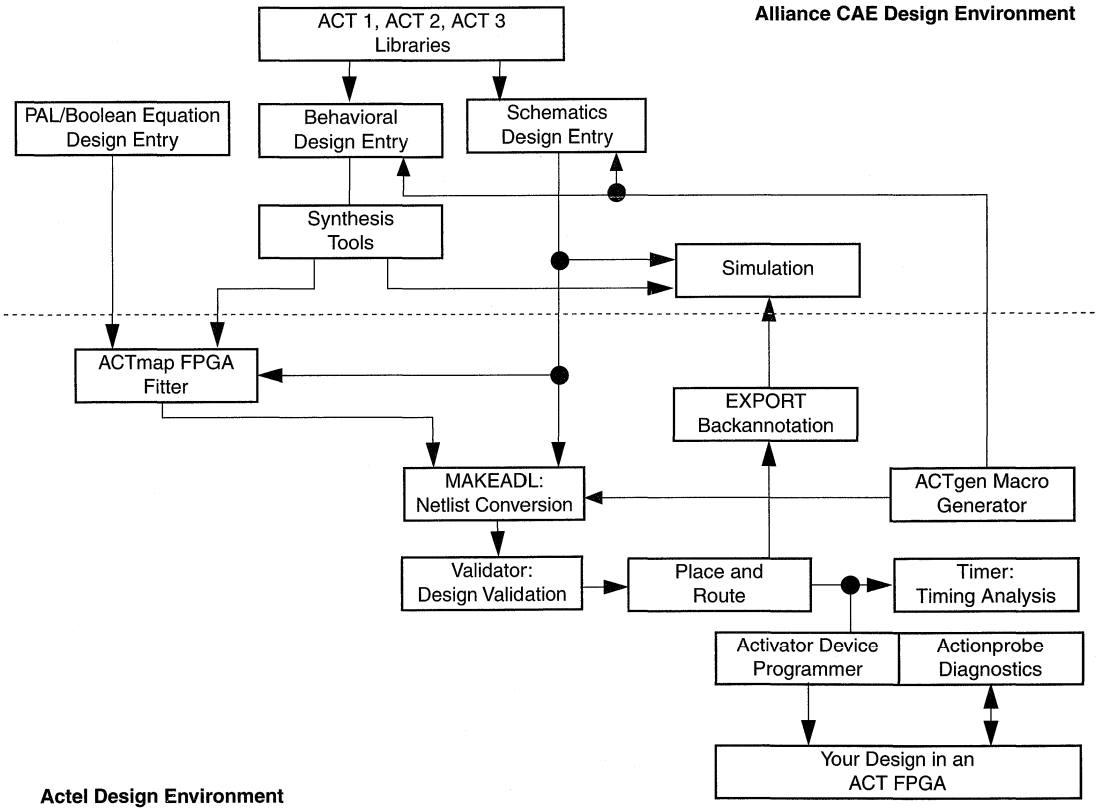


Figure 1 • Shown above are typical ways Alliance member CAE environments interface with the Actel design environment.

Alliance Members

Customers can obtain the certified design libraries and interfaces to Actel's development system and FPGA devices by contacting their respective CAE vendor. CAE Alliance members and contacts are listed below.

Current Alliance members

Company Name	Contact Name	Address	Telephone
Aldec	Stanley Hyduke	3525 Old Conejo Road, Suite 111 Newbury Park, CA 91320	(805) 499-6867
Cadence Design System	Dai Tran	2655 Seely Road, Bldg. 6 San Jose, CA 95134	(408) 428-5257
Compass Design Automation	Mahendra Jain	1865 Lundy Avenue San Jose, CA 95131	(408) 434-7950
Data I/O	Linda Burgess	10525 Willows Road N.E. Redmond, WA 98073-9746	(206) 881-6444
Exemplar	Bob Barker	2550 Ninth Street, Suite 102 Berkeley, CA 94710	(510) 849-0937
Intergraph Company	Vincent Mazur	6101 Lookout Road, Suite A Boulder, CO 80301	(303) 581-2301
Logic Modeling Corporation	Laura Horsey	19500 N.W. Gibbs Drive Beaverton, OR 97075	(503) 531-2271
Mentor Graphics	Sam Picken	8005 SW Boeckman Road Wilsonville, OR 97070-7777	(503) 685-1298
Minc	Bill Schulze	6755 Earl Drive Colorado Springs, CO 80918	(719) 590-1155
OrCAD	Troy Scott	9300 SW Nimbus Beaverton, OR 97005	(503) 671-9500
Synopsys	Lynn Fiance	700 East Middlefield Road Mountain View, CA 94043-4033	(415) 694-4289
Veda Design Automation (formerly GenRad DAP)	Rastgow Shale	2041 Mission College Blvd., Suite 259 Santa Clara, CA 95054	(408) 496-4518
Viewlogic	Ravi Ravikumar	47211 Lakeview Blvd. Fremont, CA 94538	(510) 659-4001
Zuken Redac	Dwight Dagenais	3945 Freedom Circle, Suite 1100 Santa Clara, CA 95054	(408) 562-0177

Activator® 2 and Activator 2S Programmers

Features

- Supports ACT™ 1, ACT 2, and ACT 3 device families and packages
- Versions available for 386™ PC, 486™ PC, Pentium, Sun™, and HP® workstations
- Supports functional verification with Actionprobe® diagnostics
- Activator 2 simultaneously programs up to four identical devices
- Activator 2S is a low-cost, single-site version of the Activator 2
- Supports checksum and design name verification

Product Description

Activator 2 and Activator 2S are Actel's state-of-the-art desktop programmers. They utilize Actel's Designer and Designer Advantage™ system software and PLICE® antifuse technology to program Actel's ACT 1, ACT 2, and ACT 3 field programmable gate arrays (FPGAs). Customized programming adapters for each device type allow different packages to be programmed by switching adapters. Programming, verification, and debugging are executed on these programmers. The optional Actionprobe diagnostic hardware and software support observation of all internal signals. Individual versions of these programmers are available to support industry standard platforms such as 386 PC, 486 PC, Pentium, Sun, and HP workstations.

The programmers are software-driven, providing flexibility of application and a built-in barrier to obsolescence. Independently powered, the units provide desktop device programming, functional testing, and in-circuit debugging. By using a SCSI interface, the programmers support different hardware platforms.

For easy device identification and verification, the blank check (blankchk) command in the programming software allows users to read back checksums and silicon signatures after programming. The checksum data is automatically generated by the Designer software, and the silicon signature is defined by the user during fuse file generation.

Activator 2 Base Unit

The base unit contains the control board and the analog board. The LED display on the top of the programmer shows when the unit receives power. Four adapter ports located on the top of the base unit accept different programming adapters for each device type. The adapter ports are identical, which allows programming adapters to be interchanged. SCSI connectors are located on the back of the unit.

Activator 2S Base Unit

Activator 2S is a low-cost, single-site version of the Activator 2 and supports all the Activator 2 features. A single adapter port is located on the top of the base unit; it accepts different programming adapters for each device type. The SCSI connectors are located on the back of the unit.

The Programming Adapters

There are unique programming adapters available to support each package type within a product family. The user only needs to switch programming adapters to program a different device type or package. For the Activator 2, one to four ports may be used simultaneously when programming a design. Any adapter may be used on any available port.

The Diagnostic Pod

The diagnostic pod supports Actionprobe diagnostics and connects to the programmer by a cable. The Activator 2 or Activator 2S permits the user to view any internal circuit activity in an Actel FPGA mounted on the user's PC board through Actel's on-chip diagnostic pins. These pins are connected to the programmer by the diagnostic pod; by using the Actionprobe diagnostics software to specify a circuit point, each pod can access two signals simultaneously. A single pod may be connected to the Activator 2S or Activator 2. A six-foot cord connects the pod to the programmer and provides diagnostic flexibility.

Synthesis

Section 8: Synthesis

High-Level FPGA Design in the Synopsys Environment

Summary

Many system designers have abandoned traditional design methodologies in favor of a new high-level, top-down approach. Instead of designing a system chip by chip by capturing logic gates with schematic libraries, designers can describe the entire system at the behavioral level. This method of system design capture goes beyond functional specification. It can also include descriptions of external drive requirements, internal clock rates, electrical loading, and required signal arrival times. Today's new design methodology consists of mapping an HDL functional description into a technology library via synthesis.

What follows describes how to synthesize an HDL design description into an Actel FPGA using the Synopsys Design Compiler or FPGA Compiler. Each compiler optimizes and maps the HDL behavioral description into an Actel ACT™ 1, ACT 2, or ACT 3 device family. After satisfactory optimization, the compiler can produce an EDIF 2 0 0 file containing Actel components. The Actel EDIF netlist program, cae2adl, converts the EDIF file to an Actel adl netlist. After successful adl generation, the Actel Designer Series software places and routes the design. After place and route, the Actel Timer, a static timing analysis tool, verifies the design's critical paths. Post route delays can be backannotated to the Synopsys Timing Analysis tool, Design Time, using the program sdfgen. Furthermore, VHDL users can backannotate delays to the VHDL System Simulator (VSS). Backannotation of actual routing delays to various other simulators is also possible but not covered here. Figure 1 describes the Actel/Synopsys interface design flow.

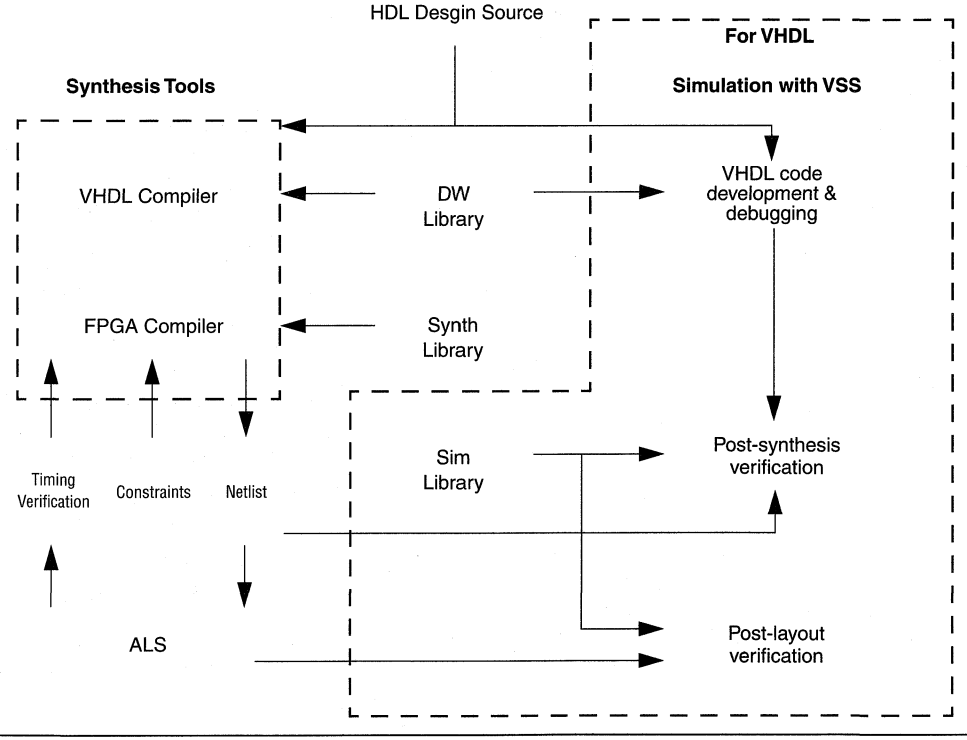


Figure 1 • Actel/Synopsys Design Flow

8

Software Requirements

This design flow requires the following software tools from Synopsys and Actel to generate an Actel FPGA:

- Synopsys VHDL Compiler or HDL Compiler for Verilog. Depending on the design, one of these tools is required to compile VHDL or Verilog into internal Synopsys database format.
- Synopsys Design Compiler or FPGA Compiler. One of these tools is necessary to optimize the input from either the VHDL or HDL compiler. Both tools provide the same basic optimization and synthesis functions.
- Actel/Synopsys Synthesis and VSS libraries for ACT 1, ACT 2, and ACT 3 device families. Actel/Synopsys DesignWare libraries for ACT 2 and ACT 3 device families.
- Designer Advantage, Actel's FPGA software, accepts EDIF input from Synopsys. The software lays out the design into an Actel FPGA and generates a fuse file for programming a device.

Design Flow

The HDL Description

The design process begins with an HDL description of the design. Partitioning the design into several blocks of HDL and optimizing them individually achieves the best results. Each block of the design should have its own timing constraints and attributes. A top level of hierarchy contains instantiations of each partition and each I/O and clock buffer. Input, output, and clock buffers need not be instantiated at the top level. The command "insert_pads" will automatically insert I/O and clock buffers in the design.

Compile Design in Synopsys

The Synopsys compiler reads the HDL description(s) of each design. Figure 2 represents the steps that are performed in the Synopsys environment. As mentioned previously, you should set constraints and attributes individually for each partition. Some constraints and attributes to consider are max_delay, clock_period, max_fanout, and set_arrival. Compile each block and verify the critical paths with the Synopsys Timing Analysis Tool. The delays in the Actel Library are based on the estimated post-route delays listed in the *Actel FPGA Data Book and Design Guide*. These timing checks are estimates only. The actual delay information is derived from place and route with ALS. Hence, you should verify the HDL code only with the Synopsys timing analysis tool. After each block is compiled, issue a dont_touch on each partition. The dont_touch will stop the compiler from optimizing across hierarchical boundaries. Nevertheless, the top level must be compiled before an EDIF file can be written. As mentioned previously, the Synopsys v3.0 or later "insert_pads" command will insert I/O and clock buffers automatically into the top level. This command is issued prior to optimizing the top-level HDL. If the version of Synopsys is not v3.0 or later, I/Os and clocks must be manually instantiated in the HDL code

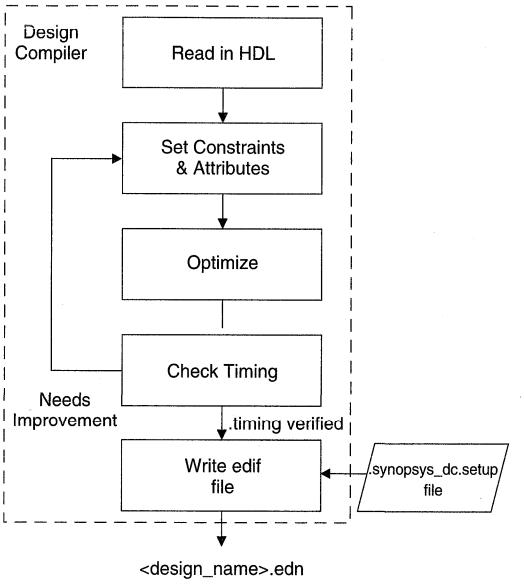


Figure 2 • The Synopsys Design Compiler

CAUTION:

The Actel libraries have fanout limits attached to each cell. However, the Synopsys Design Compiler does not maintain these limits across hierarchical boundaries. Hence, be aware of how each partition interacts with other blocks. For example, a signal in partition A may be confined to the ACT 2 fanout limit of 16, but it may also be connected to several more macros outside of its partition. The ALS software will produce an error message if the fanout exceeds the limit of 24.

The compiler writes an EDIF file for Actel translation once the entire design is compiled. Because EDIF has many "flavors," Actel requires specific switches in the Synopsys environment to produce Actel-compatible EDIF. The following EDIF switches should be placed in the .synopsys_dc.setup file.

```
edifout_netlist_only = true
edifout_no_array = true
edifout_power_amd _ground_representation =
cell
edifout_ground_name = GND
edifout_power_name = VCC
edifout_ground_pin_name = Y
edifout_power_pin_name = Y
edifout_pertty_print = true
bus_naming_style = "%s<%d>"
```

EDIF to Actel Translation

Synopsys produces an EDIF file named <design_name>.edn. The Actel program, cae2adl, included in the Actel Design Kit, *Designer Advantage*, converts the Synopsys EDIF into an Actel ADL netlist. The command is invoked from the command line in the directory where the EDIF file resides. The syntax for invoking cae2adl is

```
cae2adl -edn2adl fam:act2 ednin:FPGA.edn
NObadOrigName:T FPGA
```

In our example, cae2adl -end2adl is the netlist conversion command.

Our example defines the family option as act2. Acceptable values for the family option correspond to the available Actel library families: ACT 1, ACT 2, and ACT 3.

We used FPGA.edn as the conversion file name in our example. You would enter your design name in place of FPGA.

The NObadOrigName:T variable ensures that the EDIF file name is used by the Actel suite of programs if the original name is not completely legal.

In this example, we defined the name of the design as FPGA. You must enter the same design name you used in the "write" command. Failure to use the correct name will cause an error in the Designer Series Pin Editor and Validator programs.

When you successfully convert the EDIF file, you will create five new files in the design_name directory: design_name.adl, design_name.crt, design_name.def, design_name.ipf, and design_name.als.

Layout of an Actel Device

Once an Actel adl netlist is generated, the design is placed and routed into an Actel ACT 1, ACT 2, or ACT 3 device via the Actel Designer Series tools. For more information regarding these tools, refer to individual Actel data sheets, application notes, and User Guides.

Using ACTgen Macros with Synopsys

Hitesh Patel

Customer Applications, Actel Corporation

The ACTgen™ Macro Builder is an Actel software tool used to create macros that can be instantiated in Verilog or VHDL designs for synthesis using Synopsys. These macros are defined in a high-level language from which you can generate an EDIF netlist. The goal of the ACTgen Macro Builder is the creation of flexible library elements that effectively use the Actel architecture to achieve optimum performance and to improve designer productivity by minimizing module count.

Becuase the functions you create from a behavioral description are technology independent, they may not be speed and area efficient. The ACTgen Macro Builder ensures that such functions are implemented efficiently.

ACTgen Macro Builder Features

In this section, we describe the features of the ACTgen Macro Builder: flexibility of design, effective use of architecture, and improved designer productivity.

Flexibility of Design

Because each application has its unique attributes, flexibility of design is important. For example, an application may require a 9-bit up counter with enable, whereas another application may require a 22-bit up/down counter. Building a library to support such differing requirements is difficul, but the ACTgen Macro Builder provides the needed flexibility for these kinds of demanding applications.

Effective Use of Architecture

The ACTgen Macro Builder creates functions that effectively use the Actel architecture. Each function has been developed to limit module count, to maximize performance, and to restrict loading to acceptable levels.

Improved Designer Productivity

The ACTgen Macro Builder effectively used the underlying Actel architecture to improve your productivity by building the exact function required and guaranteeing the functionality of design.

Design Flow Diagram

The ACTgen Macro Builder generates counters, registers, latches, I/O macros, subtracters, comparators, accumulators, adders, decoders, and multiplexers. Refer to the *ACTgen Macro Builder User's Guide* for a description of the macro parameters. Figure 1 shows the ACTgen/Synopsys design Flow.

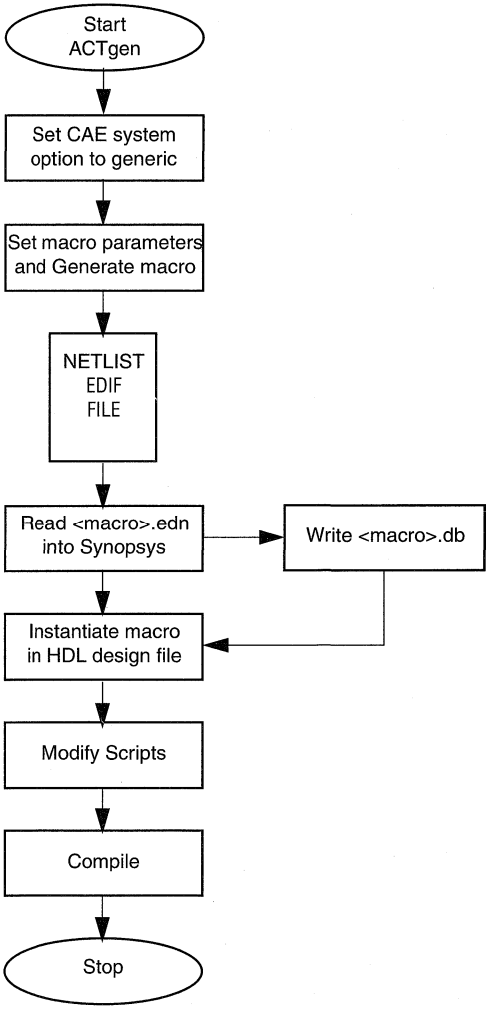


Figure 1 • ACTgen/Synopsys Design Flow

8

Generating an ACTgen Macro

This section describes the steps for generating a macro in ACTgen.

1. Set the CAE system option to EDIF.
2. Select ACT n, where n defines the family option that corresponds to one of the available Actel library families: ACT 1, ACT 2, or ACT 3. Specify the macro type.
3. Set the macro parameters and generate the macro. (ACTgen will write an EDIF netlist in the current directory.)

Integrating Macros into the Synopsys Environment

This section describes the steps for integrating an ACTgen-generated macro into the Synopsys environment.

1. Read the EDIF netlist into a Synopsys design compiler or into a FPGA compiler.
2. Optional: Write the generated macro in<macro>.db format.
3. Instantiate the macro (now read as a <macro.edn> or <macro.db> file) into your VHDL or Verilog design file.
4. Modify Scripts by applying a dont_touch command to the instantiated macro.
5. Compile your design.

Note: ACTgen version 1.1 can also generate a Verilog netlist, so you can read this netlist in your Verilog design files.

Instantiating Macros in Verilog

Verilog provides two instantiation methods. One method is *explicit port name specification* which allows signals to be defined in any order. The second method is *implied port reference by position*, whereby the instantiation ports correspond to the library cell pins in the order the port names are given in the instantiation. An example of instantiation by explicit port name specification is as follows:

```
CNT5 U0 (.Q4(net_q4), .Q3(net_q3), .Q2(net_q2),
.Q1(net_q1),
   .Q0(net_q0), .DATA4(net_d4), .DATA3(net_d3),
   .DATA2(net_d2), .DATA1(net_d1), .DATA0(net_d0),
   .CLOCK(net_clock), .ACLR(net_aclr),
   .ENABLE(net_enable),.SLOAD(net_sload));
```

In our example, CNT5 is the ACTgen-generated 5-bit balanced up counter instantiated as shown in Figure 2. U0 is the instance name.

Also in this example, pins Q4 through Q0, pins DATA4 through DATA0, CLOCK, ACLR, ENABLE, and SLOAD are those of ACTgen counter CNT5. Nets q4 through q0, nets d4 through d0, net_clock, net_aclr, net_enable, and net_sload are connected to the above-referenced pins, respectively.

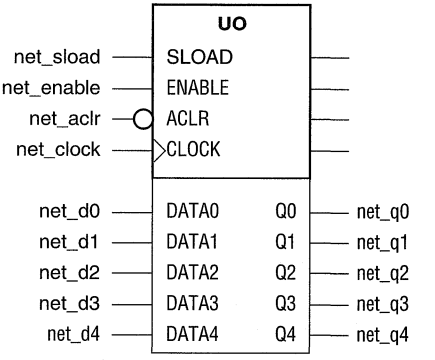


Figure 2 • ACTgen-Generated 5-bit Counter

If you want to use the bus notation as shown in Figure 4, the instantiation syntax is as follows:

```
CNT5 U0 (.Q[4:0](bus_q[4:0]), .DA-
TA[4:0](bus_d[4:0])
   .CLOCK(net_clock),.ACLR(net_aclr),
   .ENABLE(net_enable), .SLOAD(net_sload));
```

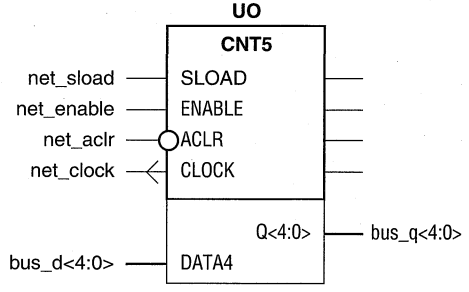


Figure 3 • ACTgen-Generated 5-bit Up Counter, Using Bus Counter

Instantiating Macros in VHDL

The following is in an example ofinstantiating ADD3, an ACTgen-generated 3-bit fast adder shown in Figure 4:

```
/*DECLARATION*/
component ADD3

port(DATAA0,DATAA1,DATAA2,DATAB0,DATAB1,DATA
B3,CIN:in BIT
   SUM0,SUM1,SUM2,COUT:out BIT);
end component;

/*CONCURRENT STATEMENT*/
U1:ADD3 port map ( net_a0, net_a1, net_a2,
net_b0, net_b1,
     net_b2, net_cin, net_s0, net_s1,net_s2,
     net_cout);
```

In the above example, /*DECLARATION*/ is the macro declaration, and /*CONCURRENT STATEMENT*/ is the macro concurrent statement.

U1 is the instance name. Nets net_a0 through net_a2, net_b0 through net_b2, net_s0 through net_s2, net_cin, and net_cout are connected to pins DATAA0 through DATAA2, DATAB0 through DATAB2, SUM0, SUM1 through SUM2, Cin, and Cout, respectively.

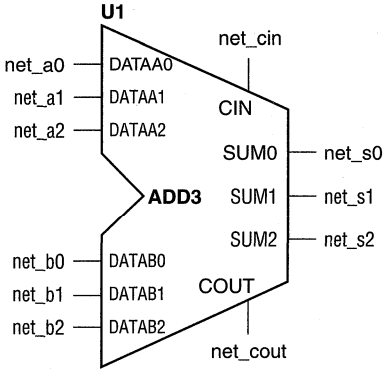


Figure 4 • ACTgen-Generated 3-bit Fast Adder

If you want to use the bus notation as shown in Figure 5, the instantiation syntax is as follows:

```
component ADD3
  port ( CIN : in BIT;
     DATAA, DATAB: in bit_vector(2 downto 0);
     SUM: out bit_vector (0 to 2);
     COUT : out BIT);
end component

U1: ADD3 port map ( net_cin, bus_a, bus_b,
bus_s, net_cout );
```

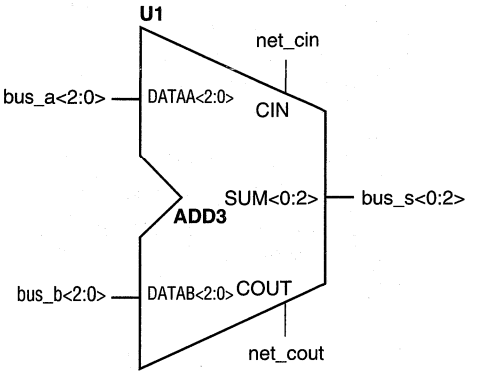


Figure 5 • ACTgen-Generated 3-bit Fast Adder, Using Bus Notation

Designing Macros

This section provides you with an example of the design flow for generating an adder using ACTgen and with information for instantiating the adder in your Synopsys HDL design.

Generating a 3-bit Adder

To design a 3-bit adder:

1. Invoke ACTgen, and select CAE systems as EDIF.
2. Select New Macro, and select Adder as the macro type. Also select ACT 3 as the target family.
3. Select 3 for "width," Fast Adder with Carry in and Carry out for "macro variation," and add3.gen for "file generate." Consequently, an EDIF netlist for the adder, add3.edn, and the macro specification file, add3.gen, are created.
4. Exit ACTgen.

Instantiating the Adder

1. Invoke Design Analyzer, and read in add3.edn.
2. Execute the following Synopsys commands (can be included in a script file). Because the EDIF netlist does not use bus notation, these commands convert the individual bus bits from expanded (net) to bus notation. You have the option of using either one of the command pairs in each set.

```
create_bus {find (port, DATAA*)} DATAA
create_bus {DATAA0, DATAA1, DATAA2} DATAA

create_bus {find (port, DATAB*)} DATAB
create_bus {DATAB0, DATAB1, DATAB2} DATAB

create_bus {find (port, SUM*)} SUM
create_bus {SUM0, SUM1, SUM2} SUM
```

3. Optional: If you want to use the Actel Timer and Debugger, execute the following commands. These commands replace square brackets ([ ]) with operands (< >) for array notation, since Actel Timer and Debugger reject [ ] for array notation.

```
define_name_rules ARRAY -restricted "["
-replacement_char "<"
define_name_rules ARRAY -restricted "]"
-replacement_char ">"
change_names -rules ARRAY -hierarchy
```

4. Optional: Save adder macro as add3.edif and add3.db. Instantiate the macro in your VHDL design file by using the following subroutine:

```
component ADD3
port ( CIN : in BIT;
     DATAA, DATAB:in bit_vector(2 downto 0);
     SUM: out bit_vector (0 to 2);
     COUT : out BIT );
end component

U1: ADD3 port map ( net_cin, bus_a, bus_b, bus_s,
net_cout)
```

Instantiate the macro in your Verilog design file by using the following subroutine:

```
ADD3 U1 (.DATAA[2:0](bus_a[2:0]),
.DATAB[2:0](bus_b[2:0])
  .SUM[0:2](bus_s[0:2]), .CIN(net_cin),
.COUT(net_cout));
```

If you want to use the Actel Timer and Debugger, be sure to use the commands described in step 4.

5. Read in the HDL design files.
6. Place a dont_touch command on the ADD3 macro that is instantiated. Verify that this property is attached by using
   report -cell ADD3.
7. Compile the design.

   If you are not using the bus notation for instantiating ADD3, ignore Step 2.

Maintaining Technology Independence

Instantiating a macro implies that the user is limited to a particular technology. However, you can remain technology independent by creating a behavioral model of the instantiated function. You can have two script files. One file reads <macro.edn> which is an EDIF netlist for the macro being instantiated. The second script file reads the behavioral model <macro.vhd> or <macro.v>, for a technology-independent design.

In our example of the 3-bit fast adder, the script file for Actel contains the following:

```
read -f edif ADD3.edn
```

The script file for the technology-independent design contains the following. You have the option of using either one of the command pairs.

```
read -f vhdl ADD3.vhd
read -f verilog ADD3.v
```

The higher-level file instantiating ADD3 will not change.

Instantiating Actel's I/O Buffers in Verilog HDL

Introduction

Actel supports design environments where Verilog HDL language is used to describe the design. This method of design entry allows for a high-level behavioral description of the design without any gate-level specifications. This HDL description, once compiled and synthesized with Synopsys and the Actel library, is mapped into an Actel field programmable gate array (FPGA). During the synthesis process, designs are mapped into Actel library macros, including the I/O buffers. Synopsys 3.0 provides this I/O mapping capability. Previous versions of Synopsys did not perform automatic I/O buffer instantiation. What follows briefly describes methods of Actel I/O buffer instantiation in a Verilog HDL design.

Instantiating I/O Buffers

With Synopsys 3.0, there is no need to instantiate I/O buffers in a gate-level format in the HDL design file. The exceptions to this rule are the tristate and bidirectional I/O buffers, which must be instantiated in a gate-level format in the HDL design file. To perform automatic I/O buffer mapping with the design_analyzer or the dc_shell command line, the following switches must be set prior to optimization:

```
set_port_is_pad all_inputs()
set_port_is_pad all_outputs()
set_pad_type all_inputs()
set_pad_type all_outputs()
set_pad_type -clock CLK
insert_pads
```

Note: *CLK is an example signal name.*

These switches will cause the Synopsys tools to map each port specified to an Actel I/O input buffer. The output ports are mapped to I/O output buffers. If there is a clock function in the HDL design file, it will be mapped to a CLKBUF. After each switch is set, messages appear showing the corresponding switch function being implemented for the applicable I/O ports, as shown in the example below:

```
Performing set_port_is_pad on port 'in1'.
Performing set_port_is_pad on port 'out1'.
Performing set_pad_type on port 'in1'.
Performing set_pad_type on port 'out1'.
```

If a version of Synopsys released before version 3.0 is being used or if a tristate or bidirectional I/O function needs to be implemented, the I/O buffers must be instantiated in a gate-level format in a Verilog HDL design file. The syntax for gate-level Actel I/O buffer instantiation in a Verilog HDL design file is shown in the example below:

```
module top(clear,clock,enable,outx);
input clear, clock, enable;
output [4:0] outx;

reg [4:0] outx_o;
reg clear_i,clock_i,enable_i;

count U0 (clear_i,clock_i,enable_i,outx_o);

CLKBUF U1 (.PAD(clock),.Y(clock_i));
INBUF U2 (.PAD(clear),.Y(clear_i));
INBUF U3 (.PAD(enable),.Y(enable_i));
OBHS U4 (.D(outx_o[4]),.PAD(outx[4]));
OBHS U5 (.D(outx_o[3]),.PAD(outx[3]));
OBHS U6 (.D(outx_o[2]),.PAD(outx[2]));
OBHS U7 (.D(outx_o[1]),.PAD(outx[1]));
OBHS U8 (.D(outx_o[0]),.PAD(outx[0]));
endmodule
```

In this example, the module "top" represents the highest level of hierarchy that instantiates the main behavioral Verilog HDL design module "count". In addition, the I/O buffers are described in a gate-level format.

The list of ports specified in the module "top" represent the Actel I/O pin names. These I/O pin names (clear, clock, enable, outx) are connected to the PADs on the I/O buffers. The items on the list of ports specified in the count module have a one-to-one correspondence to the PAD pins. The syntax of the Verilog HDL code instantiating the Actel I/O buffers is shown in Figure 1.

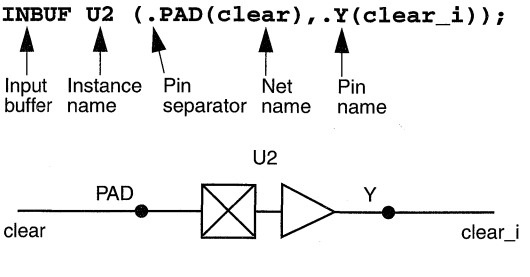


Figure 1 • Actel I/O Buffer with Verilog HDL Code Instantiation

Generating/Checking CRC for IEEE 802.3 (LAN Interface)

The Carrier Sense Multiple Access with Collision Detection (CSMA/CD) media access method is the means by which two or more stations share a common bus transmission medium. IEEE 802.3 is a standard for local area networks (LAN) employing CSMA/CD as the access method.

IEEE 802.3 consists of submodules such as Receiver, Transmitter, Clock Synchronization, and CRC, as shown in Figure 1.

The focus of this application note is the design of a Cyclic Redundancy Check (CRC) submodule compatible with the IEEE 802.3 protocol. This includes generating the CRC code before transmission and checking the coherency of the data at receiver time.

Cyclic Redundancy Check

CRC is a way to detect small changes in blocks of data. Although a few errors in a text file may be acceptable, when transmitting a computer program, an error of even 1 bit is sufficient to make a program faulty. An error-correcting protocol triggered by a CRC error detector can provide protection. The CRC code calculation usually is different from one protocol to another. This application note focuses on the LAN 802.3 protocol algorithm and includes a design implementation and the issues involved in this implementation.

IEEE 802.3 Frame Structure

A Media Access Control (MAC) frame packet is partitioned into six major sections: Preamble, Destination, Source Address, Byte Count, Data Field, and Frame Check Sequence (FCS). Table 1 demonstrates the LAN frame structure.

Table 1 • LAN Frame Structure

Preamble	8 Bytes
Destination Address	6 Bytes
Source Address	6 Bytes
Byte Count	2 Bytes
Data Field	46 to 1500 Bytes
Frame Check Sequence (FCS)	4 Bytes

The Preamble field is used for synchronization between the receiver and transmitter clock. This field has seven sequences of 101010 followed by one byte of 10101011. This is the last byte indicating the start of the frame—Start Frame Delimiter (SFD) byte.

Each MAC frame has two address fields: the destination address field and the source address field. The destination field specifies the station or stations for which the frame was intended. The source address field indicates the station sending the frame.

The Byte Count field is a 2-byte field; its value indicates the number of logical link control (LLC) data bytes in the data field.

The data field contains a sequence of n bytes that may arbitrarily appear in the data field. The maximum size of this field is (2 x (address size) + 48)/8 bytes, and the minimum frame data is ((8 x n) + (2 x address size) + 48) bits.

The FCS field contains CRC code. This field is 4 bytes long and is used to check the integrity of the transmitted data on the LAN protocol.

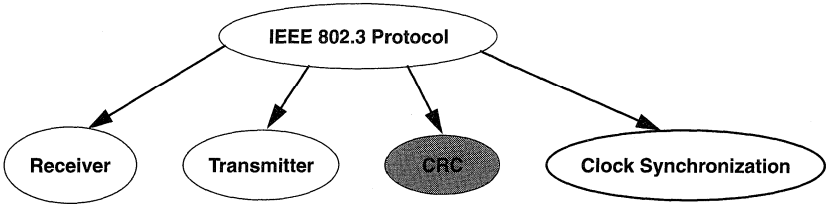


Figure 1 • IEEE 802.3 Protocol Submodules

CRC Module Design

The first task in designing a CRC module is to find and understand the linear algebra that represents the CRC code calculation. The CRC algorithm operates on a block of data transmitted serially as a unit. This block of data can be looked at as a large numerical value. The CRC algorithm divides this large number by a magic number—the CRC polynomial. This operation will leave the remainder with a unique CRC code. After CRC code calculations, the resulting CRC number is usually stored along with the data. The following describes this linear division for a 32-bit 802.3 LAN controller:

$$G(X) = X^{32} + X^{26} + X^{23} + X^{22} + X^{16} + X^{12} + X^{11} + X^{10} + X^8 + X^7 + X^5 + X^4 + X^2 + X + 1$$

When data is received from storage, the unique CRC code is received along with the data. The CRC design algorithm can be repeated, and the remaining results should be the magic CRC number. This number for LAN 802.3 protocol is DEBB20E3 hex, and since the number is inverted before transmission on the line, it is C704DD7B hex. When checking for validity of data, any other remainder in the CRC register is an indication of error.

The CRC design interface for IEEE 802.3 is completely synchronous and register intensive. This makes it ideal for Actel architecture. CRC design consists of many add and shift operations in every clock cycle. This makes it an ideal choice for behavioral design entry methodology. Using a schematic design tool is not advisable, since design is error prone and schematics are hard to change.

Design Implementation

The following is the `crc_design` file implemented in ACTmap VHDL with some predefined sequential procedures that can be used to define this design. The behavioral code is then processed by Actel's VHDL synthesis and optimization tool, ACTmap. ACTmap is a computer-aided design tool for working with the Actel families of FPGAs. It performs three basic functions:

- PALASM2, VHDL to netlist translation
- Netlist Optimized mapping
- I/O insertion

ACTmap reads the PALASM2, or VHDL source file and translated it into either an EDIF or an ADL (Actel Design Language) output file or Verilog netlist. The output file that it generates is optimized for a specific family of Actel FPGAs (ACT 1, ACT 2, 1200XL, or ACT 3).

LAN 802.3 also includes the Receiver and Transmitter submodules. Both Receiver and Transmitter submodules are counter intensive. The Actel ACTgen macro generator can create structural macros such as counters that are optimal for Actel devices. These structural blocks can easily be instantiated through Actel VHDL. Even though receiver and transmitter designs are not the focus of this application note, it is informative to know that the ACTgen module generator can easily be linked to the Actel VHDL entry tool.

This application note is focused on generating the correct CRC code at the transmission time and checking for validation of frame data at receiving time. A functional block diagram of the CRC design is shown in Figure 2.

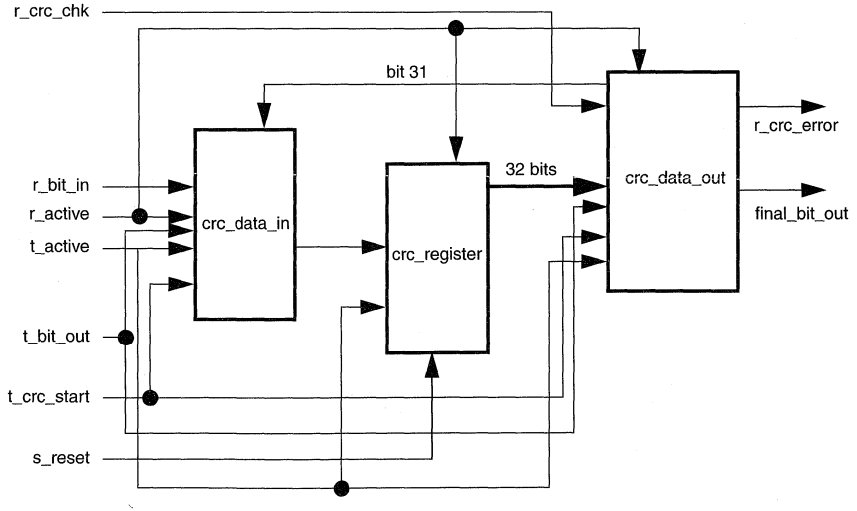


Figure 2 • Functional Block Diagram

VHDL Code Description

The following is the ACTmap VHDL code representing this design:

```
--LIBRARY ieee; used for simulation
--USE ieee.std_logic_1164.ALL; used for VHDL simulators

entity crc_data_in is

--decides which data should be
-- sent to crc_register entity.

     port (rcv_active,trm_active: in bit;
     rcv_bit_in,trm_bit_out,crc_reg_b, trm_crc_start: in bit;
     crc_data_in: out bit);
end crc_data_in;

architecture archi of crc_data_in is
signal mux_data_out: bit;
begin

  mux_data_out<= trm_bit_out when (trm_active = '1') else
            rcv_bit_in when (rcv_active = '1') else
            '0';
--This will feed bit 31 of crc_register back in, with the new data.
--It also disables the crc calculation when transmitting crc code.

crc_data_in <= not(trm_crc_start) and (crc_reg_b xor mux_data_out );

end archi;

entity crc_register is
--This is the main module that applies the crc algorithm to data.

port(   clk,system_reset,crc_data_in:   in bit;
       rcv_active,trm_active:   in bit;
       crc_reg          :out bit_vector(31 downto 0);
       crc_reg_b        :out bit);
end crc_register;


architecture crcreg of crc_register is
signal nextreg, temp :bit_vector(31 downto 0);
signal reset_crc        :bit;

begin

reset_crc <= '1' when system_reset = '1'  else
           '1' when  ( (rcv_active = '0') and (trm_active = '0')) else
           '0';

crc_reg <= nextreg ;
DFFC_V(temp, reset_crc, clk, nextreg);
temp <= (nextreg(30 downto 26) &
             (nextreg(25) xor crc_data_in) &
             (nextreg(24 downto 23)) &
             (nextreg(22) xor crc_data_in) &
             (nextreg(21) xor crc_data_in) &
             (nextreg(20 downto 16)) &
             (nextreg(15) xor crc_data_in) &
             (nextreg(14 downto 12) ) &
```

```
            (nextreg(11) xor crc_data_in) &
            (nextreg(10) xor crc_data_in) &
            (nextreg(9) xor crc_data_in) &
            nextreg(8) &(nextreg(7) xor crc_data_in) &
            (nextreg(6) xor crc_data_in) &
            nextreg(5) &
            (nextreg(4) xor crc_data_in) &
            (nextreg(3) xor crc_data_in) &
            nextreg(2) &
            (nextreg(1) xor crc_data_in) &
            (nextreg(0) xor crc_data_in) &
            crc_data_in);
            crc_reg_b <= nextreg(31);

end crcreg;



entity crc_data_out is
-- This is the crc_data_out module that checks for errors.

  port (rcv_active,trm_active: in bit;
       trm_data, crc_reg_b: in bit;
       trm_crc_start,rcv_crc_chk: in bit;
       crc_reg: in bit_vector(31 downto 0);
       trm_bit_out,rcv_crc_error: out bit);

end crc_data_out;

architecture archi of crc_data_out is
signal crc_constant : bit_vector(31 downto 0);

begin


trm_bit_out <= (not crc_reg_b) when ((trm_active = '1') and (trm_crc_start = '1')) else
         trm_data when (trm_active = '1') else
         '0';
-- checking for constant magic number
crc_constant <= x"c704dd7b";

rcv_crc_error <= '0' when (rcv_active = '0') else
        '0' when ((crc_reg = crc_constant) and (rcv_crc_chk = '1')) else
        '1' when (rcv_crc_chk = '1') else
        '0';
end archi;

-- This is the top-level module that binds all entities together.
entity enetcrc is

  port(clock,t_crc_start,r_bit_in,t_bit_out :in bit;
    s_reset,r_active,t_active,r_crc_chk :in bit;
    final_bit_out,r_crc_error     :out bit);
  end enetcrc;


architecture structure of crc_design is

component crc_data_in
       port (rcv_active,trm_active                    : in bit;
         rcv_bit_in,trm_bit_out,crc_reg_b,trm_crc_start  : in bit;
         crc_data_in                                 : out bit);

end component;
```

```
component crc_register
  port (clk,system_reset,crc_data_in:   in bit;
        rcv_active,trm_active:   in bit;
        crc_reg        :out bit_vector(31 downto 0);
        crc_reg_b       :out bit);

end component;
component crc_data_out
        port (rcv_active,trm_active: in bit;
          trm_data, crc_reg_b: in bit;
          trm_crc_start,rcv_crc_chk: in bit;
          crc_reg: in bit_vector(31 downto 0);
          trm_bit_out,rcv_crc_error: out bit);

end component;

for all: crc_data_out use entity work.crc_data_out(archi);
for all: crc_data_in use entity work.crc_data_in(archi);
for all: crc_register use entity work.crc_register(crcreg);

signal bit31,data_in      :BIT;
signal crc32bit_reg       :bit_vector(31 downto 0);

begin

  U1:crc_data_in port map
  (r_active,t_active,r_bit_in,t_bit_out,bit31,t_crc_start,data_in);

  U2:crc_register port map
  (clock,s_reset,data_in,r_active,t_active,crc32bit_reg,bit31);

  U3:crc_data_out port map
  (r_active,t_active,t_bit_out,bit31,t_crc_start,r_crc_chk,crc32bit_reg,
   final_bit_out,r_crc_error);

end structure;
```


The entity crc_data_in acts as a switch between receiver and transmitter. The code in this module is designed to use the same circuit for the data receiver CRC check and the CRC code generator at transmission time. In this module, when t_active is enabled, it enables the CRC code generation. The T_bit_out bit will be half added (XOR) with bit 32 of the CRC register and will be shifted in the CRC register for CRC code generation. T_bit_out will also be sent on line serially. At the end of the Byte Count field the t_crc_start bit in the top level will be set. This indicates that the generated CRC code needs to be transmitted. The 4 bytes of the CRC code are transmitted with bit 31 first and bit 0 last.

The crc_register entity generates the CRC code at transmission time and applies the polynomial division on data bits at receiver time. In this module, the DFFC_V predefined procedure of ACTmap_VHDL is used. The cyclic redundancy check is computed at transmission time as a function of all the frame data fields except Preamble and FCS (CRC). The CRC algorithm applies modulo 2 division on data; this means it uses the XOR operation instead of the normal add and subtract. Note that bit 31 of the crc_register is folded back into the polynomial operation. When t_crc_start is set, the uniquely calculated CRC code will be shifted on line at every clock period. As the most significant bit is shifted out, the least significant bit will be replaced by 1.

In the receiving side, all previously mentioned fields except for Preamble (including FCS) run through the CRC algorithm again. When the last CRC code is received, the CRC check flag will be set (r_set_chk).

The last entity, crc_data_out, will check the number remainder in the crc_register for the constant magic number C704DD7B hex. Any other number produces an error.

The top-level design, enetcrc, instantiates these modules and binds them together.

Synthesis and Optimization

The VHDL code described in the previous section is synthesized and optimized by ACTmap, which generates a gate level description optimized for Actel's architecture. ACTmap can output the results in any of the following netlist formats: EDIF, Verilog, ADL, and Viewlogic. The following results were obtained from ACTmap targeting an ACT 3 device.

Figures 3 through 6 show the schematic representation of the netlist generated by ACTmap.

Conclusion

Trying to find better ways to quickly bring high performance products to market is not new. What is new is a design methodology combining the use of Field Programmable Gate Arrays (FPGAs) with high-level design entry in high-speed systems. New design flows use sophisticated synthesis tools to translate generic high-level design description into device specific netlists. Synthesis targets specific FPGA architectures and devices for optimum fit and performance. The Cyclic Redundancy Check circuit described in this application note takes advantage of high-level design entry capabilities of Actel tools.

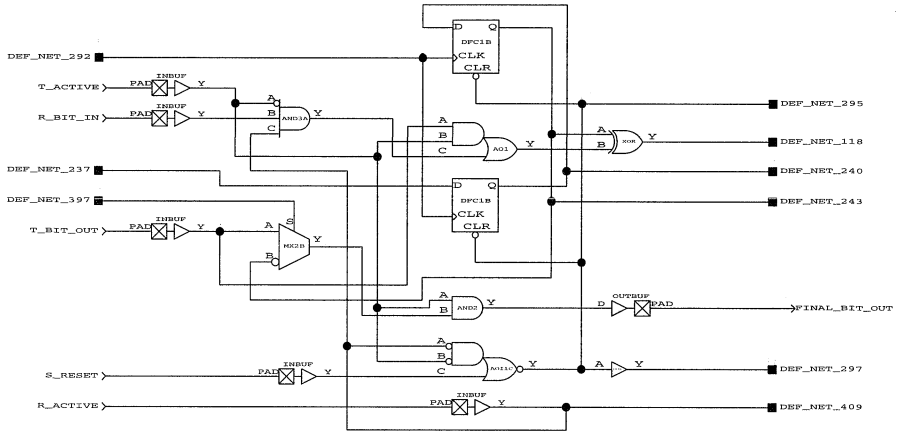


Figure 3 • Schematic Representation of the Top Level Design, enetcrc

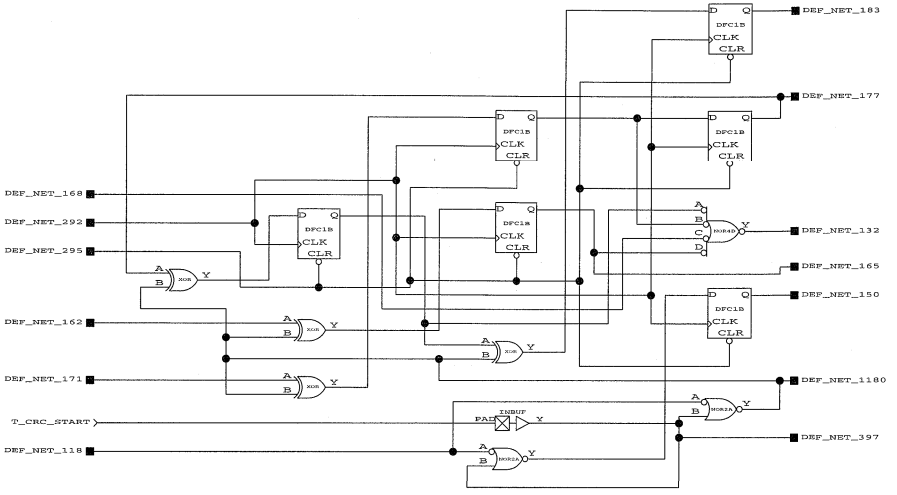


Figure 4 • Schematic Representation of the Top Level Design, enetcrc (Continued)

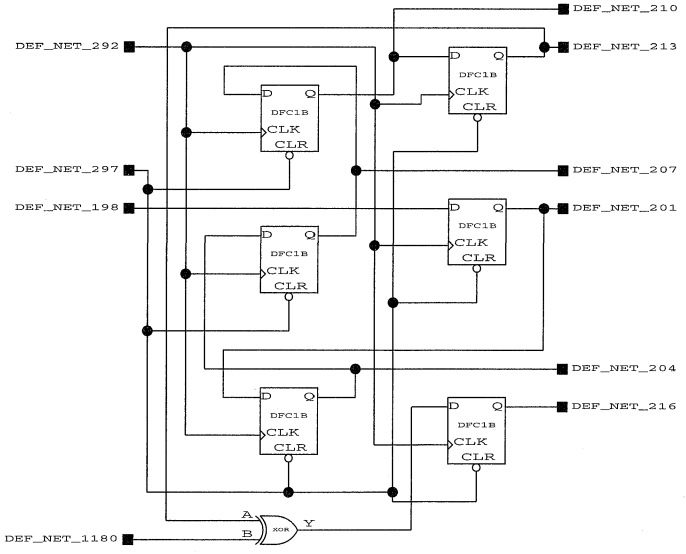


Figure 5 • Schematic Representation of the Top Level Design, enetcrc (Continued)

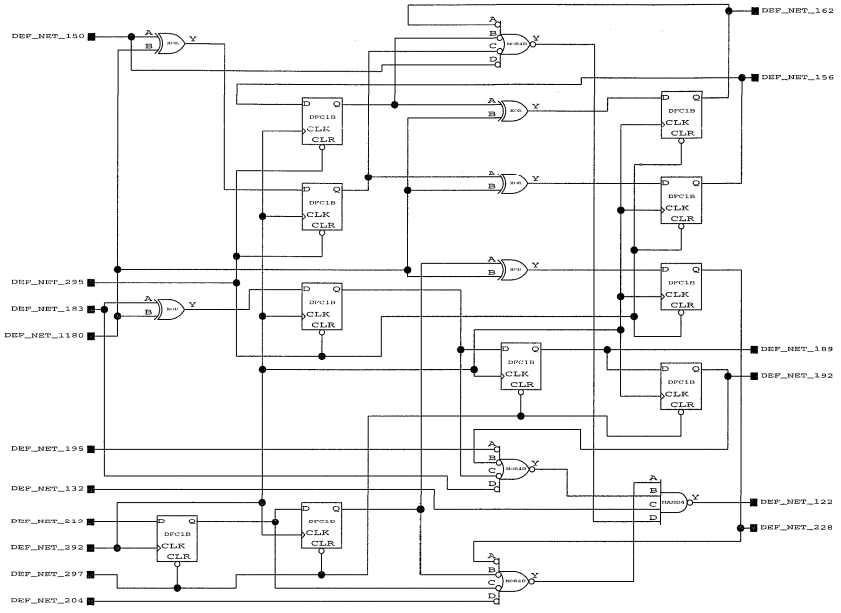


Figure 6 • *Schematic Representation of the Top Level Design, enetcrc (Continued)*

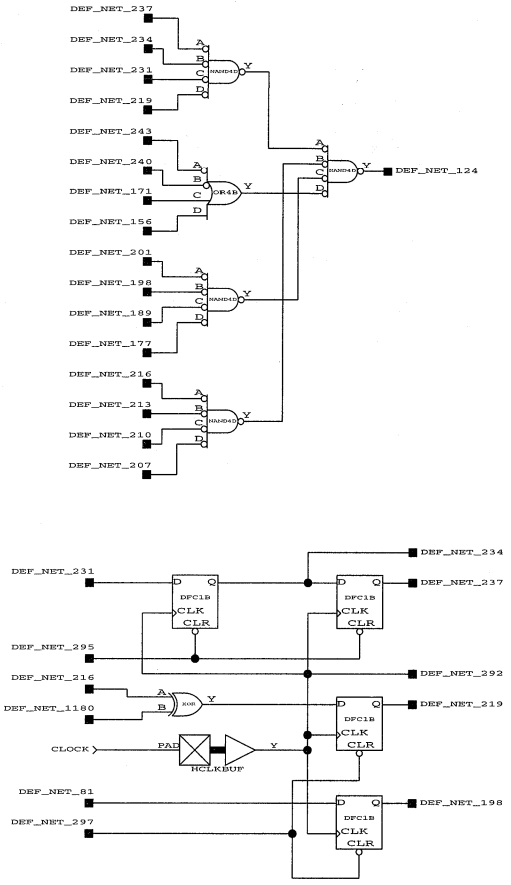


Figure 7 • Schematic Representation of the Top Level Design, enetcrc (Continued)

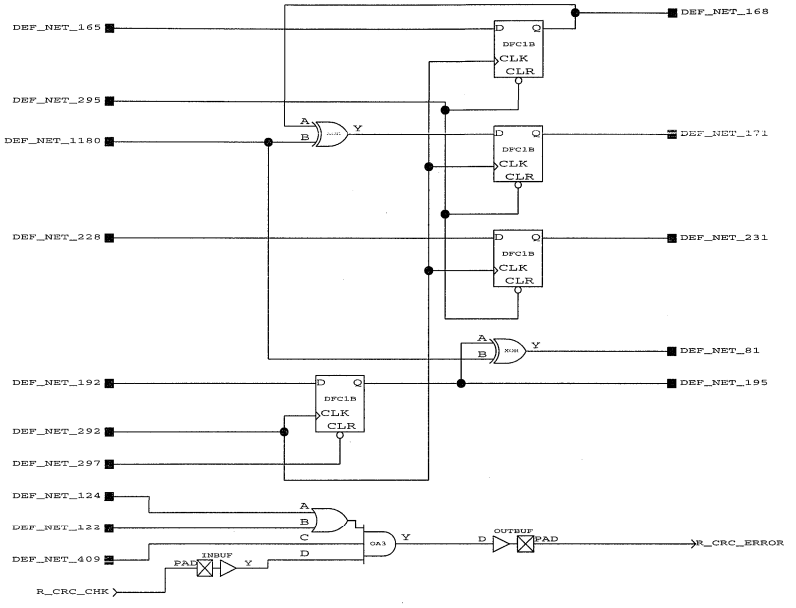


Figure 8 • Schematic Representation of the Top Level Design, enetcrc (Continued)

An Overview of Synthesis for FPGA Design

Introduction

Synthesis has been widely accepted in the ASIC design community for providing significant improvements in both time to market and user productivity. The adoption of this methodology for FPGA design, however, has been tentative due to perceived limitations in either performance or utilization. These limitations are being removed through the joint efforts of EDA and FPGA vendors. This paper provides an overview of new products and techniques that allow synthesis users to achieve device performance and utilization that match or exceed levels previously obtainable only from schematic-based design of FPGAs. Examples using these techniques will be described in the following applications articles.

The Synthesis Process

The real challenge for synthesis of integrated circuits, and especially FPGAs, is maintaining high quality of results (speed and utilization) with increasingly higher levels of abstraction in the design specification. Higher levels of abstraction provide users with the ability to design larger circuit functions in the same time. Synthesis technology has in fact allowed a tenfold increase in productivity in the last six years. In other words, synthesis technology has allowed system engineers to keep pace with Moore's Law—circuit complexity and performance double every 18 months (about tenfold every six years).

This challenge is especially important as FPGA performance and capacity tend to lag the gate-array capabilities. Users need to push the limits of the FPGA while maintaining the synthesis-based methodology they have adopted for ASIC.

Keys to Synthesis Success

There are some key requirements for ensuring success for synthesis.

Simulation

To take full advantage of language-based design, you should completely verify functionality at the behavioral level, before synthesizing to gates. This will simplify verification after synthesis and will allow the designer to focus on timing.

Understand the Target Device

Although technology independence is a touted benefit of synthesis-based design, familiarity with the key aspects of the target device is necessary. What is the approximate performance of the register, clock network, and I/O devices? What is the approximate size of the device? Are there special features such as clock network drivers and I/O clocks? FPGA and CAE vendors are constantly improving the support for special features, but the designer must be aware of what is available and should verify their correct use.

Understand the Synthesis Tool

An understanding of the synthesis tool is perhaps an obvious requirement, but each tool has numerous options controlled by user selection or *scripts*. The proper use of scripts to influence the synthesis result is crucial.

Constraints

The proper definition of area and timing constraints is the single most important design input after behavioral description. Most commercial synthesis tools use constraints to guide the synthesis process. In addition, detailed constraints simplify the verification process by using constraint-based timing reports. Constraints can be quite simple; for example, a 25 MHz requirement on 'sysclk' would be

```
create_clock -period 40 sysclk
```

Constraints should be defined for all device clocks (including duty cycle), input setup time on data arrival to registered inputs, off chip clock to out, and all pad-to-pad combinatorial paths. It is also important to indicate exceptions if possible. Even in a high-performance application, there is always some logic that is noncritical.

Structured Macros

The granularity of a programmable logic device is larger than that of a traditional masked programmed ASIC or gate array. This is a direct result of balancing routing flexibility with routing performance. In the Actel FPGAs, the logic module is approximately the complexity of a four-input multiplexor plus AND/OR control gates. The ACT 2 and ACT 3 families also contain D-type flip-flops in half of the modules. Although the

relatively small size and the simplicity of these logic modules are beneficial to general synthesis algorithms, there is still sufficient complexity that must be overcome with other techniques. Structured macro generation is a technique of mapping large MSI functions (adders, counters, compactors, decoders, etc.) into optimum implementations using application knowledge. For Actel, these generators are available to schematic-based design with the ACTgen Macro Builder. Several commercial synthesis tools (Synopsys, Exemplar, and IST) provide interfaces to Actel's Macro Builder.

The Synopsys FPGA compiler uses the Actel DesignWare library (DW_ACT). With this library, there are three ways to include structured macros in your design: *operator inferencing*, whereby the compiler infers the best implementation for an HDL operator; *explicit referencing,* whereby a specific arithmetic implementation (+, –) or logic operation (<, ≤, ≥, >) is declared by explicit statements enclosed within comment characters; *instantiation,* whereby DesignWare components are directly placed into the code. Actel currently offers a wide range of parameterized counters for instantiation.

Hierarchy

Design hierarchy is a powerful feature of language-based design. You should always partition your design into blocks with definable performance. This will simplify constraint definition and design iteration, especially if component instantiation is required for timing critical paths. Flattening the overall design can sometimes improve speed and area, but at a loss of traceability. This option should only be used late in the design process.

Language Style

There are usually several ways to describe circuit behavior but only a few good ways to describe hardware. This is especially true for structure-rich FPGA architecture. FPGA vendors provide help through design methodology notes and guidelines. Benchmark designs are emerging as guideposts for illustrating the best technique for a particular logic function or application.

For the Actel multiplexor-based module, case statements produce more efficient mapping than do long nested if-then-else statements.

Benchmarks

There is no substitute for good examples. Benchmark designs are frequently used to select target devices, but care must be taken to ensure that the benchmarks reasonably represent your design's requirements. In addition, benchmarks must include the constraints, scripts, and coding styles that contribute to the results—allowing you to adapt your design style accordingly.

Technology Mapping

Technology mapping is usually performed as the last stage of the synthesis process. Frequently it is performed automatically. Actel provides a technology-mapping utility program, the ACTmap Technology Fitter. It provides speed and area optimization, fan-out control, and automatic I/O insertion, and it contains special algorithms for mapping logic into both logic and I/O modules. In addition to optimization, ACTmap can be used for language-based design through PALASM or VHDL.

Actel VHDL

Actel VHDL is a simple behavioral entry tool with some predefined sequential procedures that can be used to define a design more easily. The ACTmap program translates, synthesizes, and maps the design to any Actel device. In addition, ACTmap supports block- or chip-level design, including automatic insertion of I/Os.

ACTmap FPGA Fitter

Features

- Actel-specific mapping algorithms—binary decision diagrams (BDDs)
- Interfaces with popular PLD and schematic tools
- Choice of speed or area optimization
- Automatic or user-controlled I/O buffering
- Three-state mapping to Actel multiplexers
- One-hot and compact state machine encoding
- Fast execution
- PC and workstation versions

Introduction

Actel's ACTmap™ FPGA fitter offers efficient logic optimization and mapping to Actel's ACT™ 1, ACT 2, and ACT 3 FPGA families. Actel-specific, logic-mapping algorithms use an advanced logic representation called binary decision diagrams (BDDs) to take maximum advantage of Actel's multiplexer-based architecture. Users select between speed or area optimization and have the option to automatically insert I/O buffers. Designers who want to use familiar tools and languages can benefit from ACTmap fitter optimization and mapping capabilities. ACTmap fitter reads the output of popular PLD tools such as ABEL™, CUPL™, LOG/iC™ from Isdata, and MINC's PLDesigner-XL™. The EDIF netlist interface links ACTmap fitter with a wide variety of schematic capture packages. State machines entered through PLD design tools can easily be combined with datapaths entered by means of a schematic. Powerful state encoding options ensure efficient implementation and high performance. See Figure 1 for a block diagram.

Actel-specific Mapping

For high-performance designs, the logic-mapping process is especially important. Poor use of chip features leads to unacceptable performance, low effective capacity, or both. ACTmap fitter fitter avoids these problems by using mapping algorithms specific to Actel FPGAs. For Actel's multiplexer-based FPGAs, ACTmap fitter uses a technique based on binary decision diagrams. BDDs facilitate logic representation that permits extremely efficient mapping to the Actel module with minimum time and memory usage (see the BDD representation in Figure 2).

Fast State Machines

The performance of state machines in FPGAs depends heavily on the choice of state encoding techniques. FPGAs have a plentiful supply of registers compared to PLDs. Thus, one-hot state assignment, which assigns one register per state, is practical and often produces superior performance. In other cases, compact encoding produces better results. Compact encoding takes advantage of a unique algorithm that creates area efficient mappings, improving area utilization by 30% over randomly selected encodings. Only ACTmap fitter provides the user the advantage of compact encoding.

Rapid Design Iteration

With optimization central to every design cycle, rapid compilation is essential. The advanced algorithms used in ACTmap fitter permit the compilation of a 4000-gate FPGA in under 15 minutes on a Sun™ SPARC10. Performance on a PC is comparably fast. Unequaled high performance means easier design modifications and improved productivity with ACTmap fitter.

Hardware Requirements

On a PC, ACTmap fitter requires:

- MS-DOS® operating system version 4.0 or later.
- Personal computer with 386 or higher microprocessor.
- Minimum of 8 MB of memory (16 MB recommended).
- Hard disk with 10 MB of available space.
- A 3.5-inch, 1.44 MB floppy drive.

Options:

- Microsoft® Windows™ version 3.1 or later. When using Microsoft Windows, 16 MB of memory is required.

On a Sun SPARC® workstation, ACTmap fitter requires:

- Sun OS 4.1 or later.
- 16 MB of memory.
- Hard disk with 10 MB of available space; additional 40 MB recommended for swap space.
- High-density cartridge tape drive.

Options:

- Sun OpenWindows™ version 3.0 or later.

8

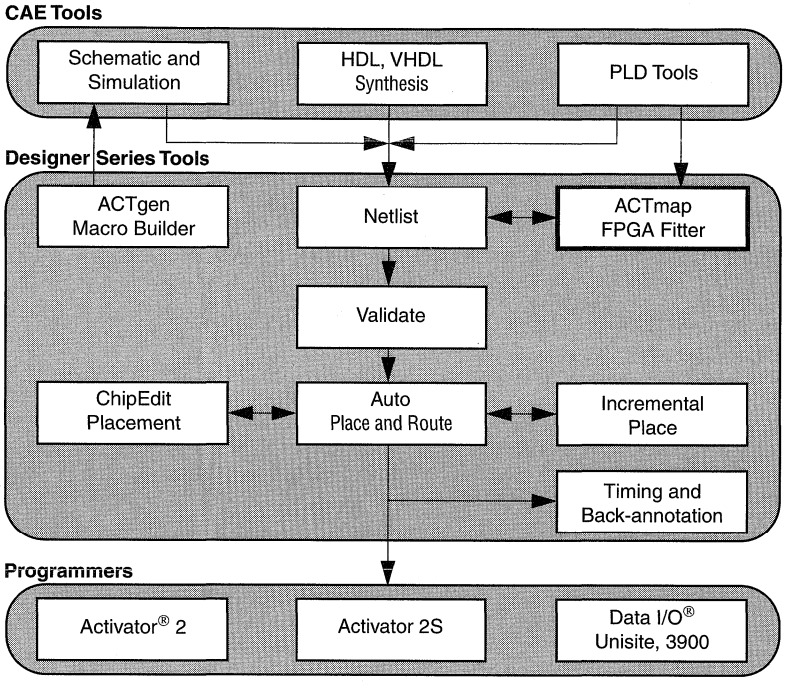


Figure 1 • Actel FPGA Design Flow

Figure 2 • BDD Representation

ACTgen Macro Builder

Features

- Parameterized macro-function generator for counters, adders, and other structured blocks
- Creates a function of any width from 2 to 32 bits for each macro
- Enhances device performance and designer productivity
- Choice of area- or performance-optimized functions
- Tailored to take advantage of Actel-specific architectural features
- Included in Designer Series Development Systems at no added cost
- Menu-driven Microsoft® Windows™-compatible user interface or X-Window workstation user interface

Introduction

Actel's ACTgen™ Macro Builder offers efficient implementation of counters, adders, and other structured blocks. Architecture-specific rules control the generation of macros to optimize implementation for either speed or area efficiency. Performance is increased by up to 114% over synthesized versions of the same function. ACTgen Macro Builder offers the flexibility to create a function of any width from 2 to 32 bits and to specify parameters such as load, clear, and enable. The quality of output is "correct by construction" so that no logic verification is required by the user. Overall design productivity is enhanced because implementation is faster and less time is spent fine-tuning the design's performance.

The ACTgen Macro Builder is an integral part of the Designer Series of development tools, which feature fully automatic placement and routing, timing analysis, and support for behavioral or schematic-based design entry via links to popular CAE and PAL®/PLD design tools. Compatible with Microsoft Windows and UNIX X-Window, ACTgen Macro Builder is easy to learn and easy to use. See Figure 1 for a block diagram.

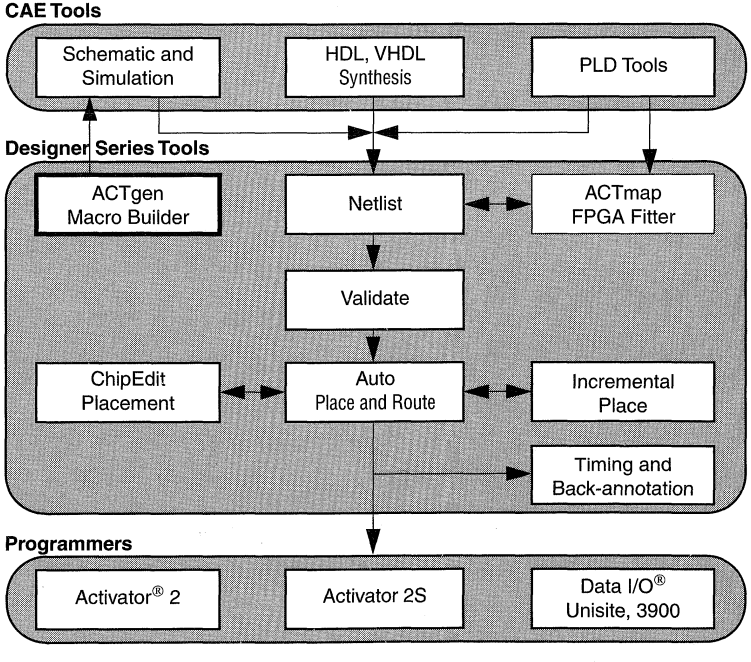


Figure 1 • Actel FPGA Design Flow

Product Details

ACTgen Macro Builder is menu driven and allows the user to choose from a wide range of functions and selectable parameters. In all, hundreds of unique macro functions can be created. For example, there are four different types of counter implementations, each with size options of 2 to 32 bits and parameter selections for asynchronous clear, synchronous load, enable, three-state enable, direction (Up, Down, or both), clock (rising or falling), and high and low slew controls (see Figure 2). This offers significantly greater breadth than would be practical in a soft macro library.

ACTgen Macro Builder version 1.0 features support for counters, adders, registers, decoders, and multiplexers, and allows targeting to either ACT™ 2 or ACT 3 devices. With choices of speed- or area-optimized versions, and a pre-route performance and area estimation capability, ACTgen Macro Builder allows what-if analysis and selection of the appropriate implementation and device target early in the design process. All macros are compatible with emerging LPM standards, and additional macros will be added with each subsequent release of the product. Designer Series users, with up-to-date support agreements, receive the current release of ACTgen Macro Builder and future updates at no added cost.

The Right Tools for the Job

Design implementation is completed using a combination of the ACTgen Macro Builder, the ACTmap™ FPGA Fitter, and the user's design entry tools-of-choice. ACTmap fitter is used to optimize control logic or state machines and target Actel's architecture-specific features. ACTgen Macro Builder is used to create datapath functions, counters, adders, and other structured blocks. ACTgen Macro Builder produces a Viewlogic PROcapture™ schematic symbol for annotation of the created block in the top-level schematic. For other CAE environments or when using HDL tools, an EDIF netlist is exported and may be used to regenerate schematics or instantiate ACTgen Macro Builder created blocks in a high-level description. Top-level description of the entire design is completed via schematic or high-level description and may include blocks created by ACTmap fitter and ACTgen Macro Builder, along with hard and soft macro library elements. This design approach ensures the highest possible performance and designer productivity—a combination of ease of use and quality of results that guarantees success in achieving design objectives and meeting time-to-market requirements. A short learning curve is ensured through integration with popular CAE and behavioral entry tools and a Microsoft Windows- and UNIX X-Window-compatible user interface.

Hardware Requirements

On a PC, ACTgen Macro Builder requires:

- MS-DOS® operating system version 4.0 or later.
- Personal computer with 386 or higher microprocessor.
- Minimum of 16 MB RAM memory.
- Hard disk with 10 MB of available space.
- Microsoft Windows version 3.1 or later.

On a Sun™ SPARC® workstation, ACTgen Macro Builder requires:

- Sun OS 4.1 or later.
- 16 MB of memory.
- Hard disk with 10 MB available space; additional 40 MB recommended for swap space.
- High-density cartridge tape drive.

Options:

- Sun OpenWindows™ version 3.0 or later.

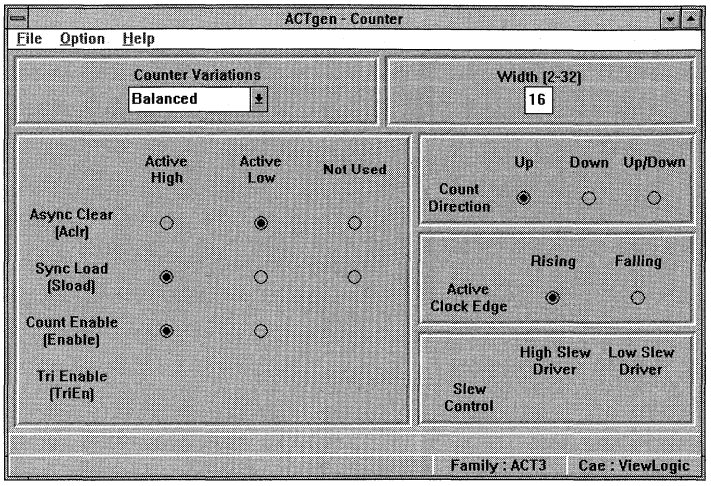


Figure 2 • ACTgen Macro Builder User Interface

ACTgen Macro Builder versus Synthesis Speed and Capacity Benchmarks

Design	Description	Performance (MHz)			Capacity (Modules)		
		Synthesis	ACTgen	Delta	Synthesis	ACTgen	Delta
ADD8	8-bit adder	18	26	46%	45	38	16%
ADD16	16-bit adder	10	22	114%	63	91	–44%
ADD32	32-bit adder	16	17	5%	259	194	25%
CNT8	8-bit counter	35	56	57%	41	23	44%
CNT16	16-bit counter	28	50	77%	89	47	47%
CNT32	32-bit counter	26	41	61%	239	101	58%
DED8	3-to-8 decoder	37	51	37%	14	8	43%
DED16	4-to-16 decoder	41	42	2%	22	24	–9%
DED32	5-to-32 decoder	34	39	14%	71	44	38%
MUX8X16	8-, 16-bit bus MUX	25	26	5%	105	48	54%
MUX16X8	16-, 8-bit bus MUX	20	27	34%	96	58	40%
	Average	27	36	41%	94.9	61.5	28%

Designs are optimized for both synthesis and ACTgen Macro Builder.

Counter frequency based on maximum internal delay (clock to data); all others based on maximum input pad-to-output pad delay.

All designs implemented in A1425A, except MUX8X16 and MUX16X8, which are implemented in A1460A.

Designing a DRAM Controller Using Language-Based Synthesis

Introduction

Why Use HDL?

A new level of abstraction for the design descriptions uses a combination of synthesis tools and HDL languages. As the trend toward more complex designs continues, the old ways of design entry do not accommodate everyone's needs. In the same way that higher level programming languages replaced assembly languages, traditional schematic capture and truth tables will be substituted by HDL design description.

A highly complex and detailed design can mislead you to overlook important details. Systematically partitioning the design into different subsystems and expressing the behavior of each subsystem in a higher level language helps to manage a big and detailed design. Such a specification is in a soft level, which can easily be changed, instead of in a hard level, which cannot be easily changed. This specification is unambiguous and complete and can hide the details of implementation.

Systems created with high-level constructs can be functionally simulated. Simulating the design at this level reveals problems before committing the design to hardware. An efficient HDL description accompanied by a sufficient test plan and host simulator that exercises the behavior of the design can minimize the number of design errors and reduce the time-to-market. Such a simulated behavioral description can later be synthesized for detailed gate-level implementations.

A behavioral description is also the best way to document a design. A well-prepared and documented HDL description can always describe a design better than a set of schematics with many gate-level details.

Why Use Verilog?

Some HDL languages, such as Verilog, resemble the C programming language very closely. Adopting the Verilog HDL-Synthesis approach can provide many benefits. Verilog HDL language is readable, elegant, and easy to learn. You only need to learn one language to create functional models, generate the design, create simulation vectors, simulate, and perform any other aspects of logic design. Verilog allows a design to be described in behavioral or structural terms or a mixture of both.

The detailed implementation of a Verilog code may be left to various synthesis tools.

Benefits of HDL Designs for the Actel Devices

Using a targeted vendor-specific library, most synthesis tools such as Synopsys, Cadence, Mentor Graphics®, and Auto Logic, can generate a gate-level representation of a design described in Verilog HDL.

Synthesis tools like the Synopsys Design Compiler are designed to optimize logic into regular conventional architectures. The Actel devices, with their highly regular channeled array architecture, can be used very efficiently with most existing synthesis tools. Synthesizing a design using ACT™ 1, ACT 2, or ACT 3 libraries can be optimized with respect to the speed, circuit size, or other cost functions where that gate utilization and performance are maximized.

Verilog/Synopsys Design Example

Complex designs are best suited for behavioral descriptions. Such a design can then be synthesized for the Actel devices using a synthesis tool such as Synopsys.

A good candidate for the HDL/Synthesis methodology is a DRAM controller. In such a design, many events, such as reset, refresh, and memory access, can happen at any time or simultaneously. Managing these events and arbitrating among them in a structural level can be mind boggling and error prone.

The following is an example of such a design.

Figure 1 demonstrates a typical design flow using Synopsys and the Cadence Actel kit. The following are the steps to create a design for an Actel device.

Step 1—Core Description

Most Verilog logic descriptions begin with a behavioral description of the core logic. The core logic description is found in the top level of hierarchy that contains the I/O and clock buffers.

Example 1 is the core logic captured in Verilog HDL for a DRAM controller. It controls the refresh, memory access, reset cycles, and address mapping for 4 MB of dynamic memory. The physical addressing space of this memory can be programmed to be mapped in any of 16 possible banks and designated with the top address bits (A22 to A25). Figure 2 is a block diagram description of this design.

A typical access from a processor is either I/O or memory. In this example, there are three control state machines that

implement the I/O and the memory cycles. Figure 3 is their state diagram representation.

These state machines are directed with parallel_case Full _case Synopsys directives. These directives cause all the states to be evaluated in parallel and create all possible states that are not covered. The main state machine is gray-coded to achieve the minimal required logic decoding.

The initial I/O access will set the upper and the lower address boundaries of the memory bank. Any memory access to this DRAM should be in this addressing space.

The reset, memory, and refresh cycles are prioritized. A valid address sent to the memory will start a Read or Write cycle. The RAS, CAS, address switch, and other handshake lines will be generated accordingly. If no other memory access is in progress, a refresh request will start a CAS cycle before a RAS cycle. If some other cycle is in progress, it will be latched to be granted at a later time. A reset always has the highest priority; it resets all the cycles to their initial states.

Step 2—I/O Buffers

I/O buffers must be either instantiated in the Verilog code or automatically inserted by Synopsys. Synopsys V3.0 contains a command, insert_pads, that automatically inserts INBUF, OUTBUF, and CLKBUF. If the design must contain BIBUF, TRIBUF, or any special ACT 3 I/O cell, they must be hand instantiated in the code. Example 2 shows how to hand instantiate I/O buffers in the Verilog code. A level of logic, or "top level," is created to merge the core logic with the I/O buffers.

Step 3—Test Circuit

A Verilog test file needs to be generated to Simulate the design and verify behavioral level functionality. Example 3 demonstrates this test circuit.

Step 4—Invoke the Synopsys Tool

The Synopsys setup file (.Synopsys_dc.setup) needs to be set for appropriate Actel devices. The EDIF options need to be set correctly to generate an EDIF netlist appropriate for ALS. Following is an example of a desirable .synopsys file.

Example of .Synopsys file

```
search_path = search_path +
{/rel/syn/act2/rel/syn/dwact}
designer ="ashena";
company = "Actel";
synthetic_library = dwact.sldb
/* link_library = act_31.db; */
link_library = act_31.db;
define_design_lib DWACT -path /rel/syn/act2/dwact
define_design_lib ACT =path /rel/syn/act2
symbol library = /rel/syn/act2/actsym.sdb;
read act_31.db;
edifout_no_array = true
edifout_power_and_ground_representation = cell
edifout_ground_name = GND
edifout_power_name = VCC
edifout_ground_pin_name = y
edifout_power_pin_name = x
edifout_pertty_print = true
read_array_naming_style = "%s<%d>"
```

The Verilog code should be read in the Synopsys environment for it to synthesize and optimize the behavioral circuit. The optimization level can be selected to be high, medium, or low.

Synopsys can write out an EDIF file as well as a Verilog structural level output file. The ALS program EDN2ADL will translate the generated EDIF file to ADL (Actel Design Language). The ALS software places and routes the ADL netlist into an Actel field programmable gate array (FPGA).

Synopsys also generates a gate-level schematic for reference or debugging.

Step 5—Simulate the Structural Verilog File

Resimulate the Synopsys generated structural level design to verify the gate-level functionality. The same test file used in step 3 is also used to further simulate the gate-level DRAM Controller design. The two simulator results are compared.

Step 6—Generate an Actel Netlist

Translate design into an Actel netlist using the "edn2adl" command, as follows:

edn2adl fam:<*device_family*> *design name*

Where:

fam	is family, ACT 1, ACT 2, or ACT 3
<design name>	is the name of the design, which must be the same name as the top level of hierarchy, that was synthesized

Step 7—Place and Route the Design

The ALS software verifies the integrity of the design, assigns I/O pin numbers, places, and routes the design, and generates a fuse file for programming. After place and route, the sdfgen program backannotates actual delays for simulation.

Step 8—Run Actel Timer

The Actel timer is a static timing verifier. Unlike simulators, the Timer does not require test vectors. The Timer tool is useful for verifying internal and external setup and hold time requirements, clock skew, and maximum frequency.

To run a command line version of the Timer, from the command line, enter:

```
als -timer <deisgn_name>
```

To run a menu version of Timer, from the command line, enter:

```
alstimer <design_name>
```

Step 9—Post-Route Simulation

To backannotate the intrinsic and routing delays to Verilog, use the "sdfgen" command. This command backannotates scaled delays for temperature, voltage, process and speed grade. These variables are either selected from the EXPORT entry on the main menu, or may be set as an argument when invoking "sdfgen."

From the project directory, type:

```
sdfgen [tempr:<temperature_range>]
       [voltr:<voltage_range>]
       [speed_grade:<speed_grade>]
       <design_name >
```

For example, if you specify ACT 2 family backannotation values of commercial temperature range, commercial voltage range, and the fast -2 speed grade, the sdfgen command would look like this:

```
sdfgen tempr:com voltr:com speed_grade:-2
dram1
```

where "dram1" is the <design_name>.

The "sdfgen" command generates the SDF file (Standard Delay File) for the postlayout simulation session. To perform post layout simulation, add SDF routine command ($sdf_annotate) to the "testfixture.template" file. The syntax for this routine is:

```
$sdf_annotate ("<designs_name>.sdf",
<instance_name>);
```

An example of the "$sdf_annotate" construct is shown Example 3.

For more detailed information regarding the Actel/Cadence/Verilog interface, refer to the following manuals:

- Designer Series for theX Windows System
- Cadence Composer Verilog Environment
- *Getting Started*, Manual part number 5029062-0

For more detailed information regarding the Actel/Synopsys interface, refer to the following manuals:

- Actel/Synopsys Library for the UNIX Environment
- *User's Guide*, manual part number 5029067-0

Example 1. DRAM Controller

```
/*              DRAM CONTROLLER
This state machine controls the memory handshake, refresh, and address decoding for a DRAM */
`timescale 1 ns / 100 ps
module dram1 (ras_,cas_,rw_en,ready_,mux_add,sysclk,add,ads_,reset,rw_,ref,data,m_io);
output ras_,cas_, rw_en, ready_; output [11:2] mux_add;
input sysclk;
input [25:2] add;
input ads_, rw_, ref, reset, m_io;
input [7:0] data;
reg [3:0] low_add;          /* lower address bytes */
reg [3:0] up_add;           /* upper address bytes */
reg [1:0] state;            /* I/O states */
reg [6:0] cs;               /* Current State */
reg [6:0] ns;               /* Next State */
reg rw_en, refreq, lat_ads_; wire mux_sel, hit, ref_end, io_ready_, m_ready_;
parameter s1=7'b1111111, s2=7'b1011111, s3=7'b1011110, s4=7'b1011100,s5=7'b1011000,        s6=7'b1010000,
s7=7'b1101011, s8=7'b1101010,s9=7'b1001010, s10=7'b0001111;
/* ---------------------------ASSIGNMENTS---------------------------------------------*/
/* SET the memory and refresh handshake signals */
assign ras_ = cs[0]; assign mux_sel = cs[1]; assign cas_ = cs[2]; assign m_ready_ = cs[3]; assign ref_end = !cs[6];
/* RAS address or CAS address */
assign mux_add = mux_sel ? add[11:2] : add[21:12];
/* Is the address in the boundary? */
assign hit = (add[25:22] >= low_add[3:0]) && (add[25:22] < up_add[3:0]); assign io_ready_ = !state[1]; /* End of
I/O access */
assign ready_ = (io_ready_ & m_ready_);/* End of I/O or Memory access */
/* I/O ACCESS TO SET THE UPPER AND LOWER ADDRESS BYTES*/
always @ (posedge sysclk)
if (reset) begin
/* Only a subset of Verilog Language is supported by Synopsys */
        state = 2'b00;
        // synopsys translate_off
                fork
        // synopsys translate_on
        up_add [3:0] = #2 4'h0;
        low_add [3:0] = #2 4'h0;
        // synopsys translate_off
                join
        // synopsys translate_on
                end
                else
        case (state)    //synopsys parallel_case full_case
        00 : if (!reset & !m_io & !rw_ & !ads_) state = 2'b01;
        01 : begin
```

Example 1. DRAM Controller (continued)

```
                        // synopsys translate_off
                                fork
                        // synopsys translate_on
                                low_add [3:0] = #2 data[3:0];
                                up_add [3:0] = #2 data[7:4];
                        // synopsys translate_off
                                join
                        // synopsys translate_on
                                state = 2'b10;
        end
        10 : state = 00;
                        default : state = 2'b00;
                        endcase
/*      MEMORY and REFRESH ACCESS */
always @(posedge sysclk) cs = #3 ns;
always @(ads_ or reset or refreq or cs or rw_ or hit or m_io) begin if (reset) begin
        ns=s1;
        #2 rw_en = 1;
        end else
        case (cs)                       //synopsys parallel_case full_case
        s1: begin
                        rw_en = 1;
                        if (refreq) ns = s7;
        /* memory cycle starts */
                        else if ((!ads_ | !lat_ads_) & !refreq & m_io ) ns=s2;
                        else ns=s1;
                        end
        s2: begin
                        if (hit) ns=s3;/* Address is in the boundary */
                        else ns=s1;
                        end
        s3: ns=s4;
        s4: begin
                        ns=s5;
                        if (!rw_) #2 rw_en = 0;
                        else #2 rw_en = 1;
                        end
        s5 : ns=s6;
        s6 : ns=s1;
        s7 : ns = s8;
        s8 : ns = s9;
        s9 : ns = s10;
        s10: ns = s1;
        default: ns = s1;
        endcase end
```

Example 1. DRAM Controller (continued)

```
always @ (posedge sysclk) if (reset) refreq = 0;
                else if (ref & !ref_end) refreq = 1;
                else if (ref_end) refreq = 0;
always @ (posedge sysclk)
if (reset) lat_ads_ = 1;
                else if (refreq & !ads_ & m_io) lat_ads_ = 0;
                else if (!m_ready_) lat_ads_ = 1;
endmodule
```

Example 2. Top-Level Design

```
`timescale 1 ns / 100 ps
module topd (pras_,pcas_,prw_en,pready_,pmux_add,psysclk,padd, pads_,preset,prw_,pref,pdata,pm_io);
input psysclk;
input [25:2] padd;
input pads_, preset, prw_, pref, pm_io;
input [7:0] pdata; output pras_, pcas_, prw_en, pready_;
output [11:2] pmux_add;
wire sysclk;
wire ras_, cas_, rw_en, ready_;
wire [11:2] mux_add;
wire [25:2] add; wire ads_, reset, rw_, ref, m_io; wire [7:0] data;
/* Instantiate the Core logic */
dram1 u1 ( ras_, cas_, rw_en, ready_, mux_add, sysclk, add, ads_, reset, rw_,ref, data, m_io);
/* Connect the I/O and Clock buffers */
CLKBUF UC1 (.PAD(psysclk), .Y(sysclk));
OUTBUF UO0 (.PAD(pras_), .D(ras_));
OUTBUF UO1 (.PAD(pcas_), .D(cas_));
OUTBUF UO2 (.PAD(prw_en), .D(rw_en));
OUTBUF UO3 (.PAD(pready_), .D(ready_));
OUTBUF UO5 (.PAD(pmux_add[2]), .D(mux_add[2]));
OUTBUF UO6 (.PAD(pmux_add[3]), .D(mux_add[3]));
OUTBUF UO7 (.PAD(pmux_add[4]), .D(mux_add[4]));
OUTBUF UO8 (.PAD(pmux_add[5]), .D(mux_add[5]));
OUTBUF UO9 (.PAD(pmux_add[6]), .D(mux_add[6]));
OUTBUF UO11 (.PAD(pmux_add[7]), .D(mux_add[7]));
OUTBUF UO12 (.PAD(pmux_add[8]), .D(mux_add[8]));
OUTBUF UO13 (.PAD(pmux_add[9]), .D(mux_add[9]));
OUTBUF UO14 (.PAD(pmux_add[10]), .D(mux_add[10]));
OUTBUF UO15 (.PAD(pmux_add[11]), .D(mux_add[11]));
INBUF UI0 (.PAD(padd[2]), .Y(add[2]));
INBUF UI1 (.PAD(padd[3]), .Y(add[3]));
INBUF UI2 (.PAD(padd[4]), .Y(add[4]));
INBUF UI3 (.PAD(padd[5]), .Y(add[5]));
INBUF UI4 (.PAD(padd[6]), .Y(add[6]));
INBUF UI5 (.PAD(padd[7]), .Y(add[7]));
```

Example 2. Top-Level Design (continued)

```
INBUF UI6 (.PAD(padd[8]), .Y(add[8]));
INBUF UI7 (.PAD(padd[9]), .Y(add[9]));
INBUF UI8 (.PAD(padd[10]), .Y(add[10]));
INBUF UI9 (.PAD(padd[11]), .Y(add[11]));
INBUF UI10 (.PAD(padd[12]), .Y(add[12]));
INBUF UI11 (.PAD(padd[13]), .Y(add[13]));
INBUF UI12 (.PAD(padd[14]), .Y(add[14]));
INBUF UI13 (.PAD(padd[15]), .Y(add[15]));
INBUF UI14 (.PAD(padd[16]), .Y(add[16]));
INBUF UI15 (.PAD(padd[17]), .Y(add[17]));
INBUF UI16 (.PAD(padd[18]), .Y(add[18]));
INBUF UI17 (.PAD(padd[19]), .Y(add[19]));
INBUF UI18 (.PAD(padd[20]), .Y(add[20]));
INBUF UI19 (.PAD(padd[21]), .Y(add[21]));
INBUF UI20 (.PAD(padd[22]), .Y(add[22]));
INBUF UI21 (.PAD(padd[23]), .Y(add[23]));
INBUF UI22 (.PAD(padd[24]), .Y(add[24]));
INBUF UI23 (.PAD(padd[25]), .Y(add[25]));
INBUF UI24 (.PAD(pads_), .Y(ads_));
INBUF UI25 (.PAD(preset), .Y(reset));
INBUF UI26 (.PAD(pref), .Y(ref));
INBUF UI27 (.PAD(prw_), .Y(rw_));
INBUF UI28 (.PAD(pdata[0]), .Y(data[0]));
INBUF UI29 (.PAD(pdata[1]), .Y(data[1]));
INBUF UI30 (.PAD(pdata[2]), .Y(data[2]));
INBUF UI31 (.PAD(pdata[3]), .Y(data[3]));
INBUF UI32 (.PAD(pdata[4]), .Y(data[4]));
INBUF UI33 (.PAD(pdata[5]), .Y(data[5]));
INBUF UI34 (.PAD(pdata[6]), .Y(data[6]));
INBUF UI35 (.PAD(pdata[7]), .Y(data[7]));
INBUF UI36 (.PAD(pm_io), .Y(m_io));
endmodule
```

Example 3. An Example of Test Design

```
module test_d;
reg [25:2] add; reg sysclk; reg ads_; reg reset; reg rw_; reg ref; reg [7:0] data; reg m_io;
wire ras_,cas_; wire rw_en; wire ready_; wire [11:2] mux_add; reg [4:0] count; reg mem_all;
integer i;
reg [2:0] state; parameter a0= 3'b000 , a1=3'b001 ,a2 = 3'b010 , a3=3'b011, a4= 3'b100 , a5=3'b101 ,a6 = 3'b110;
dram1 u0 (ras_,cas_,rw_en,ready_,mux_add,sysclk,add,ads_,reset,rw_,ref,dat a,m_io);
initial        begin
$gr_waves("clock%b",sysclk,"ads",ads_,"reset%b",reset,"m_io",m_io,"state%b",u0.state[1:0],"mem_all",mem_all,"rw_",
rw_,"low_add%h",u0.low_add[3:0],"up_add%h",u0.up_add[3:0],"io_ready_",u0.io_ready_,"hit",u0.hit,"ready_",ready
_,"addup%h",u0.add[25:22]                    ,"ras%b",ras_,"rowadd%h",add[11:2],"coladd%h",add[21:12],"cas%b",
cas_,"read_write%b",rw_en, "mux_sel", u0.mux_sel, "cs", u0.cs, "ns", u0.ns, "state", state, "add", add, "ref", ref,
"count", count, "refreq", u0.refreq, "ref_end", u0.ref_end);
               end
initial        begin
#10;           sysclk = 0;    count = 0;    ref = 0;        rw_ = 1;        reset=0;
add [25:2]= 0;ads_ = 1;       mem_all = 0; #10 m_io = 0;
#20 reset=1;  #60 reset = 0;
@ (posedge sysclk) if (!mem_all & !reset) begin
               fork
                              #2 ads_ = 0;  #2 m_io = 0;
                              #2 rw_ = 0;   #3 data [7:0] = 8'b00010000;
               join
                              end
@ (posedge sysclk) #10 ads_ = 1;
wait (!ready_)#2 m_io =1;
#4 ads_ =1;    #5 rw_ = 1;    #30 mem_all = 1; end
always @ (posedge sysclk)begin
if (reset) state = a0;
else
case (state)
a0: if (mem_all) state = a1; a1: begin
#2 ads_=0;
#2 add[25:22] = 4'b0000;
#2 rw_ = 0;
#2 state = a2; end
a2 : begin
#2 ads_ = 1;
#2 state = a3; end
a3 : if (!ready_) #2 state = a4;else #2 state = a3;
a4: begin
#2 ads_=0;
#2 rw_ = 1;
#2 state = a5; end
a5 : begin
#2 ads_ = 1;
#2 state = a6; end
```

Example 3. An Example of Test Design (continued)

```
a6 : if (!ready_) begin
                #2 state = a0;
                #2 add[21:02] = add[21:02] + 1;
                end             else #2 state = a6;
default : #2 state = a0;
endcase
end
always @(posedge sysclk)
if (reset) #2 ref = 0;
else if (count == 5'h0f) #2 ref = 1;
else #2 ref = 0;
always @(ref | count==0 ) if (reset) count = 0;
else for (i= 0; i<20; i=i+1) count = #70 count + 1;
initial begin
$monitor ($time,"%b %b %b %h %h %b %b %b" ,sysclk,add[25:22],ras_,add[11:2],add[21:12],cas_,rw_en,ready_);
end
always forever #30 sysclk = ~sysclk;
initial
$sdf_annotate("dram1.sdf",uc);
endmodule
```

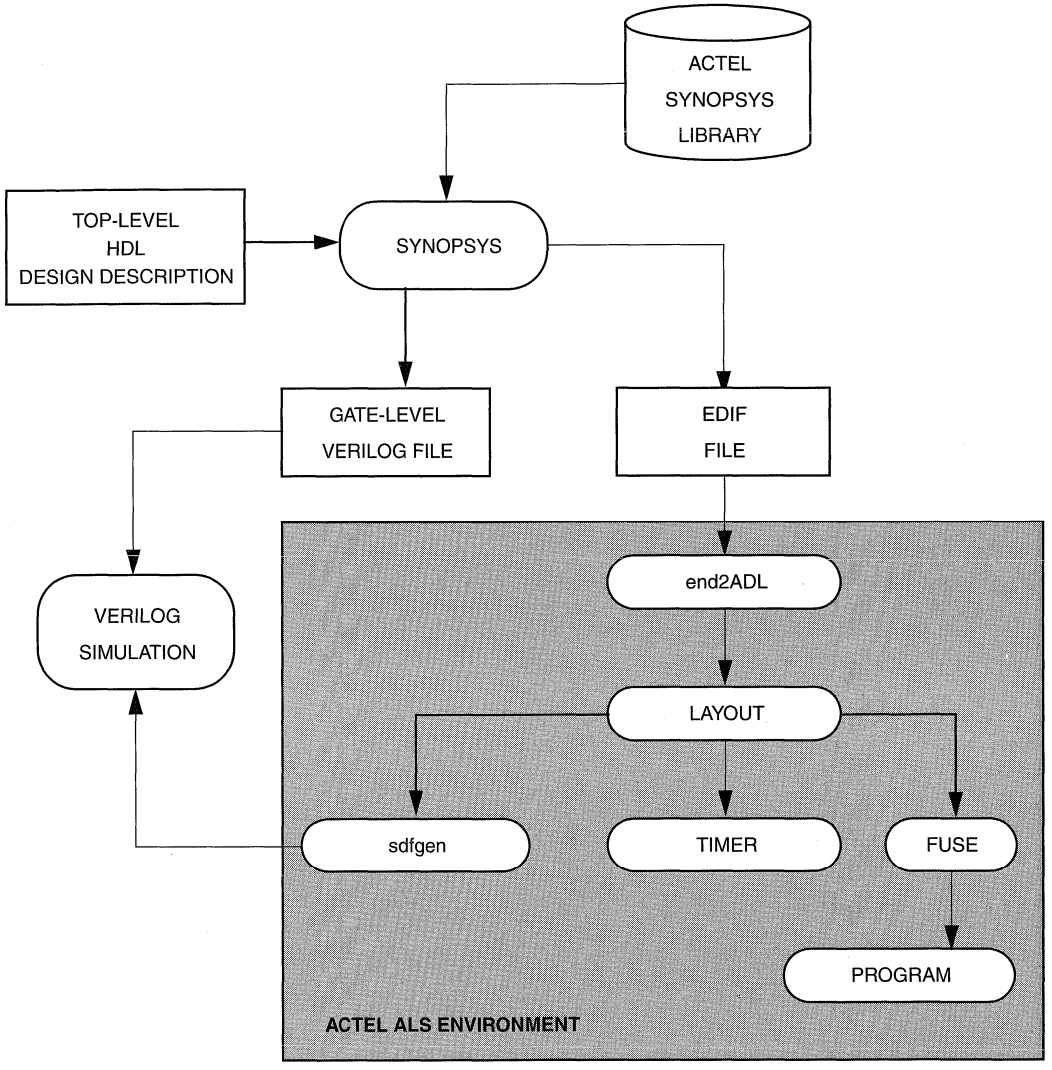


Figure 1 • *Typical Design Flow Using SynopsYs and Cadence*

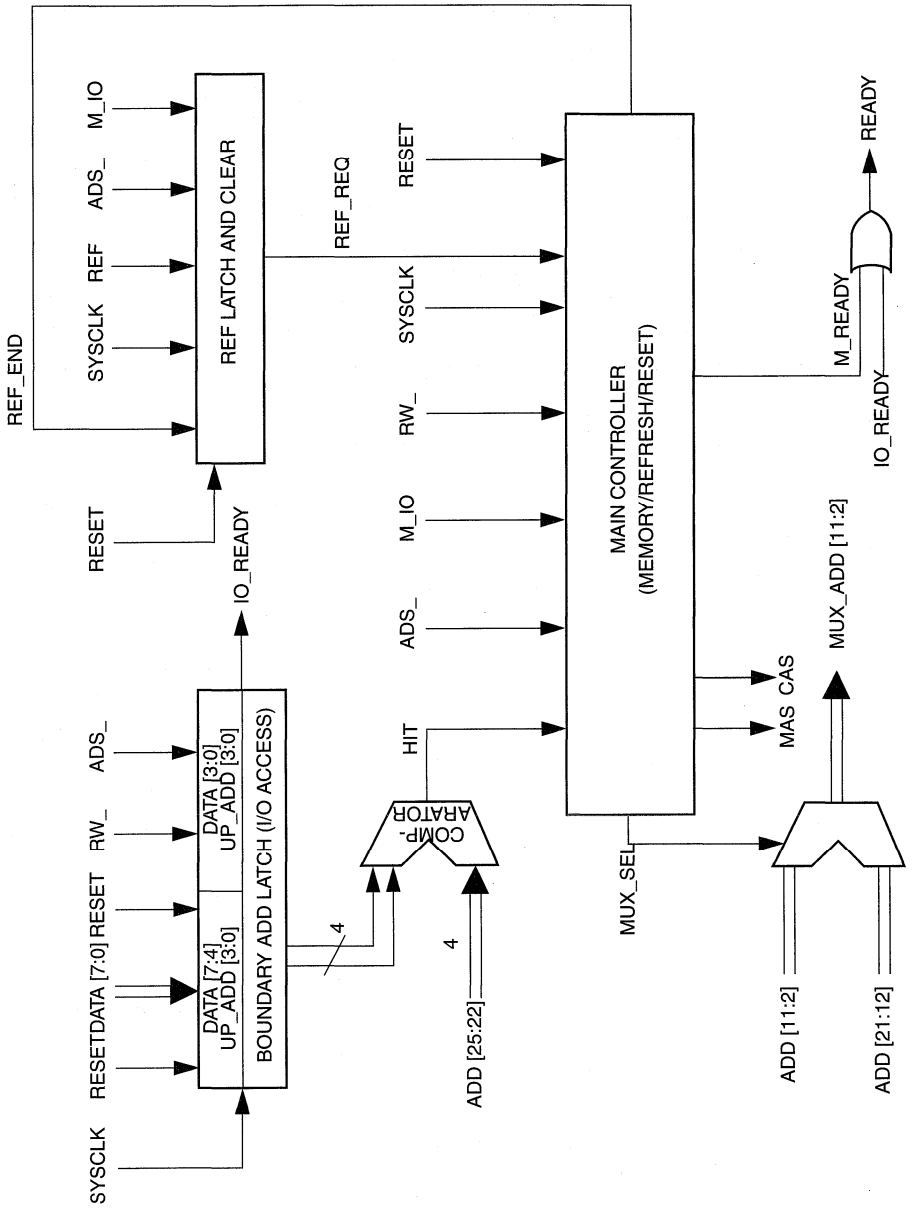


Figure 2 • Block Diagram Description for a DRAM Controller

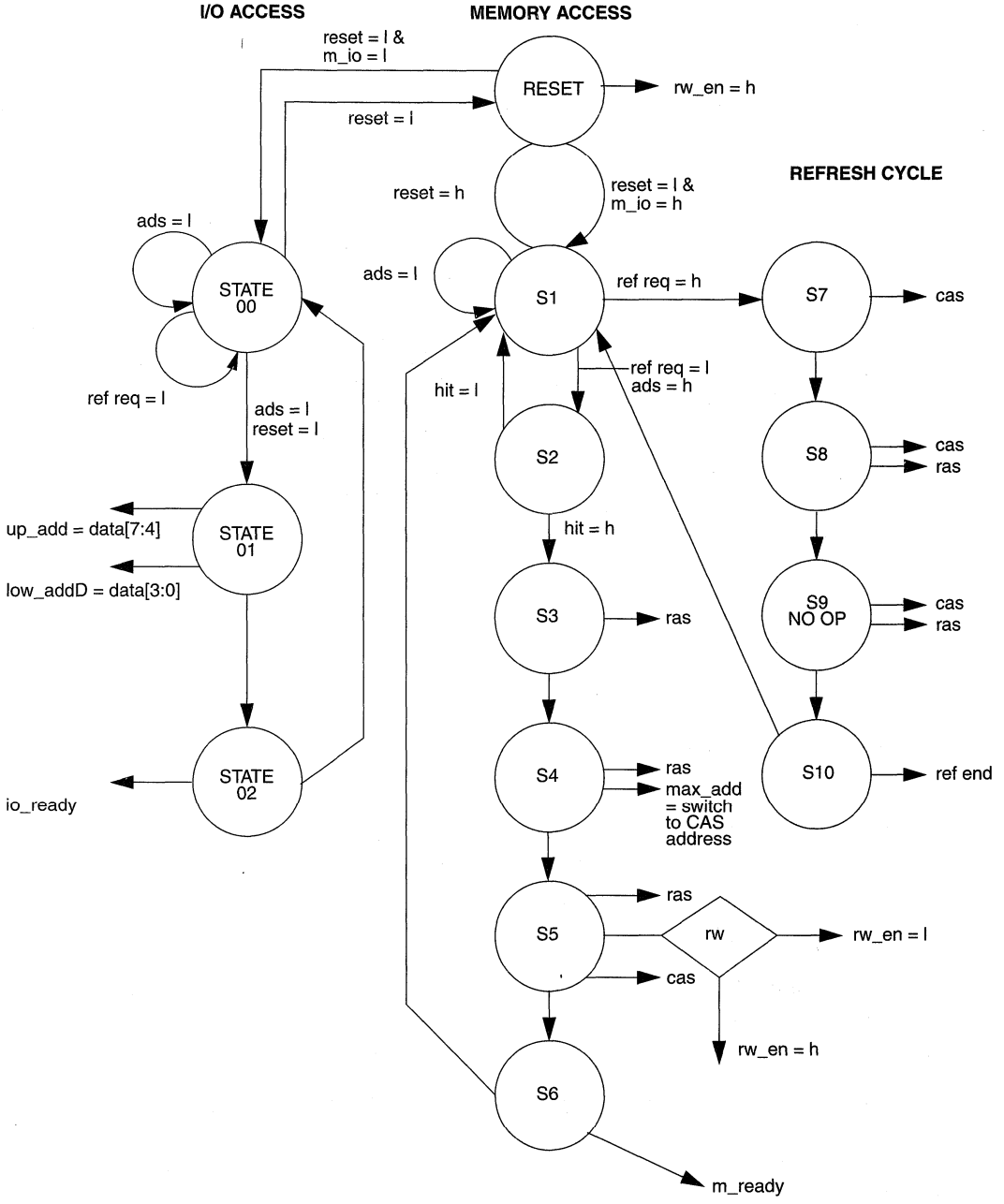


Figure 3 • Memory Access

Application Examples and Design Techniques

Section 9: Application Examples and Design Techniques

**Available on the Actel bulletin board and the Web. Contact Actel Technical Support Hotline 1-800-262-1060.*

Using Actel FPGAs to Implement the 100 Mbit/s Ethernet Standard

One of the more recent entrants into the high-speed networking standards battle is 100Base-X—Ethernet operating at 100 Mbit/s. This standard is supported by the Fast Ethernet Alliance and sponsored by several key networking companies such as Intel, National Semiconductor, Sun Microsystems, and 3Com. This proposed standard involves many of the types of digital logic functions facing high-speed network designers and, as will be shown in this application note, can be readily implemented using Actel FPGA devices.

The emerging 100 Mbit Ethernet market is expected to mushroom as network performance requirements continue to grow. Network users are expected to have almost doubled between 1991 and 1994, and networks will need to provide these new users with just as much (if not more) bandwidth. A high-speed Ethernet network could solve the bandwidth problems for many classes of users while maintaining compatibility with current equipment and software.

100Base-X Network Standards

The 100Base-X proposal uses two established networking standards to support the 100 Mbit data rate required to implement the tenfold increase in the 10 Mbit rate of the current Ethernet standard. The 100Base-X standard keeps the Media Access Control (MAC) layer the same as the current Ethernet standard, but it raises the data rate to 100 Mbit/s. Since the MAC layer was defined independently of performance level, this increase can be accomplished relatively easily, and the well-proven behavioral dynamics of the Ethernet MAC can be retained. The only change required is to reduce the physical network span to 1/10 of the 10 Mbit/s distance, resulting in a span of about 250 meters.

This reduced span fits well within current structured wiring methodologies. Building-floor wiring in modern installations of Ethernet, such as 10Base-T, are organized as physical stars with a centralized wiring closet and cable runs of less than 100 meters. For LANs, this results in a hub-station architecture with interconnections of less than 100 meters.

At the physical layer, 100Base-X leverages off the proven FDDI standard for 100 Mbit/s communications using a full-duplex 125 Mbit/s Physical Media-Dependent (PMD) sublayer. This supports fiber optic, shielded twisted-pair (STP) and unshielded twisted-pair (UTP) wiring. Combining the MAC layer of Ethernet to the PMD layer of FDDI requires a convergence sublayer (CS) between them. Using the CS, 100Base-X maps the PMD's constant signaling system to the packet-oriented half-duplex system imposed by the Ethernet MAC.

Convergence Sublayer Interfaces

The MAC transmits data to the convergence sublayer in the form of 4-bit words (Figure 1). This data is then encoded into 5-bit groups, serialized, and transmitted by the CS to the PMD sublayer as the transmitPMD signal.

Received data is sent from the PMD to the CS as the receivePMD signal and is synchronized with the 125 MHz clock. Note that the PMD also generates signalDetect when data is detected on the line. The CS decodes the serial data, converting the input 5-bit code groups into 4-bit hex characters and sends it to the MAC as the receiveMAC signal. Note that the PMD extracts the clock from the serial bit stream input. The 125 MHz frequency is recovered from the input data stream by the PMD clock circuits in the CS. In addition, receiveError is generated by the CS to indicate to the MAC that an error has occurred during reception. The carrierSense signal is provided to the MAC to indicate that the line is active. The collisionDetect signal notifies the MAC if a collision has occurred.

This application note will show you how to use Actel FPGAs to develop a complete convergence sublayer. It will subdivide the CS into its functional divisions and will show you how each can be implemented using Actel ACT 3 FPGAs.

Convergence Sublayer Functions

Figure 2 shows the basic dataflow in the convergence sub-layer. The CS receives transmit data from the MAC as 4-bit words designated transmitMAC. These 4-bit words are encoded into 5-bit symbols (designated TxSYM) that are shifted out to the PMD at the 125 MHz clock rate.

Received data at a 125 Mbit/s rate is sent from the PMD to the CS as the receivePMD signal. The CS formats input data to produce 5-bit symbol groups. Detection of the two-symbol sequence, J and K, marks the beginning of a packet and starts the synchronization of the input data stream. The 5-bit groups are then decoded by the 5B4B decoder and sent to the MAC as a stream of 4-bit words until the packet's end is detected by the reception of the end-of-packet delimiter characters, T and R.

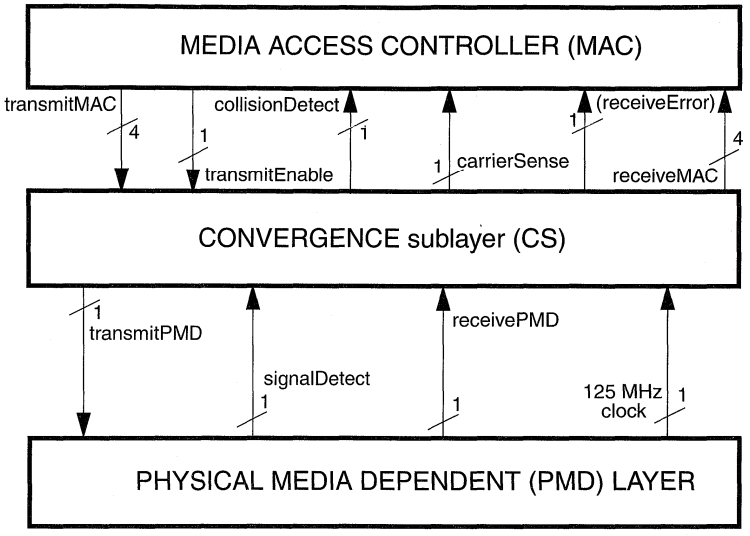


Figure 1 • Convergence Sublayer Interfaces

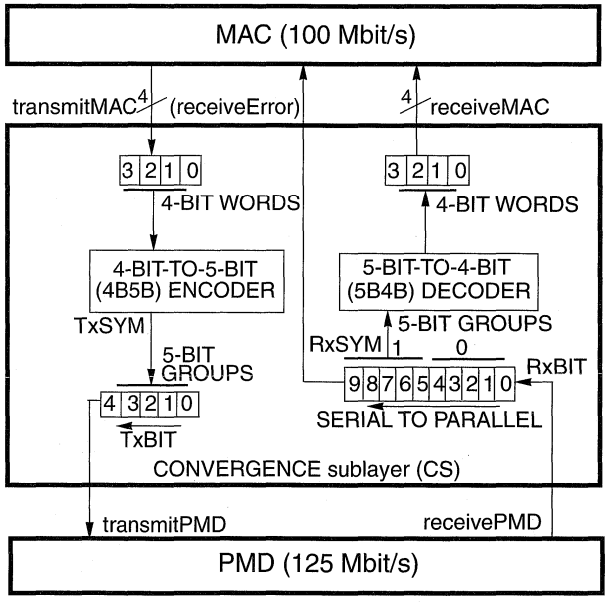


Figure 2 • Data Flow in Convergence Sublayer

The 4B5B encoding/decoding method, which is a subset of the standard FDDI 4B5B encoding method, employs 5-bits to encode/decode both 16 data (hex) characters and the signaling symbols required to indicate the start and end of the data packet. In addition to the 16 valid 4-bit-binary code groups shown in the Table 1, there are five special control signals used to indicate start of packet (J followed by K), end of packet (T followed by R), and idle (I). A number of other 5-bit combinations are designated as invalid and represent channel errors or repeater collision artifacts. Thus, the physical line idles until the start of a data packet is indicated by a J symbol followed by a K. Data symbols then follow with the end of data being indicated by a T symbol followed by an R. Idle symbols immediately follow. The job of the convergence sublayer is to extract the control characters or idles from the packet and then send a data-only packet to the MAC. Thus, the MAC never receives idle, JK, or TR symbols. When receiving, the CS reverses the process, encapsulating and encoding the data from the MAC for transmission by the PMD.

Convergence Sublayer Data Flow

The block diagram of the convergence sublayer is shown in Figure 2. The receive state machine generates receiveMAC data and receiveError for the MAC based on the receivePMD data input from the media. The transmit state machine accepts transmitMAC and transmitEnable from the MAC and generates the transmitPMD data to the physical layer. The collisionDetect function is generated by the transmit state machine, based on transmitEnable and the receive state machine's receiving signal.

The Carrier Sense function asserts the carrierSense signal when the convergence sublayer is either transmitting or receiving, based upon the two corresponding internal signals generated by the Transmit and Receive functions. The Link Monitor function generates linkTestFail based on the PMDs signalDetect. LinkTestFail is an internal signal unused by the MAC and can optionally be used by your network management entity.

Transmitted data, shown as the transmitMAC signal in Figure 3, indicates that MAC data is available and is registered in the convergence sublayer logic. Groups of 4 data bits in the transmit bit stream are converted to 5-bit code groups by 4B5B encoding prior to transmission on the 125 Mbit/s PMD. Note that TxDATA, TxSYM, and TxBIT are all different views of the same data, but at different data rates.

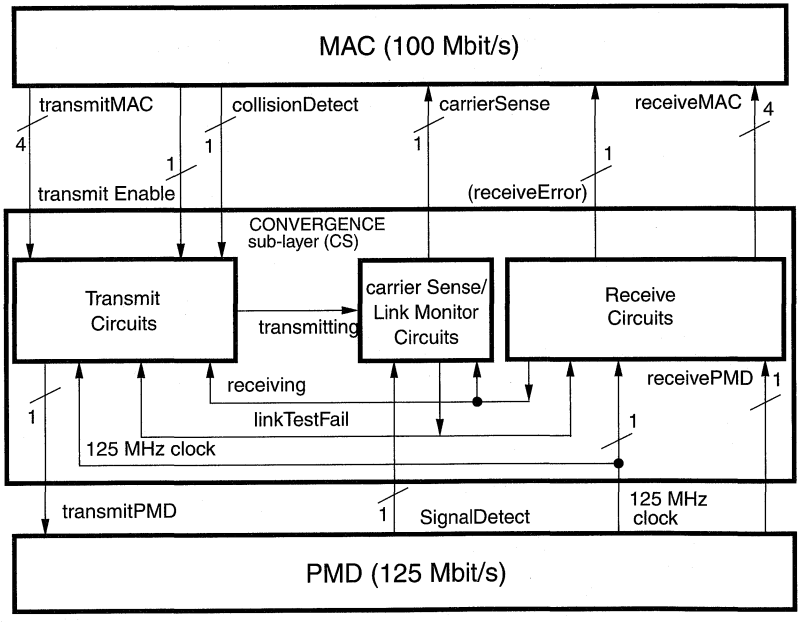


Figure 3 • Convergence Sublayer Functional Block Diagram

Using FPGAs to Implement a 100Base-X Convergence Sublayer

As will be seen in the following sections, the 100Base-X convergence sublayer can be implemented as eight functional blocks, each of which forms the subject of a separate application discussion. These are listed under the three main headings: the transmit function, the receive function, and the carrier-sense and link-monitor circuits.

Convergence Sublayer Transmit Function

The design of the transmit function shows some common design techniques used in high-speed FPGA applications. The main function of this block is to provide the requested symbol data to the PMD at the 125 Mbit/s serial rate. This requires the 4-bit MAC data words to be 4B5B encoded and then shifted out using the 125 Mhz clock. In addition, the serial data stream needs to be framed using the leading /J/K/ symbol pair and trailing /T/R/ symbol pair. When data is not transmitting, it is replaced by the constant transmission of idle symbols. The transmit function is divided into two blocks as shown in Figure 4:

- The Transmit State Machine
- The Data Path

Data Path

The data path portion of the transmit block is shown in Figure 4. The main flow of data comes from the MAC at 25 MHz with a 4-bit-wide data word and a control signal that enables transmission. When MAC data is not being transmitted, the convergence sublayer sends continuous idle (I) symbols to the PMD. When the TransmitEnable signal becomes active, a /J/K/ symbol pair is transmitted to indicate the beginning of a data packet. MAC data symbols then follow and are encoded into 5-bit symbols using the 4B5B encoding scheme shown in Table 1. The end of MAC data is indicated by the TransmitEnable signal going inactive and a /T/R/ symbol pair is inserted at the end of the data packet. Finally, the CS logic returns to transmitting idle symbols.

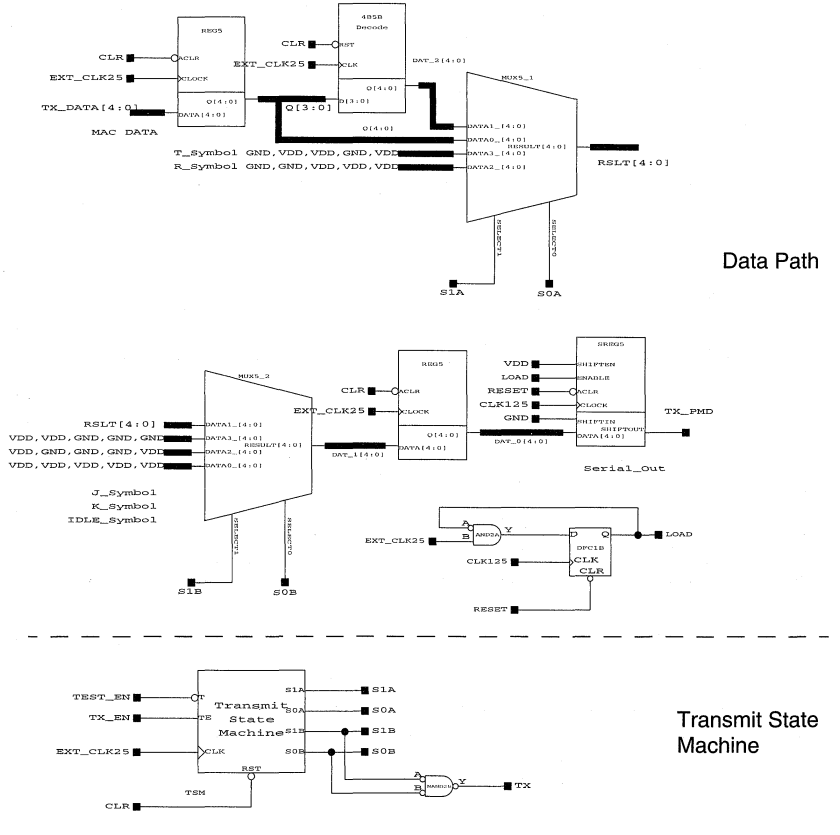


Figure 4 • Convergence sublayer Transmit Functions

Table 1 • 4B5B Symbol Coding

Symbol	5-bit Code Group (in Convergence sublayer)	4-Bit-Binary Code Group (in MAC)	Interpretation/Function
0	11110	0000	Data character: 0H
1	01001	0001	Data character: 1H
2	10100	0010	Data character: 2H
3	10101	0011	Data character: 3H
4	01010	0100	Data character: 4H
5	01011	0101	Data character: 5H
6	01110	0110	Data character: 6H
7	01111	0111	Data character: 7H
8	10010	1000	Data character: 8H
9	10011	1001	Data character: 9H
A	10110	1010	Data character: AH
B	10111	1011	Data character: BH
C	11010	1100	Data character: CH
D	11011	1101	Data character: DH
E	11100	1110	Data character: EH
F	11101	1111	Data character: FH
I	11111	--	Idle character transmitted between packets
J	11000	--	First control character in start-of-packet delimiter
K	10001	--	Second control character in start-of-packet delimiter
T	01101	--	First control character in end-of-packet delimiter
R	00111	--	Second control character in end-of-packet delimiter
V	00000	--	Invalid character
V	00001	--	Invalid character
V	00010	--	Invalid character
V	00011	--	Invalid character
V	00100	--	Invalid character
V	00101	--	Invalid character
V	00110	--	Invalid character
V	01000	--	Invalid character
V	01100	--	Invalid character
V	10000	--	Invalid character
V	11001	--	Invalid character

The implementation of the described functions involves selecting six different symbol sources for PMD data: the I (idle), J, K, T, and R symbols and the 4B5B encoded MAC data. In addition, a TestData input can be used to provide raw unencoded data to the PMD for use in diagnostics and testing. This selection is accomplished via the multiplexer in front of the output shift register. One multiplexer selects from the I, J and K symbols or from another multiplexer output. The other multiplexer selects from the T and R symbols and encoded the MAC data. Note the additional path around the encoder, which allows raw (unencoded) data to be provided to the PMD. This is used for system test and diagnostics and is the only way to inject known errors into the system, simulating collision remnants and exercising the boundary conditions of the standard.

The Actel logic implements multiplexers directly in a single logic module so, by inspection, the path through the multiplexer tree requires only two module delays and can easily meet the 25 Mhz performance requirement. The 4B5B encode block also requires only two logic levelsand can be

designed via schematics or via equations. The equations can be automatically compiled using Actel's ACTmap tool and then incorporated into the schematic.

4B5B Encoder

Symbol encoding of the 4-bit data words transmitted from the MAC into the 5-bit coded groups required by the convergence sub-layer and the PHY layer employs a modified version of the coding used in FDDI-based systems. The differences from FDDI are that the symbols S, Q, and H are not used and that R is now used as part of the /T/R/ end-of-packet delimiter character group.

Table 1 lists all 32 5-bit data- and special-symbol codes that the PMD can send to the convergence sublayer. The 16 data characters—0 through F (hex)—are shown in Table 1, both as 5-bit code groups and as their 4-bit binary equivalents, as sent by the CS to the MAC. The idle character I and the control characters J, K, T, and R are shown in Table 1 in 5-bit form only, because they are not used in the MAC. The same applies to the remaining 11 possible 5-bit combinations that might be received on the media, all of which have no meaning to the decoder and hence are treated as invalid. For simplicity, each of the 11 invalid symbols is designated as V.

Encoding of 4-bit data words into 5-bit symbols can be accomplished in a few simple logic equations, as shown in the PALASM entry format shown in Figure 5.

The above equations translate each bit in sequence. Bits D0-D3 are the 4-bit data word input to the decoder, and B0-B4 define the 5-bit output symbols from the decoder. Thus, in the first equation, bit D0 is always the same as the bit B0, as can be seen by inspection of Table 1. The decoding equations for the remaining output bits (bits B4 through B1) are derived in a comparable fashion.

ACTmap VHDL Synthesis and FPGA Optimization

These equations are then processed by ACTmap, a computer-aided design tool for working with the Actel families of FPGAs. It performs three basic functions:

- PALASM 2, VHDL to netlist translation
- Netlist-optimized mapping
- I/O insertion

ACTmap reads the PALASM 2, or VHDL source file and translates it into either an EDIF or an ADL (Actel Design Language) output file or Verilog netlist. The output file that it generates is optimized for a specific family of Actel FPGAs (ACT 1, ACT 2, 1200XL, or ACT 3).

You can specify whether the design should be optimized for area or speed. The PALASM 2 description for the 4B5B encoder shown in Figure 5 was processed by ACTmap, and the following results where achieved:

- Area = 9 modules
- Estimated worst-case delay = 8.80 ns

```
;Encoder for 4B to 5B
;Used in 100 Mbit Ethernet application
CHIP 4b5b generic
clk rst d3 d2 d1 d0 q4 q3 q2 q1 q0

EQUATIONS

q4 := d3 + (/d2 * d1) + (/d2 * /d0)
q3 := d2 + (/d3 * /d1)
q2 := d1 + (/d3 * /d2 * /d0)
q1 := (/d1 * /d0) + (d3 :+: d2) + (d2 * /d1)
q0 := d0

q4.clkf = clk
q3.clkf = clk
q2.clkf = clk
q1.clkf = clk
q0.clkf = clk

q4.rstf = /rst
q3.rstf = /rst
q2.rstf = /rst
q1.rstf = /rst
q0.rstf = /rst
```

Figure 5 • PALASM2 Description for the 4B5B Encoder

These results easily meet the 25MHz requirements of the transmit function and show the speed and capacity capabilities of the ACT3 architecture. The schematic logic implementation of the 4B5B encoder (see) shows the compact nature of the final implementation.

Transmit Operation

Transmit operational states are shown in block diagram form in Figure 7.

The actions shown in Figure 7 are assumed to be instantaneous, although, for simplicity, some time-sequenced events are contained in single states. Unconditional state transitions are unlabeled. Conditional state transitions occur when explicitly shown by the accompanying condition; a state is repeated until some transitional condition is detected. States are atomic in that conditions are evaluated only at the completion of the state's actions. Transitions shown without source states, notably linkTestFail, are evaluated at the completion of every state and take precedence over other transition conditions.

Convergence Sublayer Transmit Operation

The transmit state block diagram begins with the IDLE state. The transmitting and collisionDetect signals are initialized as FALSE and the IDLE symbol is continuously supplied to the PMD. Once the MAC has data to transmit, it asserts transmitEnable and the START state is entered. The transmitting signal is asserted (set to TRUE) to indicate to the Carrier Sense function that data is being transmitted. In addition, collisionDetect is set to the level of the receiving signal. The receiving signal comes from the Receive function; if it is also asserted, a collision has occurred. The waitNibble function synchronizes the MAC data with the PMD clock. The first 8-bits of the MAC preamble are replaced with the /J/K/ symbol pair. If transmitEnable becomes FALSE, the machine makes a transition back to IDLE. If transmitEnable stays asserted, the next state becomes TRANSMIT. During TRANSMIT state, collisionDetect is still set to receiving. The MAC data (TxDATA) is encoded using the 5B4B function, and encoding continues until transmitEnable is disabled. Once transmitEnable is deasserted, the machine makes a transition to the END state. In the END state, transmitting and collisionDetect are both FALSE. The /T/R/ symbols are transmitted to indicate the end of data, and the machine moves to the IDLE state. The assertion of linkTestFail (by the Link Monitor function) causes an immediate transition to the IDLE state and takes precedence over any MAC request.

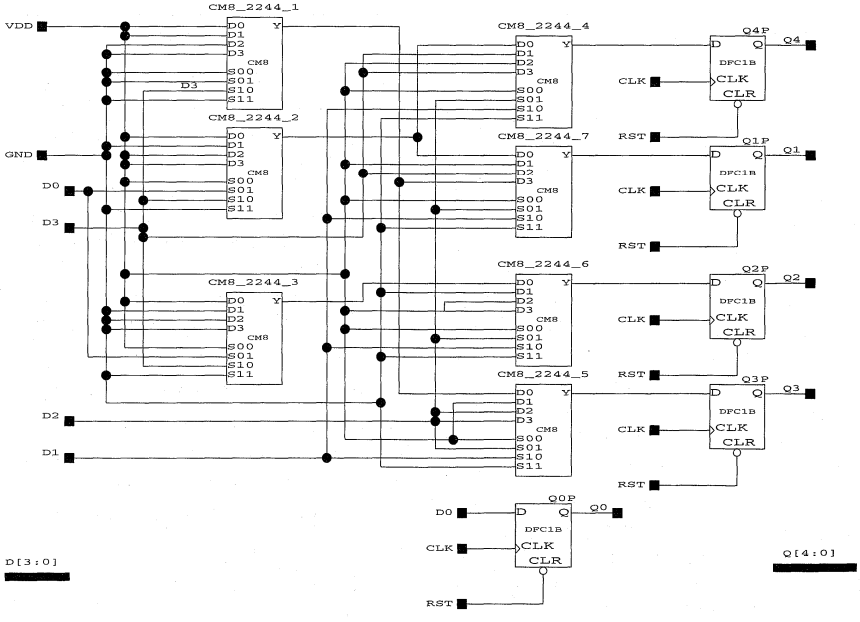


Figure 6 • Transmit Path: 4B5B Encoder FPGA Logic

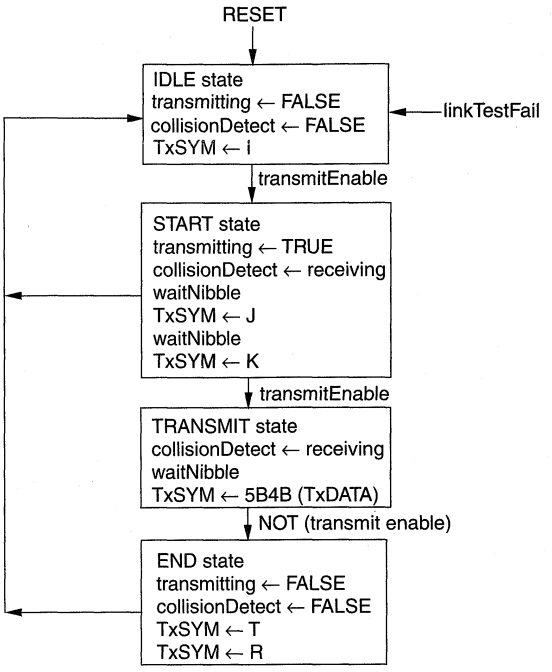


Figure 7 • Convergence Sublayer Transmit Operation

Transmit State Machine

The state-diagram implementation of transmit operation is shown in Figure 8. The state machine starts in the IDLE state and transmits the idle symbol (I) until transmitEnable (TE) is TRUE. As long as TE is TRUE, the machine proceeds through the J and K states, sending first the J symbol and then the K symbol to indicate the start of a data packet. The machine then transmits data until TE goes FALSE (i.e., transmit not enabled (/TE)), after which a T and an R are transmitted, indicating the end of the data packet. The state machine then returns to the IDLE state and waits for the next data packet. The test mode may be entered from the IDLE state by asserting Test mode (TM). In this mode, any 5-bit code symbol may be transmitted, thus allowing known error conditions to be injected onto the network.

The logic implementation of the transmit state machine is shown in Figure 9. Each state is encoded into the transmit state machine flip-flops to allow symbol selection in the transmit multiplexer. (See Note on page 8.) These transitions are controlled by the input logic for each flip-flop and depend only on the TE signal and the current state.

The resulting design employs only 11 logic modules and runs well in excess of the required 25 MHz speed.

Note: This state-machine implementation differs from the commonly seen one-hot approach in that the states are encoded into four D flip-flops rather than a single flip-flop per state. Also, the state-machine encoding shown in this application is more efficient than the one-hot approach because the state flip-flops can drive the data-path multiplexer directly, eliminating the additional encoding logic that would be required to handle the one-hot state variables and to select desired multiplexer sources.

Convergence Sub-layer Receive Function

The receive functions of the convergence sublayer are shown in the block diagram in Figure 10. These functions are discussed in the following sections.

- Shift register, sync detect, and squelch
- Clock generation
- 5B4B symbol decoder
- Receive state machine

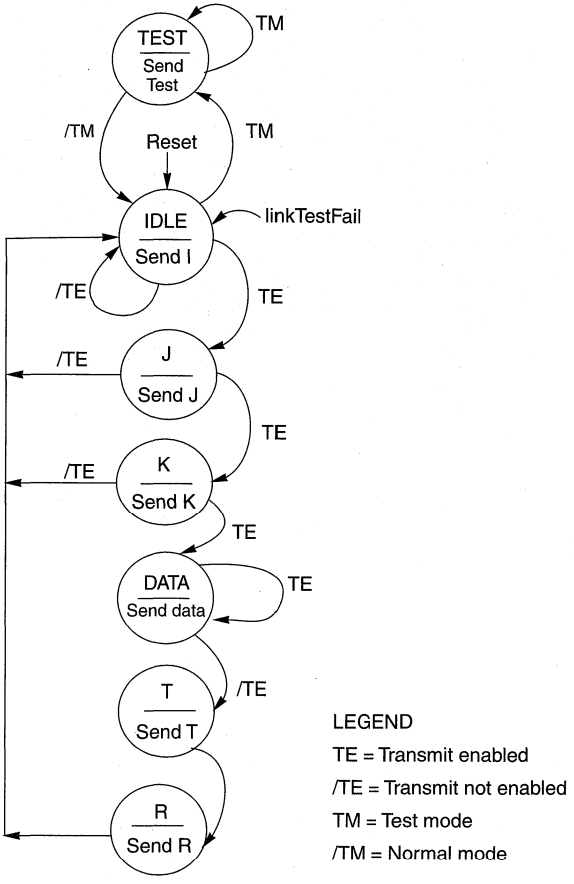


Figure 8 • Transmit State Machine Diagram

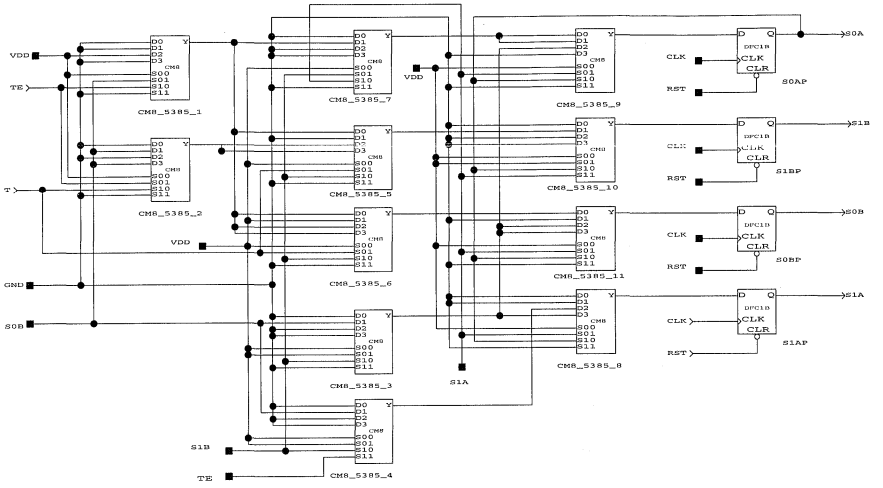


Figure 9 • Transmit State Machine Schematic

Receive Operation

The sequence of receive states is shown in the receive-operation diagram, Figure 11. The receive state machine tracks the received symbols to ensure that a complete packet has been received and indicates the current line state to the next layer of the protocol. The receive process (see Figure 11) involves two separate sets of states. The constituents in the first set—the IDLE, SCAN, CARRIER, and ALIGN states—are prealigned and operate on the raw input bits using RxBIT. The remaining states are aligned and operate on the input data stream as symbols (RxSYM). Output data, designated RxDATA, is sent directly to the MAC in these states.

The RECEIVE state sequence begins with the IDLE state. The receiving signal and the optional receiveError signal are initialized to FALSE. The SCAN state is entered next and the waitBit function synchronizes the machine to the received data stream. At this point, the squelch function filters out noise events by not allowing a transition to the CARRIER state unless two nonconsecutive zeros are detected. Because carrierSense is used by the MAC for deferral purposes, it must be asserted on the detection of any received signal (i.e., received energy, or non-IDLE input) whether or not it's an actual packet. Since carrierSense is also used to detect collisions, it's important to avoid triggering on noise, specifically a single-bit event. If CARRIER is entered, RxDATA is initialized to all zeros (0000) and receiving is set to true.

The system enters the ALIGN state next. In ALIGN, the start of packet symbols /J/K/ is searched for. If at least two idle symbols (1111111111) are found instead, no start of packet has been detected and the machine moves to the IDLE state. If the /J/K/ symbols are successfully found, the START state is entered. In START, the MAC preamble data (55) is substituted for the received /J/K/ symbols. The waitQuint function assures that MAC data is not overwritten. The RECEIVE state is entered next. Usually, in the RECEIVE state, valid data is received and a transition to the DATA state is made. In the DATA state, receiveError is deasserted and 4B5B decoded data is sent to the MAC.

From the DATA state, the machine returns to the RECEIVE state. If, during the RECEIVE state, two idle symbols are received, the PREMATURE END state is entered, receiveError is asserted, and IDLE is reentered. If, in the RECEIVE state, invalid data is received, the DATA ERROR state is entered, receiveError is asserted, and RECEIVE is reentered. Invalid data is not transmitted to the MAC. If, in the RECEIVE state, a /T/R/ symbol pair is detected, the END state is entered, receiving is deasserted, and IDLE state is reentered.

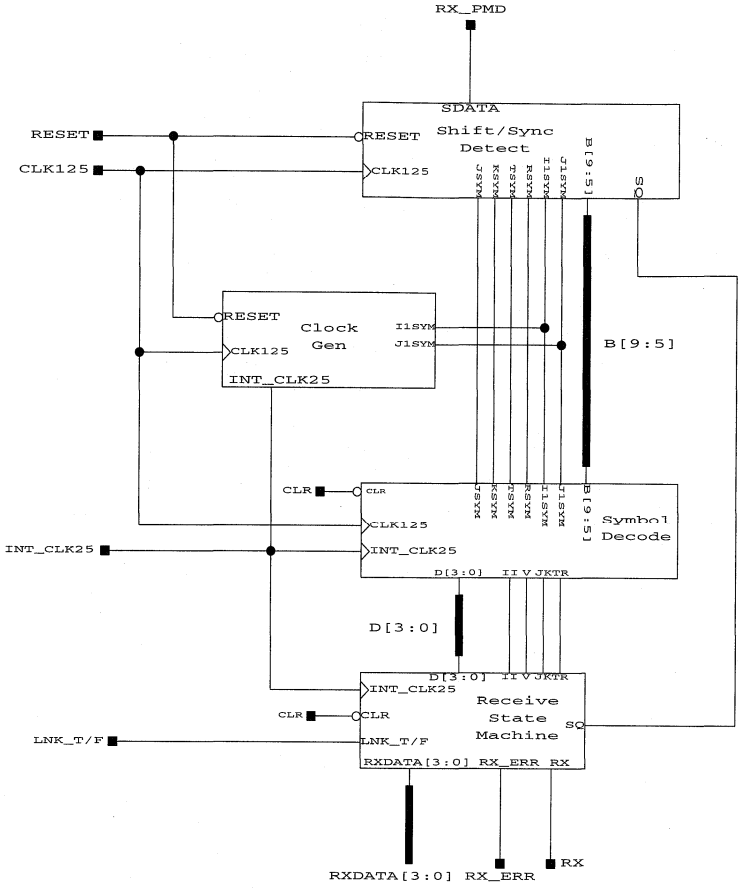


Figure 10 • Convergence Sublayer Receive Functions

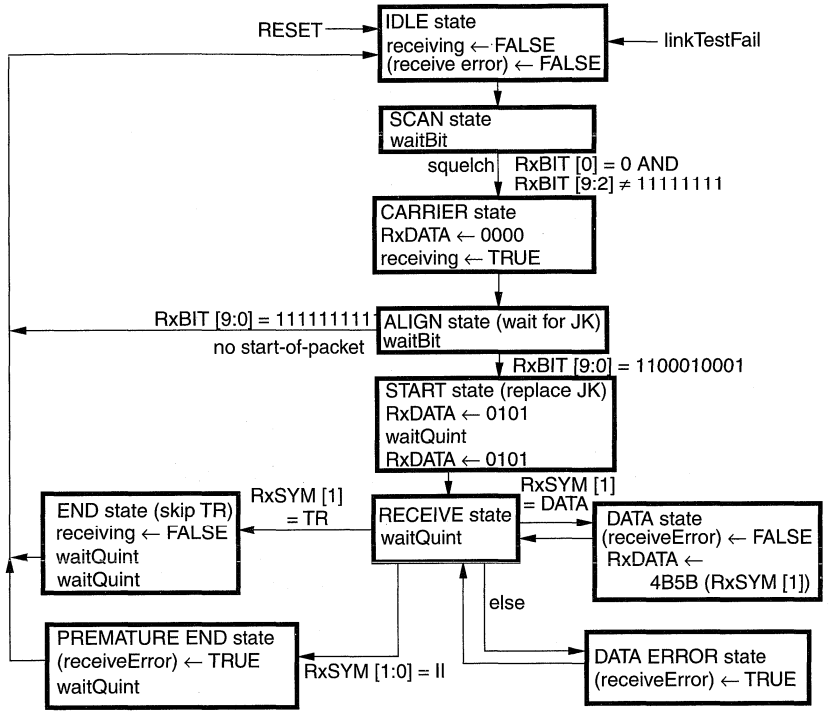


Figure 11 • *Convergence Sublayer Receive Operation*

Shift Register, Sync Detect, and Squelch

The shift register, sync detect, and squelch circuits (Figure 12) are responsible for shifting serial data at the 125 Mhz line rate and detecting clock synchronization symbols. Once a sync symbol is detected, the clock generation state machine adjusts the 25 Mhz symbol clock by stretching it the required number of 125 Mhz clocks to align it with an input symbol. Control symbols in the input data stream can then be captured correctly by the 25 Mhz clock and decoded by the 4B5B decode block.

Serial data is clocked into the shift register and sync detect block by using the 125 Mhz clock. Sync symbols are detected as the data shifts. As shown in Figure 12, only a single logic level is required to detect each of the five important sync signals (J,K,T,R and I). The code groups corresponding to each of these symbols are shown in Table 1.

Note: *The shift register generates both the true and complemented versions of bits B3, B4, B8, and B9. This is required to implement single-level decode for the I symbol because the ACT3 logic module implements five-input AND/NAND gates with at least two inverted inputs. The technique of providing additional registers with inverted outputs is common when implementing logic functions using fine-grained antifuse FPGAs. The additional registers cost little because of the fine-grained logic module, and they can be used where needed to provide additional logic signals. The abundant routing resources available with antifuse FPGAs also supply the additional routing required to create these additional logic signals.*

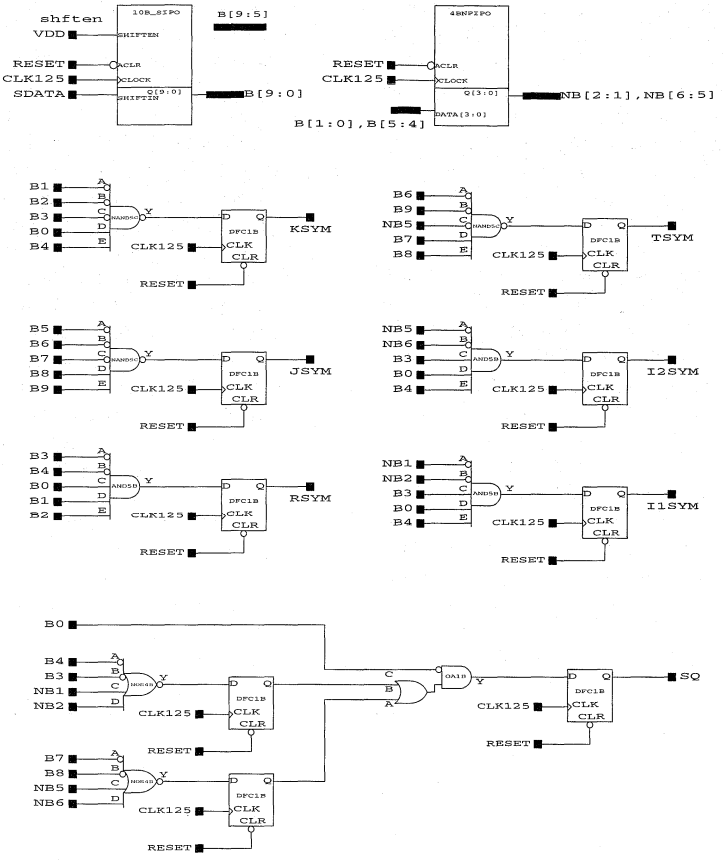


Figure 12 • *Shift Register, Sync Detect, and Squelch Schematic Diagram*

The squelch function filters out noise events from the received data stream. Zeros are ignored unless there are two noncontiguous zeros within the first 10 bits. At first glance, it would appear that the logic to detect two noncontiguous zeros in a 10-bit word should be quite extensive. However, once it is observed that this function is used only on a serial data stream, several simplifying logic reductions can be made. First, check the least significant bit (B0) for a zero, and then check that at least one of the higher-order bits (only B2 through B9, since B1 is contiguous) is also zero. Any other combination is simply a shift from B0. However, the resulting logic equation for a squelch state (S),

S:= /B0(/B2+/B3+/B4+/B5+/B6+/B7+/B8+/B9)*

is too large to implement in a single FPGA logic level.

To simplify this approach, note that because data is being serially shifted in, higher-order terms can be precomputed and then combined with the critical B0 signal using a single logic level. The logic that results can be expressed by the following three equations:

S1:= /(/B1+/B2+/B3+/B4)

S2:= /(/B5+/B6+/B7+/B8)

S:= /B0(/S1+/S2)*

These three equations are the ones actually implemented in FPGA form, (see Figure 12.)

Note: *As shown in Figure 12, bits B1–B4 and bits B5–B8 are used with registers to develop the two intermediate terms S1 and S2. These two are then ORed with bit B1 to develop the final squelch function (S). This form of pipelined operation works well in serial data applications and will almost always result in faster and more area-efficient FPGA designs.*

Also, notice that the extra inversions on the S1 and S2 terms (Figure 12) are used because NOR functions with inverted inputs map more easily into in a single ACT3 logic module. Synthesis software like Actel's ACTmap Program figures this out automatically, allowing the designer to focus on architectural and functional issues instead. Thus, what initially looks like a difficult decoding problem can be significantly simplified to only three logic modules that operate easily at the serial data rate.

Clock Generation

The clock generation state diagram and the clock generation schematic diagram are shown in Figures 13 and 14, respectively. The clock generation logic divides the 125 Mhz serial clock by 5 to generate the 25 Mhz symbol clock, Clock25. The Clock25 signal (Figure 13) is stretched when a sync symbol is detected, to align it with the 5-bit symbols. This is accomplished by a transition to the Q0 state when the JK signal (start of packet) is active. Entry into Q0 synchronizes the 25 Mhz clock (Clock25, the output from states Q2 and Q3) and the load signal. The load signal is active every 5 clocks after synchronization, which captures the 5-bit symbol from the aligned data stream. The symbol can then be safely captured by the 25 Mhz clock, Clock25 in the aligned symbol register (ASR). The schematic implementation for this process is shown in Figure 14.

Note: *Each state in the machine usesa single register. This one-hot (i.e., one register at a time) type of state machine design uses the register-intensive nature of fine-grained, antifuse-based FPGAs to reduce the logic complexity required to determine next-state transitions. In traditional encoded designs every state bit is needed to determine which state the machine is in. This can make for large transition terms in complex state machines. FPGAs, on the other hand, can use the additional register available to reduce the logic complexity, because only a single register output is required to determine the state of the machine. Thus, the FPGA's narrow, high-speed logic module can be used to generate the transition terms efficiently. In fact, on closer examination, the implementation of the state machine maps very closely to the state diagram. Transitions from one state to another result in a connection from the starting-states register to the entered-states register. Logic complexity can easily be estimated directly from the state diagram. Because only a single logic module is required to implement even the most complex transition, the entire machine runs easily at the 125 Mhz clock rate.*

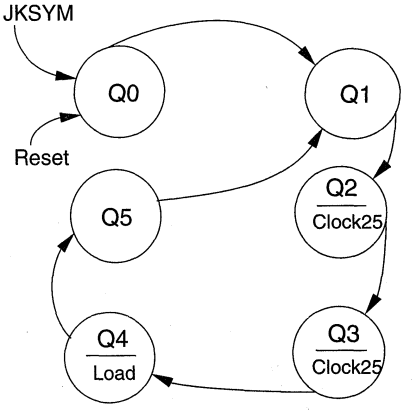


Figure 13 • *Clock Generation State Diagram*

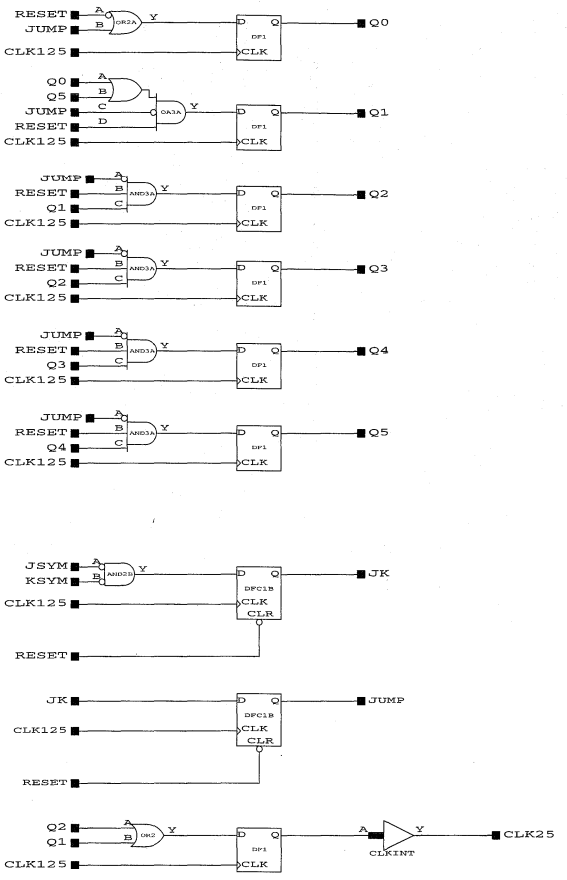


Figure 14 • Clock Generation State Machine Schematic

5B4B Symbol Decoder

Once a symbol has been aligned, the data must be extracted by converting the 5-bit input from the PHY into a 4-bit data word that is sent to the MAC. The logic diagram for this decoder is shown in Figure 15. Symbol conversion is done in accordance with the 4B5B decode table, Table 1. Implementation of the decoder in the ACT3 family is automatically generated from the logic equations developed from the encoding table by using the ACTmap tool. As shown in Figure 15, the full decode requires only 24 modules and only two levels of logic, easily meeting the speed required for the 25 Mhz clock.

Receive State Machine Diagram

The RECEIVE state diagram is shown in Figure 16. The machine begins in the START state and waits for the reception of a JK symbol pair. The RECEIVE state is entered upon the reception of this pair and exited only under one or more of the following conditions:

- Reception of a TR symbol pair (end of data packet)
- Reception of an idle (I) symbol (premature packet end)
- Reception of an invalid symbol (error condition)

Note that if an invalid symbol is received, the ERROR state is entered to capture the event. This state is cleared only by resetting the state machine.

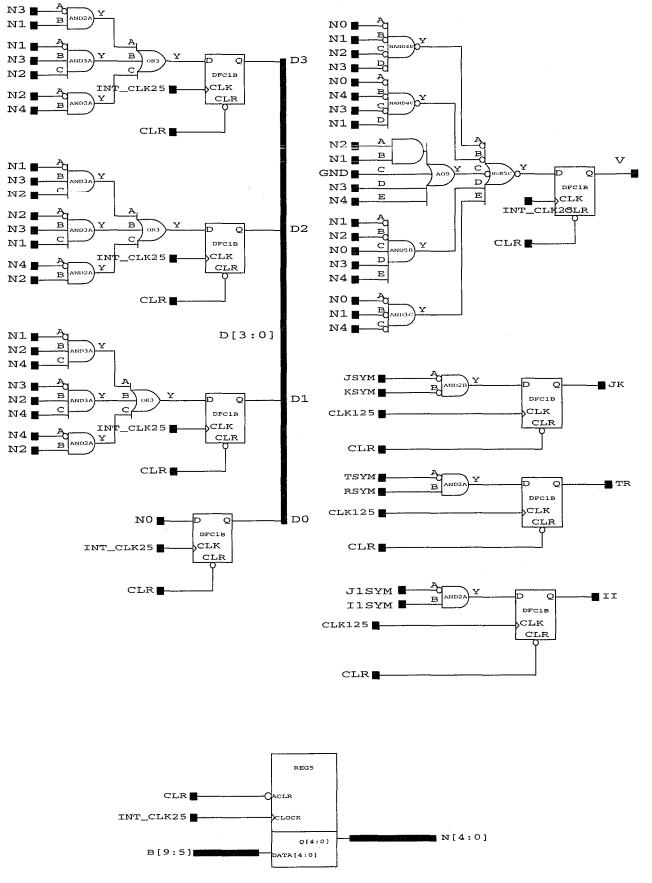


Figure 15 • 5B4B Decoder Schematic Diagram

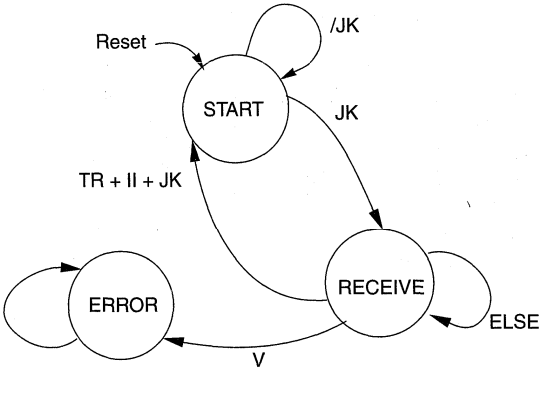


Figure 16 • Receive State Machine Diagram

Receive State Machine Logic

As shown in Figure 17, the schematic implementation of the RECEIVE state machine requires only 4 logic modules and two levels of logic. It easily meets the 25 Mhz clock rate required for this portion of the design.

Carrier Sense and Link Monitor Circuits

The carrier sense and link monitor circuits combine outputs from the transmit, receive, and PMD blocks to develop the receiveError, carrierSense, and collisionDetect signals. The logic for implementing this process is shown in Figure 18.

Conclusion

This application note has described the complete design of a 100Base-X convergence sublayer. Each building block has been fully tested and documented and is available on disk to those contemplating similar or related applications.

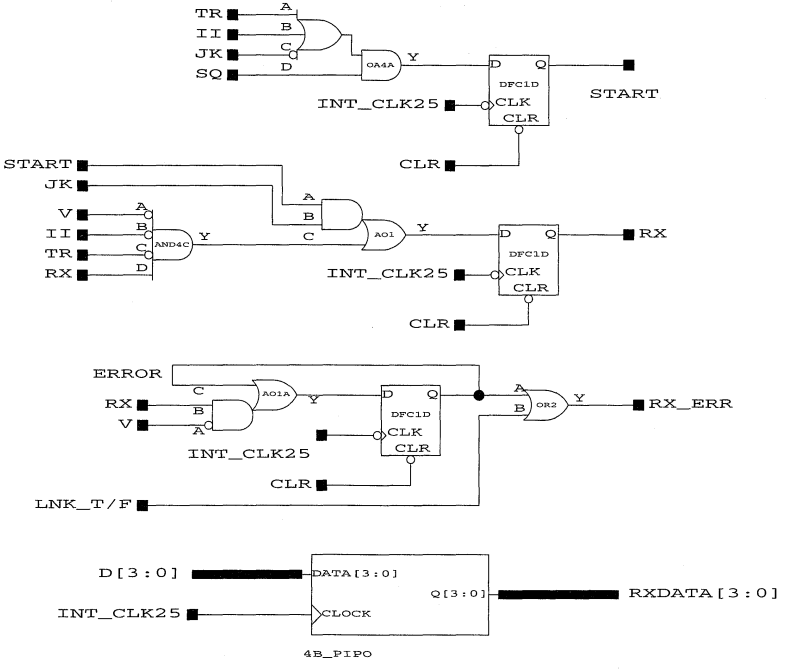


Figure 17 • Receive State Machine Schematic Diagram

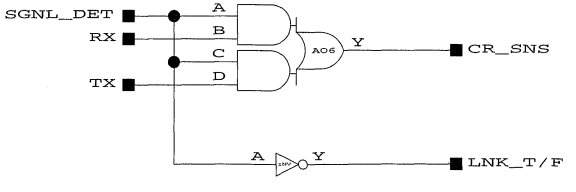


Figure 18 • Carrier Sense and Link Monitor FPGA Logic Schematic

A High Performance Networking Interface Using Actel FPGAs

Introduction

High-speed serial data transmission is quickly becoming the most cost-effective method of connecting systems for a variety of applications. From workstation and PC connectivity to factory automation and automotives, networks have become a pervasive part of the modern day computer and communications fabric. The standard use for networks is to allow a variety of different equipment to effectively communicate but there are non-standard, proprietary applications as well, where high bandwidth communications is needed. This note will discuss the design of a proprietary high-speed network that connects distributed processing elements in a document retrieval system. It uses a single Actel 1425 field programmable gate array (FPGA) running at up to 100 MHz.

Network Architecture

Reliability and high performance were the goals in this system. It was important to have immediate access to a large amount of data, usually scanned images of documents. A 100 MHz fiber-optic token ring network was chosen to ensure high performance and reliability. Network management was key to ensuring reliable transmission and managing data flow effectively. This suggested a tree architecture with concentrators distributed throughout the tree that could manage data transmission and ensure reliable connections. The resulting network architecture is shown in Figure 1.

The top of the diagram shows the network concentrators; the document processing stations are at the bottom. Disk farms containing document archives are distributed throughout the tree. Data requested by the processing stations flow from the disks through the concentrators and arrive at the requesting station. Data transfers between disks are also possible if network traffic is determined to be better managed by transferring data between disks, thereby keeping data transfers local.

Although the network architecture is shown as a tree, the data actually travels around the network as if it were a ring going from station to station until the receiving station takes the data off the network. This is illustrated more clearly by looking at a single concentrator in the network, as shown in Figure 2. Data from the network enters at the top of the diagram and flows down to node 0. If the data is not destined for node 0, it is sent back up the link to the concentrator

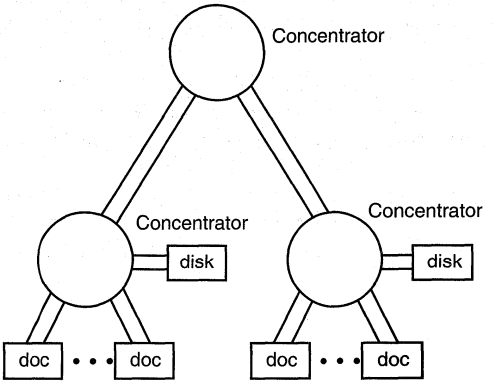


Figure 1 • High-Speed Network Architecture

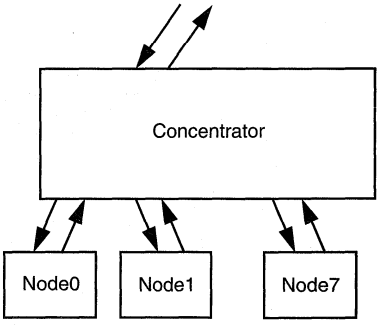


Figure 2 • Network Concentrator Interface

where it is routed to node 1. This process continues until the data arrives at the correct destination, where it is removed from the network. The header of the message is modified, showing that the data was received, so that when it arrives back at the sender the sender knows the data was received by the requesting station.

A node processor is located at the concentrator and is responsible for monitoring data flow through the local network. If a connection to a node fails because of a station malfunction or a line being removed, the node processor must detect the failure and re-route the data around the failing node. For example, if node 0 in Figure 2 experienced a network link failure, the concentrator would detect the failure and route the incoming data to node 1 instead. If node 1 also failed, the data would be routed to node 2, and so on. Thus, failing nodes do not effect the transmission of data to other nodes. This allows the system to continue operating even if some nodes are failing, being serviced, or nonexistent.

Concentrator Architecture

The concentrator architecture is shown in Figure 3. The main components are the node processor, its associated memory, the optical data links, and the field programmable gate array (FPGA) containing the digital logic needed to connect everything. The node processor monitors data packets moving through the concentrator and detects failing links, traffic jams, or other unusual flow patterns that might require data to be transferred between disks, diagnostics, and other network management functions. It can receive data as a node in its own right, and it communicates with other concentrators regularly.

The optical data links translate the electrical signals from the concentrator to optical signals carried by the optical fiber and translate optical signals from the fiber into electrical signals used by the concentrator. The 100 MHz serial data clock is extracted by the main receiving data link and used to synchronize all other links, the node processor, and the FPGA.

The FPGA is used to connect the synchronized inputs and outputs of the data links and the node processor. The node processor can control data flow between data links and bypass failed links. The node processor can also monitor transmissions between links to accumulate statistics useful in detecting bad links, network traffic flow, and other important items.

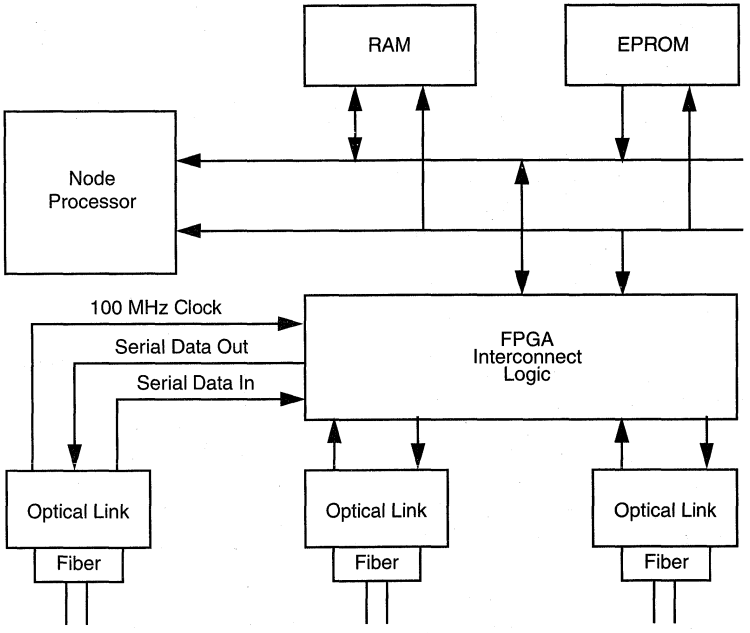


Figure 3 • Network Concentrator Architecture

Architecture of the Concentrator FPGA

The major tasks of the FPGA are to allow the node processor to control adding or removing links from the ring (bypassing), passing data from one active link to another, and capturing data packets for the node processor to use in gathering network statistics. The FPGA contains three major operational sections, as illustrated in Figure 4; the serial data path, the node processor interface, and the status/control section. The serial data path connects the serial data from the data links as determined by the path control register. A single bit is used with each serial data channel to determine if it is bypassed or not. The processor sets the desired bits in the path control register to determine interconnectivity.

The status/control section provides the node processor with the needed "hooks" to observe packets as they flow through the network and to read the contents of various status and control registers. A simple state machine manages the flow of data between the node processor and the serial network and synchronizes the transfer between these sections. A description of the major status and control registers follows.

Control Registers

The link control register determines which links (if any) are bypassed (one bit per serial channel).

The snoop address register determines which data links output is stored in the snoop data register for access by the node processor.

Status Registers

The link active register indicates which data links (if any) are inactive (no signal is present). This is used to determine if a link must be bypassed.

The snoop data register contains the data received from the addressed data link.

The node processor interface allows the node processor access to the interconnection control register, status registers, and the packet snoop register. It contains a bidirectional data bus, bus control logic, address decode logic, and the usual microprocessor glue.

Design of the Concentrator FPGA

The most challenging portion of the concentrator design is the serial data link section. It contains the 100 MHz data path, which is usually a speed out of the reach of most FPGAs. As shown in Figure 5, the serial data path for an individual bit consists of the bypass multiplexer, a shift register bit, and the output register. The output of the previous link can be selected as the data source or that link can be bypassed by selecting the other multiplexer input. The input/output pairs are cascaded together, with the main concentrator serial inputs and outputs appearing at the ends of the chain.

Given the register-intensive nature and high operating frequency requirement of the application, an FPGA was the natural choice. To hit the 100 MHz performance goal, a 10 ns

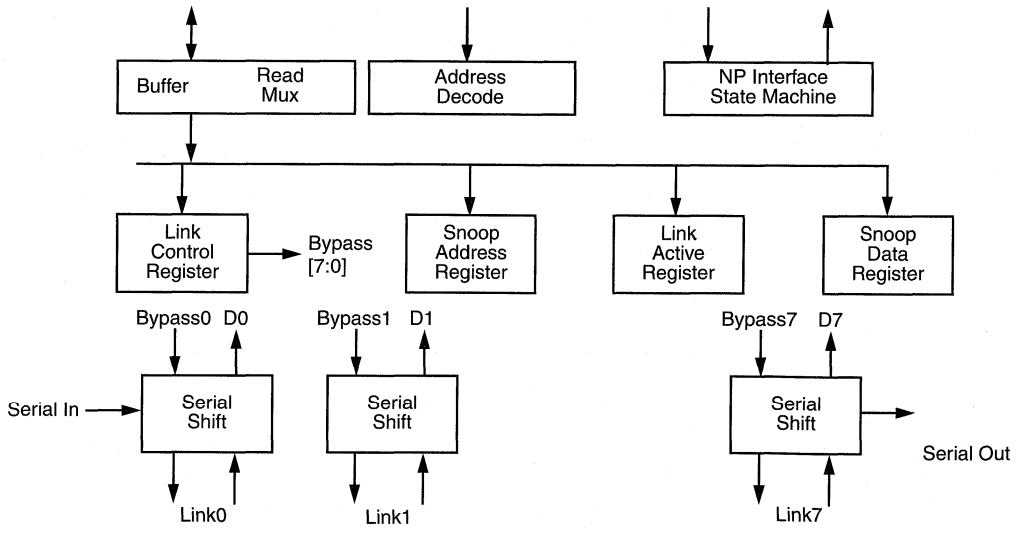


Figure 4 • Concentrator FPGA Block Diagram

clock to output is required on the output register, a 3 ns setup time is required on the input register, and an internal data path rate of 100 MHz is required. The Actel 1425 meets or exceeds all the requirements and has the capacity to implement the serial data path at only a fraction of a single device. The timing for the worst-case paths of the serial data section in a fully automatically placed and routed device is shown at the bottom of Figure 5. All numbers are worst-case commercial.

The status and control section of the design consists of the necessary registers and selection circuitry to allow access by the node processor. The registers are written into as determined by the address supplied by the node processor and read out in a similar manner. Of more interest is the snoop data register. It needs to be loaded from the selected data link at the 100 MHz network rate and read by the node processor before the next word is shifted in. These transfers are accomplished within the needed 100 ns by the node processor, using a block transfer capability. The FPGA ensures that the block transfer is accomplished and synchronizes the serial data stream from the network with the node processor data bus. Figure 6 shows the portion of the design where the serial data is stored by the FPGA, transferred to the snoop data register, and accessed by the node processor. A small state machine controls the transfer of data from the serial register to the snoop register and ensures that the processor doesn't miss any data. The transfer of data is stopped by the node processor after a complete block has been transferred. A new link can then be selected for snooping.

The node processor interface is the most straightforward portion of the design and is shown in Figure 7. It contains the processor interface, address decode, and control logic sections. The data bus is bidirectional and is synchronized with respect to the processor clock. The control logic determines which register is read from or written to according to the address inputs. Additionally, the interface to the snoop register is controlled by this section and manages the data transfer between the processor and the FPGA during burst mode transfers. When the processor has captured as much of the packet as desired, it simply stops reading data and the snoop register is allowed to be overwritten by the snoop shift register until another transfer is requested.

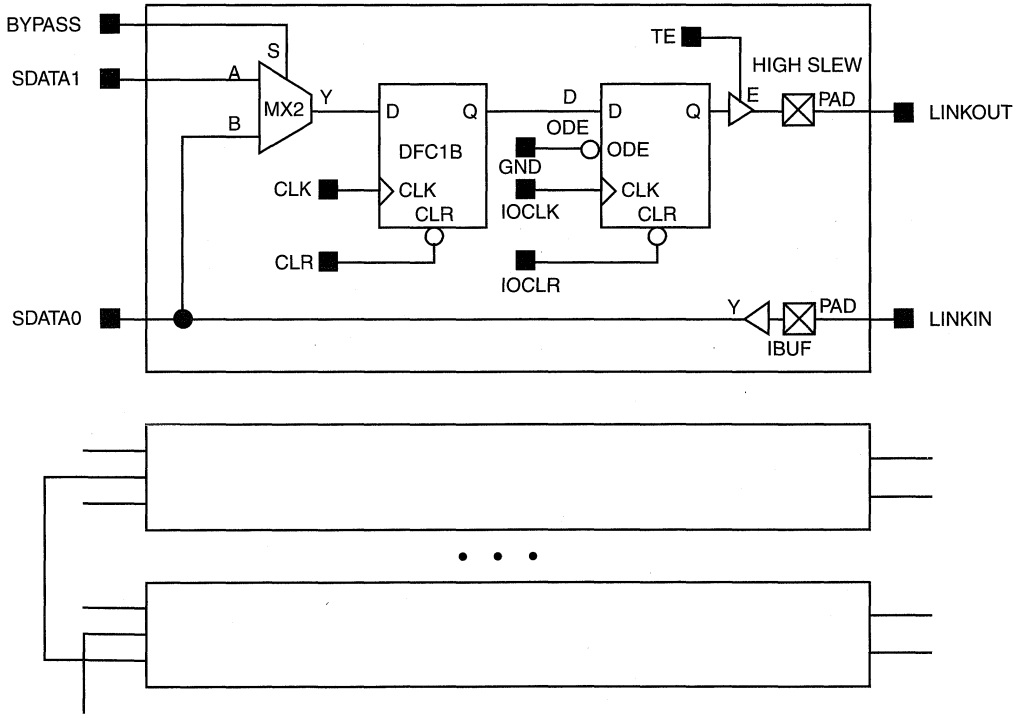


Figure 5 • Concentrator Schematic and Timing Diagram

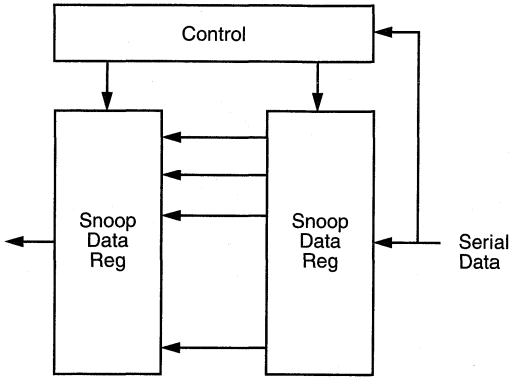


Figure 6 • Serial Data Storage

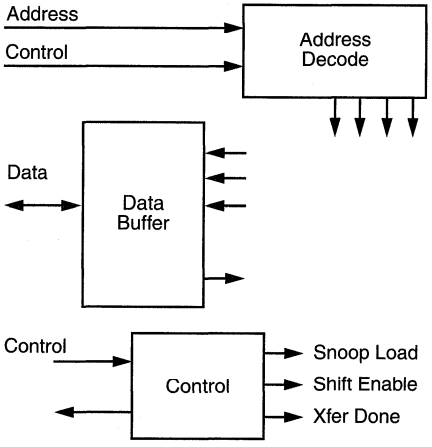


Figure 7 • Node Processor Interface

Conclusion

This application note discussed the detailed design of a high-speed data concentrator using the Actel A1425. Applications like this illustrate the types of designs this new generation of FPGAs can address, applications that were out of the reach of previous FPGA devices.

Designing High-Speed ATM Switch Fabrics by Using Actel FPGAs

The recent upsurge of interest in Asynchronous Transfer Mode (ATM) is based on the recognition that it represents a new level of both speed and simplification in telecommunication networks. The most significant characteristic of ATM is that it requires minimum cell processing in network nodes and in links such as repeaters, bridges, and routers. This means that ATM allows systems to operate at rates much higher than current packet-switching systems allow. This improved performance is due to higher media quality and to ATM operation in a connection-oriented mode that guarantees minimum packet loss. This low packet loss is the result of not granting entrance to the network until completion of a setup phase that allocates all necessary network resources.

To reduce the size of the internal buffers in switching nodes, and thus to reduce the queuing delays in these buffers, the information field length in ATM packets is kept relatively small. As a result, as packet size goes down, the speed requirement for each switching node on the network goes up. In general, to keep packet loss to a minimum, the throughput of ATM switching nodes must be in the 1-gigabit-per-second range.

This application note describes how to design typical high-speed switch fabrics that route ATM packets on broadband networks. *Switch fabric* is a term used to denote a large group of basic switching building blocks connected in a specific topology. The design, analysis, and implementation of these building blocks will be described.

ATM Switching Applications

One of the main tasks of an ATM switching node is to transport ATM cells at high speed from its input ports to its destination output ports. This task is performed by the switch fabric. The switch fabric establishes a connection between an arbitrary pair of input and output ports. Switch fabrics usually consist of identical basic units called *switching elements*. The switching elements are interconnected in a specific topology to create the switch fabric.

The Actel ACT 3 family of FPGAs, with their high-speed multiplexer-based architectures, are, as will be seen in this application note, an excellent fit for applications, such as ATM switching, that stress the heavy use of multiplexing. This application note will describe in detail the designs of two typical high-speed ATM switches:

- A pipelined 16:16 switch fabric
- A 16:16 multipath interconnect (MIN) switch fabric

Piplined 16:16 FPGA Switch Fabric

One of the simpler FPGA approaches to ATM switch fabric design is shown by the straightforward 16:16 multiplexing scheme in Figure 1. This switch fabric design is known as a *single-path network,* because the same path is always used from any given input to a given output. In this example, the switch fabric has 16 input ports and 16 output ports. To achieve connectability, it employs 16 16:1 multiplexers called FFMX16's (Figure 2). Each of the 16 FFMX16s uses five Actel DFM6A basic 4:1 multiplexed flip-flops connected in two pipelined stages.

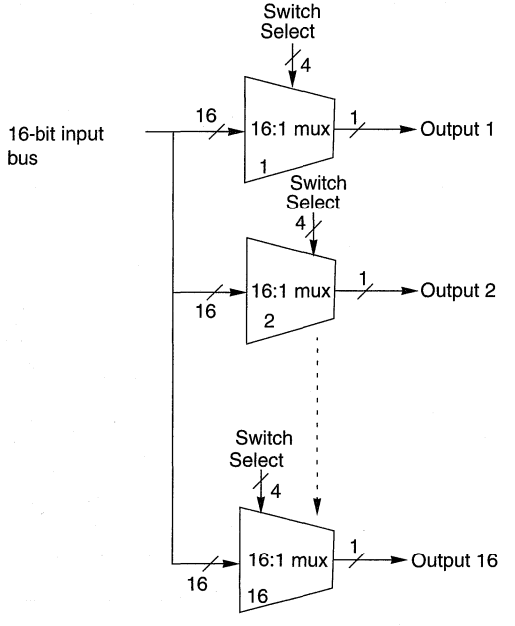


Figure 1 • 16:16 Basic Multiplexer Switch Fabric

Each DFM6A (Figure 3) is a 4:1 multiplexer driving a flip-flop and occupies a single Actel ACT 3 sequential logic module. This multiplexed flip-flop introduces only one level of logic delay in the design. In this implementation, once each of the two stages of the 16:1 multiplexer is full, the pipeline outputs data on every subsequent clock cycle. Thus, these 16:1 multiplexers are effectively implemented in one logic level, providing improved throughput.

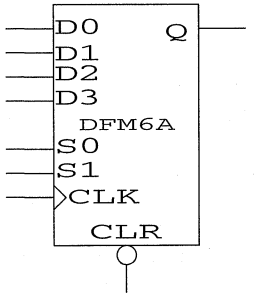


Figure 3 • Multiplexed Flip-Flops DFM6A

Switch Fabric Driving Circuit

The driving circuit for each of the FFMX16s in the 16:16 switch fabric is shown in Figure 4. As shown in the figure, selecting data for the output of each FFMX16 requires 64 signals. This number is based on the need for four switch-select inputs (S0, S1, S2, and S3) for each of the FFMX16s. Switch-select signals S0 and S1 operate with the first stage of each multiplexer, and select signals S2 and S3 operate with the second stages. A drive signal is received as a serial bit stream (SWSEL) that is converted to parallel form by a 64-bit serial-to-parallel shift register (SIPO64). Input data to the switch is received on the lines D[15:0]. Notice that the FFMX16 shown in Figure 4 represents 16 FFMX16s, so the inputs are 16 bits wide.

It takes two clock cycles to fill the FFMX16 pipeline, after which data is present on the OUTP[15:0] bus at each clock cycle. The 64-bit data selection word S[63:0] from the shift register is divided into four groups of 16 bits each, which are used to select the appropriate routing through the switch fabric. Note that there are separate clock lines for both the shift register and for the switch fabric so that data can be clocked to the output bus at a rate different from that of the shifted control bits.

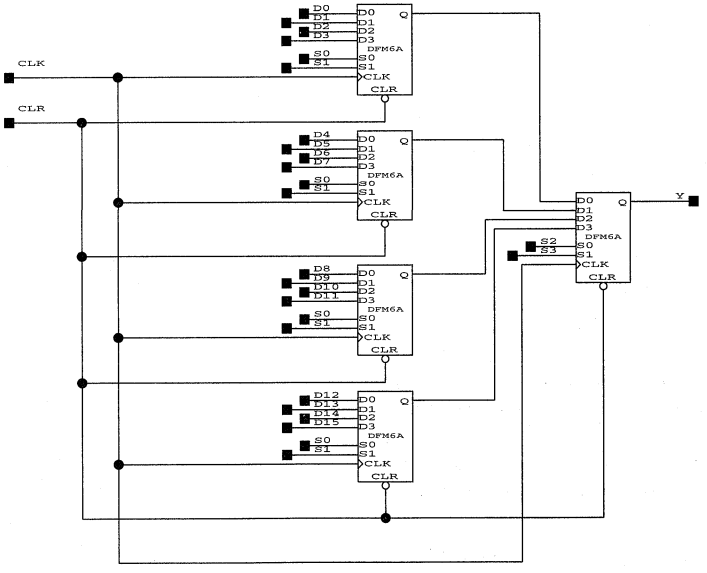


Figure 2 • Two-Stage Pipelined 16:1 Multiplexer (FFMX16)

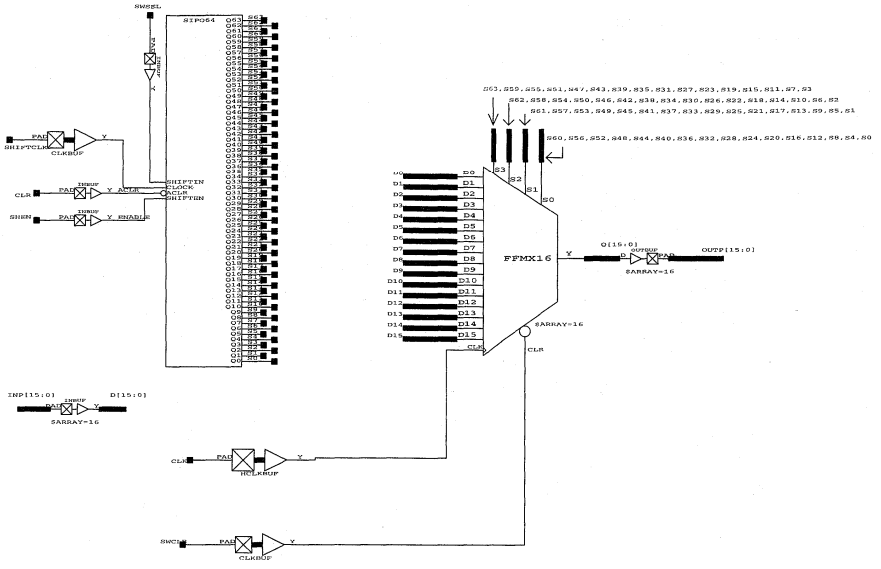


Figure 4 • Top Level View of the 16:16 Switch Fabric Design

Using ACTgen Macro Builder

The 64-bit serial-to-parallel shift register (SIPO64) is generated by the Actel macro generator, *ACTgen* included with Desinger Series software packages. With ACTgen's graphical user interface, you can build structured macros (counters, adders, etc.) by simply clicking on a few menu choices. The ACTgen Macro Builder then creates functions that effectively use the Actel architecture. Each macro is developed with the goal of limiting module count, maximizing performance, and restricting loading to acceptable levels.

In this design, the SIPO64 is generated by simply choosing the desired parameters from ACTgen's graphical user interface (64-bits, serial-to-parallel, active-low clear, active-high shift enable, and positive-edge triggered clock). The created shift register is then instantiated in the design with no need for simulation. The ACTgen macros are already tested to guarantee correct functionality.

Note: *In the N:N multiplexed structure shown here, any given input may be broadcast to all outputs simultaneously. Also, an advantage of this approach is that after only two clock cycles, the pipeline is full and ready to output data. A disadvantage of this scheme is that it allows only one possible path, and no alternative paths, between each input and output. To implement a multiple-path capability, multiples of this switch fabric can be cascaded together. However a better solution is the MIN switch fabric.*

16:16 Multipath Interconnect (Min) Switch Fabric

The primary advantage of a multipath interconnect network (MIN) is that it permits the creation of alternative paths between a given input and a given output in order to avoid possible packet collisions. One implementation of a MIN switch is the two-section Banyan network shown in Figure 5. This network consists of four stages that drive a second group of four. The second group is made up of the first four stages with a reversed topology—it is the mirror image of the first. Adding this second half produces a complete MIN switch fabric in a minimum number of stages.

The first four stages (see Figure 5) enable any output to be reached from any input via one specific path. This is the standard Banyan configuration. The second four stages use a reversed Banyan topology. Together, the two sections provide the multiplicity of paths required for a MIN switch fabric. That is, in an N: N MIN switch fabric, N internal paths are available to reach any output from an arbitrary input.

Each basic switching element used in this switch fabric is a 2:2 switch. (See Figure 6.) Depending on the value of switch line SW, the data will be either passed or crossed between the input and output lines. Figure 7 shows the implementation of the basic switching element using Actel DFME1A ACT 3 multiplexed flip-flops. As shown in the figure, two multiplexed flip-flops are required to implement the switching element. The truth table for the basic switching element shown is given in Table 1.

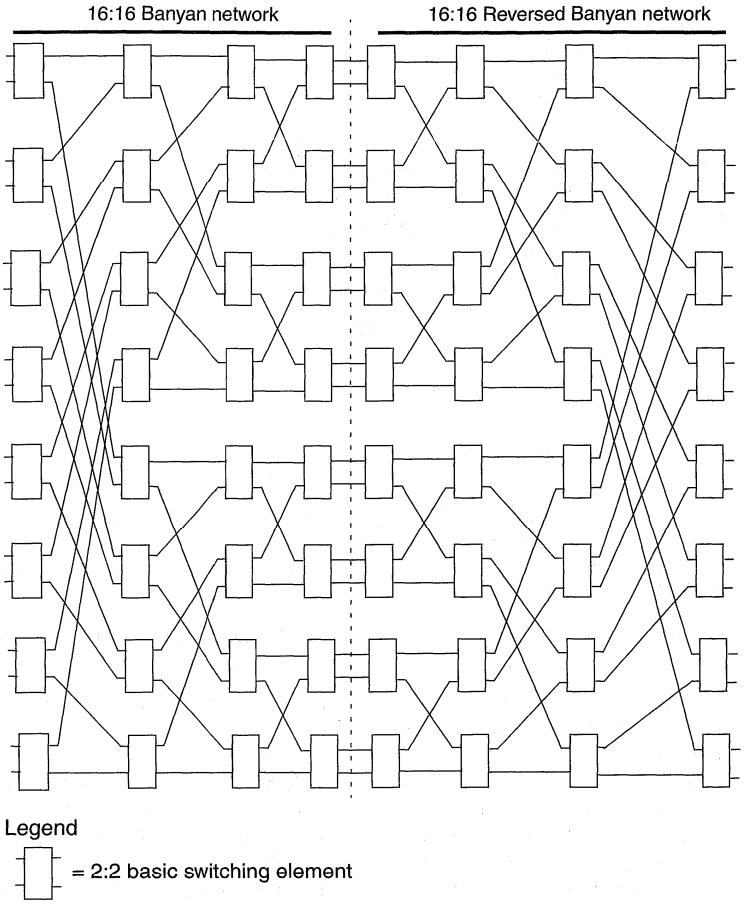


Figure 5 • 16 X 16 Mulltipath Interconnect Network (MIN) Switch Fabric

Table 1 • Truth Table for 2:2 Basic Switching Element SWCELL

Clock CLK	Enable E	Switch SW	Y0	Y1
X	1	X	As in previous state	As in previous state
↑	0	0	A	B
↑	0	1	B	A

Notes: ↑ = Triggered on positive edge of clock
X = Don't care

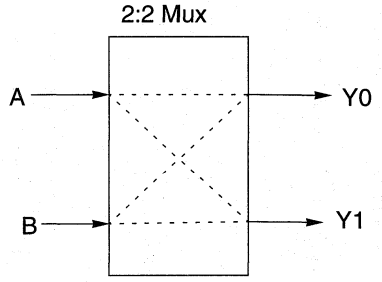


Figure 6 • Possible Signal Paths in a 2:2 Basic Switching Element

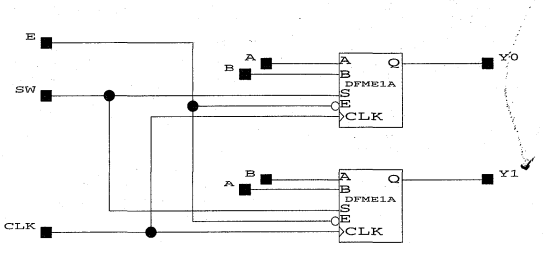


Figure 7 • 2:2 Basic Switching Element SWCELL

Assuming an N: N MIN switch fabric, the number of stages in the network is $2\log_2 N$. If N = 16 (as in the present case), the MIN switch fabric is implemented in eight stages. Thus, a 16:16 MIN can be constructed of eight stages, with each stage consisting of eight 2:2 basic switching elements.

Switch Fabric Driving Circuit

The driving circuit for the MIN network is similar to the one used for the multiplexer-based switch fabric described in the previous section. As shown in Figure 8, the switching elements (SWCELL) are connected to each other to implement the topology shown in Figure 5. The SW lines of the switching elements are driven by the outputs (S[63:0]) of a 64-bit shift register. The S[63:0] signals are received as a serial bit stream (SWSEL) and are converted to parallel form by a 64-bit serial-to-parallel shift register (SIPO64). Input data to the switch is received on the lines IN[15:0] and is clocked to the outputs once the SWCELLs are enabled.

It takes eight clock cycles to fill the switch fabric pipeline, after which data is present on the OUTP[15:0] bus at each clock cycle.The 64 bits of data selection word (S63:S0) from the shift register are used to select the appropriate routing through the switch fabric. Note that there are separate clock lines for the shift register and for the switch fabric so that data can be clocked to the output bus at a rate different from that of the shifted control bits S[63:0].

Note: *The complete multipath interconnect switch fabric is implemented by using 128 (8 x 8 x 2) multiplexed flip-flops of the type DFME1A. The straight multiplexed switch structure discussed in the previous section requires 80 (5 x 16) DFM6A multiplexed flip-flops. However, this size differential does not translate for larger values of N (where N is the number of input and output ports). As N gets larger, the number of modules required to implement the MIN network does not increase as rapidly as it does for the simple mux-structured network. Also, notice that the multiplexed flip-flop used in the MIN network is a 2:1 type, whereas the straight mux switch fabric requires a 4:1 type. The 2:1 multiplexed flip-flop offers the advantage that, because of its lower fanin, it is easier to route on the Actel software.*

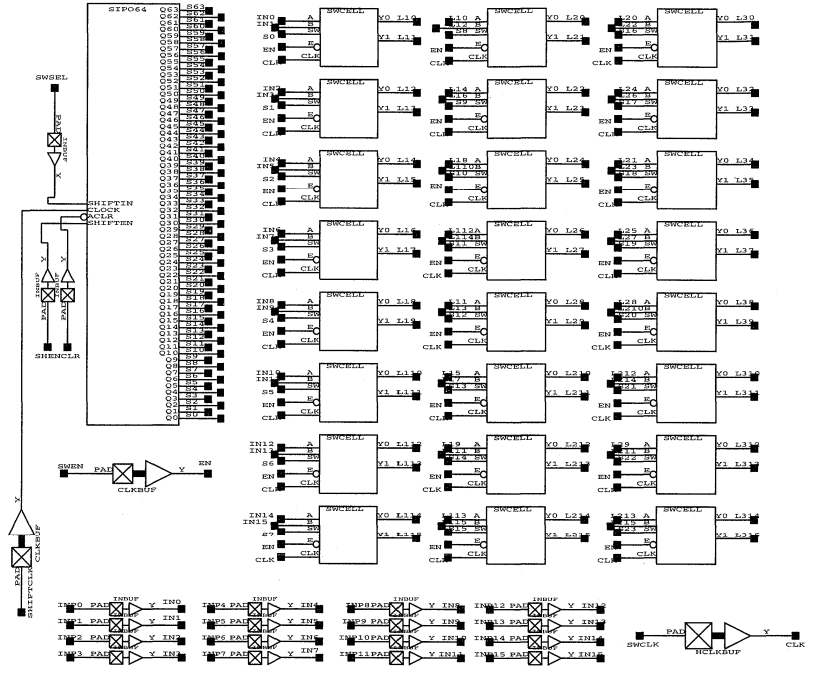


Figure 8 • Top-Level View of the MIN Switch Fabric Design

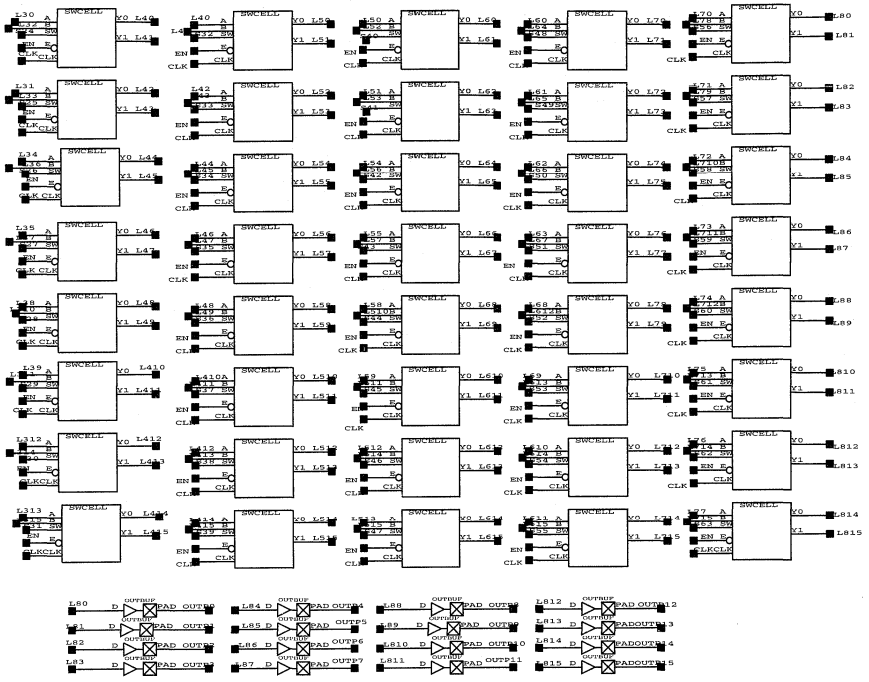
Timing Analysis

The MIN switch fabric discussed here can be implemented in most Actel ACT 3 devices, such as the A1425A, the A1440A, A1460, and the A14100A. The timing analysis given in this section was obtained from the A1440A-2.

The MIN switch fabric can be operated as fast as the slowest switching element can switch its data from input to output. The basic switching element has a 5.7 ns clock-to q (input-to-output) delay, along with 0.7 ns of setup time. Hence, the maximum frequency of the switch fabric clock is 156 MHz. In a 16:16 switch, this provides a maximum throughput of 2.5 gigabits per second. Note that it takes eight clock cycles for data to move from the switch fabric input to its output (about 51.2 ns). However, as in all such pipelines, once all stages of the network are filled up, data is output at every subsequent clock cycle.

Conclusion

The basic concept behind switch fabrics is multiplexing data from input ports to outputs. The multiplexer-based architecture of Actel FPGAs fits this requirement. High-speed switching networks of almost any topology can be implemented efficiently using the multiplexed flip-flops in the Actel library. Each of these flip-flops is mapped to only one sequential module within the FPGA to take maximum advantage of the die area within the chip.



Using FPGAs for Digital PLL Applications

In addition to purely digital applications, many designs use Field Programmable Gate Arrays (FPGAs) for DSP. We'll examine one such application, digital PLLs, to show various ways of implementing PLL designs using FPGAs.

Pulse Steal PLL

In telecommunications applications, it is often desirable to generate a digital signal that is locked to an incoming signal and is some multiple of its frequency. A drawing of a pulse steal PLL, which is a simple way to generate such a signal, is shown in Figure 1. Note that the design contains an ordinary oscillator but no VCO. Except for the crystal, the entire design will operate in an FPGA.

Note the frequency relationship that holds at points A and B in the figure, where

$$OSC/(K*M) = Input/N = Comparison\ Frequency \quad (1)$$

The technique is based on selecting a reference oscillator frequency slightly higher than OSC. This frequency (OSC+) should be chosen so that

$$1/Comparison\ Freq. - (K*M)/(OSC+) = .5 * (1/OSC) \quad (2)$$

The right side of equation 2 equals one-half the period of the reference oscillator.

The reference oscillator frequency delta will cause point B (the detector flip-flop D input) to begin to precede point A (the detector flip-flop clock input) by half a period. When the edge of the D input is sufficient, the detector will clock true and begin a pulse train through the two deglitching flip-flops. The output of the second of these clears all three flip-flops and steals a pulse by disabling the divide by K output. Stealing the pulse puts point B behind A until the reference oscillator delta can move it ahead by one period, repeating the cycle. Points A and B are always within one-half a cycle of each other.

The circuit allows the frequency of the output signal to be selected simply by adjusting the values of the dividers K and M. The lock range of the loop is given by the following:

$$Lock\ Range = \pm\ (OSC+/osc)/Input \quad (3)$$

Jitter-Bounded Digital PLL

Another technique[1] for generating a wide variety of synchronized clock frequencies with low jitter employs an accumulator Digital Controlled Oscillator (DCO) and phase and frequency comparators. The system, shown in Figure 2, can lock to any division of a reference frequency (F ref.) as selected by the data loaded into frequency divider counters.

The Successive Approximation Register (SAR) and its controller serve as a low-pass filter supplying the DCO with frequency and phase correction data. Among the three inputs to the SAR controller is the F ref. divided by a factor Q to form F q.

The other two inputs come from the phase and frequency (zero) comparators. The frequency comparator output is the DCO frequency divided by P to form F p. When the system is in lock the following equation is true:

$$F\ dco = (P/Q) * F\ ref. \quad (4)$$

The heart of the system is the accumulator DCO, which determines the ability to lock to a frequency and the amount of jitter allowed. The DCO consists of a four-bit accumulator whose input is fed by the SAR. The DCO input value is determined from the phase and frequency comparison feedback loops. The most significant bit of the accumulator output is the DCO output signal. It is generated by successively adding the SAR value to itself at the high-frequency system clock rate. The frequency comparator uses the value of P to divide the DCO frequency. If the frequency is out of lock during a period of F ref., the comparator asserts greater-than-zero or less-than-zero to the SAR controller to modify the value of the register. If the P counter output is zero, the DCO has the correct frequency.

The DCO latch acts as a phase register indicating the phase of the DCO with respect to F ref. The DCO phase is calculated by the N most significant accumulator output bits. When the DCO is out of phase, the jitter, or phase difference, is detected by the phase comparator and accumulates with time until it equals one period.The feedback loops then cause the SAR register controller to load a correcting value into the register or to clear the accumulator with a synchronizing pulse.

The Jitter-Bounded DPLL may be implemented entirely on an ACT™ FPGA. The resource requirements vary with the relationships of the system input and output frequencies, but for any F ref., system clock, and desired output frequency, the design is easily accommodated on an ACT FPGA.

References:

1. S. Walters and T. Troudet, "Digital Phase-Locked Loop with Jitter Bounded," *IEEE Transactions on Circuits and Systems*, Vol 36, No. 7, July 1989.

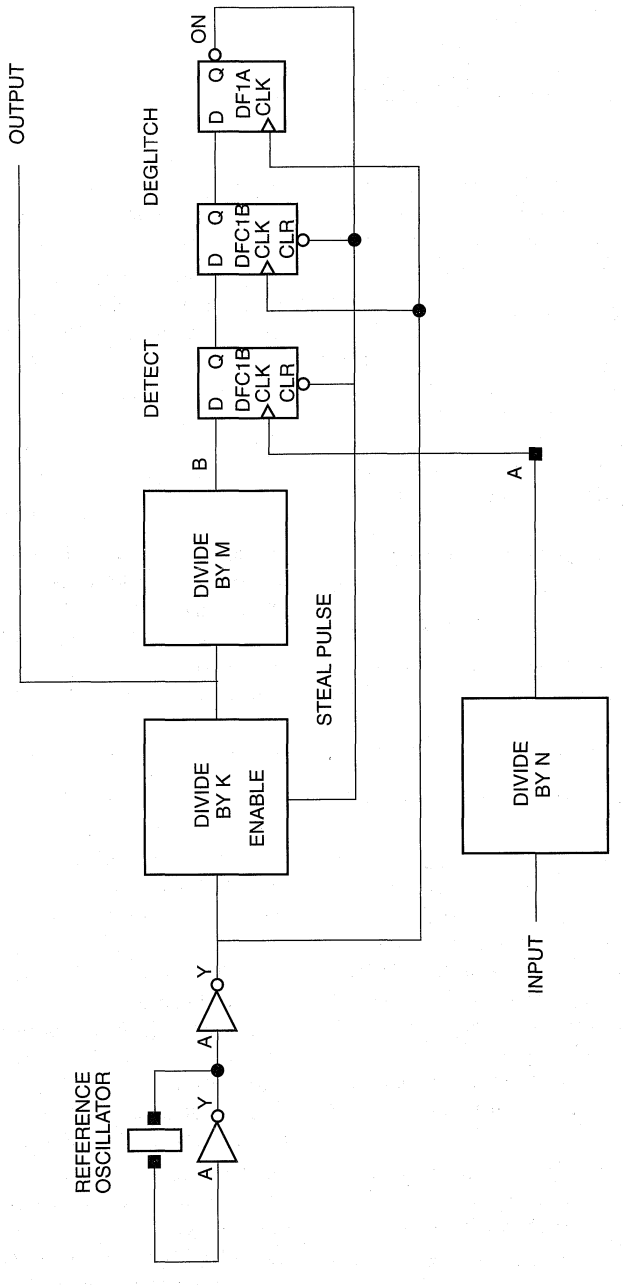


Figure 1 • Pulse Steal PLL

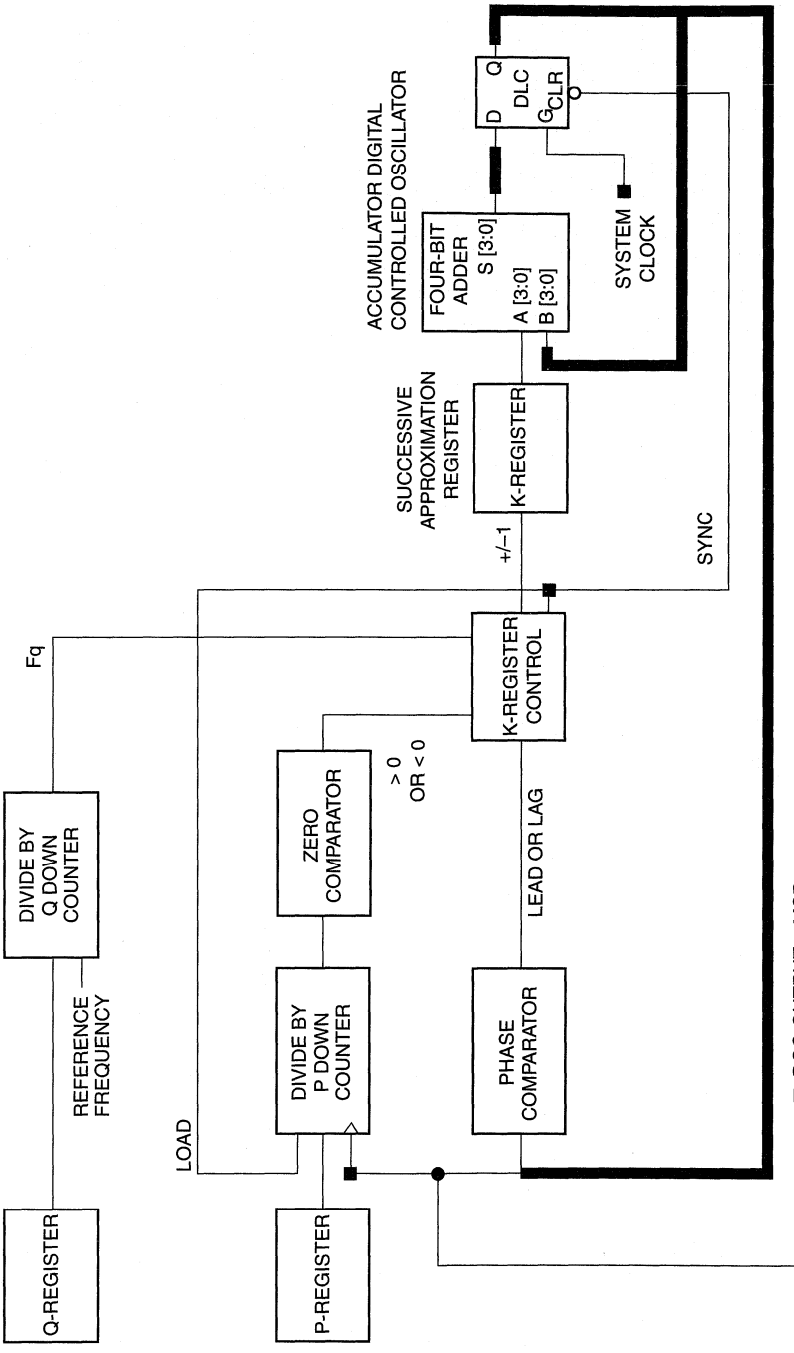


Figure 2 • Jitter-Bounded Digital PLL

PCI to Generic Memory Bus Bridge

This application note describes a Peripheral Communications Interconnect (PCI) to generic memory bus bridge. Although all design files have been compiled, this design should be taken as only one example of how a PCI interface can be constructed. No guarantees are made or implied as to the correct operation of this circuit.

The PCI specification defines a high bandwidth interchip bus protocol that is relatively inexpensive to implement. This document describes an example implementation of the PCI specification in an Actel FPGA. Although many implementation strategies are possible, this document is intended to provide insight into the general implementation strategy that best fits the Actel ACT 3 FPGA family and into techniques that maximize system clock speed. Special attention should be given to the configuration register organization, parity generation, checking, and reporting, as well as to how PCI states are organized. This example design represents a fully functional PCI to generic bus bridge. It interfaces the PCI bus to a generic memory bus and supports the required PCI configuration registers.

Figure 1 shows the top-level block diagram for the bridge. It is designed to support RAM directly connected to the FPGA pins. It responds to four types of operations that are initiated by bus masters on the PCI bus. They are write to memory (memory write), read from memory (memory read), write to configuration registers (configuration write), and read from configuration registers (configuration read). The example design does not initiate transactions and is therefore known as a PCI target. It does, however, support all necessary PCI protocol requirements such as flow control, parity generation and checking, target abort, and system error reporting. In addition, a subset of the configuration register definition is used to control the bridge's behavior for various conditions on the PCI bus. The addition of master mode states to the state machine would allow this design to perform some master mode operations.

The bridge supports the following functions:

- 32-, 16-, or 8-bit data transfers
- Even word parity generation and checking on the PCI bus
- Even byte parity generation and checking on the generic bus
- Burst or single word data transfers
- Data transfers to memory space
- PCI configuration registers

Features of the bus operation include the following:

- Positive address decoding for PCI to memory transactions
- Target initiated stop after parity error violations occur on memory write transactions
- Parity generation and checking of word parity covering 32 address/data bits and 4 CBE bits
- Full address decode and assertion of DEVSEL_ done within two cycles of FRAME_ assertion
- Parity errors reported during configuration and memory write. (master reports during reads)
- Parity errors logged in configuration registers on both read and write transactions
- Parity errors during address phase always reported and logged

The generic bus consists of a separate address, separate data, two control signals, memory write (MEM_WR_), and a memory ready signal (MEM_RDY_). Memory write writes data to RAM. Memory ready is a signal supplied to the bridge by the generic bus to indicate when valid data is available on the generic bus.

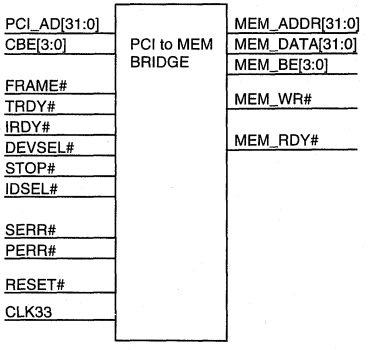


***Figure 1** • PCI Signals On the Left; Generic Memory Bus Signals On the Right*

PCI signals include the multiplexed address/data bus (PCI_AD[31:0]), four control/byte enables (CBE[3:0]), six control lines, two error reporting signals, reset, and clock[1]. The PCI clock is 33 MHz and provides the system time base. PCI supports three types of address spaces: input/output, memory, and configuration. The example bridge design

1. Refer to the PCI specification for a complete description of the PCI pins.

supports memory and configuration address spaces. A memory transaction connects data on the PCI bus to a peripheral device on the generic bus. A configuration transaction connects data on the PCI bus to configuration registers internal to the bridge. Each address space supports different address ranges. The configuration space can address up to 256 bytes. Of that, 20 bytes are implemented in the bridge. The PCI memory space can address up to 4 gigabytes. The bridge supports one megabyte.

Logic Overview

Figure 2 shows that the PCI bridge is organized as a group of major logic blocks. In addition, they are organized in roughly three columns. These columns indicate the pipelining structure used. All inputs are nonregistered, but all outputs are registered before being driven out to either the generic memory bus or the PCI bus[2]. The blocks listed from top to bottom and left to right, are the following:

- **PCI multiplexed address/data bus including word parity.** This is a bidirectional bus. The PCI address is moved into the PCI interface during the first clock cycle, followed by successive data or wait state cycles. This bidirectional block incorporates input buffers and output registers with tristate control.
- **PCI control/byte enables.** Four input pins serve two purposes. During the address phase, these four pins indicate the type of transaction to be performed. During the data phases, these pins indicate which bytes of the double word are enabled. This input block incorporates input buffers.
- **Configuration register.** The configuration register is used to control the PCI interface. It includes command, status, and base address registers (BAR). The BAR is used to set the base address for the interface as well as to allow system software to determine the addressable range required by the interface. All registers can be written or read at any time.
- **Data register.** This is part of the pipeline structure used to ensure a 33 MHz operating speed. Byte parity is generated off the data register output. Data is captured in this register during each data phase.
- **Address latch and increment.** This is similar to the data register in that the address is captured during the address phase. However, in addition, a 2-bit burst counter is used to burst up to four double words (16 bytes) of data.
- **Word to byte parity generation and check: The PCI write direction.** Byte parity is generated from the address and data register outputs[3]. PCI parity (word parity) lags the data (and address) by one clock cycle. Parity errors must be reported on the PCI parity error signal (PERR_) two clock cycles after the data appears on the PCI bus. This parity block must then generate byte parity and check it against the word parity in one clock cycle. Byte parity is then passed to the generic memory interface, in phase with the data and address. This logic is used during a PCI to generic memory write operation.
- **Byte parity check and word parity generator: The PCI read direction.** During a PCI/memory read operation, byte parity is checked against the data. In the event of a violation, the parity error flag is set. Regardless of an error, word parity is generated from byte enables. The use of byte parity in both the read and write directions minimizes the parity logic complexity. Far simpler structures are used to generate parity over 8-bits and still maintain 33 MHz.
- **PCI state machine.** The control of the PCI interface is managed by this block. These states are a subset of those used in a full-function PCI interface. However, progression to a PCI interface that supports master mode, or other ancillary PCI functions, is managed by the addition of states in this block. The control state machine in the example design was instantiated using state-per-bit design in ABEL. State-per-bit design is well suited for state machines in FPGAs.
- **Memory data bus and byte enables.** The data bus interface is bidirectional and supports double word transfers. The byte enables are output pins and indicate which bytes of the double word are valid. In all PCI transactions, an entire double word must be transferred back to the requesting master. Therefore, these pins may not necessarily be used.
- **Memory address bus.** This includes 20 bits of address and addresses up to 1 megabyte of memory.
- **Memory byte parity.** Four bidirectional pins are used to write and read parity to and from the generic memory bus.

2. The registered outputs are intended to reduce the time to output signal valid. However, in some cases, internal registers in the ACT 3 family are almost as fast when driving output pins (with output buffers) and may simplify the design. The input direction is not registered, since the PCI specification requires a response to error conditions within two clock cycles. This precludes effective use a pipelined structure with input registers.

3. The PCI specification dictates that parity violations during an address phase indicate a system error. Parity violations during a data phase indicate parity violations. System errors are fatal whereas parity violations force can either a parity error response or be ignored, depending on the mode selected in the configuration command register.

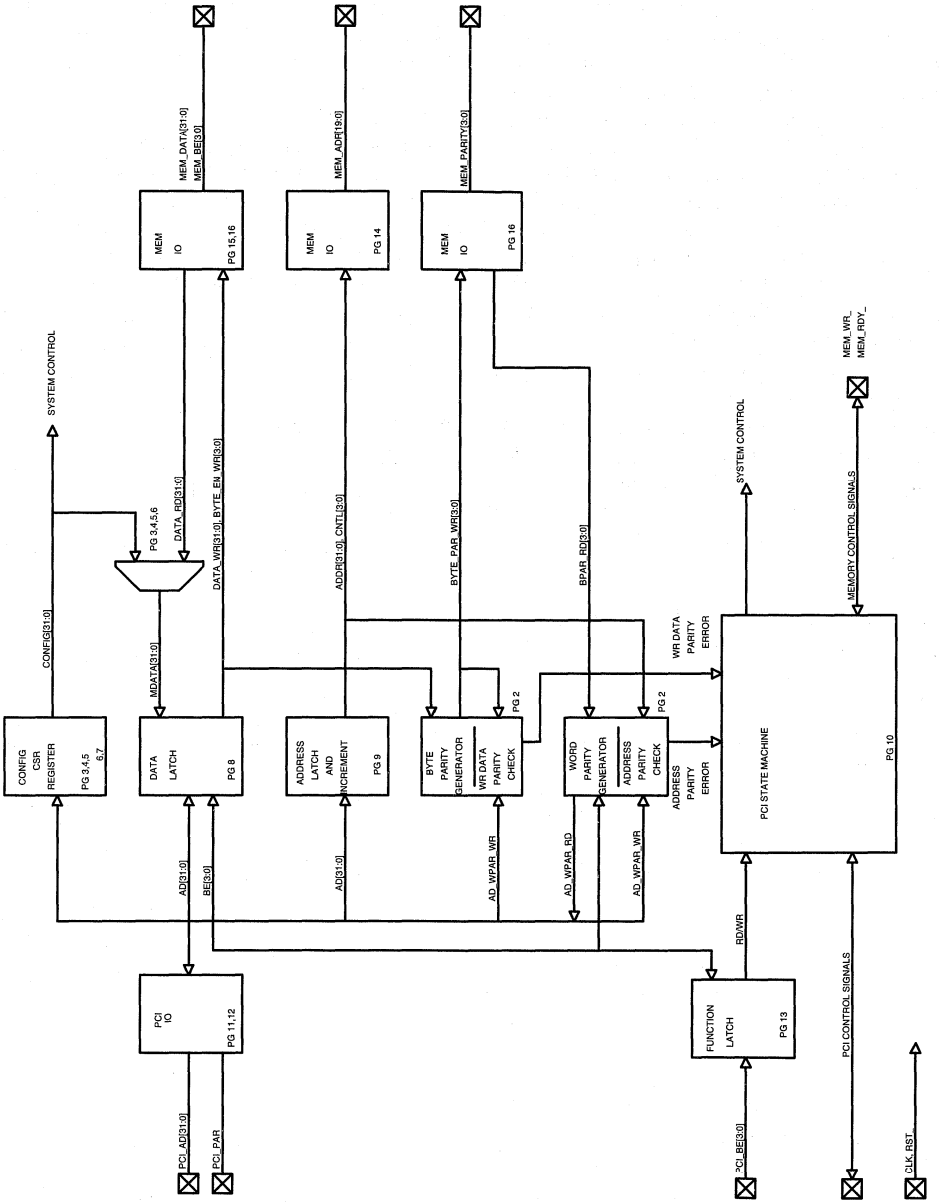


Figure 2 • PCI to Memory Bus Bridge Block Diagram

Flow Control

PCI is inherently a burst protocol bus. It achieves high bandwidth by combining bytes into a burst, adding overhead for routing (address and frame) and sending 32-bit (double) words at 33 MHz. The nature of PCI traffic and the design of the interface heavily affect the effective bus bandwidth. Two issues which affect a design are the degree of pipelining and the degree to which flow control is used. Pipelining structures are usually used to ensure that signal propagation times meet the frequency objective. In as much as they help minimize setup times, they also help reduce the number of levels of logic in a design. However, pipelining must be carefully managed, since it can lead to excessive pipeline latency. In some cases, it might be more efficient simply to use flow control to ensure signal setup times.

Flow control is used to match the bus bandwidth to the interface and secondary buses. However, it can also be used to help weak signal drivers achieve a signal level (1 or 0) over multiple clock cycles. This mechanism is called *address/data stepping* and is supported by the PCI specification. Any time flow control is used as a regular component of a bus transaction, it will severely cut into the effective bus bandwidth. There is a trade-off between pipelining and increasing bus latency versus the use of wait states as flow control devices. If bus traffic is generally made up of long bursts, then pipelining is very effective. If the traffic tends toward shorter bursts, then less pipelining and use of flow control may be more effective.

Two areas that may require pipelining or some flow control mechanisms are the parity generation logic and state machines. Both of these types of constructs tend to have high fanin. Typical design techniques are often not well optimized for FPGAs. However, the use of state-per-bit design and certain types of transformations on the parity paths can eliminate high fanin and optimize the logic for FPGAs.

Data paths don't usually have high fanin. Since 99 percent of typical designs have a fan in less than 4, it remains only to eliminate the small number of signal paths that result in excessive levels of logic. These techniques are discussed in the section on state machines and parity.

The example design decodes the address in one clock cycle. It does not use address/data stepping. Pipelining is used to minimize parity generation and checking setup times to meet a 30 ns clock cycle. Pipelining also ensures enough set up to drive 15 ns static RAM memory devices on the generic memory bus. Registering the inputs would help minimize input setup times; however, the data to parity error requirements of PCI preempt registered inputs.

Byte Enables

PCI defines four lines as control/byte enables (CBE[3:0]_). These signals convey control information during address phases and are used as byte enables during data phases. They indicate which byte lanes of the 32-bit (double) word have valid data. All read and write transactions are qualified with byte enables. Parity however, is always generated over all 32 bits plus the byte enables.

Device Selection

A PCI device is selected by two methods—either by decoding an address on the PCI bus (positive decoding) or by explicit device selection. When a device has been selected, it is required to assert the signal DEVSEL_ in order to claim the transaction. Failure to do so allows another device to claim the transaction through subtractive decoding[4].

The example design uses positive decoding during all memory transactions. It uses explicit device selection during configuration cycles. Subtractive decoding is not used.

Parity Operations

Except for a limited number of exceptions, parity generation and checking is mandated by PCI. Parity errors detected on a target system are reported only during write operations. During read, the master device detects and reports parity violations. This mechanism eliminates the need for more than one parity error pin. The target uses PERR_ to indicate to the master that a violation occurred. During a read, the master can use PERR_ to set an interrupt to the processor.

Parity errors during data phases are reported on PERR_. Parity errors during address phases are reported over SERR_, which indicates a system error. A system error is usually fatal. However, either type of error is logged in the configuration status register. SERR_, unlike PERR_, uses an open collector driver. Both master and target can assert the signal at the same time and therefore don't have the same constraints over when to assert the signal.

In the case of the example design, an open collector is simulated with a three-state gate (Figure 3).

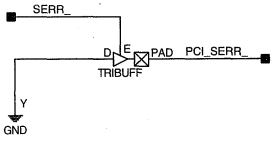


Figure 3 • Open Collector Simulated with a Three-State Gate

4. Subtractive decoding is often used by bus bridges that support secondary buses with limited control functions. Intel's SIO chip (PCI to ISA bridge) is a classic example that uses this mechanism. The SIO assumes that since the address was not claimed by a device on the PCI primary bus, that it might be destined for a device on the secondary bus that the SIO serves. Therefore, the SIO claims the address and passes it through to the secondary bus. If no device then picks it up, the transaction times out.

In the example design, byte parity is used to cover data on the generic bus and is in phase with the data. This simplifies parity generation and checking. Byte parity is generated over 8 bits and therefore reduces the number of levels of logic required. When using macro cells such as the TA280, reduce the number of input terms to just those needed. This eliminates any unnecessary logic that the compiler cannot optimize. Figure 4 shows an alternative method of generating parity over 6 bits in two levels. This method can be extended to cover 18 bits with an addition of one more level of logic. This structure is used in the example design.

In the design, word parity is generated from the byte parity bits and the byte enables. Byte parity from the memory bus is checked against the data. This means that all parity generation and checking never needs to span more than 8 bits. In addition, since word parity must be driven onto the PCI bus one clock cycle after the data, it can be pipelined first to check the data from the memory and then to generate the word parity.

Figure 5 shows how pipelining can ensure that the clock cycle is kept within 30 ns. After the example design was completed and timing analysis run, it became apparent that even with parity operations streamlined to minimize levels of logic, the word parity input path delay was too great. Restructuring the byte and word parity paths increased the byte parity path, but it reduced the word input parity setup time to within the PCI specification, in addition to reporting parity errors within two clock cycles.

The example design uses two parity generators—one for the data and another for the address. One generator would suffice; however, it would require creating a separate parity checking pipe and adding the necessary control.

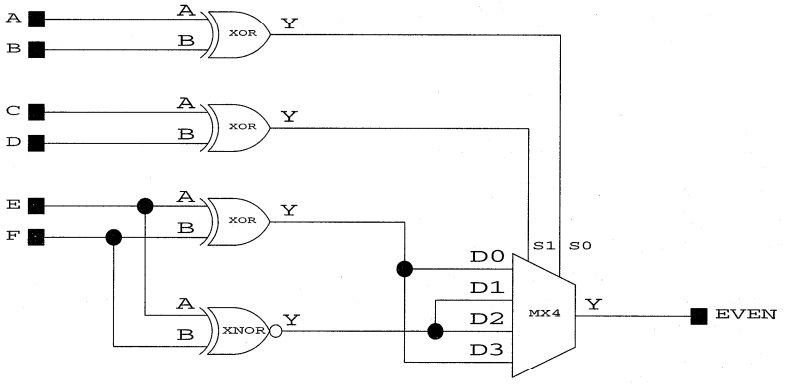


Figure 4 • Alternative Method of Generating Parity Over 6 Bits in Two Levels

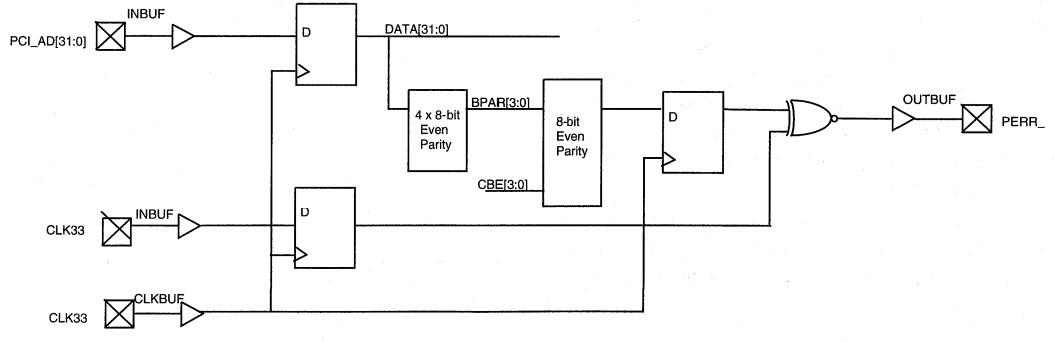


Figure 5 • Pipelining to Ensure 30 ns Clock Cycle

PCI State Machine

Figure 6 shows the inputs and outputs of the PCI state machine used in this design. It uses state-per-bit encoding techniques to minimize levels of logic. The PCI state machine consists of fourteen states. States are shown in Table 1 PCI state desctiptions are shown in Table 2.

Table 1 • States of the PCI State Machine

Memory Data Read Wait State	DR_WT
Memory Data Read Valid Data	DR_VL
Memory Data Read Turnaround	DR_TA
Memory Data Write Wait State	DW_WT
Memory Data Write Valid Data	DW_VL
Configuration Data Read Wait State	CR_WT
Configuration Data Read Valid Data	CR_VL
Configuration Data Read Turnaround	CR_TA
Configuration Data Write Wait State	CW_WT
Configuration Data Write Valid Data	CW_VL
Busy	BUSY
Abort Transaction	ABORT
End Transaction	END
Idle	IDLE

IDLE is the reset state, and all other conditions return to it, either by completing a transaction or by an error. State diagrams in Figures 7, 8, 9, 10, and 11 show the operation of configuration and memory transactions. Except for address latch, the design's outputs are dependent only on the state, which is known as a *Moore State Machine*. Address latch is implemented as a Mealy machine (the output is a function of the input and the current state) in order to save a clock cycle. State output values are shown in a Tables 3, 4, 5, 6, and 7 under each state diagram. IDLE, BUSY, ABORT, and END are identical for all cases.

When capturing data off the PCI bus and clocking it into the bridge, it would be reasonable to retime the data at the input buffers by using a registered input cell. However, due to the requirement to be able to respond to parity error violations within two clock cycles after the data appears on the PCI bus, it is not possible. Therefore, inputs from the PCI are not retimed. Address and data are instead clocked directly into their respective registers

In order to respond to PCI control inputs within the required clock cycles, the state machine was also situated so as to sense control right on the PCI bus. Generic bus control signals are generated and then merged in time with the data before it is sent to the generic bus. Likewise, responses from the generic bus must first propagate through the data pipeline path to the data latch block. The generic bus control (MEM_RDY_) is sensed there and used by the state machine to control TRDY_. The generic bus is further away in time from the state machine than is the PCI bus. This approach works in this simple example, but for more complex interfaces, FIFOs are often used to separate PCI and peripheral control and data flow issues.

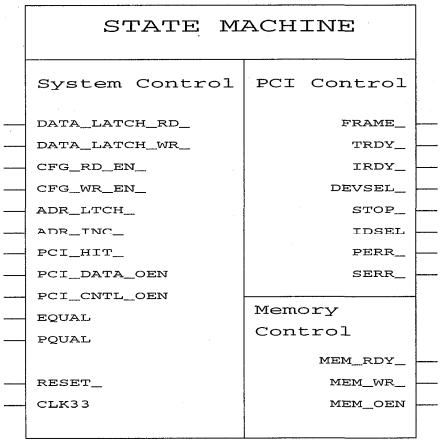


Figure 6 • PCI State Machine Inputs and Outputs

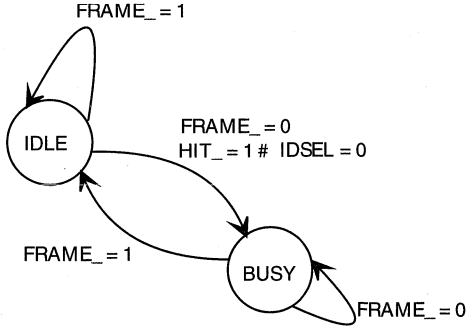


Figure 7 • Busy States

Table 2 • PCI State Descriptions

PCI Control	
FRAME_	Indicates the start and duration of a PCI cycle
TRDY_	Flow control signal used by PCI target
IRDY_	Flow control signal used by PCI master
DEVSEL_	Used by PCI device to signal that it has been selected
STOP_	Aborts PCI cycle
IDSEL	PCI select line used during PCI configuration cycles
PERR_	Indicates that a parity error occurred during data phase
SERR_	Indicates a system error; also indicates that a parity error occurred during an address phase
System Control	
DATA_LATCH_RD_	Latches data from generic bus into PCI interface
DATA_LATCH_WR_	Latches data from PCI into PCI interface
CFG_WR_EN_	Writes configuration data to selected configuration registers
CFG_RD_EN_	Reads selected configuration data; merges data into PCI write path
ADR_LTCH_	Latches PCI address
ADR_INC_	Increments burst counter
PCI_HIT_	Indicates that address on PCI bus is intended for this device
PCI_OEN	Enables PCI output drivers for PCI control
PCI_DATA_OEN	Enables PCI output drivers for address/data bus
SQUAL	Qualifies system errors
PQUAL	Qualifies parity errors
Generic Bus	
MEM_WR_	Writes data to generic bus
MEM_RDY_	Indicates valid data available on generic bus
MEM_OEN	Enables generic bus data output drivers

Table 3 • Busy State Output Values

	IDLE	BUSY
TRDY_	1	1
DEVSEL_	1	1
STOP_	1	1
DATA_LATCH_RD_	1	1
DATA_LATCH_WR_	1	1
CFG_LATCH_RD_	1	1
CFG_LATCH_WR	1	1
ADR_LATCH_	1	1
PCI_CNTL_OEN	0	0
PCI_DATA_OEN	0	0
MEM_OEN	0	0
MEM_WR_	1	1

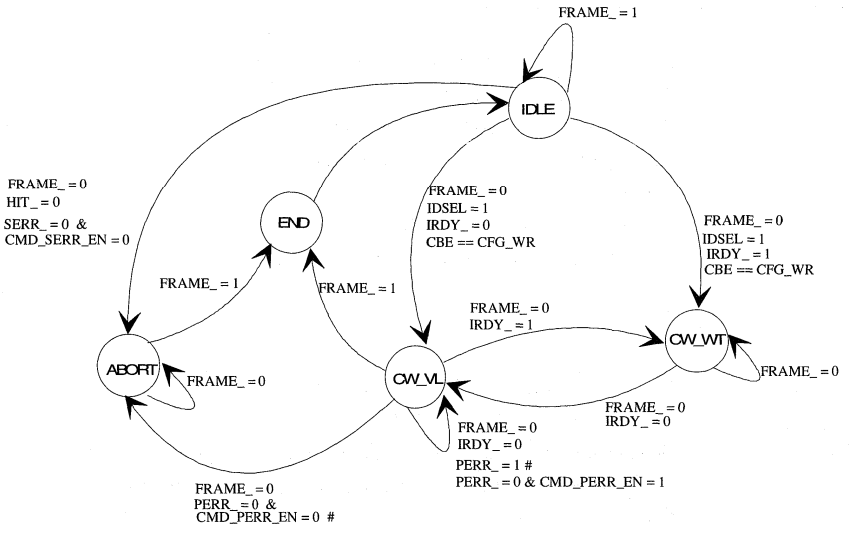


Figure 8 • Configuration Write States

Table 4 • Configuration Write State Output Values

	IDLE	CW_WT	CW_VL	ABORT	END
TRDY_	1	0	0	1	1
DEVSEL_	1	0	0	0	1
STOP_	1	1	1	0	1
DATA_LATCH_RD_	1	1	1	1	1
DATA_LATCH_WR_	1	1	1	1	1
CFG_LATCH_RD_	1	1	1	1	1
CFG_LATCH_WR	1	1	0	1	1
ADR_LATCH_	1	1	1	1	1
PCI_CNTL_OEN	0	1	1	1	0
PCI_DATA_OEN	0	0	0	0	0
MEM_OEN	0	0	0	0	0
MEM_WR_	1	1	1	1	1

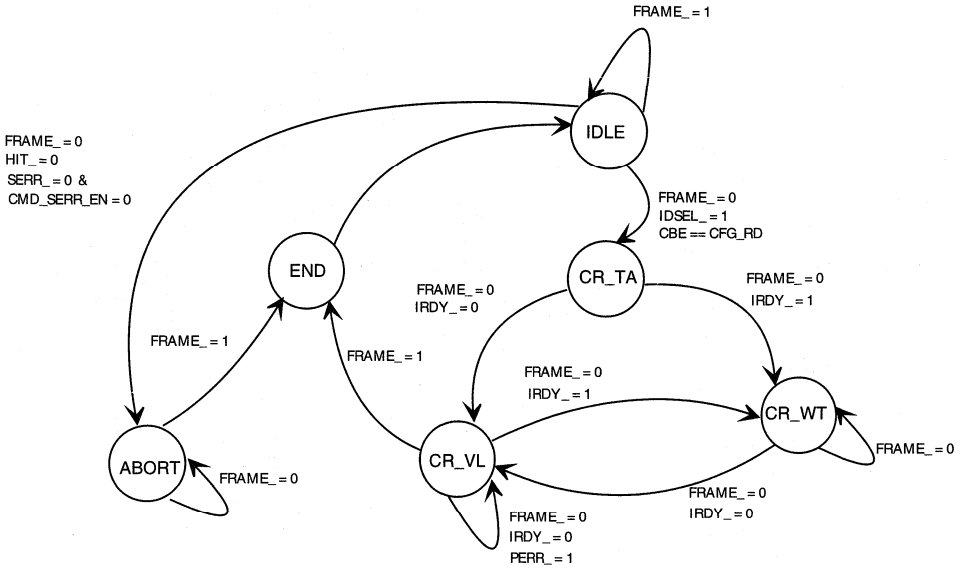


Figure 9 • *Configuration Read States*

Table 5 • *Configuration Read State Output Values*

	IDLE	CR_TA	CR_WT	CR_VL	ABORT	END
TRDY_	1	0	0	0	1	1
DEVSEL_	1	0	0	0	0	1
STOP_	1	1	1	1	0	1
DATA_LATCH_RD_	1	1	1	1	1	1
DATA_LATCH_WR_	1	1	1	1	1	1
CFG_LATCH_RD_	1	1	1	0	1	1
CFG_LATCH_WR	1	1	1	1	1	1
ADR_LATCH_	1	1	1	1	1	1
PCI_CNTL_OEN	0	1	1	1	1	0
PCI_DATA_OEN	0	0	1	1	0	0
MEM_OEN	0	0	0	0	0	0
MEM_WR_	1	1	1	1	1	1

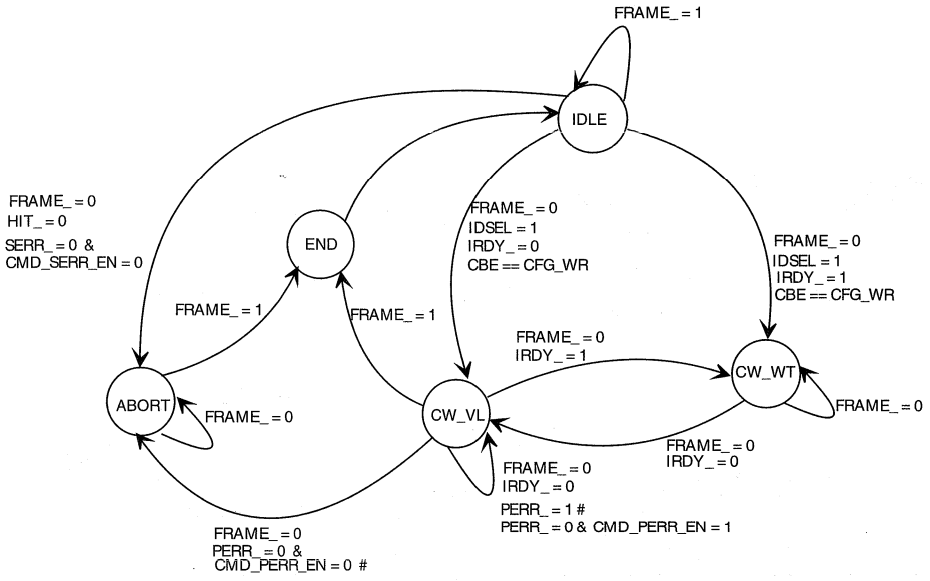


***Figure 10** • Memory Write States*

***Table 6** • Memory Write State Output Values*

	IDLE	DW_WT	DW_VL	ABORT	END
TRDY_	1	0	0	1	1
DEVSEL_	1	0	0	0	1
STOP_	1	1	1	0	1
DATA_LATCH_RD_	1	1	1	1	1
DATA_LATCH_WR_	1	1	0	1	1
CFG_LATCH_RD_	1	1	1	1	1
CFG_LATCH_WR	1	1	1	1	1
ADR_LATCH_	1	1	1	1	1
PCI_CNTL_OEN	0	1	1	1	0
PCI_DATA_OEN	0	0	0	0	0
MEM_OEN	0	1	1	0	0
MEM_WR_	1	1	0	1	1

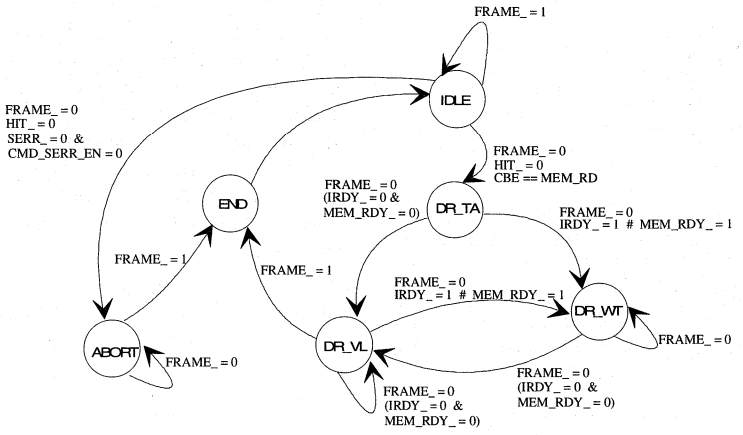


Figure 11 • Memory Read Status

Table 7 • Memory Read State Output Values

	IDLE	DR_TA	DR_WT	DR_VL	ABORT	END
TRDY_	1	0	0	0	1	1
DEVSEL_	1	0	0	0	0	1
STOP_	1	1	1	1	0	1
DATA_LATCH_RD_	1	1	1	0	1	1
DATA_LATCH_WR_	1	1	1	1	1	1
CFG_LATCH_RD_	1	1	1	1	1	1
CFG_LATCH_WR	1	1	1	1	1	1
ADR_LATCH_	1	1	1	1	1	1
PCI_CNTL_OEN	0	1	1	1	1	0
PCI_DATA_OEN	0	0	1	1	0	0
MEM_OEN	0	1	0	0	0	0
MEM_WR_	1	1	1	1	1	1

Configuration Registers

Table 8 shows all configuration registers implemented by the PCI to memory bus bridge. The PCI specification organizes the first 11 PCI registers in five 32-bit words (double words). All five registers are implemented; however, only a subset are used. A set of multiplexers is used to implement the logic. Since Actel logic cells lend themselves well to multiplexer based hardware design, this design style helps minimize hardware and levels of logic. Programmable bits are tied to flip-flops and static bits are hard-wired directly to VCC or GND. All bits not used are set to zero.

Figure 12 shows this arrangement. In addition to collecting the configuration registers, the second, four-input multiplexer allows configuration data to be merged onto the read-back path to the PCI bus.

All programmable configuration bits, whether status or command, are cleared by writing a 0 or a 1 as appropriate[5]. Addressing a configuration register is performed by selecting the device with PCI_IDSEL and applying the configuration register address on the PCI AD bus. Unlike memory transactions, configuration transactions can access only up to 256 bytes (64 double words).

Configuration register bits supported by the example design are shown in Table 8.

5. See "PCI Specification 2.0," Chapter 6.

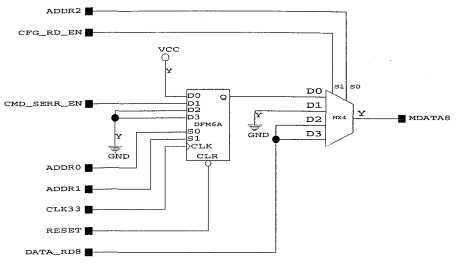


Figure 12 • Multiplexer Set Used to Implement Logic

Only one base address register is implemented in the example design. It defines the amount of address space required by this device as 1 megabyte. In addition, system software programs this register with an address. The bridge will respond to all memory read or write operations that are targeted to this address.

The most significant 12 bits of the base address register are programmable, and the 20 least significant bits are set to 0. PCI system software determines the number of least significant bits that cannot be programmed by writing 1s to the BAR and then reading them back. After the required address range has been determined, system software determines where in the PCI address space the device should be located. This address is programmed into the significant bits of the base address register.

Whenever a PCI transaction takes place, the device will compare its Base Address Register to addresses on the PCI bus in order to determine when a PCI hit has occurred.

Table 8 • Configuration Registers Implemented by the PCI to Memory Bus Bridge

Register		Example Design Values	Register Definition
Vendor ID	Read only	EF01 h	Vendor ID available from the PCI Sig group
Device ID	Read only	ABCE h	Code determined by Vendor of PCI device
Command	Read/Write	0000 000P 0P00 00P0 b	Allows control of PCI device
			Bit 1: Enables device to respond to PCI memory accesses
			Bit 6: Controls device's response to parity errors during data phase; high state allows parity errors to be reported
			Bit 8: Controls device's response to drive SERR_ line; high value on this line allows errors to be reported
Status	Read/Write	SS00 0010 0000 0000 b	Reports operating status to system software; cleared by write
			Bit 15: When set indicates a parity error during a data phase, even if parity handling is disabled by bit 6 in command reg
			Bit 14: When set indicates a parity error during an address phase
Revision ID	Read only	00 h	Value chosen by vendor
Class Code	Read only	00 08 00 h	Identifies generic function of device
Cache Line Size	Read only	00 h	Not implemented
Latency Timer	Read only	00 h	Not implemented
Header Type	Read only	00 h	Not implemented
BIST	Read only	00 h	Not implemented
Base Address Register 0	Read/Write	PPPPPPPP h	Identifies address range and location

Note: Hex or binary constants are read only. "S" identifies status bits from the PCI bridge. "P" identifies programmable command bits.

Electrical Conformance

The PCI bus electrical specification is designed to allow the use of drivers that don't have large current drive capabilities. This helps reduce ground bounce as well as the total package power dissipation. In order to achieve this objective, the bus is unterminated. Wavefront reflection is used to help boost the signal strength above the required 3.3 V 5 V thresh holds. However, factors other than current drive affect the performance of this bus. The length of the bus, the number of loads, and the length of stubs used to tap into the bus directly affect overall performance.

Initial HSPICE simulations on the ACT 3 family have produced results corresponding closely to those of PCI compliant drivers. Full compliance of the PCI AC specification (section 4.2.1.2) is currently under investigation using an A1440A-1 device programmed with this design in a PCI environment.

PCI to Memory Write Transaction

Figure 13 shows a PCI write operation. Address is latched into an address latch during the first clock cycle after FRAME_ goes low. If IRDY_ and TRDY_ are asserted low, then the first and successive data double words are latched into the data latch. During progressive phases, each long word progresses through the logic and out to the generic bus. If at any time IRDY_ or TRDY_ are deasserted, then the data flow halts. As long as FRAME_ is kept low, the transaction remains in progress. The transaction is terminated when FRAME_ goes high. The target can stop a transaction by asserting the PCI signal STOP_. In the bridge design, STOP_ is used as a response to a parity error during either the address phase or a write data to phase. The state ABORT asserts STOP_. ABORT can be invoked only when either PERR_ or SERR_ responses are enabled in the configuration command register.

PCI to Memory Bus Read Transaction

Figure 14 depicts a read operation. It initiates as was a write from the PCI bus. However, during a read, the PCI bus bidirectional drivers must be changed from inputs to outputs. In order to meet the electrical characterization of PCI, a special turnaround cycle is inserted between the address and first data phase. This extra cycle does not increase access latency, since the process of fetching from memory, and the progression of data through the interface pipeline structure will be longer than the turn-around cycle. During the TA cycle, the bus drivers are asserted into the three-state mode.

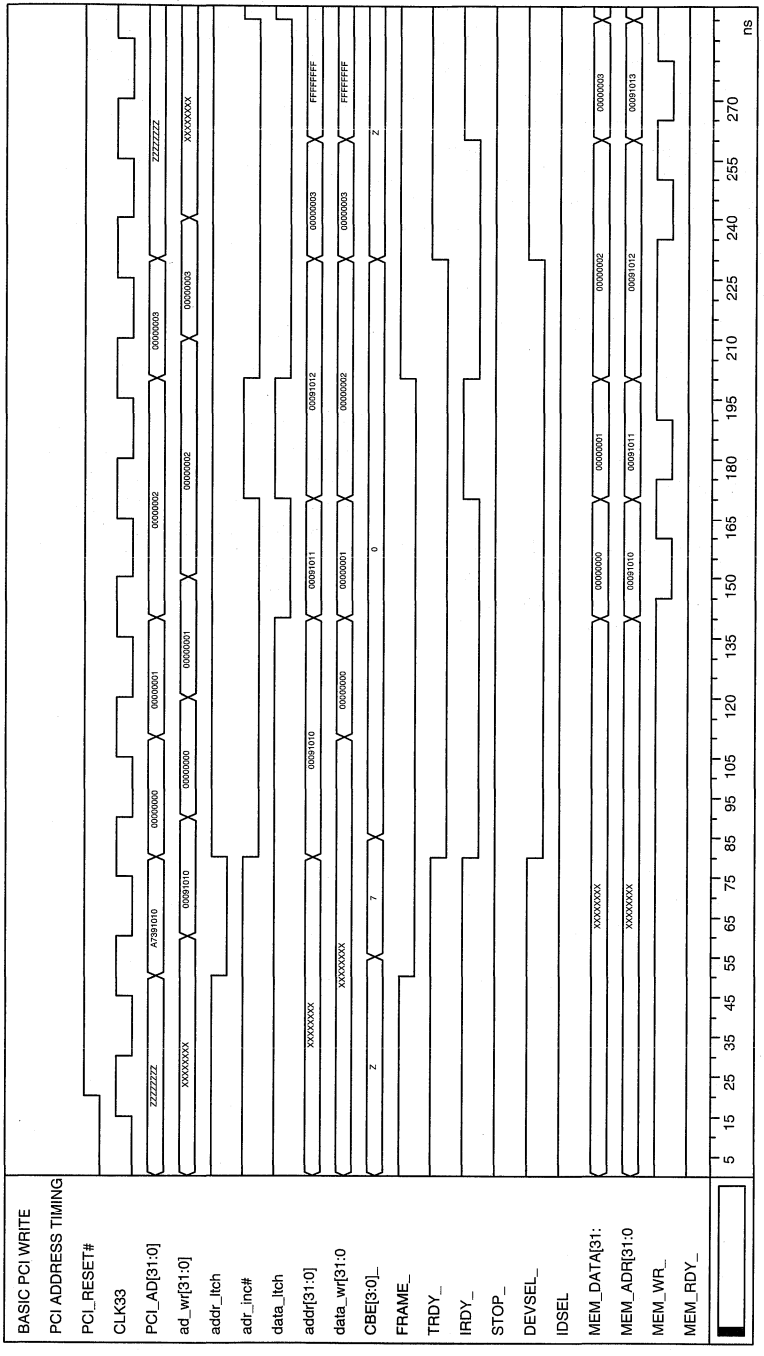


Figure 13 • PCI Write Operation Timing Diagram

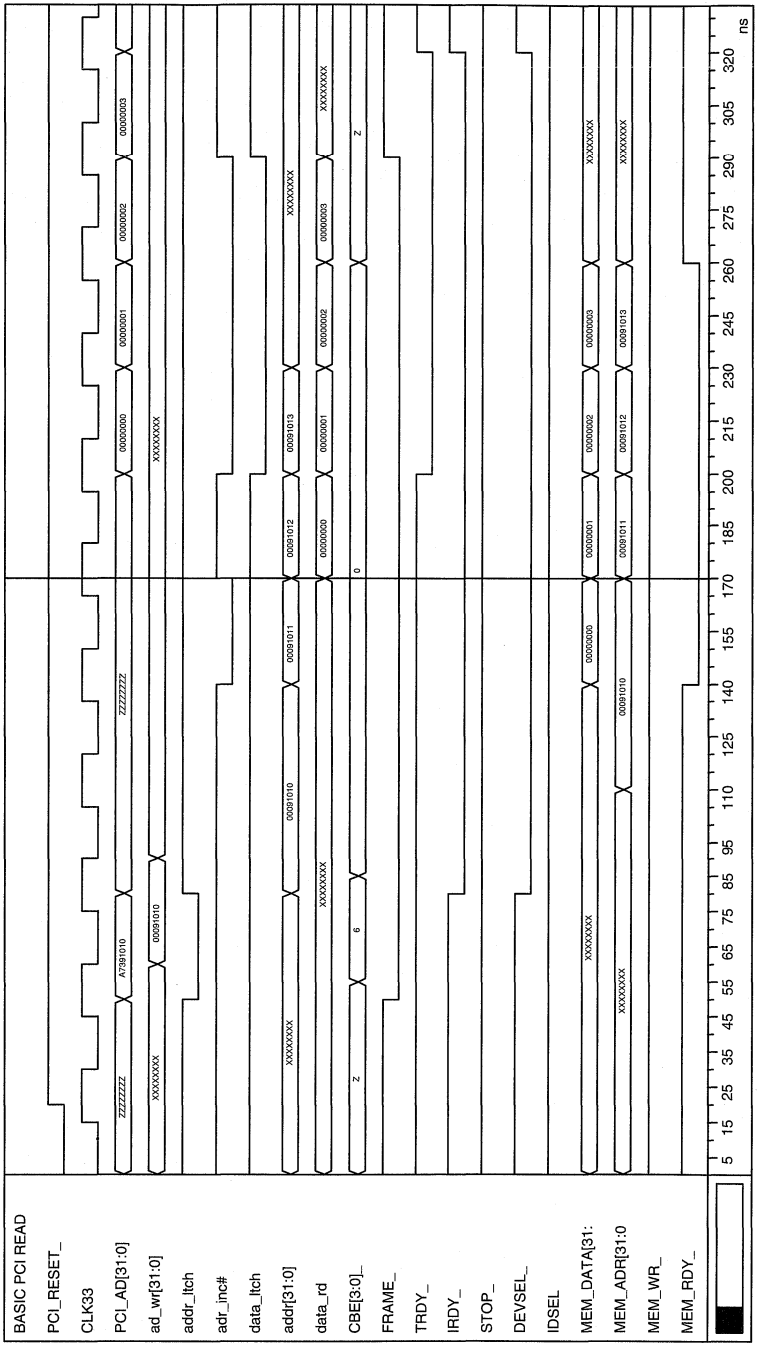


Figure 14 • Read Operation Timing Diagram

A High-Speed Graphics Processing Application Using FPGAs

Kirk A. Owyang, Staff Engineer
Actel Corporation
955 East Arques Avenue
Sunnyvale, CA 94086 USA

Introduction

Electronic hardware developers are continually driven by the need to create increasingly complex systems within short, fixed development schedules. CAE tool vendors and silicon manufacturers respond with modern development methodologies that include high levels of abstraction in design entry and intensive integration of target devices. Traditional design entry methods such as schematic capture are frequently being replaced by textual, behavioral circuit descriptions using high level languages; new Application Specific Integrated Circuits (ASICs) are continually being introduced to concentrate many sophisticated circuits into a single chip.

Trying to find better ways to quickly bring high performance products to market is not new. What is new is a design methodology combining the use of Field Programmable Gate Arrays (FPGAs) with high level design entry in high speed systems. New design flows use sophisticated synthesis tools to translate generic high level design descriptions into device specific netlists. Synthesis targets specific FPGA architectures and devices for optimum fit and performance. The result of these new methods? For designs requiring up to ten thousand gates, FPGAs are often the best solution for today's systems.

The computer graphics industry is a good showcase for the new design methodology. This dynamic field demands that companies respond quickly to market changes with fast, highly integrated systems full of the latest features. The following graphics processing application demonstrates the possibilities of high level FPGA design. It features an FPGA created from VHSIC Hardware Description Language (VHDL) input. It is targeted to an Actel ACT 3 family FPGA, the six thousand gate A1460A. With a 50 MHz clock speed, this example illustrates modern high performance system design in action.

System Description

This video graphics demonstration system employs an ACT 3 family A1460A FPGA as a system controller and central data processor. The FPGA fetches, processes and distributes video images. DIP switches control the rate of change for the image display. Then the system converts digital imaging information to RGB for display on a VGA monitor.

Figure 1 is a block diagram of the video graphics demonstration system. The system's main components are an Actel A1460A FPGA, a RAMDAC video palette, a one megabyte video RAM, a 50 MHz crystal oscillator, and a 512K EPROM. The detailed FPGA diagram shows the flow of data and control signals between each of the functional modules within the device. Each of these modules begins

9

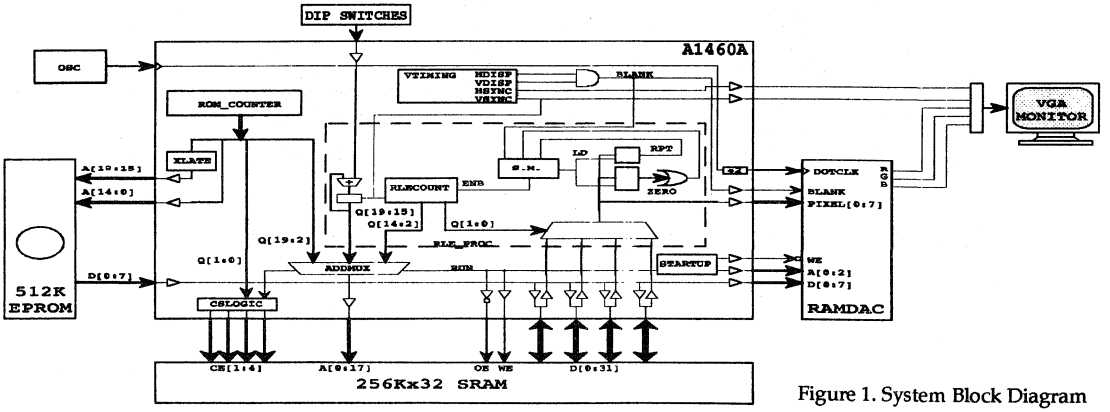


Figure 1. System Block Diagram

as a modular section of code in the VHDL device description. Synthesis software compiles and optimizes the code for a specific hardware target like the A1460A FPGA. What follows is a detailed description of the entire system; its video processor board is shown in Figure 2. A VGA monitor completes the system.

Video images for the demonstration system are permanently stored in the 512K EPROM. The system displays this information as a sequential set of 15 color images in forward and reverse order at variable speed. Since any image may be stored in memory, text messages and color pictures may be mixed in any sequence. New images are added by replacing or reprogramming the EPROM.

Before the images are fetched from memory, the A1460A initializes the RAMDAC video palette. The START_UP module (labelled "STARTUP" in Figure 1) initially stores the appropriate configuration data into the RAMDAC configuration register. Then, the module initializes the color look-up table (LUT) by setting the starting address (location '0') and the 256 24-bit red, green and blue (RGB) color values. The RGB values occupy the upper 1K addresses of the on-board EPROM. The next function uploads 15 compressed data images into the video RAM (VRAM) to finish initializing the system. The order of the images (numbered 0 to 14) in the VRAM also determines the display sequence. This particular design uses the following image sequence:

0 to 14, hold, 14 to 0, hold

This forward/reverse sequence creates the illusion of oscillating movement in the system. The ROM_XLATE module re-maps the 32 images (0 to 31) into the proper order. Simple modifications to the module's VHDL code change the sequence. After the VRAM is fully loaded, the system is ready to operate in the run mode.

The A1460A enters the run mode following initialization. In the run mode, it fetches and decompresses the 32 images from the VRAM in real time. Then the FPGA reconstructs each image from individual pixels and passes them to the RAMDAC. The images change at a variable rate set by on-board dip switches. The entire display sequence repeats indefinitely until the manual reset switch is activated. The core of the display function is the run-length encoded

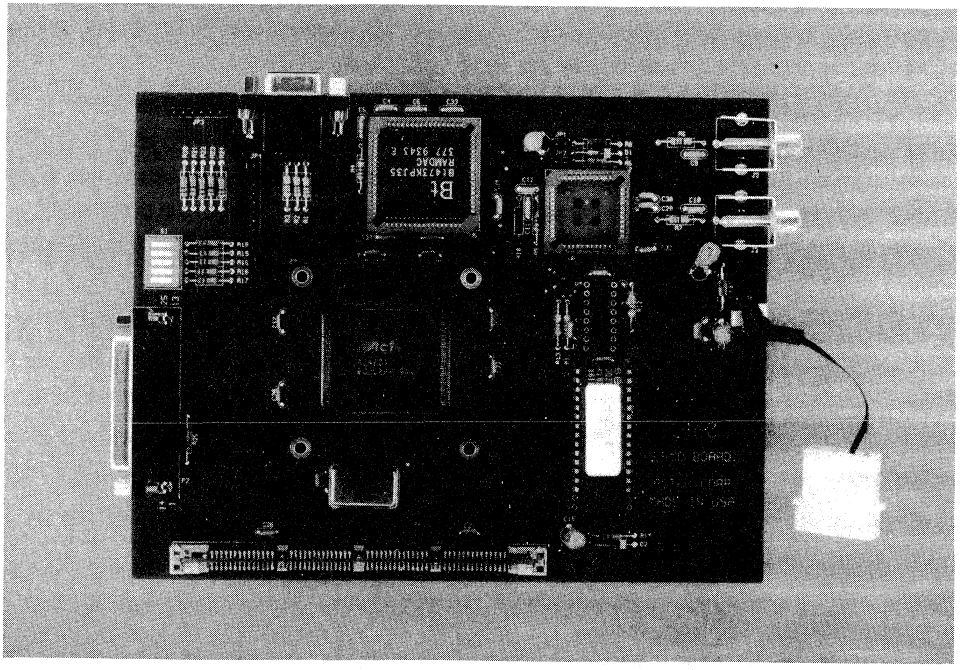


Figure 2. Video Processor Board

(RLE) processor (RLE_PROC in the block diagram). Several VHDL-encoded modules comprise the RLE processor. Detailed descriptions of each of these functional modules follow:

Run Length Encoded Processor. This is the core of the system. During active display times (when both VDISP and HDISP are active), it decompresses video images using RLE encoding. The RLE control byte is defined as:

Bit(s)	Function
0–6	Count (0–127)
7	Repeat (1 if next byte is to be repeated)

A simple RLE image is stored as a combination of control bytes followed by data bytes. For example, the control byte *REPEAT–50* followed by a single data byte results in 50 repetitions of the data byte. The control byte *non–REPEAT–50* followed by 50 random bytes displays 50 non–repeating bytes.

Address Multiplexor. This module multiplexes initialization data and run–time data to the VRAM.

Chip Select Logic for Video RAM. The VRAM read/write controller writes data into the VRAM in eight byte words during initialization. In run modes, VRAM outputs are enabled and only data reads are allowed.

Frame Incrementer. This module accumulates switch values for every vertical sync (1/60 second). The 5–bit output is used as the upper order address of the VRAM to sequence through the picture.

Horizontal Counter. This counter generates horizontal video control for display and function timing.

Vertical Counter. This counter generates vertical video control for display and function timing.

Modulo 5 Counter. This counter delays control signals for slower EPROMs (read times >200ns).

The VHDL code for some of these modules is appended to this paper for detailed review.

High Level FPGA Design

This application is a good example of a high level, top down approach to system design. In this case, VHDL code describes the behavior of the system, and synthesis software tools compile the code and create a specific mapping to an A1460A target device. Actel's Designer Advantage system reads the EDIF formatted output, checks for design rule errors, then automatically places and routes the design. Finally, a simulator reads detailed postlayout timing information from Designer for accurate backannotated system simulations.

High level FPGA design is not limited to a narrow choice of design approaches and software toolsets. Today, HDL entry may come in the form of VHDL and Verilog HDL. Actel provides technology libraries for synthesis toolsets such as ASYL+ (IST), CORE (Exemplar), AutoLogic (Mentor Graphics), and several different packages from Synopsys. In addition, Actel's own ACTmap FPGA Fitter (version 1.1) accepts VHDL entry and synthesizes optimized output for all three Actel device families. All of these software tools read HDL input and some also accept external drive, internal clock rate, output loading, and input signal arrival time information as guides for the synthesis engines.

ACT 3 Devices

The target device for this system is the A1460A housed in a 208 pin plastic quad flatpack (PQFP) package. Because of its architectural and process improvements, this third generation device runs this complex circuit at 50 MHz, necessary for computer graphics hardware systems. The family is based on a 0.8 µm double–metal CMOS process. Capacities for the five devices range from 1.5K to 10K gates and up to 228 usable I/O pins. The largest device (A14100A) has 1153 dedicated flip–flops.

Of special note is the external performance of this family. The chip–to–chip speeds make it possible to interface to high speed, 16 and 32–bit busses from microprocessors, memory chips, and other specialized devices. Since every ACT 3 I/O pad has dedicated input and output registers, on–chip and off–chip speeds are as fast as many masked gate array counterparts. The output registers deliver clock–to–out times of 7.5 ns for the A1425A–2 and 9.0 ns for the A1460A–2. The input registers require an input setup time of only 1.5 ns. Assuming a 1 ns PCB trace delay, the A1425A–2 device delivers 100 MHz chip–to–chip performance, 87 MHz for the A1460A–2.

The high internal performance of ACT 3 devices is proven in results certified by the Programmable Electronics Performance Corporation[1] (PREP). For example, the A1460A–2 datapath benchmark performance is 133 MHz under worst–case commercial conditions. The 16–bit pre–scaled counter benchmark has 130 MHz mean performance. In fact, the PREP results (as of July, 1994) for eight of nine benchmarks are the fastest for all high capacity (approximately greater than 5K gates) FPGAs.

Summary

This graphics processing application[2] shows that high level design methods using VHDL or Verilog HDL input with synthesis tools are an excellent choice for FPGA–based systems. The 50 MHz design integrates several complex, high signal count functions in a single FPGA, all described in VHDL code. The advantages of high level FPGA design are not limited to this industry. They apply to any market where time–to–market and ease–of–design are important. The benefits usually far outnumber and outweigh any risks associated with these design techniques.

1. For complete results, contact PREP at (408) 356–2169
2. Detailed information is available for this application from Actel Technical Support.

```
-- Address MUX
-------------------------------------
entity ADD_MUX is
  port(RUN      : bit;
       RUN_DATA  : bit_vector (17 downto 0);
       LOAD_DATA : bit_vector (17 downto 0);
       ADD_OUT   : out bit_vector (17 downto 0));
end ADD_MUX;

architecture dataflow of ADD_MUX is
begin
 with RUN select
   ADD_OUT <= RUN_DATA  when '1',
              LOAD_DATA when '0';
end dataflow;

-- Chip Select Logic for Video RAMs
-------------------------------------
entity CHIP_SEL is
  port(RUN, RAM_LD, MOD5 : bit;
       ADD              : bit_vector (1 downto 0);
       CSEL             : out bit_vector (4 downto 1));
end CHIP_SEL;

architecture dataflow of CHIP_SEL is
begin
 CSEL(1) <= not (RUN or ( not(ADD(0)) and not(ADD(1)) and RAM_LD and MOD5));
 CSEL(2) <= not (RUN or (     ADD(0)  and not(ADD(1)) and RAM_LD and MOD5));
 CSEL(3) <= not (RUN or ( not(ADD(0)) and     ADD(1)  and RAM_LD and MOD5));
 CSEL(4) <= not (RUN or (     ADD(0)  and     ADD(1)  and RAM_LD and MOD5));
end dataflow;

-- Frame Incrementer
---------------------
entity FRAME_INC is
  port(CLK, RST, ENB : bit;
       SWITCHES      : bit_vector (4 downto 0);
       FRAME         : out bit_vector (4 downto 0));
end FRAME_INC;

architecture behavior of FRAME_INC is
 signal sw_in : bit_vector (11 downto 0);
 signal accum : bit_vector (11 downto 0);
begin
 sw_in(11 downto 5) <= B"0000_000";
 sw_in( 4 downto 0) <= SWITCHES(4 downto 0);

 process(RST, CLK)
 begin
  if (RST='0') then
    accum <= B"0000_0000_0000";
  elsif (CLK='1' and CLK'event) then
    if (ENB = '1') then
      accum <= accum + sw_in;
    end if;
  end if;
 end process;

 FRAME(4 downto 0) <= accum(11 downto 7);
end behavior;
```

```
-- modulo five counter
----------------------------------------
entity MOD5_COUNTER is
 port (CLK, ENB, RST : bit;
                MOD5        : out bit);
end MOD5_COUNTER;

architecture behavior of MOD5_COUNTER is

 signal Q : bit_vector (2 downto 0);

begin
 process(RST, CLK)
  begin
   if (RST='0') then
                Q   <= B"000";
                MOD5 <= '0';
   else
                if (CLK='1' and CLK'event) then
                 if(ENB='1') then
                  if ( Q = B"100" ) then
                   Q <= B"000";
                  else
                   Q <= Q + B"001";
                  end if;

                  if ( Q = B"011" ) then
                   MOD5 <= '1';
                  else
                   MOD5 <= '0';
                  end if;
                 end if;
                end if;
   end if;
  end process;
end behavior;
```

A 64 MHz RISC Coprocessor Using the A1460 and VHDL Entry

Warren Miller
Product Planning Manager, Actel Corporation

Introduction

The Actel A1460 is the only Field Programmable Gate Array (FPGA) offering high capacity and high performance simultaneously. Additionally, the gate array architecture of the A1460 makes VHDL entry very efficient in terms of speed and capacity. This application note describes the implementation of a specialized coprocessor for a RISC CPU running at 64 MHz using VHDL entry.

RISC processors are used in a variety of high-performance applications, but even these speedy devices run out of processing power in some applications, or else the additional costs required to "pump up" the processor with faster memory systems or faster processor speed are prohibitive. In many cases, an FPGA can improve performance by off loading processor-intensive portions of an algorithm for direct hardware implementation. A RISC-based communications packet processor demonstrates this concept.

Communications Processor

In its simplest form, a packet processor is responsible for receiving data packets from a communications network, checking packets for correctness, and directing data to the correct port. A RISC processor could do all these functions, but matching the data rate would require a large portion of the processor's bandwidth. An FPGA can implement the packet checking portion of the algorithm and accelerate performance above that of a processor alone. The additional processor cycles thus made available can be used to improve performance of the higher level portions of the algorithm, adding intelligence and further improving throughput.

Figure 1 shows a block diagram of the communications processor. The RISC processor and the main memory provide the high-level control of the communications network. Packets received from the four communications lines are routed by the processor to the appropriate output port. The RISC processor establishes the connections between the input and output ports based on traffic levels and bandwidth limitations. It also measures traffic statistics to help determine the best solution of input and output ports. In addition, during reset and during times of low network activity, the processor supports diagnostics.

The FPGA implements the low level portions of the algorithm. Four data channels of 16 bits are received by the FPGA and four output ports are available for output data. The 16-word data packets are preceded by a command word that indicates the destination address of the packet and delineates the start of data. The FPGA counts each data word as it is received to ensure that the full 17 words exist in each packet. The 17th word is a simple CRC (an exclusive OR of each data word). The received data is selected for output at any of the four output ports.

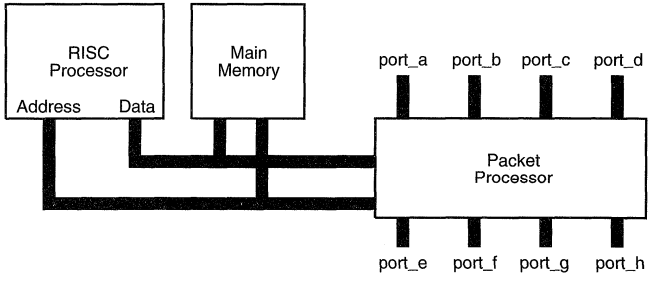


Figure 1 • *System Level Block Diagram*

Detailed FPGA Description

A block diagram of the packet processor FPGA is shown in Figure 2. The bus interface state machine manages the transactions with the RISC processor, thus allowing data registers to be read, status registers to be written to, and initialization commands to be executed. The packet processor state machine manages data flow. In normal operation, data flows from input to output without interrupting the RISC processor. If the number of data words is incorrect or a CRC error is generated, the RISC processor is informed of the error by the packet processor. The packet multiplexers are the main data path for the packer processor. Each of the four packet multiplexers implement the same functions. Input data is received by the packet multiplexer, command words detected, data words counted, CRC computed, and output data stream selected.

Referring to packet multiplexer A, input port A is the 16-bit input port, with associated data_valid, control_valid, and receive_valid signals. Input command words are separated from data words by the data_valid signal. Ports B, C, and D are the terminal versions of the data packet words that can be selected (in addition to A) by output port E by selecting signals S1 and S0.

FPGA VHDL Description

The complete packet processor design is described in a few pages of VHDL. Figure 3 gives the entire code for the packet multiplexer. The entity section defines the input and output signals of the packet multiplexer and the architecture section defines the operation of the packet multiplexer. Each of the two main parts of the design are described in separate block sections: input registers (inregs), output registers, the CRC register, and the shift register used to count data packets. The dffc and dff–v procedures call flip-flop routines from the library, simplifying the code for sequential elements (–v is used on a bit vector, 16 bits in this case). Notice also how the shift operation and the CRC operation are executed. These powerful operations require only a single line of code. As a result, a large amount of design work can be done very quickly in VHDL. The out_mux block implements the output select function so that any of the input ports can be routed to the output port.

The packet multiplexer description is called four times in the top-level design with different input and output signals to implement all four ports. The state machine section is added to complete the entire design. The design is targeted for the ACT 3 library and an Actel netlist is created. Figure 4 shows the results of running the complete design through the ACT'x'press software.

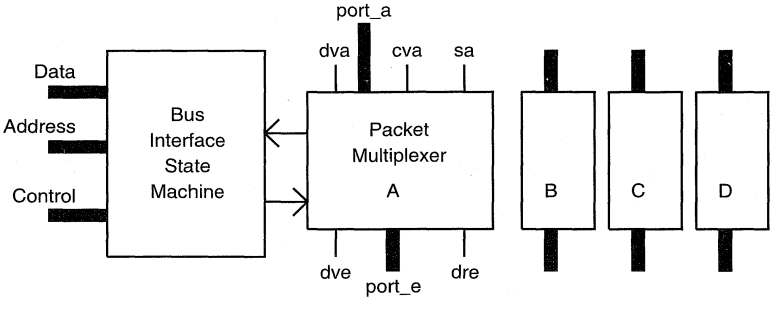


Figure 2 • System Level Block Diagram

```
library exemplar;
use work.exemplar.all;

entity pack_mux is
       port ( a_port, b, c, d : in bit_vector(0 to 15)
              e_port, a_reg : out bit_vector(0 to 15);
              pack_start, crc_error, pack_end : out bit;
              sa, dve : out bit;
              pack_addr : out bit_vector(0 to 1)
              clock, s2, s1, en_crc, cva, dva, clr, dre : in bit
       );
end pack_mux;

architecture packet of pack_mux is
       signal e, s, crc, s_reg, crc_reg : bit_vector(0 to 15)
       signal cvalid_reg, dvalid_reg, dre_reg : bit
       begin
       inregs : block
              begin
              dffc(cva, clr, clock, cvalid_reg);
              dffc(dva, clr, clock, dvalid_reg);
              dffc(en_crc, clr, clock, a_reg);
              dff_v(a_port, clock, a_reg);
              pack_start <='1' WHEN cvalid_reg = '1'1 and a_reg(15) = '0'
                     ELSE '0';
              pack-addr <= a_reg(1 downto 0)
              dffx_v(s, clr, clock, s_reg);
              s <= s_reg(14 downto 0) & '1' WHEN dvalid_reg = '1'
                     ELSE s_reg;
              ppack_end <= s_reg(15);
              dffc_v(crc, clr, clock, crc_reg);
              crc <= crc_reg xor a_reg WHEN en_crc = '1'
                     ELSE crc_reg;
              crc_error <= '0' WHEN crc_reg = x"0000"ELSE'1';
              end block;
       out_mux : block
              Begin
              dff_v(e, clock, e_port);
              dffc(dre, clr, clock, dre_reg);
              e <= e WHEN dre_reg = '1'ELSE
                     d WHEN (s1 = '1' and s2 = '1') ELSE
                     c WHEN (s1 = '1' and s2 = '0') ELSE
                     b WHEN (s1 = '0' and s2 = '1') ELSE
                     a_reg;
              end block;
       end packet;
```

Figure 3 • VHDL Source for Packet Multiplexer

FPGA Capacity and Performance Results

As indicated in Figure 4, the estimated performance of the design is 47 MHz (1/21.1 nanoseconds) and capacity of the design is 801 modules. The Actel place and route tool was run—the timer results for the longest path in the design are shown in Figure 5. The performance of the design is 81 MHz (1/12.3 ns), well over the goal of 64 MHz.

Conclusion

As shown in this example, a high-performance data path design is easily implemented using VHDL and the Actel A1460A. The high-capacity, high-speed, and fine-grained architecture of the Actel ACT 3 device family and the efficient results obtained from the ACT'x'press software make them ideal for time-to-market critical designs.

```
Exemplar Logic Synthesis System
Sun Jan 23 20:53:41 1994

Recourse Use Estimate

Design: pack_all
Technology: act3
File: PACK_MUX.VHD
Area: 801.0
Critical Path: 21.2 ns
```

Figure 4 • Synthesis Results for Packet Processor Design

```
; 1st longest path to $1I566:D (rising) (Rank: 0)
; Total    Delay    Typ      Load     Macro    Start pin        Net name
; 12.3     0.8      Tsu      0        DFC1B    $1I566:D
; 11.5     0.0      Tpd      1        AX1C     $1I315:A         PACK_MUX/CRC_0
; 11.5     8.7      Tcq      3        DF1      $1I549:CLK       A_REG0_internal
; 2.8      2.8      Psk      4                 $1I566:CLK       CLOCK_internal
```

Figure 5 • Longest Path Timing Results (CRC Register)

A High-Speed Address Search Algorithm Using FPGAs

Dennis McCarty, Consultant
Warren Miller, Product Planning Mgr
Actel Corp
955 East Arques Ave
Sunnyvale Ca, 95086 NC-055

[Abstract: A new family of FPGAs features synchronous I/Os with guaranteed clock-to-pad timing and fast internal logic. The performance of the family opens new applications to FPGAs. A brief description of the device architecture is given below followed by some applications that take advantage of device performance.]

The ACT3 Family

The ACT3 family of Field Programmable Gate Arrays (FPGAs) has been designed for the needs of high-performance product designers. Typically such designs require fast internal logic delays, predictable clock-to-out timing, and an abundance of powerful I/Os. The applications addressed by the family include such products as computers, LAN controllers and bus controllers.

Architecture

The family features a PLICE antifuse programming element in a 0.8u process. The logic is implemented using sequential and combinatorial logic modules similar to those featured in the ACT2 family.

The device includes four clocks, among them a dedicated I/O clock. The I/Os contain both an input and an output flip- flop with a guaranteed 10ns clock-to-pad performance. The output flip-flop Q also feeds back into the array allowing counters and state machines to be placed on the edge of the device and drive directly off the chip.

Applications

The performance and capabilities of the ACT3 family open new design opportunities for FPGAs that previously would have required a masked ASIC or discrete logic. The following sections describe some specific examples of applications that can be addressed by the ACT3 family.

Address Filter

A key requirement in LAN controllers is the ability to rapidly determine if a packet on the network is destined for a local node. A mechanism is needed to determine if the recipient of a message is a resident of a group.

A popular technique for comparing incoming addresses with a local address table is to use an address filter employing a binary search mechanism. The address filter determines if the incoming packet is destined for a member of the group by conducting the search of an SRAM containing a table of all the local addresses. As long as the search process does not exceed the time required to copy the packet (with the search outcome determining whether to retain or discard the packet) the search

won't entail any overhead to processing the packet.

The address filter was designed using an ACT3 FPGA filter controller and an SRAM address table is shown in Figure 1. The FPGA design consists of a 48-bit register, a 48-bit magnitude comparator, and an address generator state machine. The binary search mechanism relies on the SRAM address table to be loaded by the LAN processor with all the addresses in the node arranged successively from lowest at the least significant memory location to the highest at the most significant location.

Operation of the Address Filter

The search operates by successively dividing the address space in half each cycle until the entire table has been searched. The division is performed each cycle by always setting the address MSB below the and having the outcome of the comparison determine whether the current MSB remains set. The mechanism insures that the entire table can be searched in the number of cycles equal to the number of address bits it has.

The process begins when the destination address of the packet is loaded by the LAN processor in the 48-bit register. The load activates the state machine to assert the initial search address. That address divides the table in half having the MSB high and all other bits low. If the comparison output is equal it indicates an address match and the state machine notifies the LAN controller that the packet addressee is a member of the group.

If the comparison is not equal the search proceeds to the next cycle setting second MSB. If the comparison was that the packet address is less than the address at the center of the table, then the address MSB is cleared. If the comparison was that the address was neither equal to nor less than the packet address, then it must have been greater and the address MSB remains set.

The second cycle of the search thus begins in either the middle of the first quarter or the first three quarters of the space. The cycle is repeated until the address LSB is cleared or remains set. If the packet address is not found to be in the table after the last comparison, then the LAN processor is notified to ignore the packet.

State Machine Design

The design uses the guaranteed clock-to-pad delay of the ACT3 family by having one of the flip-flops in the I/O cell act as a state register for the state machine. That insures that state transitions will drive the new SRAM address off chip with a known delay.

The second I/O flip-flop is used to enable its companion to be cleared or remain set only on the transition where it tests the comparator's output. That allows all the state registers to determine the next state simply by sampling the comparator, simplifying the logic. The second I/O flip-flop thus acts as a shadow flip-flop chain that precedes the state chain by a cycle.

Using the I/O flip-flops as described above not only guarantees performance, it also frees internal resources for other logic.

Timing

The critical path of the filter is clock-to-pad delay to address the SRAM (10 ns), the SRAM access (10 ns), the delay from the pad through the comparator, and the setup to prepare the state machine for the next state (40 ns).

The period of that cycle determines the maximum frequency of the device and that frequency multiplied by the number of table address bits determines the slowest rate at which addresses can be located. The system can be clocked at approximately 17 Mhz.

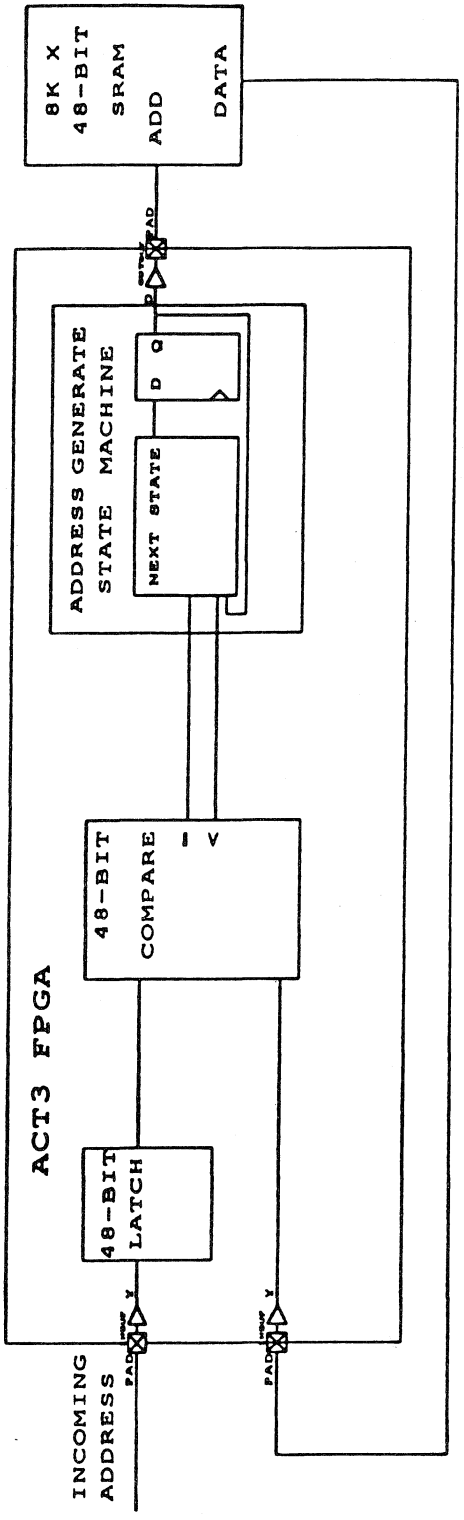


FIGURE 1. ADDRESS FILTER

A High-Performance Synchronous Memory Interface Using Actel FPGAs

Introduction

Synchronous memory interfaces are used in a broad number of applications that require high bandwidth access to large amounts of data. Graphics subsystems in particular can benefit from the use of synchronous memory architectures based on block transfers. What follows describes the design of the interface using a single Actel 1425 field programmable gate array (FPGA) running at up to 100 MHz.

Object Movement Subsystem

The graphics subsystem we discuss is the subsystem responsible for the rapid movement of a large number of objects within a window as illustrated in Figure 1. Any object may be translated by an (X,Y) coordinate to another location in the window. Each object is represented by an 18-bit word containing two tag bits, an 8-bit object number, and an 8-bit color value (objects may have the same color). The tag bits define the beginning and end of an object, as well as the existence of the object. This allows objects to be transferred in blocks even if they have irregular outlines. For example, the object in Figure 2 exists on 8 lines (only the first is shown) with the first line containing a start word, two object words, a non-object word, one more object word, and an end word.

The entire line can be moved in a contiguous transfer by beginning somewhere left of the start word, reading to the end word and writing the line out to final memory. During the line write the object will be copied beginning with the start word and ending with the end word. Only words with the tag bits set to an object will be written into. Notice that objects need not contain any words on a line if only a start and stop word are present.

Graphics Subsystem Architecture

Figure 3 depicts the hardware implementation of the subsystem. The processor loads the starting image memory independently from the FPGA by taking the FPGA off the bus during the load. After the image is loaded, the processor loads the FPGA with a series of transfer requests and grants the FPGA access to the memory. After the FPGA is done processing the transfer requests by copying the starting image memory, to the final image memory it interrupts the processor to inform it that the operations are completed. The processor can now access the final image memory and determine the results of the transfers. Notice that two separate physical memories are not needed to hold the starting and final images; the upper address bit can be used to select between starting and final images.

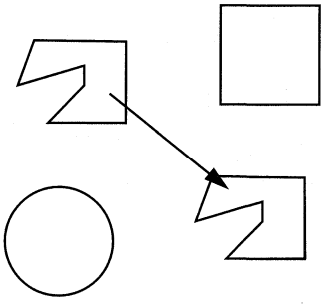


Figure 1 • Graphics Subsystem Window

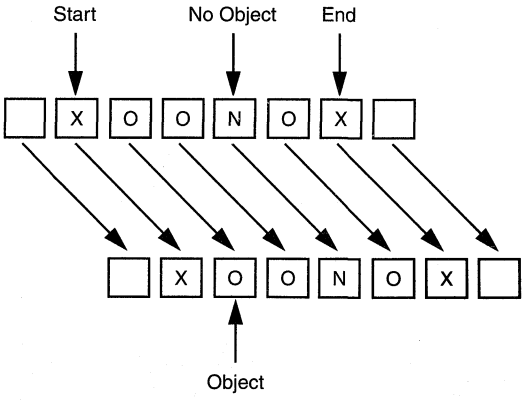


Figure 2 • Object Translation

Architecture of the Subsystem FPGA

Figure 4 shows the detailed block diagram of the FPGA portion of the design. The blocks making up the design are the object memory interface, the object stack, the processor interface, and the master control section.

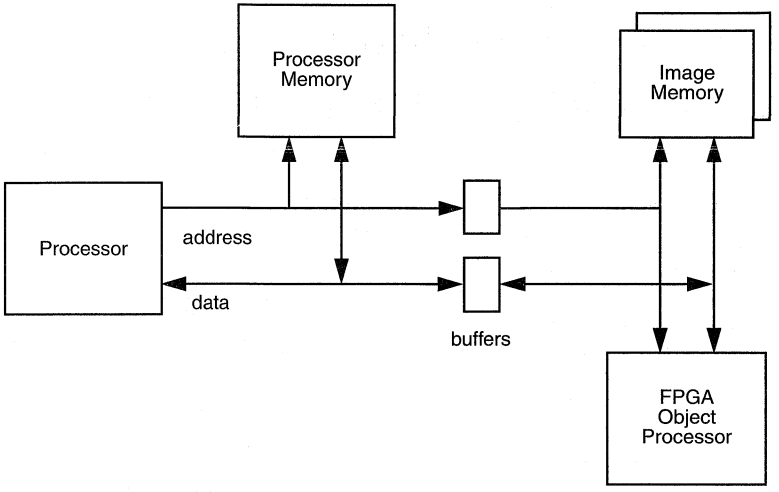


Figure 3 • Graphics Subsystem Hardware Implementation

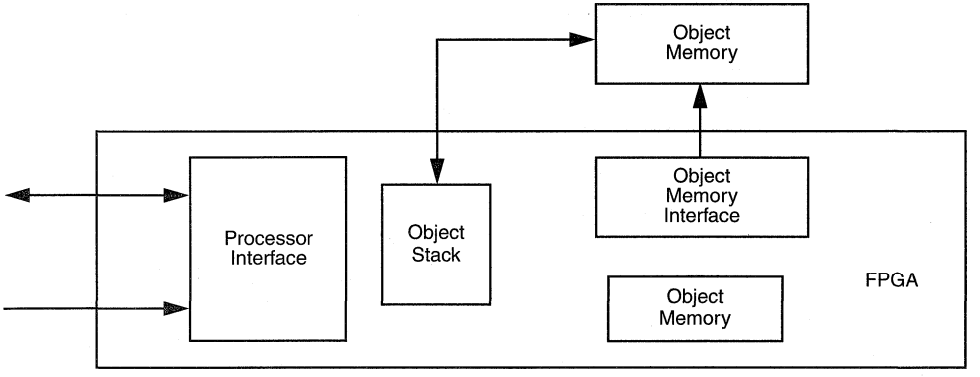


Figure 4 • FPGA Block Diagram

A major portion of the design is the memory interface that controls access to the object memory. The memory is organized as four banks, each of 16K words. This interleaved architecture allows four-word bursts at the faster output enable rate of the memory instead of the slower address access rate. This improves performance considerably in block transfer designs, but does require more memory devices (four times as many for four-way interleaved) than a noninterleaved scheme. The address for an object is thus divided into a 16K word address (14 address bits) and a 2-bit address within the block. Addresses are consecutive within the window on a line basis and wrap around after 256 words. Because both the starting and final image memories are contained in a single physical memory, the window size is 128 by 256. The more detailed diagram in Figure 5 shows the design of the interface.

The upper address bits to the memory are selected from the source and destination line counters as the object is moved from one location to another. A 6-bit address counter is used to count words on a line. A small state machine provides the chip selects and write-enables to the memory depending on the type of transfer (read or write) as well as the tag bits associated with the word. Address outputs are synchronous and are clocked on the 50 MHz system clock. Notice that the

WE and CS outputs are clocked on the 100 MHz I/O clock to generate the needed 2x cycle control signals.

The 4-word, 18-bit object stack, shown in Figure 6, holds the data during read and write bursts. Data is shifted into the stack on a read and then shifted out on a write. Tag information is used by the address generation portion of the interface to ensure that only valid object words are written into memory. Notice that the stack is organized as a LIFO (last in, first out) stack. This allows the first and last access of the object line to be unaligned with the four word interleave of the memory. For example, if the first access of an object contains only two words, the two words can be shifted in and then shifted out without needing to randomly access the words in the stack. This is a more efficient implementation in an FPGA architecture. Because of the LIFO nature of the stack, the data written to memory will be in the reverse order, and the write-enable signals are reversed to compensate.

The processor interface is shown in Figure 7. It contains a set of command registers that hold the move coordinates of a series of transfers. These registers are loaded by the processor on a bus separate from that of the image data so that object transfers can be processed concurrently with object translation operations. Status information to the processor is also available on this bus and is used by the processor to monitor progress on object transfers. The interrupt control block will inform the processor when the end transfer command is processed so that final image data can be accessed.

The master control section in Figure 8 contains the main state machine, which processes transfer commands, initializes the memory interface prior to a line transfer, and manages each object transfer. It holds starting and ending address pointers and determines when the last line of an object has been transferred by decrementing the object length counter until it reaches zero. The master control section is responsible for transferring objects, while the memory interface is responsible for transferring lines. The state machine is implemented in a high-level language using the "one-hot" methodology; it operates at 50 MHz.

Conclusion

This application note described the detailed design of a high-speed graphics memory interface using the Actel A1425. The interface operates at 100 MHz with a burst data transfer rate of 900 Mb/s (18-bits every 20 ns). The internal state machine controlling object transfers runs easily at 50 MHz and contains a variety of counters, registers, and random logic. The data stack stores a burst of four 18-bit words between transfers and can be clocked at over 100 MHz. Applications like this illustrate the types of designs this new generation of FPGAs can address, applications that were out of the reach of previous FPGA devices.

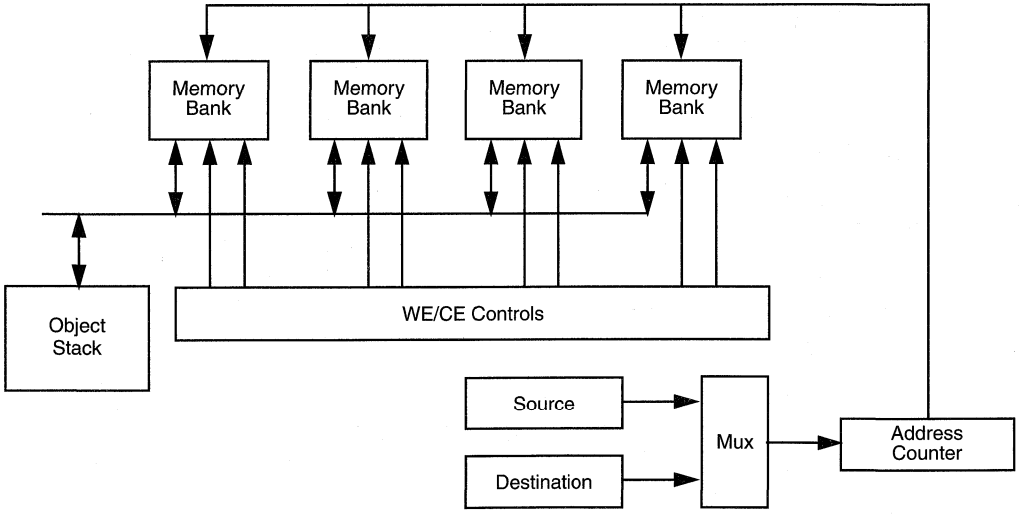


Figure 5 • Interface Design

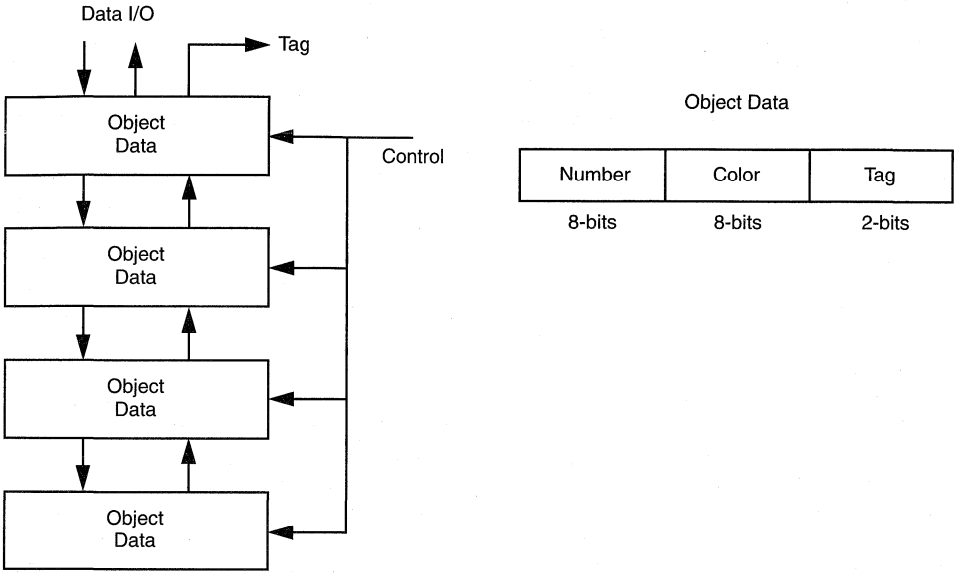


Figure 6 • Object Stack

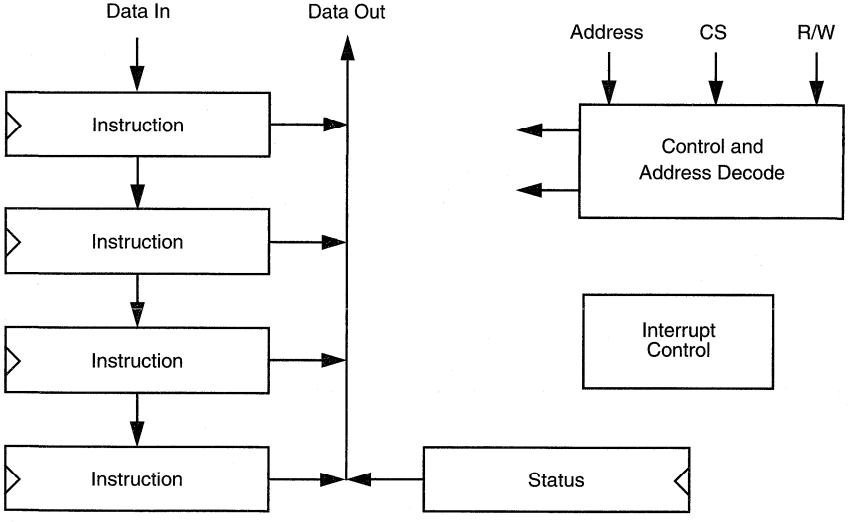


Figure 7 • The Processor Interface

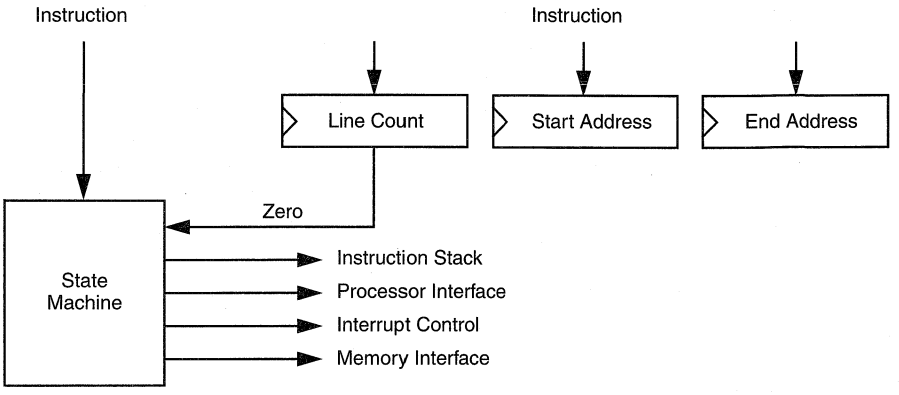


Figure 8 • Section of the Master Control

A Stepper Motor Controller in an Actel FPGA

Introduction

Stepper motors are electromechanical devices that provide accurate incremental rotation. Printer paper feeders, floppy drives, and robotic manipulators are popular stepper motor applications. The most common stepper motor uses four windings for a four-phase operation. Rotation is effected by actuating the phases in a specific sequence. Field programmable gate arrays (FPGAs) are ideal for integrating the control logic for these motors with other system control logic to minimize device count and board size.

Four-Phase Motor Circuit

A typical four-phase motor driving circuit is shown in Figure 1 using an FPGA to generate the sequence logic. The four windings have a common connection to the motor supply voltage (V_S) which typically ranges from 5 to 30 Volts. Each of the four phases is driven by a high power NPN transistor, since the FPGA cannot drive the motor directly. Each motor phase current may range from 100 mA to as much as 10 A. The transistor selection depends on drive current, power dissipation, and gain. The series resistors should be selected to limit the FPGA current to 8 mA per output. Power MOSFET devices can also be used to drive the stepper motor

Within the Actel FPGA, four inputs are required to fully control the stepper motor. The clock (CLK) input synchronizes the logic and determines the speed of rotation. The motor advances one step per clock period; the angle of rotation of the shaft will depend on the particular motor (the angle ranges from 1.8 to 90 degrees per step). To determine the clock period, consider that the stepper motor torque increases as frequency decreases. The direction (DIR) control input changes the sequence at the outputs (PH1 to PH4) to reverse the motor direction. The enable input (EN) determines whether the motor is rotating or holding. The active low reset input (RST) initializes the circuit to ensure that the correct starting sequence is provided to the outputs.

The basic control sequence of a four-phase motor is achieved by activating one phase at a time as shown in Figure 2. Figure 3 shows an enhanced sequence that uses an overlap technique with two phases active at any one time. The enhanced sequence provides increased torque but requires twice the current.

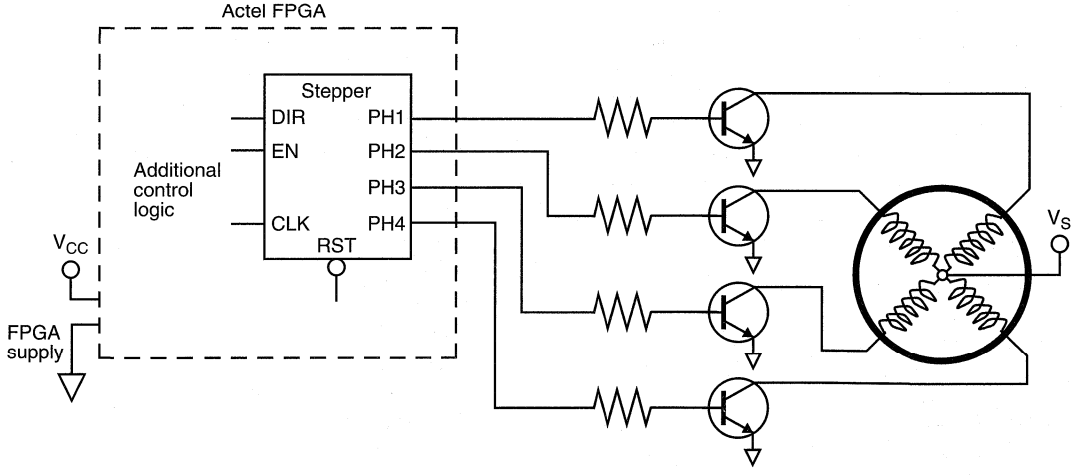


Figure 1 • Four-Phase Stepper Motor Driving Circuit

Step Sequence ->					
PH1	1	0	0	0	1
PH2	0	1	0	0	0
PH3	0	0	1	0	0
PH4	0	0	0	1	0

***Figure 2** • Basic Stepper Motor Sequence*

Step sequence ->					
PH1	1	1	0	0	1
PH2	0	0	1	1	0
PH3	1	0	0	1	1
PH4	0	1	1	0	0

***Figure 3** • Enhanced Stepper Motor Sequence*

Boolean equations using PALASM®2 syntax for the basic control sequence are shown in Figure 4. The phase equations (PH1 to PH4) are written with a colon and equal sign (:=) to indicate a registered implementation of the combinatorial equation. Each phase equation is either enabled (EN), indicating that the motor is rotating, or disabled (/EN), indicating that the current active phase remains on and the motor is locked. The value of the direction input (DIR) determines which product term is used to sequence clock wise or counterclockwise. The asynchronous equations (for example, ph1.setf = /rst) initialize the circuit. The Boolean equations for the enhanced motor stepper sequence shown in Figure 5 are simpler than the basic sequence. By inspection of the sequence from Figure 3, phases one and three are mere inversions of phases two and four. Therefore, two phase equations can be drastically reduced (ph1 = /ph2, ph3 = /ph4). Also, the next sequence for each phase is only dependent on one other phase and not all four, thereby reducing the number of terms required for the remaining phase two and four equations. Note that inverting phases one and three provides the correct initial sequence values.

```
CHIP step2 ACT
clk en dir rst ph1 ph2 ph3 ph4
EQUATIONS
ph2 := en * /dir * ph4 + en * dir * ph3 + /en * ph2
ph4 := en * /dir * ph1 + en * dir * ph2 + /en * ph4
ph2.rstf = /rst
ph4.rstf = /rst
ph1 = /ph2
ph3 = /ph4
```

***Figure 5** • Boolean Equations for Enhanced Stepper Motor Sequence*

```
CHIP step1 ACT
clk en dir rst ph1 ph2 ph3 ph4

EQUATIONS

ph1 := /dir * en * (/ph1 * /ph2 * /ph3 * ph4)
     + dir * en * (/ph1 * ph2 * /ph3 * /ph4)
     + /en * ph1

ph2 := /dir * en * ( ph1 * /ph2 * /ph3 * /ph4)
     + dir * en * (/ph1 * /ph2 * ph3 * /ph4)
     + /en * ph2

ph3 := /dir * en * (/ph1 * ph2 * /ph3 * /ph4)
     + dir * en * (/ph1 * /ph2 * /ph3 * ph4)
     + /en * ph3

ph4 := /dir * en * (/ph1 * /ph2 * ph3 * /ph4)
     + dir * en * ( ph1 * /ph2 * /ph3 * /ph4)
     + /en * ph4

ph1.setf = /rst
ph2.rstf = /rst
ph3.rstf = /rst
ph4.rstf = /rst
```

***Figure 4** • Boolean Equations for Basic Stepper Motor Sequence*

The Boolean files were synthesized and optimized for the Actel ACT™ 2 family. Modules utilized for the basic and enhanced sequences were 14 and 6 modules respectively. Figures 6 and 7 are the schematic representations of the basic and enhanced sequencing circuits respectively. Note that these schematics do not show the output I/O macros (OUTBUF) that are needed to connect to the external circuit (that is, outside the FPGA).

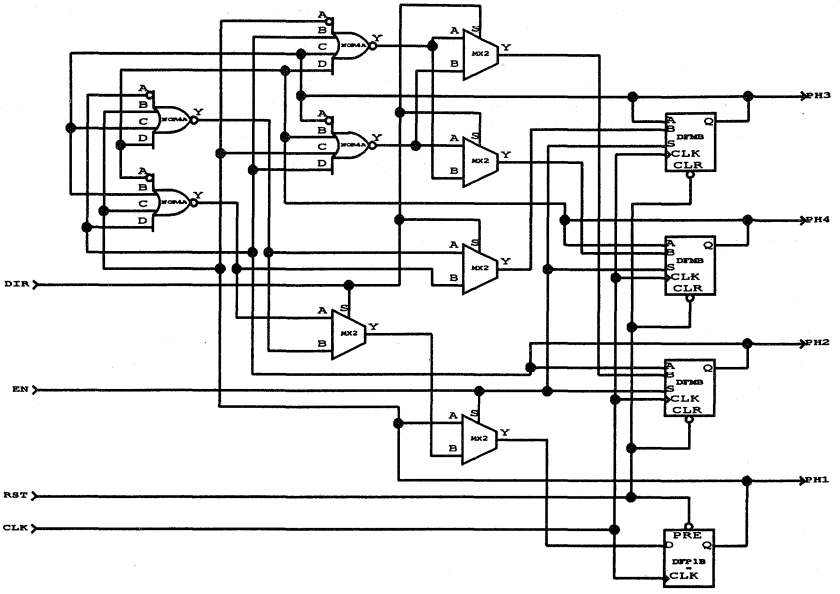


Figure 6 • Basic Stepper Motor Sequence Schematic

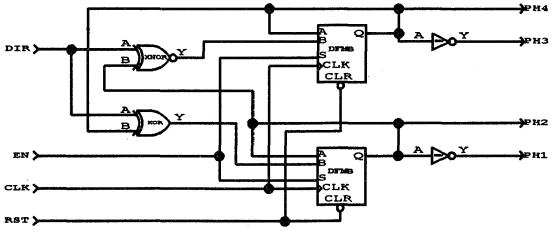


Figure 7 • Enhanced Stepper Motor Sequence Schematic

The timing diagrams in Figures 8 and 9 show the operation of the two stepper motor sequences. Reset, enable, and direction inputs are exercised for a clear understanding of full circuit operation. Additional logic can be added to control the enable and direction inputs. For example, a pre-loadable down counter with a zero detect output connected to the stepper motor enable input can be used to rotate the motor a predetermined number of steps.

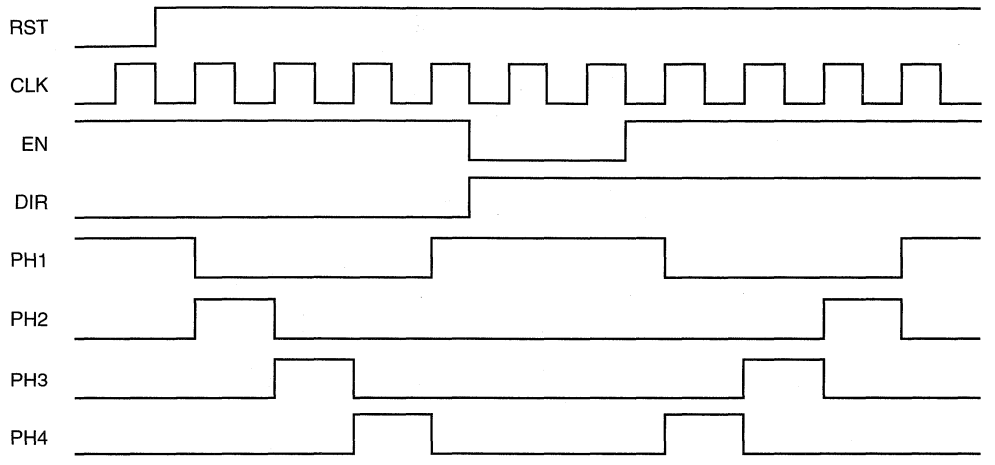


***Figure 8** • Basic Stepper Motor Sequence Timing Diagram*

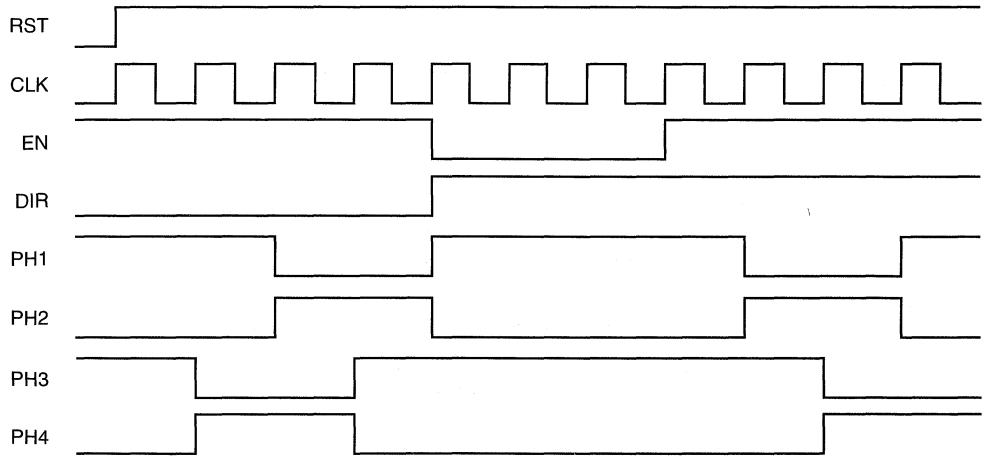


***Figure 9** • Enhanced Stepper Motor Sequence Timing Diagram*

Guidelines for Optimal FPGA Designs

Introduction

Actel field programmable gate arrays (FPGAs) offer many benefits including high levels of integration and low cost. However, to maximize these benefits, the resources available within the devices must be used in an optimal fashion. To achieve this goal, a basic understanding of the underlying technology and implementation techniques is important. The goals of this application note are first to describe architectural features of Actel devices that can affect density and then based on this knowledge, to describe design techniques that can help improve the density of the devices. All families available from Actel are discussed.

Architectural Features Affecting Device Density

The resources available in Actel devices can be broken down into three main categories: internal logic modules, I/O modules, and global clock resources. Internal logic modules, which are based on multiplexers, perform the primary logical functions of the device. The multiplexer was selected because of the wide variety of combinatorial functions that can be created from this basic structure.

I/O modules provide the interfaces to other devices within a system and may have some logic content, depending on the family. Each I/O can function as an input, output, bidirectional buffer, tristate output, or open collector output. In addition, some families have latches or flip-flops available in the I/O module.

Global clock resources are internal high drive (or high fanout) components that are typically used as clocks but that can also be used for other high fanout functions. The number of global clock resources varies by family. If more clocks are required for an application than are available, additional clocks can be built from internal logic-module resources.

The following sections break down each of these resources by family.

ACT 1 Resources

The ACT 1 family consists of a single type of internal logic module, which is shown in Figure 1. Since the logic module has no built-in sequential (latch or flip-flop) elements, sequential circuitry must be built by feeding the output back to one of the module inputs, as shown in Figure 2. Latches can be built with a single module; flip-flops require two modules (master-slave latch combination). The ACT 1 logic module can generate basic functions up to four-input basic gates (AND, OR, NAND, NOR); however, OR functions can be implemented more efficiently than AND functions because of the additional OR circuitry in the module.

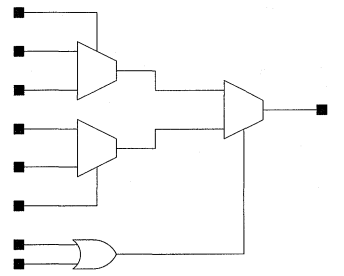


Figure 1 • The ACT 1 Logic Module

The I/O structure for ACT 1 is shown in Figure 3. The ACT 1 I/O has no associated logic and serves only as a buffer to external devices.

The ACT 1 family has a single global clock buffer. This buffer can be accessed only from an off-chip source through an I/O location specified on package drawings.

ACT 2 and 1200XL Resources

Many modifications to the ACT 1 architecture were made to create the ACT 2 and 1200XL architectures. The goal of the changes was to improve both density and performance. The primary difference is in the structure of internal logic modules. The modules were modified to accommodate a sequential element in about one-half of the total internal logic modules. These modules are referred to as *S-modules* because of the built-in sequential function, but they still contain a combinatorial portion, as shown in Figure 4. The other half of the internal logic modules are referred to as *C-modules* because they contain only combinatorial logic. (See Figure 5.) C-modules in ACT 2 and 1200XL were also modified so that both AND and OR terms could be generated with the same efficiency. Also, some five-input basic gates can be built from a single module. S-modules can be used to create latches, flip-flops, and combinatorial-only functions equivalent to a C-module. The active low clear is hard-wired on the S-module.

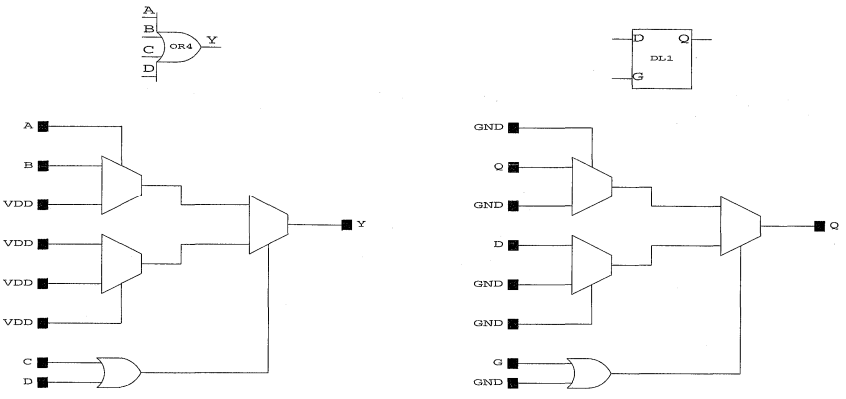


Figure 2 • Combinatorial Sequential Implementations Using the ACT 1 Logic Module, Showing Both the Library Element and the Actual Wiring of the Module

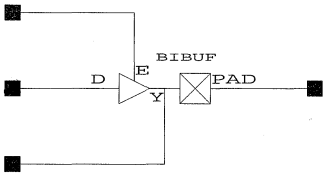


Figure 3 • The ACT 1 I/O Module

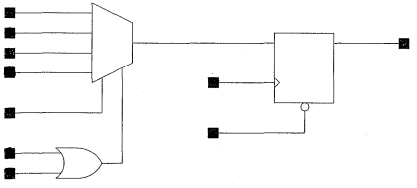


Figure 4 • The ACT 2 S-module

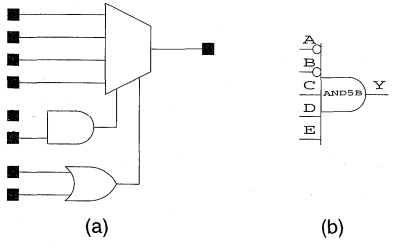


Figure 5 • The ACT 2 C-module (a) and One Possible 5-input Function (b)

Most library elements that can be built from a C-module can be mapped into the combinatorial section of an S-module. When the software tools encounter a situation in which a combinatorial function drives the data input on a basic D-type flip-flop or latch, the software will automatically combine the functions into a single S-module. This act of logic compression is referred to as *combinability* and is executed in the place and route tools by a function called the *combiner*. See "Rules for Combinability" for a more complete description of this function.

The external I/O were also modified to accommodate a latch in both the input and output portions of the I/O pad, as shown in Figure 6. These functions were added to improve on- and off-chip delays, compared with ACT 1.

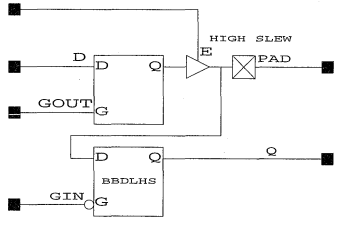


Figure 6 • The ACT 2 I/O Module

A second global clock resource was added in the ACT 2 and 1200XL families. These clock resources can be accessed either from an external I/O or from an internal source.

ACT 3 Resources

Minor modifications were made to the ACT 2 and 1200XL families to create the ACT 3 family. The first modification was to enhance the S-module so that it contained the same combinatorial function as a C-module, which enhances combinability. (See Figure 7.)

Modifications were also made in the I/O modules. The ACT 3 I/O modules contain flip-flops with enable, which significantly improves on- and off-chip performance compared with previous families. (See Figure 8.) The logic in the I/O can account for as much as 20 percent of the total device logic in the smaller ACT 3 parts. This means that effective use of the I/Os in ACT 3 is very important.

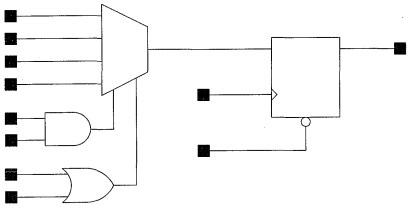


Figure 7 • The ACT 3 S-module

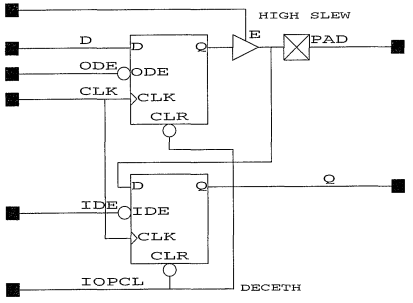


Figure 8 • The ACT 3 I/O Module

The ACT 3 global clock resources were also enhanced to add two additional clocks, which both must be accessed from external pins. One clock is a low-skew, high-frequency clock used to drive all S-modules in the device. The second clock is a low-skew, high-frequency clock used to drive the clock inputs of all I/O modules. These clocks are in addition to the routed clocks available in the ACT 2 and 1200XL architectures.

Guidelines for Optimization

Many of the guidelines used to improve the density of Actel devices are common to all families, and some are family specific. The family-specific guidelines are described in the following sections. The goal of these rules is to put the most function in the fewest modules possible. General guidelines applicable to all families are as follows:

- Use global clock resources to drive clock inputs. If unused global clocks are available after clocking requirements have been satisfied, these signals can be used to drive other high fanout signals. Examples of high fanout signals are flip-flop enables and asynchronous clears.
- Use buffers to control fanout only as needed. The Actel software generates many warnings about high fanout. The purpose of the information is to provide the designer with feedback indicating that a high fanout situation exists and will probably result in longer delays than would other logic. If this is not a concern, then the designer should simply ignore these warnings.
- An alternative to using buffers to control fanout is to use logic duplication. The goal of logic duplication is to replicate a function to distribute fanout by using fewer modules than the buffered equivalent. An example is shown in Figure 9.

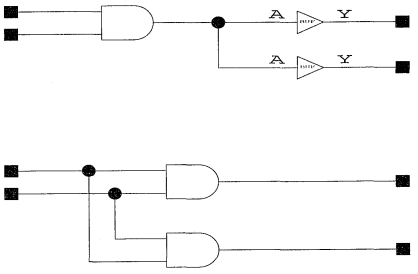


Figure 9 • Fanout Correction Using Buffering and Logic Replication

- Analyze library elements to ensure the best fit. This pertains to both hard macros (single or dual module macros) and soft macros (multiple module macros). Figure 10 illustrates the importance of hard macro module selection for consolidating logic in fewer modules. Soft macros are built as a convenience to designers; however, the soft macros often have more functionality than may be required for a specific application. The additional functionality is often the result of additional logic resources, which can be eliminated by an alternative soft macro selection or modifications to the soft macro.
- Apply De Morgan's Law to reduce combinatorial logic expressions to their least complex forms. Reducing the logic equations reduces logic module use and complexity. In many cases, macros with fewer inputs can be used, as a result of simplier logic equations. The overall routability usually increases as a result of simplifying the design's implementation.

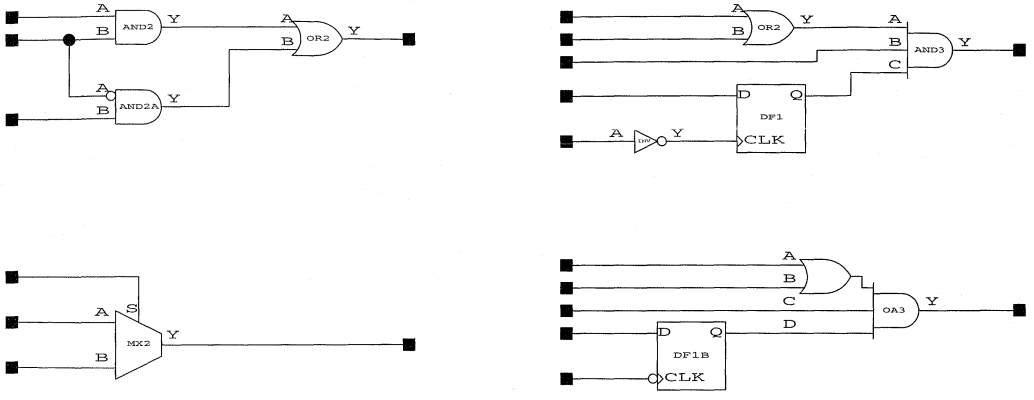


Figure 10 • Examples of Hard Macro Selection Options (The top circuits represent suboptimal implementations with the optimal implementation shown below.)

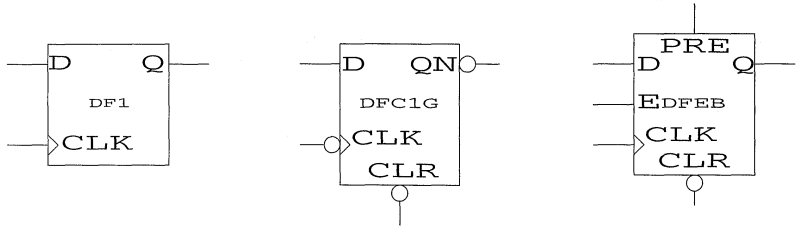


Figure 11 • Examples of Available ACT 1 Flip-flops

Guidelines for ACT 1

The ACT 1 family is unique, because all sequential functions must be built from combinatorial logic. To get the most out of the ACT 1 family, the following guidelines should be used:

- Whenever possible, use a latch rather than a flip-flop for a storage element to save logic modules. (Flip-flops require two modules; latches require only one.)
- Many forms of flip-flops exist because of the dual module implementation. To eliminate the need for additional resources, always look for the flip-flop that contains all of the function required. See Figure 11 for an example.
- Use the muxed flip-flops when possible. Since a 2:1 mux can be used to generate basic two-input gates, the use of muxed flip-flops can occasionally save a logic module as well as a logic level. See Figure 12 for an example.
- Use active low decoding when possible. Active low decoding is more effectively mapped into ACT 1 logic modules than is active high decoding (because of OR term efficiency). For example, downcounters can be built much more efficiently in ACT 1 than upcounters can be because 0 s are decoded rather than 1s.
- The CM8A macro represents the complete function of a logic module. Use this macro with appropriate tie-offs to VCC and GND to implement functions not available in the Actel Library, if such mapping is possible.

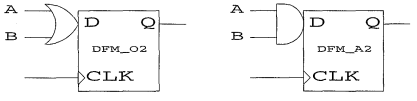


Figure 12 • Basic Gate Construction Using the Muxed Flip-flip Macros in ACT 1

Guidelines for ACT 2 and 1200XL

The ACT 2, 1200XL, and ACT 3 families are more complex than the ACT 1 family because of the addition of the sequential module and combinability. Use the following guidelines to ensure optimal density:

- Use only combinable flip-flops and latches (e.g., DF1, DFC1B, DF1A, or DFC1D), or use precombined functions (DFMB, DFM6A, etc.) to maximize efficiency. When combinatorial functions are driving combinable flip-flops, check library information to ensure the macro is combinable. See Figure 13 for the symbols of these macros.
- The DFM7A and DFM7B macros represent the complete sequential module for ACT 2 and 1200XL. Use these macros with appropriate tie-offs to VCC and GND to implement functions not available in the Actel Library, if such mapping is possible.
- Avoid asynchronous presets or active high clears unless they are essential to circuit function. Both of these functions require two modules to create, and both are noncombinable.
- Use latches available in I/O modules, if possible, to save on internal latch resources.
- For circuitry that is predominantly combinatorial, the S-module resource can be exhausted before the C-module resource. In these cases, certain sequential macros can be selected from the library. These macros are built from only C-modules (similar to ACT 1 sequential macros) to expand sequential resources in the device. Examples are the DFPC and DLC1, shown in Figure 14.

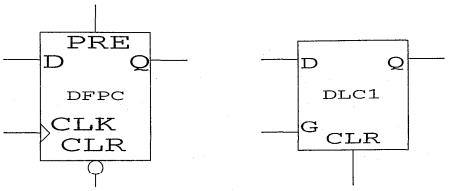


Figure 14 • Examples of C-module Sequential Elements for ACT 2 and ACT 3

Guidelines for ACT 3

The ACT 3 family is similar to the ACT 2 and 1200XL families, so the following guidelines should be used in addition to those above:

- Any macro that can be built from a single C-module is combinable. This feature eases the library selection process for the combinatorial functions to ensure combinability.
- It is important to use the registered I/Os when possible to help improve density. The registers in the ACT 3 I/O are much more flexible than those in ACT 2 and 1200XL, and they account for a significant percentage of available logic resources.
- The DFM8A and DFM8B macros represent the complete sequential module function. Use these macros with appropriate tie-offs to V_{CC} and GND to implement functions not available in the Actel Library, if such mapping is possible.

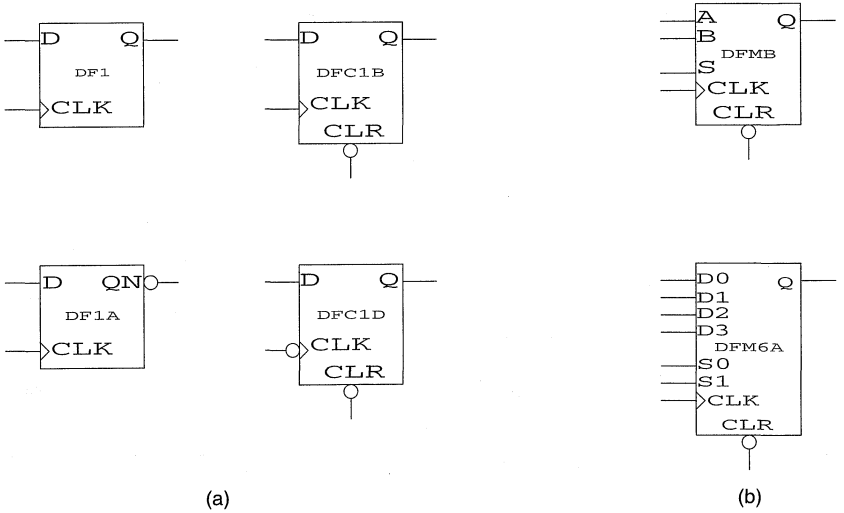


Figure 13 • Example of Noncombinable (a) and Combinable (b) Macro Selection to Implement Identical Functions

Place and Route Optimization Tools

Previous sections of this application note have been targeted at improving density by manual methods. Additional software functions that are part of the place and route tools also aid the designer in the optimization process. One of these functions, the combiner, has already been discussed.

The combiner has another function, which is stripping unused logic functions from the netlist. It identifies all floating outputs or circuitry that does not control any external I/O and then automatically eliminates this circuitry. It is especially useful on soft macros, because unused portions are automatically removed from the circuit. A simple example is illustrated in Figure 15.

The tie-off pusher is another optimization program that can eliminate redundant functions from the input standpoint. The tie-off pusher identifies inputs that have been tied to GND or V_{CC}. If all of the inputs to the module are tied off, then the module is eliminated from the design. This function is especially useful for soft macros from which buffers on unused control signals are automatically removed.

Complex Design Techniques for Improved Density

FPGA density can also be affected by design techniques. More complex design techniques exist to improve FPGA density. These techniques include bit-per-state state machines and pseudo-random number generators. These design techniques are in the "Designing State Machines for FPGAs" in this section of the data book and in the "Designing with Pseudo-Random Number Generator" application note.

Using Synthesis Tools to Aid in Optimization

Synthesis tools are used for a variety of reasons. One purpose is to improve design time by allowing the designer to concentrate on the more global functions and let the synthesis tools generate the detailed gate-level implementation. One problem with this approach is that the designer does not have control over the implementation, so the results can be less than optimal. But synthesis tools can help. The following is a list of possible uses for synthesis tools to ensure maximum device utilization:

- Use synthesis on complex Booleans such as wide decodes. Synthesis can often find combinations of library elements that may not be readily apparent to the designer.
- Use synthesis on soft macros that will be used many times in a design. Synthesis can occasionally find an implementation for a soft macro that is superior to those that a designer might have considered. Savings are then magnified if the soft macro is used repeatedly.
- Use synthesis on logic that can change many times during the design process. State machines are an example of this type of logic. As each change is made, the tendency is to patch up the design to implement the fix—rather than to reminimize the equations and reimplement the state machine. Synthesis can be readily used to avoid this problem because the state equations are changed at a text level. The synthesis tool then does the minimization and the gate-level implementation, potentially saving both time and logic resources.

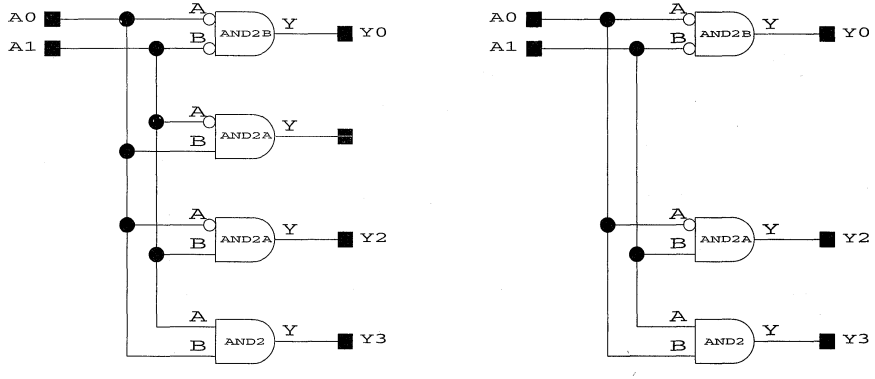


Figure 15 • Simple Example of Unused Gate Removal (Only Y0, Y2, and Y3 are used by system functions.)

Macro Generator—Automated Optimization

As another option, the Actel ACTgen Macro Builder quickly generates custom macros tailored to the user's requirements. ACTgen supports the following categories of functions: adder$, register$, muliplexer, decoder, counter, I/O register, subtractor, and accumulator. The user can specify circuit attributes such as clear, enable, bad, and carry in/out. Many of the optimization guidelines described in this document have been incorporated into the ACTgen algorithms and are dependent on the Actel family specified by the user.

Conclusion

Optimization is the process of packing the mat function into the fewest logic resources possible. The optimization process is important to maximize the density and cost benefits of Actel FPGAs. To achieve optimized results, a basic understanding of the architecture is important. Specific implementation guidelines based on the architecture can then be used to improve a device's density. this process is supported by manual procedures as well as by automated software tools. The manual procedures include hard and soft macro selection from the library, effective use of global clock resources, I/O logic use when available, and specific design techniques. Automated software tools make use of both synthesis and special place and route functions—including the combiner, the gate gobbler, and the tie-off pusher.

Designing with FPGAs Compared with SSI/MSI Devices

Field programmable gate arrays (FPGAs) are powerful devices for implementing complex digital systems. FPGAs are best used with an understanding of the key differences between FPGAs and previous logic technologies. This document focuses on FPGAs compared with small scale integration (SSI) and medium scale integration (MSI) devices. Understanding these differences and using design techniques appropriate for FPGAs results in 50 to 100 percent improvement in speed and density compared to design styles that treat FPGAs and SSI/MSI equally.

Discrete Logic Replacement

An estimation of the size of an existing SSI/MSI design may be determined by using the design's parts list. The data book lists the number of Actel logic modules needed to build each of the soft macros. Multiplying the number of logic modules needed by the quantity of any part used will give the total number of logic modules required for any one function. Consider the parts list in Table 1 for a TTL design. Select an Actel equivalent macro for each item on the parts list, and note the number of modules required. (In this case, there are S-modules and C-modules, since this is an ACT 2 design.) Sum the product of macro quantity and number of modules per macro for all listed macros. In this case, the design requires 250 C-modules and 134 S-Modules, which fit in an A1225 device with a utilization of 63 percent.

Comparing Technologies

SSI/MSI building blocks are created by optimizing the number of pins on popular functions to fit in the small packages available. Logic functions are typically constructed of a few hundred popular building blocks such as counters, multiplexers, shift registers, and comparators. The typical design is optimized to reduce package count, and techniques have evolved to make the most use of a device. For example, simple state machines are constructed from counters and decoders with appropriate pins tied to logic one or zero. This technique minimizes package count, which is the primary cost factor in SSI/MSI designs. The interconnect in these designs is done on the PC board with negligible timing delays. FPGAs, on the other hand, have abundant package pins but are constrained in routing resources, so different techniques are required.

Macro Libraries

Actel provides libraries with the basic system for popular schematic capture tools. The library contains both hard macros and soft macros. Hard macros are similar to SSI components. They form the basic functional building blocks, such as gates and flip-flops. Some Actel hard macros are identical in function to TTL devices, although they have different names. For example, Actel's TA00 is a single two input NAND gate that is a substitution for the quad two input NAND SSI device 74LS00.

Table 1 • Converting Sample Designs from TTL to Actel Macros

					Per Macro		Total	
No.	Part no.	Description	Quantity	Actel Macro	S-module	C-module	S-module	C-module
1	74LS161	4-bit counter	3	TA161	4	10	12	30
2	74F151	8:1 multiplexer	4	TA151	0	5	0	20
3	74LS684	8-bit mag comparator	2	MCMPC8	0	36	0	72
4	74LS377	8-bit register	6	TA377	8	0	48	0
5	74F166	8-bit shift register	8	SREG8A	8	0	64	0
6	74LS74	Dual D flip-flop	9	DFPC	1	0	18	0
7	74F113	Dual JK flip-flop	4	JKF1B	1	0	8	0
8	74F04	Inverter	2	n/a	0	0	0	0
9	74F32	Quad OR gate	2	OR2	0	1	0	8
10	74F08	Quad AND gate	1	AND2	0	1	0	4

Soft macros are more complex functions built from a number of hard macros. Some soft macros examples are counters, decoders, and adders. All soft macros are easily copied, modified, and saved in user libraries. Should you need an SSI/MSI function for which there is no equivalent in the library, it is easy to build it by creating the schematic from a TTL manual, using the Actel library. The Actel FPGA Macro Library Guide contains complete information on all the Actel library components, including a specific TTL section.

If you do not require all the outputs of a soft macro, you do not have to modify the macro. The Actel combiner program (invoked during validate) will automatically eliminate any unused modules before the design is placed and routed. All macro inputs, however, must be connected to another module output, V_{CC}, or ground. For optimal module usage, it is best to copy the macro and the underlying schematic and then edit both of these to reflect the logic reduction.

Three-State Implementation

Many SSI/MSI devices have three-state outputs allowing them to be connected to a common bus. Three-state functions do not work well with FPGAs because they tend to be slow and inefficient. The three-state function can be implemented with a multiplexer, which is particularly effective in an Actel FPGA. Figure 1 shows a three-state SSI/MSI implementation as well as the FPGA multiplexer implementation. An 8-bit bus with four possible drivers can be implemented with only eight logic modules, or less than 3 percent of an Actel A1010 device.

State Machine Techniques

A common state machine design technique with MSI devices uses a loadable counter to implement a state machine. Load inputs are tied to a jump address. (Sometimes logic is used if more than one jump address is needed.) The counter either counts (to advance to the next state) or loads (to jump to a different state). This is efficient in MSI devices, since it requires only a couple of packages to implement a simple state machine. Although this design technique reduces MSI package count, it results in inefficient logic usage for FPGAs. The bit-per-state technique is much more efficient and easier to design when using FPGAs. (See the "Designing State Machines with FPGAs" application note in this section of the data book.)

Another common inefficient FPGA design technique uses a single large state machine instead of multiple communicating small state machines. In MSI devices, sometimes a single microcoded state machine controls a complex data path. This works well since large registered PROMs are available to implement a design in a small number of packages. However, these designs are complicated, since each state could have several activities occurring simultaneously, and the interactions between each activity need to be checked in every state. In FPGAs, multiple communicating state machines are easier to design, since most of the communication is local, and only a few activities need to be communicated between different state machines. The distributed machines tend to have much simpler logic requirements that also fit better with the FPGAs register-rich, small-logic building-block characteristics. This approach is also better for FPGA routing, because the routing resource requirements are more distributed.

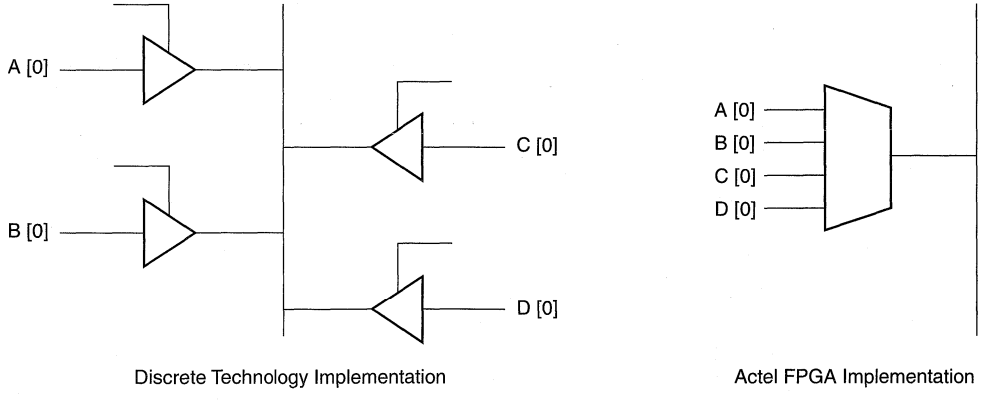


Figure 1 • Least Significant Bit of a Bus with Four Possible Drivers

Data Path Oriented Techniques

If counters that are loaded only occasionally are required, prescaling techniques can be used to improve operating frequency. However, these techniques also result in slower load capability. Applications that need to generate long address sequences (for example, memory access) can use this load latency counter very effectively and can operate at a higher speed compared with nonlatency versions.

Nonlatency counters designed using look-ahead techniques have better performance than do MSI equivalents. These methods do not impose any additional constraints on the application, as does the load latency counter does, but they take advantage of the register-rich nature of FPGAs in implementing counter functions. For example, in a 16-bit downcounter, each register should roll over after the counter reaches all zeros. Instead of detecting the all-zero case by placing combinatorial logic after the counter registers, logic can be placed in front of a register to detect the case in which the counter contains a 1 and is counting down. The register will then be active on the same cycle in which the counter contains all zeros, saving the combinatorial delay associated with the all-zero detection.

Random Logic Oriented Techniques

Many of the SSI-oriented techniques that designers use for random logic translate directly into FPGA devices because of the similarity of the basic building blocks. You must keep in mind that routing resources are limited inside FPGAs, whereas routing resources in SSI designs on PC boards are virtually inexhaustible. Sections of logic that use too many different clock sources and high fan-in may overly constrain routing. For example, it is usually more efficient to use a synchronous clock source with synchronous enables instead of a large number of individual clock signals to load individually selected data bits into registers because synchronous enable signals have more routing flexibility than do clock signals.

Since FPGAs are most efficient at implementing logic at the input of registers, a good rule of thumb in implementing random logic is to use logic at the input of registers instead of at the outputs wherever possible. For example, multiplexing a signal prior to a register is more efficient than multiplexing the signal after the register.

Conclusion

SSI/MSI designers can easily use FPGA technology to reap the benefits of lower cost, smaller board size, and lower power. However, using different techniques that are better suited for FPGAs will allow between 50 and 100 percent improvement in performance and capacity.

Designing with FPGAs Compared with PLD Devices

Field programmable gate arrays (FPGAs) are powerful devices for implementing complex digital systems. FPGAs are best used with an understanding of the key differences between FPGAs and previous logic technologies. This document focuses on FPGAs compared with programmable logic devices (PLDs). Understanding these differences and using design techniques appropriate for FPGAs results in 50 to 100 percent improvement in speed and density compared with design styles that treat FPGAs and PLDs equally.

Comparing Technologies

PLDs are array-oriented devices that typically have an AND-OR structure with wide-input AND gates feeding a narrower OR gate. A register is typically available at the output of each OR. This architecture is termed *logic rich* because there are typically many more logic gates than registers available. The ratio of gates to registers can be as high as 5 to 1. Because of the large delay through the wider logic module, PLDs pay a significant speed penalty when multiple levels of logic are required. Speeds tend to be more predictable in PLDs because of the larger *speed quanta*.

FPGAs on the other hand, are register rich, with a logic-to-register ratio closer to 2 to 1. (This ratio is equivalent to the traditional gate array usage and tends to be related to high density designs' need for more registers than are needed by the traditional "glue logic" oriented low-density applications.) FPGA logic structures are optimized for functions narrower than those of PLDs. FPGAs have a smaller speed quanta than do PLDs, so logic functions can be incremented in complexity while incrementing the delay only a little each time. In addition, signals that need to be fast can be sourced near the bottom of the logic tree, minimizing the number of logic levels required, and slow signals can be sourced at the top of the logic tree, where more logic levels are required.

Estimating PLD Logic Replacement

Estimating the number of logic modules needed to replace an existing PLD is straightforward. Figures 1 shows an example of a typical PLD file converted to Actel logic modules. Typical PLD functions are address decoding and state machine control. Figure 1 is a wide-register AND function decoding 25 inputs and using only 6 logic modules. Since the first AND5B gate is combined with the DFC1B flip-flop in a sequential module, the 25-bit decode is done in only one logic level. A second level could be added to provide 32-bit decode with 8 modules or 64-bit decode with 16 modules.

Figure 2 is a familiar three-product term AND/OR array using only four logic modules to implement a typical state machine equation. Figure 3 shows the product and sum terms expanded further. In the PLD equation examples, note how easily the number of either product or sum terms can be expanded. Small incremental changes in both delay and size are added each time the array is enlarged. This is in sharp contrast to the large step-function increase in delay and size when a second PLD array must be used to implement a particular equation.

PLD equation conversion shows the flexibility of the Actel logic module, implementing a variety of different combinatorial and sequential functions in few modules without wasting dedicated resources. With ACT 2, ACT 3, and 1200XL devices, the cost of using a sequential function is equal to the cost of using a combinatorial function. In contrast, PLDs are rich in AND/OR gating but lacking in registers. Encoding states in a PLD requires using the smallest number of flip-flops possible, but with Actel you are free to use any combination of flip-flops and logic.

State Machine Techniques

The traditional PLD design techniques for implementing state machines are geared toward the logic-rich and register-lean architecture of the standard PLD. A small number of state registers are used (usually the theoretical minimum), since registers are scarce. This approach requires a larger amount of combinatorial logic to decode the state, but PLDs usually are able to provide enough combinatorial logic to do this effectively. Using this technique for FPGAs would not be an efficient use of FPGA strengths—numerous registers and fast narrow logic gates. A bit-per-state approach, where by each state uses a separate register instead of encoding states in multiple registers, results in faster and more efficient state machines in FPGAs. (Refer to the "Designing State Machines for FPGAs" application note in this section of the data book.) In many cases, speed improves by 50 to 100 percent compared with the PLD-oriented methodology of an encoded state machine.

In PLD-oriented designs, logic is typically used to develop outputs from state machines. Usually this requires an additional level of logic after the state register and adds

delay. In FPGAs, this level of logic can be eliminated in many cases by combining the logic in front of the state bits in which an output is active. For example, if the chip enable (CE) output from a state machine needs to be active in states 3 and 5, the logic feeding state bits 3 and 5 can be ORed together and registered to create the CE output without incurring a logic delay after the register. Since the logic in front of state bits is simple, usually no additional delay or logic resources are required in front of the new register.

Another popular state machine design technique for PLDs uses counters to generate a sequence of wait states. For example, a state machine may need to wait for 16 cycles until a data transfer can begin. A 4-bit counter can be used to generated the required state sequence. This is fairly efficient in PLD architectures because of the logic-rich and register-lean characteristics of the count function. It is not as good a fit for FPGAs, however. In FPGAs, registers are rich, and a shift register is more efficient and faster than a counter. A normal shift register will require one register per wait state. If very large delays are required, a feedback shift register can be used to implement only one state fewer than a counter, but it requires much less logic and is significantly faster.

Conclusion

PLD designers can use FPGA technology to reap the benefits of lower cost, smaller board size, and lower power. However, using new techniques that are better suited for FPGAs will allow between 50 and 100 percent improvement in performance and capacity.

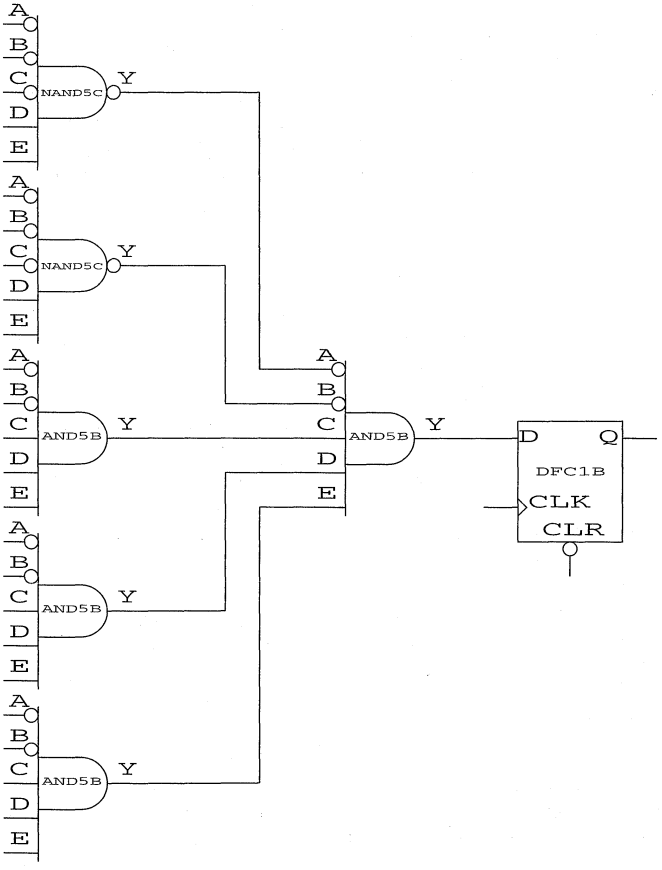


Figure 1 • PLD Implementation with Wide Fan-In

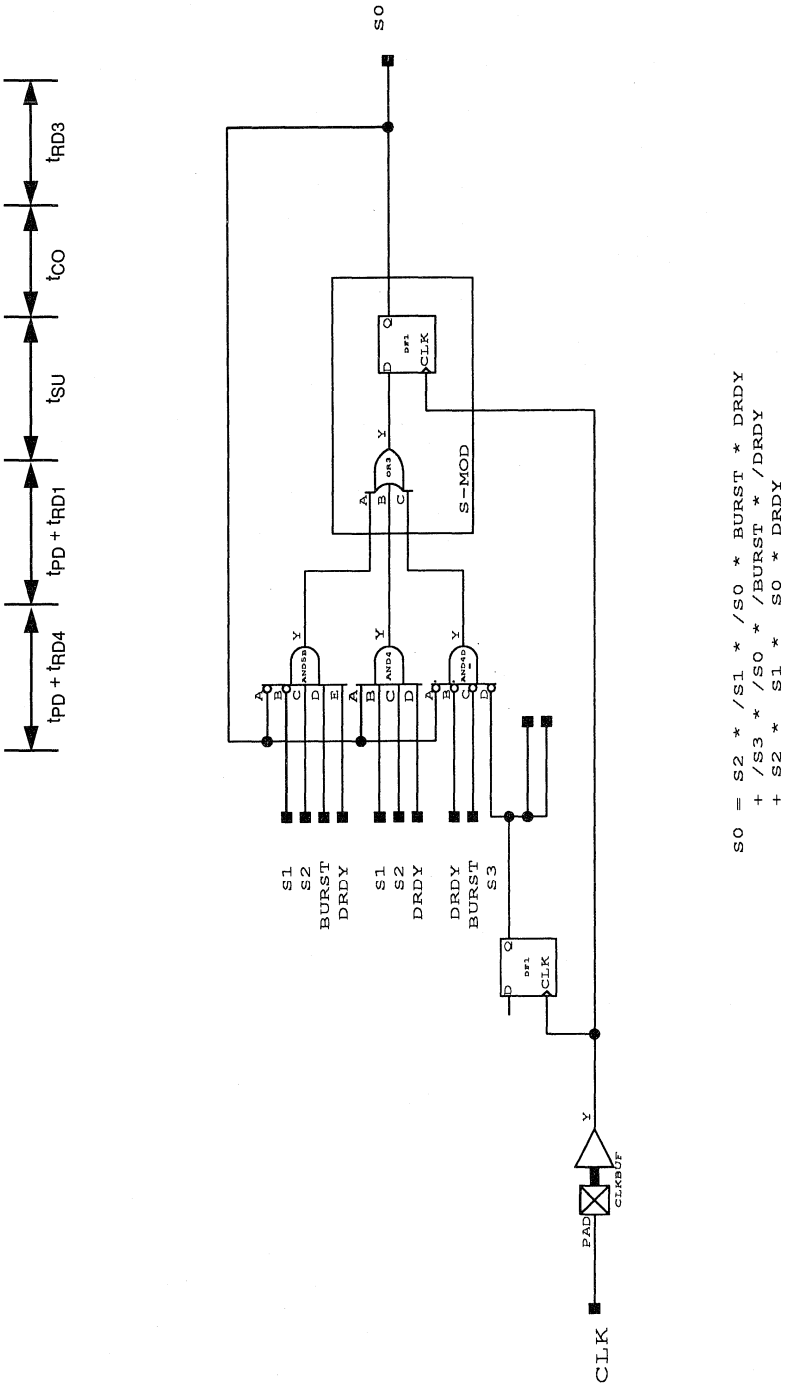


Figure 2 • Implementing State Machine

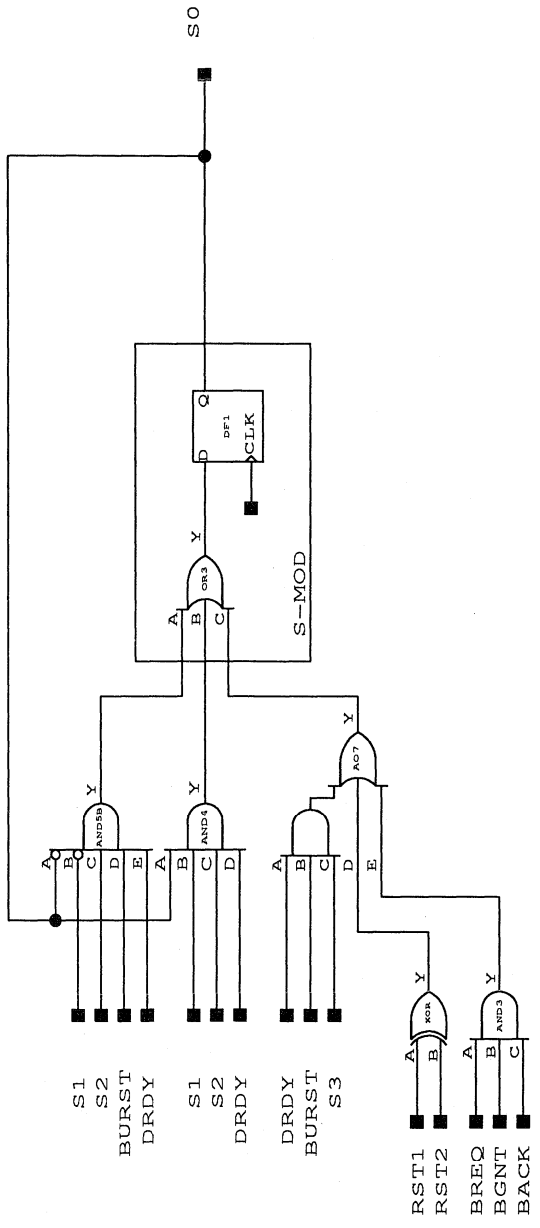


Figure 3 • Expanded State Machine

Designing Adders and Accumulators with FPGAs

Introduction

Many designers implement adders by using carry-propagation techniques. The multiplexer-based combinatorial module (C-module) and the Actel ACT 3 and 1200XL devices allow you to use the more efficient carry-select design. This method partitions the add functions into blocks that perform two additions simultaneously on a number of bits of the two operands.

The ACTgen Macro Builder software tool, bundled with the Designer Series, is based on all the rules presented in this application note. ACTgen allows the designer to customize adders and accumulators quickly up to 32 bits and with a number of options (carry in, carry out, and so on). As such, this document should be used only for reference, since users can create adders and accumulators in minutes with ACTgen.

The two additions are the same except that one assumes a carry-in and one assumes no carry-in. The two sums are input to 2 to 1 multiplexers, one for each bit pair. The carry line from the low bits to the high bits is used to select the appropriate sum for each block.

The ACT 2, ACT 3, and 1200XL are good architectures for implementing adders of various sizes using the carry-select technique. A sample design for a 16-bit adder, as shown in Figure 1, will be used to illustrate adder design.

Balancing Sum and Carry Levels

To obtain optimal performance from a carry-select adder, design it so that the number of levels of logic required for the carry chain equals as closely as possible the number for the largest sum block. When they have the same number of levels, the sum bits arrive simultaneously at the data pins and the select pins of the output multiplexing stage.

To balance the levels of logic modules for the sum blocks with the carry, partition the sum blocks by considering the logic levels required for the sums and the levels for the carry between sums. The size of the partitions varies with width of the data. The Actel libraries contain some powerful hard macros used to shorten the levels of logic required for generating sums and carries. The description of the sample design will illustrate the use of the macros.

Sample Design

For the 16-bit adder, the optimal organization is to perform two 2-bit additions on the four least significant bits with the remaining higher order bits broken into four sections of three bits each.

In the top-level schematic, the addition logic of the two least significant bits is visible. The other additions are performed in lower levels of the design hierarchy described in the next section.

Carry Logic

The ACT 2, ACT 3, and 1200XL libraries include two 2-level carry hard macros. One macro generates a carry for the two-bit pairs, assuming the carry-in is true; the other assumes it is false. The latter macro can be seen at the bottom of Figure 1 making the carry for the two least significant bits.

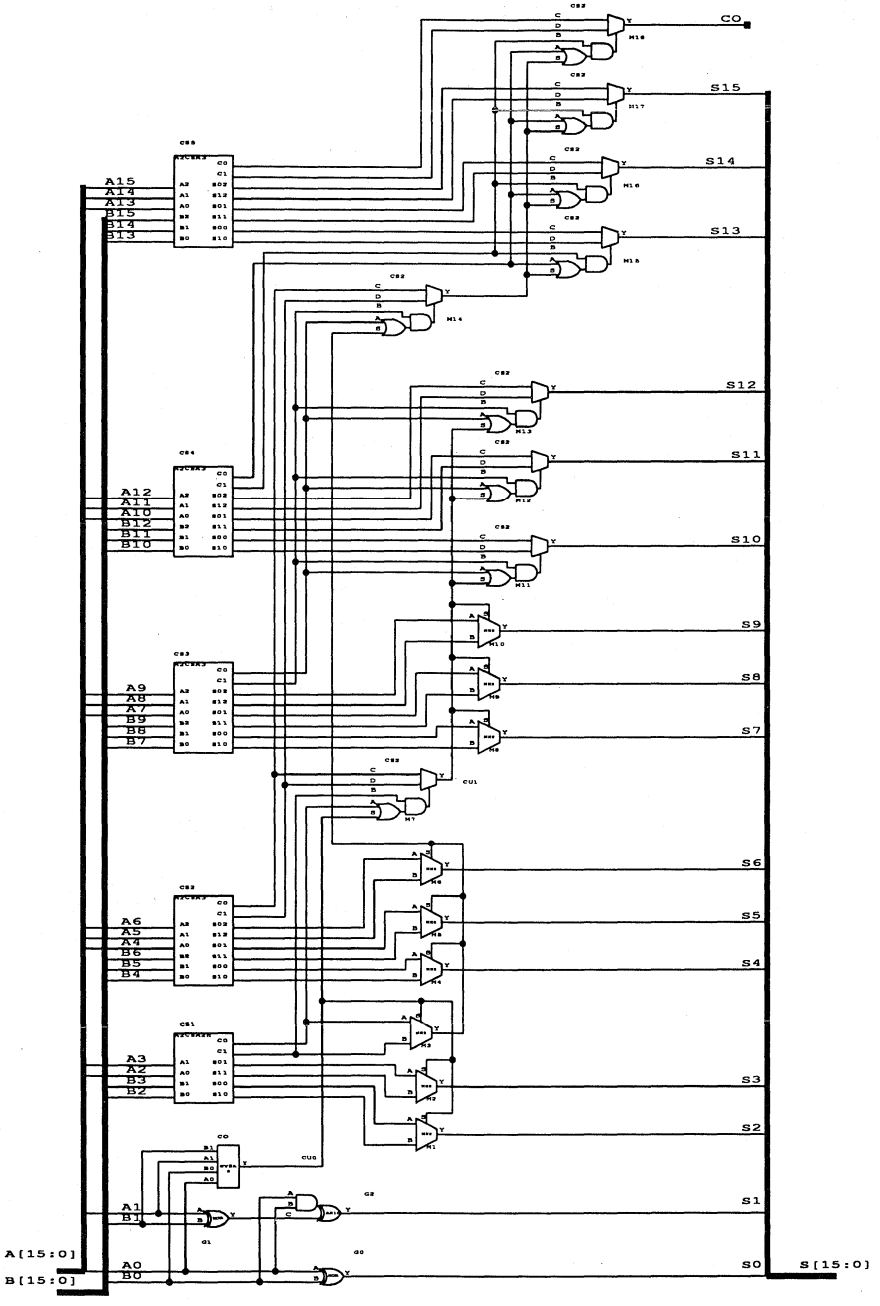


Figure 1 • 16-Bit Adder Schematic

The carry macro output drives the select line for the 2 to 1 multiplexers for sum bits two and three. It also drives the select line on the cascade multiplexer. The cascade multiplexer is a special hard macro that can propagate two levels of carry. The macro is depicted in Figure 2 and has five inputs. The top multiplexer inputs select the most significant sum or carry. The three lower inputs drive logic that implements a simplified form of a 2 to 1 multiplexer.

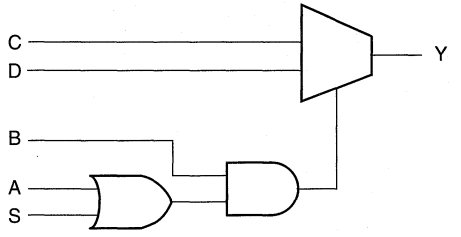


Figure 2 • *Cascade Multiplexer Macro*

A fully implemented 2 to 1 cascade multiplexer does not map into the Actel module efficiently, but the full functionality is not required in a carry-select adder. There is a simplified version of the cascade multiplexer that maps into a C-module or combines with a flip-flop in a sequential module (S-module).

The simple version has logic driving the select for the upper level multiplexer, consisting of only a two-input OR driving one input of a two-input AND. The two OR gate inputs are driven by the carry output from the next lower sum block, assuming no carry-in, and by the carry-in from the rest of the lower bits of the adder. The remaining AND input is the carry from the sum block, which assumes a carry-in.

The logic is correct for a carry-select adder because, if the assume-no-carry-in input is true (meaning that a carry was generated within that sum block), then the assume-carry-in is always true (since it equals the false plus one). This completes the AND function.

If the carry from the lower bits is true (meaning a carry is propagated to the sum block), then we complete the AND if the assume-carry is true.

3-Bit Carry Select Adder

The schematic for the 3-bit adder block appears in Figure 3. The adder requires thirteen logic modules to generate the three sum and carry pairs. All the output paths are two levels of logic or fewer.

The two carries for the 3-bits come from two-level carry hard macros driving a 3-bit majority macro.

All the sums are generated from exclusive OR or NOR gates. The Actel library contains several two-module adder macros, each with both a sum and a carry output. These macros can be used as part of a sum, depending on how well they fit into the overall adder soft macro structure.

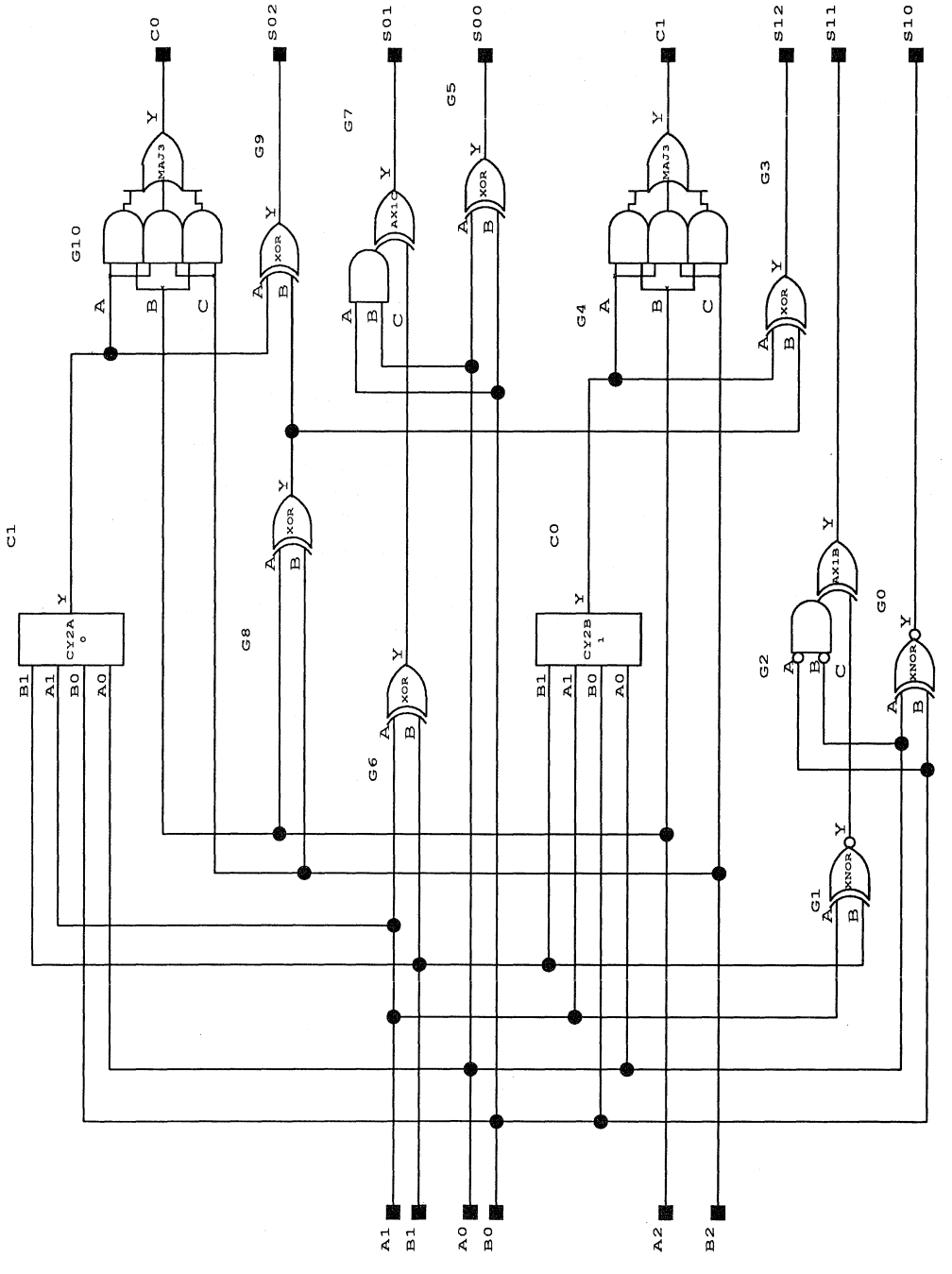


Figure 3 • Three-Bit Adder Block

Making an Adder into an Accumulator

All the sum and carry outputs of the adder macro can be combined into a single S-module. This feature means that if the data inputs of a 16-bit register are schematically connected to an adder's outputs, the Actel Designer Series software will automatically put the adder output macros (2 to 1 multiplexers or cascade multiplexers) into their respective flip-flops in the register.

The registered-output adder will suffer no degradation in performance from combining, because the delay through the combinatorial part of the S-module is less than that of an uncombined macro. Tying the register output back into the inputs will make the circuit into an accumulator. A sample design for an accumulator made from an adder and a register is seen in Figure 4.

Sample Design Results

The sample design uses 82 modules. The slowest path in a function is usually the one with the most levels of logic. In this case, it is the carry chain, which has four levels of logic. As mentioned previously, all other paths have fanouts of three or less.

The modules in the chain have fanouts of three, four, seven, and four. Criticality may be used to optimize the path performance. Criticality works best when fanout is low. When the fanout of a speed-sensitive net exceeds seven, performance can usually be improved most by adding redundant logic. For fanouts of less than seven, adding a redundant module may bring no improvement. Using redundant logic for fanouts of seven should be considered on an individual basis. Adding a redundant module to the carry path would change its fanouts to three, five, three, and four. The expense of one module may be justified by the performance improvement from lowering the fanout.

It is also possible to improve performance by pipelining an adder. Since all of the combinatorial functions used in the adder can be combined (if the function's output drives a flip-flop, the Designer Series software will put both into a single S-module), designers can pipe the adder at the points that provide the best performance at no cost in additional modules.

Other Adder Macros

The carry select architecture can be extended to adders of any size. Adders of 8 to 15 bits can be designed using the technique in three levels of logic. Adders from 16 to 24 bits can be done in four levels.

When adapting the adder design to other operand sizes, remember to repartition the sum block sizes to match the logic levels of sums and carries.

Very Fast Adders

The VAD series of very fast adders in the Actel soft macro library uses several techniques to create optimally designed macros for the Actel FPGA architecture. The VADC32C very fast 32-bit adder macro uses variations of the VADC16C very fast 16-bit adder macro as a building block. This configuration yields extremely high performance for 32-bit system designs. What follows illustrates these adder design techniques as applied to the VADC32C macro.

The carry propagation technique (ripple carry addition) for use in serial addition networks is defined as

$$S_i = A_i \oplus B_i \oplus C_i$$

where S_i is the sum bit, A_i and B_i are the ith bits of each operand, and C_i is the carry to the ith stage; the carry to the next stage is defined as

$$C_{i+1} = A_iB_i + C_i(A_i + B_i)$$

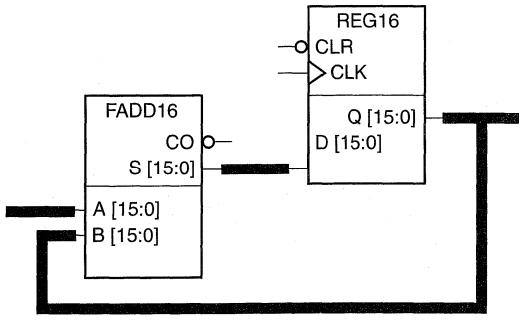


Figure 4 • 16-Bit Accumulator

Therefore, adding two n-bit operands takes at most n-1 carry delays and one sum delay. This delay becomes large for adders with n > 4 and significantly reduces operating speeds.

A much faster algorithm for adding n-bit operands uses the combinational sum technique. This is a multiplexer-based design that partitions the add functions into blocks that perform two additions simultaneously. The two additions are the same, except that one assumes a carry in while the other assumes no carry in. The two sums are input to 2:1 multiplexers, and the carry in of this addition controls the output of the multiplexer. The adders provide high performance, requiring only four logic module delays from input to output.

Design

The VADC16C macro uses the carry look ahead technique to generate the carry out of the 16-bit addition while performing the summation of the operand twice. One addition assumes a carry in, and the other assumes no carry in. Both additions are performed simultaneously. Both sums are input to a multiplexer, and the true carry in controls the output of the multiplexer—the sum of the operands. This initial addition is performed with a total of three logic module delays.

Cascading two 16-bit very fast adders is the simplest way to design a 32-bit adder. But this technique involves three additional logic delays, which is unacceptable for a very fast adder. A novel implementation of the VADC16C soft macro extends the width of the inputs to 32 bits, with only one additional logic delay.

The design is split into two blocks, as shown in Figure 5. The first block is macro VADC16CN, which performs addition on the first 16 bits and generates a carry out. Actually, two carry outs are generated. One assumes a carry in, and the other assumes no carry in; the true carry in controls the multiplexer so that the true carry out is at the output of the carry out multiplexer. Similarly, two sums are generated for the 16-bit addition. One assumes a carry in, and the other assumes no carry in. The control input of the multiplexer is connected to the true carry in, thereby steering the correct sum to the output of the sum multiplexer. This macro is the same as soft macro VADC16C, except that it has several carry outs to allow for fanouts. These additional logic modules reduce possible loading effects to maintain high operating speeds.

The second block is macro VADC16X, which is further split into five lower levels of logic blocks: VADC16XC, VADC16XL, VADC16XM, VADC16XU, and VADC16MX. VADC16XC

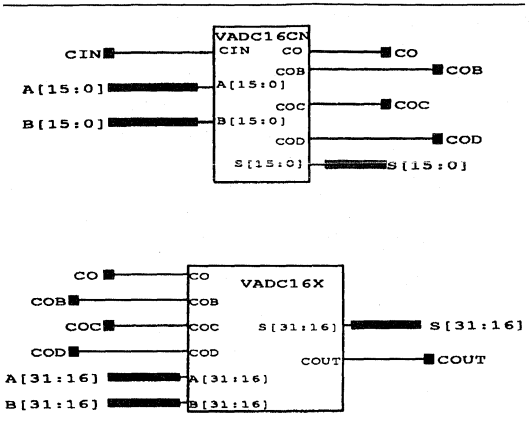


Figure 5 • Block Diagram of Soft Macro VADC32C (Very Fast 32-bit Adder with Carry In)

generates the carry look ahead for the upper 16 bits of the 32-bit adder. Two carry outs are generated. One assumes that there is a carry in, and the other assumes no carry in from the addition of the first 16 bits. This result is input to a 2:1 multiplexer, with the actual carry from the lower 16 bits controlling the output of this multiplexer—the carry out of the 32-bit addition. VADC16XL performs two additions for bits 16 to 24. One assumes a carry in, and the other assumes no carry in. Similarly, VADC16XM performs two additions for bits 25 to 28, and VADC16XU performs two additions for bits 29 to 31. The sums generated by these blocks are input to the 2:1 multiplexer block VADC16MX. The carry out of the sixteenth bit is applied to the control of this multiplexer, yielding the true sum at the output of the multiplexer block. This technique has a total of four logic delays—one-eighth the traditional ripple carry addition technique, which results in 32 logic module delays.

This adder design results in a 32-bit addition requiring a total of only four logic module delays and approximately 250 logic modules. It is useful for applications that require very fast additions in Actel FPGA devices.

Conclusion

To obtain the highest performance adders and accumulators in FPGAs, carry propagation techniques must be used. Using these approaches for 16- and 32-bit adders is not trivial. It is highly recommended that the designer use the ACTgen Macro Builder to create custom adders and accumlators quickly to meet their needs.

Designing Counters with FPGAs

Perhaps the most common digital logic function is the synchronous binary counter. Regardless of the technology employed to implement counters, they are found in every type of application. Designers familiar with counter design using discrete logic can predict the performance of the counter by reading a datasheet. When using field programmable gate arrays (FPGAs), there is more to consider. The ACTgen Macro Builder software tool, bundled with the Designer Series, is based on all the rules presented in this application note. ACTgen allows the designer to customize counters of up to 32 bits quickly with many options (clear, enable, load, and so on). As such, this document should be used only for reference, since users can build counters customized to their needs in minutes with ACTgen. The following describes some of the considerations for designing or modifying a counter in the ACT 2, ACT 3, and 1200XL Soft Macro libraries.

Performance Factors

The advantages of using FPGAs instead of discrete devices are well known. To maximize the benefits of high integration and low power, it is important for the designer to understand how best to implement the counter. The performance of the counter is variable, so it is important to use the optimal design. Four criteria influence performance of logic in FPGA designs, in descending order of importance:

- *Levels of logic.* The fewer the number of combinatorial logic levels between flip-flops, the faster the counter frequency.
- *Fanout.* Propagation delays in FPGAs are sensitive to fanout. Limiting fanout on individual nets improves performance.
- *Fan-in.* Fan-in measures the number of nets connected to a logic module's inputs. Library functions with heavy fan-in efficiently use the logic of the module; they aren't a problem when used sparingly. Too many high fan-in macros, though, can congest routing and reduce performance.
- *Number of modules.* Fewer logic modules allow them to be placed closer to one another. Shorter distances between modules speed the connection paths.

While each of these considerations is important in itself, it must be remembered that they are interrelated; an improvement in one may cause a degradation in another; for example, limiting the number of logic levels tends to increase fan-in. A balance must be found among these considerations so that none becomes a drag on performance.

Before proceeding to a detailed description of a sample design, it would be useful to review some fundamentals of the architectures. In the design of a counter or any soft macro, device architecture should always be considered to obtain the best results.

Two Types of Modules

The ACT 2, ACT 3 , and 1200XL architectures features two types of modules. Combinatorial modules (C-modules) are used to implement any combinatorial functions. Sequential modules (S-modules) can be used for sequential functions (for example, flip-flops) or combinatorial functions or both. When the S-module is used to implement both a sequential and a combinatorial function (for example, a gate followed by a flip-flop), it is being used in the most efficient way.

The S-modules for ACT 2/1200XL and ACT 3 are shown in Figure 1 and Figure 2, respectively.

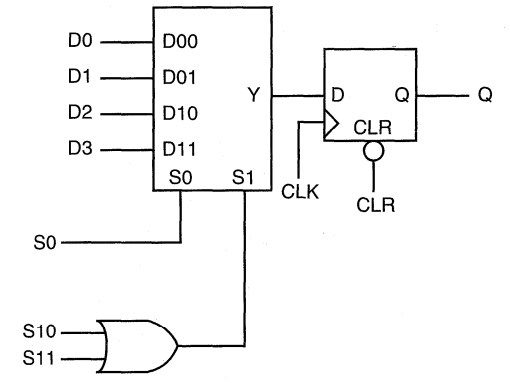


Figure 1 • ACT 2 and 1200XL Family Sequential Logic Module

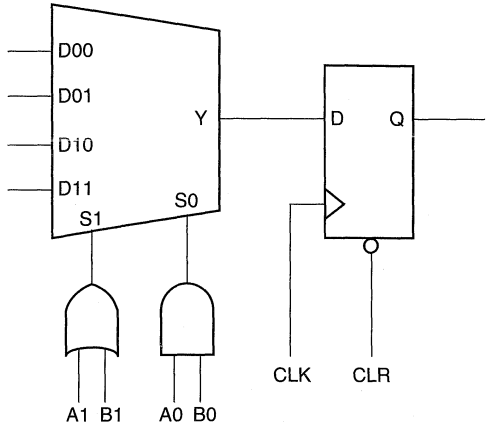


Figure 2 • ACT 3 Sequential Logic Module

The module types exist in roughly equal numbers on these devices. The place and route software will automatically select the appropriate module for each library component in the schematic. It is up to the designer to understand how to select components from the library to take best advantage of the logic in the modules.

As the sample design will show, you can construct a 10-bit counter with only one level of combinatorial logic between flip-flops, and a 16-bit counter with only two combinatorial levels. If some counter outputs may be active low or if additional modules are used for redundant logic (for example, some bits have both an active-high and an active-low output), then larger counters may be designed without additional logic levels by using five-input gates. Such decisions should consider the implications for module count and fanout as detailed later.

Sample Design

The sample design for a 16-bit synchronous loadable binary counter with a count enable will illustrate some of the considerations for using counters in FPGA designs. The functional description for the counter appears in Table 1.

Table 1 • Counter Function

RST	LD	CE	CLK	Q
0	X	X	X	0
1	1	X	↑	D
1	0	1	X	Q
1	0	0	↑	Q + 1

Using the Modules

The counter design makes extensive use (bits 0 through 5) of a 4 to 1 multiplexer driving a flip-flop, as depicted in Figure 3. Both the multiplexer and the flip-flop will be combined into a single S-module by the Actel Designer Series software. The select lines on the multiplexer are operated by the load control (S1) and by the counter enable and carry from the lower order bits (S0). The multiplexer data inputs are used for data to be loaded, held, or incremented.

C-modules are used as AND-EXORs and for ANDs to qualify the count function. The AND function is also used to bring the count enable (CE) to the multiplexer by means of the select line in bits Q3 and above. Using both the select inputs as well as the data inputs to AND lower order bits allows for more paralleling, which results in fewer levels of logic.

For the bits Q6 through Q15, an S-module macro with both a 4 to 1 multiplexer and an OR gate driving one select line is used to allow for more parallel propagation of the lower bits. Figure 4 shows the implementation of the most significant bit of the counter.

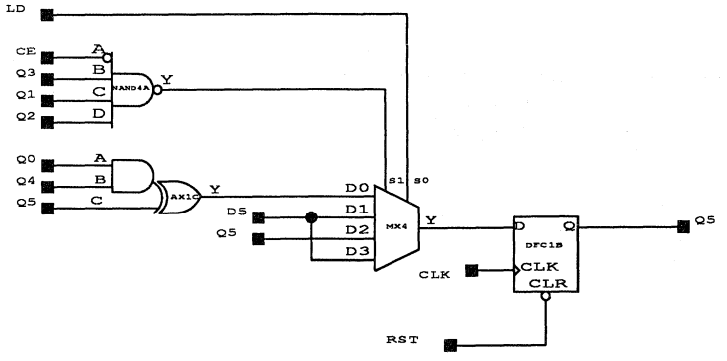


Figure 3 • Counter Bit Q5

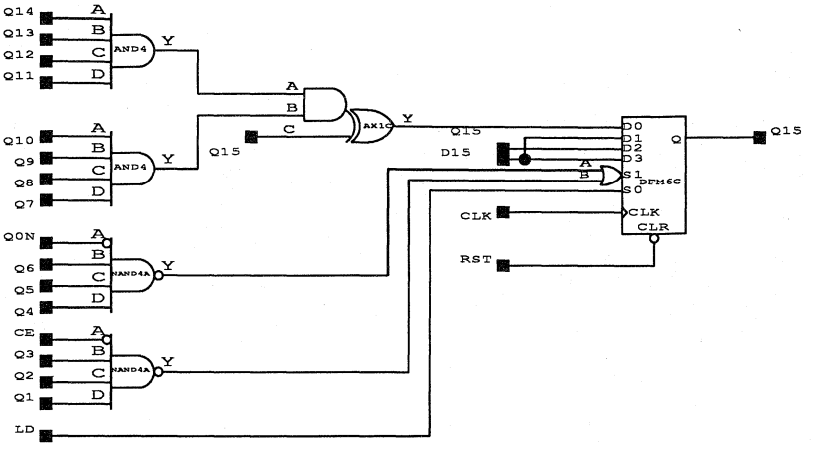


Figure 4 • Counter Bit Q15

The S-module OR gate is used as a two-input NAND with active-low inputs that are, in turn, driven by NAND gates to propagate the lower bits and the count enable. The active-low output of the built-in two-input gate (OR used as a NAND) is adjusted for by shifting the position of the multiplexer data inputs.

Most four-input gates are implemented with a single C-module, but a four-input NAND with no bubbled inputs requires two modules. The limitation is avoided by using a NAND with a bubbled input. The count enable is active low, so it may be used to drive a bubbled input on a gate. Active-low counter bits are used to drive other bubbled gate inputs.

Levels of Logic

As can be seen in the complete counter schematic (Figures 5 and 6), two bits (Q0 and Q6) use inverters. The Q0 inversion toggles the flip-flop. A toggle flip-flop could have been used instead of a D flip-flop, but it could not have been combined with the multiplexer into a single S-module. Moreover, the inverter output is available as a resource to share the fanout load with the flip-flop and to allow the use of bubbled inputs on gates whenever it is desirable.

It could be argued that the use of the inverted output to drive gates causes the lower level bits to use two levels of combinatorial logic when it is not necessary. For a design of 10 bits or fewer, the point would be valid, because no path requires more than one combinatorial level. In the example design, however, two levels are already required by the upper bits and the improvement in fanout from the use of the inverter output at no additional cost in module count makes the practice worthwhile.

Limiting Fanout with Redundant Logic

The paths in the design most likely to limit performance are those with the largest number of logic levels and the highest fanout. In general, when fanout exceeds 9 on a critical path, redundant logic is often called for. For lower fanouts, the decision to use redundant logic might be a problem and must be balanced by considering both the cost in additional logic modules and the fanout to the outputs driving the redundant logic.

In the sample design, two redundant modules are added to illustrate the concept. One is the XNOR gate, whose output is the inversion of Q1. The other is the four-input NAND gate that propagates flip-flop outputs. No fanout in the design exceeds seven, and the worst-case path is the redundant gate whose inputs are driven pins with fanouts of seven, six, six, and five, and whose output fanout is four. A total of 48 modules are used in the design.

Load Latency Fast Counter

An important realization in designing high-performance counters is that the least significant bits (LSBs) of the counter change the most frequently, and higher order bits change much less often. This fact can be used to optimize counter performance by ensuring that the least significant bits enter the logic trees at the lowest (fastest) level. Higher order bits can enter further up the tree, since they have a longer time to propagate through the logic. Consider the design of a 6-bit counter using this technique. The counter will be loadable with asynchronous clear. It will also be a downcounter, suitable for timing or address generation in a typical digital design. An upcounter is an easy modification to the design, but conceptually they are similar.

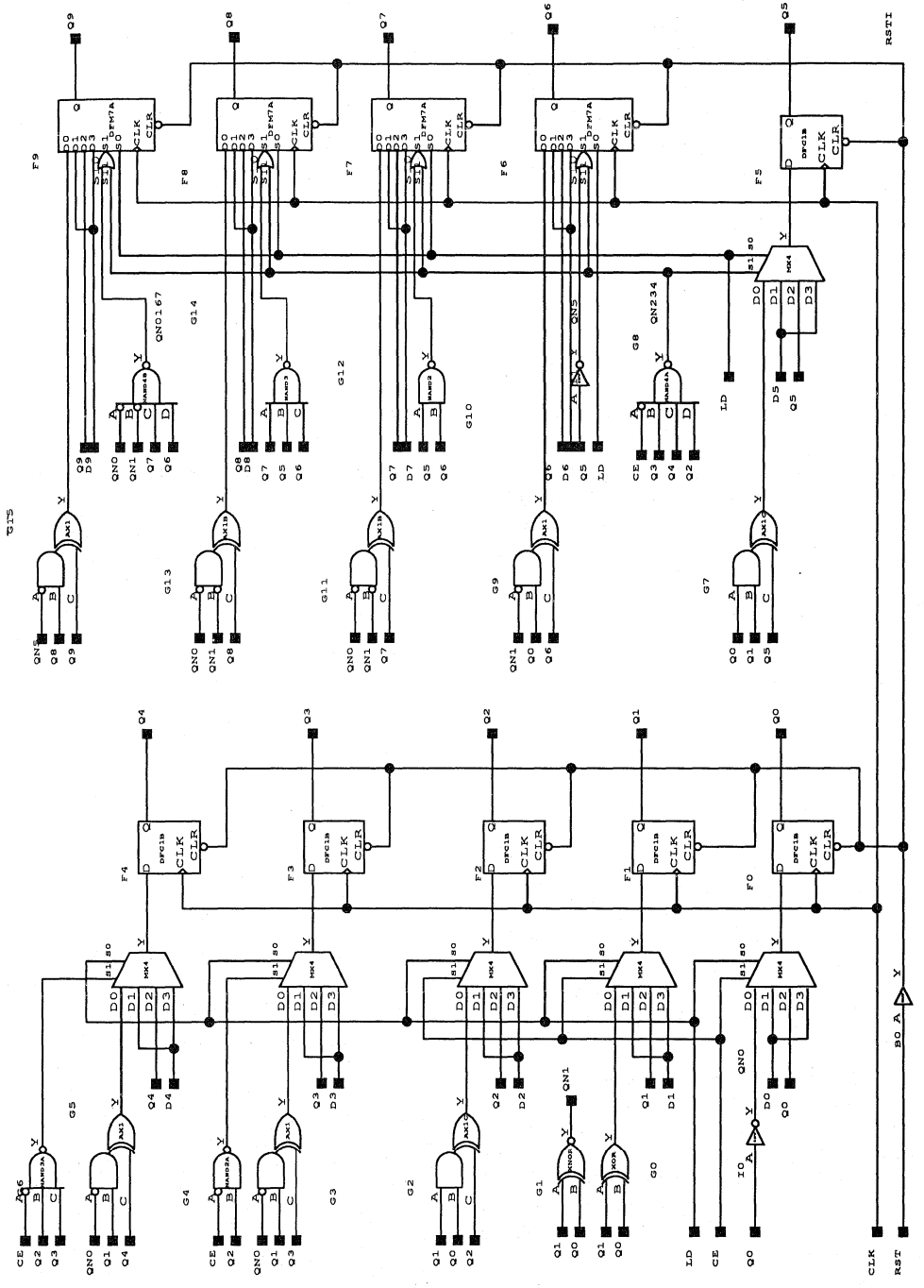


Figure 5 • 16-Bit Counter Schematic

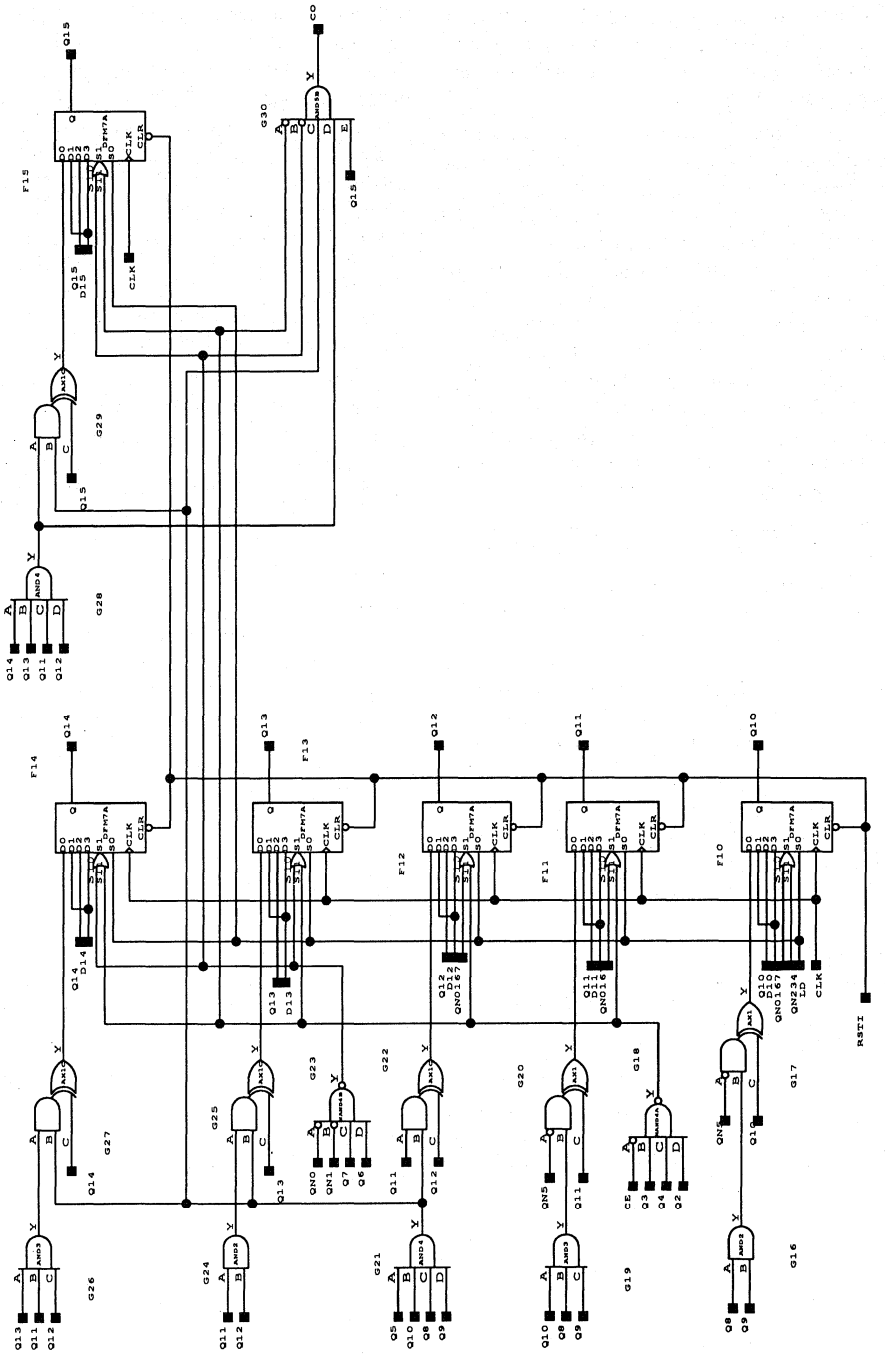


Figure 6 • 16-Bit Counter Schematic (Continued)

Least Significant Bit

The counter must have a least significant bit that can toggle at the highest possible rate. These logic modules can be used to construct a least significant bit with clear, load, and count enable, as shown in Figure 7

Data can be loaded into the register, when the load enable (LD) signal is high, by selecting the D1 or D3 input on the multiplexer. The count enable signal (CNT) is used to toggle Q0 when counting is enabled. If LD is disabled, the multiplexer input is either D0 or D2, depending on the state of Q0. If Q0 is a zero and CNT is a one, the register will be loaded with a zero (NOT CNT) from the D0 input. Thus Q0 toggles if CNT is a one. If CNT is a zero, Q0 will hold, not toggle, since the D0/D2 inputs will not be zero and one, respectively. Thus, the least significant bit needs only a single level of logic to operate. Since the next most significant bit will toggle only when S0 is zero (implementing a down-counter), there is one extra clock cycle to develop the signals for Q1's next state.

Figure 8 shows the implementation of Q1 using the DFM7A macro. It is similar to the LSB except that Q1 can be inverted to develop the toggle signal. Notice that this signal is selected only when Q0 and /CNT are low. This keeps the slower /Q1 signal from participating in the logic function until it has settled, ensuring fully synchronous operation. This technique will be the cornerstone of the most significant bits (MSBs), where slower signals are gated until there is sufficient time to settle and until they are needed to compute the toggle of the associated counter bit.

Figure 9 shows the entire 2-bit prescaler (CNR2P) for fast counters. Added to Q0 and Q1 are three registers to source CNT, /CNT, and LD. These will be used to reduce fanout in the faster counter implementation.

Most Significant Bit

A 4-bit macro for the MSBs is shown in Figure 10. It uses the LSB from the CNT2P macro connected to CT0 (on S0A) and CNT (on S0B) to enable the multiplexer inputs used for toggling, similar to the CNT2P macro. In addition, the next LSB is connected to CT1 (on B of AXB1), and the counter bits in the macro (Q0–Q2) are used to gate the input to the XOR (see Q3 for example), which determines whether the register bit holds or inverts. A ripple carry input (RCI) is also used to allow results from previous stages to participate. This technique allows MSBs at least four clock cycles (since the LSBs transition from 00 to 00 in four clock cycles worst case) to develop the input to the associated bits in the counter. A 6-bit counter is shown in Figure 11.

The limiting frequency for the 6-bit counter is based on the longest clock-to-clock delay in the design. There are four possibly limiting delays in the design: the minimum clock input period of the device, a single-level delay from Q0 to Q0–5, a two-level delay from Q1 to Q1–5, and a three-level delay from Q2 to Q2–5. Estimates for each delay are given below in Table 2 for an A1425A-3, over worst-case commercial conditions.

An 18-bit counter implemented with these macros is given in Figure 12 The limiting frequency for the counter is determined similarly to that in the 6-bit example, and the delays are given below for an A1425A-3. Figure 13 shows a modified 18-bit counter with reduced fanout.

Conclusion

To obtain the highest performance counters in FPGAs, specific design techniques need to be used. Actel has imbedded this information in the ACTgen Macro Builder to allow users to customize counters quickly for specific design needs.

Table 2 • Critical Delays for 6-Bit Counter Implemented in A1425A-3

Delay	Datasheet Parameter(s)	Value	Requirement
Minimum Clock Input Period	t_A =	4.0 ns	Must be ≤1 Clock Cycle
Q0 Delay	$t_{PD1} + t_{RD8} + t_{SUD}$ = 2.0 + 2.8 + 0.5 =	5.3 ns	Must be ≤1 Clock Cycle
Q1 Delay	$2[t_{PD1} + t_{RD8}] + t_{SUD}$ = 2[2.0 + 2.8] + 0.5 =	10.1 ns	Must be ≤2 Clock Cycles
Q2–5 Delay	$4[t_{PD1} + t_{RD8}] + t_{SUD}$ = 4[2.0 + 2.8] + 0.5 =	19.7 ns	Must be ≤4 Clock Cycles

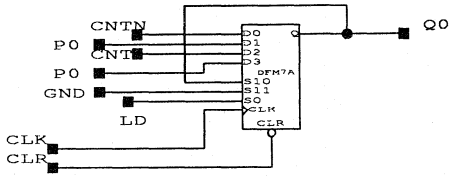


Figure 7 • Counter Bit Q0

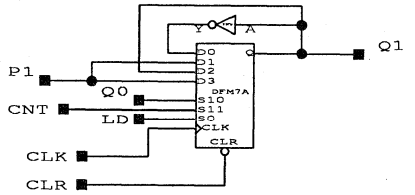


Figure 8 • Counter Bit Q1

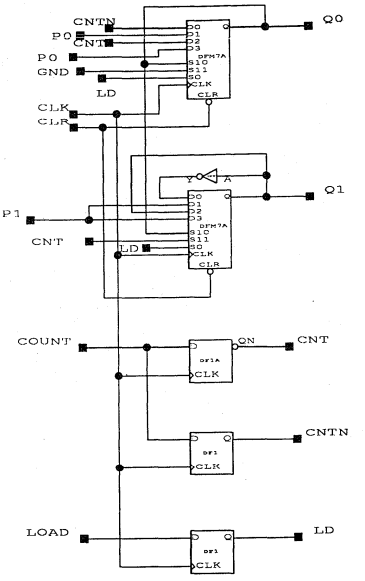


Figure 9 • CNT2P Macro

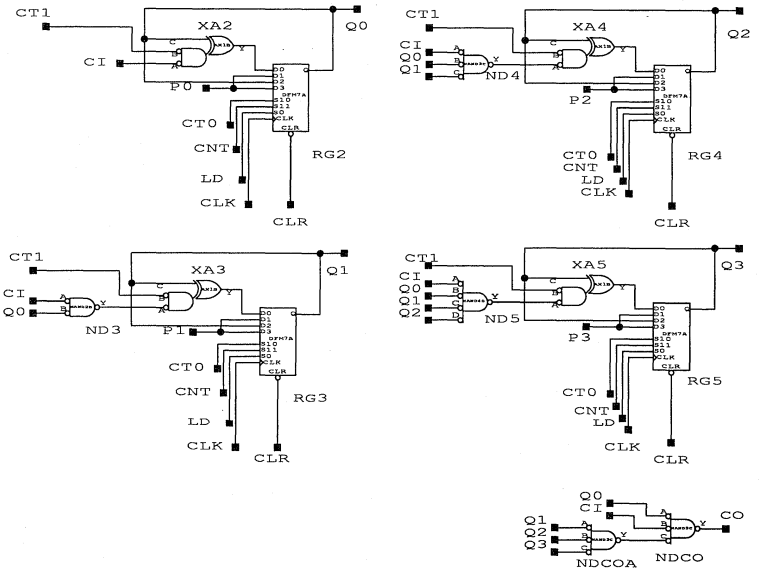


Figure 10 • 4-Bit Counter Macro

9

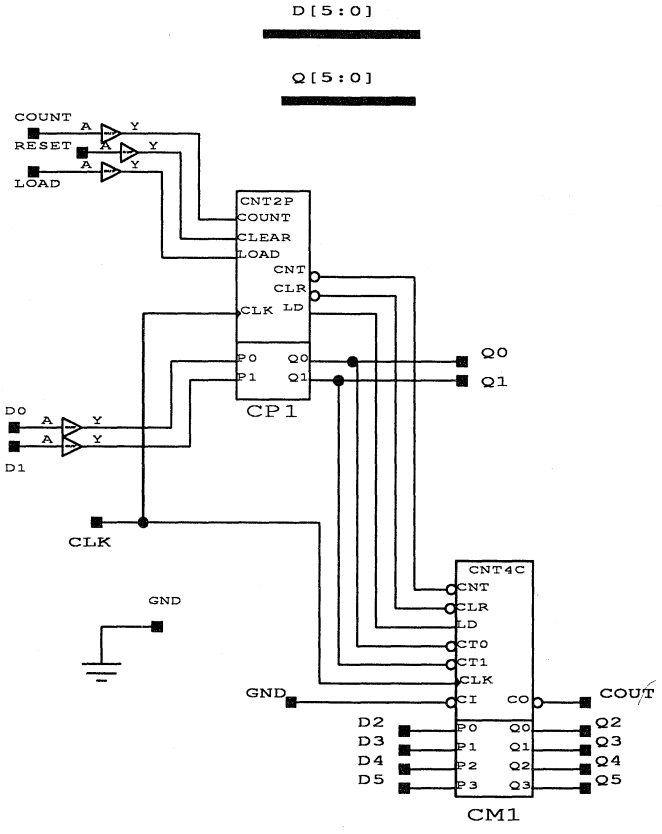


Figure 11 • 6-Bit Counter

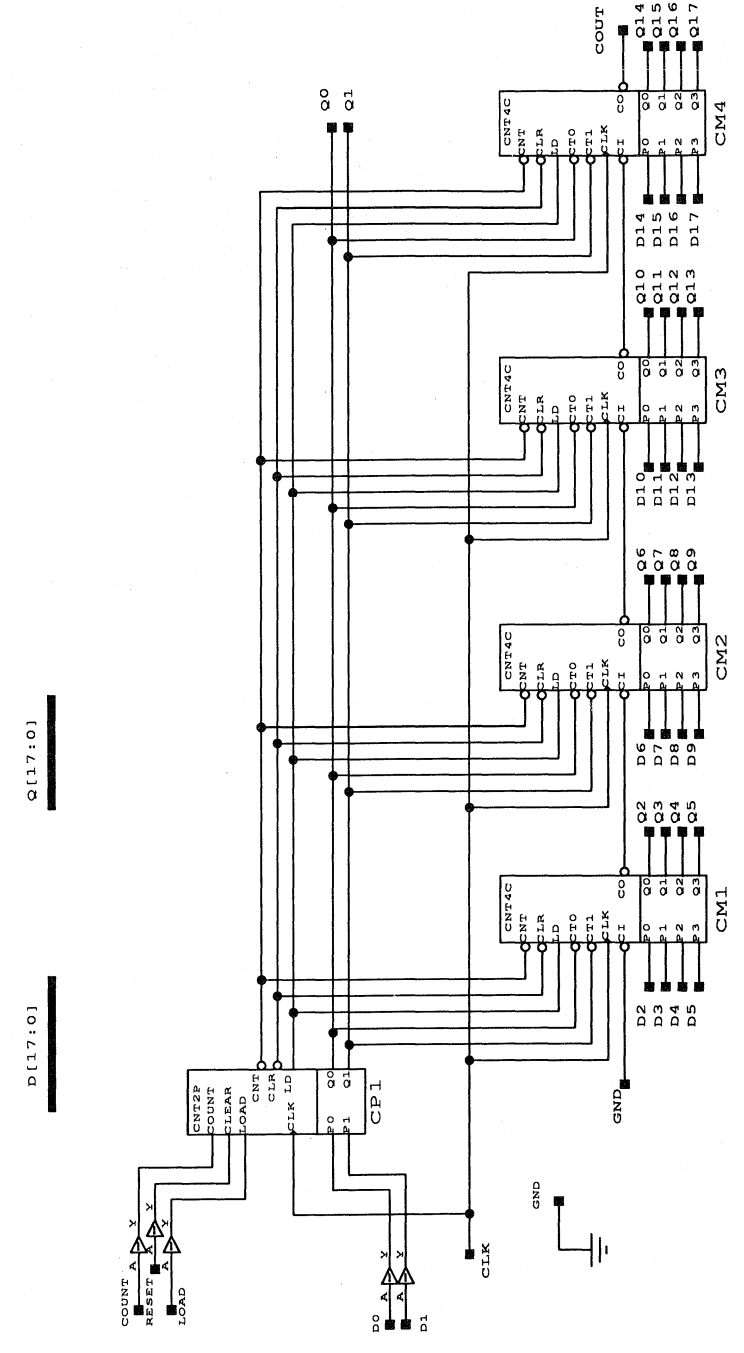


Figure 12 • 18-Bit Counter

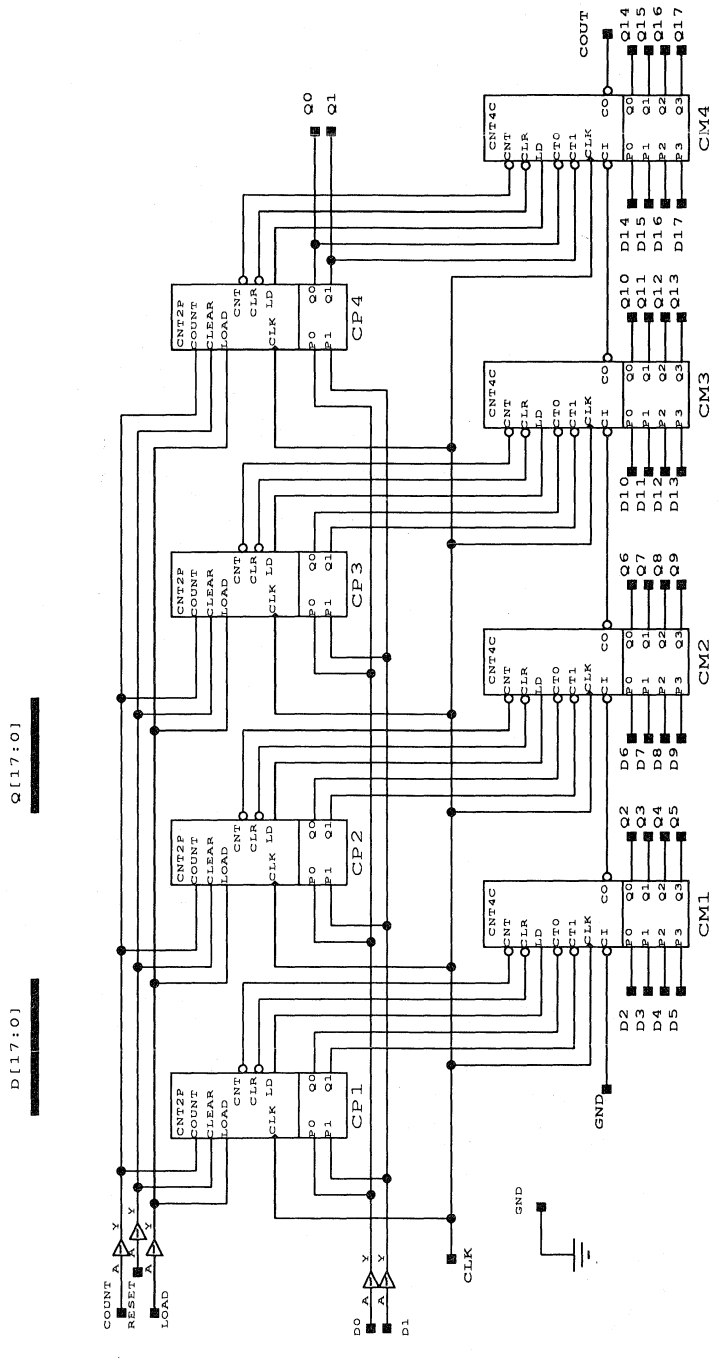


Figure 13 • 18-Bit Counter with Fanout Reduced Design

Implementing Multipliers with Actel FPGAs

Introduction

Hardware multiplication is a function often required for system applications such as graphics, DSP, and process control. The Actel architecture, which is multiplexer based, allows efficient implementation of multipliers with high performance. Furthermore, the Actel development tools allow the user quickly to create multipliers by using the appropriate algorithm and bit width needed for a specific application. The 1200XL family is the focus of the implementation of this application note, although other Actel families could also be used.

Multiplier Theory

The function of a binary unsigned multiplier, like its decimal counterpart, consists of a multiplicand (X), a multiplier (Y), and a product (P). The result is the product of the multiplier and the multiplicand (P = X * Y). Figure 1 shows the complete multiplication of two four-bit numbers producing an eight-bit product. As in decimal multiplication, the least significant digit of the multiplier combines with each digit of the multiplicand, forming a partial product (Y0X3, Y0X2, Y0X1, Y0X0). Three other partial products are similarly formed. To arrive at the final result, all four of the partial products are added (P7, P6...P0). Note that the most significant bit of the product (P7) is required, due to a possible carry from the other bits.

Conventional Multiplier Algorithm

The conventional approach to implementing a multiplier in digital logic is to AND individual multiplier and multiplicand bits to generate the partial products (PP1, PP2, PP3, PP4). For a four-bit multiplier, this would consist of 16 dual-input AND gates and three adders, as shown in Figure 2. The simplest method to sum the partial products is to have all three adders to be eight bits. Not all of the partial products generate eight bits, so smaller adders could be used. However, tracking which partial sums can be dropped and which need to propagate as carries to the next stage becomes complex and time-consuming, especially with larger bit widths. More important, the conventional multiplier implementation is resource intensive and does not produce optimal performance. Fortunately, another approach is possible.

L-Booth Algorithm Implementation

The L-Booth algorithm employs an alternative technique based on multiplexers, which are an ideal fit for the Actel architecture. For the four-bit implementation, the multiplier's two least significant bits are handled separately from the two most significant bits. Effectively, the multiplexer replaces the first stage of partial sum generation. Figure 3 illustrates the mathematics that explains the L-Booth algorithm. The two least significant multiplier bits (Y1, Y0) are handled separately from the two most significant bits (Y3, Y2). In both cases, the four possible combinations of the multiplier bits are covered with the multiplexer, resulting in the partial products PPA and PPB. Specifically for multiplexer A, the four combinations are zero (the trivial case), X (when Y1=0 and Y0=1), X shifted left or 2X (when Y1=1 and Y0=0), and 3X (when Y1=Y0=1). The multiplexers are eight bits deep to accommodate all eight possible inputs for the adders. To obtain the final product, the two partial sums are added with an eight-bit adder. High-speed adders are used in this implementation since shortest delay from input to output is the primary design constraint. Figure 4 shows the implementation of the L-Booth multiplier. The complete schematic for the fout-bit L-Booth multiplier is shown in Figure 5. Note the use of the five-bit adder to generate the required 3X input for the multiplexers. The name of the schematic is *LBMULT4*, indicating that it uses the L-Booth algorithm.

Multiplicand	>					**X3**	**X2**	**X1**	**X0**
Multiplier	>				x	**Y3**	**Y2**	**Y1**	**Y0**
1st partial product	>					Y0X3	Y0X2	Y0X1	Y0X0
2nd partial product	>				Y1X3	Y1X2	Y1X1	Y1X0	
3rd partial product	>			Y2X3	Y2X2	Y2X1	Y2X0		
4th partial product	>	+	Y3X3	Y3X2	Y3X1	Y3X0			
Final product	>	**P7**	**P6**	**P5**	**P4**	**P3**	**P2**	**P1**	**P0**

Figure 1 • Four-Bit Binary Multiplication

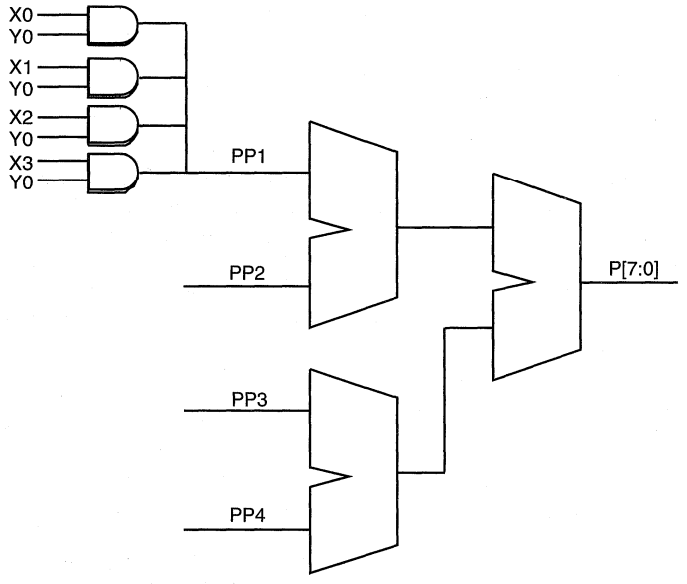


Figure 2 • Classical Implementation of Four-Bit Multiplier

						X3	**X2**	**X1**	**X0**
Only 2 LSB used	>				x			**Y1**	**Y0**
if Y1=0,Y0=0		I	0	0	0	0	0	0	0	0
if Y1=0, Y0=1		I	0	0	0	0	X3	X2	X1	X0
if Y1=1, Y0=0		I	0	0	0	X3	X2	X1	X0	0
if Y1=1, Y0=1		I	0	0	0	X3	X3+X2	X2+X1	X1+X0	0
Multiplexer A result	>		0	0	PPA5	PPA4	PPA3	PPA2	PPA1	PPA0
							X3	**X2**	**X1**	**X0**
Only 2 MSB used	>					x	**Y3**	**Y2**		
if Y3=0,Y2=0		I	0	0	0	0	0	0	0	0
if Y3=0, Y2=1		I	0	0	X3	X2	X1	X0	0	0
if Y3=1, Y2=0		I	0	X3	X2	X1	X0	0	0	0
if Y2=1, Y2=1		I	0	X3	X3+X2	X2+X1	X1+X0	0	0	0
Multiplexer B result	>		PPB7	PPB6	PPB5	PPB4	PPB3	PPB2	0	0
			0	0	PPA5	PPA4	PPA3	PPA2	PPA1	PPA0
		+	PPB7	PPB6	PPB5	PPB4	PPB3	PPB2	0	0
Final product	>		P7	P6	P5	P4	P3	P2	P1	P0

Figure 3 • Four-Bit Binary Multiplication Using L-Booth Algorithm

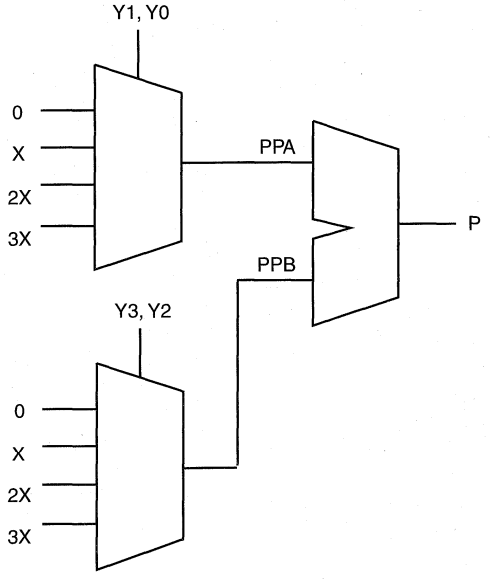


Figure 4 • L-Booth Multiplier Implementation

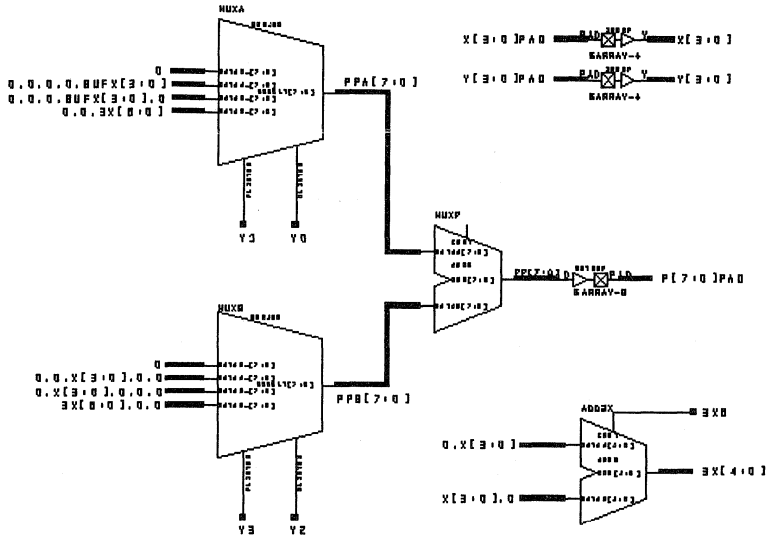


Figure 5 • Schematic Implementation of Four-Bit L-Booth Multiplier

Pipelined Multiplier

The previous multiplier implementations were entirely combinatorial. The output product is valid after all input values have propagated through the combinatorial logic. By introducing registers between the levels of logic, the stages of the multiplication can be broken up and synchronized with a clock. By doing so, the effective speed of multiple multiplications is increased, although the result is delayed by the number of register stages that are added. This delay is referred to as the *circuit latency*. Figure 6 shows the pipelined version of the four-bit multiplier, PMULT4.

Two levels of registers are used in the PMULT4 design, resulting in a latency of one cycle (i.e., the result appears one clock cycle later as shown in Figure 7). The distribution of registers is optimal since both stages contain three levels of logic. (The five-bit adder has two levels combined with one for the multiplexer, for a total of three levels. The eight-bit adder has three levels of logic internal.) This means that the multiplication can be done with three levels of combinatorial logic, one register, and data setup. Furthermore, with the 1200XL family, a combinatorial level is absorbed within the sequential module. This means that a four-bit multiplier could be done at a frequency in excess of 60 MHz. (Actual device performance is discussed in detail later in this application note.) The performance can be further increased at the cost of additional registers and circuit latency.

Design Tools

Designing multipliers with the Actel development tools is particularly easy since the basic blocks required (adders, multiplexers, and registers) can be quickly created with the ACTgen Macro Generator. Figure 8 shows the ACTgen main menu with macro category selections. As an example, an eight-bit fast adder with the name of *SAMPLE* will be created with ACTgen. The adder menu in Figure 9 shows the available options: adder variations, bus width, carry in, and carry out. The summary report is shown in Figure 10 and the generated symbol is shown in Figure 11. The multiplexers and registers can be created equally quickly with ACTgen. Using this approach, it is very easy to create multipliers of any bit width by changing the ACTgen parameters. Another modification that can be changed is the type of adder created by ACTgen. By selecting a ripple adder instead of a high-speed one, a more compact multiplier can be created. By making such a change, the four-bit multiplier would require 20 percent fewer modules.

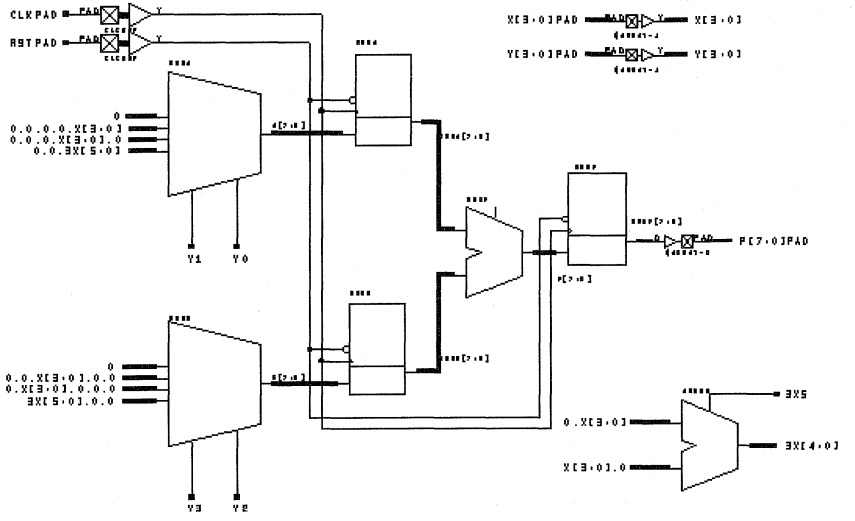


Figure 6 • Pipelined Four-Bit Multiplier

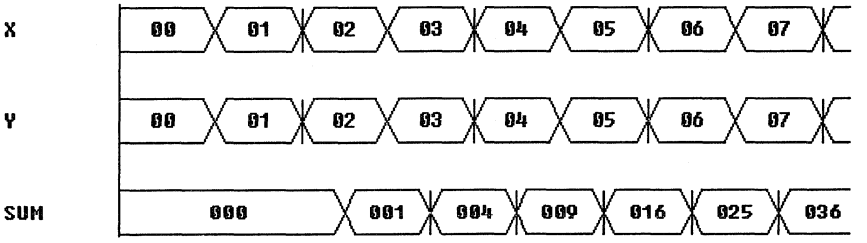


Figure 7 • Timing Waveform of Pipelined Four-Bit Multiplier with One Cycle Latency

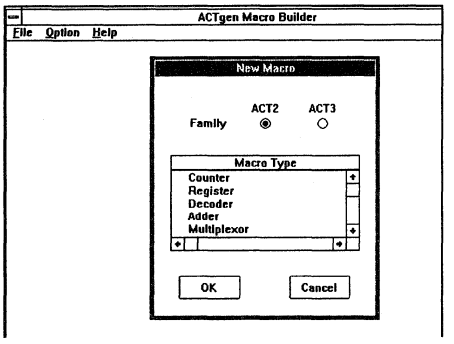


Figure 8 • ACTgen Macro Builder Main Menu

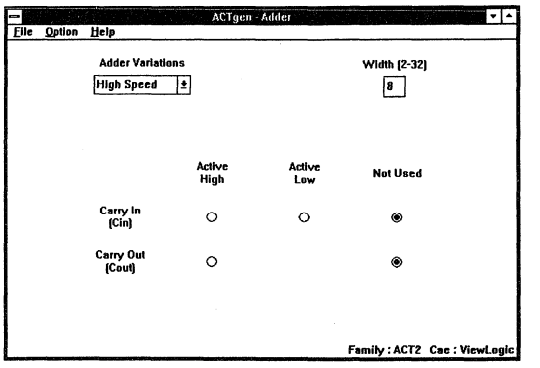


Figure 9 • ACTgen Macro Builder Adder Menu

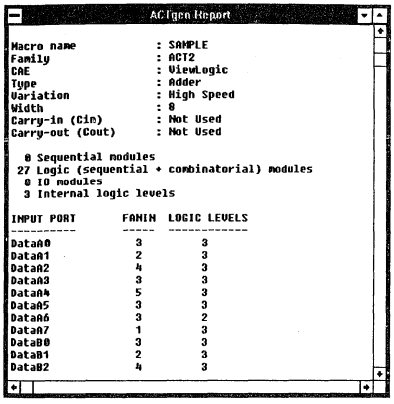


Figure 10 • ACTgen Macro Builder Summary Report

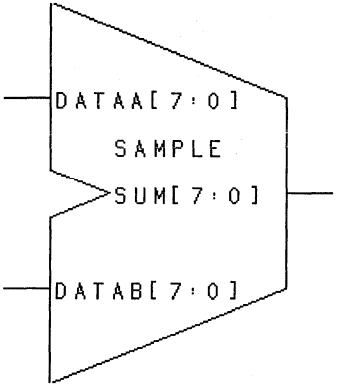


Figure 11 • ACTgen-Generated Adder Symbol

Hardware Description Language

Another approach to implementing multipliers is defining the functionality in Hardware Description Language (HDL). Generally, higher-level descriptions are fast to implement but somewhat slower than a manual or macro generator approach. A multiplier is particularly challenging since it is exclusively an arithmetic function, which is typically not well suited to pure synthesis algorithms. However, the Actel ACTmap Optimizer performs very well. The design was optimized for area and required only two modules more than the best manual implementation.

The VHDL source file for a four-bit multiplier is shown in Figure 12. Note how compact the required description is compared with previous schematic implementations. Furthermore, to create an eight-bit, sixteen-bit, or n-bit version would require only changing the BIT_VECTOR definitions.

```
library ASYL;
use ASYL.pkg_arith.all;

entity MULT4 is
        port(X, Y: in BIT_VECTOR(3 downto 0);
                P: out BIT_VECTOR(7 downto 0));
end MULT4;

architecture ARCHI of
        MULT4 is
        begin       P <= X * Y;
end ARCHI;
```

Figure 12 • VHDL Source Code for Four-Bit Multiplier

Multiplier Performance

All of the multiplier approaches discussed have been implemented in the Actel 1200XL devices. Table 1 shows all of these including many of the statistics required to make an educated decision on the best approach, depending on design needs. As with most designs, there is almost always a trade-off of system performance and design resources. In the case of the multiplier implementations, this is also true. In fact, there is a monotonic relationship of increasing performance with increasing module count for every multiplier implementation. The speed is obtained by implementing the design for the 1225XL-1 device and obtaining static timer worst-case commercial conditions after place and route. The speed refers to the worst-case path from input pad to output pad for the combinatorial multipliers and the longest internal-clock-to-data path (including data setup time). The "Util" column indicates what percentage of the 1225XL is being used by the multiplier.

Tables 2 and 3 show the statistics for eight-bit and sixteen-bit multipliers, respectively. As before, performance is based on actual placed and routed designs using the Actel static timing analyzer.

Conclusion

Multipliers can be quickly and easily implemented in Actel FPGAs, providing both efficient usage and high performance. The 1200XL family is particularly well suited for the applications. There is a range of options available to the user when designing multipliers: speed/area trade-offs, latency, and design method. Armed with this application information, the Actel development tools, and the 1200XL devices, designers can effectively create multipliers to meet their individual requirements.

Table 1 • Statistics of Four-Bit Multipliers

Design	Description	Device	Speed	# Modules	# Levels	Latency	Util
PMULT4	4 by 4 pipelined	1225XL-1	57 MHz	67	3	1 cycle	15%
LBMULT4	4 by 4 comb	1225XL-1	24 MHz	60	6	n/a	13%
PRMULT4	4 by 4 pipelined, ripple adder	1225XL-1	21 MHz	48	10	1 cycle	11%
VHDMULT4	4 by 4 comb, VHDL source	1225XL-1	19 MHz	50	8	n/a	12%
RBMULT4	4 by 4 comb, ripple adder	1225XL-1	14 MHz	48	12	n/a	11%

Table 2 • Statistics of Eight-Bit Multipliers

Design	Description	Device	Speed	# Modules	# Levels	Latency	Util
PMULT8	8 by 8 pipelined	1225XL-1	44 MHz	276	3	3 cycles	62%
LBMULT8	8 by 8 comb	1225XL-1	14 MHz	232	10	n/a	52%
PRMULT8	8 by 8 pipelined	1225XL-1	10 MHz	188	17	3 cycles	42%
RBMULT8	8 by 8 comb	1225XL-1	8 MHz	164	22	n/a	37%
VHDMULT8	8 by 8 comb, VHDL source	1225XL-1	8 MHz	325	22	n/a	73%

Table 3 • Statistics of Sixteen-Bit Multipliers

Design	Description	Device	Speed	# Modules	# Levels	Latency	Util
PMULT16	16 by 16 pipelined	1280XL-1	28 MHz	1011	4	3 cycles	83%
LDMULT16	16 by 16 comb	1280XL-1	8 MHz	844	16	n/a	69%
PRMULT16	16 by 16 pipelined	1280XL-1	5 MHz	786	33	3 cycles	64%
RBMULT16	16 by 16 comb	1240XL-1	4 MHz	656	40	n/a	96%

Designing State Machines for FPGAs

Introduction

The traditional methodology for designing state machines has been to draw a state diagram, map the states into the minimum number of register bits, and determine the next state function for each register bit. The minimum number of register bits needed can be determined by rounding up the natural log of the number of states. This methodology results in a minimum number of registers but usually requires wide gating and complicated logic to encode the next state bit. This scheme is necessary to implement state machines using programmable logic devices (PLDs) because of the PLDs inherent lack of registers. Because FPGAs do not have this limitation, other approaches are used for efficient state machines.

This application note will discuss techniques for efficiently implementing state machines by using the bit-per-state for Actel FPGAs.

Sample Bit-Per-State

Bit-per-state decoding is an effective approach because it takes advantage of the abundance of registers. The state diagram of a sample state machine is illustrated in Figure 1. This state machine is the control section of a four-channel DMA controller supermacro for Actel FPGAs. The state machine contains six states, seven inputs, and five outputs. Each circle represents a different state, and each arrow represents a transition between states. Inputs that cause state transitions are listed adjacent to the state transition arrows. Control outputs are labeled along with the states inside the circles. For example, in state S2, the state machine asserts the CNTD and CMREQ outputs. If the MACK input is low, the state machine remains at state S2. If the MACK input is high, the state machine goes to state S3 and asserts the CE output in the process.

State Transition Equations

The next step is determining the logic to generate the state sequence. For bit-per-state implementation, assign each state to a separate register and write a state transition equation for each register. The state diagram in Figure 1 shows that state S0 can be realized when state S4 is asserted and the input CONT is low, or it remains at state S0 if all four inputs—A, B, C, and D—are low. Therefore the transition equation of state S0 can be written as

S0 := /A*/B*/C*/D*S0 + /CONT*S4

Similarly, state S1 can be achieved when state S0 is asserted and any one of the four inputs—A, B, C, or D—is high, or it remains at the same state (S1) if the input PBGNT is low. The transition equation of state 1 can be derived as

S1 := (A+B+C+D)*S0 + /PBGNT*S1

The complete state transition equations for the state machine are listed in Table 1. Note that state S3 will go to state S4 unconditionally; therefore, the transition equation for state S3 can be written as S3 := S4.

Table 1 • State Machine Transition Equations

```
S0 := /A*/B*/C*/D*S0 + /CONT*S4;
S1 := (A+B+C+D)*S0 + /PBGNT*S1;
S2 := PBGNT*S1 + /MACK*S2;
S3 := /MACK*S2 + MACK*S3;
S4 := S3;
S5 := CONT*S4 + /MACK*S5;
```

Output Equations

Once the state transition equations are determined, the output equations can be written by encoding the states. In some cases, the output is active only in one state. Therefore, the output is simply a function of that state. For example, CE is active only in state S3, so the output equation for CE is simply CE = S3. Other output may be active in more than one state. The output equations can be written simply as functions of those states. For example, CMREQ is active in state S2 as well as state S5. Therefore, the output equation is

CMREQ = S2 + S5

Table 2 lists the output equations of the state machine.

Table 2 • Output equations

```
PBREQ = S1;
CLD = S4;
CNTLD = S2;
CMREQ = S2 + S5;
CE = S3;
```

9

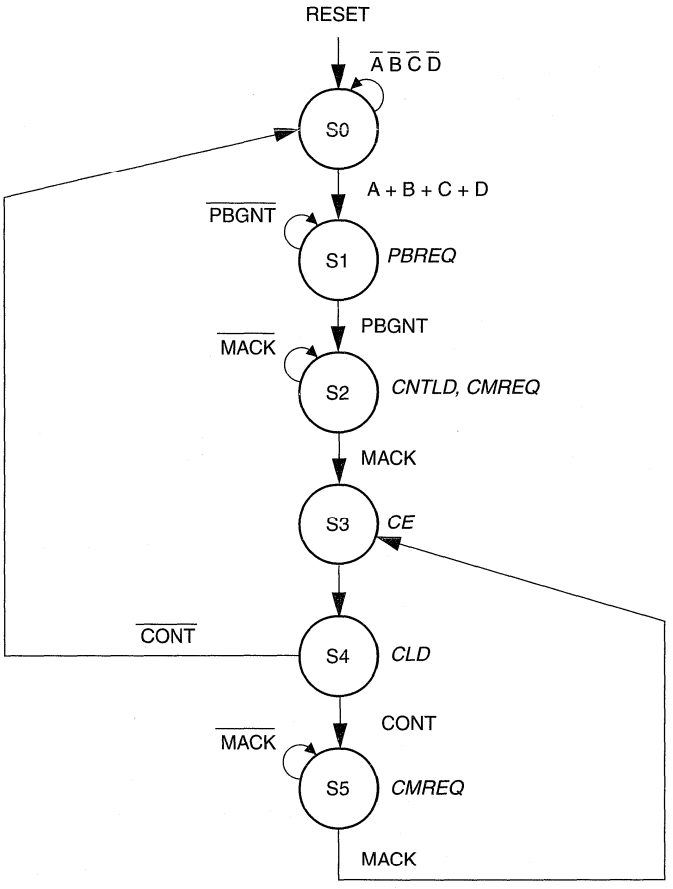


Figure 1 • Four-Channel DMA Controller State Diagram (Control Section)

The state machine can be captured using schematic entry, or it can be automatically mapped by using synthesis tools such as Actel ACTmap Optimizer. The completed PALASM file is shown in Figure 4. A reset signal RST has been added to initialize the circuit. The equations have been modified to implement a synchronous reset to ensure combinable flip-flops are selected by ACTmap.

Figure 2 shows the schematic of the state machine implementation using the bit-per-state approach in the ACT™ 3 family. The circuit requires 18 logic modules after combining timing for the 1425A-2, as shown in Figure 3.

Summary

A summary of the bit-per-state methodology as follows:

1. Draw a state diagram.
2. Assign each state to a separate register.
3. Write a state transition equation for each register.
4. Derive output equations based on active states.

Larger state machines can be implemented using this technique by distributing control to several smaller state machines and using a single master machine to coordinate activities among the state machines. This usually results in higher-performance designs. It is also easier to design and debug simpler and smaller state machines.

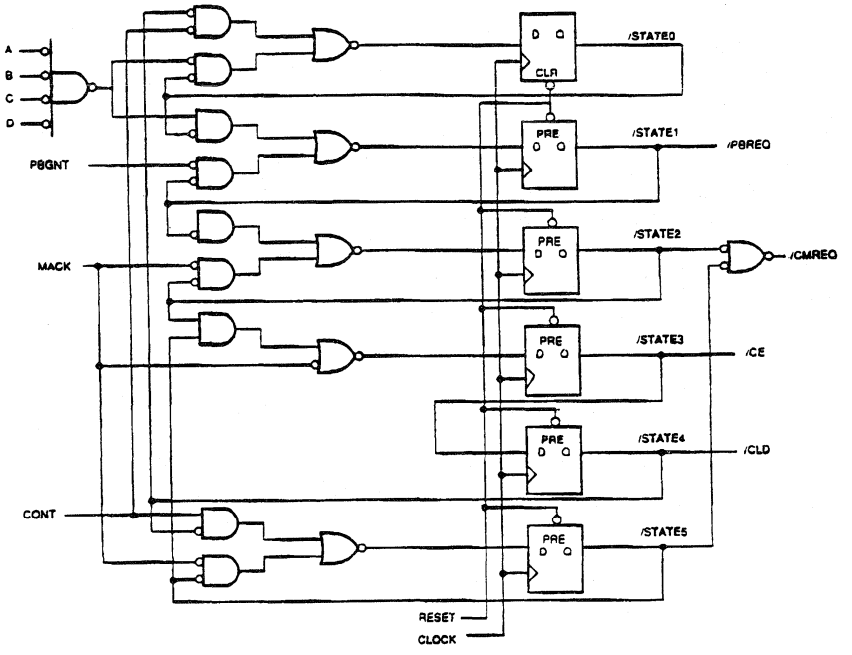


Figure 2 • State Machine Schematic in ACT 3 Family

DF1 NOR4A NOR2A

Worst-Case Path (S0 feedback)

$= t_{CO} + t_{RD2} + t_{PD} + t_{RD1} + t_{PD} + t_{RD1} + t_{SUD}$

$= 2.3 + 1.4 + 2.3 + 1.0 + 2.3 + 1.0 + 0.6$

$= 10.9$ ns

Figure 3 • State Machine Timing Using 1425A-2

```
CHIP state1 generic
clk rst s0 s1 s2 s3 s4 s5 a b c d cont pbgnt mack pbreq cld cntld cmreq ce

EQUATIONS
s0 := rst*(/a*/b*/c*/d*s0 + /cont*s4) + /rst
s1 := rst*((a+b+c+d)*s0 + /pbgnt*s1)
s2 := rst*(pbgnt*s1 + /mack*s2)
s3 := rst*(/mack*s2 + mack*s3)
s4 := rst*s3
s5 := rst*(cont*s4 + /mack*s5)

pbreq = s1
cld = s4
cntld = s2
cmreq = s2 + s5
ce = s3
```

Figure 4 • Complete State Machine PALASM file

Shift Register Design

Shift registers with serially controlled data inputs and parallel outputs can be used as powerful controlled sequence generators. As a result, shift registers can be used in high-speed state machine designs.

For this application, a shift register sequentially shifts a single logic high signal while the rest of the output bits are low. This function can be implemented by simply connecting the output of the one flip-flop to the input of the next flip-flop and the output of the last flip-flop back to the input of the first flip-flop. Table 3 shows the function table of an 8-bit serial shift register. The width of the shift register is expandable by serially adding more flip-flops to the last stage. Figure 6, on the next page, shows the schematic of an 8-bit shift register. Note that the shift register is designed so that it can be set to a known state with a reset signal.

State Machine Implementation

The state machine implementation is best illustrated by an example. The sample state machine has six states with three output bits. The sequence is organized such that only one output bit changes state for every clock pulse. Figure 5 shows the state diagram of the state machine.

A six-state state machine requires a six-bit shift register with one register per state. Using this shift register determines the state sequence map of the state machine. The state machine creates the three output bits based on decoding the outputs of the shift register. The decoding logic is greatly minimized because there is only one output bit asserted in any state. Table 4 shows the state table of the state machine. When the outputs of the state machine are 001, the shift register outputs are 000010. In this state, Q5, Q4, Q3, Q2, and Q0 are all low and Q1 is high. Therefore, the state machine uses only Q1 for the decoding logic. The logic for the state machine outputs is based on the shift register outputs. Out0 is high when Q1 or Q2 or Q3 is high, Out1 is high when Q2 or Q3 or Q4 is high, and Out2 is high when Q3 or Q4 or Q5 is high. The complete decoding logic equations are the following:

Out0: Q1 + Q2 + Q3
Out1: Q2 + Q3 + Q4
Out2: Q3 + Q4 + Q5

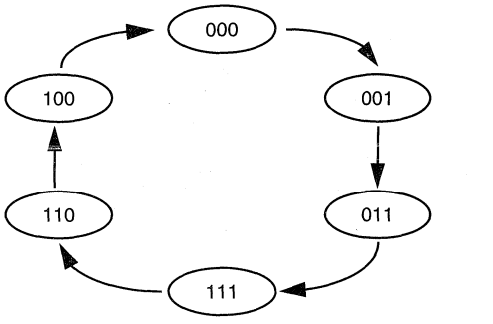


Figure 5 • State Diagram of State Machine

Table 4 • State Table for Example State Machine

Shift Register Outputs						State Machine Outputs		
Q5	Q4	Q3	Q2	Q1	Q0	Out2	Out1	Out0
0	0	0	0	0	1	0	0	0
0	0	0	0	1	0	0	0	1
0	0	0	1	0	0	0	1	1
0	0	1	0	0	0	1	1	1
0	1	0	0	0	0	1	1	0
1	0	0	0	0	0	1	0	0

Figure 7 shows the schematic diagram of the state machine using ACT™ 2 or ACT 3 macros. Note that the decoding logic requires only one level of logic to implement. Thus, using a shift register to implement state machines improves performance significantly.

Table 3 • 8-Bit Shift Register Function Table

RST	CLK	Q0	Q1	Q2	Q3	Q4	Q5	Q6	Q7
0	X	1	0	0	0	0	0	0	0
1	↑	0	1	0	0	0	0	0	0
1	↑	0	0	1	0	0	0	0	0
1	↑	0	0	0	1	0	0	0	0
1	↑	0	0	0	0	1	0	0	0
1	↑	0	0	0	0	0	1	0	0
1	↑	0	0	0	0	0	0	1	0
1	↑	0	0	0	0	0	0	0	1
1	↑	1	0	0	0	0	0	0	0

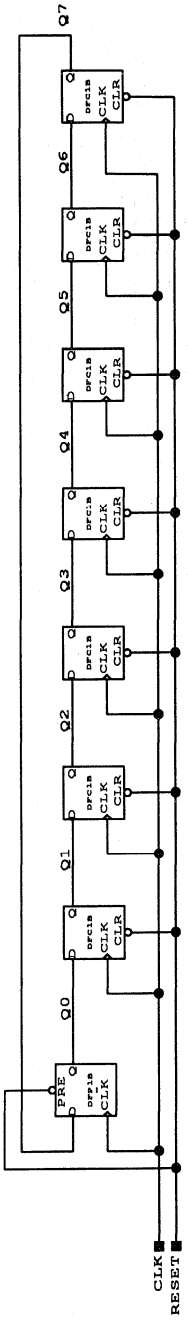


Figure 6 • Eight-Bit Shift Register Schematic

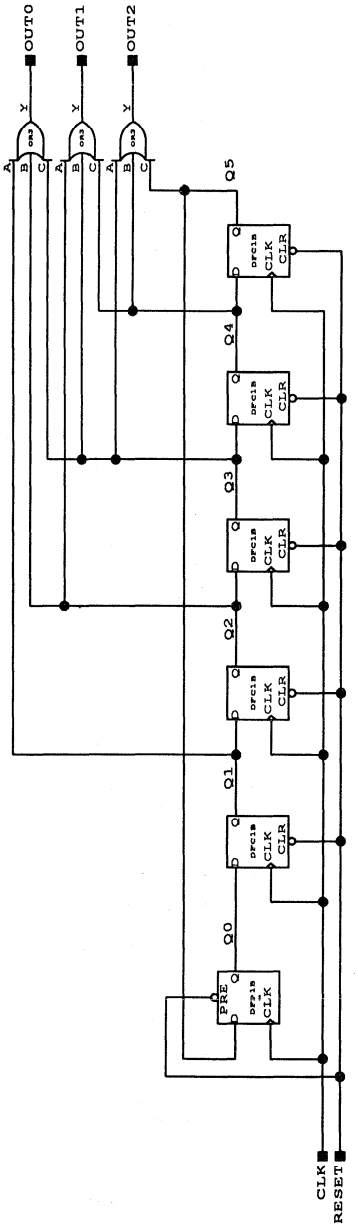


Figure 7 • State Machine Schematic

Conclusion

To achieve the high performance possible with FPGAs, new register-rich techniques are needed. Two effective approaches include bit-per-state and shift register decoding. The user can manually implement these techniques or select FPGA-friendly algorithms with optimization tools such as Actel ACTmap Optimizer.

Designing with Pseudo-Random Number Generators

Introduction

As counter sizes increase, the amount and complexity of support logic also increase. With this increase, the maximum operating speed of the counter decreases. This application note describes an easy way to build pseudo-random number (PRN) counters using very few logic resources. These counters produce nonlinear sequences that can be used in many applications.

In many cases, only a simple modulo counter is required to generate a stream of clock pulses. Nonlinear counters can also be used to generate memory addresses. For example, in a FIFO, the order in which the memory is accessed is irrelevant as long as the data is stored and retrieved in the same order. This memory addressing technique could also be used for a ROM look-up table (LUT). In this case, the LUT data would be stored in ROM in the PRN sequence using a simple software routine to precalculate the addresses.

Pseudo-Random Counters

The most popular (and the simplest) PRN counter is the feedback shift register. Figure 1 shows a shift register of length m bits with the exclusive-OR (XOR) of the nth bit and the last (mth) bit fed back to the first bit location.

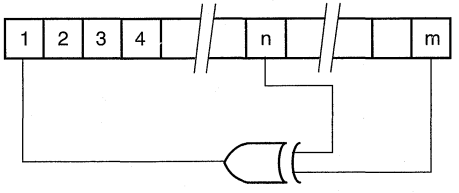


Figure 1 • Feedback Shift Register

The PRN counter goes through a set of states (defined by the set of bits in the registers after each clock), eventually repeating itself after K clock pulses. The maximum number of conceivable states of an m-bit register is $K = 2^m$, that is, the number of binary combinations of m bits. However, the state of all zeros would get "stuck" in this circuit, since the XOR would generate a zero at the input. Thus, the maximum number of unique states is $2^m - 1$. It turns out that you can make "maximal-length shift register sequences" if m and n are chosen correctly. The resultant state sequence is pseudo-random. As an example, consider the 4-bit feedback shift register in Figure 2. Beginning with the state 1111 for the XOR implementation type and 0000 for the exclusive-NOR (XNOR) type, there are 15 distinct states ($2^4 - 1$), after which states repeat.

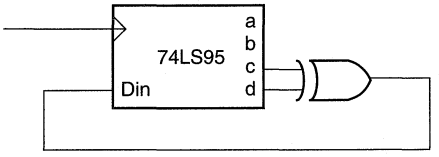


Figure 2 • A 4-Bit Feedback Shift Register

Feedback Taps

Maximal-length shift registers can be made with XOR/XNOR feedback from more than two taps (in these cases, you use several XOR/XNOR gates in the standard parity tree configuration, that is, the modulo 2 addition of several bits). In fact, for some values for m, a maximal-length register can only be made with more than two taps. Table 1 shows a listing of all values of m up to 33 for which maximal-length registers can be made with just two taps, that is, feedback from the nth bit and mth (last) bit. A value is given for n and for cycle length K, in clock cycles. In some cases, there is more than one possibility for n, and in every case the value of $m - n$ can be used instead of n; thus the earlier 4-bit example could have used taps at $n = 1$ and $m = 4$.

Since shift register lengths of multiples of eight are common, you may want to use one of those lengths. In that case, more than two taps are necessary. There are other cases of shift register lengths that require more than two taps. Table 2 shows the taps points for the multitap point counters. Because of the greater complexity of the feedback path, the multitap counters tend to operate slower than the single tap version. It is sometimes more prudent to use a counter that is larger than needed just to reduce the number of taps. For example, if a modulo 250 counter is required, an 8-bit counter with three tap points could be chosen. But the wiser choice would be to use a 9-bit (max modulo 511) counter. The 9-bit counter actually uses two less logic modules (see next section) and has a higher operating speed.

Table 1 • Table 1.Values for Maximal-Length Registers Made with Two Taps

Length (m)	Tap (n)	Maximum Count (K)
2	1	3
3	2	7
4	3	16
5	3	31
6	5	63
7	6	127
9	5	511
10	7	1,023
11	9	2,047
15	14	32,767
17	14	131,071
18	11	262,143
20	17	1,048,575
21	19	2,097,151
22	21	4,194,304
23	18	8,388,607
25	22	33,554,431
28	25	268,435,455
29	27	536,870,911
31	28	2,147,483,647
33	20	8,589,934,591

Table 2 • Table 2.Tap Points for Multitap Counters

Length (m)	Tap (n)	Maximum Count (K)
8	3,4,5,7	255
12	1,9,10,11	4,095
13	6,10,11,12	8,191
14	1,11,12,13	16,383
16	10,12,13,15	65,535
24	16,21,22,23	16777215

PRN Counter Construction

Constructing a PRN counter requires only two elements—a loadable shift register bit and a fast XOR/XNOR gate. The Actel family of field programmable gate arrays (FPGAs) is ideal for building these types of elements. To construct the loadable shift register bit, use the DFM (D-type flip-flop with a 2 to 1 multiplexer) shown in Figure 3. Use the DFM6A hard macro with enable inputs to build the shift registers for cascadable PRN counters. The fast XOR/XNOR gate is implemented by one combinatorial logic module. A PRN counter can be constructed with these two basic circuits elements. Finally the END pattern can be chosen to minimize the complexity of the end-of-sequence (EOS) detection circuit.

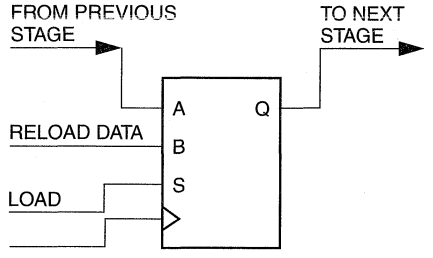


Figure 3 • Loadable Shift Register Bit

One key feature of the PRN counter is that, as the counter length increases, the extra bits only add one extra shift register bit without increasing the feedback delay. For example, a 10-bit continuous PRN counter will take only 10 shift register bits (one DFM each) plus a fast XOR (one module). Increasing the length of the counter to 20 bits requires only an additional 10 shift register modules. The important point is that the feedback path still has only one level of logic delay. In other words, this 20-bit counter has the same feedback delay (maximum operating frequency) as the 10-bit counter.

PRN Sequence Software

Actel has developed a software utility called PRN_GEN2 for constructing PRN sequence counters. The software prompts you for the counter size (in bits), the end-of-sequence (EOS) pattern, the type of counter (XOR/XNOR), the desired modulo (cycles), and the pipeline delay of the EOS detect. Then, it provides you with the proper number to load to achieve the desired modulo and the tap points needed.

Note: You can have multiple modulos and use a multiplexer to select them.

Conclusion

PRN counters are easy to design and implement. They are useful for many types of applications and designs in Actel FPGAs. For more information about PRN counters and their implementation in Actel FPGAs, contact Actel Technical Support at 1-800-262-1060.

References:

1. Paul Horowitz and Winfield Hill, *The Art of Electronics*, Cambridge University Press.

JTAG Implementation in Actel Devices

Introduction

JTAG (Joint Test Action Group) is a test approach used to debug various problems caused during manufacturing. It requires special functions built into a device. This application note describes the basic information needed to implement JTAG; however, the designer should refer to the IEEE Std. 1149.1 to ensure compliance. This note also considers alternate test approaches significantly less costly than JTAG. There are 15 JTAG macros available in the standard ACT 2 macro library and these are briefly discussed.

JTAG Implementation

JTAG consists of two major functions, the scan cells and the TAP (Test Access Port) controller that controls the shifting and loading of the scan cells. There are three types of scan cells: the basic I/O scan register, the instruction register, and the bypass register. The block diagram for the complete JTAG structure is shown in Figure 1. The TAP controller drives the control signals for the shifting and loading data in the bypass register, instruction registers, and I/O scan registers. The instruction register contains information that defines the current test type for the I/O scan registers, and both the instruction register and TAP controller supply information to the MUX SELECT DECODE to determine the source for TDO. The different types of tests supported by JTAG are shown in Table 1. Implementation for the EXTEST, SAMPLE/PRELOAD, INTEST, BYPASS, and RUNBIST will be considered in the following sections. For a complete description of optional JTAG functions and test types, please refer to the IEEE Std. 1149.1.

Table 1 • JTAG Test Types

BYPASS	Mandatory
SAMPLE/PRELOAD	Mandatory
EXTEST	Mandatory
INTEST	Optional
RUNBIST	Optional
IDCODE	Optional
USERCODE	Optional

JTAG Soft Macros

Actel has created 15 soft macros to aid the designer in implementing JTAG. The list of macros and their function in shown in Table 2.

Table 2 • ACT 2 JTAG Soft Macros

Macro Name	Macro Function
IDREG1, IDREG2	ID Register
IRCELL1, IRCELL2	Instruction Register
BBCELL1, BBCELL2	Bi-directional Boundary Scan Cell
UBCELL1, UBCELL2	Uni-directional Boundary Scan Cell
TIM1, TIM2	Test Interface Macro
IREG1	2-bit Instruction Register
IREG2	3-bit Instruction Register
TDOCELL	Test Data Outpull Cell
BYPREG	By-pass Register
TAP	Test Access Port

Implementing the Basic I/O Scan Register

The signal SHIFTDR is used to select between system data and scan data. The signal CLOCKDR is used to capture the data selected by the SHIFTDR signal. The UPDATEDR signal controls the clock to a flip-flop used to isolate the output pin from data being shifted through the device.

Implementing Instruction Register Cells

Instruction registers are used to encode the current instruction. These values are decoded and used to create the MODE signal that defines the functions in the JTAG scan macros. The number of instruction registers required is defined by several factors. The first is the number of test types that must be performed and are used to encode the MODE signals. The second is to control the multiplexer select signals used to define the source for the TDO signal shown in Figure 1.

The DATA input is used to load optional test information into the instruction scan register. JTAG requires that the DATA input on the least significant bit of the instruction register (that is, closest to TDO) be set to a logical one and that the second least significant bit be set to a logical zero. Optional status data can be input to any additional instruction registers.

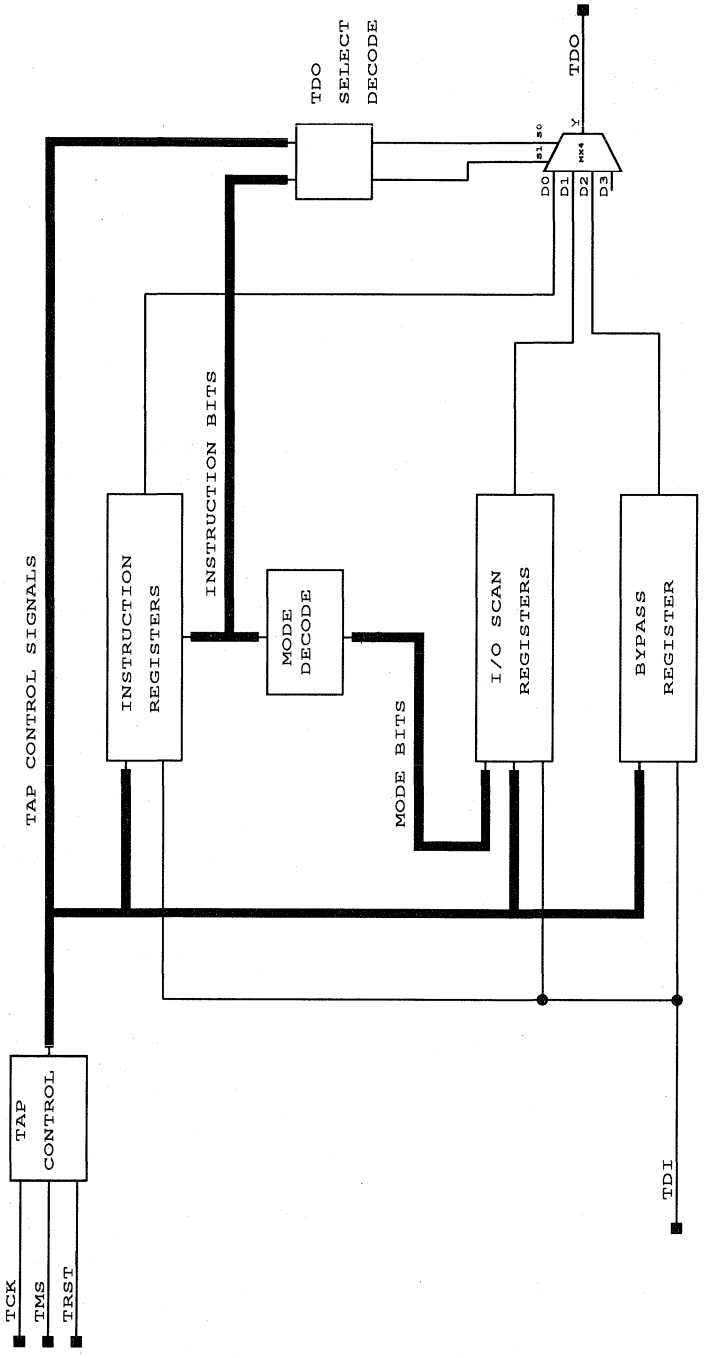


Figure 1 • JTAG Block Diagram

The TRST_ and RESET_ lines are used to force the value of the instruction registers to the default test type. These signals can be used to SET or RESET the instruction bits. The instruction IDCODE is the default instruction if it is implemented. If the IDCODE test is not implemented, then the BYPASS test is the default test type. The instruction bits to define the BYPASS test should all be logical ones. Therefore, the flip-flops should all be SET for this test case.

Information from the instruction register is passed to the TDO output whenever the TAP controller signal SHIFTIR is active (high).

Bypass Register

The bypass register is a required element for JTAG implementation. This register is used to bypass the normal I/O scan cells to shorten test time. The long scan chains are bypassed by storing the current information present on TDI and passing it to TDO selected by the multiplexer values as shown in Figure 1. Once again, the instruction register must be set to all ones to define the BYPASS function.

Implementing the TAP Controller

The TAP controller is the state machine that creates the various signals controlling the activity of the scan registers. The state machine is updated on the rising edge of TCK and the registered control signals are updated on the falling edge of TCK. The TMS signal is used to define branching in the state machine.

Architectural Considerations

When implementing JTAG in an Actel ACT 2 device, the following should be considered:

- Since the scan chains are basically shift registers, the designer must guarantee that there are no hold time violations caused by clock skew. The easiest way to prevent this is to use a global clock buffer (CLKINT) to drive the CLOCKDR signal. Otherwise, a buffer tree must be constructed and careful timing analysis performed.
- All I/O scan chains should be tied together after pin assignment has been made and preliminary place and routes have been done. Tying together I/Os that are physically close together reduces routing requirements. The same is true for the UPDATEDR, MODE, and SHIFTDR signals. Typically these signals will be buffered before the actual I/O scan cells. Any given buffer should only drive the UPDATEDR and SHIFTDR signals on I/Os close together (in groups of 4 to 8 pins). To determine which I/Os are close together, please refer to bonding diagrams and pin cross-reference charts available from Actel Technical Support.

Actel's tool set contains a function that eliminates all modules that do not drive a sink. These recommended steps will cause this to happen and will skew the total required logic resources. To prevent this from occurring, the PRESERVE option described in the *Designer Series User's Guide* should be used on all nets without sinks (this can be done in the JTAG soft macros).

Impact on Performance

JTAG implementations can also affect performance. The input scan cell puts an additional load on the normal system circuitry. The additional load will typically add some delay (less than one nanosecond) to the system signal. Outputs are much more heavily impacted because of the need to place a 2 to 1 multiplexer between system output data and the output pin. The multiplexer can add delay of 5 to 10 ns depending on routing and speed grades.

Impact on I/O and Clock Resources

JTAG implementation also requires I/O dedication. The TAP controller requires a minimum of five I/O resources. The I/Os are as follows:

- TCK – Test Clock Input
- TMS – Test Mode Select
- TDI – Test Data Input
- TDO – Test Data Output
- TRST – Optional Test Reset Input

A global clock resource is also used for the CLOCKDR signal, leaving only one for system level logic.

Alternatives to JTAG

The purpose of JTAG is to ease the testing and debugging of manufacturing problems at the board level. Different test approaches exist that use significantly fewer logic and I/O resources than JTAG and could be used when either is a problem. The following sections describe various test structures that aid board-level tests that have significantly lower logic requirements.

Using Actel Built-in Test Structures

One of the functions built into Actel devices is a feature called the Actionprobe® diagnostics. This function provides 100 percent observability of internal nets by serially addressing various points in the array. Internal nets are addressed by driving address information and a qualifying clock on the SDI and DCLK pins. Internal activity can then be observed on the PRA and PRB pins. This function can be used to verify device connection to a board by first addressing an internal net associated with the I/O, then stimulating an external pad, and finally observing the effect on an external probe pin. Equivalently, NETD could be selected, and then activity on inputs IN1, IN2, and IN3 could be observed. This test method requires no dedicated logic to implement. Control of the Actionprobes is described in the *Designer Series User's Guide*.

Use of the Actionprobes can be extended to all pins, including outputs, by changing the selection of the output cells. If bidirectional outputs are chosen rather than standard outputs, a new path is created back into the array that can be observed with the Actionprobes. Since the outputs will continue to drive during the test (unless disabled), the tester must temporarily back-drive the pin to a specified state for observation on the probe pins. Back-driving can be avoided by adding tristate circuitry to the outputs under test and disabling the outputs for the test. Since the input portion is not tied to any internal resources, the PRESERVE option (described in the *ALS User's Manual*) must be used to prevent the software from eliminating the nonfunctional input net. Once again, no additional logic is required to implement this test method.

In addition to the Actionprobe diagnostics, a special function to tristate all I/Os is also built into all Act 2 devices. For a complete description of this test feature, please contact Actel Technical Support.

Non-JTAG Test Structures

The following sections describe test structures that can be designed into the part to aid in board-level tests. The test structures described use significantly fewer logic resources than JTAG and also can reduce the impact on performance.

I/O Mapping

This built-in test method simply maps inputs directly to outputs with a multiplexer at the output selecting between normal system data or input test data. The test is done by first asserting the TEST pin. The inputs can then be toggled and their function observed on the outputs. If the number of inputs exceeds the number of outputs, then simple combinatorial functions can be added to map many inputs to a single output. If the number of outputs exceeds the number of inputs, then a single input can be tied to multiple outputs. Bidirectional pins can be mapped as inputs or outputs, but not both.

I/O mapping also requires architectural considerations in mapping inputs to outputs and should only be done once pin assignment is complete. Inputs should be mapped to outputs that are close together to minimize routing and performance problems. The impact of this test method is basically one pin for test and one multiplexer for each output in addition to the delay associated with the multiplexer.

Input Only Test Structure

Typically, a functional test to toggle outputs can be generated with a subset of functional patterns. The difficulty is creating test vectors that can trace faults to specific inputs. The input only test method adds logic to inputs apart from normal chip function. This test method requires all of the inputs and bidirectional pins to be tied together through an OR/AND structure. The test is performed by asserting the test pin and driving all inputs low. Each input is then toggled and observed on the output pin. When bidirectional pins are used as inputs, the test must guarantee that the bidirectional pins are in the tristate condition during the test. If this cannot be guaranteed, additional test logic must be built in for the bidirectional pins. The cost of this test method is

summation k

k = 1,2,3

$$\frac{n+m}{4^k}$$

n = inputs

m = bidirectional pins

This test can be further extended by making every output bidirectional and using the TEST pin to disable the driver during the test. No functional test patterns are then required. To estimate costs, the total I/O should be considered in the summation. This test method would require about 4 percent of a 1280 with all pins utilized.

When implementing this test method, pins should be tied into the OR/AND test array only after pin assignment. All pins that drive a unique OR/AND module should be closely associated on the chip.

Conclusion

JTAG is an effective test method to aid in verifying and debugging board-level problems. When implementing JTAG in Actel ACT 2 parts, the resources required as well as the impact on performance should be considered. Architectural considerations should be made to minimize the impact on performance and routing. Where the impact to logical resources or pins is unacceptable, the designer should consider optional test methods that can also help in the testing and debugging process.

Synchronous Dividers in Actel FPGAs

Synchronous dividers are useful in numerous applications, such as prescalers, timing generators, and multi-phase clocks. Although ripple dividers use less logic, glitches and potential race conditions make them hazardous in all but the simplest applications. Synchronous dividers offer a better solution for reliable operation.

Figure 1 shows divide-by-two to divide-by-ten synchronous dividers using combinable Actel logic modules. Using the Actel ACT 2, 1200XL, and ACT 3 architectures, the combinatorial gates will be absorbed with the following flip-flop. The flip-flops used do not have reset pins as dividers and are generally used as astable devices. The OR-AND gates are used for some of the dividers to avoid nonconvergent illegal states. To initialize the dividers for simulation, the output can be held high and clocked once for each flip-flop, then released. For example, with the ViewSim simulator, the divide-by-three synchronous divider (with output node DIV3) would be initialized with the following command file sequence:

```
h DIV3
cycle 2
r DIV3
cycle 5
```

The resulting waveforms for each of the dividers are shown in Figure 2. Note that the duty cycle is 50 percent for all even dividers and as close to this as possible for the others.

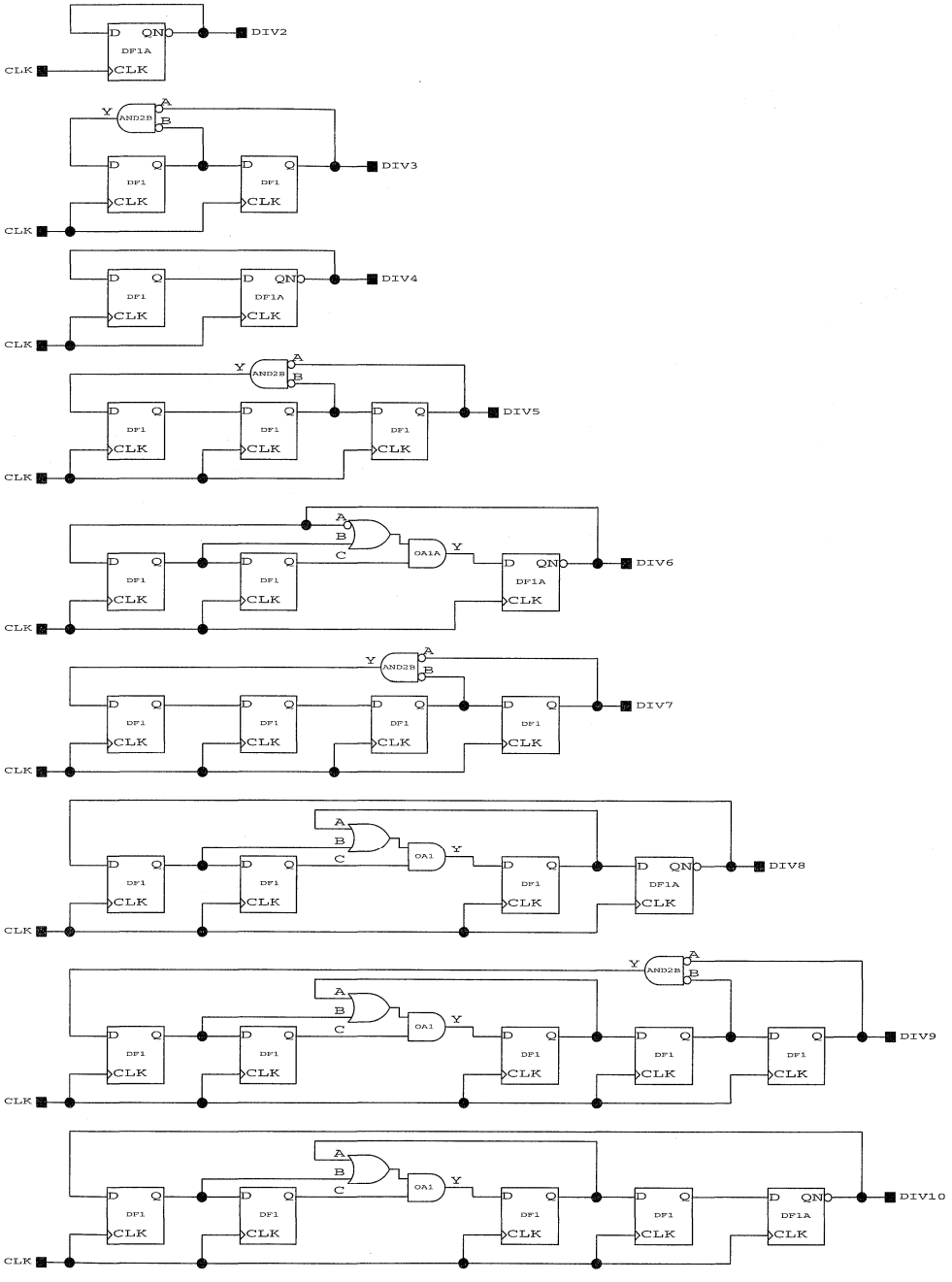


Figure 1 • Actel Implementation for Synchronous Dividers

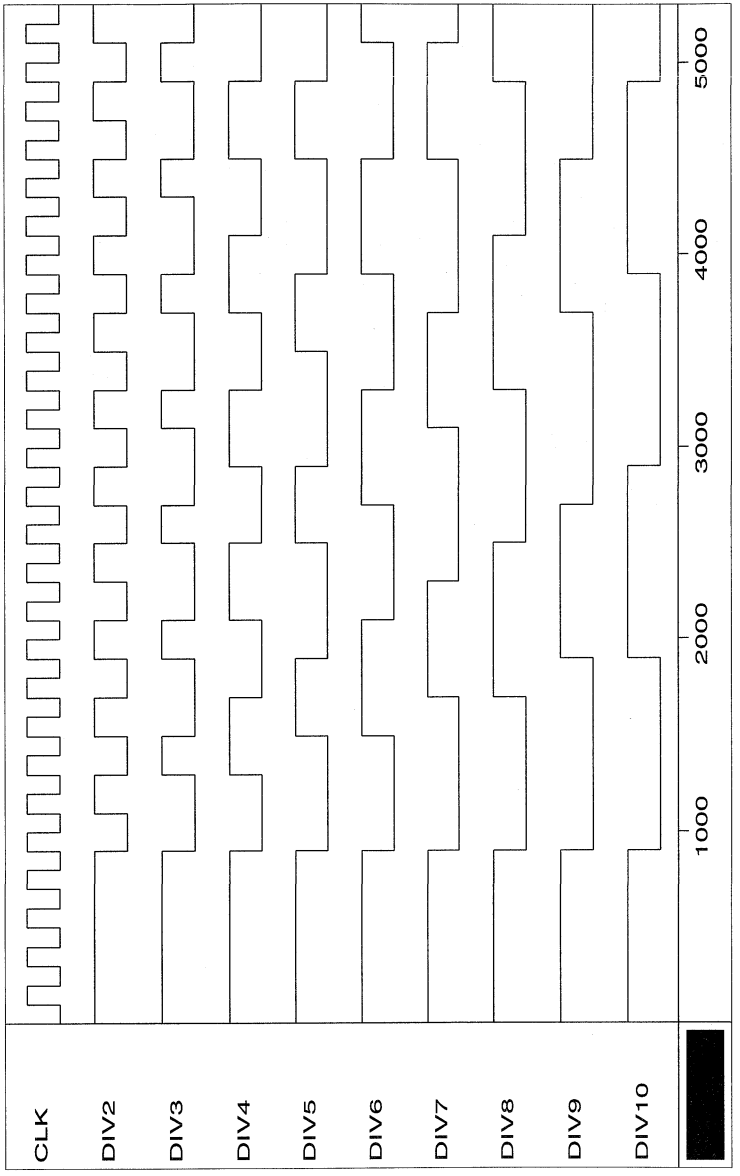


***Figure 2** • Synchronous Divider Waveforms*

A Pulse Stretching Circuit for Actel FPGAs

The ability to stretch a pulse is often useful in applications requiring specific timing. For example, synchronizing external short events to a processor may require pulse stretching to ensure signal recognition. One method to accomplish this function is to use a "one-shot" RC delay network. However, this technique yields wide variations in pulse accuracy due to its dependence on resistor and capacitor tolerances and temperature sensitivity. Another method is to put a large number of inverters in series to create a delay. This technique is most commonly used to meet the setup or hold time requirements of a critical circuit. This procedure is also not recommended due to the 30 percent variation in device processing between best- and worst-case specifications.

For reliable operation, a digital synchronous pulse stretching design is ideally suited for accurate timing and close tracking with the system clock. The block diagram in Figure 1 shows the circuit inputs and outputs. A downcounter provides a variable output pulse width (OUT) controlled by the binary preset inputs (D0, D1, D2, D3). The output is triggered by an input pulse (IN). The reset signal (RST) initializes the circuit.

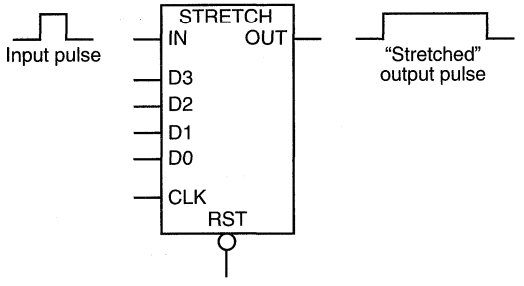


Figure 1 • Pulse Stretcher Block Diagram

The pulse stretcher circuit is described in Boolean format using PALASM2 syntax as shown in Figure 2. There are four register equations in the design. The left hand side defines the logic preceding the D input of a flip-flop (:= indicates a register). Each of these equations has two components. The first component defines loading of the counter (preset inputs D0 through D3 ANDed with IN). The second component of these equations specifies the decrement logic for each bit when there is no input pulse present and the count is not zero (that is, ANDed with the inversion of IN and the inversion of ZERO). The zero flag equation is merely the combinatorial decoding of all registers equal to zero. The output equation is asserted whenever the count is not zero or the input pulse is present.

```
CHIP stretch act
clk rst in d0 d1 d2 d3 out
EQUATIONS
q0 := d0*in + (/q0*/zero)*/in
q1 :- d1*in + (/q1*/q0 + q1*q0)*/zero*/in
q2 := d2*in + (/q2*/q1*/q0 + q2*q1 + q2*q0)*/zero*/in
q3 := d3*in + (/q3*/q2*/q1*/q0 + q3*q2 + q3*q1 +
q3*q0)*/zero*/in
zero = (/q3*/q2*/q1*/q0)
out = /zero + in
q0.RSTF = /rst
q1.RSTF = /rst
q2.RSTF = /rst
q3.RSTF = /rst
```

Figure 2 • Pulse Stretcher Boolean Description

The Boolean description of the circuit was optimized for an ACT™ 2 device. The schematic in Figure 3 was generated from the output netlist for a visual representation of the actual micro implementation. Note the use of the DFMB flip-flops, which require a single sequential module in ACT 2 (or ACT 3) and take advantage of a built-in multiplexer (inputs A, B, and S) to reduce logic. The complete circuit is shown implemented in 14 logic modules (the AX1B macros require two modules).

The output pulse is up to one clock cycle longer than preset inputs because it is ORed with the IN signal. If the OR gate is removed from the circuit, the output pulse will have a width exactly equal to n clock periods where n is the value of the data inputs. The input signal may only be active for one clock edge. If it is active for a longer period, the output will remain active for this duration in addition to the selected value. To have the output reflect only the selected delay if the input is longer than one clock period, change the output equation as follows: OUT = /ZERO * /IN. The timing diagram in Figure 4 illustrates the output pulse with a selected "stretch" value of four.

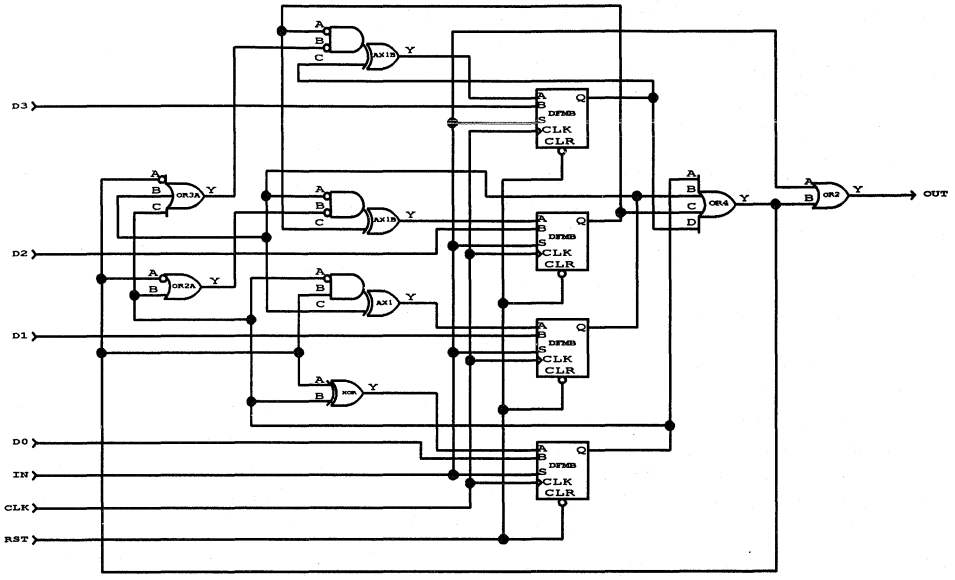


Figure 3 • Pulse Stretcher Schematic

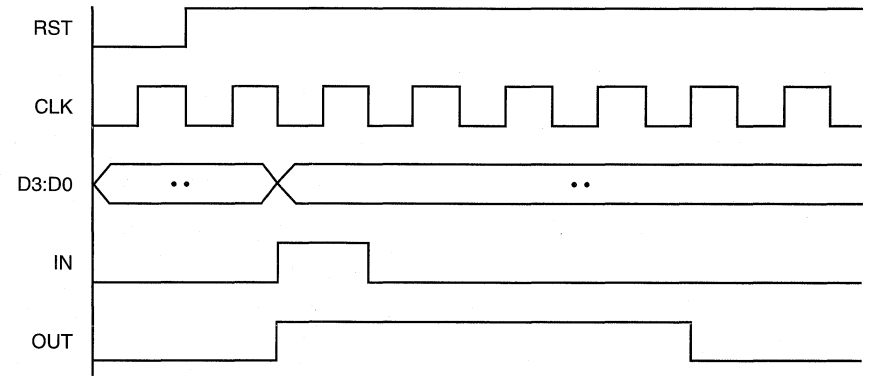


Figure 4 • Timing Diagram for Pulse Stretcher

Real World Applications for FPGAs — An Overview

Introduction

Field Programmable Gate Arrays (FPGAs) have gone mainstream. No longer are applications limited to performance and cost insensitive designs. The newest generation of FPGAs have hit performance and cost goals which allow a much wider spectrum of applications support. This application note will provide an overview of these types of applications and project future performance and cost for FPGA devices.

FPGA Applications

Traditionally, FPGAs have been used in applications with time to market pressures. Because of the ability of FPGAs to be changed quickly, designs can be easily modified to hit aggressive market windows. These markets are usually fast growing markets where being first is important in establishing market share. Communications and networking markets, for example, have traditionally been large consumers of FPGAs due to the fast growth and need to establish standards in these types of markets.

New applications in networking and communications require higher performance and higher capacity than previous generations of designs. ATM and 100 Mbit Ethernet, for example, are very speed and capacity hungry. In ATM applications, wide data paths need to operate at over 100 MHz for simple multiplexing and buffering operations. Frame counting and timing functions are also required to align data packets at similar rates. High speed CRC operations also need to be implemented and must run at the frame rates, usually in excess of 64 MHz.

In 100 Mbit Ethernet serial data transfer rates run at 125 Mhz. Internal state machines need to run at least at 25 Mhz and may need to run at twice the required rate (50 MHz). The hardware required to implement these types of designs can easily consume 10 K gate array gates.

Computer oriented designs are also increasingly gate count and performance hungry. The most recent generation of RISC processors run at speeds of 64 to 100 Mhz. FPGAs can be used in common functions like DMA control, peripheral interface and as hardware accelerators for specialized algorithms. A growing set of applications in the computer area use FPGAs to add customized options to standard processor boards to better serve specific customer applications. The quick time to market capabilities of FPGAs make it possible to address these applications quickly and generate additional high value business.

Another common class of applications are in the peripherals markets. Everything from laser printers, to PCMCIA Ethernet adapters can use FPGAs in prototypes and production. The performance requirements of peripheral applications are usually less stringent than processor applications, but cost is usually more important. As FPGA prices have fallen, more high volume applications have become available. In previous years FPGAs could not have been used in cost sensitive applications like PCMCIA cards, but the most recent price reductions have allowed these types of applications to be served by FPGAs.

Graphics applications have made effective use of FPGAs as hardware accelerators for the *inner loops* of complex graphics algorithms. This use frees up valuable processing cycles from the main processor and allows slower processors and memories to be used, or achieves performance levels unreachable by processor only based solutions.

Perhaps the single largest class of FPGA applications could be described as memory control and interface functions. Just as simple PLDs were used very often for address decode functions, FPGAs seem to be used in many cases to control the flow of data between memory and a processor or processor bus. DMA controllers, cache controllers, FIFO controllers, and interfaces between busses and peripheral memory all fall into this general category. The high capacity, high performance and combination of datapath and control functions required of these applications make FPGAs a natural device choice.

FPGA Cost

Since the introduction of FPGAs in 1985 the cost per gate has dropped by more than a factor of 10. The A1020, a 2,000 gate (gate array gates) FPGA cost $100 in 1985 and now cost less than $10. The A1280 cost $250 at introduction in 1991 and now costs less than $80. These dramatic cost changes have also expanded the volume level at which FPGAs can cost effectively be used.

A common method of comparing gate array and FPGA costs are to compute the break-even volume for the FPGA. The break-even volume is the volume at which the FPGA total cost is the same as the gate array total cost. The total cost of the FPGA is simply the number of units multiplied by the unit cost. For gate arrays the total cost is more complicated.

Gate array costs involve both unit costs and NRE costs associated with the fabrication of the devices. Thus the total cost for gate arrays includes the one time charges as well as the cost per unit multiplied by the number of units. The break-even equation is developed by setting the total costs equal and solving for the volume.

$$NRE + Costga * Volume = Costfpga * Volume$$

$$Volume = NRE / (Costfpga - Costga)$$

Thus the break-even volume is simply the NRE charge divided by the difference in unit cost between the FPGA and the gate array. As the FPGA gets closer to gate array pricing, the break-even volume goes up dramatically.

For example, with an NRE of $24,000, a gate array price of $10 and an FPGA price of $20 the break-even volume is 2,400 units. When FPGA prices are $8 and gate arrays are $4 the break-even point is 6,000 units.

The above calculation is very simplistic and does not include the advantages the FPGAs offer in terms of lower inventory, faster time to market, lower engineering costs, reduced risk from lower than expected volume ramp rates, reduced risk associated with design changes after production builds, and more rapidly falling prices. Once all these factors have been considered, the real NRE should be increased by a factor of 2 or 3, effectively increasing the break-even point by 2 or 3 also.

FPGA Performance

Current FPGA devices offer system performance of over 100 MHz for simple datapath applications. Serial data paths with the associated code conversion (such as that required for 100 MHz Ethernet), and wide bus packet switches with framing and timing recovery (for ATM) are popular high performance examples. More complex systems for RISC processors can run at the 64 MHz to 75 MHz rate. Familiar functions like 16-bit counters run at over 80MHz and 16-bit adders run at over 50 MHz. Clock-to-output delays have been reduced to below 7.5 ns (Pad-to-Pad) and internal clock rates are over 150 Mhz.

Future Trends in FPGA Cost and Performance

Current FPGA families address a wide and growing range of applications. In the future, device cost and performance will continue to improve dramatically. New applications will open and a larger share of the gate array market will be addressed by FPGAs over time.

Over the next five years costs will continue to improve. An 8 K gate FPGA can be expected to cost less than $16. At this price most applications can use FPGAs at 8 K and below even in high volume production. The major factors that will allow these cost reductions are primarily reductions in die size and packaging costs. Die size reductions will come about through the normal lithographic process shrinks, additional interconnect layers and architectural enhancements. One of the major architectural advances coming will be in antifuse based devices. Because antifuses can be placed between metal layers directly and require no active silicon area, it is possible to put the routing above the logic modules and migrate from a channeled architecture to a sea-of-modules architecture, similar to the migration of gate arrays from a channeled architecture to a channelless architecture. This migration will bring die sizes down for antifuse based FPGAs in addition to the normal lithographic shrinks and additional interconnect levels.

Over the same period performance can be expected to improve by 3X. This will push internal clock rates to 500 MHz, 16-bit counters to 240 MHz, 16-bit adders to 150 MHz, and clock to output delays of 2 to 3ns (Pad-to-Pad). These performance levels will allow FPGAs to replace even the fastest PLDs, address even more of the gate array market, and keep up with the fastest processors.

Conclusions

FPGAs are mainstream logic devices. They solve a wide range of application problems currently and will continue to address new applications as costs are reduced and performance is increased.

Application Notes—Using Actel Tools

Section 10: Application Notes—Using Actel Tools

Actel Literature Cross Reference List

This section contains information about how to use Actel tools within specific environments. To obtain more detailed information regarding software installation, CAE setup, authorization, general usage, limitations, and workarounds please review any of the documents listed below. All documents pertaining to the system purchased are included with your system. However, they may also be ordered by contacting Actel Customer Service (408) 739-1010.

Getting Started Manuals

These documents contain general information on choosing a library for the creation of your design, instructions for installing and setting-up the Designer Series software and environment, instructions for generating a netlist for your design, and back-annotating actual post route delays. Each getting started is CAE and Platform specific

Designer Series for Windows, PC Environment, Viewlogic	5029026-1
Designer Series for Windows, PC Environment, OrCAD	5029029-0
Designer Series for the X Window System, Sun and HP700, Mentor Graphics	5029068-0
Designer Series for the X Window System, Sun and HP700, Viewlogic	5029069-1
Designer Series for the HP 700, Cadence Composer Verilog	5029054-0
Designer Series for the X Window System, SUN, Cadence Composer Verilog	5029062-0
Designer Series for the X Window System Cadence Concept RapidSIM Verilog	5029063-0

User's Guides

These documents describe the Actel user-interface and design flow for the Actel software. They include details on each program function from selecting the package to programming the chip.

Designer Series for the X Window System	5029008-0
Designer Series for Windows	5029028-0
APS2 FPGA Programming System, PC Environment	5029031-2
APS2 FPGA Programming System, Sun and HP700 Environment	5029070-0
ACTmap FPGA Fitter, PC Environment	5029037-1
ACTmap FPGA Fitter, Unix Environment	5029049-1
ACTgen Macro Builder, PC Environment	5029051-1
ACTgen Macro Builder, Unix Environment	5029064-1
Actel/Synopsys Library, Unix Environment	5029067-0
Actionprobe Diagnostic Tools	5029021-1

Release Notes

These brief documents contain the latest information on the release that wasn't available when the Getting Started or User's Guides went to print.

ALS 2.3.2s Release notes	5139019-0
ALS 2.3.1 and 2.3 Release Notes	5139011-1
Mentor Graphics, ALS Version 2.3.1	5139052-0
Actel Synopsys 3.1 Library	5130951-0
ALS 2.2S2, PC	3139042-1
ALS 2.2.1 Cadence Concept RapidSIM	3139045-0
Verilog Libraries for Sun Workstation	5139044-0
ProSeries Version 5.1.1a Limitations	5139017-0

Miscellaneous

Designer Series for the SUN, Network License, Installation & Setup	5029034-0
Designer Series for the HP 700, Network License, Installation & Setup	5029036-0
Installation Notes, Sun & HP700, Version 2.3.1 & 2.3	5139018-0
Installation Notes, PC Environment, Version 2.3.1 & 2.3	5339024-0

Other technical information such as Application Notes, Errata's, Device Information and more can be obtained via Actel's 24-hr fax back service, Action Facts (800) 262-1062.

Actel EDIF Reader and Writer

The diversity of Computer Aided Engineering (CAE) systems and tools without common standards has created many difficulties in transferring data among dissimilar design systems. The search for a solution to these problems led to the development of the Electronic Design Interchange Format (EDIF), a standard interchange format for electronic design data. The benefits of adopting EDIF are wide-ranging. Among its many benefits, EDIF enables a wide choice of ASIC vendors by maintaining compatibility with a variety of devices and CAE tools. The Actel system offers EDIF read and write capabilities to ensure compatibility with EDIF-based design tools.

Actel EDIF Reader

The Actel edn2adl EDIF reader translates EDIF 2 0 0 netlists into Actel ADL netlists. The EDIF reader also creates an External Name Map (ENM) file that contains all original names renamed during translation from the original schematic. Properties in an EDIF source may be selectively imported into an ADL file. The usage statement for the EDIF reader is shown below. The various options in this program allow for different traversal algorithms, power and ground representations, and use of a set of preferences from a configuration file.

As shown in Figure 1, you specify the name of the design as the last argument following any of the options. These options control the treatment and characteristics of the input and output files. You may not use most of these options, except FAM and EDNINflavor, for the Actel design. The minimum usage of the edn2adl program is as follows:

edn2adl fam:<device_family> <design_name>

where <device_family> is ACT 1, ACT 2, or ACT 3 and <design_name> is the name of the design.

What follows are detailed descriptions of all the options of the Actel EDIF reader.

FAM : {ACT1 ACT2 ACT3}

This switch decides the type of target Actel library — ACT 1, ACT 2, or ACT 3 — for your design.

EDNIN : <EdifFile>

By default, the *edn2adl* program reads *<design>.edn* for the input EDIF file. You use this switch to specify the name of your EDIF file if its name is different or it does not reside in your current project directory.

```
edn2adl[ FAM : {ACT1 ACT2 ACT3} ]
[ EDNIN : <EdifFile> ]
[ EDNINflavor : {generic wv mentor or} ]
[ ADL : <AdlFile> ]
[ AAL : <AalFile> ]
[ ENM : <EnmFile> ]
[ EDNINcfg : <ConfigFile> ]
[ LIBoverRide : T ]
[ NObadOrigName : T ]
[ USEedifName : T ]
[ GNDnetname : <GlobalGroundNetName>  ]
[ VCCnetname : <GlobalPowerNetName> ]
[ GNDnetProp : <GroundNetProperty> [ : <value> ] ]
[ VCCnetProp : <PowerNetProperty> [ : <value> ] ]
[ EDNINprops : <prop1> [ + <prop2> ... ] ]
[ BUSformat : %s<%d> ]
[ WARNnoContents : T ]
[ Noinfo : T ]
<Design_name>
```

Figure 1 • Actel EDIF Reader Program Syntax

EDNINflavor : {generic wv mentor or}

This switch selects one of a few known flavors (for example, wv for Viewlogic) by automatically choosing the values of various switches of the *edn2adl* program. The default is generic.

ADL : <AdlFile>

By default, the EDIF reader creates *<design>.adl* as the output Actel netlist file. You use this switch to create an ADL file with another name.

AAL : <AalFile>

By default, the *edn2adl* program creates the <design>.aal file when you use the -edntoaal switch (not shown in usage statement of the EDIF reader). Use the AAL:<AalFile> switch to create an AAL file with another name. Note that you have to use the -edntoaal switch along with the aal switch. For example:

edn2adl -edntoaal fam:act2 aal:paige.opt paige

The above command creates an AAL file called *paige.opt* instead of *paige.aal* from the EDIF file using the ACT 2 library for the design named paige.

ENM : <EnmFile>

The EDIF reader can choose the names of netlist objects that are named in the EDIF file. The ENM file records a map of the original names of such objects. These objects have the ENM property with the original names as their values. By default, the *edn2adl* program creates *<design>.enm* as the map file. Use this switch to create a map file with another name.

EDNINcfg : <ConfigFile>

The EDIF reader uses a configuration file from the $ALSDIR/data/edncfg directory containing a collection of edn2adl switches, depending on the EDNINflavor option. Use this switch if you want to use your own EDIF configuration file.

LIBoverRide : T

If, in the EDIF file, you have a cell (macro) that has the same name as one in the Actel library file (*adl06.lib*), the *edn2adl* program, by default, uses the contents of the EDIF cell. In this case, the EDIF reader creates the Actel netlist file with following type of warning:

Contents of cell "X" hides the definition in library
$ALSDIR/data/a1000/adl06.lib

where X is the name of a cell.

You set this switch to T (TRUE) only when you want the EDIF reader to ignore the contents of the cell in the EDIF file and use the contents from the Actel library file. In this case, the *edn2adl* program issues the following type of warning:

Overriding contents of cell "X" from the library
$ALSDIR/data/a1000/adl06.lib

NObadOrigName : T

If you don't specify this switch, the *edn2adl* program uses the original names of the netlist objects unconditionally. You set this switch to T to ensure that the names of the netlist objects imported are legal.

USEedifName : T

You set this switch to T to use the names of netlist objects that are renamed in the EDIF file rather than their original names.

GNDnetname : <GlobalGroundNetName>
VCCnetname : <GlobalPowerNetName>

These switches specify the names of global ground and power nets. For example in the case of Viewlogic, you use

GNDnetName:GND

VCCnetName:VDD

When a net with these names is encountered, both GLOBAL and POWER properties will be attached. The net names used to match the given values are controlled by the NObadOrigName and USEedifName switches.

GNDnetProp : <GroundNetProperty>
[: <value>]
VCCnetProp : <PowerNetProperty>
[: <value>]

These switches specify the name of a property along with a value to recognize ground and power nets. The value should follow the name after a ":" or "=". It can be empty if the property is Boolean. For example, if you use

GNDnetProp:INIT=0SF

VCCnetProp:INIT=1SF

any net containing

(property INIT (string "0SF") ...)

is converted to a ground net, and any net containing

(property INIT (string "1SF") ...)

is converted to a power net.

The property names used for matching are controlled by the NObadOrigName and USEedifName switches.

EDNINprops : <prop1> [+<prop2>...]

The EDIF reader imports any property on netlist objects if the name of the property has the ALS prefix. Use this switch to import all properties that do not have the ALS prefix in their names. Use the "+" separator in this switch to list more than one property.

BUSformat : %s<%d>

The value of this switch determines the formatting style of array members. The default value is %s<%d>. This results in array member A 2 being translated to A<2>. If you set this switch to ^%s[^%d], the array member A 2 is translated to A[2]. Note that the "^" character is needed to escape the "%" character.

WARNnoContents : T

If the EDIF file contains any cell that does not have underlying contents, the EDIF reader issues following error messages:

Cell "X" has no contents and is not defined in library

where X is a name of a cell.

To turn the above error message into a warning, set the WARNnoContents switch to T. This enables you to get an output ADL file with no description for the cell that does not have underlying contents.

NOinfo : T

When you run the *edn2adl* program, it displays the name of cells in the EDIF file. Set this switch to T if you don't want to see these names.

Actel EDIF Writer

The Actel adl2edn EDIF writer translates Actel ADL netlists into EDIF 2 0 0 netlists. The usage statement of the EDIF writer is shown in Figure 2. The various options for this program allow for different traversal algorithms, power and ground representations, and use of a set of preferences from a configuration file.

As shown in Figure 2, you specify the name of the design as the last argument following any of the options. These options control the treatment and characteristics of the input and output files. What follows are detailed descriptions of all the options of the Actel EDIF Writer.

FLAT : T

By default, the *adl2edn* program creates a hierarchical EDIF. You set this switch to T if you want to flatten the hierarchy.

POWERstyle : {cell net port portVerbose}

By default, the EDIF writer uses Cell style in the output EDIF file to represent power. You can force the *adl2edn* program to use the other styles such as net, port, and portVerbose.

EDNOUTadl : <adl_file1> { + <adl_file2> ...}

By default, the *adl2edn* program passes the properties only from an ADL file to an EDIF file. Use this switch if you also want to pass properties from the supplementary files such as CRT, IPF, or STF. Note that you also have to list these properties by using the EDNOUTprops switch. For example:

als -adl2edn EDNOUTadl:paige.adl+paige.crt+paige.ipf
EDNOUTprops:CRT+PIN+FIX paige

Usage:

```
als -adl2edn[ FLAT : T ]
[ POWERstyle : {cell net port portVerbose} ]
[ EDNOUTadl : <adl_file1> { + <adl_file2> ...} ]
[ EDNOUT : <edif_file> ]
[ EDNOUTflavor : {generic wv synps} ]
[ EDNOUTcfg : <config_file> ]
[ EDNOUTprops : <prop1> { + <prop2> ....} ]
[ REPLACEname : T  ]
[ GNDnetname : <newName>  ]
[ VCCnetname : <newName> ]
[ GNDnetProps : <name1> = <val1> { + <name2> = <val2> ...} ]
[ VCCnetProps : <name1> = <val1> { + <name2> = <val2> ...} ]
[ EDIFOUTcriticality : T ]
[ NOportDirection : T ]
[ONEcaseName : {U L} ]
<Design_name>
```

Figure 2 • Actel EDIF Writer Program Syntax

This command will pass properties not only from the ADL file but also from CRT and IPF files to the output EDIF file for the design named *paige*.

EDNOUT : <edif_file>

By default, the EDIF writer uses the name *<design_name>.edn* for the output EDIF file. You use this switch if you want to create the EDIF file with another name.

EDNOUTflavor : {generic wv synps}

This switch selects one of a few known flavors — for example, wv for Viewlogic — by automatically choosing the values of various switches of the *adl2edn* program. Note that you don't need to choose the synps flavor for Synopsys, because Synopsys can read the default generic flavor.

EDNOUTcfg : <config_file>

The EDIF writer uses the default configuration file from the $ALSDIR/data/edncfg directory, depending on the EDNOUTflavor option. Use this switch if you want to create your own EDIF configuration file containing a collection of various switches of the *adl2edn* program and to override the generic flavor.

EDNOUTprops : <prop1> [+<prop2>...]

You use this switch to list the names of the properties that are exported from the supplementary files given in the EDNOUTadl switch.

REPLACEname : T

You set this switch to T if you want to replace the names of netlist objects in the rename clauses with the legal EDIF names.

GNDnetname : <newName>
VCCnetname : <newName>

These switches, if specsified, replace the names of power and ground nets in the output EDIF file. For example:

GNDnetName:GND

VCCnetName:VDD

In the above case, all power and ground nets in the ADL file are assigned GND and VDD names, respectively, for the EDIF file.

GNDnetProps :
<name1>=<val1>{+<name2>=<val2>...}]
VCCnetProps :
<name1>=<val1>{+<name2>=<val2>...}]

Each of these variables, if specified, emits a property along with a value to recognize ground and power nets. The value should follow the name after a ":" or "=". It can be empty if the property is Boolean. For example:

GNDnetProp:INIT=0SF

VCCnetProp:INIT=1SF

In this case, all ground and power nets in the ADL file are assigned (property INIT(string "0SF")) and (property INIT(string "1SF")), respectively, in the EDIF file.

EDIFOUTcriticality : T

If you set this switch to T, nets with the CRT property in the ADL file are translated to the EDIF criticality construct by using Table 1.

Table 1 • Criticality Mapping

CRT Value	EDIF Criticality
U or L	–10
M or H	10
F	20

NOportDirection : T

You set this switch to T if you want to suppress the direction of the ports in the EDIF file. For example, Viewlogic will not instantiate IN and OUT components when you suppress the direction of ports.

ONEcaseName : {U L}

By default, the *adl2edn* program passes the names of netlist objects in their original cases in the output EDIF file. Use this switch to translate to uppercase or lowercase names. For example, since Viewlogic prefers only uppercase names in the EDIF file, use ONEcaseName:U.

Mentor Graphics V7 to V8.2 Design Conversion

Introduction

Converting Mentor Graphics® designs is a topic of increasing concern as more V7 ASIC customers move to V8.2. To make the conversion, one uses the V8.2 conversion tools to migrate V7 designs into the V8.2 world.

What follows is a summary of the conversion of an Actel version 2.11 (or earlier) design compatible with Mentor Graphics V7 to an Actel version 2.2 design compatible with Mentor Graphics V8.2.

However, be aware that Actel's support for Mentor Graphic's V7 was for the DN3000, DN3500, DN4000, and DN4500 platforms, and that Actel currently supports only HP700 and Sun 4 (or fully compatible) Workstations for Mentor Graphics V8.2.

The conversion program from Mentor Graphics is compatible with DN workstations only. It reads the ALS 2.2 libraries and map files, which are first installed on the Sun workstation and copied to the DN workstation. After the conversion is completed on the DN workstation, the design must be transferred back to the Sun environment to recompile with ALS 2.2.

Follow these steps to complete the process:

1. Prepare the design for conversion.
2. Remove the "dollar sign" from the V7 design.
3. Convert the design.
4. Update symbols and nets.

Preparation

Use the following procedure to prepare each design for conversion to Mentor Graphics V8:

1. Install ALS 2.2 software on the Sun Workstation and add $ALSDIR, the Actel environment variable, to the path. More information on ALSDIR is found in the *CAE Guide: Sun/Mentor Graphics for ALS 2.2*, Chapter 3.
2. Transfer the Actel Libraries and map files from the Sun to DN workstation and set $ALSDIR. The library files are in the $ALSDIR/lib/me/parts directory, and the map files are in the $ALSDIR/des_arch directory.
3. Install Mentor Graphics V8 software on the DN and Sun workstations. Make sure that gen_lib is included in the software.
4. On the DN workstation, set all paths and environment variables for V7 and V8. Refer to the *CAE Guide: Sun/Mentor for ALS 2.2*, Chapter 3.
5. On the HP700 or Sun workstation, set all paths and environment variables to V8. Refer to the *CAE Guide: Sun/Mentor Graphics for ALS 2.2*, Chapter 3.
6. Create the MGC_LOC_MAP file (Mentor Graphics Location Map file), which should include the correct soft and hard links to the working directories for both ALS 2.2 and Mentor Graphics V8. A sample copy of the MGC_LOC_MAP file is included in the *CAE Guide: Sun/Mentor Graphics for ALS 2.2*, Chapter 3.
7. Verify that the V7 design is free of errors and that every block is expanded. Then delete extra versions of the design using "dlv ... 1" unless otherwise desired.
8. Verify that $MGC_WD is set to 'pwd' (project working directory).
9. Check to see if the V7 design has references to the old Actel library.

Remove Dollar Sign

The "remove_dollar" program from Mentor Graphics removes dollar signs ($) from the Mentor V7 design files. The dollar sign ($) has a special meaning for Location Map Variables in Mentor Graphics V8. The dollar sign references must be removed before converting from V7 to V8. the "remove_dollar -help" command gives more information on this, as does the Mentor Graphics document, "Transition Guide for V8 Capture Products."

If the "remove_dollar" command is not available on your system, follow these steps:

1. From one level above the design directory (not in the design directory), type the V7 command:

   $MGC_HOME/unrlsd/listref <design_name>.

   References should be pointing to the old ALS library path.
2. *cd* to the design directory and list references by typing: *ls *.ref*
3. Remove files that have the following extension: **.erel_**. These files refer to the flattened design.

4. To remove directories associated with the flattened designs, type: *rm -r *.erel_*\$*.bak*. At this stage, the directory should contain all ALS files in addition to the sheet and symbol files generated with Mentor Graphics V7.
5. Make sure that $MGC_WD is still set to 'pwd', otherwise type: *setenv MGC_WD 'pwd'*
6. To change references from the old Actel library to the new Actel library, type:

   chref <design_name> '/user/als/lib/x' '/user/als/lib/me/parts'

   where x = a1000 for ACT 1 library or x = 1200 for ACT 2 library.
7. To verify whether the design is referencing to the new library, type: *listref <design_name>*

Conversion

After preparing the design and removing dollar signs from the design files, the conversion program can be executed. The conversion program is called cvt_comp. This V8 program is located in $MGC_HOME/bin on the DN workstations only. To run the conversion program, type:

cvt_comp <design_name> -preview -convert -nodelete -nomodel -map <map_path>

Where:

- *preview* checks the conversions needed and produces a list of parts within the design that are to be converted.
- *convert* selects both symbols and schematics to be converted.
- *nodelete* keeps V7 intact during the conversion process until conversion is complete.
- *nomodel* means no behavioral models exist (i.e., VHDL).
- *map <map_path>* specifies an ACT 1, ACT 2, or ACT 3 family map file for referencing new V8 symbols, and *<map_path> =$ALSDIR/des_arch/actx_map_file*.

After a successful conversion, the program issues the following message:

conversion complete, 0 errors.

Now the conversion is complete. During the conversion process, the convert command creates the V8 schematic and symbol files with updated references. If the conversion fails, all files referring to the V8 design must be deleted before running convert a second time.

To verify that all references are updated, type the V8 command:

$MGC_HOME/bin/listref <design_name>

Updating Symbols and Nets (V7.X to V8.X and V8.1 to V8.2)

There is some cleanup necessary of the V8 schematics due to the changes in the basepoints of the symbols from V7 to V8. All Actel symbols must be selected individually or in groups and then updated to properly route the schematic nets. To update the symbols, invoke Design Architect (DA) with the design and from the popup menu, select:

edit > select > area > instance
edit > update > instance

Now the nets are connected to the symbols properly but are not straight. The nets must be moved manually to make them orthogonal. The basepoint is not the same for all parts.

Selecting and Modifying I/O Assignments

Introduction

Effectively selecting I/O assignments is critical to the success of an Actel field programmable gate array (FPGA) design. Although manual I/O assignment is optional in the design flow for the Designer Series software, the importance of obtaining an effective assignment to the overall flow should not be underestimated. The placement and routing of a design may fail if the pin placements are not optimized for the design and ACT family device. Its overall performance is largely determined by the I/O selections as well. Poorly placed designs often require longer routing tracks, which slow the maximum operational frequency.

System designers know that I/O assignments have a great impact on schedules and deadlines as well. The I/O assignments for all ASICs must be chosen before printed circuit board (PCB) layouts can be completed. Aggressive schedules may require that the PCBs are designed and ordered before the electronic circuits have been tested and debugged. Completing the I/O assignments and the PCB layout becomes one of the gating items for on-time system delivery.

The ACT FPGA design flow includes several different methods of selecting I/O placements. The flexibility of the Designer Series enables many design methods to lead to successfully completed designs. One of its benefits is that there is no exact "right" or "wrong" way to use the system. However, there are recommended methods that will increase the probability of getting optimal I/O assignments and fast, routable designs.

I/O Assignment Methods

The Designer Series allows I/O assignments to be chosen after a design has been converted into an Actel netlist and its device type and family have been selected. There are three options available for assigning I/Os at this point. Each offers benefits that may apply to different design methodologies and system requirements. The I/O assignment options and their benefits are:

Automatic Assignment

No manual assignment is made for an I/O signal. The Designer Series automatically selects the best pin location depending on characteristics of the design such as criticality assignments, device type and family, and overall design topology.

Manual, Fixed Pin Assignment

Pin numbers are manually assigned to I/O signals and are fixed in place. The Designer Series will not move I/O assignments that are fixed. Manual, fixed assignments allow the maximum control over the design flow since the I/O assignments remain fixed regardless of design changes.

Manual, Unfixed Pin Assignment

Pin numbers are manually assigned to I/O signals but are not fixed to the I/O location. Since they are not fixed, the I/O assignments are used by the Designer Series as suggested locations—they may be moved to other locations by the software. This method allows some input to the desired I/O configuration but gives the Designer Series the flexibility to optimize the configuration further.

Any combination of these three methods may be used for the I/O signals of a design. Designs may have all pins manually assigned or automatically assigned, or have a percentage of each type. How these methods are used depends on the requirements of the each project.

Suggested Design Flow

Since the three methods of determining I/O assignments for an ACT family FPGA may be used in any combination, many methods are commonly used to create quality pin placements. The best method for a design depends on many factors such as performance specifications, design changes, and scheduling constraints.

Generally, the most effective method for pin assignment is to use automatic assignment for 100% of the I/O signals. The Designer Series has been designed to optimize the placement and routing especially if it is given the flexibility to automatically assign all of the I/O placements. The Designer Series is able to select, evaluate, and optimize many different I/O configurations specifically to the device architecture. If as few as 10% of the pin assignments are inefficiently assigned manually, the quality of the placement and routing may be compromised. Manual assignments should be minimized as much as possible. To obtain a 100% automatic assignment, skip the Pin Edit program in the Designer Series and go directly to the Layout programs. The final pin assignments are stored in the ASCII file, <design name>.pin, located in the design directory.

If some of the I/Os require manual selection, use unfixed assignments as much as possible. This allows the Designer Series the flexibility to modify some of the assignments as needed. In general, maintain the percentage of manual assignments, fixed and unfixed, to as low a level as possible.

The Most Effective Fast Design Flow

I/O assignment does not have to be a gating item to the overall system schedule. A near-optimal automatic assignment may be obtained from the Designer Series before the circuit design is completely tested and debugged. As long as the number and function of the I/O signals are finalized, the Designer Series can be used to assign all the pins before the details of the FPGA design are complete. Use the following method to get a quick start on the PCB layout and speed schedules for fast turnarounds:

1. Begin by entering the design as completely as possible, ignoring small details that may need to be modified later. It is important to include all the major functions that will be in the FPGA circuit. The objective is to obtain a netlist that approximates the final design topology. The circuit does not need to be functionally correct at this point but must have the same major functional blocks (adders, counters, and so on) and approximately the same number of logic modules as the final design.
2. Determine the exact number of input, output, and bidirectional pins for the FPGA and include them in the netlist. Select the appropriate ACT™ family device for performance level, logic module capacity, and I/O count.
3. Skip Designer's Pin Edit step.
4. Run the Layout programs in the Designer Series. Ignore any warnings from the Validate program that are due to the unfinished state of the design. Run the Layout programs with no clock balancing and incremental placement turned off.

The Layout programs will create a 100% automatic I/O assignment specific to the design topology. This is done before all the functional bugs have been discovered and resolved. Minor functional changes that may be made later will have little effect on the quality and effectiveness of the pin placements. The PCB layout can be completed with the confidence that these pin assignments will require no modifications in the future.

After the design is complete, use the Pin Edit program to read the automatic I/O assignments and fix them permanently. Once they have been fixed, the Designer Series will always use the pin placements initially created by the Layout programs. The Layout programs will not modify any placements that have been fixed.

This method is of value even if the I/Os will be manually assigned. The results of 100% automatic assignment by the Layout programs can be used as a template for an effective manual I/O assignment. It is easy to use the existing placements as a guide and make small modifications as required.

Manual Assignments

If any or all of the I/O assignments must be determined manually, a broader knowledge of Actel FPGA architecture is required. This databook contains pertinent information regarding different architectures of the ACT families. The structure of the logic modules and I/O modules, routing tracks, antifuses, and other architectural details are covered in detail. With this source of information, it is possible to manually assign I/O signals with the specific details of the device in mind.

During the process of assigning I/Os for the device, keep these guidelines in mind to increase routability and performance:

1. Try to force signal paths, especially large data buses, to flow horizontally across the die since there are more horizontal than vertical routing resources. In most cases, a large data bus requires many interconnections as it traverses the circuit. The greater number of horizontal routing tracks handles these interconnections to reduce routing congestion. The horizontal and vertical orientation of the die is the same as shown in the package pin assignment and mechanical detail drawings in the *Actel FPGA Data Book and Design Guide*.
2. Count the number of levels of logic between I/Os to determine placement. For example, I/Os separated by one gate should be placed closer than I/Os separated by a long shift register. In general, the architecture of ACT devices allows signals to be routed across two vertical rows and over one-third of the columns of the device without using a long routing track. For an A1225 device with 13 rows and 46 columns, two I/O signals should not be placed on opposite sides of the device unless there are at least two horizontal or three vertical levels of logic between them.
3. Use the top and bottom I/O pins for slow or local signals or both. The fact that there are fewer vertical routing resources will not harm the performance of these signals.
4. Use the global clock buffers. Each of these pins is connected to a dedicated, low-slew distribution network optimized for high fanout signals.

5. Refer to the information in the PLI (PLacement Information) and RTI (RouTing Information) files as feedback for I/O assignment iterations. Each of these files indicates potential problem areas of the design that may be due to the I/O assignment. The PLI and RTI files report specific problems with the placement and routing of the device respectively. Determine from this information whether reassigning some I/Os will remove these problems.

Modifying Existing I/O Assignments

Unfortunately, it is not always possible to allow the Designer Series to assign most or all of the I/O placements. In addition, the placements may have to be finalized before optimization by the Designer Series for design routability and performance considerations. In any case, a variety of factors may result in a nonoptimal I/O assignment being created and used for a design. This may cause the initial placement and routing to fail or degrade the performance level unacceptably. If this occurs, there are several actions that can improve the placement, routing, and performance of the design.

Recognizing Poor I/O placements

It is important to recognize when placement, routing, or timing problems of an Actel FPGA design are caused by poor I/O assignments. The Designer Series will indicate potential problem areas in several different ways. For example, the <design_name>.VLD file, created by the validate program, contains several messages and statistics that may indicate I/O placement problems. If more than 33% of the I/O pin assignments are fixed, a warning message is issued during the routability check. The same routability check will fail if a critical path (M or F) must use a long routing track.

The outputs of the place and route programs are the <design_name>.PLI and <design_name>.RTI files respectively. Both files contain information to help understand the cause of possible placement and routing problems. The PLI file lists every net that requires the use of a long track for routing. There are a limited number of long vertical tracks (LVTs) and long horizontal tracks (LHTs) in each device. If an input or output pin is directly connected to a long track, the I/O placement can probably be improved.

Improving Routability and Performance

If the I/O placements for a design are fixed and causing placement, routing, or performance problems, there are several ways to modify the design and its implementation into an Actel FPGA. Try the following steps to increase the routability and performance of the design:

- Use criticality assignments but don't exceed the maximum number of critical nets for the device. Determine the speed critical and uncritical paths in the design and specify them in the criticality file, <design_name>.CRT. The criticality assignments help the place and route software to make critical nets faster and improve the overall performance. Be careful not to exceed the maximum number of critical nets for each device as specified in the *the Designer Series User's Guide*.
- Do not use clock balancing if possible. If there are place and route problems, clock balancing may reduce routability and actually increase delay times. Reduce the clock balancing strength to the lowest level that will satisfy the clock skew requirements of the design.
- Reduce high fanout nets between I/O modules. Routing congestion between I/Os is especially critical to the placement and routing success. For low critical nets especially, use buffers to reduce fanout to a few loads.
- Watch the overall fanout statistics of the design. The validate program reports the average fanout for the design in the VLD file. A high average fanout (greater than 3.5 loads) will usually be harder to place and route. If possible, reduce the overall fanout by buffering high fanout nets. In addition, reduce the use of high fan-in macros to less than 25% of the total logic module count. High fan-in macros such as the DFM8A and CM8 have up to eight inputs per module. Using a large percentage of these macros may cause unsolvable routing congestion problems.
- Be careful of designs with high I/O utilization (greater than 80%) and low logic module utilization (less than 50%). Since most of the I/O modules but a low percentage of the logic modules are used, many long routing tracks may be required. This will deteriorate the routability and performance of the design. Remember that high logic module utilizations (greater than 90% and up to 100%) are common, so increase the utilization percentage accordingly or change to a smaller device.
- Check the PLI file for the list of long routing tracks and buffer them appropriately if they are high fanout nets.

Socketing Option

In a few cases, the previous suggestions may not solve problems caused by a poor I/O placement. A hardware solution may be necessary. There are commercially available IC sockets that re-map the pinout of the device to any chosen configuration. If such a socket is available, the Designer Series may be used to reselect the I/O placements for the design without any constraints. The probability of a successful place and route with higher performance can be greatly increased this way.

Summary

The most effective design methodology for Actel FPGAs includes 100% automatic I/O assignment by the Designer Series. The software is designed to optimize I/O assignments especially if it is given a large degree of freedom from manually fixed placements. If manual assignments are necessary, a broader background of the FPGA architecture is necessary to achieve similar results. If manual selections are necessary, use the suggestions from this application note to select the placements as well as possible. If the placements of the I/Os are fixed and they appear to be hampering the routability and performance of the design, there are still several modifications that may be made that can improve the probability for success.

Critical Path Analysis Using the Timer

Introduction

The critical paths of Actel designs determine the maximum performance of the device in a system. These critical paths must meet system specifications to function properly. A worst-case timing analysis of the critical paths ensures that the device will work reliably in production. Critical paths are design dependent; they can be combinational paths from input pads to output pads, sequential paths from flip-flops to other flip-flops, clock to output pad paths, setup times and hold times of the field programmable gate array (FPGA) relative to other devices on board, or any combination of these paths.

The Timer in the Designer Series software is a static timing analysis tool used to verify device timing. The Timer is used interactively or with command files to automate the timing analysis process. The Timer generates delay reports, including derating factors, for different operating conditions due to variations in temperature, voltage, and device process. Voltage and temperature ranges for commercial, industrial, and military devices are included in the menu selection.

This applications brief describes how to analyze critical path delays of Actel's FPGAs using the Timer in the Designer Series software. It illustrates strategies to determine the propagation delays of combinational and sequential paths. It also explains the concept of net delays and how the Timer reports them. Please refer to the User's Guide for detailed Timer commands.

Net Delays in the Timer

The Timer reports propagation delays between a Startset and Endset of pins in the design. The Startset contains the start pins and the Endset contains the end pins of the paths under investigation. The Timer lists propagation delays from the input pins of a macro in the Startset to the input pins of a macro in the Endset, including the net delays. These net delays are calculated from layout and fanout information. Keep in mind that the Timer reports timing from input pins to input pins so that all net delays are automatically included. Figure 1 shows that delay DEL1 includes the delays of the gate U1 as well as the net delay arriving at gate U2. Also note that the Timer can reference the output (PAD) pin of output buffers to include the propagation delay to the actual package pin. The output pins of output buffers are the only non-input pins that the Timer can reference for timing analysis. DEL3 in Figure 1 shows the propagation delay from the D pin to the pad pin of the output buffer

Combinational Paths

As mentioned earlier, critical paths are design dependent. They can consist strictly of combinational logic. In many cases, the combinational paths are paths from input pads to output pads. The cumulative propagation delays of all elements in the critical paths can be determined using the Timer. The following examples of combinational paths do not involve asynchronous feedback.

Internal Combinational Paths

To analyze this type of critical path, first set up a Startset and an Endset. The Startset contains all input pins that will initiate the changes through the critical paths. The Endset contains all input pins of macros at the end of the combinational paths. Based on the specified Startset and Endset, the Timer will report delays in all paths between these two sets.

10

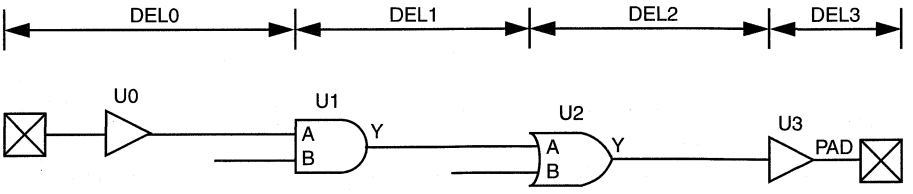


Figure 1 • Net Delays

Figure 2 shows examples of internal combinational paths. In this circuit, A and B pins of gate U0 as well as A and B pins of gate U1 are specified to be start pins. The D pins of U5 and U6 are specified to be end pins. The Timer reports delays in all paths between the start pins and end pins. The delay breakdown of each path and the fanout of each element in the path are also reported.

Input Pad to Output Pad Combinational Paths

Combinational paths from input pads to output pads are special paths where the start pins are input pads and end pins are output pads (see Figure 3). As previously stated, the output pins of output buffers are the only non-input pins that may be specified in the Timer. The Timer references the PAD pins of the input buffers as start pins and the PAD pins of the output buffers as end pins. An example of this type of path is a decoder, where inputs of the decoder come on chip via input pads and outputs of the decoder exit via output pads.

The Timer automatically puts all input pads in a set named Inpad and it also puts all output pads in a set named Outpad. Use Inpad and Outpad as the current working sets to analyze the timing of these paths. Use the breakdown in delay elements that make up the path for delay optimization. The fanout of each element and the type of macro is listed.

Sequential Paths

The design's critical paths can also consist of sequential paths. Sequential paths are paths that include sequential elements such as flip-flops and transparent latches. Sequential paths must be broken into segments for timing analysis. They are input pads to clocked macros, clocked macros to clocked macros, and clock pads to output pads. In most synchronous designs, one of these three segments will determine the maximum operating frequency of the device.

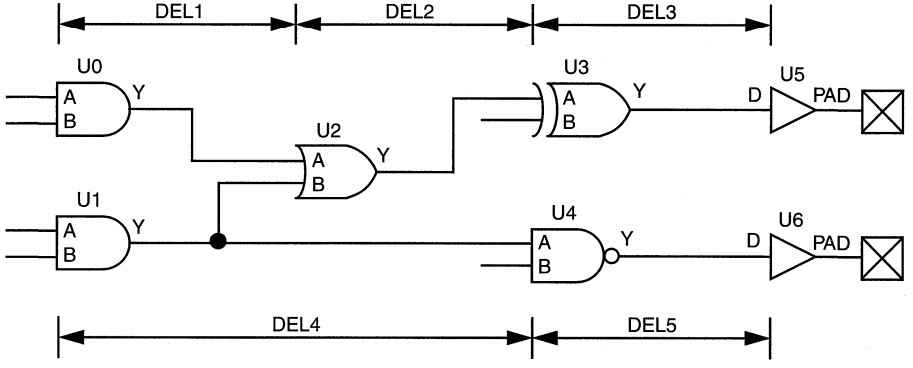


Figure 2 • Internal Combinational Paths

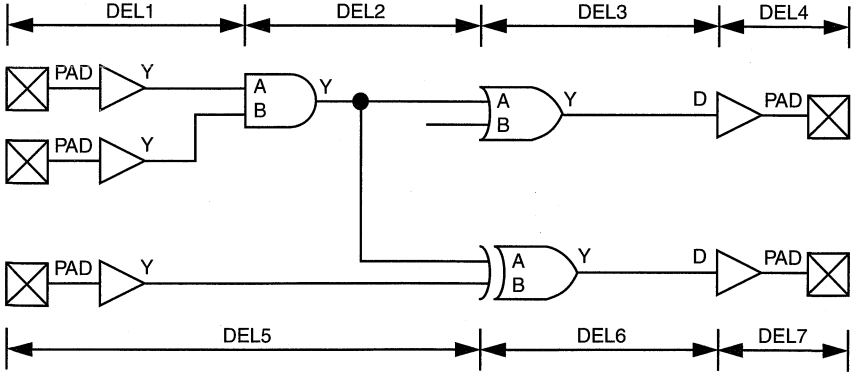


Figure 3 • Pad to Pad Combinational Paths

Input Pads to Clocked Macros Paths

The input pads to clocked macros paths are data signals that come onchip via the Actel FPGA input pads, propagate through the logic, and arrive at the gated inputs of other clocked macros. These data signals have to be stable some period of time before the clock edge arrives at the clocked macros. This period of time is the intrinsic setup time of the clocked macros, which is specified in the datasheet. The Timer presents these data paths, including the setup time of the clocked macros, in the longest to shortest order. Figure 4 shows several typical data paths. As previously stated, the Timer includes all input pads in a set named Inpad and all gated inputs of clock macros (all inputs of flip-flops or transparent latches except clock inputs) are included in a set named Gated. Simply change the current working Startset and Endset to be Inpad and Gated respectively and list the propagation delays for these paths. The clocked macros' setup times are automatically included in these paths.

Clocked Macro to Clocked Macro Paths

The paths between clocked macros and clocked macros are sometimes the design's critical paths (see Figure 5). These paths begin at the clocked inputs (CLK or G) of the clocked macros and terminate at the gated inputs of the clocked macros. In cases like counters or state machines, these paths actually feed from the outputs back to the inputs after propagating through several levels of logic. Similarly, the Timer includes the setup times of the clocked macros so that the maximum frequency can be easily determined.

The Timer facilitates the timing analysis of these paths by saving all clock pins in a set named Clock and saving all gated pins in a set named Gated. In this example, all CLK pins are saved in the Startset Clock and all data and enable pins of sequential-type macros (flip-flops and latches) are saved in the Endset Gated. Use these sets to investigate all paths between clocked macros easily. Again, the clocked macro's setup times are included in the report.

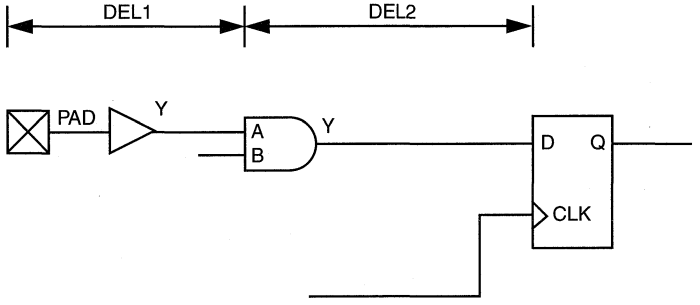


Figure 4 • Input Pads to Clocked Macros Paths

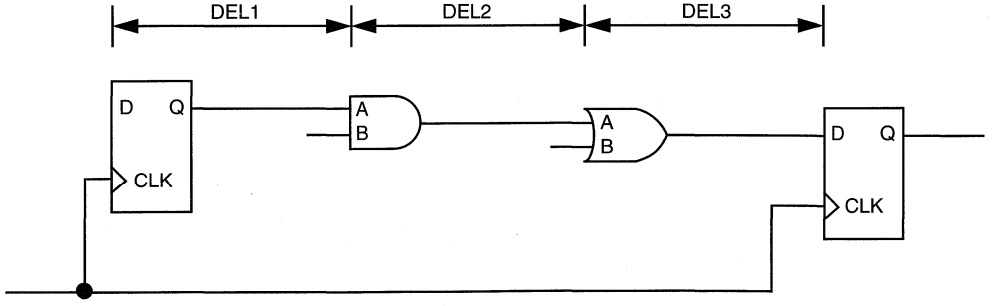


Figure 5 • Clocked Macros to Clocked Macros Paths

Clock Pad to Output Pads Paths

The clock pad to output pads paths are slightly different in the source and destinations. These paths begin at the external clock pin and terminate at the output pads. These paths go through the clock pad, pass through the clock macros, and exit the device via the output pads. In many cases where the output pads are parts of a bus, there will be a small output skew between these output pads. Therefore, use the slowest output pad as the limiting path. To analyze the clock pad to output pads paths, create a Startset containing the clock pad and use the default set Gated as the Endset. Figure 6 shows some paths starting at the clock pad and ending at the output pads. The Gated set is automatically loaded by the Timer as the working Endset. For these paths, simply list all paths between the sets and take the slowest path into consideration. In many pipelined designs, the clock pad to output pads paths turn out to be the longest paths and are the design's critical paths.

Setup and hold times and skew time are also important. Be aware of these characteristics to ensure that the device works reliably in production. The device setup time is the time that data signals must be stable before the clock transition. The device hold time is the time that data signals must be valid until after the clock transition. It is important to understand that the device setup and hold times are not the flip-flop setup and hold times given in the datasheet. The device setup and hold times are design dependent, and they must be characterized to guarantee that the Actel FPGA will work in sync with other devices on board.

Device Setup and Hold Time

The device setup time can be easily determined by comparing the input pads to gated macros data paths with the clock pad delay. The difference in propagation delays between these two paths is the device setup time. It is possible that this delay is a negative number. The setup time is the minimum required time that data have to be stable before the transition of the clock signal; in this way, data can be stable long before the transition of the clock signal. The hold time can be determined similarly by comparing the data paths with the clock pad delay.

Skew Time

Skew time is important in synchronous designs. The skew time is the difference in arrival time of clock signals at the clock inputs of commonly clocked macros. Ideally, the skew time must be as close to 0 ns as possible. A good example is a serial shift register. If the clock skew is such that the clock edge arrives late, wrong data will be shifted in the register. The skew time is especially important in designs using a normal input pad for clock purposes. A normal input pad is not buffered internally like a clock pad. The input pad used for a clock signal may introduce additional delays in the paths and result in a larger skew time. Using the input pad for clock signals may result in a skew time of more than 2.5 ns, which is the maximum skew time of the clock network. This skew time must be added to the critical path timing to ensure that the design will work reliably.

The skew time (Figure 7) may be determined by comparing the longest clock path to the shortest clock path. The difference in delay of these two paths is the skew time. List all paths between the clock pad and all clock inputs of clocked macros and calculate the difference between the delays to the clock inputs of adjacent clocked macros. This skew time should be added to the overall design's critical path for the result to be valid.

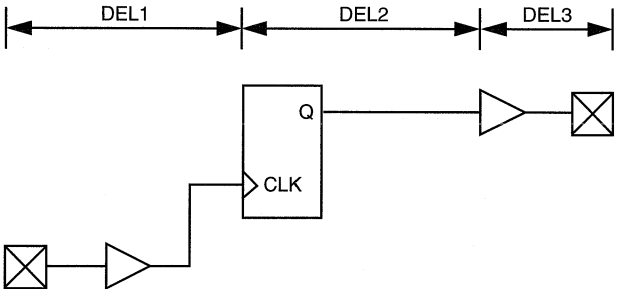


Figure 6 • Clock Pad to Output Pads Paths

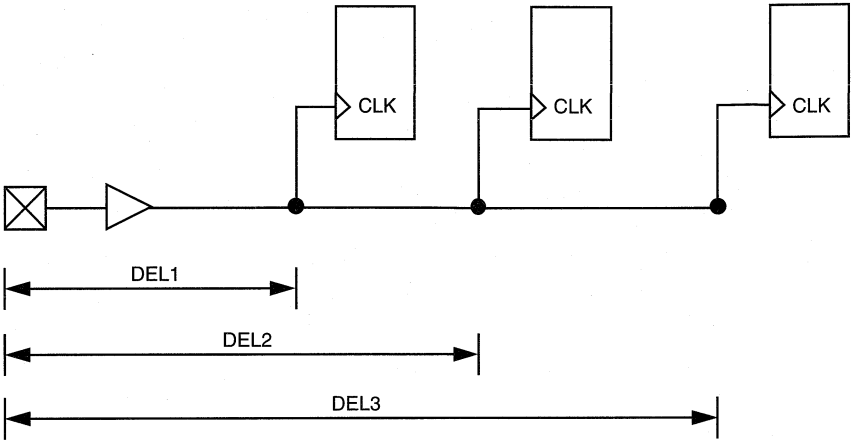


Figure 7 • Skew Time

Multichip Simulation Using Powerview Viewsim

System designs are typically divided into functional modules that are implemented by several Actel devices. To check the functionality of the system, it is important to simulate all Actel devices together. The Actel Designer Series System is capable of multichip simulation with many common simulators.

This Application Note describes an example of multichip post-layout simulation using Powerview Viewsim 5.2 (or above) with the Designer Series system on a Sun or HP700 workstation. Notice that you can only simulate multiple Actel devices of the same family. You can't simulate mixed family Actel devices together. This is because you can't refer to multiple Actel family libraries using one viewdraw.ini file as they use the same alias, i.e., ACTELCELLS.

To simulate multichip Actel designs, use the following procedure:

1. Create a top level schematic and instantiate the individual chip designs. For this example, assume there are three designs with instance names "chip1," "chip2," and "chip3." The name of the top level schematic is "top." Figure 1 depicts the directory structure for this example. Notice that the names written in normal text represent file names and those in bold text represent directory names.

   Note: *This example contains only single-sheet schematics for each design. Similar procedures apply to multisheet schematic designs as well.*

2. Extract the timing of the three chip designs using the Actel Designer Series software so that you get the "chip1.del," "chip2.del," and "chip3.del" files.
3. Run the backannotation program (del2vl) on each of the three chips so that you get the "chip1.dtb," "chip2.dtb" and "chip3.dtb" files.
4. With an editor, generate a "top.dtb" file for the top level schematic. The format of the top level DTB should be as follows:

```
.ba
c chip1
a dtb=chip1.dtb
c chip2
a dtb=chip2.dtb
c chip3
a dtb=chip3.dtb
.ab
```

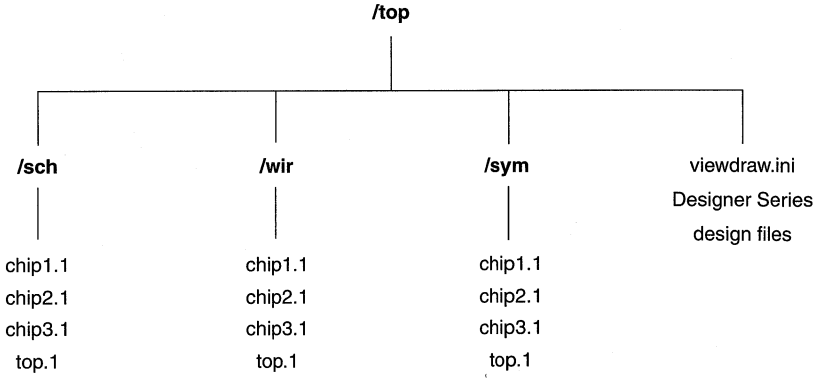


Figure 1 • *Required Directory Structure for Multichip Simulation Using Powerview Viewsim*

Note: *The C line in the above example specifies an instance name, for example, "chip1." If you haven't labeled an instance, you can use the default handle name of an instance, for example, "$1I38" as it appears in your top-level schematic. Also, the individual DTB files should reside in the top level design directory, for example: "top."*

5. Run the VSM program on the top level design. This is done from the Powerview Cockpit. Reference the "top.dtb" file in the VSM pop-up dialog box. The VSM program will then process the DTB files for each chip and create the "top.vsm" file with back-annotated post-layout timing delays.
6. Use the above "top.vsm" file to run the post-layout simulation, using Powerview Viewsim. Refer to the Powerview manuals for more information on system level simulation.

Multichip Simulation Using PRO Series/PROsim™

System designs are typically divided into functional modules that are implemented by several Actel devices. To check the functionality of the system, it is important to simulate all of the Actel devices together. The Actel Designer Series system is capable of multichip simulation with many common simulators. Here is an example of multichip post-layout simulation using PRO Series/PROsim with the Designer Series system on a PC.

This example requires the use of the PRO Capture Project Manager utility to set the project directory and to switch to different projects. The top-level system design schematic is *mltchip*, which contains two components, chip1 and chip2. These components represent two different Actel devices in the system. There are three subdirectories — chip1, chip2, and mltchip — in the c:\designs directory. Each of these subdirectories has sch, sym, and wir subdirectories. The chip1 and chip2 directories must contain all Designer-generated design files required for post-layout, backannotated timing simulation. Figure 1 represents the directory structure for this design. Note that the names written in normal text represent file names and those in bold text represent directory names. This example contains only single-sheet schematics for each design. Similar procedures apply to multisheet schematic designs as well.

Before you simulate multichip designs, make sure that you have the directory structure shown in Figure 1.

To simulate multichip Actel designs, use the following procedure:

1. Run the following two commands from the c:\designs\chipl directory. The sch and wir subdirectories must have all schematic and *wir* files for chip1.

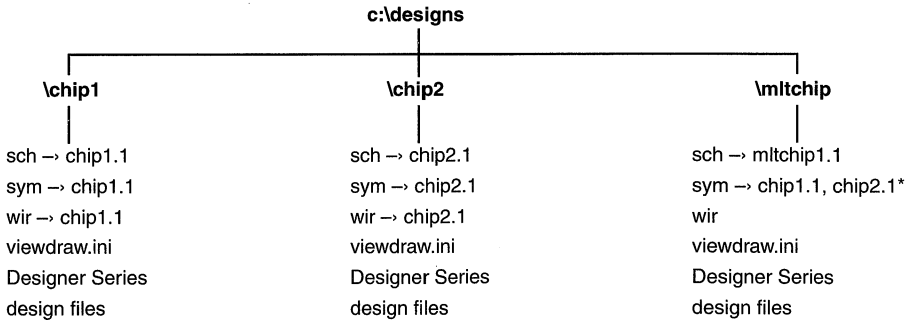
   ```
   del2vl chip1
   vsm chip1 -w -d chip1.dtb
   ```

   The *del2vl* program creates *chip1.vsm* and *chip1.dtb* files in the c:\designs\chip1 directory. The *vsm* program creates a flattened *wir* file chip1.1 in the same directory. At this point, there is a nonflattened *wir* file in c:\designs\chip1\wir and a flattened *wir* file in c:\designs\chip1.

2. Repeat step 1 for chip2 by running the following commands from the c:\designs\chip2 directory:

   ```
   del2vl chip2
   vsm chip2 -w -d chip2.dtb
   ```

3. Copy each flattened *wir* file (chip1.1 from c:\designs\chip1 and chip2.1 from c:\designs\chip2) to the c:\designs\mltchip\wir directory.



*These files are copies of chip1.1 and chip2.1 of c:\designs\chip1\sym\chip1.1, and c:designs\chip2\sym\chip2.1 files.

***Figure 1** • Required Directory Structure for Multichip Simulation Using PROsim*

4. Select Link to PROsim from the menu of the mltchip PRO Capture schematic window. This will place *mltchip.vsm* and *mltchip.1* files into the c:\designs\mltchip and c:\designs\mltchip\wir directories. Both of these files contain backannotated delay information. At this point, open a viewwave window to observe signal waveforms for the design.
5. (Optional) Copy the schematic files *chip1.1* and *chip2.1* from c:\designs\chip1\sch and c:\designs\chip2\sch to c:\designs\mltchip\sch. Then, each schematic can be observed with backannotated simulation values from the mltchip schematic. However, if you have more than two levels of hierarchy, the backannotated values will not be displayed in your schematic.

WARNING:

To run multichip simulation again for the same design, first remove the design schematic files chip1.1 and chip2.1 from the c:\designs\mltchip\sch directory. Then, follow the procedure from step 4. Otherwise, Link to PROsim (step 4) will create nonflattened vsm and wir files for chip1 and chip2, which will not contain backannotated delays.

Board Level Post-Layout Simulation Using Designer and Quicksim II

Introduction

System designs are typically divided into functional modules that are implemented by several Actel devices. To check the functionality of the system, it is important to simulate all of the Actel devices together. The Actel Designer Series system includes board-level, multichip simulation capability using Mentor Graphics software for Actel devices.

The software requirements are identical for chip-level and board-level simulations. The requirements are Mentor Graphics' Design Architect (DA), Design Viewpoint Editor (DVE), Quicksim II, and Actel's Designer Series system.

Create the Design

Consider the board-level design, *board*, which contains the chip-level components chip1, chip2, and chip3. Each design is composed of a symbol and a schematic. The board schematic includes chip1, chip2 and chip3 symbols, which are ACT 1, ACT 2 and ACT 3 family designs, respectively. The following procedure is depicted in Figure 1.

1. Create the chip1 design including all I/O pads and ports. Execute check sheet, and save the design.

   Open a symbol sheet to create a symbol for chip1 with its I/O pins. Add a model property with the name *als_technology* and a property value of ACT 1, ACT 2, or ACT 3. Execute check sheet, and save as chip1. If the message *"Property* als_technology *on the symbol is not on the interface."* appears, make no modifications to the design. This warning message does not indicate any problems with the symbol and will no longer appear after the symbol is saved. Verify by rerunning the check sheet after saving the symbol.
2. Run *mgc2adl* on the design to generate the Actel netlist by typing the following command:

   ```
   mgc2adl fam:<family name> chip1
   ```

   where <family name> = ACT 1, ACT 2 or ACT 3.
3. Use the Designer Series software to layout and extract post-layout delays for the device. Backannotate these delays to Quicksim II by typing the following command:

   ```
   del2mgc chip1
   ```

   This step creates the back annotation file, *chip1.bao*.
4. Repeat steps 1 through 4 for the chip2 and chip3 designs.

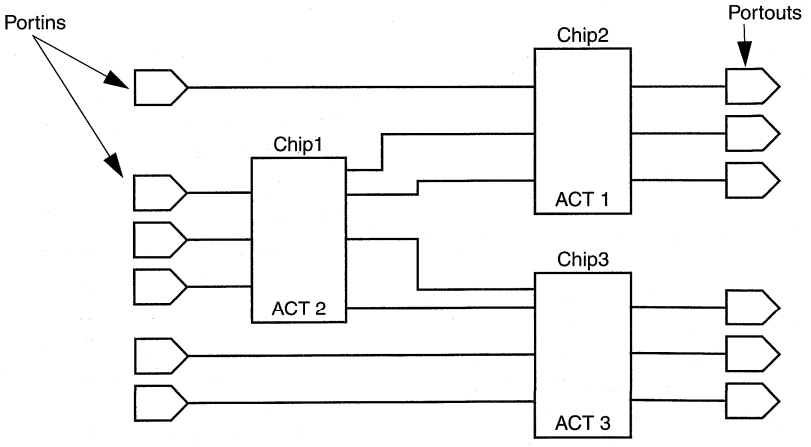


Figure 1 • Board Schematic Sheet

5. In Design Architect, create the schematic sheet board. From the popup menu, select

   ```
   Instance > Choose Symbol
   ```

   Use the Navigator to select and place chip1 on the schematic sheet. Repeat this procedure for chip2 and chip3. Connect the symbols and add Portins and Portouts to the schematic. Execute the check sheet command and save changes. Open a new symbol sheet, and create the board symbol. Execute the check sheet command, and save the changes to disk.

DVE Configuration (ACT 1 and ACT 2 only)

Next, configure the Design Viewpoint Editor to set up the design with backannotated post-layout delays.

1. From the board directory, invoke DVE for the design as follows:

   ```
   dve <viewpoint file name>
   ```

   If <viewpoint file name> is not specified, DVE uses the default viewpoint file, *default*.

2. Select File > Open > Sheet from the pull-down menu. Enter the board-level schematic name, *board*.

3. Select the chip1 and chip2 symbols, and execute Report > Object > Long. An Object window opens and lists the path contexts for the symbols, including references and property information. Verify that the path contexts are of the form

   ```
   /I$1 and /I$2.
   ```

4. Verify that the Design Configuration window is open. If it is not active, select File > Open > Viewpoint and verify that the board viewpoint is selected.

5. From the pop-up menu, select Add > Primitive to open an Add Primitive window. The primitive name is *model* with value of ACT 1 and ACT 2. The primitive type is *string*.

6. Verify that the PARAMETER is set to the als_technology variable with value equal to the ACT family, ACT 1 or ACT 2 or both. If the PARAMETER is not set, then from the pop-up menu, select Add > Parameter to open an Add Parameter window. The parameter name is als_technology with a value of ACT 1 or ACT 2 or both.

7. Select File > Back Annotation > Import from the pull-down menu to open an Import Back Annotation window. In the ASCII BA field, enter the path to the *chip1.bao* file. Leave the BA Name field set to the default viewpoint. In the import context field, add the path context of chip1. Repeat step 6 for chip2. Click the OK button to import the delays to DVE.

8. Save the viewpoint information. From the pull-down menu, select File > Save Design Viewpoint. Exit DVE.

9. Invoke Quicksim II with backannotated delays by entering the command

   ```
   quicksim . -tim max <viewpoint file name>
   ```

The board-level design is ready for simulation with post-layout, backannotated delays from Designer.

DVE Configuration (Including ACT 3)

An extra step is required to simulate ACT 3 chips on board with ACT 1 or ACT 2 devices. The simulation models for ACT 1 and ACT 2 I/O macros are defined at the symbol level, which is considered to be the primitive level; the simulation models for the ACT 3 I/O macros are defined one level below the symbol level.

1. To set the ACT 3 simulation models, type the following command:

   ```
   presimvpt fam:act3 board
   ```

2. From the board directory, invoke DVE for the design as follows:

   ```
   dve <viewpoint file name>
   ```

   If <viewpoint file name> is not specified, DVE uses the default viewpoint file, *default*.

3. Select File > Open > Sheet from the pull-down menu. Enter the board-level schematic name, *board*.

4. Select the chip3 symbol, and execute Report > Object > Long. An Object window opens and lists the path context for the symbol, including references and property information. Verify that the path context is of the form

   ```
   /I$1
   ```

   Remember the path context for future reference. Close the Object window.

5. Verify that the Design Configuration window is open. If it is not active, select File > Open > Viewpoint and verify that the board viewpoint is selected.

6. From the pop-up menu, select Add > Primitive to open an Add Primative window. The primitive name is *model* with value of ACT 3. The primitive type is *string*.

7. Verify that the PARAMETER is set to the als_technology variable with value equal to the ACT family, ACT 1, ACT 2, or ACT 3. If the PARAMETER is not set, then from the pop-up menu, select Add > Parameter to open an ADD PARAMETER window. The parameter name is als_technology with a value of ACT 1, ACT 2, or ACT 3 — or all three.

8. From the pull-down menu, select File > Back Annotation > Import to open an Import Back Annotation window. In the ASCII BA field, enter the path to the *chip3.bao* file. Leave the BA Name field set to the default viewpoint. In the import context field, add the path context of chip3 found in step 4. Click the OK button to import the delays to DVE.
9. Select the chip2 symbol, and execute Report > Object > Long to extract the path context for the symbol, including references and property information. Remember the path context for future reference. Close the Object window.
10. In the board schematic sheet, open the chip2 symbol. Select all I/O macros, and add a comp property with a prim value. This will prevent DVE from setting the ACT 2 simulation models at a level below the ACT 2 I/O symbols.
11. From the pop-up menu, select Add > Primitive to open an Add Primitive window. The primitive name is *model* with a value of ACT 2. The primitive type is *string*.
12. From the pull-down menu, select File > Back Annotation > Import. In the ASCI-I BA field, enter the path to the *chip2.bao* file. In the import context field, add the path context of chip2. Click on the OK button to import the delays to DVE.
13. Repeat steps 8 through 11 for chip1, and replace ACT 2 with ACT 1 at step 10.
14. Save the viewpoint information. From the pull-down menu, select File > Save Design Viewpoint. Exit DVE.
15. Invoke Quicksim II with backannotated delays by entering the command

```
quicksim . -tim max <viewpoint file name>
```

The board-level design is ready for simulation with post-layout, backannotated delays from Designer.

External Probe Circuit Control for Actel FPGAs

Introduction

Each ACT device includes internal Probe Circuitry to observe and analyze any signal inside an Actel chip. This feature is unique among FPGA vendors. The Probe Circuitry allows any internal signal to be observed on either of two external special function pins on the device.

The Probe Circuitry is normally accessed and controlled by either the Actel Debugger software or the Actionprobes. The Debugger software accesses the probe circuitry while the device is mounted on the Activator. This provides functional debugging capability before the device is inserted into a board. The Actionprobe is an extension of the Activator's ability to control the probes and allows access to the probe circuitry while the device is mounted on a board. This allows on-board, in-circuit debug and failure analysis of the logic design.

This application note contains the necessary information to control the probe circuitry via user-created external control signals and hardware. These control signals emulate the Activator to access the internal nodes of the chip.

Probe Circuit Control Pins

The ACT devices have three special function input pins to control the Probe Circuitry: Serial Data In (SDI), Data Clock (DCLK), and Mode selection (MODE). These three pins control the setup of the probe circuitry to allow observation of the internal signals on the external probe pins called *Probe A* (PRA) and *Probe B* (PRB). SDI and DCLK are the serial data input and clock of an internal serial Shift Register. (See Figure 1.) The MODE pin is a binary switch between the normal operating mode (MODE = 0) and the Probe Circuit mode (MODE = 1) of the chip. The pin numbers for all these special pins can be found in Section 1 of the data book in the "Package Pin Assignments" section for each family.

Important:

1. Avoid assigning input or bidirectional macros to the PRA and PRB pins. Those macro types will be active during probing, and the signals that appear on PRA and PRB will be fed back into the design. If you assign an output macro to the probe pins, their signals will not be observable and will not be fed back into the design.

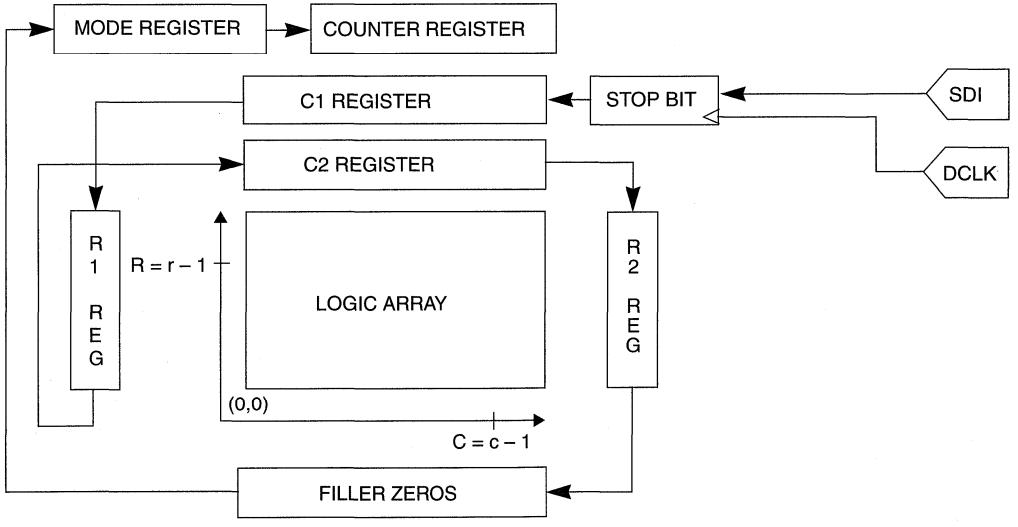


Figure 1 • Shift Registers for ACT Devices

10

2. Avoid assigning input or bidirectional macros to the SDI and DCLK pins. The Shift Register input and the input macro are in parallel, and the data being loaded into the Shift Register will affect the functionality of your design. Assigning output macros to these pins will disable the Probe Circuit on devices in the ACT 1 family.
3. Do not program the security fuses. Programming the security fuses will disable the Probe Circuits.

This is explained in more detail in the "Security Fuse Configuration" sections.

The Shift Register

The behavior of the External Probe Circuitry is controlled by the contents of the internal Shift Register. The Shift Register is composed of several short registers. The definitions of the Shift Registers can be found in Tables 1, 2, and 3. The short registers depicted in these tables appear in their respective loading sequence from top to bottom. For example, from Table 2 we can see the first short register loaded into the Shift Register of an ACT 2 device is the Counter Register (B0...B9) which is followed by the Mode Register (B0...B20), and then n filler zeros. The third and forth registers are the PRB row address (R2 register) and the PRB column address (C2 register). Finally, the PRA row address (R1 register), the PRA column address (C1 register), and the stop bit are loaded. How to "Determine the Shift Register Pattern" is explained in detail in that section.

Probe Circuit Control Signals

Use the following procedure to externally control the Probe Circuit:

1. Connect V_{pp} and V_{sv} to V_{cc}.

   V_{sv} is a power pin associated with ACT 2 and ACT 3 devices only. For ACT 1 devices, you only need to connect V_{pp} to V_{cc}.
2. Connect V_{KS} to GND.

   V_{KS} is a power pin associated with ACT 2 and ACT 3 devices only. Skip this step for ACT 1 devices.
3. Set the MODE pin to logic 0 upon power up.
4. Set the MODE pin to logic 1 after power has been applied to put the chip into the Probe Circuit mode.
5. Load the Shift Register pattern (see Figure 2) through the SDI pin by clocking each bit in with the DCLK pin. How to determine the Shift Register pattern for the desired signal is explained in detail in the next section.

   The waveform in Figure 2 shows the timing sequence for the Probe Circuit control signals, MODE, SDI, and DCLK. Fmax for DCLK is 10 MHz. Setup and hold times for SDI are both 10 ns with respect to DCLK. Fmax for the Probe Circuitry is technology dependent. Fmax for each device is summarized in Table 11.

Note: *DCLK is a falling edge triggered clock for the ACT 1 devices and a rising edge triggered clock for ACT 2 and ACT 3 devices.*

6. Observe the waveform at the probe pins.
7. If you want to look at another internal signal, MODE should be pulsed to a logic 0 before another shift register pattern is loaded into the shift register.
8. If you are finished probing, the chip is returned to the normal operating mode by placing a logic 0 on the MODE pin.

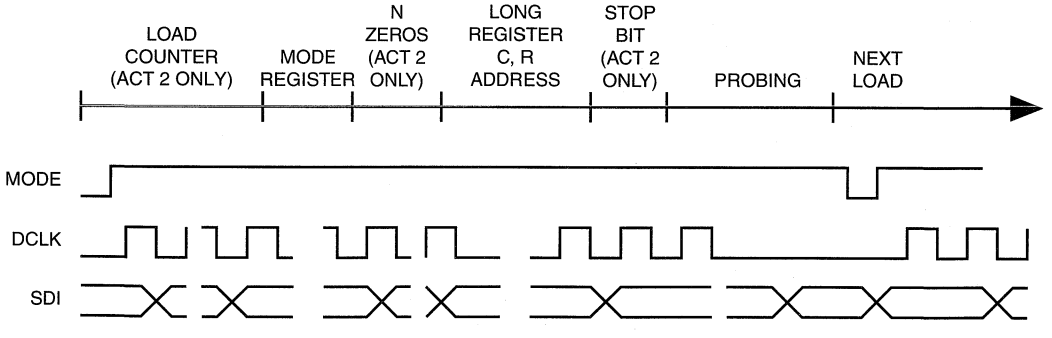


Figure 2 • Timing Sequence

Determining the Shift Register Pattern

Tables 1, 2, and 3 define the Shift Register's structure for each family. Each table is unique with respect to the specific family. For example, the bit orders for the R and C Registers vary significantly between families. Unlike ACT 1 devices, ACT 2 and ACT 3 devices have a Counter Register. In these tables, the short registers are depicted in their respective loading sequence from top to bottom.

Table 1 • ACT 1 Shift Register Definition

Register Name	Bits	Comments
Mode Register	7 bits	See Table 4
R2 Register	r bits	bit(r-1)...bit0
C2 Register	c bits	bit(c-1)...bit0
R1 Register	r bits	bit0...bit(r-1)
C1 Register	c bits	bit0...bit(c-1)
R1, C1 = Row and Column addresses for PRA R2, C2 = Row and Column addresses for PRB		

Table 2 • ACT 2 Shift Register Definition

Register Name	Bits	Comments
Counter Register	10 bits	See Table 6
Mode Register	21 bits	See Table 5
Filler Zeros	n bits	See Table 6
R2 Register	r bits	bit0...bit(r-1)
C2 Register	c bits	See Tables*
R1 Register	r bits	bit0...bit(r-1)
C1 Register	c bits	See Tables*
*The column addresses must be entered in the order specified in Tables 8, 9, and 10. R1, C1 = Row and Column addresses for PRA R2, C2 = Row and Column addresses for PRB		

Table 3 • ACT 3 Shift Register Definition

Register Name	Bits	Comments
Counter Register	10 bits	See Table 6
Mode Register	21 bits	See Table 5
Filler Zeros	n bits	See Table 6
R2 Register	r bits	bit(r-1)...bit0
C2 Register	c bits	bit(c-1)...bit0
R1 Register	r bits	bit0...bit(r-1)
C1 Register	c bits	bit0...bit(c-1)
R1, C1 = Row and Column addresses for PRA R2, C2 = Row and Column addresses for PRB		

Tables 4 and 5 contain the Mode Register's bit patterns for each family. The number of bits varies between families. ACT 1 devices only require 7 bits while ACT 2 and ACT3 devices require 21 bits. However, the bits are always loaded from LSB to MSB (M0...Mn). There are three probe modes for each family: PRA Only, PRB Only, and both PRA and PRB.

Table 4 • ACT 1 Mode Register Patterns

Probe Mode	Mode Register Pattern (M0...M6)
PRA Only*	1001001
PRB Only*	0101001
PRA & PRB	1101001
*When using only one probe, a dummy address (all zeros) must be used for the other probe.	

Table 5 • ACT 2 & ACT 3 Mode Register Patterns

Probe Mode	Mode Register Pattern (M0...M20)
PRA only*	000000110001101000000
PRB only*	000000101001101000000
PRA & PRB	000000111001101000000
*When using only one probe, a dummy address (all zeros) must be used for the other probe.	

The number of filler zeros required after the Counter Register's bit pattern is depicted with the Counter Register's patterns in Table 6. The number of clock cycles to load the Shift Register is also contained in this table. The number of clock cycles includes loading the stop bit. The Counter Register's bits are also always loaded from LSB to MSB (Bit 0...9).

Table 6 • Counter Register Patterns

Device	Fill Zeros (n)	Counter Pattern (Bit 0...9)	Clock Cycles*
A14100	664	0111001011	906
A1460	506	0011000011	690
A1440	363	1100100001	527
A1425	295	0111001011	423
A1415	256	1101101011	368
A1280	443	0011011111	675
A1240	357	1111000001	541
A1225	308	1101011010	458
A1020	0	NA	124
A1010	0	NA	112
* The number of clock cycles includes the stop bit.			

The contents of the R and C registers are unique with respect to the signal you wish to probe in your design. Therefore, they must be constructed from the information in Table 7 and the instructions that follow this table.

Internally, the logic is arranged in arrays of Logic Modules and routing channels. The row and column configurations are depicted in Table 7.

Table 7 • Array Definitions

Device	Rows		Columns	
A14100	19	(R0-R18)	86	(C0–C85)
A1460	18	(R0-R17)	58	(C0–C57)
A1440	14	(R0-R13)	52	(C0–C51)
A1425	12	(R0-R11)	36	(C0–C35)
A1415	10	(R0-R9)	30	(C0–C29)
A1280	18	(R0-R17)	82	(C0–C81)
A1240	14	(R0-R13)	62	(C0–C61)
A1225	13	(R0-R12)	46	(C0–C45)
A1020	14	(R0-R13)	44	(C0–C43)
A1010	8	(R0-R7)	44	(C0–C43)

The signal you are interested in probing is associated with the output of a single Logic Module. The *<design_name>.loc* file contains the XY coordinates of the desired Logic Module. These coordinates are translated into an address that is loaded into the Shift Register. The Shift Register then temporarily routes the output of the Logic Module to one of two probe pins on the device via the Probe Circuitry. The information in Table 7 depicts the length of the R and C registers for each device. This information is used to construct the addresses for PRA and PRB.

The PRB address (R2 + C2) and the PRA address (R1 + C1) are the last registers in the Shift Register pattern. Obtain the X and Y coordinates for the Logic Module from the *<design_name>.loc* file. The X coordinate corresponds to the column address (C register), and the Y coordinate corresponds to the row address (R register). Addressing for the row and column locations is active high. In other words, to select the output signal of the Logic Module at XY, bit(Y)=1 for row Y and bit(X)=1 for column X. Deselection of the remaining rows and columns is active low. The ACT 2 devices are an exception to the rule with respect to the relationship between the X coordinate and bit(X). The ACT 2 column address is scrambled with respect to the bit order. For ACT 2 probing, use Tables 8, 9, and 10 to determine the bit pattern for the C1 and C2 registers. The numbers in these tables correspond to the X coordinates found in the *<design_name>.loc* file. For example, if the device is an A1240A and the X coordinate determined from the *<design_name>.loc* file is 58 and we are using the C2 register to direct a signal onto PRB, then a one is placed into bit(2) of the C2 register and the remaining C2 bits are zeros.

A more detailed example of how to construct the Shift Register's pattern can be found at the end of this application note.

Table 8 • A1280A C Register Order

Register	Bit Order (Bit0...Bit81)
C2	< 80, 81, 78, 79, 77, 76, 74, 75, 73, 72, 70, 71, 69, 68, 66, 67, 65, 64, 62, 63, 61, 60, 58, 59, 57, 56, 54, 55, 53, 52, 50, 51, 49, 48, 46, 47, 45, 44, 42, 43, 41, 40, 38, 39, 37, 36, 34, 35, 33, 32, 30, 31, 29, 28, 26, 27, 25, 24, 22, 23, 21, 20, 18, 19, 17, 16, 14, 15, 13, 12, 10, 11, 9, 8, 6, 7, 5, 4, 2, 3, 1, 0 >
C1	< 1, 0, 2, 3, 5, 4, 6, 7, 9, 8, 10, 11, 13, 12, 14, 15, 17, 16,18, 19, 21, 20, 22, 23, 25, 24, 26, 27, 29, 28, 30, 31, 33, 32, 34, 35, 37, 36, 38, 39, 41, 40, 42, 43, 45, 44, 46, 47, 49, 48, 50, 51, 53, 52, 54, 55, 57, 56, 58, 59, 61, 60, 62, 63, 65, 64, 66, 67, 69, 68, 70, 71, 73, 72, 74, 75, 77, 76, 78, 79, 80, 81 >

Table 9 • A1240A C Register Order

Register	Bit Order (Bit0...Bit61)
C2	< 60, 61, 58, 59, 57, 56, 54, 55, 53, 52, 50, 51, 49, 48, 46, 47, 45, 44, 42, 43, 41, 40, 38, 39, 37, 36, 34, 35, 33, 32, 30, 31, 29, 28, 26, 27, 25, 24, 22, 23, 21, 20, 18, 19, 17, 16, 14, 15, 13, 12, 10, 11, 9, 8, 6, 7, 5, 4, 2, 3, 1, 0 >
C1	< 1, 0, 2, 3, 5, 4, 6, 7, 9, 8, 10, 11, 13, 12, 14, 15, 17, 16,18, 19, 21, 20, 22, 23, 25, 24, 26, 27, 29, 28, 30, 31, 33, 32, 34, 35, 37, 36, 38, 39, 41, 40, 42, 43, 45, 44, 46, 47, 49, 48, 50, 51, 53, 52, 54, 55, 57, 56, 58, 59, 60, 61 >

Table 10 • A1225A C Register Order

Register	Bit Order (Bit0...Bit45)
C2	< 44, 45, 42, 43, 41, 40, 38, 39, 37, 36, 34, 35, 33, 32, 30, 31, 29, 28, 26, 27, 25, 24, 22, 23, 21, 20, 18, 19, 17, 16, 14, 15, 13, 12, 10, 11, 9, 8, 6, 7, 5, 4, 2, 3, 1, 0 >
C1	< 1, 0, 2, 3, 5, 4, 6, 7, 9, 8, 10, 11, 13, 12, 14, 15, 17, 16,18, 19, 21, 20, 22, 23, 25, 24, 26, 27, 29, 28, 30, 31, 33, 32, 34, 35, 37, 36, 38, 39, 41, 40, 42, 43, 44, 45 >

Timing Characteristics of the Probe Circuitry

The Probe Circuitry has its own maximum operating frequency. The Probe Circuit's Fmax is typically less than the Fmax of the device in the normal operating mode. The Probe Circuitry will not function correctly if the internal signal you want to probe exceeds the Probe Circuit's Fmax. In some cases, it may be necessary to decrease the frequency associated with the internal signal you wish to probe to use the Probe Circuitry. Table 11 summarizes the maximum operating frequency of the Probe Circuitry.

Table 11 • PRA & PRB Fmax Summary

Devices	Fmax
A1010, A1020, A1010A, A1020A	10 MHz
A1010B, A1020B (std, –1, –2)	25 MHz
A1010B, A1020B (–3)	35 MHz
A1225, A1240, A1280	45 MHz
A1225A, A1240A, A1280A	45 MHz
A1415A, A1425A, A1440A, A1460A, A14100A	80 MHz

Probing Devices in the BGA Package

Actel is now using a new technology in packaging devices. Select devices will now be packaged in the Ball Grid Array (BGA) package. This package is similar to the PGA package. However, once the package is mounted on a board, there is no direct access to the device pins. Therefore, to use the Probe Circuit you must break out SDI, DCLK, PRA, and PRB to an edge connector or test points on the board. In general, this is a good technique, especially for any type of preproduction board. If you can afford the board space for the signal lines, we highly recommend this technique of the BGA package.

ACT 1 Security Fuse Configurations

ACT 1 devices contain two security fuses: PROBE and PROGRAM. Programming the PROBE fuse disables the Probe Circuitry, which disables the Debugger and the Actionprobe diagnostic tools. Programming the PROGRAM fuse prevents further programming of the device, including the programming of the PROBE fuse.

Table 12 summarizes the effects of programming the security fuses on the SDI, DCLK, PRA, and PRB pins.

In the normal operating mode (MODE=0), all undefined device pins in a design are automatically configured as active Low outputs. However, the SDI and DCLK pins are handled in a slightly different manner. If the PROGRAM fuse is not programmed and SDI and DCLK are undefined, then they are configured as inactive inputs. In this case, SDI and DCLK pins should be tied to ground. If the PROGRAM fuse is programmed and SDI and DCLK are undefined, then they will become active Low outputs.

Table 12 • ACT 1 Security Fuse Configurations

Mode[1]	Program	Probe	PRA, PRB	SDI, DCLK
low	no	no	user-defined I/O	user-defined input[2]
low	no	yes[3]	user-defined I/O	user-defined input[2]
low	yes[4]	no	user-defined I/O	user-defined I/O
low	yes[4]	yes[3]	user-defined I/O	user-defined I/O
high	no	no	Probe Circuit outputs[5]	Probe Circuit inputs[6]
high	no	yes[3]	Probe Circuit disabled	Probe Circuit disabled
high	yes[4]	no	Probe Circuit outputs[5]	Probe Circuit inputs[6]
high	yes[4]	yes[3]	Probe Circuit disabled	Probe Circuit disabled

Legend for Table 12

1. The MODE pin switches the device between the normal operating mode (MODE=0) and the Probe Circuit mode (MODE=1).
2. The PROGRAM fuse must be programmed if the SDI or DCLK pin is to be used as an output or a bidirectional pin.
3. If the PROBE fuse is programmed, the Probe Circuit is permanently disabled, which disables the Debugger and the Actionprobe diagnostic tools.
4. If the PROGRAM fuse is programmed, all programming of the device is disabled. Programming the PROGRAM fuse also disables programming of the array fuses and the PROBE fuse. Therefore, the PROGRAM fuse must always be the last fuse programmed.
5. The PRA output and a separate I/O buffer share the use of a single device pin. The PRA output and the output function of the I/O buffer are multiplexed. The same is true for PRB. The Probe mode from Table 4 that is loaded into the Mode Register will determine which output buffer is active during probing. There are three possible Probe Modes: PRA Only, PRB Only, and both PRA and PRB.

   When the PRA Only mode is selected, the PRA output will become active, and the output function of the I/O buffer associated with the PRA pin will be inhibited. However, the input buffer portion of the I/O buffer associated with the PRA pin will still be active, and any internal signal that appears on the PRA output will be fed back through that input buffer to the internal Logic Modules. This may interfere with the expected function

of the design while probing. We recommend you use an input latch on PRA to prevent the feedback while probing. PRB will function as a normal I/O in the PRA Only mode.

The PRB Only mode is functionally equivalent to the PRA Only mode. PRA will also function as a normal I/O in the PRB Only mode.

When the PRA and PRB mode is selected, both the PRA and PRB outputs will become active and the output function of the I/O buffers associated with both pins will be inhibited. However, the input buffer portion of the I/O buffers associated with both pins will still be active, and any internal signals that appear on the PRA and PRB outputs will be fed back through the input buffers to the internal Logic Modules. Again, this may interfere with the expected function of the design while probing. We recommend you use an input latch on PRA and PRB to prevent the feedback while probing.

6. The SDI input and a separate I/O buffer also share the use of a single device pin. The SDI input and the input function of the I/O buffer are connected in parallel. When the MODE pin is high, both inputs are active. DCLK and its I/O buffer are connected the same way. Therefore, the external Probe Circuit control signals to those pins will also be sent to the internal Logic Modules. This may interfere with the expected function of the design while probing. Again, we recommend you use an input latch on SDI and DCLK to prevent the external Probe Circuit control signals from affecting the functionality of your design during probing.

   If either SDI or DCLK is configured such that the output function of the I/O buffer is active, then the PROGRAM fuse must be programmed. In this configuration, the signals from your design will feed back to the Shift Register and will interfere with the function of the Probe Circuitry. In addition, the I/O drivers will fight the external SDI and DCLK drivers. Damage to both drivers may occur.

ACT 2 and ACT 3 Security Fuse Configurations

The ACT 2 and ACT 3 devices contain one security fuse. Table 13 shows the ACT 2 and ACT 3 security fuse configurations. Programming the security fuse disables the Probe Circuitry and the use of the Debugger function and the Actionprobe diagnostic tools.

In the normal operating mode (MODE=0), all undefined device pins in a design are automatically configured as active Low outputs. You do not need to program the SECURITY fuse to enable SDI and DCLK as active Low outputs.

Table 13 • ACT 2 & 3 Security Fuse Configurations

Mode[1]	Security	PRA, PRB	SDI, DCLK
low	don't care	user-defined I/O	user-defined I/O
high	no	Probe Circuit outputs[3]	Probe Circuit inputs[4]
high	yes[2]	Probe Circuit disabled	Probe Circuit disabled

Legend for Table 13

1. The MODE pin switches the device between the normal operating mode (MODE=0) and the Probe Circuit mode (MODE=1).
2. If the SECURITY fuse is programmed, the probe circuit is permanently disabled, which disables the Debugger and the Actionprobe diagnostic tools.
3. The PRA output and a separate I/O buffer share the use of a single device pin. The PRA output and the output function of the I/O buffer are multiplexed. The same is true for PRB. The Probe mode from Table 5 that is loaded into the Mode Register will determine which output buffer is active during probing. There are three possible Probe Modes: PRA Only, PRB Only, and both PRA and PRB.

   When the PRA Only mode is selected, the PRA output will become active and the output function of the I/O buffer associated with the PRA pin will be inhibited. However, the input buffer portion of the I/O buffer will still be active, and any internal signal that appears on the PRA output will be fed back through that input buffer to the internal Logic Modules. This may interfere with the expected function of the design while probing. We recommend you use an input latch on PRA to prevent the feedback while probing. PRB will function as a normal I/O in the PRA Only mode. An input latch is an integral part of the I/O buffers in the ACT 2 and ACT 3 devices.

   The PRB Only mode is functionally equivalent to the PRA Only mode. PRA will also function as a normal I/O in the PRB Only mode.

When the PRA and PRB mode is selected, both the PRA and PRB outputs will become active and the output function of the I/O buffers associated with both pins will be inhibited. However, the input buffer portion of the I/O buffers will still be active, and any internal signals that appear on the PRA and PRB outputs will be fed back through the input buffers to the internal Logic Modules. Again, this may interfere with the expected function of the design while probing. Therefore, we recommend you use an input latch on PRA and PRB to prevent the feedback while probing. An input latch is an integral part of the I/O buffers in the ACT 2 and ACT 3 devices.

4. The SDI input and a separate I/O buffer also share the use of a single device pin. The SDI input and the input function of the I/O buffer are connected in parallel. When the MODE pin is high, both inputs are active. DCLK and its I/O buffer are connected the same way. Therefore, the external Probe Circuit control signals to those pins will also be sent to the internal Logic Modules. This may interfere with the expected function of the design while probing. Again, we recommend you use an input latch on SDI and DCLK to prevent the external Probe Circuit control signals from affecting the functionality of your design during probing. An input latch is an integral part of the I/O buffers in the ACT 2 and ACT 3 devices.

   The output function of the I/O buffers associated with SDI and DCLK will not interfere with the function of the Probe Circuitry while in the Probe mode. When the MODE pin is driven high, these outputs are inhibited. Therefore, the I/O drivers will not interfere with the external drivers. However, these outputs will not be observable in the Probe mode.

Example:

The bit stream shown in Figure 3 will probe the outputs of an AND gate with the instance name U1 and an OR gate with the instance name U2 in an A1240 device.

The behavior of the external Probe Circuitry is controlled by the internal Shift Register. The Shift Register is composed of several short registers. The definition of the ACT 2 shift register can be found in Table 2. The short registers depicted in Table 2 are in the loading sequence from top to bottom. Therefore, the first short register of the Shift Register is the Counter Register, and the A1240's bit pattern can be found in Table 6.

The next short register in the Shift Register's bit stream, is the Mode Register. Since two locations will be probed, both PRA and PRB will be used. The Mode Register's bit pattern for that configuration can be found in Table 5.

Filler zeros are next in the Shift Register's bit stream and the necessary number can be found in Table 6. The A1240 needs 357 filler zeros.

Next, the row and column addresses for both probes must be determined. The PRB address (R2 + C2) is next in the Shift Register's bit stream, followed by the PRA address (R1 + C1).

The following data is an excerpt from the *<design_name>.loc* file. The X coordinate corresponds to the column address (C register), and the Y coordinate corresponds to the row address (R register).

```
USE; U1;
XY: 8%5.
USE; U2;
XY: 58%10.
```

PRA will be used to probe the output of the AND gate (U1), and PRB will be used to probe the output of the OR gate (U2). Based on the data from the *<design_name>.loc* file, the R and C registers for this example are defined as follows:

R1 = 5, C1 = 8

R2 = 10, C2 = 58

Use Table 9 to determine the bit pattern for the C1 and C2 registers. The numbers in this table correspond to the X coordinates found in the *<design_name>.loc* file. In this example, the X coordinate for the C2 register is 58. Therefore, a one is placed in bit(2), and the remaining bits are zeros. Likewise, the X coordinate for the C1 register is 8. Therefore, a one is placed in bit(9), and the remaining bits are zeros.

The Y coordinates correspond to the bit numbers of the R registers for all the device families. In this example, the Y coordinate for the R2 register is 10. The bit pattern would have a one in bit(10). Since the Y coordinate for the R1 register is 5, the bit pattern for the R1 register would have a one in bit(5). The remaining bits are zeros. The shift sequence of the R register bits are specified in Table 2. The R registers for ACT 2 devices are shifted in LSB to MSB. However, all the device families have unique bit sequences.

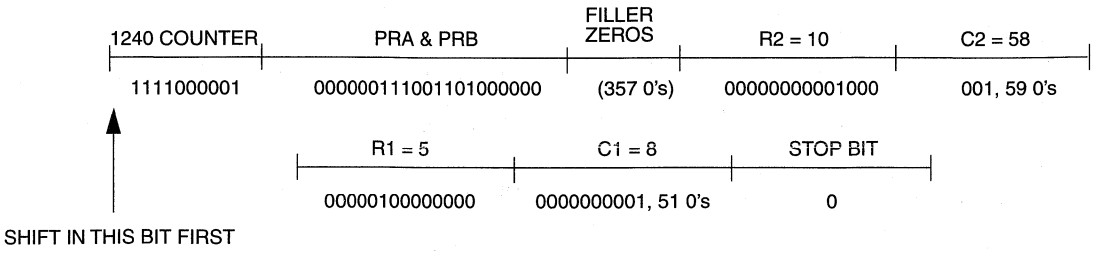


Figure 3 • *Example Shift Pattern*

Using Actionprobe® Diagnostic Tools

Introduction

The Actionprobe and the ACT™ device's unique internal Probe Circuitry can be used to observe and analyze any signal inside an Actel chip. Actel is the only FPGA vendor that provides this type of capability. This unique diagnostic tool allows on-board, in-circuit, real-time, 100 percent observability of the circuit design's logic. Designers have unrestricted access to all of the circuit design's individual component's inputs and outputs in real time by using the Actionprobe.

Probe Circuitry is accessed and controlled by either the Actel Debugger software or the Actionprobe and the In Circuitry Probe (ICP) software. The ICP software is a subprocess of the Debugger software. The Debugger software accesses the Probe Circuitry while the device is mounted on the Activator® programming unit. This provides functional debugging capability before the device is inserted into a board. The Actionprobe is an extension between the Activator and the designer's board. The Actionprobe facilitates the Activator's ability to control the Probe circuitry while the device is mounted on a board. This allows observation of real-time responses to actual operating stimuli and provides the optimal debug, design-verification, and failure-analysis capability.

Actionprobe Control Pins

The ACT devices have three special-function input pins to control the Probe circuitry: Serial Data In (SDI), Data Clock (DCLK), and Mode selection (MODE). These three pins control the setup of the probe circuitry to observe the internal signals on the external probe pins called *Probe A* (PRA) and *Probe B* (PRB). SDI and DCLK are the serial data input and clock, respectively, of an internal serial Shift Register. The MODE pin is a binary switch between the Normal operating mode (MODE = 0) and the Actionprobe mode (MODE = 1) of the chip. The pin numbers for all these special pins can be found in Section 1 in the "Package Pin Assignments" section for each family.

Important:

- Avoid assigning input or bidirectional macros to the PRA and PRB pins. Those macro types will be active during probing, and the signals that appear on PRA and PRB will be fed back into the design. If you assign an output macro to the probe pins, their signals will not be observable and will not be fed back into the design.
- Avoid assigning input or bidirectional macros to the SDI and DCLK pins. The Shift Register input and the input macro are in parallel, and the data being loaded into the Shift Register will affect the functionality of your design. Assigning output macros to these pins will disable the Probe circuit on devices in the ACT 1 family.
- Do not program the security fuses. Programming these fuses will disable the Probe circuits.

This is explained in more detail in the sections on security fuse configuration.

Setup

The Actionprobe hardware for the Activator 2 and 2S programmers consists of a diagnostic pod that connects to the programmer by cable. The pod is connected to dedicated test points in the target system by using test points. The Activator 2 programmer supports up to four Actionprobe diagnostic pods. The Activator 2S supports one Actionprobe at a time. The device is verified and debugged in the target board as it receives real-time stimuli from the system.

SDI, DCLK, and MODE should be terminated to ground through a 10K or greater resistor. Both probe pins can be connected directly to a logic analyzer or oscilloscope.

In-Circuit Probing

Initializing and changing the signal nodes is done simply by changing node names with the ICP software. The newly assigned signals are connected automatically to the probe pins. The internal signal passes through an inverting buffer in the A1010, A1010A, A1020, and A1020A devices before reaching the probe pins. All other devices are noninverting at the probe pin.

Timing Characteristics of the Probe Circuitry

The Probe circuitry has its own maximum operating frequency. The Probe circuit's Fmax is typically less than the Fmax of the device in the normal operating mode. The Probe circuitry will not function correctly if the internal signal you want to probe exceeds the Probe circuit's Fmax. In some cases, it may be necessary to decrease the frequency associated with the internal signal you wish to probe to use the Probe circuitry. Table 1 summarizes the maximum

operating frequency of the Probe circuitry.

Table 1 • PRA & PRB Fmax Summary

Devices	Fmax
A1010, A1020, A1010A, A1020A	10 MHz
A1010B, A1020B (std,-1,-2)	25 MHz
A1010B, A1020B (-3)	35 MHz
A1225, A1240, A1280	45 MHz
A1225A, A1240A, A1280A	45 MHz
A1415A, A1425A, A1440A, A1460A, A14100A	80 MHz

Probing Devices in the BGA Package

Actel is now using a new technology in packaging devices. Select devices will now be packaged in the Ball Grid Array (BGA) package. This package is similar to the PGA package. However, once the package is mounted on a board, there is no direct access to the device pins. Therefore, to use the Probe circuit, you must break out SDI, DCLK, PRA and PRB to an edge connector or to test points on the board. In general, this is a good technique, especially for any type of preproduction board. If you can afford the board space for the signal lines, we highly recommend this technique for the BGA package.

Probe Calibration

The probe circuitry does not introduce any additional loading, so the AC characteristic of the observed internal nodes remains unchanged. And, because probe propagation delay is independent of layout, probe delay remains unchanged for all points in the device.

The skew of the probe pins can be measured and used to calibrate the propagation delay. When both probe pins are assigned to the same point on the device, the measured delay difference is the skew of the probe pins. This skew is subtracted from subsequent delay measurements in the circuit. In Figure 1, PRA and PRB are connected electrically to node Net0. The delay difference is the skew, calculated as

$$t_{SK} = t_{PRA} - t_{PRB}$$

Using the Debugger software, the slower probe (PRA) is assigned to node Net3. Figure 2 shows this configuration. In this example, actual propagation delay is the measured delay between the output of G0 and the output of G3, minus the probe skew time. Actual delay is calculated as

$$t_{PD} = t_{PRA} - t_{PRB} - t_{SK}$$

Note: *Because of the difference in propagation delay between rising and falling signals, both probe pin signals must be either rising or falling when they are used to calibrate delay measurements.*

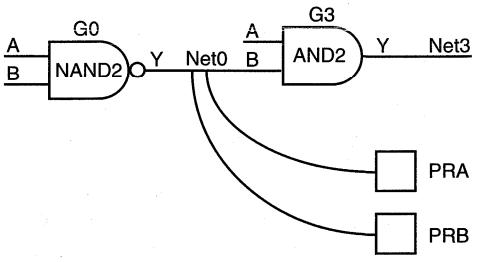


Figure 1 • Measuring Skew of Probe Pins

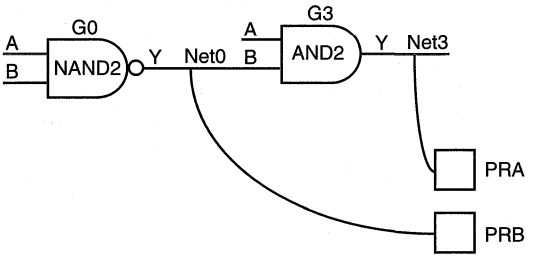


Figure 2 • Calibrating for Accurate Propagation Delay Measurement

ACT 1 Security Fuse Configurations

ACT 1 devices contain two security fuses: PROBE and PROGRAM. Programming the PROBE fuse disables the Probe circuitry which disables the Debugger and the Actionprobe diagnostic tools. Programming the PROGRAM fuse prevents further programming of the device, including the programming of the PROBE fuse.

Table 2 summarizes the effects of programming the security fuses on the SDI, DCLK, PRA, and PRB pins.

In the normal operating mode (MODE=0), all undefined device pins in a design are automatically configured as active LOW outputs. However, the SDI and DCLK pins are handled in a slightly different manner. If the PROGRAM fuse is not programmed and SDI and DCLK are undefined then they are configured as inactive inputs. In this case, SDI and DCLK pins should be tied to ground. If the PROGRAM fuse is programmed and SDI and DCLK are undefined then they will become active LOW outputs.

Table 2 • ACT 1 Security Fuse Configurations

Mode[1]	Program	Probe	PRA, PRB	SDI, DCLK
low	no	no	user-defined I/O	user-defined input[2]
low	no	yes[3]	user-defined I/O	user-defined input[2]
low	yes[4]	no	user-defined I/O	user-defined I/O
low	yes[4]	yes[3]	user-defined I/O	user-defined I/O
high	no	no	Probe Ckt outputs[5]	Probe Ckt inputs[6]
high	no	yes[3]	Probe Ckt disabled	Probe Ckt disabled
high	yes[4]	no	Probe Ckt outputs[5]	Probe Ckt inputs[6]
high	yes[4]	yes[3]	Probe Ckt disabled	Probe Ckt disabled

Legend for Table 2

1. The MODE pin switches the device between the normal operating mode (MODE=0) and the Actionprobe mode (MODE=1).
2. The PROGRAM fuse must be programmed if the SDI or DCLK pin is to be used as an output or a bidirectional pin.
3. If the PROBE fuse is programmed, the Probe Circuit is permanently disabled, which disables the Debugger and the Actionprobe diagnostic tools.
4. If the PROGRAM fuse is programmed, all programming of the device is disabled. Programming the PROGRAM fuse also disables programming of the array fuses and the PROBE fuse. Therefore, the PROGRAM fuse must always be the last fuse programmed.
5. The PRA output and a separate I/O buffer share the use of a single device pin. The PRA output and the output function of the I/O buffer are multiplexed. The same is true for PRB. The probe mode that is loaded into the mode register will determine which output buffer is active during probing. There are three possible probe modes: PRA Only, PRB Only, and both PRA and PRB.

   When the PRA Only mode is selected, the PRA output will become active and the output function of the I/O buffer associated with the PRA pin will be inhibited. However, the input buffer portion of the I/O buffer associated with the PRA pin will still be active, and any internal signal that appears on the PRA output will be fed back through that input buffer to the internal Logic Modules. This may interfere with the expected function of the design while probing. We recommend you use an input latch on PRA to prevent the feedback during probing. PRB will function as a normal I/O in the PRA Only mode.

   The PRB Only mode is functionally equivalent to the PRA Only mode. PRA will also function as a normal I/O in the PRB Only mode.

   When the PRA and PRB mode is selected, both the PRA and PRB outputs will become active and the output function of the I/O buffers associated with both pins will be inhibited. However, the input buffer portion of the I/O buffers associated with both pins will still be active, and any internal signals that appear on the PRA and PRB outputs will be fed back through the input buffers to the internal logic modules. Again, this may interfere with the expected function of the design while probing. We recommend you use an input latch on PRA and PRB to prevent the feedback duringprobing.
6. The SDI input and a separate I/O buffer also share the use of a single device pin. The SDI input and the input function of the I/O buffer are connected in parallel. When the MODE pin is high, both inputs are active. DCLK and its I/O buffer are connected the same way. Therefore, the external Probe circuit control signals to those pins will also be sent to the internal Logic Modules. This may interfere with the expected function of the design while probing. Again, we recommend you use an input latch on SDI and DCLK to prevent the external Probe circuit control signals from affecting the functionality of your design during probing.

   If either SDI or DCLK is configured such that the output function of the I/O buffer is active, then the PROGRAM fuse must be programmed. In this configuration, the signals from your design will feed back to the Shift Register and will interfere with the function of the Probe circuitry. In addition, the I/O drivers will fight the external SDI and DCLK drivers. Damage to both drivers may occur.

ACT 2 and ACT 3 Security Fuse Configurations

The ACT 2 and ACT 3 devices contain one security fuse. Programming the security fuse disables the Probe circuitry and the use of the Debugger function and the Actionprobe diagnostic tools.

Table 3 summarizes the affects of programming the security fuses onthe PRA, PRB, SDI, and DCLK pins.

Table 3 • ACT 2 and ACT 3 Security Fuse Configurations

Mode[1]	Security	PRA, PRB	SDI, DCLK
low	don't care	User-defined I/O	User-defined I/O
high	no	Probe ckt outputs[3]	Probe ckt inputs[4]
high	yes[2]	Probe ckt disabled	Probe ckt disabled

In the normal operating mode (MODE=0), all undefined device pins in a design are automatically configured as active lowoutputs. You do not need to program the SECURITY fuse to enable SDI and DCLK as active low outputs.

Legend for Table 3

1. The MODE pin switches the device between the normal operating mode (MODE=0) and the Actionprobe mode (MODE=1).
2. If the SECURITY fuse is programmed, the Probe circuit is permanently disabled, which disables the Debugger and the Actionprobe diagnostic tools.
3. The PRA output and a separate I/O buffer share the use of a single device pin. The PRA output and the output function of the I/O buffer are multiplexed. The same is true for PRB. The probe mode that is loaded into the Mode Register will determine which output buffer is active during probing. There are three possible Probe Modes: PRA Only, PRB Only, and both PRA and PRB.

   When the PRA Only mode is selected, the PRA output will become active and the output function of the I/O buffer associated with the PRA pin will be inhibited. However, the input buffer portion of the I/O buffer will still be active, and any internal signal that appears on the PRA output will be fed back through that input buffer to the internal logic modules. This may interfere with the expected function of the design while probing. We recommend you use an input latch on PRA to prevent the feedback during probing. PRB will function as a normal I/O in the PRA Only mode. An input latch is an integral part of the I/O buffers in the ACT 2 and ACT 3 devices.

   The PRB Only mode is functionally equivalent to the PRA Only mode. PRA will also function as a normal I/O in the PRB Only mode.

   When the PRA and PRB mode is selected, both the PRA and PRB outputs will become active and the output function of the I/O buffers associated with both pins will be inhibited. However, the input buffer portion of the I/O buffers will still be active, and any internal signals that appear on the PRA and PRB outputs will be fed back through the input buffers to the internal Logic Modules. Again, this may interfere with the expected function of the design while probing. Therefore, we recommend you use an input latch on PRA and PRB to prevent the feedback during probing. An input latch is an integral part of the I/O buffers in the ACT 2 and ACT 3 devices.
4. The SDI input and a separate I/O buffer also share the use of a single device pin. The SDI input and the input function of the I/O buffer are connected in parallel. When the MODE pin is high, both inputs are active. DCLK and its I/O buffer are connected the same way. Therefore, the external Probe ircuit control signals to those pins will also be sent to the internal Logic Modules. This may interfere with the expected function of the design while probing. Again we recommend you use an input latch on SDI and DCLK to prevent the external Probe circuit control signals from affecting the functionality of your design during probing. An input latch is an integral part of the I/O buffers in the ACT 2 and ACT 3 devices.

   The output function of the I/O buffers associated with SDI and DCLK will not interfere with the function of the Probe circuitry while in the probe mode. When the MODE pin is driven high, these outputs are inhibited. Therefore, the I/O drivers will not interfere with the external drivers. However, these outputs will not be observable in the probe mode.

Using the Actel Debugger as a Functional Tester

Introduction

Actel's Activator® programming and diagnostics unit, together with the Debugger software, provide powerful tools to functionally test an ACT™ device. Device debugging begins after design configuration and device programming. Debugging is performed with the device inserted in the Activator programming module.

A Debugger functional test allows the observation of any internal node or external pin of the device. Device inputs are defined with Debugger menu commands, with a command file, or with a combination of the two. Command files and test vectors are created with an ASCII text editor, then loaded into the Debugger. User-defined macros may be created in a command file, then executed in the Debugger.

This application note shows Debugger commands for a sample design. The sample design is Actel's TA269 (TTL 74269), an eight-bit binary loadable up/down counter with count enable. Figure 1 shows the sample design; Table 1 shows the truth table for the part; and Figure 2 shows a typical Debug command sequence. P0 through P7 are parallel load inputs; Q0 through Q7 are counter inputs; CLK is the counter clock; and UD is the up/down counter selector. Internal nodes (nets) should be labeled during design capture for easy reference during debugging.

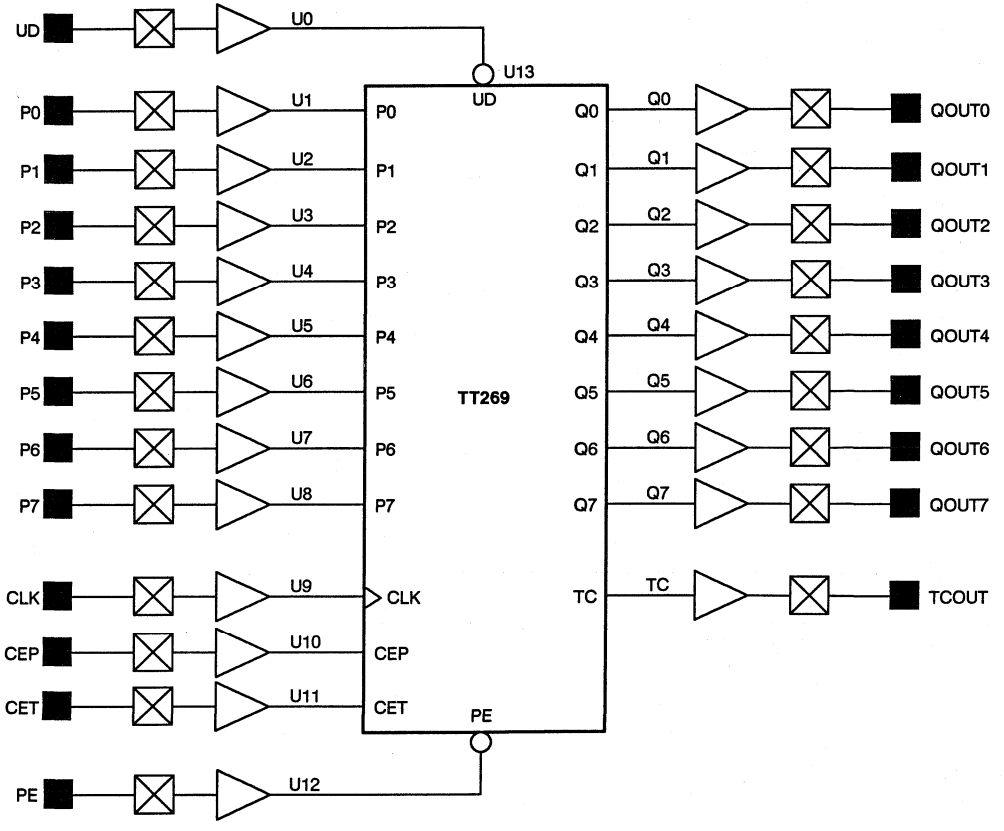


Figure 1 • Debugger Sample Design with TA269 Soft Macro

10

Table 1 • TA269 Truth Table

Inputs						Outputs	
PE	CEP	CET	UD	P[7:0]	CLK	Q[7:0]	TC
0	X	X	X	0	↑	0	1
0	X	X	X	FF	↑	FF	0
1	X	1	X	X	↑	Hold	
1	1	X	X	1	↑	Hold	
1	0	0	1	X	↑	Increment	
1	0	0	0	X	↑	Decrement	

Assigning Test Vectors

The default input radix for all text vectors is decimal. To specify a binary, hex, or octal radix, add a 0b, 0h, or 0o prefix, respectively, to the vector (for example, 0b1010 or 0h7e). Outputs are in binary format. To interactively define input test vectors, use the Debugger menu. As an alternative, use input command files to define test vectors. To view outputs and internal nodes, print them to the screen display or to an output file.

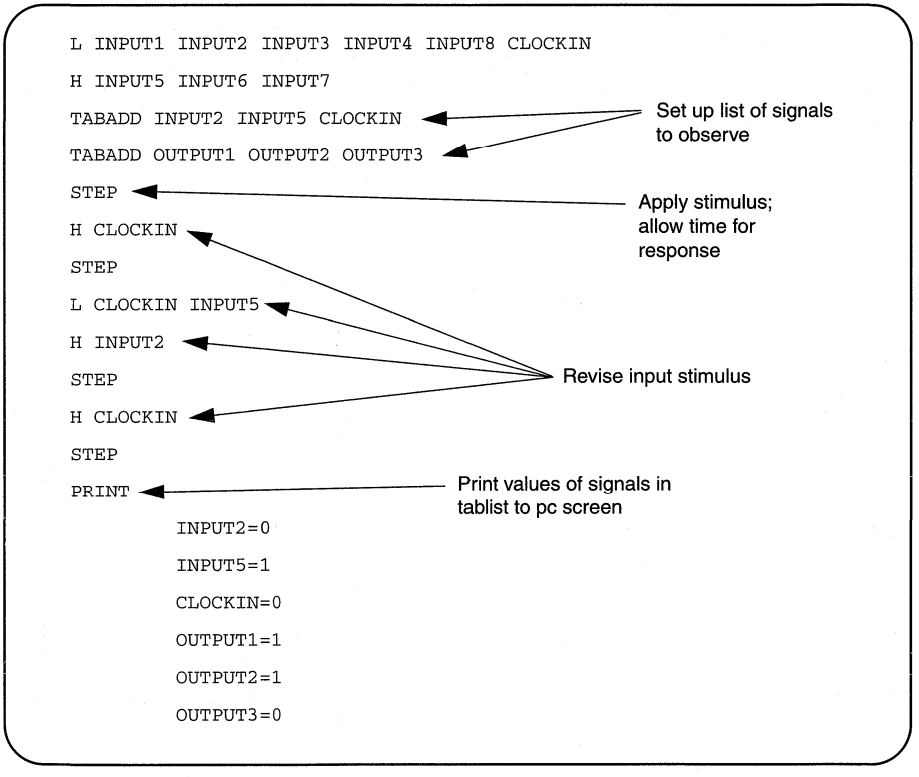


Figure 2 • Typical Debug Command Sequence

Defining User Macros

You may save time by creating user-defined macros for the Debugger. These macros may contain a series of basic Debug commands or may nest any combination of basic commands and user-defined macros.

The sample macro in part A of Figure 3, `clk10`, provides 10 clock pulses to the pin CLK and prints the value of internal vector Q to a specified output file after each clock pulse. The `outfile` command specifies the output file.

Part B of FFigure 3 shows a nested macro, `clk100`, that executes the `clk10` macro 10 times, providing 100 clocks to the CLK pin.

A

```
define (clk10) (repeat 10 (1 CLK) (step) (h CLK) (step) (fprint Q))
```

B

```
define (clk100) (repeat 10 (clk10))
```

Figure 3 • Sample Command Macros

Creating a Command File (`TA269.cmd`)

The command file (see Figure 4) applies test vectors to the `TA269` counter. It redirects results to an output file, `TA269.out`, and compares the output vector Q of the counter to an existing results file, `TA269.cmp`. The following notes correspond to each line in the command file.

Lines 1 and 2: The `vector` command defines eight parallel load input bits as vector P and counter output as vector Q.

Line 3: The `tabadd` command defines the internal or external nodes to be displayed or printed when the `print` or `fprint` command is executed.

Lines 4, 5, and 6: The `infile`, `outfile`, and `compfile` commands define input and output files. The `infile` command opens a file containing input test vectors. The `outfile` command contains the output results. The `compfile` command contains the data to be compared against the current device status. Use the full path name of the file, and enclose it in quotation marks.

Line 7: The `define` commands create user-defined macros. In this example, the `clk10` macro provides 10 clock pulses to the CLK input, `fprint` prints all nodes in the `tabadd` command to a file defined by `outfile`, and fcomp compares the status of vector Q to the file specified by `compfile`.

Lines 8, 9, and 10: Defines three user macros, `up`, `down`, and `load`.

Loading a Command File

The `loadfile` command loads a defined command file into the Debugger. Select `setup | loadfile` from the Debugger menu, then enter the full path name of the command file. For example:

```
/design/TS269/TA269.cmd
```

Executing User-Defined Macros

To execute any previously defined macros, type the macro at the command line while in the Debugger.

```
Line
  1    (vector P P0 P1 P2 P3 P4 P5 P6 P7)
  2    (vector Q Q0 Q1 Q2 Q3 Q4 Q5 Q6 Q7)
  3    (tabadd PE CEP CET US P CLK Q TC)
  4    (infile "/designs/ta269/ta269.pat")
  5    (outfile "/designs/ta269/ta269.out")
  6    (compfile "/designs/ta269/ta269.cmp")
  7    (define (clk10) (repeat 10 (1 CLK) (step) (h CLK) (step) (fprint) (fcomp Q)))
  8    (define (up) (1 CEP CET) (h PE UD) (step) (clk10))
  9    (define (down) (h PE) (1 CEP CET UD) (step) (clk10))
 10    (define (load) (1 PE CLK) (step) (fassign P) (h CLK) (step))
```

Figure 4 • Debug Command File (`TA269.cmd`)

Running the Sample Command File

In the following example, APS 2 assigns input test vectors from an input pattern file named `TA269.pat` and writes output results to a file named `TA269.out`. The command file (`TA269.cmd`) compares the counter's output vector Q to an existing results file (`TA269.cmp`).

Output Results File (`TA269.out`)

```
  S     P C C U          P C          Q T
  T     E E E D            L            C
  E       P T              K
  P
00005: 1 0 0 1 00000000 1 10000000 0
00007: 1 0 0 1 00000000 1 01000000 0
00009: 1 0 0 1 00000000 1 11000000 0
00011: 1 0 0 1 00000000 1 00100000 0
00013: 1 0 0 1 00000000 1 10100000 0
00015: 1 0 0 1 00000000 1 01100000 0
00017: 1 0 0 1 00000000 1 11100000 0
00019: 1 0 0 1 00000000 1 00010000 0
00021: 1 0 0 1 00000000 1 10010000 0
00023: 1 0 0 1 00000000 1 01010000 0
00028: 1 0 0 0 10000000 1 00000000 1
00030: 1 0 0 0 10000000 1 11111111 0
00032: 1 0 0 0 10000000 1 01111111 0
00034: 1 0 0 0 10000000 1 10111111 0
00036: 1 0 0 0 10000000 1 00111111 0
00038: 1 0 0 0 10000000 1 11011111 0
00040: 1 0 0 0 10000000 1 01011111 0
00042: 1 0 0 0 10000000 1 10011111 0
00044: 1 0 0 0 10000000 1 00011111 0
00046: 1 0 0 0 10000000 1 11101111 0
00051: 1 0 0 1 01000000 1 11000000 0
00053: 1 0 0 1 01000000 1 00100000 0
00055: 1 0 0 1 01000000 1 10100000 0
00057: 1 0 0 1 01000000 1 01100000 0
00059: 1 0 0 1 01000000 1 11100000 0
00061: 1 0 0 1 01000000 1 00010000 0
00063: 1 0 0 1 01000000 1 10010000 0
00065: 1 0 0 1 01000000 1 01010000 0
00067: 1 0 0 1 01000000 1 11010000 0
00074: 1 0 0 0 01000000 1 01000000 0
00076: 1 0 0 0 11000000 1 10000000 0
00078: 1 0 0 0 11000000 1 00000000 1
00080: 1 0 0 0 11000000 1 11111111 0
00082: 1 0 0 0 11000000 1 01111111 0
00084: 1 0 0 0 11000000 1 10111111 0
00086: 1 0 0 0 11000000 1 00111111 0
00088: 1 0 0 0 11000000 1 11011111 0
00090: 1 0 0 0 11000000 1 01011111 0
00092: 1 0 0 0 11000000 1 10011111 0
```

Input Pattern File (`TA269.pat`)

```
0b00000000
0b10000000
0b01000000
0b11000000
```

Output Compare File (`TA269.cmp`)

```
0b10000000
0b01000000
0b11000000
0b00100000
0b10100000
0b01100000
0b11100000
0b00010000
0b10010000
0b01010000
0b00000000
0b11111111
0b01111111
0b10111111
0b00111111
0b11011111
0b00011111
0b11101111
0b11000000
0b00100000
0b10100000
0b01100000
0b11100000
0b00010000
0b10010000
0b01010000
0b11010000
0b00110000
0b01000000
0b10000000
0b00000000
0b11111111
0b01111111
0b10111111
0b00111111
0b11011111
0b01011111
0b10011111
```

ChipEdit

Manual pin assignment and module placement is one approach for extracting maximum performance on a device for certain critical paths. ChipEdit is the tool, in the Designer software, that simplifies manual pin and placement assignments. It is a graphical tool that displays module placement on the device floor plan and allows you to assign and move pins or logic modules quickly. ChipEdit is most effective after the design has gone through automatic place and route, and some modules need more precise placement for timing considerations. ChipEdit is also good for fixing hierarchical blocks in the design. There are many considerations when using ChipEdit, and this document explains when to use ChipEdit effectively in the design.

Critical Path

If a critical path is not meeting timing specifications, you can check how the modules are placed by using ChipEdit. If the logic modules in the critical path are not closely spaced, or if one or two modules are placed on opposite sides of the chip, longer delays on your critical path will occur. In general, you should try to keep the modules on the critical path close together.

For example, in Figure 1, assume that the critical path is through the DF1 and OR2 modules, which are boxed for clarity. The modules are then placed and routed, and their placement is shown in Figure 2. Since the placement is close together, the performance will be good. Now, assume that the critical path passes through the DFC1B or the XOR2 module. As you can see in Figure 2, these modules are not closely spaced. Placing the modules in the circled area would result in better timing.

Hierarchical Blocks

Another use of ChipEdit is to locate and fix hierarchical blocks of a design. If there is a part of the design not meeting the timing specifications, ChipEdit is a simple solution to try first. If you are satisfied with timing on certain blocks of your designs, you can fix the whole hierarchical block, thereby fixing your placement, routing, and timing on that block. This will allow you the freedom to replace and reroute the other blocks not meeting your timing specification. For detailed information on how to fix hierarchical blocks, refer to *Designer for Windows User's Guide*.

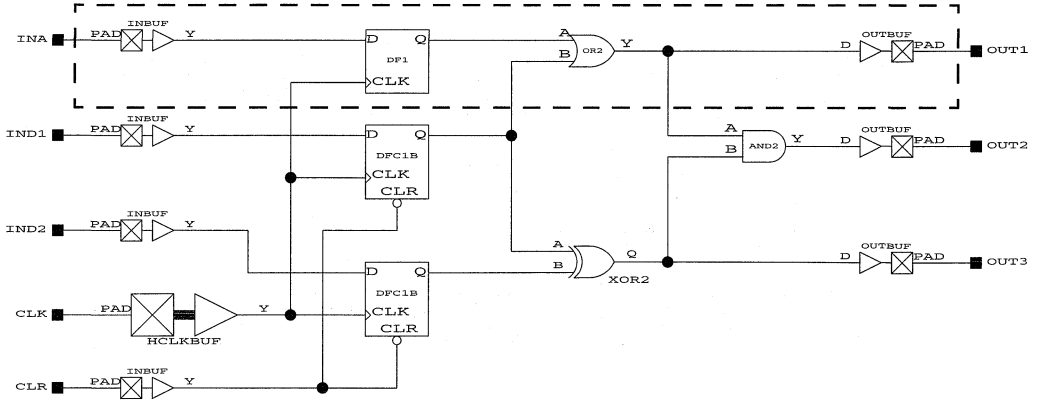


Figure 1 • Critical Path Flow

I/O Placement

Keep in mind that automatic place and route selects the best pin location depending on characteristics of the design. However, if manual I/O is necessary, you should keep these guidelines in mind to increase routablity and performance.

ChipEdit is also useful for placing and fixing I/O pins. On the ChipEdit screen, I/O modules are shown placed on the actual floor plan. You can manually change and fix I/Os in ChipEdit. For more detailed information on fixing and placing I/O pins, refer to *Designer for Windows User's Guide*. If any or all of the I/O placements must be assigned manually, a broader knowledge of Actel FPGA architecture is required. Refer to the *FPGA Design Guide* and *ACT Family Field Programmable Gate Array Databook* for pertinent information regarding different architectures of the ACT families. The structure of the logic modules and I/O modules, routing tracks, antifuses, and other architectural features are covered in detail. With this source of information, it is possible to assign I/O signals manually with the specific details of the device in mind.

Follow these guidelines for increasing routablity and performance:

- Try to force timing-critical signal paths, especially large data buses, to flow horizontally across the die, since there are more horizontal than vertical routing resources. In most cases, a large data bus requires many interconnections as it traverses the circuits. The greater number of horizontal routing tracks handle these interconnections to reduce routing congestion. The horizontal and vertical orientation of the die can be seen in ChipEdit, and you can also see the pin numbers of I/Os easily.
- Count the number of levels of logic between I/Os to determine placement. For example, I/Os that are separated by one gate should be placed closer than I/Os that are separated by a long shift register.
- Use global clock buffers. Each of these pins is connected to a dedicated, low-skew distribution network that is optimized for high fanout signals.
- Refer to the information in the DFR (Design for Routablity), PLI (Placement Information), and RTI (Routing Information) files as feedback for I/O assignment iterations. Each of these files indicates potential problem areas of the design that may be due to the I/O assignment. The DFR file reports problems due to routing congestion around groups of I/O pins. The PLI and RTI files report specific problems with the placement and routing of the device. Determine from this information where reassigning some I/Os will remove these problems.

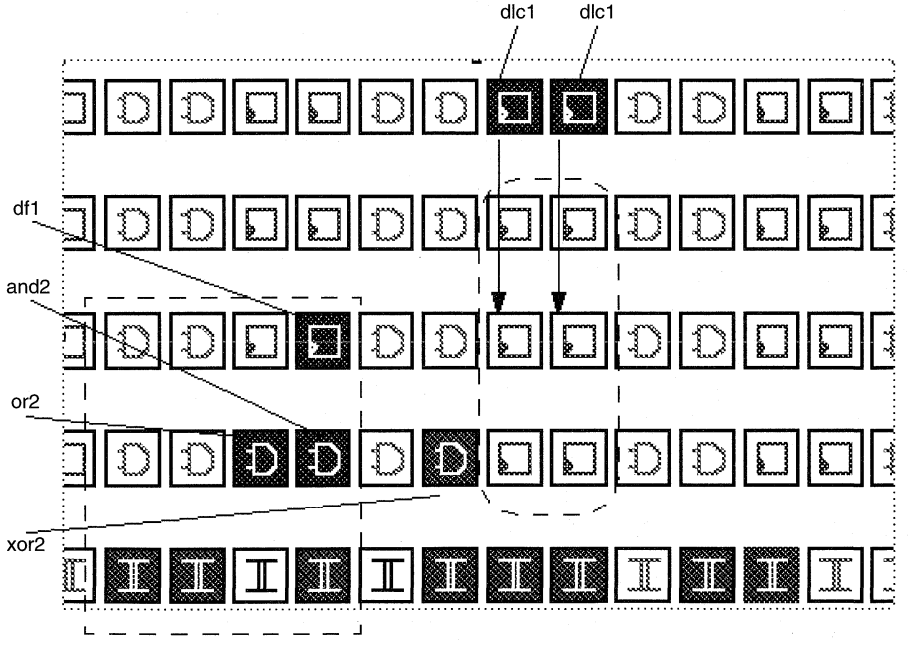


Figure 2 • Logic Floor Plan

Moving Actel FPGA Designs from Prototype to Production

Introduction

Requirements for system designs often change as a project moves from concept to prototype to production stages. Prototype designs and systems may be required to conform to specifications that are necessary in the final production release, but a system is usually faster and less expensive to develop with relaxed system constraints for the prototype. For example, prototype designs frequently employ "cut and jumper" techniques to modify quickly the connections between devices on a printed circuit board (PCB). Although these modification are effective and useful for debugging and verifying designs, they are unacceptable for the final product, which is produced only after incorporating the changes made during the prototyping stages.

Several features are built into ACT devices and the Designer Series software to provide the flexibility and ease of use that enable quick prototyping and transition to production. With these features, early prototypes can be quickly developed and modified to speed engineering schedules. Last-minute changes can be made without excessive delay, and final production versions of ACT designs can be produced without requiring additional, "schedule killing" development cycles.

Prototype Development

Ideally, system designs can be tested and debugged before PCBs are completed for production release. However, PCBs are often finalized and released to production well before the rest of a project is finished. This requires that the pin assignments for programmable devices with user-defined I/O pins be complete before the rest of the design. Since the pin assignments for all FPGAs largely determine their performance and routability, it is critical that the pin assignments be as close to ideal as possible even if the rest of the design is not complete.

Fortunately, the Designer Series software and ACT devices accommodate this type of scheduling requirement. To finalize the pin assignments early in the development cycle, you should first complete as much of the internal design (schematic) as possible. Then, let the Designer Series software automatically assign the locations of the I/Os. The Designer Series software considers the topology of each design to place I/Os in optimum locations. Therefore, you should approximate the design description (schematics, equations) as close to its final form for the best results. Refer to *Selecting and Modifying I/O Assignments* in this data book for more information.

After a design has been fully placed and routed, timing analysis with the Timer in the Designer Series software and backannotated simulation verifies the functionality and performance of the device. At this point, the pin assignments and PCB layout are complete. Small, last-minute design changes could lead to major problems if they resulted in reduced performance or if they required time-consuming modifications. The Designer Series software addresses this issue by including a feature that preserves the timing and performance of a design if small changes are required. If a design has satisfactory performance but less than five percent of its netlist must be modified, then the Incremental Placement option of the Designer Series software saves the timing of the rest of the design while replacing the modified portions. Another timing analysis of the design is often unnecessary, saving many hours of verifying and debugging.

Moving to Production

ACT devices are available in a wide variety of die sizes, package types, and performance grades. For maximum flexibility, prototype designs can be developed in device configurations different from the production device versions. All the "hooks" are incorporated into the design flow to ensure seamless, convenient changes from prototype to production. For example, designs can be prototyped in one package and then put effortlessly into production in another package. Suppose a military project requires ceramic pin grid array (CPGA) packages in the final product. In this case, prototypes can be initially developed in cheaper, plastic quad flat pack (PQFP) packages. When the design is complete, the repackager program in the Designer Series software can automatically transfer a design to another package type. The timing characteristics for the design are completely transferred to the new package without need for further timing analysis.

If a design is transferred to a package type with fewer I/O pins, there may not be a one-to-one correspondence between the I/Os of the two packages. The Action Facts (an automated document service) provides I/O cross-reference guides for equivalent die locations to help solve this problem. For example, if an A1280 design is transferred from a PG176 package (140 I/Os) to a PQ160 package (125 I/Os), some of the I/O pins may not map to bonded die locations of the new

device. In this case, the Designer Series software cannot automatically repackage the design. However, the cross-reference guides can be used to transfer accurately most of the I/O pins to equivalent locations.

Another feature of the ACT device involves the use of different speed grades. Each family of ACT devices is available in different speed grades to meet various performance requirements. Initial designs may incorporate cheaper, standard speed grade devices, but development can proceed even if the FPGA speed is inadequate for the particular application. In this case, you simply slow the system clock speed until the device performance is adequate. If the standard grade performance is within 25 percent of the system requirement, you choose the appropriate speed grade to satisfy the increased performance needs.

Summary

These are only some of the ways that the Designer Series software and ACT devices enable a smooth transition from prototype to production. With these tools, PCB procurement becomes less of a scheduling concern. Last-minute designs don't require that complete timing analysis be redone. Package and speed grade options allow a cheaper, faster development cycle without the usual associated hassles. Throughout the Designer Series design flow, many other convenient features enable similar time savings and development ease. Refer to the *Designer Series for the X Window System User's Guide* or the *Designer Series for Windows User's Guide* for a complete description of the Designer Series programs and capabilities.

Production Programming for Actel FPGAs

Introduction

Actel offers several programming options for the ACT™ families of field programmable gate arrays (FPGAs). The Activator® 2 desktop programmer supports all ACT device families. Up to four devices may be programmed simultaneously with the Activator 2. The Activator 2S is a single device desktop programmer with capabilities similar to the Activator 2. These Actel programmers support functional testing and in-circuit debugging with the APS 2 programming software and Actionprobe® diagnostic pod. The Actel programming hardware and software are compatible with 386/486 Pentium PC, Sun™, and HP700® workstations. In addition, Actel supports third-party programmers such as the UniSite from Data I/O.

In the early development stages of an Actel FPGA design, single devices are typically programmed at an engineer's desk or in the prototyping laboratory. The Activator 2S is ideally suited for this application with its single module adapter, compact size, and reduced cost. In most cases, the same computer workstation is used to develop, simulate, program, and debug the prototype devices. As the project moves from the engineering development to production phase, the device programming quantities may increase dramatically. The Activator 2 programmers are better suited for this application. The change in programming quantity requires the development of independent programming stations to meet this demand. To build reliable, stand-alone programming stations, several hardware and software issues must be adequately addressed.

Production Programming Work Area

Creating and using a dedicated programming work area is the most effective way to program Actel FPGAs in production-level quantities. Since Actel FPGAs are CMOS devices, appropriate measures are necessary to ensure that programming problems are minimized. Even though all Actel devices have built-in ESD protection, proper handling procedures are still required. All personnel in the programming work area should be properly equipped with antistatic clothing and wrist straps. In general, use standard CMOS device handling procedures for Actel devices.

The Activator programmers should be placed in a clean, well-ventilated location. Since each Activator contains an internal power supply, avoid exposure to heat sources such as direct sunlight or heating vents. Allow adequate ventilation on all sides of the unit including the bottom.

Hardware Requirements

The Activator 2 programmers may be connected to the SCSI port of 386/486 Pentium PC, Sun, and HP700 workstations. For Sun and HP700 workstations, use the built-in SCSI port; a SCSI card is included with Activator 2 programmers connected to PCs and Apollo workstations. For these two types of computers, the Activator 2 will only operate with the supplied SCSI card. Other types of SCSI cards cannot be used to program Actel devices. Refer to the appropriate *APS 2 FPGA Programming System User's Guide* for more information regarding RAM and hard disk requirements.

Activator 2 programmers may program up to four identical devices simultaneously. Each programmer has four adapter ports, which accept adapter modules for all available ACT device packages. There are different adapter modules for each package of each ACT family. Thus, there are different programming modules for ACT 1, ACT 2, and ACT 3 devices in the 84PLCC package.

Software and Design Requirements

The **aps2** programming software facilitates communication between the Activator 2 and the host computer. This software is included with every Actel software system. The *APS 2 FPGA Programming System User's Guide* contains complete instructions for its installation and operation.

Many files are created for each design throughout the Designer Series software design flow. Only two of these files are required for programming devices with the Activator 2 programmer and **aps2** programming software, <design name>.def and <design name>.fus. To archive or transfer the design to another host computer *for programming purposes only*, only the .def and .fus files are required. No other Designer Series programs may be executed without the balance of the design files. In a similar situation, design files updated from previous versions of Designer Series software using the "**als -update**" command may only be used for programming devices. All other Designer Series functions are disabled by this program. To update design files for other Designer Series functions, use the "**als -convert**" program.

Timing Violation Description

Introduction

Storage elements such as flip-flops and latches are important components in digital designs. These elements are different from basic digital logic because latch and flip-flops contain feedback circuitry. Whenever a feedback condition exists, oscillations are possible. When oscillations occur in digital storage elements, it is referred to as *metastability*. Timing information, such as setup, hold, and minimum pulse widths, is specified to eliminate the possibility of a metastable condition.

In order to assist the designer, simulation tools have the ability to recognize a timing violation and then report the violation to the user. The designer can then analyze the problem circuit to see if the timing violations should be corrected. The purpose of this application note is to describe the various types of timing violations that are reported by simulators and their possible consequences if not corrected.

Types of Timing Violations

There are six main types of timing specifications that are defined in simulation models:

- Input data setup
- Input data hold
- Asynchronous recovery
- Asynchronous hold
- Minimum clock pulse width
- Minimum asynchronous pulse width

Figure 1 is used in the following sections to assist in describing the various timing errors. The figure illustrates the gate-level implementation of a latch using transmission gates and a two-input AND gate to create an active low asynchronous clear function. The signal names are as follows:

- G Gate/Clock
- D Data input
- CLR_ Asynchronous clear
- Q Output
- X, Y Internal nodes

All timing violations with respect to the G input will be referenced on an edge (falling for this circuit), so the explanations defined can be readily extended to flip-flop models.

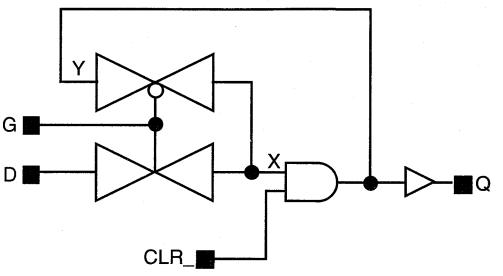


Figure 1 • Gate Level Implementation of a Latch

Data Setup and Hold Violations

The input data value (D) changing close to a gate transition (G falling) creates the potential for a setup or hold violation. For the circuit in Figure 1 to be stable, the new data value must propagate from D to X to Y before the gate function closes the feedback loop. If this does not occur, then both a high and a low (for example, a high at X while Y is still low) condition can exist in the feedback loop. If G transitions low and closes the feedback loop at this point, then an oscillation at the output Q will occur. This feedback delay path essentially creates a minimum pulse width for data that is referenced to clock edge to be stable. Two conditions are possible and are shown in Figure 2. The first condition, D1, has a positive setup time and a 0.0 ns hold. The second condition, D2, has both a positive setup and a positive hold.

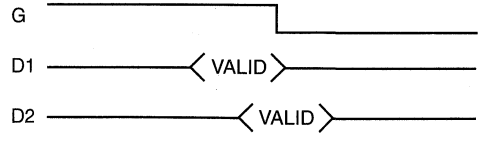


Figure 2 • Data Input Setup and Hold

Simulators can report only violations that have positive values. For example, the D1 case for Figure 2 can have only setup violations because the hold time is not greater than 0.0 ns. The second case, D2, can have both setup and hold time violations since both values are positive.

10

When a simulator encounters a data setup or hold time violation, the simulator will report the violation on the data input pin for simple flip-flops. For flip-flops with multiple data input pins (for example, DFM), the simulation model will consist of a gate feeding a single data input flip-flop as shown in Figure 1. The timing violation information is assigned to the data input pin on the flip-flop. The gate is assigned a delay of 0.0 ns. Therefore, the actual input pin causing the violation must be determined by analyzing the waveforms of each input pin (A, B, and S).

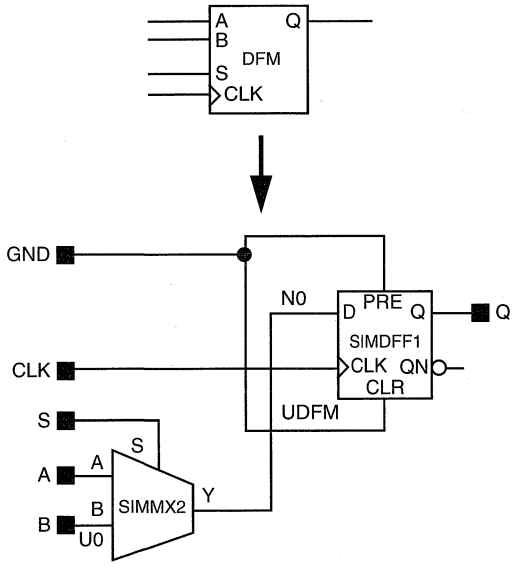


Figure 3 • Simulation Modeling for Multiple Data Input Flip-Flops

Asynchronous Recovery and Hold Times

Referring to Figure 1, if G is high, D is high, and CLR_ is low, then X is high and Y and Q are both low. The feedback loop has different values and could potentially oscillate if the feedback loop were closed (G transitions low) at the same time that the CLR_ input is driven high. Asynchronous recovery and hold times are defined to guarantee correct circuit operation and to prevent oscillations for the asynchronous input from going inactive relative to the transition of the clock input.

To guarantee that the asynchronous control is disabled prior to a clock transition (enabling the D value to pass to Q), the asynchronous recovery time must be met as illustrated in Figure 4. The CLR_ signal must first go inactive (high) long enough before G transitions (high to low) to guarantee that Y is charged high. Once this is done, correct operation (no oscillation) can be guaranteed and the value of D will be passed to Q.

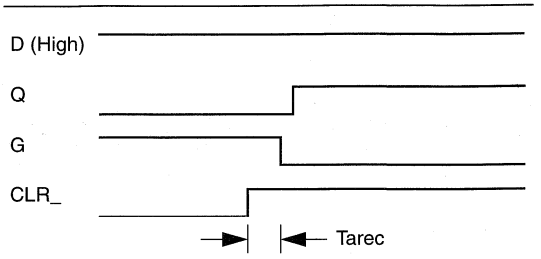


Figure 4 • Asynchronous Recovery (Tarec)—time prior to clock edge that CLR_ must go inactive to guarantee the D input will be passed to the output Q

If the asynchronous control is required to be active during a clock transition (keeping the output Q forced low for Figure 1), then the asynchronous hold time must be met. The CLR_ signal must be held low long enough to guarantee that the low value at Y is passed to X once the G input transitions low, as shown in Figure 5. Once this is done, the CLR_ input can be driven inactive (high) and the circuit will not oscillate. The output Q will remain low regardless of the value at D.

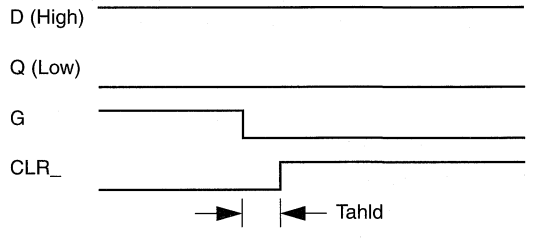


Figure 5 • Asynchronous Hold (Tahld)—time after the clock edge that CLR_ must be held active to guarantee the asynchronous function will be preserved at the output Q

Minimum Pulse Width

Both the G (or clock) input and the CLR_ input are subject to minimum pulse width values. The minimum pulse width is defined to guarantee that when the state of the feedback loop is being changed because of a transition on the G (going high) or CLR_ (going low) input, there is enough time for the value to propagate through the entire feedback loop. Refer to Figure 1 as you read the following sections.

Minimum Clock Pulse Width

Initial State: G low, Q low, D high, CLR_ high

If G transitions high, then the value at D must propagate to X and then to Y before G again transitions low, in order to eliminate the chance for oscillations. The minimum time that G can be high to guarantee correct circuit operation is referred to as the *minimum clock pulse width*.

Minimum Asynchronous Pulse Width

Initial State: Q high, G low, D high, CLR_ high

Initially, X and Y are charged high. If CLR_ transitions low, then a low value begins to propagate to Y and then to X. The CLR_ signal must be held low long enough for X to charge high; otherwise, oscillations may occur. This minimum period that CLR_ must be held low is referred to as the *minimum asynchronous pulse width*.

Interpreting Data Book Timing Information

Again referring to Figure 1, if the internal feedback path is hard-wired, then the timing parameters for a given type of sequential element should always be the same and should be relatively small (approximately 1.0 ns or less). Some sequential elements in the Actel libraries use routed feedback paths that have longer timing values for data setup, asynchronous recovery, and minimum pulse widths that can vary from one instance to the next. The following sections describe how hard-wired and routed flip-flops can be distinguished.

ACT 1

All feedback paths in ACT 1 are routed. Therefore, each flip-flop/latch will have a unique set of timing parameters for data setup, asynchronous recovery, and minimum pulse widths, Refer to the data book for the actual timing values for ACT 1 sequential elements.

ACT 2 and ACT 3

ACT 2 and ACT 3 support flip-flops and latches that have been routed and hard-wired feedback paths. Table 1 defines the feedback path implementation for a given module type. The hard-wired values are defined in the data book, but the routed delays are not. The routed modules should have timing parameters similar to ACT 1 delays; however, the variability can be much greater than for ACT 1. The recommendation is to always use S-type sequential modules when possible to minimize the impact flip-flop timing parameters have on performance.

Table 1 • *ACT 2 and ACT 3 Feedback Path Description*

Element	Module Type	Feedback Path
Flip-Flop	S	Hard-wired
	SC	Hard-wired
	CC	Routed
Latch	S	Hard-wired
	C	Routed

Conclusion

Timing parameters for sequential elements have been defined to guarantee correct operation. When these timing parameters are violated, metastable conditions can occur. Simulators can report timing violations to the user so that the problem can be analyzed and corrected if necessary.

Verilog Backannotation Simulation

Once a design is placed and routed using Actel Designer Series software, the next important thing in the design flow is to run post-layout simulation. The post-layout simulation (also referred to as *backannotation simulation*) uses all the delay information based on the place and route results, and it reflects the different operation conditions (voltage, temperature, and process). Previously, the user needed to use a special PLI routine ($als_add_delays is a special Verilog function provided by Actel) to backannotate delays into Verilog simulation. The disadvantage of using the PLI routine in backannotation simulation for Actel parts is that the user needs to use Actel-modified Verilog executables and therefore loses the flexibility to use new releases of the Verilog-XL executables provided directly by Cadence Design Systems. Another disadvantage is that it's not convenient to run board-level simulation or to combine the Actel PLI routine with other PLI routines created by other vendors. Now, Actel provides the new Verilog simulation models that support the SDF format. (SDF stands for Standard Delay File.) Backannotation simulation can be run using the Verilog SDF annotator ($sdf_annotate) which reads the SDF file, and no special PLI routine is needed. Users can use the new releases of the Verilog executables directly from Cadence Design Systems. For more detailed information, please refer to the *Composer Verilog Environment Getting Started* manual.

Generating the SDF File for the Post-layout Simulation

Actel Design Series software provides the *sdfgen* program to generate the Standard Delay File. To backannotate the propagation delays of each module and the wire delays to Verilog simulation, use the **sdfgen** command at the command line. This command backannotates scaled delays for temperature, voltage, process, and speed grade. These variables are either selected by the **export** command or set as an argument when invoking the **sdfgen** command.

From the project directory, type

sdfgen [tempr:<temperature_range>]
[voltr: <voltage_range>]
[speed_grade:<speed_grade>]
<design_name>

where the valid values for temperature_range are ind, mil, and com for industry, military, and commercial conditions; the valid values for voltage_range are ind, mil, and com; the valid values for speed_grade are std, –1, and –2.

For example, if you specified ACT 1 family backannotation values of commercial temperature range, commercial voltage range, and the fast –2 speed grade, the **sdfgen** command would look like this:

```
sdfgen tempr:com voltr:com speed_grade:-2
uart1
```

where uart1 is the design_name.

A user can also add this **sdfgen** command to a script file, and this command should be executed after the place, route, and extract functions.

Preparing a Stimulus File for Backannotated Simulation

To perform post-layout simulation using Verilog, add the SDF system task ($sdf_annotate) to the stimulus file. The syntax for this task is shown in Figure 1.

```
initial
$sdf_annotate ("design_name.sdf", instance_name);
```

An example of the *$sdf_annotate* construct is shown below.

```
`timescale 1ns/100ps
//*************************************************
//This is a stimulus file
//*************************************************
module test;

reg[15:0] load_data;
reg clk, reset, load, enable;

wire [15:0] prng_data;
wire zero;

// Instantiation of a user-defined module block1
// The instance name of block1 is U0
//To run simulation, you should have Verilog gate-level
//description codes for block1

block1 U0(clk,reset,enable,load,load_data,prng_data,zero);

//*****************************************************
//Add the $sdf_annotate task for backannotated simulation
//The syntax is:
//$sdf_annotate("design_name.sdf", instance_name);
//Before running backannotated simulation, you need to run
//place, route, and extract on block1, then use sdfgen
//command to get the block1.sdf file
//*****************************************************

initial
$sdf_annotate("block1.sdf",U0);

//In this example, block1 is the design name and U0 is its
//instance name

//add test patterns

always #50 clk= ~clk;

initial
$gr_waves("data",load_data, "clk", clk "load", load,
"enable",enable,"reset",reset,"out",prng_data,
"zero",zero,"clkin", U0.clk);
initial
begin
clk=0;
reset =1;
enable = 0;
load = 0;
load_data = 16'h5555;
#100 reset = 0;
#100 reset = 1;
#500 enable = 1;
#10000 load =1;
#100 load = 0;
#2000 $stop;

end

endmodule
```

Figure 1 • Syntax for Post-Layout Simulation using Verilog

To invoke the Verilog simulator using maximum delays, type the following command at your project directory:

```
verilog -f <stimulation_command_file> +maxde-
lays
```

Maximum delay is used for worst case simulation.

The commands for minimum delays and typical delays are

```
verilog -f <stimulation_command_file> +mind-
elays
verilog -f <stimulation_command_file> +typde-
lays
```

It is always a good idea to run a functional simulation to test the functions of the circuit. A functional simulation usually can be run before running place and route. Unit delays are normally used for functional simulation, where a 1-unit delay is assumed for propagation delays of all gates, and no wire delay is calculated. To run a unit-delay simulation, you need to comment out the $sdf_annotate task in the stimulus file, and then type

```
verilog -f <stimulation_command_file>
+delay_mode_unit
```

The example of a simulation command file for an ACT 2 design is show below:

```
block1.v
test.v
-y  /usr/als/lib/vlog/act2
+libext+.v
-a
```

Note that this example assumes the Designer Series software is installed in the /usr directory, so the ACT 2 Verilog libraries are in the /usr/als/lib/vlog/act2 directory. You should replace this path with the actual path of the Verilog library.

Setup and Hold Time Analysis Using the Actel Timer

Introduction

Both gate array and FPGA designs are susceptible to race conditions, which require careful analysis of setup and hold times, and clock skew across best-case and worst-case operating conditions. This application note describes how to use the Actel Timer to analyze accurately these types of potential timing problems. The Timer is a powerful static timing analysis tool that can be used successfully to check setup and hold times and clock skew.

Since gate array devices are not production tested for setup and hold times, these parameters must be sufficiently guardbanded to guarantee they will never cause a failure. This is difficult when using backannotated timing simulation since simulation software does not allow best-case and worst-case timing analyses at the same time. Often such analysis is done by hand, if at all. In some cases, designers simply switch their data with the inactive edge of the clock to avoid such timing problems. For Actel FPGAs, the analysis can be done with the Actel Timer and Timer scripts.

Note: *This procedure does not apply to internal setup and hold times with respect to global clock nets.*

Actel Timer Setup and Data Gathering

This application note assumes a basic working knowledge of the Actel Timer and will only apply to ALS 2.3.x and Designer 3.0. Later releases will automatically calculate setup and hold times. For a more detailed discussion on how to use the Timer, refer to the *Designer Series User's Guide*.

In most cases, the Timer is used to report the maximum delays between two points. But when calculating setup and hold times, it is necessary to know both minimum and maximum path delays.

Carefully execute the following steps for gathering the necessary data to calculate setup and hold times. These steps can be automated in a Timer script. An example is provided in the Timer script section.

1. Invoke the Timer with worst-case operating conditions for temperature, voltage, and process. Select the actual speed grade for the target system. If the target system is designed to work with the standard speed grade device, then set the speed grade to Std in the submenu.

   The ACT 3 family requires special handling to select the speed grade. For an ACT 3 device, speed grade selection is done during the die selection process. For example, to select a –1 speed grade for the A14100A, the A14100A-1 die must be selected.

   The worst-case commercial operating conditions can be found in Table 1.

2. Establish a Start Set that contains the clock pin (i.e., Clk:Pad for an external clock), and specify the End Set to use the default set Clock with the command End Set Clock. The Clock set contains all the clockable pins for the design. Use the Longest command to obtain the worst-case delays for the clock net.

3. Similarly, establish a Start Set that contains all the data pads, and specify the End Set to use the default set Gated with the command End Set Gated. The Gated set contains the data input pins for all the sequential elements. Use the Longest command to obtain the worst-case delays for the data paths.

4. The Shortest command is similar to the Longest command and is used to obtain the best-case delays at the selected operating condition. Repeat steps 2 and 3 with Shortest substituted for Longest to obtain the best-case delays.

5. Exit the Timer, and reinvoke it with the best-case operating conditions for temperature, voltage, and process then repeat steps 1 through 4. The best-case commercial operating conditions can be found in Table 1.

Table 1 • Timer Parameters for Setup and Hold Analysis

Operating Conditions	Temp	Voltage	Speed Grade	Process
Best Case	0°C	5.25	Actual	Best
Worst Case	70°C	4.75	Actual	Worst

Setup Time Calculation

The goal is to guarantee that the slowest data path arrives early enough, ahead of the fastest possible clock. To calculate the minimum required setup time, subtract the fastest clock from its slowest corresponding data path at both best-case

and worst-case operating conditions. The delay obtained from the Shortest command represents the fastest clock delay. The slowest data path delay is obtained from the Longest command and includes an internal setup time for the flip-flop being analyzed. To view the individual delays for each level of logic in the data path, the flip-flop's internal setup time, and the edge direction, use the Expand command.

Do this calculation at both best-case and worst-case operating conditions:

$T_{su} = T_{data} - (X * T_{clk})$

where T_{su} is the minimum setup time, T_{data} is the Longest data delay, T_{clk} is the Shortest clock delay, and X is the family multiplier (Table 2)

The most positive T_{su} is the minimum required setup time. A negative setup time implies the data can arrive at the same time as the clock externally and be valid when the clock arrives at the flip-flop internally.

Hold Time Calculation

The fastest data path must be held long enough for the slowest clock to arrive. To calculate the minimum required hold time, subtract the fastest data path from its slowest clock at both best-case and worst-case operating conditions. The delay obtained from the Longest command represents the slowest clock delay. The data path delay obtained from the Shortest command will include an internal hold time for the flip-flop being analyzed. Again, to view the individual delays for each level of logic in the data path, the flip-flop's internal hold time, and the edge direction, use the Expand command.

Do this calculation at both best-case and worst-case operating conditions:

$T_{hd} = T_{clk} - (X * T_{data})$

where T_{hd} is the minimum hold time, T_{clk} is the Longest clock delay, T_{data} is the Shortest delay, and X is the family multiplier (Table 2)

The most positive T_{hd} is the minimum required hold time. A negative hold time implies the data can change at the same time as the clock externally and be retained when the clock arrives at the flip-flop internally.

Family Multipliers

Each family and net type in that family has its own timing characteristics. The multipliers for each family have been characterized and listed in Table 2. These multipliers are specific with respect to the current technology. The ACT 1 multipliers apply to the rev B die with 1.0 micron geometrics. The ACT 2 multipliers apply to the rev A die with 1.0 micron geometrics. And the ACT 3 multipliers apply to the rev A die with 0.8 micron geometrics. Use a 0.70 multiplier for all other die revisions and families. Select the correct multiplier from the table, based on the net type and family.

Table 2 • Family Multipliers by Net Type

Net Type	ACT 1	ACT 2	ACT 3
Combinatorial Data Path	0.70	0.78	0.60
Combinatorial Clock (Inbuf)	0.70	0.78	0.60
Routed Global Clock (Clkbuf)	0.70	0.85	0.70
Hardwired Global Clock (Hclk)			0.85
I/O Clock (IOClkbuf)			0.95

Timer Scripts

The following Timer scripts are examples of how the data gathering procedure can be automated. Each Timer script contains the necessary commands to invoke the timer function. Hence, execute each script from the Timer's top-level window. The design that these example scripts are based on is shown in Figure 1.

Invoke each script from the Timer's top-level window by typing the Loadfile command into the command window. The syntax is

Loadfile 'File_Name'

The cursor is automatically moved to the command window when text is being entered from the keyboard. Each example script sets several variables that should be easy to understand, except the *binproc* variable. The binproc variable is a combination of the speed grade and the process. The possible values for binproc are device dependent, but some examples are described in Table 3. Some device types may not support a –3 Speed Grade at this time.

Table 3 • Possible Binproc Values

Binproc	Speed Grade	Process
b0bc	Standard	Best Case
b1bc	–1	Best Case
b2bc	–2	Best Case
b3bc	–3	Best Case
b0wc	Standard	Worst Case
b1wc	–1	Worst Case
b2wc	–2	Worst Case
b3wc	–3	Worst Case

Timer scripts can be written for the command line Timer too. In general, to modify each example script for the command line Timer, remove all the commands that invoke the Timer and set the variables from the command line by using the Als -set command. To invoke a command line script, redirect the command file into the Timer by using the following syntax:

als -timer 'Design_Name' < 'Command_File'

It is not necessary to set the binproc variable if the command line Timer is being used.

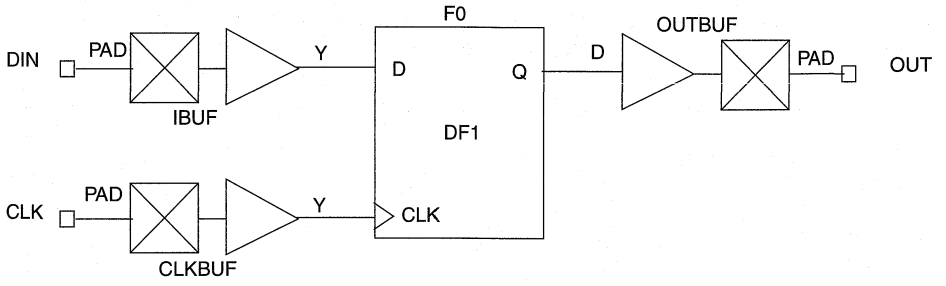


Figure 1 • Timer Script Example Circuit

Example Timer Scripts

Script 1

This script is designed to obtain the worst-case timing data. The output will be saved in a file called <design_name.log> when the script exits the timer.

```
;This script is invoked from the Main menu of the Timer window by using the Loadfile command.
(set "timing" "PostLayout")
(set "temp" "70")
(set "volt" "4.75")
(set "speed" "-2")
(set "proc" "worst")
(set "binproc" "b2wc")
;Here 'b2' refers to the -2 speed grade and 'wc' refers to the worst-case process.
;
;This group of commands invokes the Timer submenu.
(epm-enter "timer-image" 0 0)
(epm-help-text "Initializing Timer...")
(derate (get-default "temp") (get-default "volt") (get-default "binproc"))
(sequence (designstage) (timer) (epm-help-text "Ready"))
;
(emit "\nWORST-CASE PROCESS TIMING ANALYSIS @ 70 C & 4.75V.\n\n")
(startset a)
(addstart clk:pad)
(endset clock)
(longest)
(shortest)
(savestart "start_a.set")
(saveend "end_clk.set")
;
(startset b)
(addstart din:pad)
(endset gated)
(longest)
(shortest)
(savestart "start_b.set")
(saveend "end_gate.set")
(emit "\nWORST-CASE TIMING ANAYLSIS IS COMPLETE.\n")
;
Exit the timer completely at this point.
```

Make sure to save the <design_name.log> file from this pass through the timer; otherwise, it will be overwritten by the next pass.

Script 2

Script 2 takes advantage of the saved Start Set and End Set from the first pass through the timer with script 1. This can save a lot of time. This script is designed to obtain the best-case timing data.

```
;This script is invoked from the Main menu of the Timer window by using the Loadfile command.
;
;Reset the temperature, voltage and process.
(set "timing" "PostLayout")
(set "temp" "0")
(set "volt" "5.25")
(set "speed" "-2")
(set "proc" "best")
(set "binproc" "b2bc")
;Here 'b2' refers to the -2 speed grade again, but 'bc' refers to the best-case process.
;
;Reinvoke the Timer submenu.
(epm-enter "timer-image" 0 0)
(epm-help-text "Initializing Timer...")
(derate (get-default "temp") (get-default "volt") (get-default "binproc"))
(sequence  (designstage) (timer) (epm-help-text "Ready"))
;
(emit "\nBEST-CASE PROCESS TIMING ANALYSIS @ 0 C & 5.25V.\n\n")
(loadfile "start_a.set")
(loadfile "end_clk.set")
(longest)
(shortest)
;
(loadfile "start_b.set")
(loadfile "end_gate.set")
(longest)
(shortest)
(emit "\nBEST-CASE TIMING ANAYLSIS IS COMPLETE.\n")
(sequence (cp-quit) (epm-leave 1))
;
Exit the timer completely at this point.
```

Again, make sure to save the <design_name.log> file from this pass through the timer; otherwise, it will be overwritten by the next pass.

Customer Case Histories

Section 11: Customer Case Histories

Solutions in Time—High Precision TDCs Controlled by Actel FPGAs

Wolfgang Gaubatz, Project Manager ASICs and FPGAs
MSC Vertriebs GmbH

For many years the ASIC design center at MSC Vertriebs GmbH in Germany has been dedicated to solutions for the exact measurement of time. The very broad experience gained in this area has resulted in the development of a series of innovative TDCs (Time to Digital Converters). These parts offer a time resolution down to 60 picoseconds.

In the past, the precise measurement of time was the domain of scientific investigations. The TDCs available on the market were extremely expensive and because of this they were not suited for industrial applications. The TDCs available from MSC now enable demanding measurement tasks to be realized. All TDCs from MSC are purely digital and have been realized in standard CMOS ASICs. The devices consume very low power and can be operated from between 2.7V and 5V power supply. They are therefore ideally suited for battery applications. In addition to the extraordinary performance the outstanding price of under $100 per device compared to alternative solutions of more than $10K make these TDCs very attractive for industrial applications.

Applications

The following represents a list of industrial and physical applications where TDCs are already used:

- Gas and Fluid Flow Measurements
- distance measurement —in particular with Lasers
- different physical experiments in the area of nuclear and high energy physics
- positioning equipment
- high performance scales

Time is one of the most fundamental units of our physical world—TDCs have often been the missing parts in the puzzle of a successful development.

Evaluation Boards

These innovative TDC products are now available, but customers were often reluctant to use these parts for their developments. The concerns were obvious. Do these TDCs really solve our problems? Is the time resolution good enough? Is the power consumption as low as promised? Even when the customer was far away from a solution, they often did not want to spend the time, to evaluate these TDCs.

MSC decided to solve this problem by developing two evaluation boards for the different TDCs. The goal was to develop an evaluation board that would allow a quick and easy characterization of the TDCs without the customer having to spend time building a prototype board. This was seen as an obstacle to sell the TDCs by MSC and by the customer as a barrier to time to market of their end product.

This hurdle has been overcome and the evaluation boards are now available. FPGAs from Actel played a key part in the development of these boards. On the TMS-2001 evaluation board an A1020B is the only higher integrated circuit apart from the TDC. On the TMS-System2 board up to 4 A1010B FPGAs realize the interface between the TDCs and a T425 Transputer on one side and the PC on the other side.

Actel FPGAs, The Only FPGA Technology Guaranteeing Design Security

Actel has been extensively used in the ASIC design center at MSC to prototype complex Gate Arrays. The fine granularity of the multiplexer based logic element allows an efficient use of synthesis tools and a technology independent description of the design. The channeled Gate Array architecture of the Actel FPGA favors these FPGAs from a hardware point of view to prototype Gate Arrays. In the past about 20 ASICs from medium to high complexity have been prototyped with Actel. A considerable know-how concerning the development tools and Actel's FPGAs was gained and made Actel the first choice for the development of the TDC evaluation boards. However all projects require the reevaluation of possibilities and alternatives. Evaluation boards have to be flexible enough to allow individual customization of the board to each user's needs. Therefore discrete logic and Gate Arrays were not suitable to build the logic around the TDCs. In addition the quantities of only a few hundred to thousand pieces would not have justified the implementation of the logic in an additional Gate Array.

Due to the fine granularity of the Actel Antifuse an abundant number of routing reserves are available on any Actel device. This makes Placement and Routing easy and very flexible. On the TMS-2001 evaluation board one A1020B contains the complete logic to implement a PC interface and to allow SW control of the TDC-2001. Although the A1020B is 90% utilized, many customer changes have been implemented successfully and rerouted whilst maintaining the IO fixing, so that no PCB

layout change was necessary. This flexibility is only available with Actel FPGAs and is a direct result of the fine granularity Actel antifuse.

As with any other innovative products, design security is a major concern. On that basis discrete and SRAM based FPGA technologies were not considered as suitable to take the logic of the TMS-2001 and TMS-System2 evaluation board. Only Actel guarantees design security and gives competitors no possibility of copying your high performance design.

In the past Actel has always been chosen as first choice for high performance FPGA designs within MSC. With the different TDCs MSC has launched a range of innovative products on the market with no competition at all. It was our definite goal to keep competition out of our market. Therefore we chose Actel, the only FPGA technology which guarantees design security.

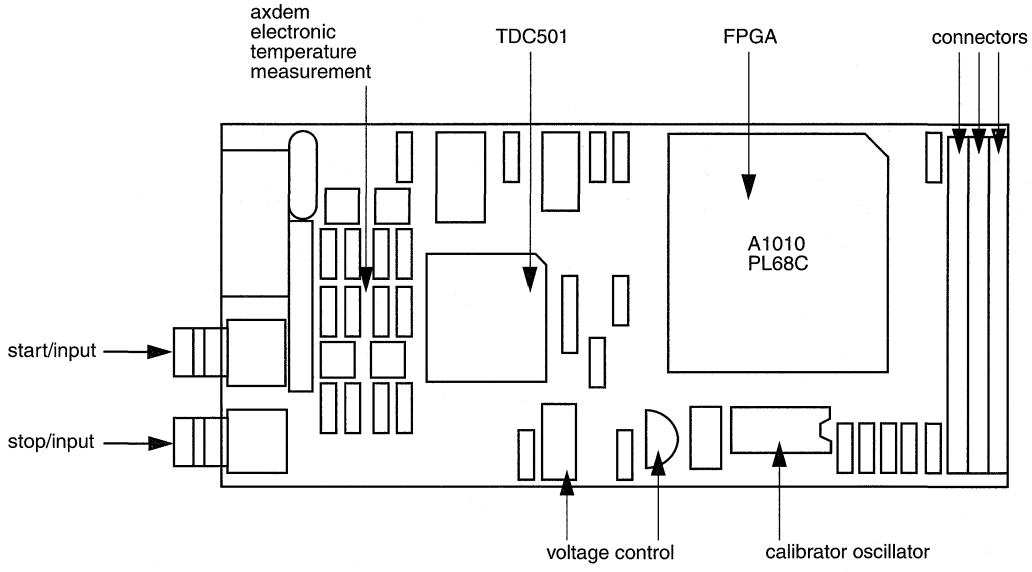


Figure 1 • *The TMM501 module implements the interface between the transputer and the TDC501. This module can be placed four times as add-on module on the TMS motherboard.*

The Design of a Numerical Accelerator Using RISC Processors

Ken Linton, Hardware Engineering Manager
SKY Computers, Inc.

Introduction

This paper describes the design methodology used to develop the SKY Computers, Inc. ShamrockII multiprocessor daughterboard, which heavily uses Actel FPGA technology. The daughterboard is part of the SKYboltII family of products. SKYboltII is a modular product line consisting of a motherboard coupled with one or more daughterboards. The motherboard carries a i960CA system processor, the bus and I/O interfaces, and the SKYchannel high-performance packet bus, which connects the system components. The daughterboard is the arithmetic processing platform. It contains the i860XR processors and system memory.

The ShamrockII incorporates four i860XR arithmetic processors and DRAM memory. One ShamrockII daughterboard can be connected to a standard SKYboltII board, providing a four-processor i860XR board in a convenient 6U VME form factor. The ShamrockII runs at 40 MHz and provides 320 Mflops peak performance. It used the fastest available i860XRs and is aimed at all three of our major market categories: high-performance DSP, array processing, and desktop supercomputing.

The ShamrockII can contain a total of 64 MB of DRAM memory using 16-Mbit DRAM technology. The design gives each i860XR an available linear 64 MB address space. The memory can be viewed as a shared space, or it can be seen as four distinct spaces. Physically, the DRAM is configured as four independent memory banks, each containing 16 MB of memory. Each memory bank is connected to each i860XR and the SKYchannel through a high-speed digital crosspoint switch. This provides each i860XR with the ability to access memory at full bandwidth as long as there is no interprocessor contention.

The ShamrockII is used for applications in which all four processors work on the same problem, therefore making processor-to-processor synchronization becomes necessary. This can be accomplished through shared memory locations, which, however, do not provide optimum performance. A hardware mechanism called *barrier synchronization registers* has been provided for quick processor-to-processor synchronization.

The functional block diagram in Figure 1 shows the main architectural features of the ShamrockII daughterboard.

Implementation Guidelines

The technology selections made for the implementation of the ShamrockII were based on a few design guidelines. The first was that the board would be designed using top-down design methodology, supporting full system-level simulation. Verilog was selected as the Hardware Description Language to be used to meet the goal. Therefore, all component selections would have to support the Verilog environment.

The design was done by first implementing high-level behavioral models to test the system functionality. These models where then refined to structural models, which where instantiated in the simulation environment in place of their behavioral models. The simulation environment supported a mixture of behavioral and structural models. This allowed the development to progress without all structural models' having to be completed simultaneously. Structural model simulations run slower and consume more system resources than do behavioral model simulations, so, simulations could be run more efficiently by using behavioral models for the blocks in which structural models were not required.

Once the structural models were completed and verified it was important to be able to turn the model into a physical part quickly and efficiently. This required a logic synthesis package that was compatible with Verilog. In addition, the timing characteristics of the physical part had to be able to be passed back into Verilog for the timing simulation. For these reasons, Exemplar was the synthesis tool chosen for this design, so the chip technologies chosen had to be supported by Exemplar.

The project time-to-market goals did not allow for ASICs to be included as part of the design strategy, but there were clearly some challenging design blocks that could have benefited from ASIC technology. They were the DRAM controller, the protocol controller, the DMA controller, and the register bank. Each of these designs had the following requirement:

- 40 MHz operation
- PQFP packaging
- High I/O count

x

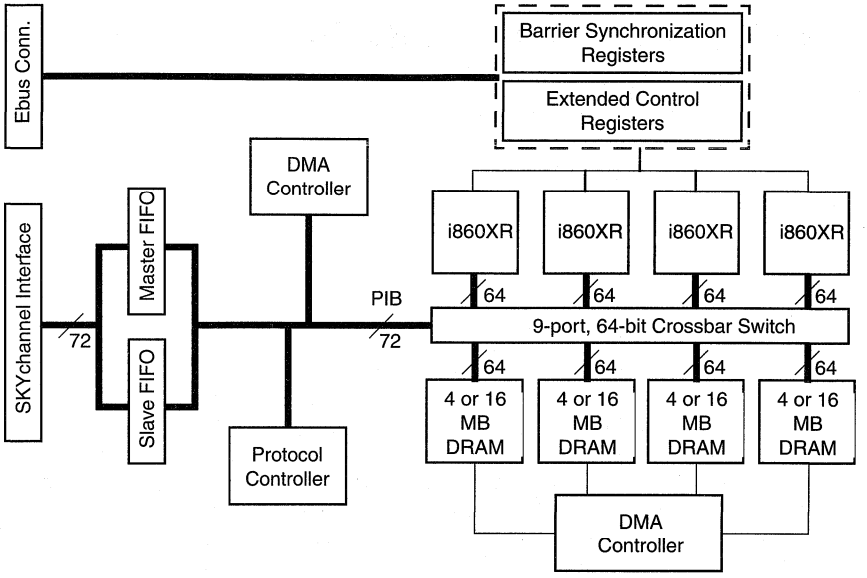


***Figure 1** • ShamrockII Functional Block Diagram*

It was determined that FPGA technology would be used to implement these functions. The available FPGA families were evaluated against all of the above criteria, as well as against some FPGA-specific criteria. The FPGA-specific requirements included the parts being able to support a 40 MHz system clock directly from the synthesis tool—for example, no hand optimization postroute. High gate count utilization had to be achievable directly from the high-level tools. First-route results (pins tied down) also had to be very good. We found the Actel ACT 3 family to meet all of the stated requirements, so we chose the Actel 1460A part to implement all four functions. This part is supplied in a 208-pin PQFP package with 168 available user I/O pins.

The Actel designs were written in Verilog, synthesized with Exemplar's CORE, and then placed and routed in the Designer Advantage tools from Actel. Backannotated timing results were then generated by having the Exemplar tools create a Verilog netlist from the Actel-generated adl netlist. A verilog task supplied from Actel allows all of these elements to be tied together to provide a complete chip-level and board-level simulation solution.

The following sections provide brief descriptions of the four Actel implementations. To illustrate, we have included a Verilog code segment for the register design.

ShamrockII Extension Registers

The ShamrockII defines control registers that reside on the platform. These registers are used to communicate between the processors. Some registers are dedicated to the i960CA, and others are dedicated to the i860 processors. The register chip contains five ports. One is for the i960CA and interfaces to the i960CA bus running at 25 MHz. The other four ports connect to the i860s and interface to the i860 bus at 40 MHz. One of the interesting features of this device is that the i860s all have exactly the same address map, which makes the software efficient; however, each i860 has a unique register set. The register set selected is position dependent and selected by the device itself. This design consumes about 90 percent of the Actel A1460A and uses 145 of the available I/O pins.

The following Verilog code is an implementation of a one-hot-state machine that allows reads from one of the five requesters of data for this chip. One-hot encoding is used to reduce the logic module count and routing required to implement this circuit. At 40 MHz, the following logic could be synthesized and simulated with no timing violations. The state bits are written so as to provide an asynchronous reset condition, with everything else transiting from clk40.

One timing problem did have to be overcome in the first iteration of the design. Note the JKF instantiated at the bottom of the code. A signal was needed to last from one clock after the time p0_read_cyc occurred until a clock after p0st3 occurred. When this signal was originally described, it was made up of an OR function of all the state bits, which created several logic modules and a D-type flip-flop. When the circuit was simulated with backannotated timing, it became apparent that this solution would not work. It was replaced with a JKF, which was instantiated with the noopt flag set on this module for synthesis. The noopt flag keeps synthesis from trying to optimize a solution that you have hand-picked. This circuit then easily performed at 40 MHz.

The following is Exemplar's estimate of the logic modules and delay for the entire design. Pass #2 was used, and it combined to 762 modules (out of 848) in the Designer Advantage router. Exemplar indicates the worst-case delay path to be 50 ns; however, this delay is combinatorial. The actual part is capable of a running at 40 MHz.

DRAM Controller

The ShamrockII memory is implemented using Fast Page Mode DRAMs. This technology provides a fast page access time and is able to maintain the maximum i860XR data rate as long as access falls within the DRAM page. The 64 MB of memory on the card is divided into four 16 MB independent banks. The architecture allows all four banks to be accessed simultaneously by the i860XRs and the SKYchannel. The i860 XRs can access all four banks at the same speed, and there is no additional arbitration penalty for accessing any bank. This allows the i860XRs to communicate at the full 160 MB per second per processor. The DRAM Controller handles the following tasks:

- DRAM refresh
- Access arbitration
- DRAM RAS/CAS control
- Executing atomic cycles

The design consists of four identical state machines, one for each memory bank. There is also a lock arbiter for processing read-modify-write cycles. The part is 95 percent utilized, and it uses 127 of the available I/O pins.

```
/*************************************
READ FROM PROCESSOR ZERO: READ LOGIC
*************************************/
wire p0_read_cyc;
assign p0_read_cyc = (!p0_regcs_ && !p0_wr
&& !p0_read);

always @(posedge clk40 or negedge reset_)

    begin
    if(!reset_)
        begin
        #50 p0st0 = 0;
           p0st1 = 0;
           p0st2 = 0;
           p0st3 = 0;
        end
    else
        begin
           if(p0_read_cyc)
            begin
            #50 p0st0 = 1;
            end
        else if(p0st0)
           begin
           #50 p0st0 = 0;
            p0st1 = 1
           end
        else if(p0st1)
           begin
           #50 p0st1 = 0;
            p0st2 = 1;
           end
        else if(p0st2)
           begin
           #50 p0st2 = 0;
            p0st3 = 1;
           end
        else if(p0st3)
           begin
           #50 p0st3 = 0;
           end
        end
    end

    jkf M20 (
           .J(p0_read_cyc),
           .K(!p0st3 && reset_).
           .CLK(clk40),
           .Q(p0_read)
           );
```

Figure 2 • Verilog Code Fragment

Table 1 • Examplar Logic's CORE

Pass	Area (modules)	Delay (ns)	Comb. Seq. I/O (modules)	Combinable	CPU min:sec
2	764	50	679 213 143	128	13:51
5	759	59	677 212 143	130	20:25
7	754	59	672 212 143	130	14:24

Table 2 • Resource Use Estimate

Design:	register_2
Technology:	act3_noopt
File:	register.v
Area:	764 modules
Critical path:	50 ns
Comb. modules:	670 modules
Seq. modules:	213 modules
I/O modules:	143 modules
Combinable:	128 modules

Table 3 • Device Utilization Report

Device	Combinatorial modules		Sequential modules		I/O modules	
	avail	util	avail	util	avail	util
A1415	96	707%	104	205%	80	179%
A1425	150	453%	160	133%	100	143%
A1440	276	246%	288	74%	140	102%
A1460	416	163%	432	49%	168	85%
A14100	680	100%	697	31%	228	63%

Protocol Controller

The protocol controller is used to implement the SKYchannel packet bus interface. It is a 72-bit-wide bus capable of transferring data at 320 MB per second. The protocol controller is responsible for building header words and assembling packets when the platform is a master and for stripping packet headers when the platform is a slave. This design consumes over 95 percent of the Actel A1460A and uses 161 of the available I/O pins.

DMA Controller

The i860 DMA function allows the i860 to fetch up to 60 MB of data using multiple 2048-byte SKYchannel packets. This provides a high-performance method for retrieving data structures from remote memory. The data is transferred directly into memory without i860 intervention. The design contains several control registers, a 32-bit address generator, and a 26-bit transfer size counter. This chip is 90 percent utilized, and it uses 116 of the available I/O pins.

An Image Compression Application Using Programmable Logic

Peter Symanski, Senior Design Engineer
Kofax Image Products

Introduction

This application note will describe an image processing application using a Field Programmable Gate Array (FPGA) in conjunction with a 32-bit Graphics System Processor intended for document image processing. Detailed results show the performance, utilization, and functionality achievable with FPGAs in these types of applications. Additionally, this note will show techniques and tricks that illustrate successful design techniques using FPGA devices.

System Description

Figure 1 shows a block diagram of the document image processing system. Key features of the system are a 40 MHz Graphics System Processor, real-time 2-D scaling and rotation, support for scanners and printers operating at video rates of 7 MB/second, 32-bit wide data bus, hardware support for raster image color and bit-ordering within a byte and an interrupt storage area (ISA) bus interface. Additionally, the input from the scanner is written into 100 nanoseconds DRAM memory and can be output to a printer.

The system contains eight main functional blocks. The 34094 provides the interface to the PC bus. The 34020 is the main graphics processor and shares the DRAM array with the SCSI-2 module. The SCSI-2 module interfaces the peripheral port along with a DUART and a FIFO. The two custom coprocessors are an image rotator and a compression expansion engine. The SCSI-2 module uses an FPGA to implement a specialized direct memory access (DMA) control function and will be described in more detail below.

SCSI I/O Module

The SCSI I/O module, shown in Figure 2, is a single 2.75-inch by 5-inch card connected to the KF-9275 main board with a 100 pin .05-inch grid, surface-mounted header. The header provides full access to the TMS-34020 bus. On the other end, the card interfaces with the SCSI bus through a high-density SCSI-2 connector.

The module includes a DMA controller, a SCSI-2 controller, and a "glue logic" PLD to interconnect the SCSI controller to the 32-bit TMS-34020 bus. The DMA controller is implemented in a single Actel A1425 FPGA.

The entire module appears within the TMS-34020 address space as a block of sixteen SCSI registers, a block of four 32-bit DMA registers, a terminator disable register, and a 53C94 hard reset register. Through those registers the host can initialize, define transfer mode, and transfer data. Each of the blocks occupies 2 Mwords of address space.

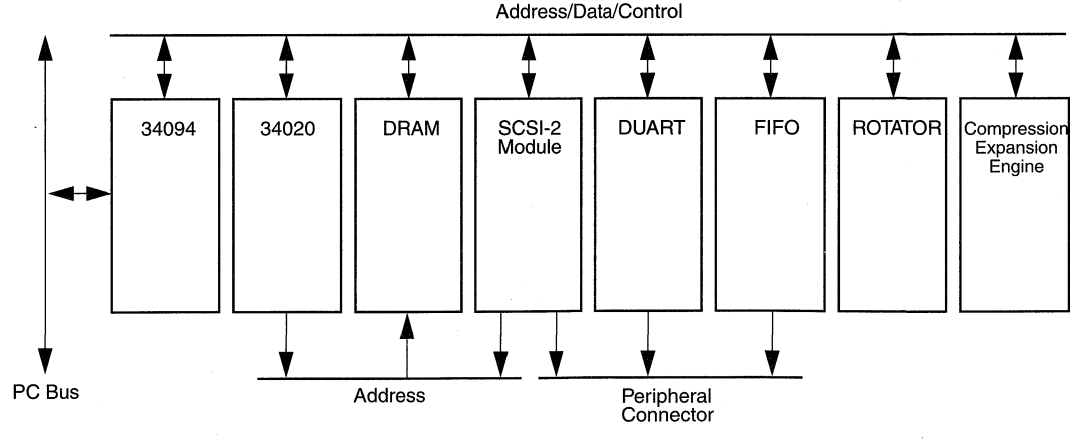


***Figure 1** • System Block Diagram*

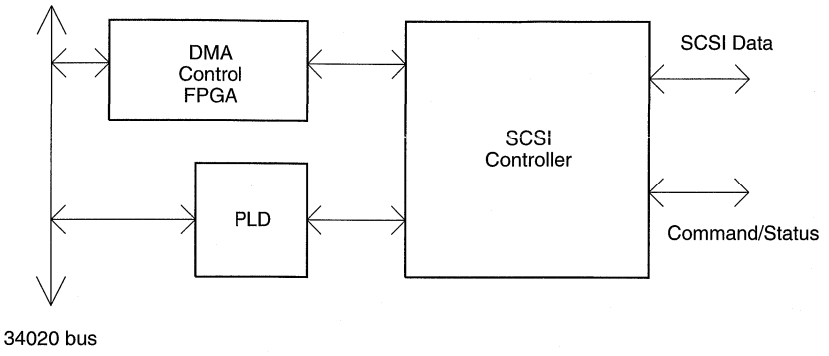


Figure 2 • SCSI I/O Module Block Diagram

DMA Controller Functionality

The DMA consists of a single channel TMS-34020 bus master capable of transferring data blocks up to 256 KB long. It supports "move through DMA" transfers from a 16-bit peripheral to DRAM space 32 bits wide. Built into the data path are "color and sex" imaging functions. The address is formed by combining the 11-bit page register, 16-bit up-counter, and 5-bit fixed bus status bits. Transfers can be up to 64 K double words long, but they cannot cross page boundaries. Both the transfer counter and address generator post decrements and increments. At TC (terminal count), the address generator maintains a valid memory address value; thus, it is not necessary to reload the address register at TC. Residual 16-bit data can be detected in the status register and read out of the data register. The controller also provides TMS-34020 BLAD, RCA, and page mode emulation. The controller has the TC interrupt wired-or with the 53C94 interrupt. Interrupt Service Routines need to determine the source by checking the Interrupt bit of the DMA controller and the Interrupt bit of the SCSI controller. A detailed list of the DMA controller registers follows:

- Source/Destination Address:
  Specifies the source or destination of the next DMA operation to or from the SCSI controller.
- Count:
  Specifies the total number of bytes to transfer.
- Direction:
  Specifies the direction of data flow to or from the SCSI controller.
- Residual Control:
  Manages residual data as a result of a partially full DMA word.
- DMA Control:
  Returns the DMA TC pending interrupt and DMA available bits on a read. A write resets the DMA controller and should be issued on power-up. The DMA controller does not have a hardware reset line.

Description of Main Blocks

The main functional blocks shown in Figure 3 are the Decode and Control Block, the Peripheral Handshake Block, the DMA Counter Block, the Bus Interface Blocks, and the DRAM Interface Block.

DRAM Interface Block

The DRAM Interface Block is a 12-bit, 2:1 multiplexer that selects between the row address strobe (RAS) and column address strobe (CAS) addresses. The Processor Bus Interface Block is a 32-bit, 4:1 multiplexer that provides access to the internal status and control registers.

DMA Counter Block

The DMA Counter Block contains the 16-bit Word Count Register (WCR), the 16-bit Address Count Register (ACR), the four Control bits (Sex, Color, Interrupt, and Go), and the 11-bit Page Register (PR). The PR and ACR provide the address to the DRAM Interface Block. The WCR decrements as each transfer occurs to determine the end of a transfer.

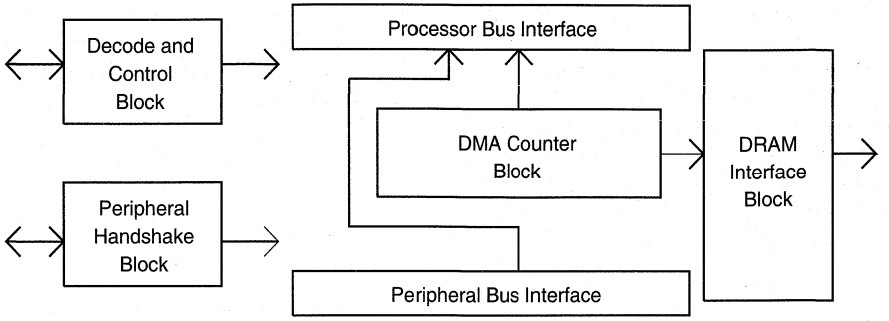


Figure 3 • DMA Controller Block Diagram

Decode and Control Block

The Decode and Control Block provides the timing and control of the DMA and DRAM sections. A more detailed block diagram of this section is shown in Figures 4 and 5. Figure 4 shows the Address Decode Logic and Figure 5 shows the Master Timing Generator (MTG) state machine. The Address Decode and Register Select logic is used to determine the register addressed during a read or write cycle. The MTG state machine is the master timing generator. It produces timing signals for transfers to and from the processor bus, as well as DRAM writes.

Address Decode Logic

The Address Latch signal (ALTCHI) is used to store the result of the decode (in the CS register) and the A5 and A6 address signals (in the A5 and A6 registers). These signals are then used to load the data from the bus into the correct register using LD_A or LD_C. (Notice the use of /PWE to ensure that access is a processor write and /RAS to ensure the correct timing of the transfer on the LD_A and LD_C signals. These signals are generated by the MTG state machine described in the next section.)

OM0 and OM1 control the Output Multiplexer and determine the register to place on the bus during a processor read. Bit ordering (Sex) is controlled by switching the lower multiplexer bit (Om0) based on the contents of the Sex Control Register (SCR). Timing is controlled by the BGRANT and A_D signals provided by the MTG state machine.

MTG State Machine Logic

The MTG state machine, in Figure 5, is implemented using four state bits to encode eight states. This technique makes excellent use of the register-rich architecture of FPGAs. In this implementation, a simple feedback shift register is used to generate the required state sequence. The full sequential module of the ACT 3 device is used to implement each state bit. (The logic diagram for the sequential module is shown in Figure 6. Notice that it is basically a 4:1 multiplexer with two-input gates driving the select lines.)

The MTG timing sequence begins with BREQ going Low. This sets Q0 and begins shifting a logic High through the shift register. The High stops at register Q1 until BGRANT goes Low. Once BGRANT goes Low, Q1 is loaded with the High from Q0. This sequence continues to set Q2 and Q3. Once Q3 is set, the select line on Q0 is switched to source a logic Low, loading a Low into Q0. The Low propagates through Q1, Q2, and Q3 until the all Low Idle state is reached. Timing signals RAS0, CAS0, RAS_CAS, CCLK, and A_D are all decoded from the state outputs. The sequence is shown in the state table in Figure 5.

Notice that this technique for state machine implementation produces 2N states with N state bits. The "one-hot" technique only produces N states with N state bits. For simple state machines with four or more states, the feedback shift register implementation is usually superior.

DRAM Control Logic

The control of the DRAM is accomplished by first arbitrating for the address bus, issuing a RAS, then switching the address (with the RAS_CAS signal), and then issuing a CAS. Timing on RAS and CAS is preserved by using bidirectional buses on the RAS and CAS signals, a common technique when interfacing DRAMs. Figure 7 illustrates this technique.

Peripheral Handshake Block

The Peripheral Handshake Block controls the access to the PD bus. A simple feedback shift register state machine is used to generate the Peripheral Data Available (PD_Avail) signal, the Data Acknowledge (DACK) signal, and the Enable Data (ENA_D) signal to load data from the SCSI bus. The details of the design of this block will not be presented, since the design technique is similar to the MTG state machine.

Implementation

The DMA controller was implemented in a single Actel A1425 FPGA. The design was entered using Viewlogic schematic capture and was placed and routed using the Actel automatic place and route tools. The device operates at 40 MHz. The design required 290 of the 310 logic modules and uses all 100 IOs. The entire design required only three weeks to implement and another six to simulate. The first programmed part was functional and the third was released to production

Conclusion

This application note describes the design of a Document Image Processing engine using an FPGA as a specialized DMA controller. The controller operates at 40 MHz and uses a single A1425 device.

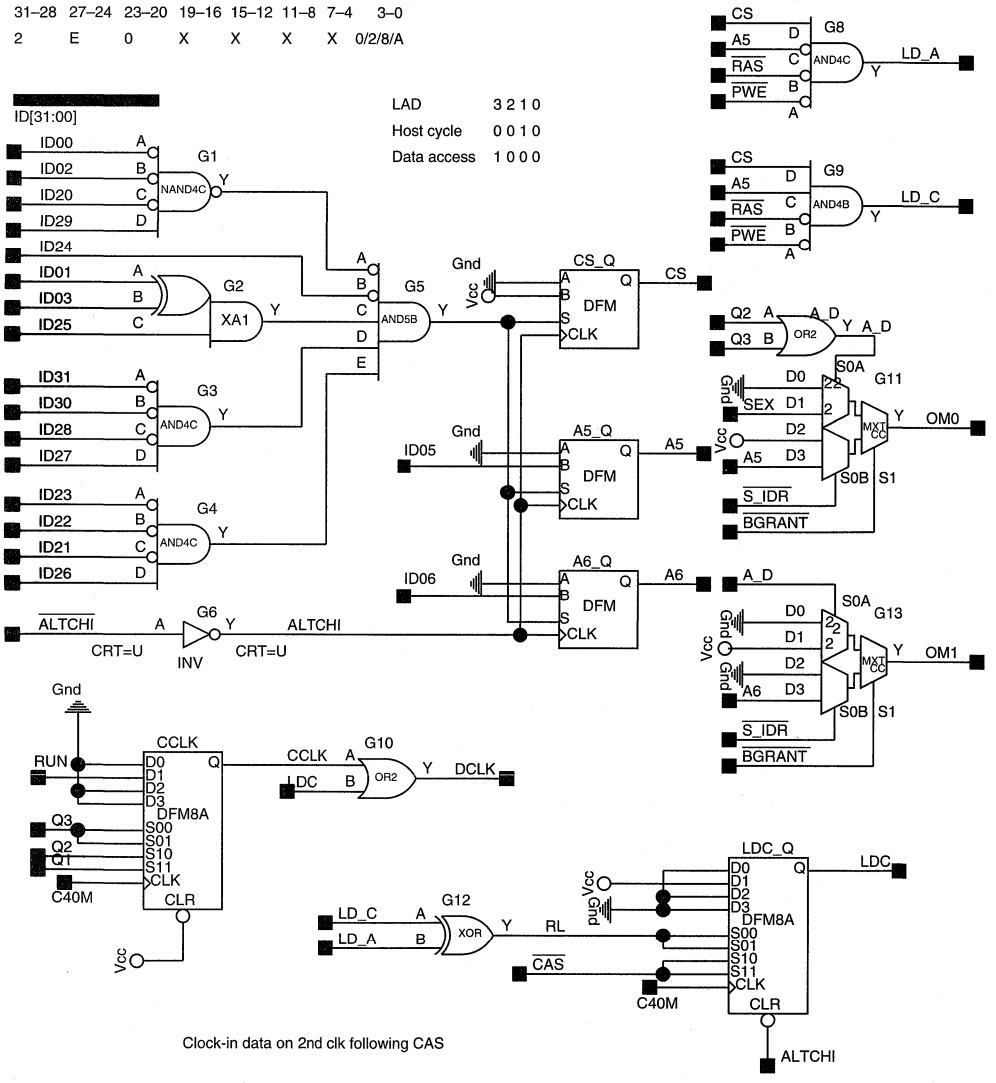
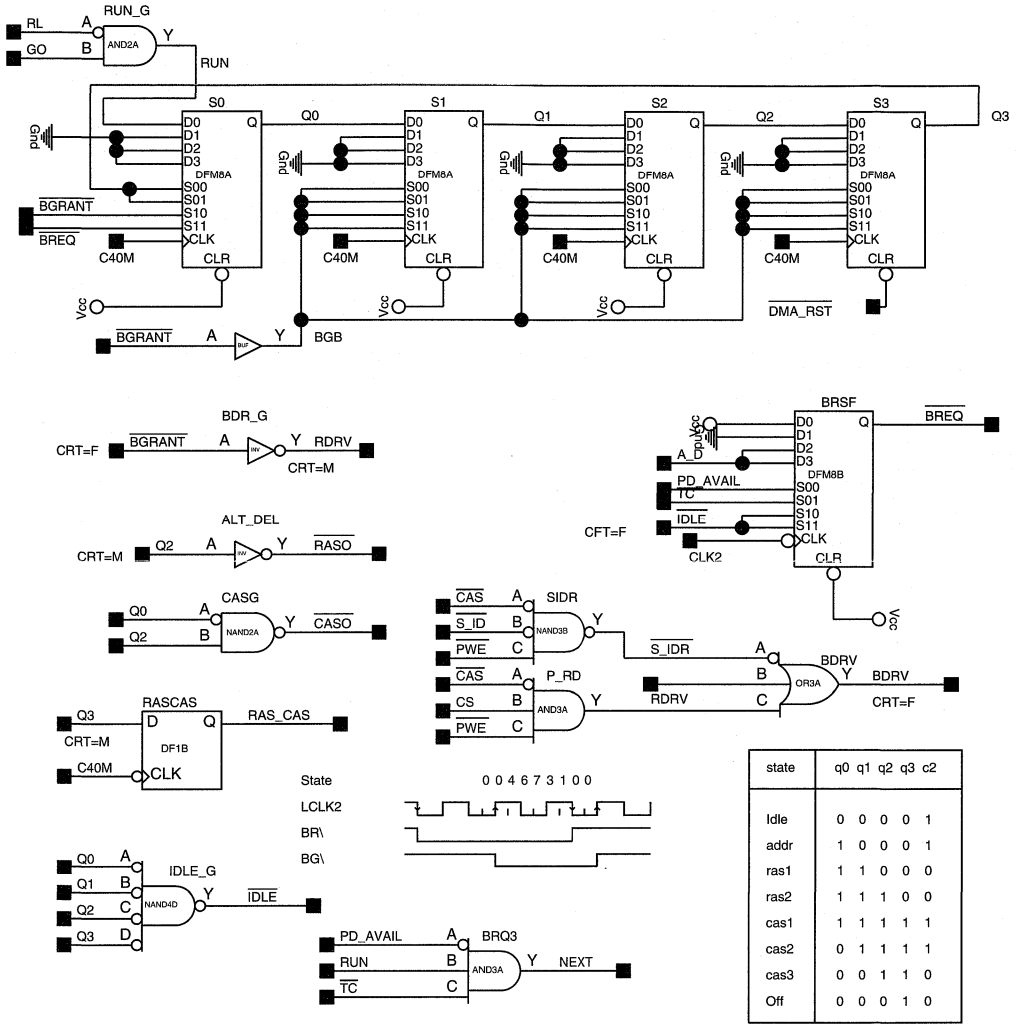


Figure 4 • Address Decode Logic



state	q0	q1	q2	q3	c2
Idle	0	0	0	0	1
addr	1	0	0	0	1
ras1	1	1	0	0	0
ras2	1	1	1	0	0
cas1	1	1	1	1	1
cas2	0	1	1	1	1
cas3	0	0	1	1	0
Off	0	0	0	1	0

Figure 5 • MTG State Machine

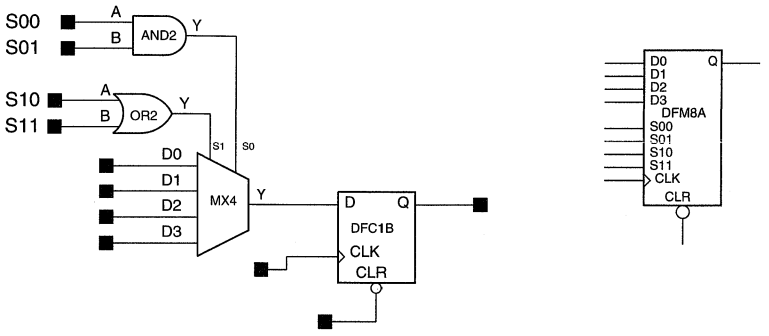


Figure 6 • S-module Logic Definition

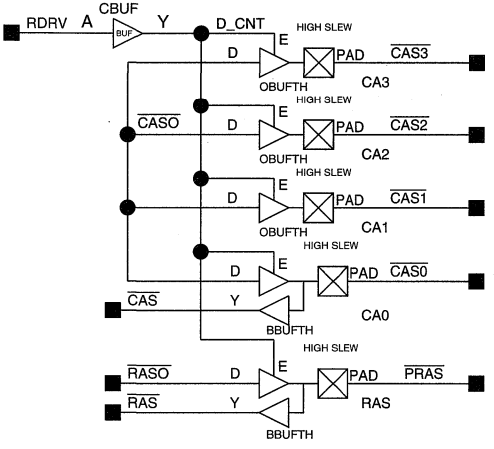


Figure 7 • DRAM Signal Generation

A Finite Impulse Response Filter for E3 Using Field Programmable Gate Arrays

David Coble, Manager, Digital Signal Processing
Microwave Radio

Introduction

A typical digital microwave line-of-sight system transmits the combined data of several low speed channels using a modem and an RF transmitter. The receiving component demodulates the output of the RF receiver and demultiplexes the data into the original low-speed channels. This paper will describe the implementation of the digital microwave radio's modem, and in particular the modem's Finite Impulse Response (FIR) filter using a single Actel A1460 Field Programmable Gate Array (FPGA).

System Description

The system block diagram of the digital microwave radio is shown in Figure 1. The four-level Frequency Shift Keying (FSK) modem encodes the serial bit stream into a two-bit parallel bit stream with each of the two bits representing a level. Each level is a two-bit symbol, or dibit. In most communications systems, multibit symbols are used to allow more information to pass smaller less channel bandwidth. Many separate channels must be used in close proximity to each other without interference and with maximum throughput. Limiting the bandwidth for these purposes causes a phenomenon known as the Intersymbol Interference (ISI). As channel bandwidth is decreased or more information is squeezed into a given bandwidth for higher channel efficiencies, ISI gets worse. Figure 2 shows an example of a smooth voltage transition input to a narrow bandwidth channel and the expected output with ISI imposed. The ringing shown in the output preceding and succeeding the transition is ISI when the ringing merges into the sample points of both the preceding and succeeding symbols.

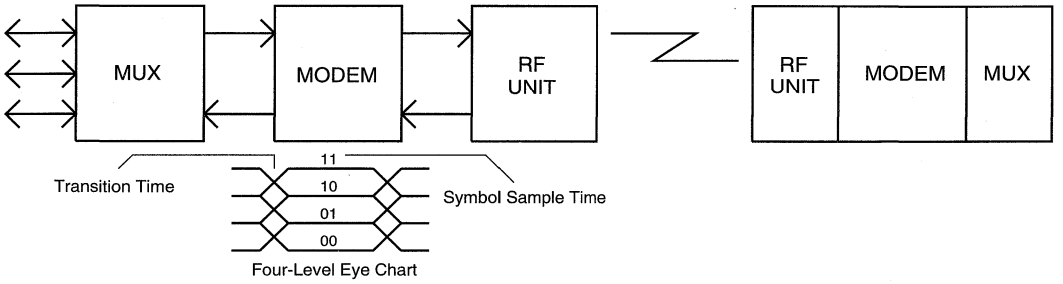


Figure 1 • *System Level Block Diagram*

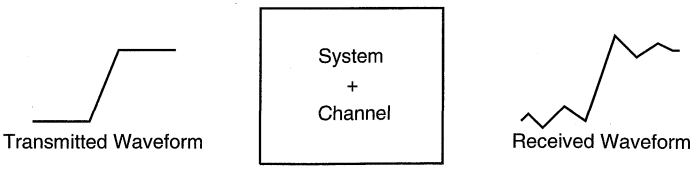


Figure 2 • *Sample Channel Transmission Function*

The purpose of an FIR filter is to minimize the ISI effects by using weighted coefficients on the preceding and succeeding symbol values to nullify the effects on the symbol being received. Telephone line modems employ FIR filters written in software to counteract ISI in their receivers. They must overcome the effects of the channel characteristics of the phone line. These FIR filters actually train and adapt the coefficients to the particular phone line.

In line-of-sight digital microwave radio applications, the symbol rates are typically 1 to 30 megacycles per second in multiplexed T3 or E3 carrier systems worldwide. These symbol rates exclude the use of software-driven DSP techniques. Adaptive FIR filters are not needed because the channel characteristics do not change and System Transfer Function makes the radio system's characteristics well known. These facts and the following allow the FIR filter to be implemented in the transmit circuitry to "predistort" the signal in the transmit process because the sender "knows" what the symbols are instead of processing the noisy signals in the receiver.

Filter Design

The FIR filter is implemented in a single Actel A1460 FPGA as shown in Figure 3. The signal is a two-bit value and is shifted into the device using the system clock. A shift register stores each time value so that the filter can compute the output as a function of the previous eight inputs. Each sample is multiplied by a filter coefficient and then added together to determine the final output value. The major blocks are thus the shift register, the constant multipliers, and the adders.

Shift Register Block

The shift register is an eight-word delay element that holds the previous eight time values of the input data. The register is clocked at the system clock rate and is implemented with eight two-bit D-type registers connected serially. In the top-level schematic, the number of clock delays is indicated in the signal name: D3D1 is the third delay (D3) of the input signals MSB (D1). The intermediate values are all available for multiplication by the Constant Multiplier Blocks.

Constant Multiplier Blocks

The Constant Multiplier Blocks multiple each of the eight delayed two-bit input values by the required filter constant. Each multiplier takes the two-bit value and creates an eight-bit twos complement result using the necessary filter coefficient. The multiplier is implemented using 2:1 multiplexers that deliver the correct eight-bit constant using the two-bit data values. An example of a Constant Multiplier Block is shown in Figure 4A. The values for each of the four data values are shown in Table 1. These values are then wired to the multiplexer data inputs so that the correct result is selected based on the two-bit input data value. This approach is particularly efficient in the Actel architecture, since the 4:1 mux is accomplished in a single logic module. Furthermore, the logic can be combined with any register so that the entire multiplier and pipeline register requires only a single logic module. Multipliers 1 through 4 use this technique.

Table 1 • Constant Generation

Dibit	Output Value	Twos Complement
00	+102	01100110
01	+34	00100010
10	–34	11011110
11	–102	10011010

Another technique is used in multipliers 5 and 6. In these blocks, two symbols are provided to the block. The symbols are multiplied by the correct coefficient and then added together. This can be implemented using 16:1 multiplexers instead of using an adder as in the previous implementation. It is shown in Figure 4B. The implementation can be selected based on the number of logic modules required to minimize the total number.

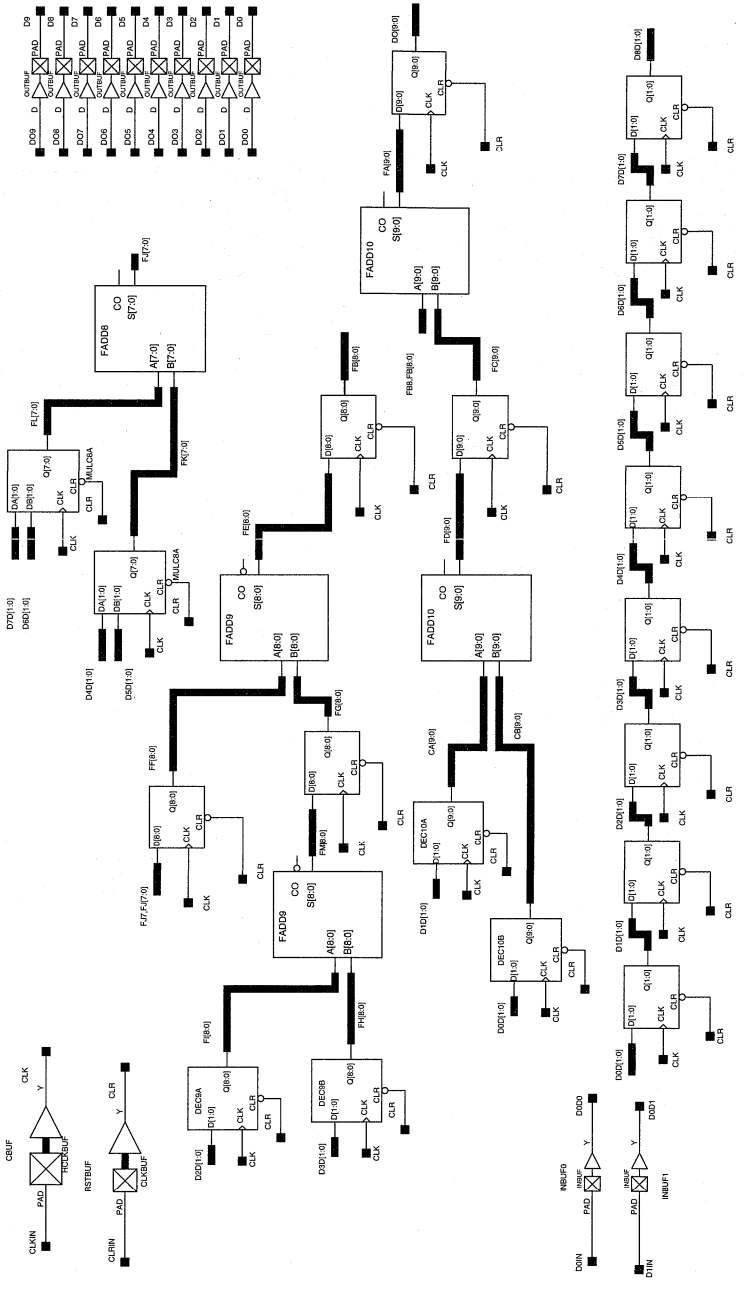
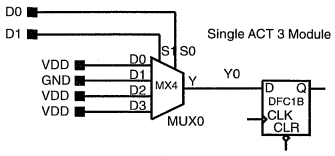
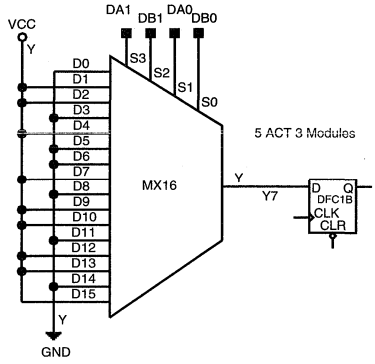


Figure 3 • Filter Block Diagram



(A). One of eight for eight-bit result



(B). One of eight for eight-bit result

Figure 4 • Constant Multiplier Block

Adder Blocks

The adder blocks use the standard fast adder macro available in the Actel library. The adders require three logic levels; the last logic level can be combined with the final output register. The adder requires the most logic levels and is the critical path that limits the FPGA clock rate. Because the filter is latency friendly (allows arbitrary numbers of pipeline registers) the adders could be pipelined to reduce the levels of logic required and thus increase system performance. With one additional level of pipeline register, the number of logic levels can be reduced to two, increasing system performance by about 30%. Filter taps would need to be adjusted to account for the additional level of delay, but the module count does not increase much since the Actel architecture has registers available at the output of one half of the modules.

Implementation Results

The filter is implemented in an Actel A1460. The design was entered using schematic entry and automatically placed and routed using Designer Advantage from Actel running under the Windows operating system. The design occupies less than 50% of the A1460. The resulting device was analyzed using the Actel timer and runs at 40 MHz.

Conclusion

An FIR filter running at system data rates of 40 MHz was implemented in the Actel A1460 FPGA. The design was automatically placed and routed using the Actel Designer Advantage for Windows software package; the design uses less than 50% of the device. This design can be easily modified by simply changing the filter coefficients to implement a wide range of filters for a variety of application tasks.

Efficient BIST Techniques for Field Programmable Gate Arrays

Warren Savage
Tandem Computers

Tight gate resources and tight timing need not dissuade engineers from attempting to implement Built-In-Self-Test (BIST) structures in their Field Programmable Gate Array (FPGA) designs. With careful analysis and a little creativity, excellent results can be achieved with minimal overhead.

BIST is typically associated with large ASICs which due to their ample size, can sacrifice gates for test purposes in numbers larger than is practical for smaller chips. While this is still true, it has been found that given the right circumstances, it is fairly easy to implement a good BIST mechanism that is fast and efficient—even in an FPGA.

This paper provides an overview of a basic BIST methodology and several techniques that can be applied in the design of both FPGAs and masked gate arrays to allow for good fault coverage and minimum test times. These ideas were developed during a recent project that used an Actel FPGA.

Background

The FPGA was designed to capture environmental information, which can be reported back to a central maintenance processor from various subsystems of a large modular computer system (see Figure 1).

To minimize the required amount of interconnect and to mitigate timing problems, a serial interface was deemed most suitable for the interface between the maintenance processor and the FPGAs. The IEEE 1149.1 boundary scan interface (also know as JTAG) was chosen because of its excellent manufacturing test characteristics, its simplicity to implement, and the existence of off-the-shelf JTAG controllers that the maintenance processor could use in controlling the FPGA devices. For fault tolerance, each module has a redundant FPGA. In case of failure, the backup string is used while the primary is repaired.

The Actel 1240A-84PLCC was used to implement the FPGA design. This device has approximately 4100 gates and up to 72 user I/Os. The design had extremely high gate utilization (95.3 percent) and moderately high pin utilization (83.3 percent). The Actel layout tools were able to auto-place and route the entire design.

As seen in Figure 2, more than 50 percent of the gates are used implementing the JTAG (28 percent) and BIST

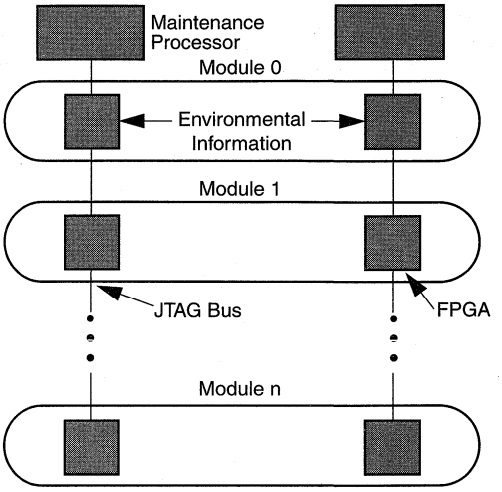


Figure 1 *• Block diagram of system. A JTAG bus is used to connect remote FPGAs to a central maintenance processor. Redundant strings are employed for fault tolerance.*

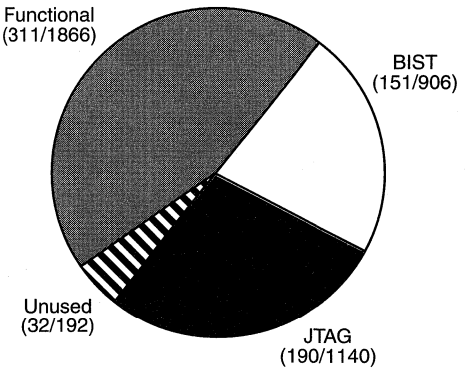


Figure 2 *• Distribution of gates by functional area. Functional logic comprises only 311 logic modules (1866 gate equivalents). Although the JTAG and BIST circuits used nearly half of the usable gates of the chip, this overhead is made palatable by the system-level usage of the circuits.*

(26 percent) control circuitry. Since JTAG is not built-in to the I/O structure of the Actel device, internal resources had to be used to construct the TAP controller and the boundary scan ring. The overhead for devices with built-in JTAG boundary scan rings would obviously be less.

One point worth noting is that JTAG logic in this design is not pure test overhead; it is used as the communication path to the maintenance processor. A similar case can be made for the BIST logic, which is used by the maintenance processor. The message here is that the designer should maintain a keen eye toward exploiting test logic for system purposes. The duality of function makes using gates for test purposes much more palatable.

Execution of the BIST is implemented as a JTAG instruction code. After the instruction is loaded, the TAP controller places the JTAG interface in the Run-Test-Idle state for about 300 ms while clocks are supplied. At the conclusion of the test, the BIST signature is scanned out for comparison to an expected "good" signature.

BIST Anatomy

When an ASIC is fabricated, gate-level functionality is verified on an IC tester by applying test vectors to the chip inputs and comparing the output response to an expected value. (See Figure 3.) These vectors are typically calculated with an IC design tool such as Synopsys Test Compiler which has the facility for calculating a precise set of design vectors and grading their efficiency. The tool tries to minimize the number of vectors while maximizing the fault coverage.

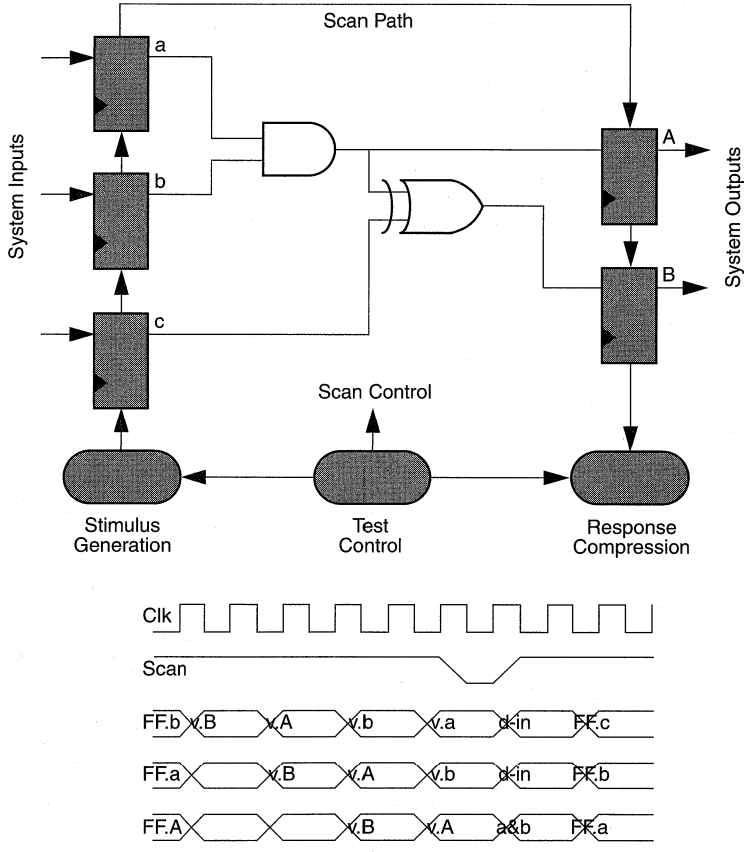


***Figure 3** • Organization of a design with BIST. Sequential elements are configured into a scan chain. Serial test patterns are applied and results are compressed.*

BIST requires that the ASIC provide a self-contained means of applying vectors to itself and evaluating the response for correctness. To achieve this, a pseudorandom number generator (PRNG) is used to apply an exhaustive (ideally) set of vectors to the chip and to collect the result into a single signature that can be interpreted as either good or bad.

Whereas Test Compiler may provide high test coverage with only a few thousand vectors, millions of pseudorandom vectors may be required to achieve comparable fault coverage. Pseudorandom vectors however, can be easily computed in hardware with a PRNG. Signature analyzers (SA), cousins of the PRNG, can easily be constructed to evaluate the responses from a large set of vectors.

Binding the PRNG and SA together with the core logic of ASIC requires a BIST controller. The controller is responsible for

- Initiating the execution of the BIST as a result of receiving an external command to do so. The mechanism may be as simple as a input pin "DO_TEST" going active or as complicated as a JTAG instruction.
- Initialization of the PRNG and SA to a fixed value before the execution of the test.
- Configuration of the scan path. During the BIST, all flip-flops are chained together to allow the pseudorandom vectors to be delivered to the core logic and the response delivered to the signature analyzer.
- Isolation, as required, of input and output pins from the logic being tested.

An important matter is the isolation of chip inputs and outputs during test. Inputs must be made deterministic during the execution of the BIST; otherwise, test results will not be consistent. Outputs must be placed in a "sage" state to prevent ASIC outputs from causing data corruption or component damage (for example, RAM write enables and bus fights).

Inputs can be handled in several ways, depending on the environment in which the BIST will run.

- If the BIST is used only during the manufacturing board test, it may be possible for the in-circuit tester to condition all the input pins to a deterministic value while the BIST executes.
- Place a gate after the input buffer to allow the input signal to be forced to a deterministic value while the BIST is executing.
- Place a JTAG cell between the input buffer and the input signal. Include the JTAG cell in the scan path for the design. This is the most elegant solution because it allows for the input signal to be toggled and allows the BIST to execute on line.

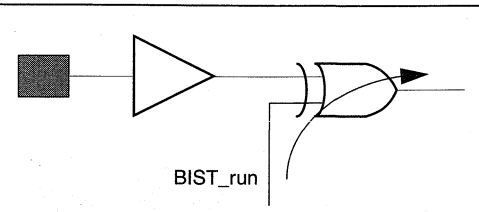


Figure 4 • A gate can be inserted into the path of the input signal to force a deterministic state on signals going to core logic.

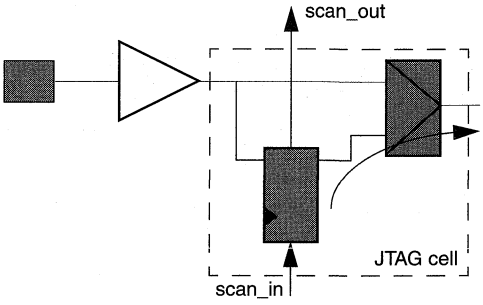


Figure 5 • A JTAG input cell can be inserted into the path of the input signal to provide isolation from the I/O pad.

Even if your design doesn't have boundary scan, the technique of isolating the input with a cell like the one in Figure 5 can be used. Using an implementation like this one maximizes fault coverage of internal nodes.

Each of these three methods has corollaries regarding the handling of output pins. Output pins, unless conditioned, will wiggle in pseudorandom fashion while the BIST is executing. If this is acceptable, then no conditioning is required. Simple conditioning in the manner shown in Figure 4 can be used to force outputs to a "safe" state.

The best solution is to employ a JTAG or JTAG-like cell on each output pin to allow data from the core logic to be collected by the BIST signature analyzer.

Figure 6 shows the configuration of such a cell. The first flip-flop is part of the BIST and JTAG scan path and is used to capture the test data format from the core logic during the BIST. The second flip-flop is called (in JTAG circles) the "update" flip-flop. Prior to execution, the flip-flop is set to a value that will result in a safe state appearing on the output pin during the BIST. A more efficient but non-JTAG-compliant solution would be to eliminate the update flip-flop entirely and connect the BIST mode input of the multiplexer directly to VDD/VSS to establish the safe state.

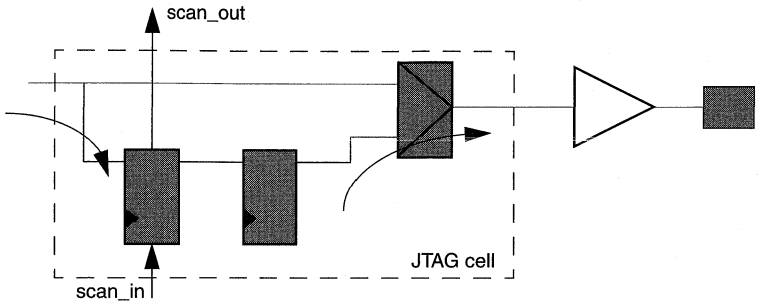


Figure 6 • A JTAG output cell can be inserted into the path of an output signal to allow capture of test data from core logic as well as to allow the output pin to be placed in a "safe" state.

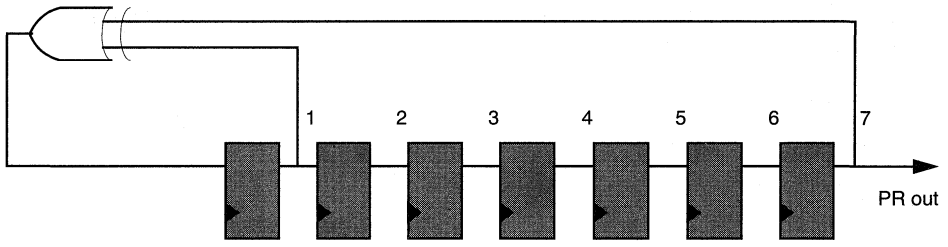


Figure 7 • A 7-bit pseudorandom number generator (PRNG) built using a LSR based on the polynomial $p(x) = x + 1$. The PRNG can be initialized to any nonzero value and generates 2^7-1 unique vectors in pseudorandom order.

Pseudorandom Number Generators

Linear feedback shift registers (LFSR) are used to construct pseudorandom number generators and signature analyzers. They are easily implemented using flip-flops and XOR gates that are hooked up according to a recursion relationship that is specified by a primitive polynomial.

The primitive polynomial specifies how the XOR gates are connected in the feedback to ensure a maximum length pseudorandom sequence. Primitive polynomials exist for shift registers of any degree and can be found in a number of texts, especially those dealing with communication theory and error correction codes.[1]

The following Verilog source for this counter synthesizes into a compact 9 Actel logic module (54 gates).

```
module PRNG(cnt,en,reset_1);
input clk,en,reset_1;
output [7:1] cnt;
reg [7:1] Count;
//Polynomial used is: p(x) = x + 1
wire feedback = Count[7] ^Count[1];
always @ (posedge clk or negedge reset_1)
  if (!rest)
    Count <= #1 7'h01;//init to nonzero
  else
  if (cnt_en)
    Count <= #1 {Count[6:1],feedback};
  else
    Count <= #1 Count;
assign cnt = Count;
endmodule
```

Signature analyzers (SA) are also based on LFSRs and are used to compress an arbitrary stream of data into a characteristic signature. The SA is based on the same mathematical principles as the PRNG and thus is similar in structure. Feedback from the sequential elements is XORed, with the incoming data stream being compressed. The SA used in the FPGA design is 16 bits wide and uses the same

1. The polynomials used in the FPGA design were taken from the appendix of [1]. These are polynomials with minimal terms thus are more efficient for implementation in hardware.

polynomial as the ones used in Hewlett-Packard test equipment. It is shown in Figure 8.

Pseudorandom Counters

Counters are used in the BIST controller to control the duration of the scan cycle and to count the number of vectors applied. Conventional counters that increment or decrement are expensive to implement because of the number of gates required to make them count in sequence. They are also slow, unless look-ahead logic is added, which further adds to the gate overhead.

An alternative to the conventional counter is to construct one using an LFSR. A de Bruijn counter is a variation of a PRNG. It differs from the PRNG in that it generates 2^7 rather than 2^7-1 vectors. This is accomplished by a NOR function that prevents the all-zeros state from causing the LFSR "stuck" state. These work well for applications in which the output of the counter is not needed directly—and in which decoders are used to indicate count values.

The counter in Figure 9 is implemented in 10 Actel logic modules—only one more than the basic PRNG shown in Figure 7. An equivalent linear counter would require 18 logic modules. The benefit is much greater for larger counters.

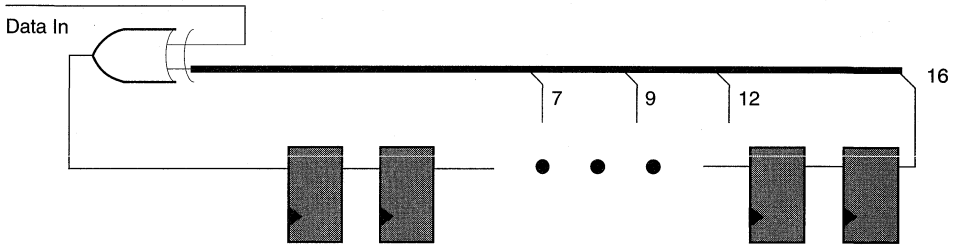


***Figure 8** • The Hewlett-Packard 16-bit signature analyzer.*

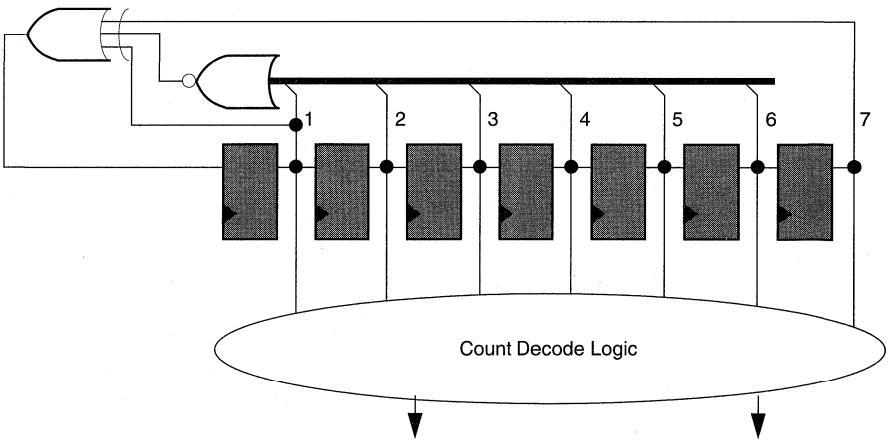


***Figure 9** • A 7-bit de Bruijn counter built using LFSR. The counter can be initialized to any value and generates 2^7 unique vectors in pseudorandom order.*

In this FPGA design, de Bruijn counters are used extensively as both pseudorandom number generators and counters. A useful Verilog coding technique used to create count decode logic for de Bruijn counters is a definition file that maps the pseudorandom count sequence to a linear count sequence. In the Verilog source, a continuous assignment statement would be used to create a decoder such as this:

```
assign done_testing = (count ==
`DEBRUIJN_128_CNT_123);
```

The desired result is a set of AND/OR gates that would produce a signal indicating that the counter has counted 123 times.

de Bruijn counters of many different lengths were used in the FPGA design, each having a unique counting sequence. This required a separate definition file for each counter size. The definition files can be created automatically using the following procedure:

1. nstantiate the particular counter into a test module where stimulus can be applied.
2. Design a stimulus such as the counter initialized to 0 and clocked to 2^n times, n being the number of bits in the counter.
3. Format the output such that it is a syntactically correct Verilog define statement that indicates the length of the counter, the linear count value, and the associated pseudorandom value.

An excerpt of the define file generated for the 7-bit de Bruijn counter is shown below:

```
`define DEBRUIJN_128_CNT_00
`define DEBRUIJN_128_CNT_11
`define DEBRUIJN_128_CNT_23
`define DEBRUIJN_128_CNT_37
`define DEBRUIJN_128_CNT_415
`define DEBRUIJN_128_CNT_531
`define DEBRUIJN_128_CNT_663
`define DEBRUIJN_128_CNT_7127
`define DEBRUIJN_128_CNT_8126
`define DEBRUIJN_128_CNT_9125
`define DEBRUIJN_128_CNT_10122
`define DEBRUIJN_128_CNT_11117
`define DEBRUIJN_128_CNT_12106
`define DEBRUIJN_128_CNT_1385

//etc

`define DEBRUIJN_128_CNT_1222
`define DEBRUIJN_128_CNT_1234
`define DEBRUIJN_128_CNT_1248
`define DEBRUIJN_128_CNT_12516
`define DEBRUIJN_128_CNT_12632
`define DEBRUIJN_128_CNT_12764
```

BIST Controller

The execution of the BIST is managed by a controller that consists of a finite state machine (FSM), two counters, and decode logic used for controlling the FPGA during the test. (See Figure 10.)

The FSM, coupled with decode logic on its state outputs, is used to control the test hardware and to sequence the BIST through its various states. One counter, the SCANCNT, is used to control the length of the scan cycle. The value to which this counter counts is dependent on the number of bits in the scan chain. The other counter, the TESTCNT, is used to count the number of scan vectors applied.

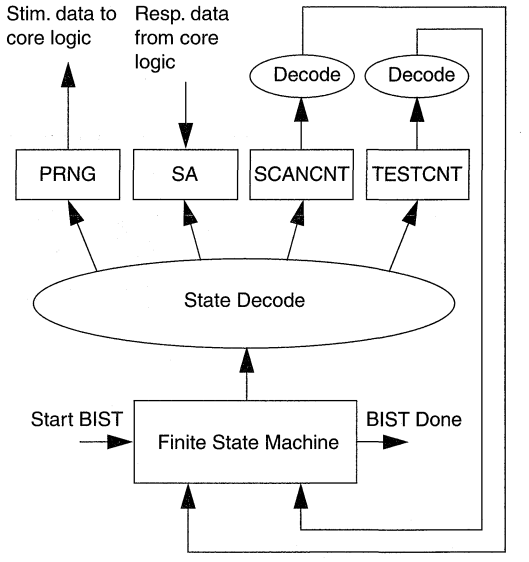


Figure 10 • Organization of BIST Controller logic.

The BIST controller initiates execution by the application of an external "start" signal. This can be a pin or, in this case, a JTAG instruction. When the test is concluded, a "done" flag is set, indicating that the FSM completed the entire sequence. Because of the long duration of the BIST, the done flag is useful for informing the host system that the BIST completed. It can be wired into the system's interrupt structure or be accessible for polling by the maintenance processor.

The state diagram for the FSM is shown in Figure 11. Here's an explanation of what goes on in each state:

- State 0, the idle state. The FSM waits here until commanded to initiate a BIST.
- States 1 and 2. During state 1, zeros are clocked into the scan string to initialize the core logic. The core logic is clocked during state 2. The SA is turned off during these states to prevent indeterminate data from being captured.

- States 3 and 4. Ones are scanned into the design and clocked. The SA is turned on for states 3, 4, 5, and 6.
- States 5 and 6. Pseudorandom data is scanned into the chip as the result from the previous clock is scanned out into the SA. These states are looped on until all the vectors have been applied.

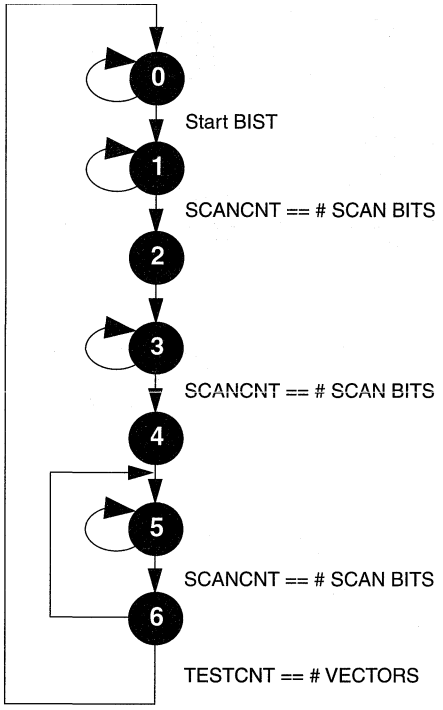


Figure 11 • State Diagram for the BIST state machine.

Consideration for Stimulus Generation

The goal of the stimulus generation is to apply an exhaustive set of patterns to the core logic in order to exercise the truth table of every combinational function contained in it.

The success in achieving this is related to the size of the PRNG and the characteristics of the design.

The wider a PRNG is the more patterns that can be applied. A 7-bit de Bruijn counter can provide 2^7 patterns covering combinatorial functions of up to seven inputs. Larger combinatorial functions can be tested with wider PRNGs. The drawback of extremely wide PRNGs is the exponential increase in test time required to apply all the patterns. For instance, a 32-bit PRNG clocked at 20 MHz requires more than 3.5 minutes to generate all of the patterns. A 64-bit PRNG would require more than 29,000 years! Furthermore, if the scan string is longer than the PRNG width, the set of test patterns applied is not exhaustive.

Because of the conflicting characteristics of test time and test coverage, trade-offs are usually made to provide a reasonable solution. The strategy employed for the FPGA was to analyze the design such that the average logic cone size was the same size as (or smaller than) the PRNG. A further constraint was that the maximal sequence length of the PRNG should be less than the scan-string length. The FPGA design used a 7-bit PRNG (128-bit pattern) that would supply patterns to a 141-bit scan string.

Figure 12 shows how pseudorandom vectors align themselves during execution of the BIST. Note that the maximal pattern length of the PRNG is shorter than the length of the scan chain. The PRNG begins repeating its sequence in the middle of the vector, concluding it on the next vector.

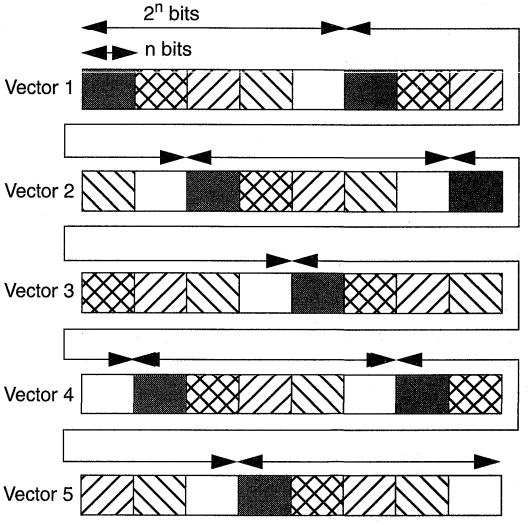


Figure 12 • Application of pseudorandom vectors to the scan chain.

Exhaustive coverage is achieved by continually applying vectors until the PRNG pattern has been applied starting at every bit position. For the patterns to precess as such from vector to vector, it is necessary for the scan chain length and the PRNG pattern length to be relatively prime (having no common divisors) with respect to each other.

With this precession, the number of test vectors to be applied is equal to the length of the scan chain. This will result in an exhaustive pattern being applied to all scan bits within a distance of n bits from each other (where n is the width of the

PRNG). This effect can be seen by looking at Figure 12 and scanning up and down the n-bit columns. After the application of each vector, the vector within an n-bit boundary will be unique.

Test time is second order and simply computed as

$$t(m) = m\,x\,(m+1)\,x\,T$$

where m is the length of the scan string. The 1 is the system clock cycle that occurs after scanning in each vector. T is the clock period of the BIST.

A shortcoming of this method is that the exhaustiveness of the pattern applied is limited to a width of n adjacent bits in the scan chain. Disjointed bits will not be exhaustively covered. Also, the length of the complete pseudorandom pattern (e.g., 2^n) must be constrained to be shorter than the scan-chain length, m, to ensure exhaustive coverage within the n-bit window.

Logic Cone Analysis

To achieve the best fault coverage, an optimal PRNG width must be chosen. Two factors are involved in this selection. First, as discussed earlier, the maximal pattern length must be shorter than the scan-string length. Second, the ordering of bits in the scan chain should be such that the logic cones are no wider than the PRNG width. A *logic cone* is defined as the primary inputs (PI) to combinational core logic associated with each primary output (PO). (See Figure 13.)

The ability to keep the logic cones smaller than the PRNG width is very design dependent. Fast designs have few layers of combinational logic (due to pipelining) and generally smaller cones. Slower designs with more layers are more problematic. The all ones and all zeros pattern helps cover large AND and OR functions; however, large complex cones are inherently less testable using this technique, and coverage is lowered as a result.

Determining cones was done (unfortunately) by hand for this design. This task is straightforward and computable from a netlist; thus, a program written in LISP (or another functional programming language) could be easily written to handle this highly recursive job.

This program, along with the *set_test_routing_order* command of Synopsys Test Compiler, would allow the BIST scan chain to be easily and quickly organized for optimal coverage.

Verification and Results

The fault-coverage results of the BIST were determined by generating a set of functional vectors that invoked the BIST and grading those vectors with Accugen software.

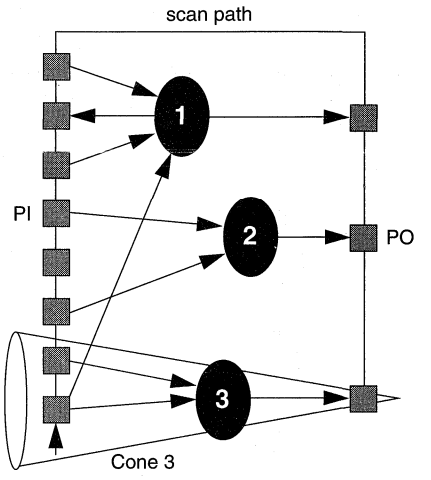


Figure 13 • *Logic cones. Logic cones represent the causal primary inputs associated with primary outputs. Test coverage can be optimized by minimizing the size of the cones through careful ordering of the scan-chain bits.*

Two sets of vectors that contained a high-quality set of functional vectors were developed, with and without execution of the BIST. There were 3K functional vectors that were constructed from Verilog simulation. The BIST required 23K clocks to complete.

Pin-level coverage was 100 percent for both sets of vectors—as expected because of the JTAG implementation. Internal nodal coverage for the non-BIST vectors was 66 percent and 83 percent for the BIST vectors. (See Figure 14.)

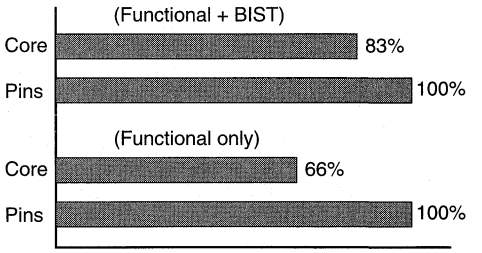


Figure 14 • *Test coverage of BIST.*

Over 48 hours of CPU time were required to grade the BIST vectors on a 66 MHz 486 machine.

Tips for Efficient Implementation

As mentioned throughout this paper, a number of structures need to be added to the design to allow BIST to work. Here are suggestions for minimizing the cost of implementation:

- Controllability of I/O.

  If a boundary scan structure is not viable, jamming logic can often be constructed by using unused functions in a logic module. For instance, a combinational module used for an AND gate can be made jammable by replacing it with an AND-OR such as this:

  ```
  wire core_signal = (run_bist ? 1'b1
  "pad_in1 & pad_in2);
  ```

- State machines.

  State machines encoded in the one-hot style are more efficient than encoded versions because of the reduced amount of decoding logic required for generation of the next state. States should be assigned such that the flow is as sequential as possible. This allows for synthesis to approximate the design as a shift register with only minor combinational logic required.

- Scan insertion.

  Scan insertion is fairly painless in FPGAs because most of the sequential modules have built-in multiplexers in front of the registers. Unless the design uses the mulitplexer, adding scan requires no overhead. The Actel device has sequential elements with 4:1 input multiplexers, so even designs that use a 2:1 multiplexer can incorporate scan with minimal cost.

  Scan can be inserted at the HDL level or in Synopsys. There are advantages to each. By having scan in the HDL, the basic BIST behavior can be simulated before entering Synopsys. However, using the Synopsys Test Compiler allows the order of the scan chain to be manipulated much more easily.

The following Verilog code fragment shows a scannable, one-hot state machine:

```
reg [3:0] State;
always @(posedge clk) begin
  if (scan)
    State <= #1 {State[2:0], scan_in};
  else
    if (!reset_1)
      State <= #1 4'h1;
    else
      case (1) //synopsys parallel_case
        State[0] : if (cond1)
             State <= #1 4'h2;
           else
             State <= #14'h1;
        State[1] : State <= #1 4'h4;
        State[2] : if (!cond1)
             State <= #1 4'h1;
           else
             State <= #1 4'h8;
        State [3]: State <= #1 4'h1;
      endcase
```

Conclusion

BIST techniques can be applied to FPGA designs with minimal gate overhead. The key to achieving this is in using the logic-module resources of the FPGA to their fullest. Also, by incorporating chip-level test techniques into system-level diagnostic strategies, system benefits are achieved in terms of fault isolation and repair.

Acknowledgements:

Thanks to Larry Rossi for his efforts in grading the BIST vectors using the Accugen software.

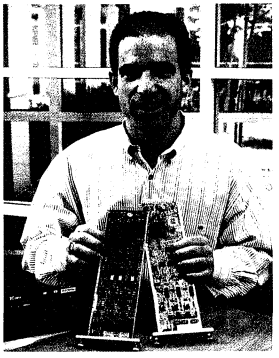
References:

1. Bardell, P. H., McAnney, and Savir, *Built-in Test for VLSI*, John Wiley and Sons, 1987.
2. Parker, K. P., *The Boundary Scan Handbook*, Kluwer, Norwell, MA, 1992.
3. Frohwerk, R. A., *Signature Analysis: A New Digital Field Service Method*, Hewlett-Packard Journal, May 1977.

3COM Corporation

CUSTOMER CASE HISTORY

3Com Selects Actel FPGAs for LAN Logic Integration, Gate Array Migration Path



David Kranzler
Hardware Design Engineer

3Com Corporation, an industry leader in local area network systems, faced several challenges when designing the TP module for its popular MultiConnect Modular Multiport Repeater platform. The MultiConnect TP Module was needed to enable Ethernet network users to communicate over previously installed telephone wiring using the IEEE 10BASE-T communications protocol standard. When installed within 3Com's MultiConnect Repeater, the TP Module would facilitate wiring hub connections for operation of 10-Mbps standard Ethernet on voice- and data-grade dual twisted pair wiring.

To design the TP Module efficiently for market use, 3Com's engineering department had to maximize the number of 10BASE-T ports that could be integrated onto a standard-sized 3Com module board, 15 of which could be accommodated by the MultiConnect Repeater chassis. The solution to this logic integration challenge was found in Actel's ACT 1 family of field programmable gate arrays (FPGAs) which ultimately helped lead to the module's design success.

The Design Task: A Need for Logic Integration and Migration Path

The design task fell onto the shoulders of David Kranzler, a hardware design engineer within 3Com's Network Adapter Division. Kranzler recalls that three factors affected the design of the product – high system density, low production cost and the need for a field-programmable solution early on in the design cycle.

"We faced various obstacles up front in designing the MultiConnect TP Module," stated Kranzler. The main problem was how to maximize the number of 10BASE-T connections on a four layer board that would fit into the MultiConnect chassis. System density played a critical role as 3Com aimed to maximize the number of ports on the module in order to optimize the number of users for the repeater. Kranzler also pointed out that since the 10BASE-T was quickly becoming a standard, the MultiConnect Module had to be designed to fully accommodate the emerging spec. Thus, 3Com needed a solution that could be reprogrammed if the standard changed. Because it was important for 3Com to be one of the first to market with its new module, it was critical that a means for field programming be implemented as quickly as possible. In addition, based on potential sales volume, it was determined that only the most cost-effective mode of high volume production could be employed to manufacture the new modules.

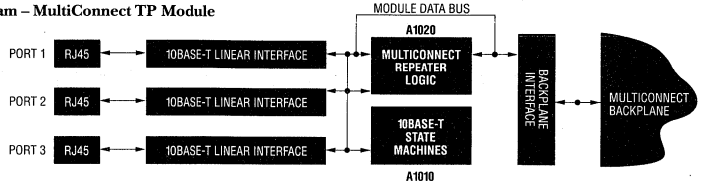
Under such constraints, Kranzler knew from the beginning that only a masked gate array could satisfy the density goals of the high-volume design. And because the market for the 3Com product was very cost competitive, a gate array would also be required to lower the module's production costs. However, the initial use of a masked gate array could not satisfy all the module's design requirements; the field programmability issue still remained.

Because the 10BASE-T standard had not been formalized at the time of the module's conceptualization, there were not a lot of off-the-shelf parts available to implement the standard, Kranzler recalls.

Actel's FPGAs: The Answer for Logic and Conversion

3Com initially looked at a number of different options to design the MultiConnect TP Module. First, Kranzler considered implementing the entire design discretely. Unfortunately, given the module's form factor, a discrete logic design would have yielded a board with only one 10BASE-T port – an unacceptable solution. Kranzler next considered designing a two-port module with high-density surface mount components, using a daughter board and the same discrete logic technology. But this alternative would have only allowed two ports and the cost would have been high. Next, the company also considered going to a full custom gate array from the onset. This solution would have provided the density needed within the cost targets but there would have been a great deal of risk involved. Finally, a field programmable gate array logic integration solution was proposed to avoid the risk element of gate arrays, the density problem of the discrete version and to eliminate the 10BASE-T specification-change

Block Diagram – MultiConnect TP Module



problem. According to Kranzler, Actel's FPGA, with an open migration path to an eventual masked array, was the clear choice.

"I was well aware of Actel's technology from the beginning and knew that its FPGAs could be used in 3Com's designs at some point," mentioned Kranzler. Actel was well suited for the design because of the generic gate-array like, granular architecture. Because Kranzler knew that these FPGAs would be utilized to almost maximum capacity, this granular architecture was a plus. In addition, with Actel FPGAs, an upgrade path to a gate array was available. Because it was proposed from the beginning to use FPGAs and then upgrade to gate arrays, this proven migration path was crucial to 3Com. Finally, because Kranzler needed maximum density within a minimum amount of space, the choice in density available to him through the ACT 1 family suited the specific needs. Ultimately, the A1010 with 1200 gates and the 2000-gate A1020 devices were chosen because they closely matched the amount of logic-integration capacity required.

According to Kranzler, Viewlogic's ViewDraw schematic software was used to capture the three-port 10BASE-T logic design and ViewSim to simulate its operation. After schematic capture, it was confirmed that an Actel A1010 and a A1020 would be needed. The choice of two Actel devices allowed the 3Com designer to cleanly partition the design of the MultiConnect TP module: The 1200-gate A1010 was used to integrate the module's 10BASE-T logic; the 2000-gate A1020 accommodated the portion of the module needed for the Ethernet repeater logic. And because Kranzler simulated the design extensively following capture, the design worked the first time it was programmed.

"The Actel devices worked out exceptionally well. I was not only happy that I could alter my design with the FPGAs in the event that the 10BASE-T specification changed, but I was pleasantly surprised that we were able to integrate as much logic as we could into the parts," explained Kranzler. "In fact," he added, "the spare gates on the devices allowed us to program logic that was used elsewhere on the board as well." According to Kranzler, the Actel devices replaced upwards of 60 TTL parts in the register-intensive 10BASE-T module design. Given the fact that the module had to accommodate three ports, a discrete-logic implementation would have been impossible.

Programmable Pinouts: An Unexpected Benefit

The programmable nature of Actel's FPGAs came in handy, Kranzler recalled, when he discovered that the pinout pattern on the board did not correspond to the FPGA pinout – the board pinout had been shifted over inadvertently by one pin during layout. "This situation would normally have been a nightmare," he said. "But with the Actel system, I was able to easily remedy the problem by reassigning the pinouts under Actel's software to accommodate the board layout, and then program another device." Thus, Kranzler was able to save the hours it would have taken to rework the board (not to mention the cost of board re-design) by simply programming the Actel device to the new pinout in a matter of minutes.

According to Kranzler, the Actel FPGAs were utilized to 98 percent of their capacity. With this utilization, 3Com was able to integrate the equivalent of three discrete TP module logic designs onto a single board. The Actel devices routed very intelligently, Kranzler remarked, and as a result, he was able to achieve his design goals and meet all of the system speed requirements. He was also pleased with the cost-effectiveness of the Actel solution. As a measure, Kranzler took his design to two other FPGA companies to see how well their products compared with the Actel two-chip solution. According to the designer, neither company could integrate the TP module with a similar dollar's worth of chips. Thus, Actel clearly was the lowest cost FPGA solution.

The FPGA-to-Gate Array Migration

The A1010 and A1020 parts were used on the initial TP Module boards that were shipped. But as the volume expanded to over nearly 1000 modules a month, it was decided, as initially intended, to convert to a 5000-usable-gate, gate array from a popular vendor for further cost reductions. According to Kranzler, the Actel FPGAs, combined with additional logic on the board, mapped into approximately 3600 gates in the masked gate array, more than the stated combined capacity of the Actel FPGAs. In fact, according to Kranzler, there was almost a 1:1 mapping of FPGA gates to masked array gates – a testament to the Actel architecture. The conversion process took approximately one month to implement and worked out extremely well, remarked the 3Com designer.

"3Com is in a high volume business and the anticipation for shipping many modules encouraged us to convert to a masked gate array," said Kranzler. There were no timing problems in migrating from FPGAs to the gate array and the teamwork between Actel and the gate array vendor made the process even smoother. 3Com currently is shipping thousands of modules a month with the gate array.

Actel Rates High with 3Com

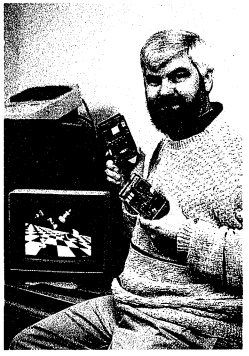
3Com, as a first-time user, was extremely happy with Actel's FPGAs and their application into the design of the MultiConnect TP Module. Actel's FPGAs perfectly fit Kranzler's design needs. Kranzler stated that if he had a design in the future that required the use of FPGAs, he would strongly consider using Actel parts again. Stated Kranzler, "I still follow the FPGA market and I am well aware of the progress Actel has made in terms of FPGA technology. Not only are they offering denser, faster parts, but I have noticed that they are now aligning with other industry players to provide more open solutions for the designer. Indeed, the use of standard technology-transparent synthesis tools and high-level design languages will be important requirements in my next design."

Actel Corporation
955 East Arques Avenue
Sunnyvale, CA 94086
408.739.1010

Beckworth Enterprises, Inc.

CUSTOMER CASE HISTORY

Actel FPGAs Provide High Capacity and Multiple Clocking Network for SBus-to-Printer Channel Controller Board



Billy Beckworth, *President*
Beckworth Enterprises Inc.

Beckworth Enterprises Inc. (BEI), a Mesquite, TX-based systems design firm had a fairly routine design task when it was asked by a major customer to provide an SBus interface card that would allow a mainframe laser printer to interface to Sun Microsystems' SPARCstations.® BEI's SBus channel driver was developed to allow a SPARCstation to drive a laser printer for those applications where customers did not already have a mainframe but did need the speed and performance of a mainframe printer or for printing in remote locations using lower cost SPARCstations.

With a BEI interface board integrated into Sun SPARCstations and with appropriate software, a mainframe printer would be accessible to a host of workstations – rather than a single mainframe – and, therefore, would become more versatile. But when that same customer came back a year later and asked the company to provide roughly twice the functionality in the same space, an otherwise straightforward design became a challenge.

A Solution that Reduced Board Space by 50 Percent Was Needed

Due to the board size limitations of a single slot board for the SPARCstation, Beckworth Enterprises' entire design needed to fit on 15 square inches of board space. This proved to be a challenge since Beckworth Enterprises had done a similar design with discrete logic and used over 30 square inches of board space. Although that earlier design involved an interface to a PC/AT bus, rather than an SBus, the design constraints were roughly the same, and the implementation would require the same number of devices.

With the size restrictions for the Sun workstation, Billy Beckworth, president, realized a higher integration solution was needed and began to explore higher capacity programmable logic that would allow him to cut the real estate in half. Additionally, Beckworth had design-security concerns for this project, as the SBus channel driver is proprietary and the company wanted to protect the design.

Although a custom gate array would have satisfied his real-estate concerns, cost restrictions prevented the consideration of masked devices. Moreover, Beckworth liked the flexibility of programmable logic. Therefore, the goal was to select a programmable device that was as granular and as close to a gate array architecture as possible with all the advantages of desktop programming. Having determined that programmable logic was the only solution, the requirements Beckworth had for this design were high capacity, high I/O, good speed performance and a solid security feature.

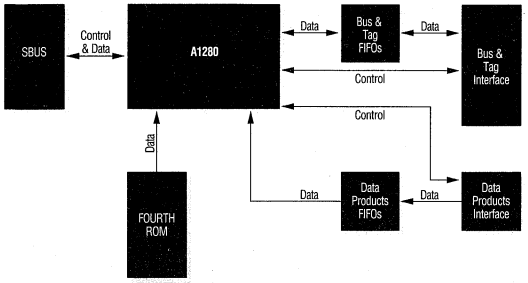
Beckworth researched the various programmable logic options and determined that the smaller programmable logic devices had the speed required for the design but did not provide the necessary density, I/O or security features. Beckworth's choice boiled down to a higher capacity solution – a Field-Programmable Gate Array (FPGA). And within that category, there were only two choices: Actel and Xilinx.

Actel Provides Security Features, Granularity Needed

With an external static-RAM memory to hold the device configuration program, as is the case with the Xilinx architecture, the risk of competitors mapping out the SRAM and determining the circuit design were too high. "There are a handful of companies that design and sell PC/AT interface cards, but as the only company that makes SBus channel drivers," says Beckworth, "we needed to protect our design. SRAMs, or any memory programming element, make it too easy for competitors to read the design and quickly benefit from our blood, sweat and tears."

Beckworth chose Actel because the chip is completely programmed internally with antifuses and provides security elements that can be blown to protect the design by preventing the map from being read. Actel's programming element, the PLICE® antifuse, is extremely small – 1.8 μm^2 – and the only way to read the antifuse map would be to view minute cross-sections of the chip under an electron microscope, a difficult, if not impossible, task.

Mega Board Data Flow Diagram



In addition to the security features, Beckworth felt that Actel's architecture was a higher grained pitch – more granular – and would allow a more flexible design approach. "Because the Actel design entry methodology closely reflects the techniques I use when designing with conventional standard logic," says Beckworth, "I could design the interface using the Actel device much the same way I would using a discrete logic implementation. Therefore, ease-of-use was high."

Board 1: Two Actel FPGAs Solve the Problem

Using Viewlogic for schematic entry, Beckworth implemented his initial design using two A1020s, Actel's 2000-gate FPGAs. The BEI interface board design was partitioned in such a way that the two devices represented distinct functions and shared no common circuitry other than power and ground. This partitioning proved useful as each device had to be driven with separate, asynchronous clocks.

Within one Actel A1020, Beckworth integrated the SBus interface and control and status registers for the board and the interface to the FIFO memories. The second A1020 handled all of the logic for the Bus and Tag channel interface. This latter device required a 5 MHz clock while the former was driven by a 25 MHz signal.

Beckworth maximized the high I/O provided in the A1020, using all but one pin on the first chip and all but six on the other. More important, real estate was also maximized. The Beckworth design implemented in 15 square inches what would normally have taken twice that amount using discrete logic.

But a year later, Beckworth's customer requested a modified version of the design. The modification called for a Data Products parallel input port plus two serial synchronous/asynchronous (USART) I/O ports to be combined with the original channel controller logic, with logic provisions to support all, onto the same, single SBus card form-factor. For this "MEGA" board design, it became clear to the designers that the approach used on the first channel controller with multiple A1020s would not fit in the same amount of board real estate. A higher density solution was needed.

A1280 Makes MEGA Board Design Possible

Due to the extremely high levels of logic integration that the MEGA board design would require to meet the equivalent board size specifications – 15 in.^{2} – Beckworth chose to use Actel's new A1280, the 8000-gate device, to replace the two A1020s and to implement the additional logic required by the modification. The A1280 provided the SBus interface and Control Section, 3211 Channel Interface and Data Products input port interface all in one 176-pin PGA package. This level of integration required the requisite drivers and receivers for the various interfaces to reside on an external interface board and a 76-pin cable connecting the two boards. Within the A1280, 133 of 140 I/O pins and 700 of 1200 logic modules were used for the MEGA design.

"The arrival of the A1280 with its 140 user I/Os and 8000 equivalent gates was the key to making the design possible at all," says Beckworth. In addition, Beckworth added that the A1280's dual clocking networks made the device the ideal choice for logic integration. This feature proved especially useful, particularly as the original design required two independent clocks. "The two independent clocking networks would not have been possible without the A1280. While I might have been able to piece a solution together using multiple subnets, that approach is not the most elegant solution."

Two Clocks are Better than One

Specifically, the two clocking networks allowed Beckworth to simultaneously clock the A1280 at 25 MHz for the SBus interface and 5 MHz for the Data Products and Bus and Tag Channel Interfaces. Separate clocks were needed because the complex state machine can't be driven as fast as the backplane due to the prop delays.

The wider gate inputs of the A1280 also proved useful to Beckworth. In fact, only one-half of the A1280's capacity is utilized because of the ability to map to the wider inputs and the combinatorial nature of the sequential and combinatorial blocks. "I tried to make it as efficient a state machine as possible," says Beckworth. "Basically, I've got two or three state machines that operate on the different inputs and outputs. That technique makes for a very clean design." As important, the larger density device offered the performance and I/O he needed to meet his size specifications.

The conversion to the A1280 from the A1020-based design was very easy and integral to the success of the new design. "The conversion process was a snap," says Beckworth. "All that was required was to take the basic [A1020] design and modify it for the additional logic functions and implement it with an A1280." Beckworth points out that although he could have recompiled the A1020 design for the A1280, it was not the most efficient approach to implement the modifications. Instead, Beckworth took the A1020 building blocks and enhanced them for the A1280 architecture. "The nice thing about the Actel architecture is that once you've designed with one device, the knowledge you've gained enables you to shorten your design time of the next project."

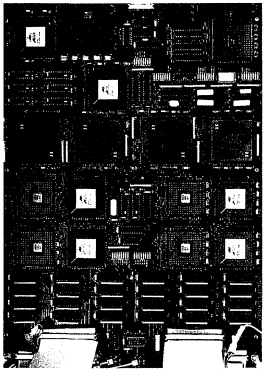
Both versions of the interface card using the Actel parts were very successful. In addition to the device benefits, Beckworth was pleased with the design support he received from Actel and felt the applications engineers were very helpful in solving his design problems. Beckworth plans to use the Actel devices in his next revision of the interface card in order to guarantee the same level of integration and performance.

Actel Corporation
955 East Arques Avenue
Sunnyvale, CA 94086
408.739.1010

Interstate Electronics

CUSTOMER CASE HISTORY

Interstate Electronics Opts for Actel FPGA's Speed, Density and Non-Volatile Technology



Interstate Electronics' high-speed parallel processing demodulator board

Delivering performance on time often requires getting a head start. So when a major customer tells you that the high-speed modem you've just designed and delivered needs to be 26 times faster to satisfy the requirements of the next generation product, the time to start the upgrade is now. Such was the case with Interstate Electronics, a defense systems contractor located in Anaheim, California. Interstate had just reached the final stages of development for a 25 megabit per second (Mbps) modem for NASA's second Tracking Data and Relay Satellite Subsystem (TDRSS) ground terminal. Destined to replace the present system in White Sands, New Mexico, which had been on-line since 1985, the new Interstate modem was in its production phase when the company received word that NASA was preparing a Request For Proposal (RFP) on an advanced version of the TDRSS Ground Terminal. The modem for this new system would require a 650 Mbps data rate – 26 times faster than the one Interstate had just completed. Although the RFP wasn't expected until the end of the year, Interstate wanted to step up to the challenge and prove the feasibility of meeting such a requirement with a modification to its existing digital modem design.

As far as the Interstate system designers knew, no one had ever built a digital modem that even approached these speeds. "The project presented an intriguing design opportunity for us," said Mike Casteloes, a senior engineer assigned to the project. "Moreover, because it is highly likely that NASA has plans to place a similar modem in space, that facet of the project held an even greater interest for us," he noted.

Design Objective: A Cost-Effective Approach to Higher Speed

Clearly, winning the project spurred interest at Interstate. But first, the company needed to solve the modem's speed problem, so it began working on a solution at once. Two choices presented themselves to Casteloes as he reviewed the problem of handling a 650 Mbps data rate: he could maintain the speed throughout the system with high-speed, gallium arsenide (GaAs) circuitry, or he could decrease the speed in successive stages by converting the serial input to a parallel data path.

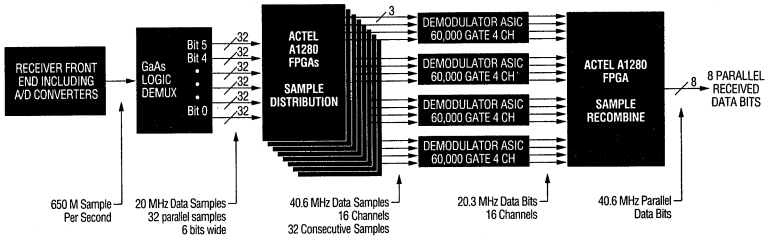
"A careful examination of the signal processing required to convert the satellite's microwave transmission to a digital output revealed that the complexity of the system would make the GaAs serial solution too expensive to implement," said Casteloes. "We opted for the parallel processing solution that employed a judicious application of GaAs, ASIC and some sort of lower cost parallel CMOS logic integration solution such as the new high-complexity PLDs (Programmable Logic Devices) or FPGA (Field Programmable Gate Array) technologies." According to the Interstate engineer, the combination offered a good performance-speed compromise to the more expensive all-GaAs approach.

The modem system design begins with a receiver front-end that converts the microwave input to an Intermediate Frequency (IF) range. At this point, the data exists as a digitally phase shifted IF waveform – a sine wave that has been modified by a stream of digital data so that every state transition causes a phase shift in the sine wave.

The receiver's IF output is then converted to baseband (mixed) and sampled using high-speed analog-to-digital (A/D) converters. The resulting serial stream of 6-bit samples is then fed to a GaAs demultiplexer circuit that separates the 650 megasample-per-second data into 32 parallel samples, each six bits wide. Splitting the data into parallel samples slows the data rate to 20.3 MHz, a rate that easily can be handled by less expensive CMOS circuitry.

The next stage of the system required a massive array of shift registers to store and distribute the sample data to a bank of four 60,000-gate custom gate arrays. "This was a critical section of the design," Casteloes recalled. "To convert the sampled analog data to logical ones and zeros, we needed to collect the data into blocks of 32 consecutive samples. Otherwise we would never have been

650 MHz Receiver Structure Using Parallel Processing



able to reassemble them accurately. To accomplish this we designed an array of 32 shift registers, each one 32 x 3 bits wide. By combining their outputs, we could produce 16 channels, each containing 32 consecutive samples, and each running at 40 MHz."

The Selection: Non-Volatile Fuse Map is the Key

"Designing the circuit was straightforward enough, but implementing it posed a problem with board space," said Casteloes. "It simply wasn't feasible to implement that many shift registers in PALs; it would have taken up far too much room. And even if we used any one of the higher-complexity PLDs, it still would have required too many devices to implement the design." The designer recalls that creating another ASIC was also out of the question. "We had spent a considerable amount of non-recurring engineering (NRE) charges with the custom ASIC already. Spending even more on NRE was immediately discounted," he added.

FPGAs seemed the only logical choice, and Casteloes was already familiar with these devices – he had used Actel's FPGAs and design tools for an earlier project. "We calculated that our Sample Distribution circuit would only require eight A1280s – less than a quarter of the number of high-complexity PLDs that would have been required to implement the same function," said Casteloes. "As it is, the amount of I/O on our 10″ x 18″ board is substantial," he added.

Casteloes also looked at the A1280 because of the high speed requirements of the system. "Finding an FPGA that can operate with a 40 MHz system performance is no easy task," he said. "We believed we could get that kind of performance out of the A1280 devices because of our prior experience with the Actel architecture."

Actel's FPGA was not the only product considered. "Although we could have used a greater number of Xilinx FPGAs because they had adequate performance for this particular application, the A1280 device had one major advantage – its fuse map used a non-volatile technology," Casteloes explained, "which guarantees that the device will be mapped correctly on each power up." According to Casteloes, designing with the static-RAM-based Xilinx product would have required down-loading all fuse map information each time the system was powered up. "With all the high-speed digital switching on board," Casteloes observed, "I was afraid the slightest noise glitch could cause the Xilinx device to be configured incorrectly and malfunction, a situation that you wouldn't want to tolerate on the ground and which couldn't be tolerated in a satellite-based system."

Along with the potential vulnerability of the system, programming the fuse map each time would require additional circuitry, including a PROM to store the fuse map information for each unique FPGA design, as well as support for diagnostics to test each FPGA after power up. "The choice was obvious," said Casteloes, "Actel was the clear winner."

The next stage in the system reassembles the 6-bit samples into digital data. Four ASICs synchronize the sample data received from the FPGAs and reconstruct them into 16 channels of useable data. The ASICs also provide timing and phase error estimates. Finally, the 16-channel parallel output of the ASICs are recombined into eight parallel data bits at 40 MHz by an additional FPGA. Ultimately, the digital data is fed to a GaAs 8:1 multiplexer used to convert the parallel data back to a serial data stream for retransmission.

Actel FPGAs are the Right Choice

Implementing a complex design in a high-performance, high-density environment is a challenge under any circumstances. Developing a working design quickly can seem insurmountable without the aid of FPGAs. "Even though we are just now shipping the first of the 25 Mbps systems to NASA, we have already proven the 650 Mbps system works, and we were able to stay within the circuit board space and budget constraints," said Casteloes.

"If I had my way I would use Actel FPGAs for every high-density design," Casteloes said. "Actel's products are easy to work with, offer the highest available logic density and performance and can be relied upon to satisfy the most demanding design requirements. The way I look at it, once you have a good design environment, stay with it!"

Actel Corporation
955 East Arques Avenue
Sunnyvale, CA 94086
408.739.1010

51921050

Delphi Systems

CUSTOMER CASE HISTORY

Delphi Combines Actel FPGAs and DSP Technology for Superior Speech Compression

Peter Schwartz, *CEO*
Delphi Systems

With the increased use of conventional phone lines as a major communications medium, many large companies are relying on private telephone networks to reduce their communications costs. In such cases, higher-cost leased lines would primarily be used for long distance traffic. Although these leased lines often carry data, the predominant requirement is to carry voice traffic.

In the UK, British Telecomm (BT) and Mercury Communications Ltd. are the typical suppliers of long distance services. A BT KiloStream service, normally used to carry data with a capacity of 64kbits/sec, can carry only one PCM (Pulse Code Modulated) speech channel, while MegaStream at 2.048Mbits/sec was developed for higher traffic and is equivalent to 30 KiloStream channels. Although MegaStream is considerably more costly than the KiloStream service, users have been willing to pay a premium to gain the higher capacity channels. It's to no surprise, therefore, that there have been considerable efforts by several firms to improve the channel-handling capabilities of lower-cost long distance communications lines.

At least one company, Delphi Systems (Salisbury, Wilts), has pioneered a solution. Delphi is using speech compression techniques to enable multiple-channel communications over the single-channel KiloStream service. In fact, the company's newly developed Delphi Speech Compression CODEC (DSC) provides speech compression sufficient to allow KiloStream to carry up to eight toll-quality voice channels plus control information, or twelve at near toll-quality, potentially reducing costs and offering the communications manager more flexibility in planning his network. Savings vary according to the amount of traffic. As an example, for between five and 20 lines, DSC speech compression on a KiloStream service would cost around half the MegaStream equivalent in the first year, reducing to around 20 percent of the cost thereafter.

The DSC exploits a technique called Code Book Excited Linear Prediction (CELP) to achieve this compression. A prediction based approach, CELP was first detailed by AT&T Bell Labs in a 1985 IEEE conference paper, where speeds 100 times slower than real time on a Cray 1 supercomputer were reported. By the late 1980s, the emergence of high performance digital signal processors made it possible to carry out the heavy computation needed for CELP in real time. Delphi was founded to exploit the market for CELP-based communication products, particularly the private telephone network market.

Different Product Phases Require Flexible Solutions

According to Peter Schwarz, chief executive of Delphi Systems, the product development for the DSC was segmented into three distinct phases: algorithm development and feasibility study, evaluation products and final implementation.

The first phase, algorithm development, had the twin objectives of creating and efficiently coding the CELP technique and confirming the feasibility of running CELP in real time. Schwarz recalls that at the time the project was initiated, only AT&T's 32 bit floating point processor, the DSP32, was available; the higher performance DSP32C was due within the next six months. Software was initially developed on a personal computer and coded in Turbo Pascal, with the intention of porting to the DSP32C when it became available.

Initially running 5,000 times slower than real time, a subsequent port to a higher-performance PC – a 386 machine with a 387 co-processor – produced a fifty-fold speed improvement. Porting to the DSP32 saw a further 80 times improvement, sufficient to both prove the technique and provide confidence that with the next move, to a DSP32C, real time speech compression was viable.

The second stage was to develop a real time system on a PC board to enable potential customers to evaluate the CELP technology, educating the potential user base on the technology and providing a revenue stream to help fund future developments. The evaluation systems used the first commercially available DSP32C parts available in the UK and

were sold to a number of telecommunications and related companies. One commercial product was based directly on the PC board: a multi-channel voice logging system for emergency services. Compressing speech and storing it on hard disks made it possible to instantly retrieve and replay random messages while continuing to record incoming calls, something impossible on previous tape based systems. Customer feedback from the evaluation systems was positive and provided valuable input to the final design stage.

Choosing the Logic: Density, Security and Flexibility Requirements

At the beginning of the third phase, the development of the final product, there was a technology evaluation of the most appropriate form of logic. The board was to be a compact daughter board, placing a severe limit on the chip count after the DSP, EPROM and SRAM circuitry were added. The technology evaluation criteria included the potential chip count, security (both in use and to protect the proprietary elements of the product), ease of design and cost. Products evaluated included a microcontroller, masked gate arrays and field programmable gate arrays (FPGAs). The microcontroller was ruled out on both chip count and lack of security. The masked gate array was inflexible, had long turn-around times, was expensive with large up-front NRE charges and did not suit Delphi's preferred incremental development route. FPGAs, therefore, became the technology of choice.

Two FPGA families were evaluated, one from Actel Corporation and the other from Xilinx Inc. According to Schwarz, "although both FPGA approaches had no NRE and offered fast system implementation, we quickly concluded that the Actel part was the logical choice. Indeed, because it used an antifuse technology for programming rather than a static-RAM plus external EPROM-based technique, it was nonvolatile and provided a much higher level of security – a strong requirement for the DSC." The Actel gate array-like architecture, Schwarz continued, also offered a straightforward migration path to a full masked gate array as production volumes rose. The design tools were available on a PC platform, further minimizing the up-front costs.

Using Actel FPGAs For Integration

Once the decision to go to Actel was made, an Action Logic™ development system was purchased from Gothic Crellon, a UK-based sales representative. Schematic capture and simulation from Viewlogic® and place and route device programming from Actel were used.

The challenge to Delphi's design team was to create a standard set of modules, integrating these to create larger functional blocks. "As an area was integrated," said Schwarz, "a device was programmed and tested in a prototype board. At the early stages of development, both the EPROM and the DSP could be run as emulation systems via the PC with connections to the prototype board." The first A1010 (1200-gate) device contained only the clock circuitry and the circuits to load code from the EPROM to the DSP32C. There was no I/O, so the PC tools were used to ensure that this part was functioning correctly.

The step-wise approach allowed the final development of the software to take place in parallel to the hardware, an approach impossible with a standard gate array. This approach provided very tight project control and was a major factor in the ultimate success of the project.

Timing Mystery Solved with Actionprobe

Schwarz points out that at the final iteration of the design, an unexpected problem arose. "After adding the final security circuitry to the tested and fully functional design," he remarked, "some programmed devices inexplicably failed to work correctly." He remembers that a period of careful analysis identified the problem as a timing error that had not been present previously. Following a design change, the Actel system reruns the place and route, resulting in a different layout on the FPGA. In this case, a timing alteration had occurred with a circuit that had not been identified as a critical net. Further analysis localized the problem. However, even the post layout simulation failed to locate the problem – the circuit simply did not operate in accordance with the simulation.

To isolate the problem, Schwarz decided to take advantage of a special diagnostic feature that was developed for Actel FPGAs. Called Actionprobe," the feature exploits the antifuse programming circuitry to probe any pair of internal nodes in an in-circuit functioning device without loading the internal nodes or modifying the circuit operation. Coupling Actionprobe to an oscilloscope allowed signals from the faulty circuit to be displayed. These were directly compared with the post layout simulation waveforms, displayed on the PC screen. This direct comparison reduced the actual time spent in locating and probing the timing problem to minutes.

With the timing problem identified, place and route was instructed that this was a critical net and the re-layout proved to be 100 percent successful. The Actionprobe was used again with the relayed chips to measure the timing tolerance in the critical nodes.

Actel FPGAs Are the Right Choice

The DSC is now in production with six major variants. These include versions that operate at either 6,400 bps or 4,800 bps and a dual speed switchable version; each of these is available with or without an echo canceller. In each case, the PCB and assembly is identical, with the actual variant being produced by selecting one of three different FPGAs and one of two different EPROMs.

Manufacturing is contracted out. When an order is placed, parts are ordered from Gothic Crellon, programmed by them to meet the order requirements and passed to a subcontractor for board build. Capital tied up in stock is kept to a minimum with this just-in-time approach and the programmed FPGAs are produced as needed. The latest version of the CODEC now uses double sided surface mount technology (SMT) to shrink the board from 4" by 3.75" (15 square inches) to 2" by 3" (6 square inches), but still uses the same Actel FPGA.

It is fair to say that without the Actel FPGA approach not only would Delphi not have the same range of CODECs as are offered today, but the product itself might never have reached the market.

Actel Corporation
955 East Arques Avenue
Sunnyvale, CA 94086
408.739.1010

 5192103-0

GE Medical Systems

CUSTOMER CASE HISTORY

Actel Corporation

February, 1993

GE Medical Chooses Actel FPGAs for their Design Flexibility and Ease-of-Use

GE Medical Systems, one of the leading suppliers of medical imaging equipment, continues to push the limits of technology through on-going product development efforts in order to provide customers with the most advanced real-time imaging systems available. One of the recent systems from the Waukesha, Wisconsin-based organization, required the design of a complex image acquisition and display board to facilitate the processing of images. The image acquisition and display board, a VME-based board residing in a Sun SPARCServer®, would need to take the image data from a proprietary data detection device, store it into the board's image memory, process the image and drive a high-line (1024 x 1024) display monitor. However, this was not an easy design task because of the significant amount of logic to be incorporated on a board strictly limited by the fixed VME form factor.

Additionally, in the fast paced world of medical imaging, where systems are constantly being enhanced and new technologies replace old, limiting the system design time is of the essence. Therefore, the engineers were working to meet a six month design time frame for a fully implemented imaging board.

Clearly, choosing the logic integration vehicle would be key in maximizing the engineering team's productivity. Moreover, the solution would need to meet the dense logic requirement of the application in the limited space of the 14" x 14", 9U VME board and all in the short six month time frame. Mike Juhl, senior design engineer with GE Medical Systems, initially evaluated several options before deciding on the one that best met his needs.

One option was to use a completely discrete solution. However, Juhl immediately rejected this because he knew the application was heavily logic intensive and would require a large number of discrete devices, which would exceed the limited board space. In fact, if implemented using discretes, the solution would need at least 600 MSI devices, translating into 3 VME boards, for implementation.

GATE ARRAY USE DISCOUNTED

A second alternative was to use masked gate arrays, which could integrate the logic required to accommodate the limited board space. However, the design, implementation and fabrication of a single gate array can itself be a six month project, and Juhl needed a completed board in six months. To use the gate array solution, it would be absolutely necessary to have first-time-design success. Even a single design flaw would require a new gate array. Juhl determined he could not take the risk.

"The decision seemed obvious," Juhl recalled. "The only viable solution that would meet all of our requirements would be the flexibility of field programmable gate arrays (FPGAs). FPGAs would provide the high capacity required for the application in the board space available, and, because of its field programmability, the design could be implemented quickly."

The next step was to determine which FPGA supplier to choose. Juhl knew approximately how much I/O and flip-flop capacity he would need for the design implementation. Because he wanted to partition the design by functionality, he knew that he would need to use multiple FPGAs, each in the 6,000 to 8,000 gate arena.

After comparing the available solutions, Juhl concluded that he would be able to get the densest solution available by using Actel's A1280, an 8,000 gate device from the ACT™ 2 family. The A1280 had a usable gate equivalent of 6,400 gates, with

955 E. Arques Avenue Sunnyvale, CA 94086 408 739-1010

120 user I/Os and 998 flip-flops. Additionally, he had implemented a design using Actel's ACT 1 family in a prior system and was very happy with their performance.

AN EXTREMELY FLEXIBLE SOLUTION

Another factor that influenced Juhl's selection of Actel was his previous experience with the ACT 1 family and Action Logic™ System for design and programming. "The migration from the ACT 1 tools to the ACT 2 tools was very easy. In fact, the design process didn't change at all," stated Juhl.

Juhl designed the board using Mentor Graphic's schematic capture tool, NetEd, and digital simulator, QuickSim, which support Actel's library. These tools provided support for both funtional and post-layout timing simulation. "This is a key requirement for designers today. Not only does the FPGA need to work, but eventually the entire board needs to work. With Actel's devices and tools, in conjunction with QuickSim, the FPGA can be simulated by itself and then re-simulated in the complete board environment," stated Juhl.

FPGA INTENSIVE: USING 10 ACTEL DEVICES

When Juhl completed the design and implementation, within his six month window, the image acquisition and display board required approximately 75,000 gates of logic. To accommodate the logic requirement, Juhl used 10 Actel devices: nine A1280s and one A1020, a 2,000 gate device from Actel's ACT 1 family. Each of the Actel devices is partitioned by its functionality in the system. Functions include VME interface, receiver data decode and format, image memory data interface, memory controller, transmitter data format and video data format.

Using multiple FPGA devices from any other supplier would have made the placing and routing of the individual device an extremely difficult and time-consuming task. Thus, Actel's ability to perform 100% automatic place and route was a key advantage for multiple-FPGA designs. "Once the partitioning of the 10 devices was completed, programming them was as easy as pressing 10 buttons, walking away and coming back to find they were 100% placed and routed. This saves engineering time that can then be applied to other activities, rather than spending it manually placing and routing," commented Juhl.

Operation of the board occurs by the image data coming in from a detector source, via a fiber optic channel. Working with the Sun host CPU, the VME interface consisting of one A1280 device, receives commands that tell it what format that the data is coming in. For example, the commands inform the board of whether the data is encoded and whether it is coming in at a 768 x 768 or a 1024 x 1024 line rate format. The receiver data decode and format devices then process the incoming data, with two A1280s to process two pixels in parallel, and send the formatted data to the image memory data interface. The four A1280s used for the image memory data interface connect the on-board image pipe-line to the 64 MByte DRAM buffer, which stores up to 32 images in a 1024 x 1024 format.

The memory controller is the main control function on the board arbitrating the storage of received images with the real-time display of images and the transfer of images over VME to the host CPU. In the video display mode, the controller can be configured to display any sequential set of images in the buffer in a variable rate video loop. Additionally, the memory controller can be configured to operate as a simulator for the detector source and transfer images out over a fiber optic channel at a 30 frame per second rate.

The memory controller was the most challenging design on the board. However, Juhl hoped that with the A1280 and careful design he would be able to integrate the solution into one device. Due to the efficiency and flexibility of the Actel PLICE® antifuse-based architecture, Juhl was able to implement the memory controller using only one A1280, utilized at the extremely high rate of 95%. "This was the most difficult design in the system and was efficiently implemented in the Actel A1280 device," stated Juhl.

Because the system performance requirement for the design was specified at 28 MHz, Juhl did not have a performance concern with the Actel parts,

955 E. Arques Avenue Sunnyvale, CA 94086 408 739-1010

which offered operation to 55 MHz. Juhl added, "I had no problem obtaining the performance I needed from the Actel devices."

UNEXPECTED BENEFITS FOR ADDITIONAL FUNCTIONALITY

In addition to the density, flexibility and performance Juhl expected from the Actel devices, he gained additional functionality not originally anticipated. "One benefit of the Actel approach was the ability to put boundaries around the logic by partitioning functions to each of the FPGAs —I felt confident that the FPGAs would be able to handle those functions. I was then able to start placing and routing the board, even before I had all the detailed design completed on a couple of the FPGAs."

Juhl also discovered that the Actel devices allowed him to add functionality late in the design cycle, typically a monumental task with other programmable logic solutions. "I was pleasantly surprised that I could add functionality to the Actel devices late in the design cycle with basically no extra cost. This happened a couple of times during the design cycle when we put additional features on the device and once when we decided to change functionality after we had fabricated the board."

NEXT GENERATION ALREADY IN DEVELOPMENT

Juhl was very satisfied with his choice of Actel FPGAs and continues to use them in his next generation design. Currently, GE Medical Systems' next generation of the image acquisition and display board is in the R & D process. "In general, the new design is a bit more complicated, but Actel FPGAs are easily handling the increased logic complexity. We will also be able to save even more time in this second generation by leveraging the design experience and the macros from the initial design."

Indeed, GE Medical has chosen to stay with the Actel FPGAs but has changed the mix of the Actel parts to accommodate the new partitioning of functions on the board.

Juhl commented, "Again, we are under a tight time frame for the development of the second generation board and believe that the Actel devices offer the functionality that we will need. The devices provide high available logic density, offer an easy-to-use device and design tools package that is well integrated with third party board level design and simulation tools, and provide the performance necessary for this logic intensive system."

955 E. Arques Avenue Sunnyvale, CA 94086 408 739-1010

Chipcom Co.

Article published in the 1993 PLD Conference Proceedings

1.1.3.C

Real World State Machine Design with FPGA's

Ira D. Hart, Section Manager
Chipcom Co., Southborough, MA

Introduction

Designing state machine control circuitry for implementation in an FPGA is easier today than ever before. A designer no longer needs to spend a great deal of time optimizing state machines with Karnaugh maps and state tables. With the advent of synthesis tools such as those made by Exemplar or Synopsys, a state machine can be described in an HDL such as ABEL or VHDL and quickly converted to FPGA gates. The state machines can be synthesized with the constraint of gate count or speed. They can be represented in your schematic as small behavioral "black boxes" which can be linked together and simulated. The FPGA and it's associated tools allow a designer to quickly capture the design and burn it at his or her desk to prove the design functionally in the lab.

This paper will describe a design example and the trade-offs made for an interface between a processor, 2 Ethernet controllers and a shared SRAM bank. The design contains 6 state machines, shift registers, counters and multiplexing circuitry for the data and address busses. It was implemented in an ACTEL 1280 FPGA which can fit roughly 8,000 gate array gates and has 140 user I/Os of which 125 were used in this design

Arbiter Description

A memory arbiter is used in this design to allow the 2 lances and the CPU fair access to a shared SRAM. The scheme used is first come first serve. If there is contention, the last requester to have been granted access to SRAM will wait. The arbiter was described in ABEL HDL in a "one hot" or state per bit encoding scheme. ABEL outputs a .PDS description format which is basically a Boolean description. This is read in by the Exemplar synthesis tool which will output Actel ACT2 gates. The Exemplar tool spends 5 minutes analyzing the Boolean description and providing different solution sets. For the Arbiter in this example, Exemplar was setup to optimize for fastest speed. The Exemplar tool made 9 passes with the best pass occurring on the first iteration. The resulting arbiter was implemented in 77 ACT2 modules with a worst case setup time to the FF of 46.3nS.

Problems with One Hot Encoding

Classroom state machine design techniques usually involve binary encoding for state machines. For example 4 bits can be used to represent 16 states. Decoding logic for the outputs or next state then look at the inputs and the previous state to determine what to do at the next clock edge. Frequently, the decoding logic necessary to decode the transition conditions for changing to the next state use multiple module levels. This can cause timing problems if the number of levels are high enough that the propagation delay through the decoding logic causes a setup violation at the input to the state bit flip flop.

One hot encoding is frequently recommended by FPGA vendors as a way to improve speed and module count. This is done by increasing the number of flip-flops used to represent the states. The state transition conditions usually include a very low number of logic levels in the decode

block, making the operating frequency high. Simple combinatorial logic is used to decode the state bits. For example, a case where an output needs to be low if it is in state 3 and state 4. A 2 input NOR gate can be tied to the Q output of each one hot flip flop.

This output would theoretically remain low in each state and during the transition between states. In reality though, the output will glitch during the transition due to the simple fact that the flip flop for state 3 may turn off before the flip flop for state 4. This glitch would cause a serious problem if the output were controlling an asynchronous input such as a chip select or output enable of a memory device.

A Better Approach

The problem descibed above was encountered in this Actel 1280 design for the signal UDTACKl. This signal is the data acknowledge handshake to the board CPU. The CPU runs at a different clock speed than the arbiter block. It is an asynchcronous signal which needs to remain glitch free. One option was to synchronize it with the CPU clock but this would of course have led to a performance hit equal to up approximately two CPU clocks. One approach is to use an HDL which decodes the inputs and previous states synchronously, to "look ahead" and provide glitch free transitions of the output on state machine clock changes. This is accomplished in the arbiter example. The disadvantage of this approach is that these outputs must now decode the inputs and previous states to switch. This can result in the same problems we had with the encoded state machines, because of the complexity of the logic needed in front of the flip/flop. An alternative approach is to use a pseudo- 'one hot' technique where more than one state is active at a time, insuring glitchless transition between states which require it. This usually will not require any additional logic, since the state flip/flop is simply cleared on exit of a different state.

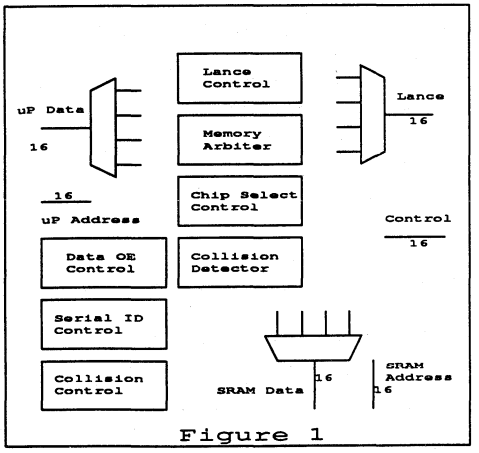


Figure 1

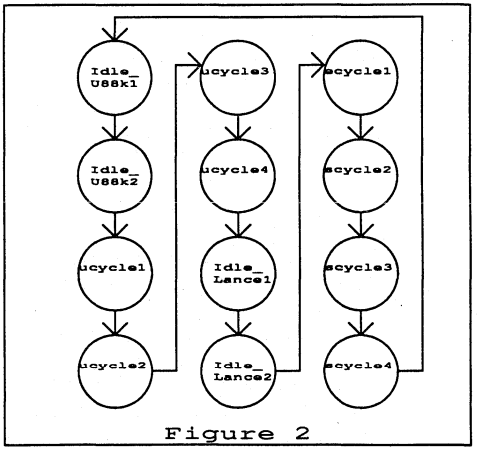


Figure 2

1.1.3.C

Arbiter Abel file

```
MODULE memarb
title 'Dual Port Memory Arbiter
Ira D. Hart     Copyright  CHIPCOM Co. 1992  '
declarations
"InputsClk20pin istype 'buffer';
Resetlpin       istype 'buffer';
SRAMCSlpin istype 'buffer';
"Outputs
UDTACKlpin istype 'reg, buffer' ;"68k
ADDMUX1pin istype 'reg, buffer'
EDTACKlpin istype 'reg, buffer'
s12pin istype 'reg, buffer' ; "states
s11 pin istype 'reg, buffer' ; "states
s10 pin istype 'reg, buffer' ; "states
s9pin istype 'reg, buffer' ; "states
s8pin istype 'reg, buffer' ; "states
s7pin istype 'reg, buffer' ; "states
s6pin istype 'reg, buffer' ; "states
s5pin istype 'reg, buffer' ; "states
s4pin istype 'reg, buffer' ; "states
s3pin istype 'reg, buffer' ; "states
s2pin istype 'reg, buffer' ; "states
s1pin istype 'reg, buffer' ; "states

ADDMUX =[ADDMUX1,ADDMUX0];
U68K  =[ 0,   0    ];
Lance =[ 0,   1    ];
PP    =[ 1,   0    ];
CC    =[ 1,   1    ];
High,Low   =1,0;
H,L,c,x,z  =1,0,.C.,.X.,.Z.; "test vector characters

Qstate=[s12,s11,s10,s9,s8,s7,s6,s5,s4,s3,s2,s1]; "
Idle_U68K1= ^b000000000001; "s1
Idle_U68K2= ^b000000000010; "s2
Idle_Lance1= ^b000000000100; "s3
Idle_Lance2= ^b000000001000; "s4
uCycle1=    ^b000000010000; "s5
uCycle2=    ^b000000100000; "s6
uCycle3=    ^b000001000000; "s7
uCycle4=    ^b000010000000; "s8
eCycle1=    ^b000100000000; "s9
eCycle2=    ^b001000000000; "s10
eCycle3=    ^b010000000000; "s11
eCycle4=    ^b100000000000; "s12
```

```
1.1.3.C

equations
[UDTACK1,EDTACK1,ADDMUX1,ADDMUX0,memcycl,s12,s11,s10,s9,s8,s7,s6,s5,s4,s3,s2,s1
s2,s1].CLK =                                    Clk20;
[ADDMUX1,s12,s11,s10,s9,s8,s7,s6,s5,s4,s3,s2].RE=    !Resetl;
[memcycl,ADDMUX0,EDTACK1,UDTACK1,s1].PR=    !Resetl;

state_diagram Qstate

State Idle_U68K1:
EDTACK1:=!(!EDTACK1 & (!EAS1 # !clk10)); "Sustain till end of EAS. w/clk10
If (!SRAMCS1) Then uCycle1                    "68K has first priority.

WithADDMUX :=U68K;
memcycl:=0;
UDTACK1 :=1;
Endwith;

Else Idle_U68K2
WithADDMUX  := Lance;
memcycl := 1;
UDTACK1 := 1;
Endwith;

State Idle_U68K2:
If (!SRAMCS1) Then uCycle1      "68K has first priority.

WithADDMUX:=U68K;
memcycl:=0;
UDTACK1 :=1;
EDTACK1:=!(!EDTACK1 & !EAS1); "Sustain till end of EAS.
Endwith;

Else If (!EAS1 & clk10 & EDTACK1) Then eCycle1
                        "Lance has 2nd priority.
WithADDMUX:=Lance;
memcycl:=0;
UDTACK1 :=1;
EDTACK1 :=1;
Endwith;
```

1.1.3.C

Arbiter Exemplar Summary File

Exemplar Logic Synthesis System Thu Nov 19 13:58:14 1992

Pass	Area (modules)	Estimated Delay (ns)	CPU min:sec
1	77	46.3	00:30
2	85	55.2	00:32
3	81	47.1	00:30
4	81	47.1	00:31
5	81	46.3	00:30
6	117	55.2	01:31
7	81	47.1	00:31
8	81	47.1	00:32
9	81	47.1	00:31

Resource Use Estimate
Design: memarb
Technology:act2
File: memarb.pds
Area: 77.0
Critical Path: 46.3 ns

Device	Area avl/usd /pct	Registers avl/usd/pct	i/o avl/usd/pct
A1225	451/ 77/ 17%	341/ 0/ 0%	82/ 24/ 29%
A1240	684/ 77/ 11%	565/ 0/ 0%	104/ 24/ 23%
A1280	1232/ 77/ 6%	998/ 0/ 0%	140/ 24/ 17%

Delay Summary

Node:	Slack rise	Slack fall	Arrival rise	Arrival fall	Required rise	Required fall	Load
S4	: 31.00	15.30	15.30	46.30	46.30		1.60
S1	: 32.60	13.70	13.70	46.30	46.30		1.60
S3	: 32.60	13.70	13.70	46.30	46.30		1.60

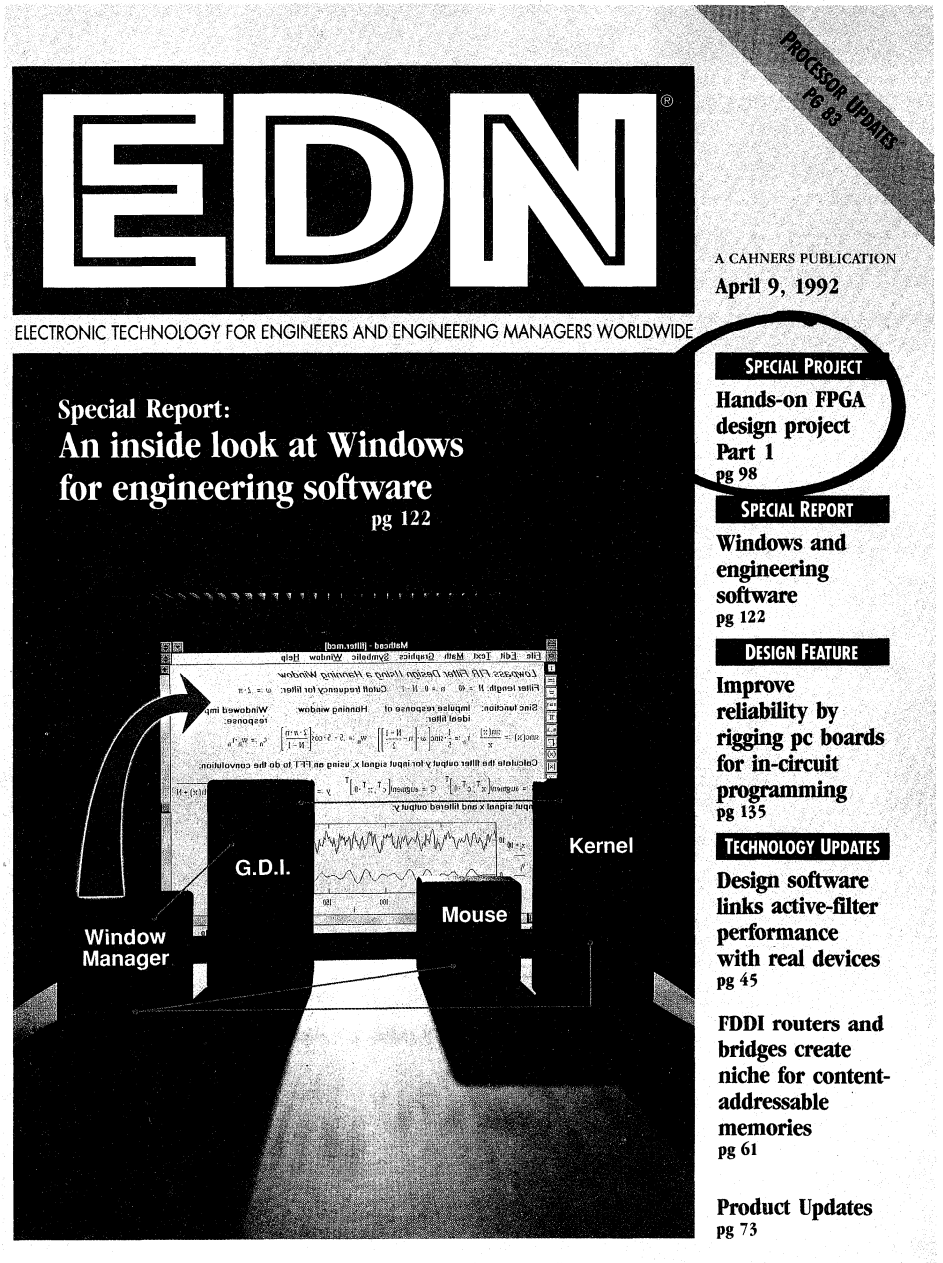
1.1.3.C

Cell Usage Summary

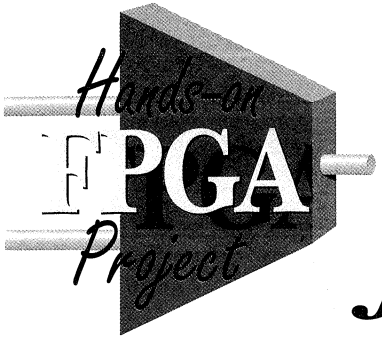
Cell	Uses	Cost	Total
AND2	4 uses(s)	1.00 modules	4.00 modules
AND2A	2 uses(s)	1.00 modules	2.00 modules
AND4A	1 uses(s)	1.00 modules	1.00 modules
AO1C	2 uses(s)	1.00 modules	2.00 modules
AO2	3 uses(s)	1.00 modules	3.00 modules
AO3	4 uses(s)	1.00 modules	4.00 modules
AO6A	2 uses(s)	1.00 modules	2.00 modules
AO7	2 uses(s)	1.00 modules	2.00 modules
AOI1	1 uses(s)	1.00 modules	1.00 modules
AOI2B	1 uses(s)	1.00 modules	1.00 modules
AOI4A	1 uses(s)	1.00 modules	1.00 modules
BUF	9 uses(s)	1.00 modules	9.00 modules
DFC1B	13 uses(s)	1.00 modules	13.00 modules
		Total =	77.00

Number of combinable modules is: 5

Taking the First Steps



EDN·SPECIAL PROJECT



Taking the first steps

If you're considering designing with FPGAs, this 2-part hands-on design project will show you exactly what is involved. Part 1 covers the design and schematic entry, and part 2 covers simulation and the functioning circuit.

DOUG CONNER, Technical Editor

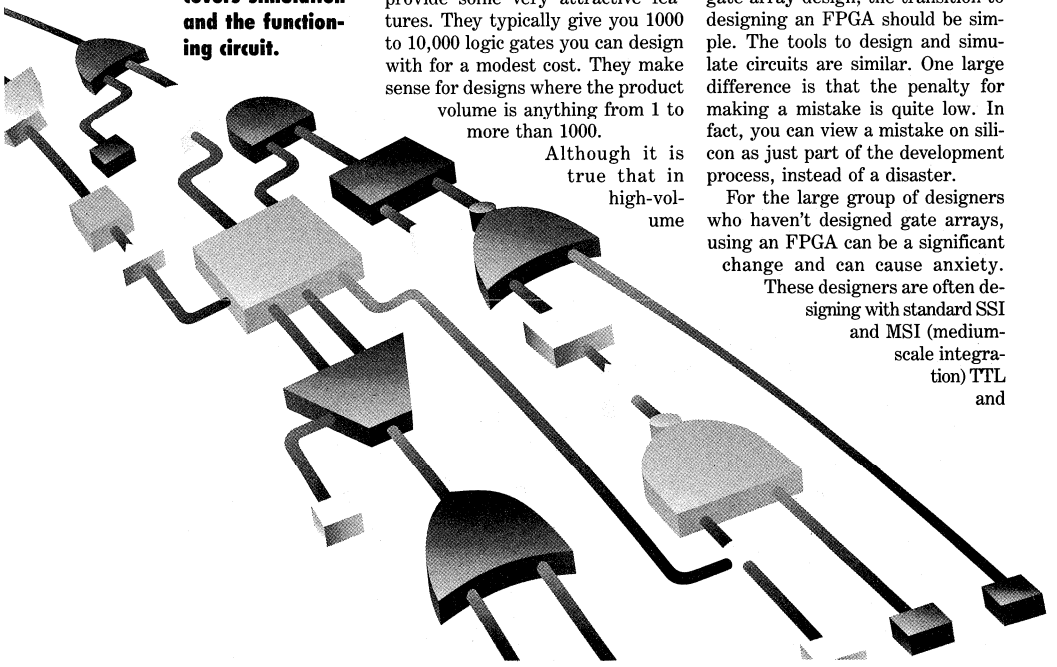
The fear and uncertainty of making a major shift in your design and development methodology is always compounded by tight schedules. As a result, you may be putting off designing with field-programmable gate arrays (FPGAs) because you don't know what to expect from them and you don't have the time to find out.

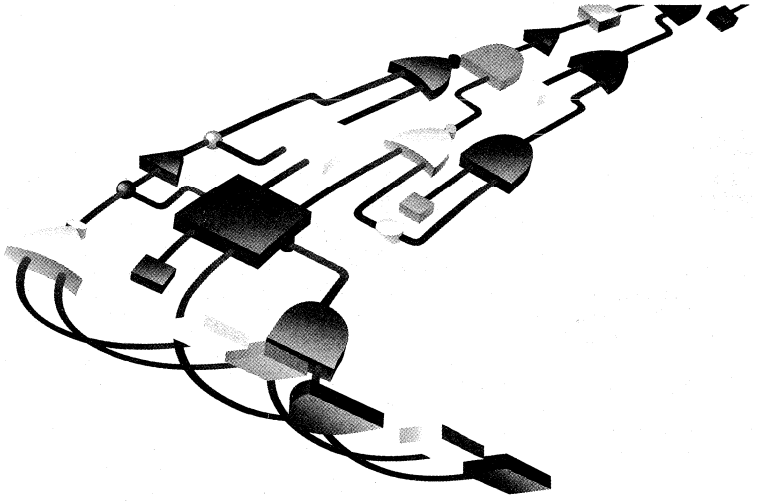
FPGAs and high-density PLDs provide some very attractive features. They typically give you 1000 to 10,000 logic gates you can design with for a modest cost. They make sense for designs where the product volume is anything from 1 to more than 1000.

Although it is true that in high-volume production a masked gate array can offer substantial savings, it is also true that they offer a much larger financial commitment up front. Penalties for an inexperienced designer who makes a design mistake or a system-definition mistake is high, both financially and in time lost in making another design turn.

For a designer experienced with gate-array design, the transition to designing an FPGA should be simple. The tools to design and simulate circuits are similar. One large difference is that the penalty for making a mistake is quite low. In fact, you can view a mistake on silicon as just part of the development process, instead of a disaster.

For the large group of designers who haven't designed gate arrays, using an FPGA can be a significant change and can cause anxiety. These designers are often designing with standard SSI and MSI (medium-scale integration) TTL and



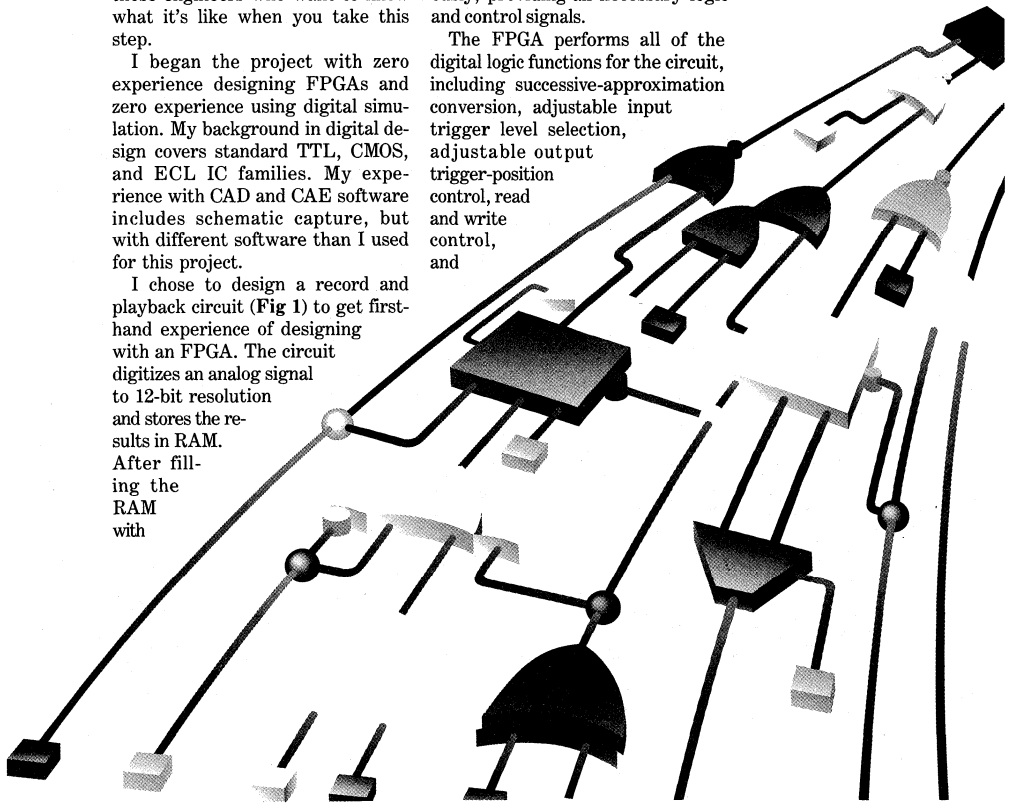


CMOS devices that interface to microprocessors, analog circuits, or both. Many have never used digital simulation. Moving to FPGAs is a step up for them. This project is for those engineers who want to know what it's like when you take this step.

I began the project with zero experience designing FPGAs and zero experience using digital simulation. My background in digital design covers standard TTL, CMOS, and ECL IC families. My experience with CAD and CAE software includes schematic capture, but with different software than I used for this project.

I chose to design a record and playback circuit (**Fig 1**) to get first-hand experience of designing with an FPGA. The circuit digitizes an analog signal to 12-bit resolution and stores the results in RAM. After filling the RAM with 32k words of data, it plays back the data, reconverting it to analog. The circuit is designed to work with an analog oscilloscope to capture a one-time event and play it back continuously, providing all necessary logic and control signals.

The FPGA performs all of the digital logic functions for the circuit, including successive-approximation conversion, adjustable input trigger level selection, adjustable output trigger-position control, read and write control, and



addressing the RAM. The design incorporates more than 1500 true logic gates and includes large regular structures, such as counters and compare circuits, plus plenty of gate-and register-level logic. (For a detailed circuit description and schematics for the full circuit, see **box**, "Pack the digital logic into one FPGA.")

Selecting the FPGA

I decided to use the Actel Act 1 FPGA family for my design. The choice of Actel was an arbitrary one—there are perhaps a dozen companies with products that fall into the FPGA and complex-PLD category that are appropriate for my design (**Ref 1**).

I chose the Act 1 family over Actel's higher performance and higher density Act 2 family because I didn't need the extra features. And, the Act 1 family costs less—the A1020A FPGA costs $36.25 (100).

To begin the project, I took Actel's 2-day training class. The class is included in the price of a system ($2950), or you can purchase it separately for $495. The class takes you through the process of designing an FPGA with Viewlogic schematic capture and simulation tools, and Actel's ALS software tools for all other functions. The basic design flow is shown in **Fig 2.**

The class uses canned files that you modify. For example, you'll add some components to a partially completed schematic to finish it. The class runs at a reasonably fast pace, but you won't fall behind even if you're unable to complete a step in the time allotted, because finished files are available. For example, if you haven't finished the schematic when it's time to move on to simulation, you can use a file that contains the completed schematic.

The class also covers some tools I didn't use in the project. A synthesis tool (ALES) lets you convert Boolean equations directly into logic. You can use the synthesized logic blocks in your schematic as you would use other macro symbols. Another tool, called the Timer, is a static timing tool that lets you look at path delays, both before layout and after place and route. At the end of the class you program an FPGA that contains a timing circuit and drives a 7-segment display.

Because Viewlogic CAE tools were used in the class, I elected to use them on the project, although Actel provides libraries and support for a variety of other workstation and PC-based tools.

Illustration by Gregg Dinderman

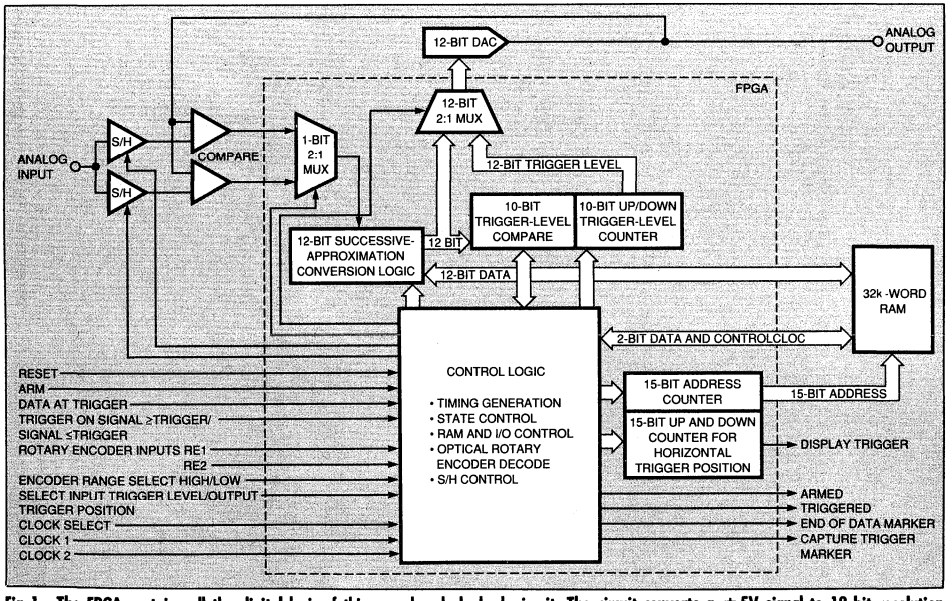


Fig 1—The FPGA contains all the digital logic of this record and playback circuit. The circuit converts a ±5V signal to 12-bit resolution at 167 ksamples/sec and plays it back continuously for viewing on an analog oscilloscope.

EDN-SPECIAL PROJECT

The building block on an Actel Act 1 FPGA is a logic module. What you actually design with is a logic module or group of logic modules configured as a hard or soft macro. A logic module starts as a flexible uncommitted block of logic; it can perform many different logic functions depending on how its connections are programmed. Actel provides hard macros, which define the logic-module connections to perform specific functions.

The hard-macro building blocks for designing an Actel FPGA are gates, gate combinations, latches, flip-flops, multiplexers, adders, and buffers. You can also configure every I/O pin as an input buffer, an output buffer, a bidirectional buffer, or a 3-state buffer. One input pin is designated as a clock buffer. You can see many of the basic building blocks and variations on pages of the circuit schematic (see **Figs 4** to **15**, which begin on pg 107).

Designing with the FPGA building blocks is similar to designing with 7400 series SSI devices, except in most cases the FPGAs are more flexible. For example, 2- and 3-input AND gates are available with any or all of their inputs inverted. You can select D flip-flops with positive clear, negative clear, and so on. Every gate macro I used requires a single module. Even a relatively complex gate combination, such as the 4-input AND/OR gate shown in **Fig 15**, is a single module. Although there are a few combinations that require two modules, I was able to avoid using them.

Latches also require only one module, even with a clear, an enable, or multiplexed inputs. Flip-flops, however, require two modules. In cases where a latch will work as well as a flip-flop, the module savings makes the latch a better choice. For example, the circuit needed to generate the DLY shown at the bottom of **Fig 6** uses two latches instead of flip-flops.

Another gate-saving consideration is to use multiplexed data inputs on both latches and flip-flops to bring 2-input gates inside them. The result saves a module. For example, the latch generating DISP_TRIG in **Fig 6** effectively ANDs together DISP_TM and PLYBK.

Part way through the design, I learned that the ALS software automatically combines 2-input gates with flip-flops and latches wherever possible. Therefore, you can see cases where I've left the gate separate, such as the latch and AND gate in **Fig 15**. The schematic is easier to read with the AND gate separate, so I'd recommend letting the software do its job. The end result on the FPGA is the same.

When your design calls for larger blocks (such as counters, adders, multipliers, decoders, and large registers), you've got several choices. You can use a soft macro if one exists, alter one if it's close but not quite what you need, or build what you want from scratch. The soft-macro library includes a wide selection of functions.

For example, you can select an adder with 8-, 12-, 16-, 24-, or 32-bit capacity. The soft-macro library also includes macros that are equivalent to some MSI TTL circuits. For example, the 8-bit up and down synchronous counter with rip-

Text continued on pg 104

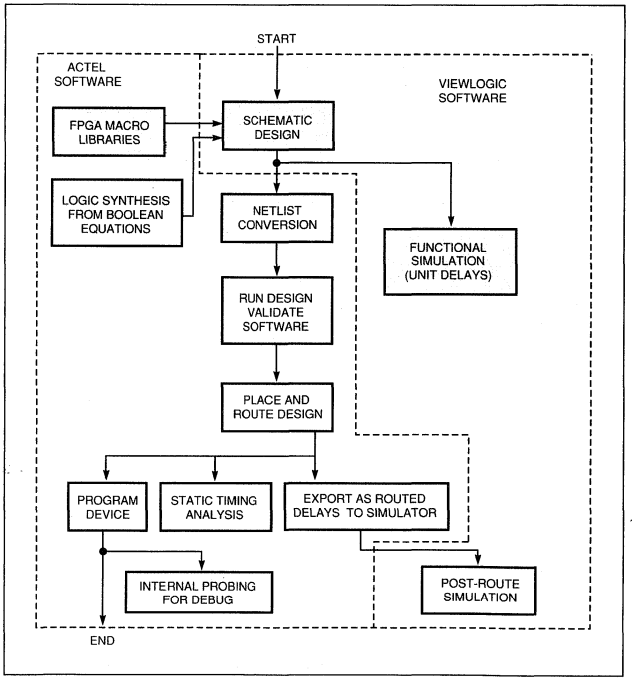


Fig 2—The total time for running netlist conversion, design validation, place and route, and exporting as routed delays took about half an hour for the design. This relatively fast turnaround lets you make quite a few design iterations in one day. The logic synthesis and internal probing for debug were not used on the project.

Pack the digital logic into one FPGA

When I decided to design an FPGA (field-programmable gate array) and write about it, I wanted to use it in a circuit with a minimum of other parts, yet I wanted the circuit to be moderately complex so that it would be a true test of designing with an FPGA. The record and playback circuit I chose packs all the digital logic into the FPGA, and the only other parts it requires are RAM and a few analog ICs (see **Figs 4** to **15** beginning on pg 107).

The top-level schematic for the overall circuit is shown in **Fig A**. The circuit uses the same 12-bit DAC and op amp for successive-approximation conversion during record and for generating the analog output during playback. Because conversion and playback use the same DAC, the gain and offset errors of the DAC and op amp do not add to the system error.

During conversion, the circuit compares the DAC's current output, converted to voltage by a high-speed op amp, with the sampled input voltage. The comparator output drives the successive-approximation logic. Two parallel paths alternately sample and compare the input against the DAC output. The alternating approach saves both the sampling time and the hold-settling time. Each bit decision takes 500 nsec, providing a complete 12-bit conversion every 6 μsec.

The design depends on closely matched offsets in each of the two S/H and comparator paths. You can expect close matching because both comparators are on the same monolithic IC. The same is true for the S/H channels.

Gain accuracy of the circuit depends on the gain accuracy of the S/H circuit and on the comparator's CMRR. The AD684 provides a worst-case gain error of ±5 mV over the ±5V input range. The LT119A used in the circuit has a minimum CMRR of 90 dB, contributing less than a 0.4-mV error over the ±5V input range. Although the LT119A used in the circuit has a minimum CMRR of 90 dB at dc, the CMRR is not specified at the 2-MHz frequency of the design. In fact, depending on high CMRR at frequency is risky, and generally frowned upon by knowledgeable analog designers. In this design I felt the risk was justified by being able to use one DAC for both record and playback.

The digital part of the circuit has four basic states (**Fig 5**): clear memory, armed, triggered, and playback. Playback is the default state when the circuit is reset. The other three states are also ORed together in the circuit to form the recording state (RECD).

To start recording, you depress the momentary arm switch to initiate the clear memory state. The clear memory state starts writing A/D conversions from the successive-approximation conversion into RAM, but disables the trigger until you fill the entire memory with new data, writing zeros to D13 and ones to D14. After you overwrite the entire memory, the state changes to armed, and the circuit continues to record data until the trigger logic is satisfied. Once triggered, the state changes to triggered (TRIGD) and the circuit converts 24,000 more samples, stores them in memory, and returns to the playback state.

You set the trigger level using a rotary encoder to adjust a 10-bit up-and-down counter (**Fig 14**). The logic performs a 10-bit magnitude compare (**Fig 15**) of the successive-approximation converter output with the trigger level to determine when to trigger the circuit.

The trigger-level compare is a full-magnitude compare that tests whether the digitized input signal is greater than or equal to the trigger-level setting or less than or equal to it, depending on the input (TRIG_GE). The 10-bit range provides a trigger-level resolution of 10 mV and gives time for the magnitude-compare results to become valid while the successive approximation is finishing the last two bits.

Control logic (**Fig 10**) also generates the RAM write enable (N_WE), the RAM output enable (N_OE), and the FPGA's output enable (F_OUT). **Fig B** diagrams the basic record and playback timing.

A 12-bit shift register (**Fig 4**) generates the 12 timing states needed for the successive-approximation conversion. These timing signals are also used to control all timing-related logic in the FPGA.

A clock-select circuit lets you select between two clocks. The circuit can play back the data at a much higher rate than it can during recording, because the DAC only changes state once every 12 clock cycles during playback.

Successive-approximation conversion

The A/D conversion starts with sampling and then holding the input. The timing generator uses a 12-bit shift register to control the 12 states of the successive-approximation conversion. I created a macro, called SAR, for the conversion and used one for each bit (**Figs 7** and **8**). The details of the macro are shown in **Fig 3**.

The conversion starts at the beginning of the T1 cycle, DAC data inputs are reset to a low state, except the MSB, which is set high. The correct analog-comparator input is multiplexed to the successive-approximation logic, and near the end of the T1 cycle, the global clock signal (GCLK) clocks in the comparator's output state. At the beginning of cycle T2, the next bit, DAC2, is set high, and driving the MSB remains in the state latched in at the end of T1. The conversion process continues in a similar manner through T11 and the 11th bit. The LSB is slightly different. Near the end of T12, the FPGA will write all 12 bits to the RAM. For this reason the data for the LSB comes straight from the comparator without being clocked into the flip-flop.

When in the playback state, the DAC receives data

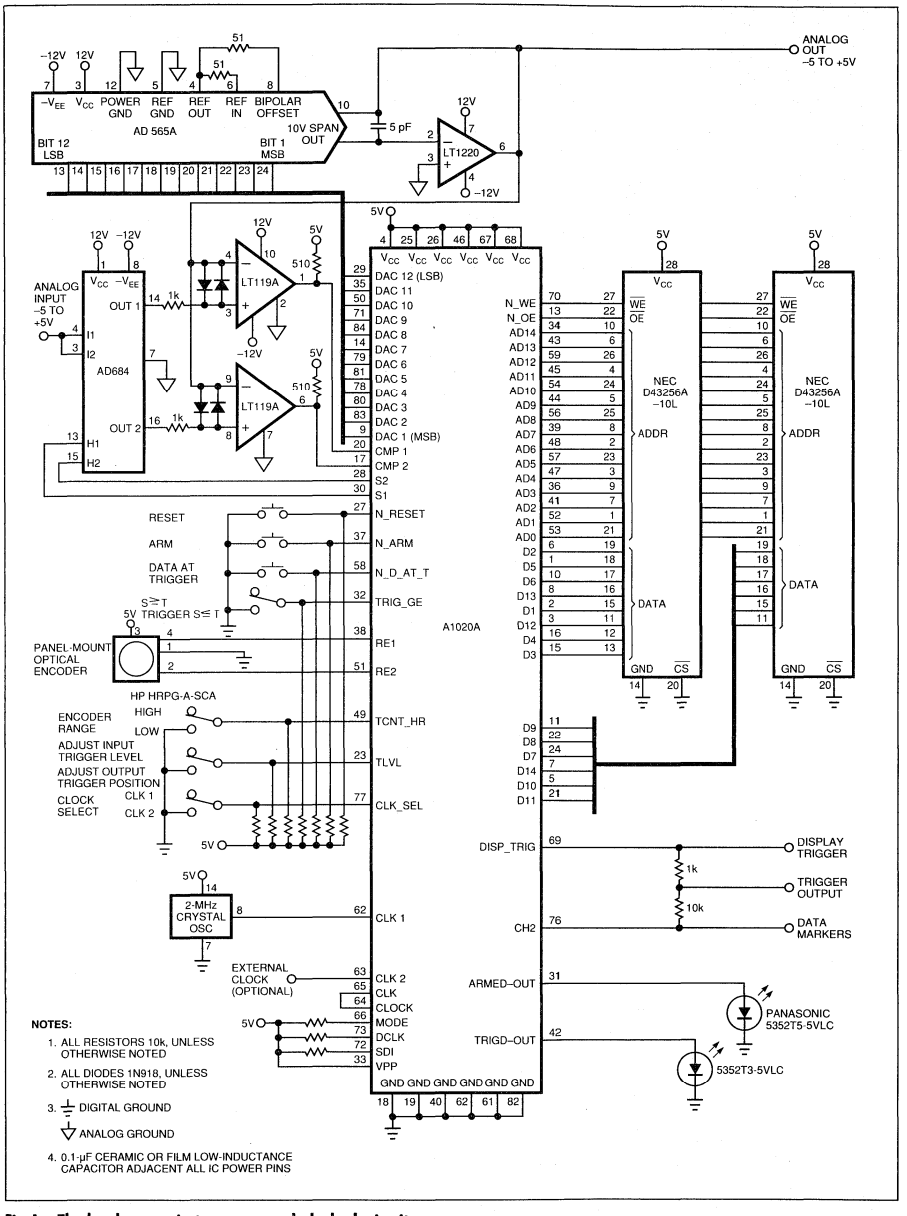


Fig A—The hands-on project was a record playback circuit.

ple carry used in **Fig 12** performs the function of a 74269.

When you need something a little different from the stock parts, the flexibility of a soft macro really shines. Unlike hard macros, which you cannot alter, you can copy and then alter soft macros to perform exactly the function you want. In fact, any time during the design that you want to see what is schematically in the guts of any soft macro, you just select the device and push down into the next level of the hierarchy.

The device labeled CNT 128 in **Fig 13** is a 7-bit version of the TA269 in **Fig 12**. I created CNT 128, my first soft-macro conversion, in approximately 10 minutes. Now that I know how, it should take less than 5 minutes. It really is that simple. All you do is copy and rename the macro's schematic and symbol, then make the modifications to the new schematic and symbol. When you want to use the new function, you call up the symbol and put it on your schematic. You can find other customized soft-macro examples in the schematic, such as 8-bit counters (**Fig 10**) and 7-bit latches (**Fig 12**).

Making a custom macro takes a little longer than merely modifying an existing macro because you need to create the full schematic and symbol. However, it isn't really any more difficult. SAR, used in **Figs 7** and **8**, is a custom macro I created to save a few pages on the schematic. The schematic for the macro is shown in **Fig 3**.

You don't necessarily have to modify a standard soft macro if you don't need all of it. The rule is that

Pack the digital logic into one FPGA *(continued)*

from the RAM and clocks it into the flip-flops at the end of cycle T12. A multiplexer switches the trigger-level setting (TL1-TL10) into the DAC input when the adjust trigger-level signal (TLVL) is asserted.

The 32k-word RAM stores conversion data from the successive-approximation conversion, plus two control signals (D13 and D14) (**Fig 6**). A 15-bit counter generates addressing for the RAM. While in record mode, the address counter is free running. The FPGA continuously writes the A/D results into RAM. When the trigger-level compare condition is satisfied by the incoming signal, the current value of the address counter is latched, the 15-bit up-and-down horizontal trigger-position counter is loaded, and the memory-trigger

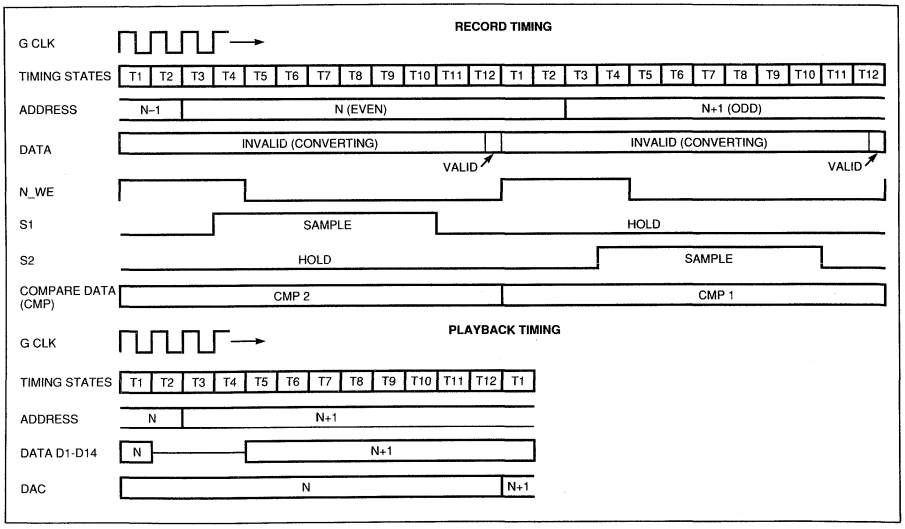


Fig B—At the end of each A/D conversion, the FPGA writes the data to RAM. During playback the FPGA latches data from RAM at the end of T12 to drive the DAC.

EDN-SPECIAL PROJECT

you can't leave any unused inputs—all inputs must be tied to a signal, V_{CC}, or ground. You may leave outputs unused; the software should remove any unnecessary logic associated with the unused outputs. The software will issue a warning whenever an output is unused, giving you a chance to verify that the omission is intentional.

You shouldn't tie unused inputs to V_{CC} or ground if it's possible to eliminate them. The flip-flops in **Fig 10** should be changed to macros without the preset. CNT4B on **Fig 6** loads all zeros. A more efficient design would just use a Clear and eliminate the load function on the counter. Even if the change doesn't result in a module savings, unnecessary inputs tied to power and ground restrict routing flexibility, which might affect the overall performance of the circuit.

Fan-out limits are perhaps the most noticeable change from standard TTL design. The software gives you a warning for more than 10 loads, and an error for more than 24. For the special cases of nets you designate as "fast criticality" (I'll discuss criticality in part 2), the fan-out limit drops to six loads. The only exception is the global clock signal. There is only one global clock signal on ACT 1 devices, and it can drive any number of loads.

On the surface, these fan-out limits may not seem too stringent, but you have to remember that macros are just a graphic convenience, no signal buffering occurs unless you put it inside the macro.

For example, the latch-control input of the 8-bit latch shown in **Fig 12** is eight loads, not one. You'll note a buffer in front of it. In fact, you'll see quite a few buffers scattered throughout the pages of the schematic.

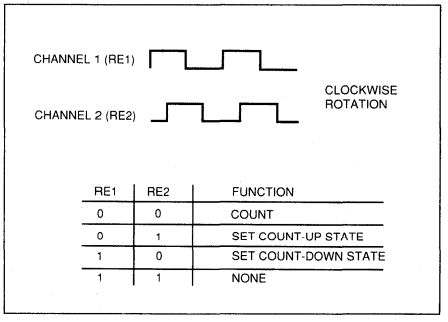
Buffers are easy to add, and the software errors and warnings tell

match condition (N_MEM_TM) is set up to stop acquisition after recording 24,000 more words of data, providing 8k words of pretrigger data. The FPGA writes the display-trigger match (DISP_TM) bit to RAM (D13) to mark the capture-trigger location during playback. N_MEM_TM is also recorded in RAM (D14) to mark the beginning and end of data.

During playback, the circuit compares the address counter with the 15-bit horizontal-trigger-position, up-and-down counter. When the two 15-bit words match the display, trigger-match condition (DISP_TM) is satisfied, and the display-trigger signal (DISP_TRIG) goes high for 12 clock cycles to drive an oscilloscope trigger. Initially, the circuit sets the horizontal-trigger position counter to the capture-trigger address. Subsequently, you may change it with the rotary encoder to trigger the oscilloscope at any address.

The address counter runs continuously during playback except when interrupted by one of two events:

- When the data at trigger signal (N_D_AT_T) goes low, the address counter is disabled the next time it reaches the display-trigger match and stays disabled until N_D_AT_T goes high. With the address counter disabled, the data going to the DAC is frozen so that the DAC continually outputs the voltage at the display trigger address. You can read the dc voltage on the analog output with a voltmeter.
- When D14 from the RAM goes low, indicating the end of the data, the address counter is disabled for 16 periods of 12 clock cycles each, before repeating the data. A high signal on the CH2 output provides a marker to use on an oscilloscope to indicate the beginning and end of data.



RE1	RE2	FUNCTION
0	0	COUNT
0	1	SET COUNT-UP STATE
1	0	SET COUNT-DOWN STATE
1	1	NONE

Fig C—The counter either counts up or down one cycle each time both encoder outputs are low. The count direction is determined by the previous state of the encoder.

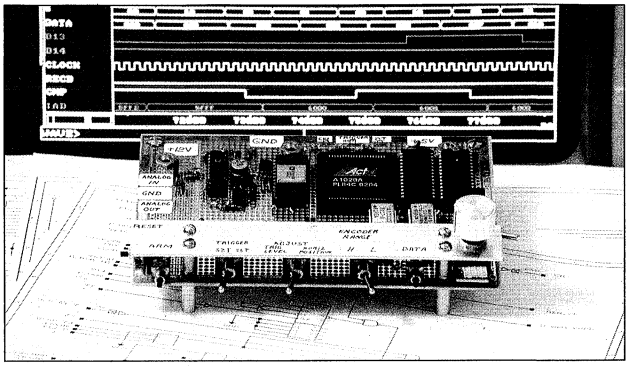
Logic shown in **Fig 9** decodes the quadrature signals (**Fig C**) from the panel-mount, rotary, optical encoder (RE1 and RE2) into count-up and -down signals and count-enable signals. A panel-mount switch lets you select between adjusting the input-trigger level and adjusting the output horizontal-trigger position.

Because the horizontal-trigger position covers a 15-bit range (32k-word address) a high- and low-range select lets you count in increments of 1 or increments of 256 addresses. The rotary encoder provides 120 quadrature cycles per revolution, so using the low range you'd need to turn the knob 273 revolutions to scroll the full address range. Using the high range, you can scroll the whole range in just over one revolution.

you where they are needed. Nonetheless, they are a minor nuisance and one of the few blemishes to what I consider a nearly ideal design environment. Of course, the addition of buffers should remain under the designer's control and not be made automatic because buffering is more than just a cosmetic change to the schematic.

Buffers require a module and add a module delay to the signal (In part 2, I'll discuss timing in detail). Letting module fan-out increase above the warning limit can cause large time delays too. For my particular design, the timing was not too tight, so I just added buffers as needed to eliminate errors and warnings. I probably could have left the warnings and still been okay. If your design has tight timing and you can't afford extra module delays, you can regenerate the signal.

For example, in **Fig 10** you'll find F_OUT and a buffered version F_OUT_A. Had this signal been timing critical, I could have cloned the preceding flip-flop to generate two identical versions of the signal without any additional module delays. The cost in this case would be an extra module because the flip-flop hard macro requires two modules, compared with the single module for the buffer. Also it means doubling the load on the flip-flop's input signals because they'll be driving two flip-flops instead of one.



Getting from a concept to a finished circuit using an FPGA requires learning new software. Even though all the software was new to me, I learned to use it effectively for this project in less than a week.

On the overall schematics (**Figs 4** to **15**) I've only labeled nets where I needed to for design reasons, with very few exceptions. One exception is IA0 in **Fig 11**. It's labeled for simulation reasons I'll discuss in part 2. In future designs, however, I plan to label every net and every module. Although labeling takes time, it pays off when simulating, using the static timing analysis software, and reading error reports. I have to note that several people recommended labeling everything, and I ignored the advice. In the end, I didn't save any time by omitting the labels. You can take my advice or learn the way I did.

After reading this far you may have come to the conclusion that designing an FPGA is not much different from designing with SSI and MSI ICs. That's my conclusion too. I spent my time during schematic design battling with system design issues and how to improve the design, not fighting with tools or wondering if the clever use of a different MSI device would make a cleaner design. In part II, I'll show you some of the bugs I caught in simulation and two that I didn't catch until I tested the circuit. I'll also present you with the chronological account of the project so you can see how much time I spent in each step. EDN

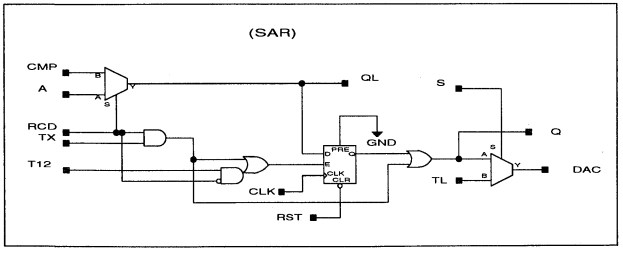


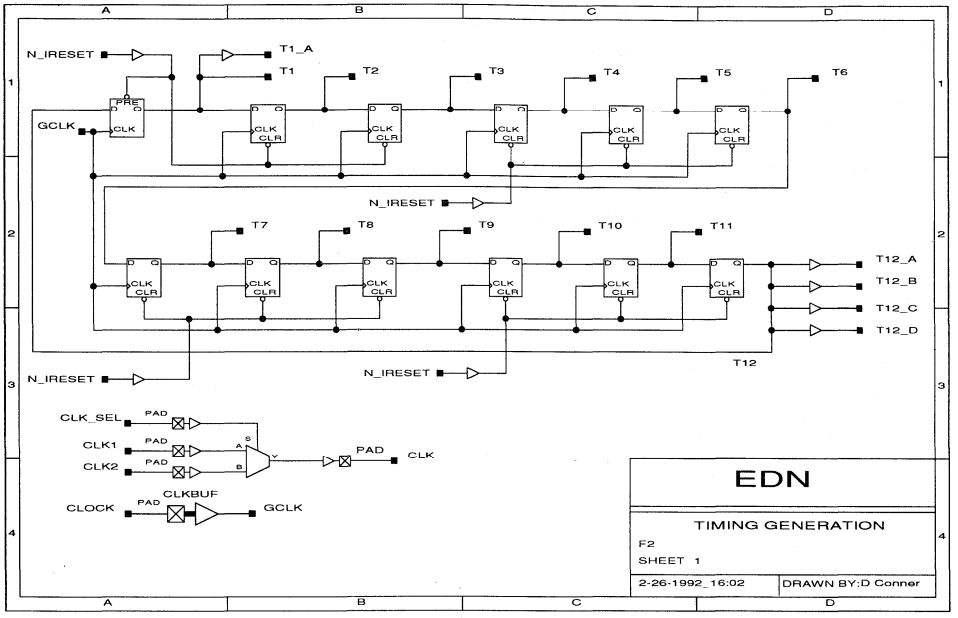
Fig 3—Creating custom macros is very simple and simplifies the schematic of the overall design. SAR is used in Figs 7 and 8 of the FPGA schematic. If you need to make changes to the design later, you need to change only the macro.

Reference

1. Conner, Doug, "High-Density PLDs," *EDN*, January 2, 1992, pg 76.

Doug Conner is a technical editor for EDN based in California. You can reach him at (805) 461-9669.

EDN-SPECIAL PROJECT



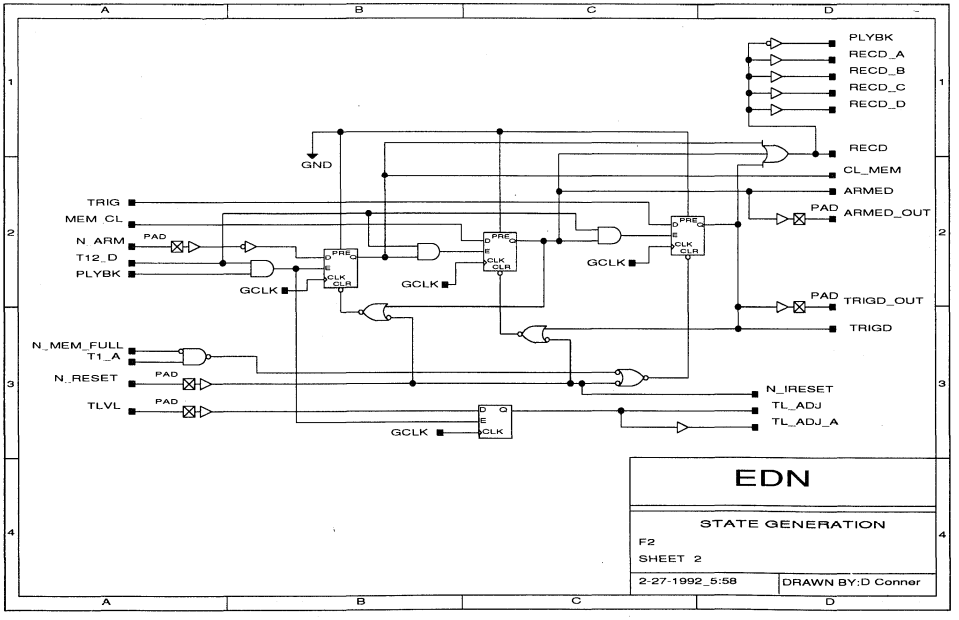


Fig 4 (top)—The schematic shows the logic for generating the 12 timing states used for all record and playback operations; Fig 5 (bottom) shows the logic for generating the playback state and the 3 record states: clear memory, armed, and triggered.

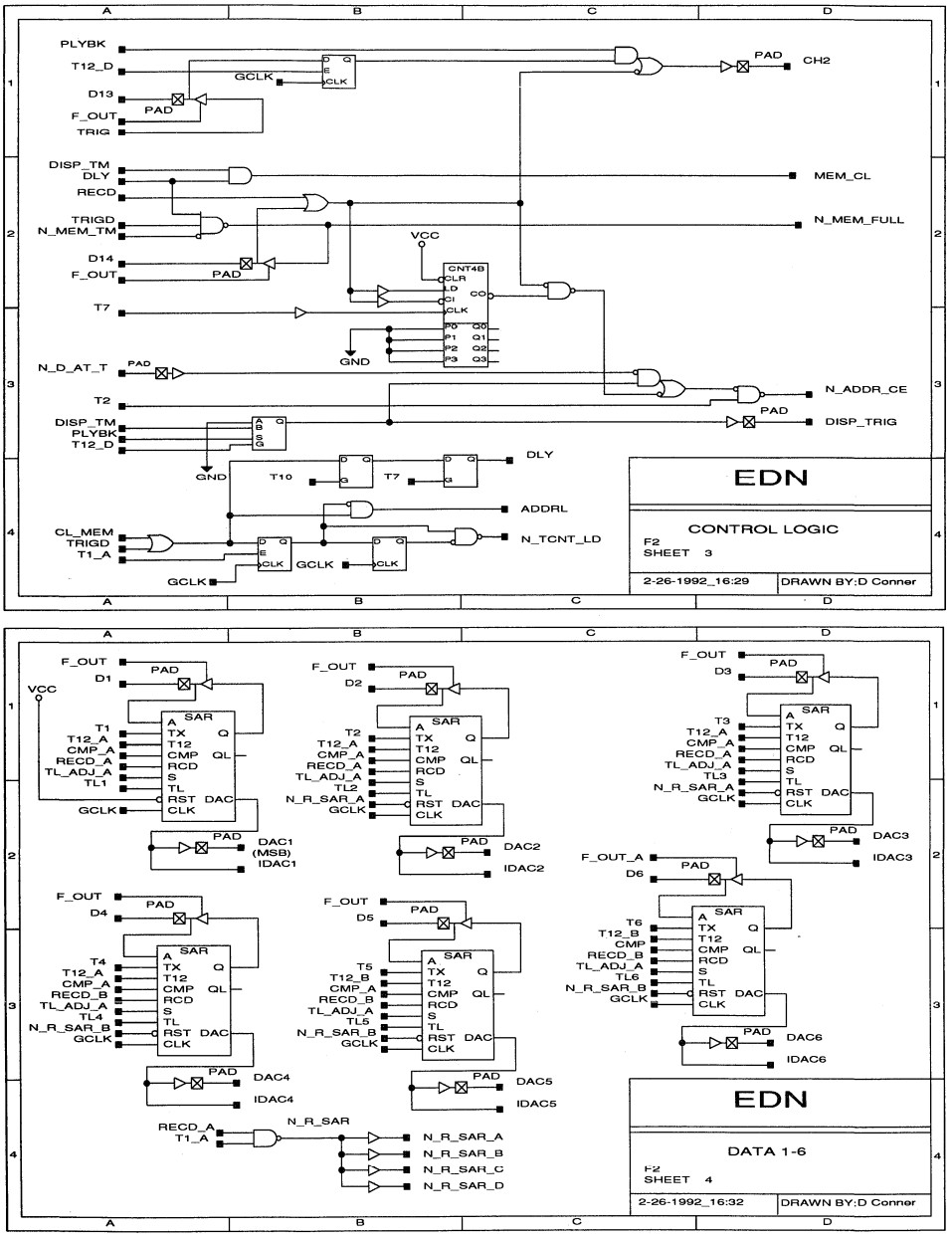


Fig 6 (top)—Logic for miscellaneous control functions is illustrated in this diagram; Fig 7 (bottom) shows the logic for the lower six data bits (see Fig 3 for SAR macro schematic).

EDN·SPECIAL PROJECT

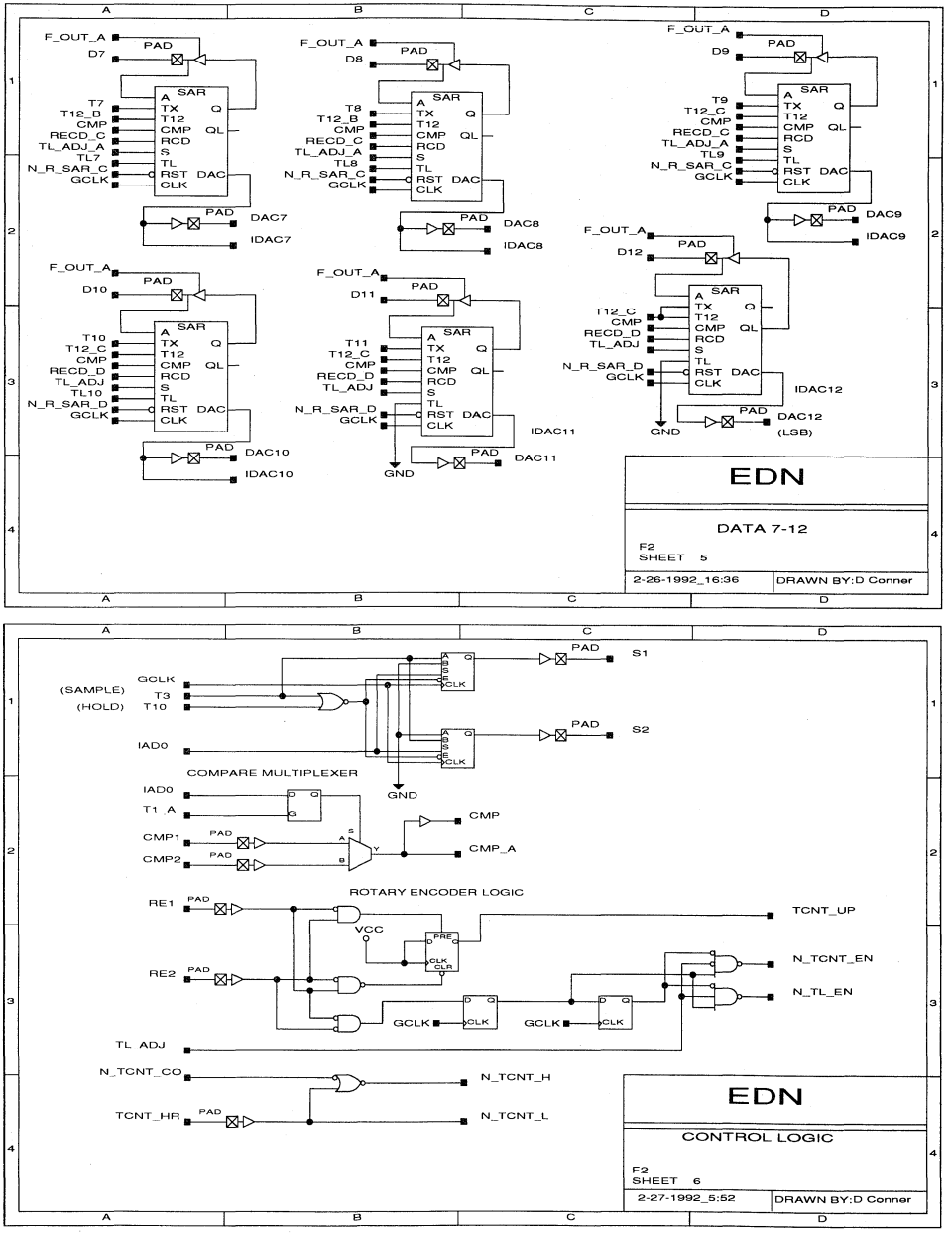
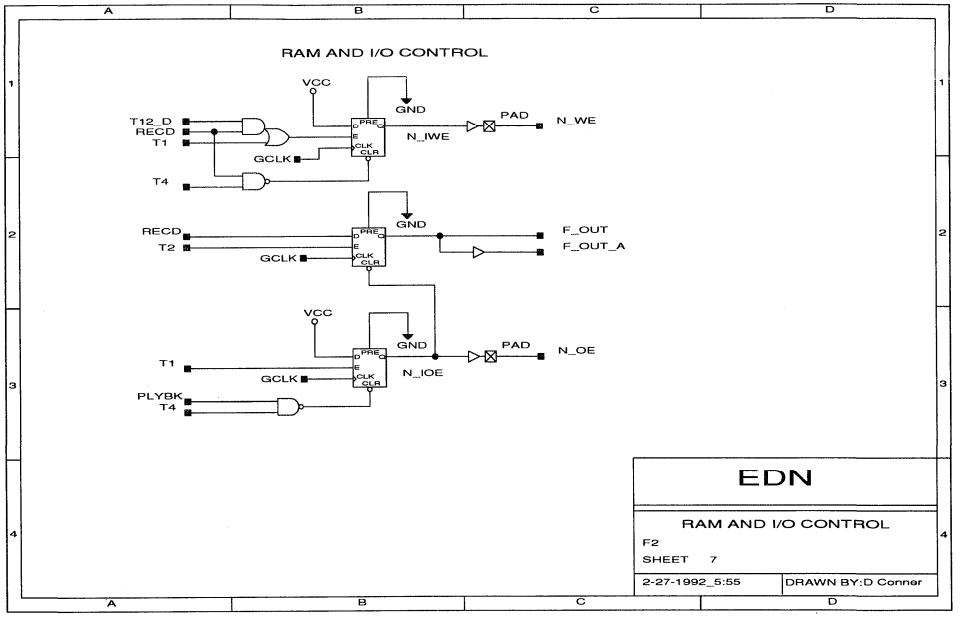


Fig 8 (top)—This schematic details the logic for the upper six data bits (see Fig 3 for SAR macro schematic); in Fig 9 (bottom), you can see the control logic for the S/H circuit, compare multiplexer, and rotary-encoder decode logic.



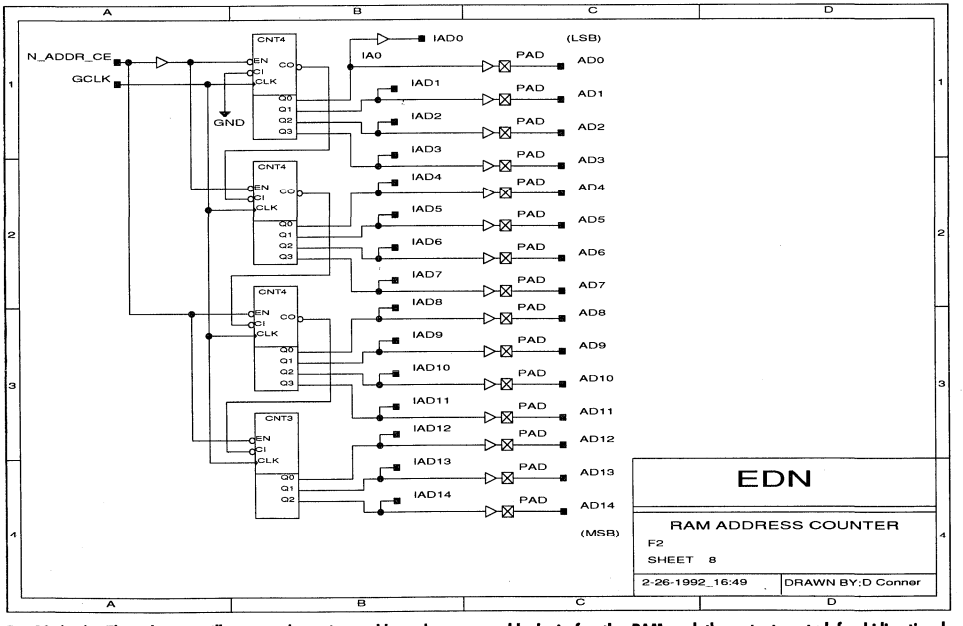
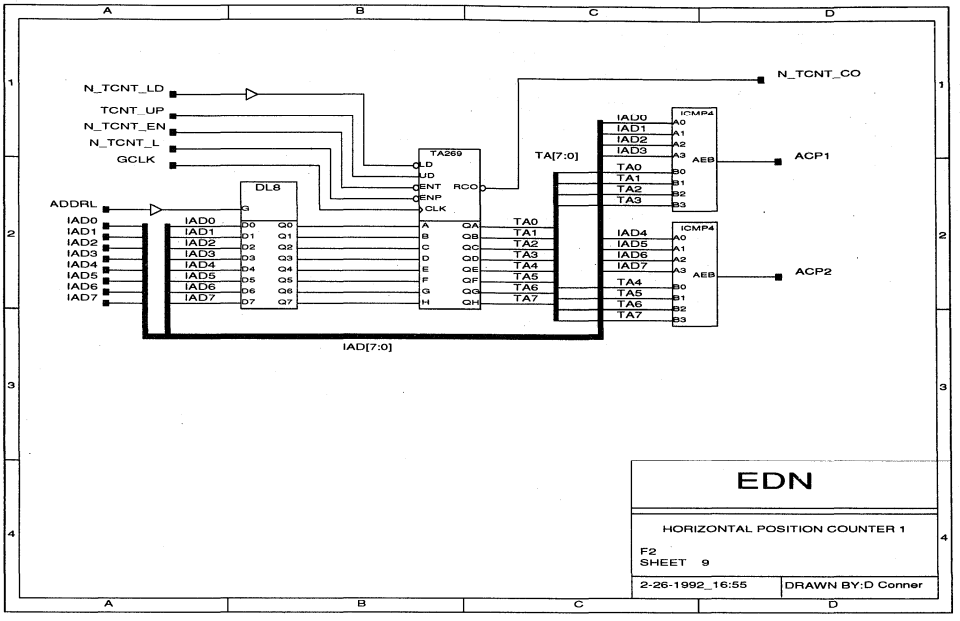


Fig 10 (top)—The schematic illustrates the write-enable and output-enable logic for the RAM and the output control for bidirectional data lines; Fig 11 (bottom) highlights the 15-bit counter for RAM address lines.

EDN-SPECIAL PROJECT



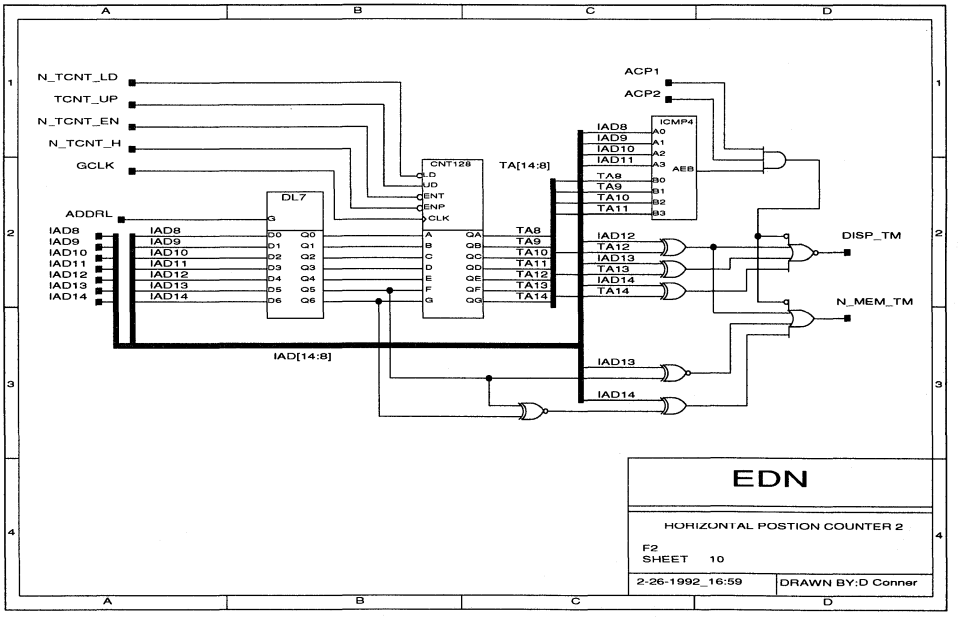
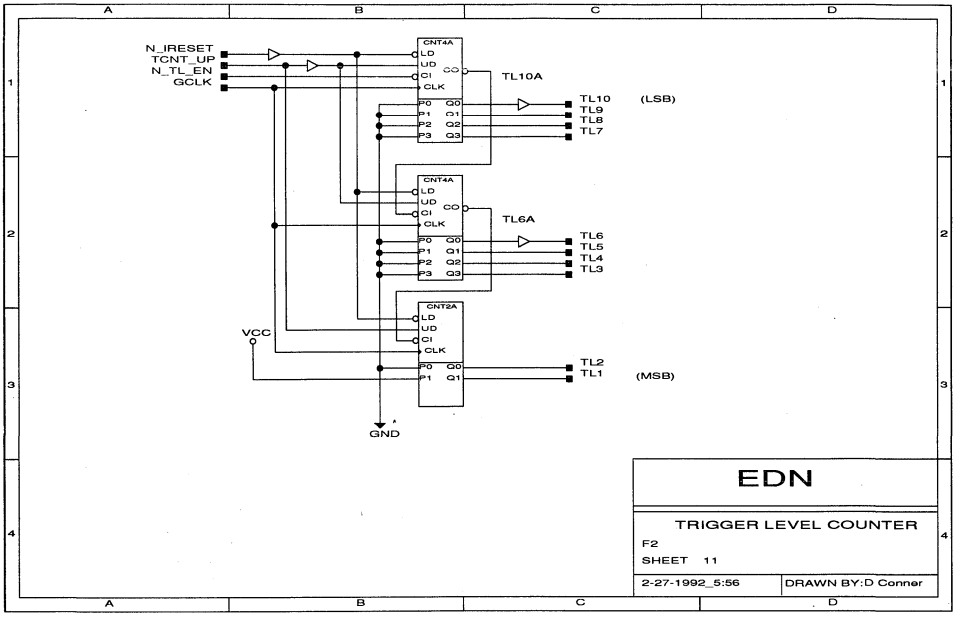


Fig 12 (top)—The lower eight bits for horizontal-output position control are illustrated here; Fig 13 (bottom) shows the upper seven bits for horizontal-output position control.

EDN-SPECIAL PROJECT



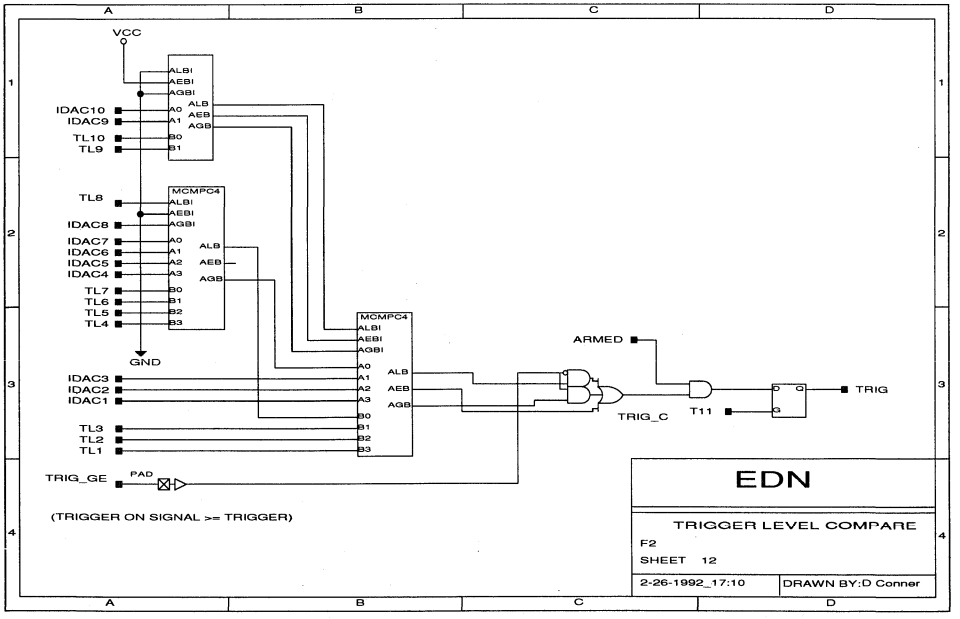
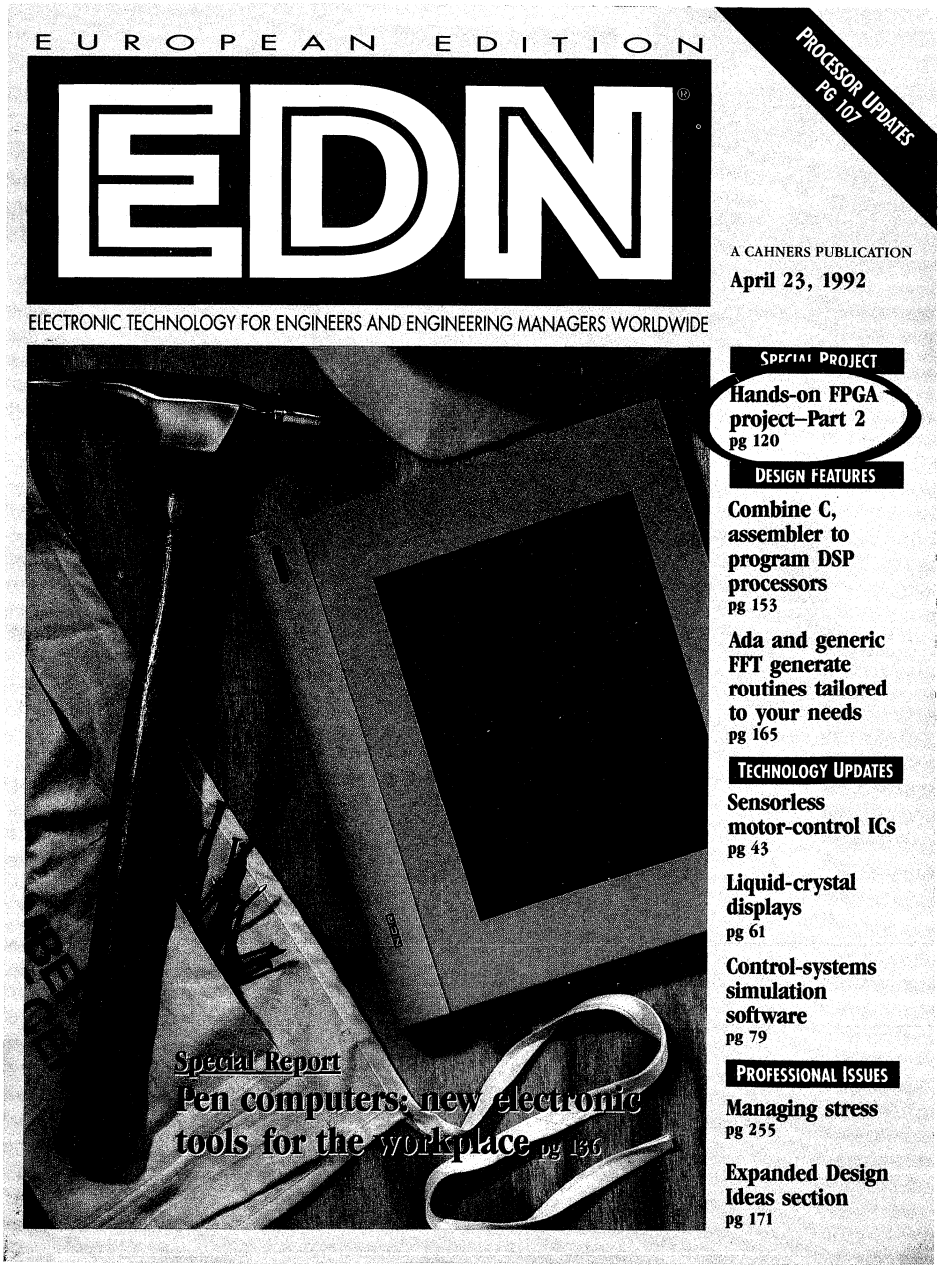
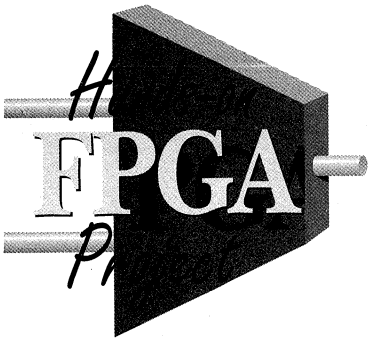


Fig 14 (top)—This schematic shows the 10-bit counter for setting the input trigger level; Fig 15 (bottom) illustrates the 10-bit magnitude-compare circuit for detecting the input-trigger event.

Taking the First Steps



EDN·SPECIAL PROJECT



Migrating to FPGAs: *any designer can do it*

Part 1 of this 2-part hands-on design project (in the April 9 EDN) discussed the overall circuit and the schematic entry of this FPGA (field-programmable gate array) design project. Part 2 concentrates on the steps from simulation to the final functioning circuit. The project took 29 working days from start to finish.

DOUG CONNER, Technical Editor

In the April 9 issue I discussed the first part of my journey into FPGA design. The schematic part of the design wasn't much different from ordinary SSI or MSI TTL design. Because logic simulation is seldom used in SSI or MSI TTL design, it was a new experience for me and was the greatest worry when I started the project. If you haven't been using logic simulation, you'll find this phase of verifying your FPGA design a big change.

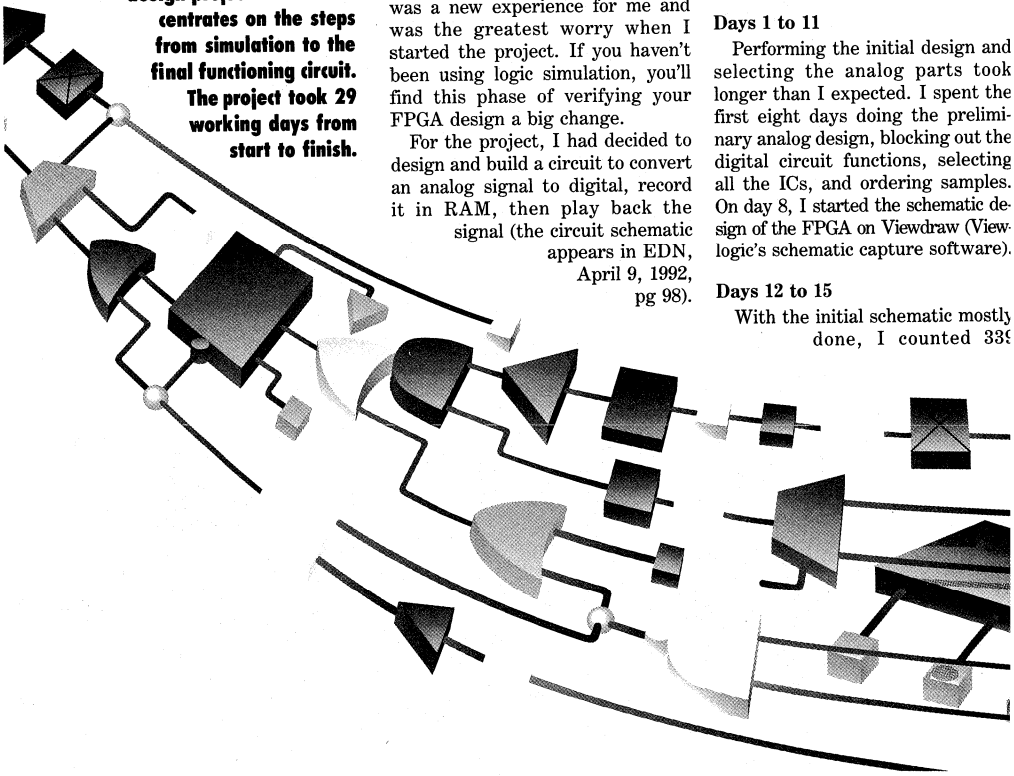
For the project, I had decided to design and build a circuit to convert an analog signal to digital, record it in RAM, then play back the signal (the circuit schematic appears in EDN, April 9, 1992, pg 98). All digital functions would be performed in the FPGA. I wanted to keep the analog portion of the circuit simple, yet be as fast as possible. I began the project by blocking out the design and figuring out what the overall circuit should do.

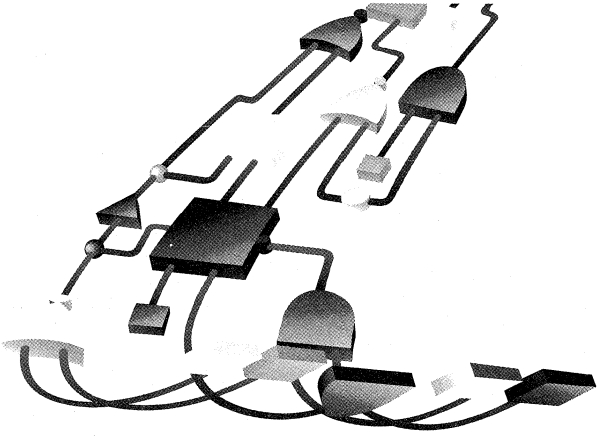
Days 1 to 11

Performing the initial design and selecting the analog parts took longer than I expected. I spent the first eight days doing the preliminary analog design, blocking out the digital circuit functions, selecting all the ICs, and ordering samples. On day 8, I started the schematic design of the FPGA on Viewdraw (Viewlogic's schematic capture software).

Days 12 to 15

With the initial schematic mostly done, I counted 33





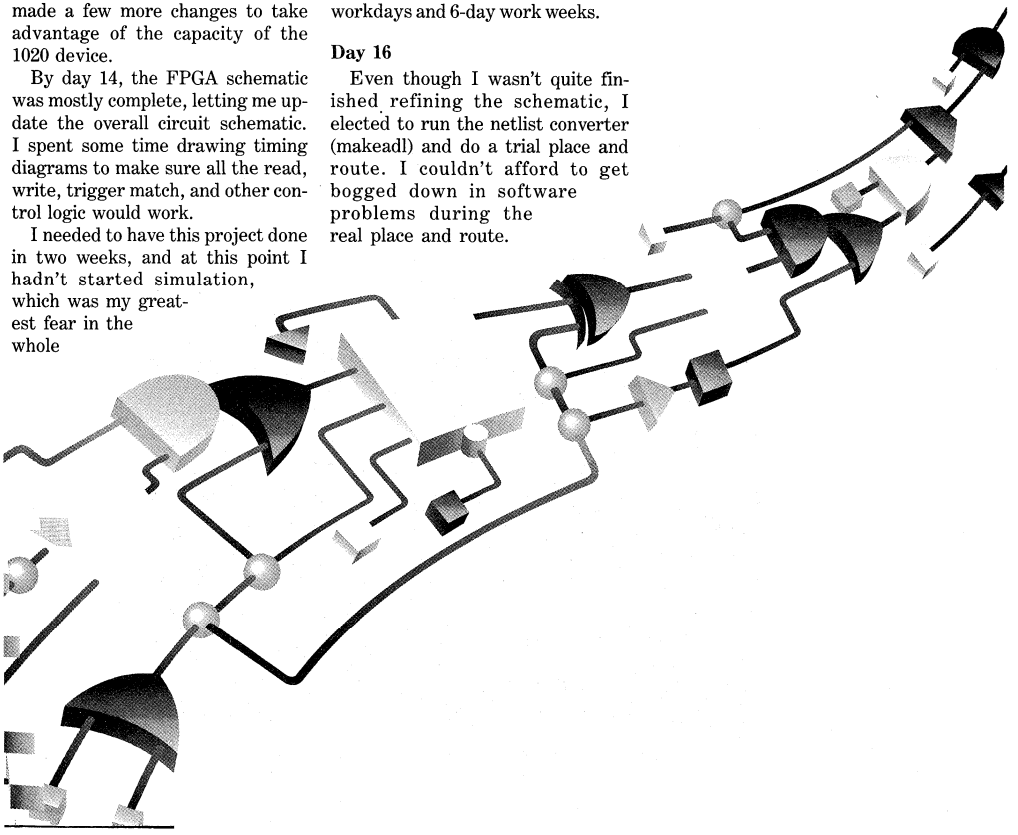
used modules. That's too big for the 295 modules on an Actel 1010 device and only 60% of the 547 modules on a 1020 device. I decided to bring triggering inside the FPGA and made a few more changes to take advantage of the capacity of the 1020 device.

By day 14, the FPGA schematic was mostly complete, letting me update the overall circuit schematic. I spent some time drawing timing diagrams to make sure all the read, write, trigger match, and other control logic would work.

I needed to have this project done in two weeks, and at this point I hadn't started simulation, which was my greatest fear in the whole project. The schematic-entry tools were working fine, but I kept finding design problems that had to be solved or improvements I considered necessary. I initiated 12-hour workdays and 6-day work weeks.

Day 16

Even though I wasn't quite finished refining the schematic, I elected to run the netlist converter (makeadl) and do a trial place and route. I couldn't afford to get bogged down in software problems during the real place and route.



I wanted to find them immediately and get them out of the way.

The netlist conversion is a single command and takes about five minutes to run. The Validator software checked the design and the computer returned the errors and warnings: I had one incorrectly named net, four fan-out errors, and ten fan-out warnings. I corrected the errors and went on to try an auto-pin placement.

The autopin software under the Pin Edit menu wouldn't run because it needed a file named *design*.pin, which I hadn't created yet. I put in a call to Actel, but before they returned my call, I tried some experiments. Under the configure selection on the menu I found an automatic I/O assignment selection. Reviewing the documentation, I learned that this appears to be the right menu selection to use to do the automatic I/O pin assignment. I tried it, and it worked. After you run the automatic I/O pin assignment once, the *design*.pin file is created. After that I could edit the pin assignments if I wanted to.

I left the automatic I/O pin assignments because they should let the automatic place-and-route software do its best job. The first time I ran the place-and-route software, it produced a fully placed-and-routed design in 16 minutes. I had used 486 modules and 57 I/O pins.

Day 17

Day 17 marked the beginning of simulation, or more accurately, the preparations. I was a complete novice at simulation, so I alternated between reading the Viewsim (Viewlogic's digital simulator) reference manual and setting up the files to provide stimulus for the circuit. I ran a few simulations to see if the file would do what I wanted.

I found learning to use the simulator similar to learning a relatively simple computer language. I used fewer than 20 of the roughly 60 commands available.

The Viewsim simulation environment (**Fig 1**) lets you work in three different modes. A text mode lets you input data and commands and output results. A graphical-waveform mode gives a logic-analyzer-type display showing the states of signals. You can bus signals together, such as data and address lines, so you can view many signals at once. You can also create stimulus in the graphical-waveform mode, although I didn't use that method on the project. The third method is to drop down into the schematic and see the actual data values on every node. You can even push down through the levels of hierarchy into macros so you can see what is happening inside a counter or the SAR macro (a soft macro I created).

I created my input commands and data in text files, viewed results with the waveform graphical display and occasionally dropped down into the schematic display to verify exactly where a problem identified in simulation was occurring. I found this method similar to debugging and testing real hardware.

I did not use any expect-data files to make automatic comparisons with simulation results. I visually examined the waveform graphical displays to get the most information. Although I reduced the number of signals displayed in the photographs to make them easier to read, in practice I crammed as many signals as I could onto the graphical waveform display for maximum information.

For those who haven't used digital simulation before, I can offer some comments on my experience. You have to set the initial conditions for all inputs (high, low, high-impedance, or unknown) for every input or I/O pin on the FPGA. Setting the inputs for control lines and changing them during simulation is straightforward. For bidirectional data lines such as the ones connecting to the RAM in this design, you need to drive them with

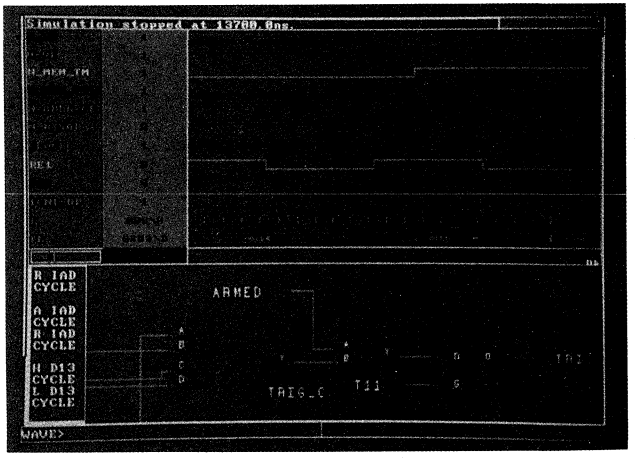


Fig 1—The FPGA software lets you observe simulation results in three different formats: text, graphical waveform, and on the schematic.

Illustration by Gregg Dinderman

EDN·SPECIAL PROJECT

the correct data during RAM reads, and release them during RAM writes. Even after you've set up the proper conditions on all external inputs and I/O lines, you're still not done. You may have to initialize internal conditions. Some of the initializing may be covered by reset logic you've designed into your circuit, and some of it may not.

For example, the address counter (**Fig 11** in Part 1, pg 98) is a free-running counter. When the simulation starts (and in the real hardware) the address is in an unknown state. After each count cycle, the hardware progresses from some initial state n to $n+1$, then $n+2$, and so on.

When you start simulation, the counter is in an unknown state and will stay there. To get useful simulation results, you have to force the counter into a known state, and then it will begin counting. I forced the counter's outputs to an initial state, then released the outputs, and the counter ran properly. You should label all counter output nets so that you can force them to known states during simulation.

Simulation is slow compared with real hardware, so you need to keep the clock-cycle count low if you want to make many simulation runs. Most of my simulation runs were a few hundred clock cycles or fewer. I don't believe a simulation ever took more than five minutes to run; a typical simulation run took about two minutes. For reference, the design is 1514 gates by standard gate-array counting measures. I ran the software on a 33-MHz 486 PC with 8 Mbytes of RAM.

To run the 15-bit counter through all 32,768 cycles would have taken nearly 400,000 clock cycles. Rather than having the simulator run for a day and generate reams of data, I forced counters to states near where some event should happen (or shouldn't happen, but might), released the counter, and let it count through the cycles I wanted to see. Then I forced the counter to the next event of interest (**Figs 2** and **3**).

My approach to simulating the design was twofold. First I tried to identify every part of the circuit where I had concerns, list them, and then make simulation test cases to verify that the function per-

Table 1—FPGA and other products used on this project

Manufacturer	Product	Price	Description
Actel	ALS-115	$2950	All Actel software used on the project plus Viewlogic's schematic capture software.
	ALS-017	$2950	Viewlogic's Viewsim for 3000 gates or fewer with Actel libraries.
	A1020A-PL84C	$36.25 (100)	FPGA.
Analog Devices	AD565AJD	$17.60 (100)	12-bit DAC.
	AD684JQ	$23.50 (100)	4-channel, 1-μsec S/H amplifier.
Hewlett-Packard Co	HRPG-A-SCA#16C	$15.75 (100)	Panel-mount rotary optical encoder, 120 counts per revolution.
Linear Technology Corp	LT1220CN8	$3.85 (100)	Very-high-speed op amp.
	LT119AH	$6.35 (100)	Dual comparator (commercial version LT319A is $1.95 (100)).

Editor's analysis

All things considered, my opinion based on this project is that designing with an FPGA is actually easier than designing with SSI and MSI logic. You can use the same schematic design approach you are familiar with. You don't have to use simulation to design with FPGAs, but I'd highly recommend it. My first attempt at simulation wasn't perfect, but it got me close. Finding the last few bugs in hardware and correcting them was not a problem. Correcting the bugs did not even necessitate pc-board changes.

I no longer wonder what applications FPGAs are useful for, but rather what applications still make sense for small- and medium-scale integration TTL and CMOS logic. High-current bus drivers are one; extremely simple logic circuits are another. Some high-speed logic will favor SSI and MSI devices, but a number of FPGAs are available with the capability of loading and operating 16-bit counters in the 50- to 100-MHz speed range.

Price is perhaps the biggest barrier to FPGAs' taking over the low- to medium-volume logic market. At $36.25 (100) for the device I used, it's competitive with SSI and MSI devices, especially if you factor in the cost of pc-board space and the flexibility to make design changes easily. As you move to higher-density and higher-speed FPGAs, you'll pay a premium over SSI and MSI parts.

formed as expected. These cases include counters, magnitude compares, optical-encoder signal decode, and others. Second, I simulated the FPGA as a whole, performing entire sequences of clearing, arming, recording, and playing back the data. Other simulation sequences tested the trigger-level and trigger-position-adjust operations.

My concerns on this design were mostly functional. Counters have to count, so I simulated all major transitions. In fact, it's a good idea to test all macros you create or alter separately. And because macros contain relatively few gates, they simulate quickly. Even though I tested the counter macros that I modified, I still tested the entire counter in the full schematic.

Because I had already run a place and route on the design, I could export the as-routed delays to the simulator and have a more accurate simulation. But I didn't want to use the as-routed delays at that time because I was still fixing a few bugs, and the unit-delay simulation provided a faster turnaround. I could change a schematic, recompile the simulation, and simulate in about five minutes. The turnaround time with as-routed delays requires a netlist transfer, design validation, place and route, and exporting the delays to simulation. The total time for doing all those things is about half an hour.

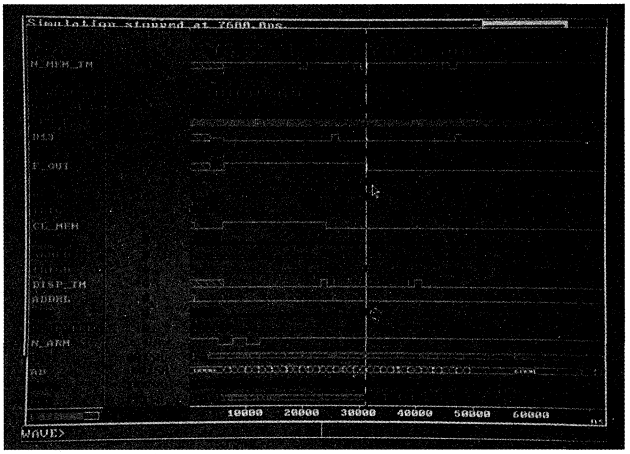


Fig 2—This sequence takes the design through reset, clearing memory, arming, triggering, completing data acquisition, playback with capture trigger marker, and end of data marker. To see more details, you need to zoom in as shown in Figs 3 and 4.

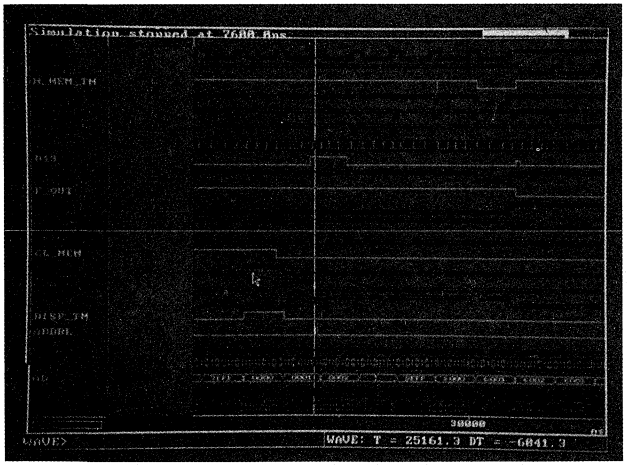


Fig 3—I've zoomed in to show a portion of the display in Fig 2 where the Memory Cleared (MEM_CL) signal indicates the memory has been cleared, and the FPGA changes its state to Armed. A short time later I force the trigger signal (TRIG) high, which causes the FPGA to change its state to Triggered (TRIGD).

Day 18

I spent lots of time on day 17 reading the simulation reference manual to understand the simulation commands and trying to figure out how to test my circuit with the available commands. By day 18 I was writing command files containing series of commands that initialize the FPGA and start taking it through its paces.

It seems incredible, but in two days I was able to learn enough to make simulation a useful tool. One of the attractive aspects of simulating an FPGA is that I didn't have to worry about simulation models. Anything I can design into the FPGA is covered by the simulation library. For this design, all the digi-

EDN·SPECIAL PROJECT

tal logic was in the FPGA, so I didn't see any need for board-level simulation.

Here are some of the bugs I found in one day (I corrected the bugs on the schematics in Part 1):

- D-14 needed to be gated so it only inhibited ADDR_CE (address count enable) on playback, not during record.
- &DISP_TM needed more delay so it wouldn't reset immediately when initiating CL_MEM. I had put in a delay, but simulation indicated it needed to be longer than I thought.
- The circuit was changing the compare multiplexer from compare 1 to compare 2 when the address changed during T2, instead of at the beginning of T1 (when it should happen).

I had been dreading simulation. I assumed I'd be bogged down in learning to use difficult software. Instead, I find myself a simulation convert—learning the software is reasonably easy. The bugs jump right out once you start exercising the circuit functions. It's just as much fun as debugging hardware, and you don't have to put probes on difficult-to-reach pins.

Day 19

I made a few changes to the successive-approximation logic and decided to create a soft macro (called SAR). I ran more simulation, and then I ran the place-and-route software and generated as-routed delay information for simulation. I spent a couple of hours making a final check of what remained to be simulated before I was ready to call simulation done.

The list of what was left to simulate seemed long, but because I was more familiar with the simulation commands and had simulation command files to perform most of the major functions on the FPGA, it went faster. I put simulations together quickly by calling initialization routines I'd already written, adding some commands, or modifying an existing command file.

I made some changes to the schematic and compressed the design onto fewer sheets. I then wanted to delete the excess pages, but couldn't find a utility for deleting schematic sheets. Instead, I deleted them from DOS.

A few hours later while simulating the design, things weren't quite right. I traced the problem to a 2-input multiplexer with the correct data going in, the correct data on the select pin, and unknown data on the output. I expected to find another output driving the net, but a double-check of the schematic indicated that that wasn't the problem.

This was my first, and only, serious problem with the simulation tools. It lasted for about two hours. Finally, I made the connection that perhaps the schematic sheets I deleted were not completely gone. It turns out that when you save a schematic sheet, the software creates a wirelist description file in the WIR directory. I deleted the files in the WIR directory for the schematic sheets I had deleted earlier, recompiled the simulation file, and was back on track. Of course, the simulation tools weren't really at fault, but I never did find anything in the documentation about how to delete schematic sheets properly.

I continued on to simulate the as-routed delays, exporting the delay information after a place and route into the simulator. The relatively short place-and-route time (approximately 15 minutes) is really useful when you make a design change and want to get back into a simulation with accurate timing.

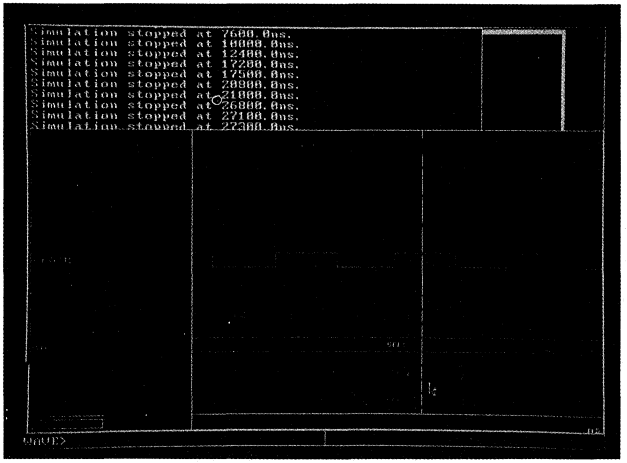


Fig 4—Zooming in still further on Fig 2, you can see the data written to RAM when the negative Write-Enable line (N_WE) goes high. Here I'm looking at the hold time on the Data bus.

Day 20

On day 20, my schedule called for having the design done and a functioning prototype board in one week, leaving me a week to tie up any loose ends before the article was due. However, I still hadn't got the design to the point where I wanted to freeze the FPGA pinouts. After that, I needed to lay out the prototype board, build the

board, program the FPGA, and get the circuit working.

Later that day, while running through my simulation test cases, I discovered a serious error: The FPGA would never write the memory-full signal to D14 because of a setup timing error. The problem was very easy to fix, but it made me wonder how many other errors remained.

I made a few minor changes and tried to run a unit-delay simulation, which didn't work. All the timing is in even clock cycles, but unit delay simulation should give each logic module a 1-nsec delay. I'm reasonably sure what the problem is, but don't know how to fix it.

Once you run the place and route software and export as-routed delays, the simulation software shifts from unit delay to zero delay per interconnect. It then looks up the real delay for each interconnect in a file. Because I've changed the schematic, the software apparently knows it can no longer use the as-routed delay file, but it isn't resetting the simulation to unit delays. I could have called Actel and found out how to fix the problem, but I elected just to run the place-and-route software, export the delays, and get on with simulation. I was at the point where I needed the timing accuracy anyway. (After I finished the project, I called Actel and got the answer to my problem. After you export the as-routed time delays, the software creates a file named *design*.VAR in your work-view directory. You need to delete that file and run *export wirelist* in the schematic window to return to the unit-delay simulation.)

My particular design had very few cases of critical timing because I was running at low clock rates. My requirement was 2 MHz, and 10 MHz was my goal for the digital. The analog part of the design could play back at more than 2 MHz, but the record mode probably couldn't go beyond 2 MHz and still settle properly during conversion. With this extremely loose timing, I didn't have to make any changes for speed in the design; I was more concerned about saving gates.

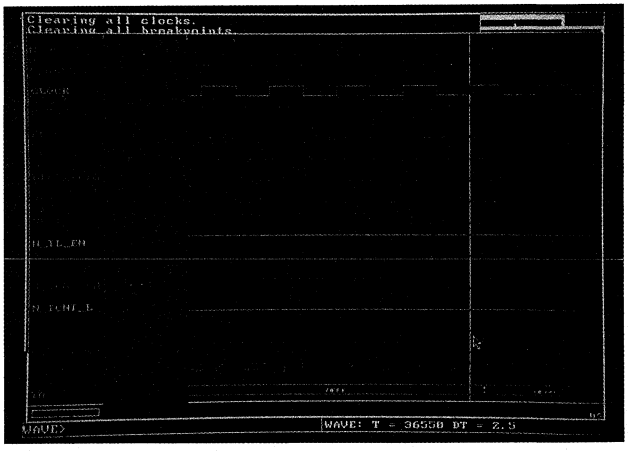
When designing faster circuits, you need to be sure critical networks don't end up with long interconnect delays. These long delays happen when two interconnecting modules are spaced far apart on the FPGA. The automatic place and route software attempts to place the design into the modules with a minimum of long interconnects. For this FPGA, long-vertical tracks are the worst, and long-horizontal tracks are the next worst. My design ended up having 16 long-vertical and 54 long-horizontal tracks.

The way you protect critical networks from long delays is with a network-criticality assignment. You can assign networks a criticality value of fast, medium, or uncritical. Fast or medium criticality keeps nets off long interconnects. You can designate as many as 5% of the nets fast and 15% medium. Assigning nets uncritical when they can tolerate long delays lets the routing software connect them with long tracks when necessary.

In my design, I designated fast criticality for the write-enable circuit because the tightest timing is at the end of a write cycle to the RAM. The RAM requires a zero hold time on the data when write-enable goes inactive. Initially, the design had a <10-nsec hold time in simulation, which should be okay. The fast criticality assignment widens the margin. Eventually, I added extra gating just to be sure my timing margin stayed on the proper side of zero.

Day 21

By day 21 I was still simulating. But since the circuit was in reasonably good shape, I started to push



Here I pushed the clock up to see where the circuit fails. At 20 MHz the signal enabling the upper 7-bit counter for the horizontal trigger position (N_TCNT_H) has only 2.5-nsec setup time before the rising edge of the clock, insufficient for the counter. You can see the Trigger Address bus (TA) does not make the transition from 0FFF to 1000 but goes to 0F00.

EDN-SPECIAL PROJECT

the speed. I had been working with a 2.5-MHz clock because that is all the speed I needed to have. I pushed the circuit up to 10 MHz, and then to 20 MHz.

At higher clock speeds, the simulator often puts out a warning that the circuit is not yet stable. What this means is that the results of the last data or clock transition are still propagating through the circuit.

An example of a relatively long path where I would get a circuit-not-yet-stable warning is in the trigger-level compare circuit. The 4-bit magnitude-compare soft macro was listed in the data book as having four module delays. The 2-bit compare had three module delays. The last bit to change in the compare will always be IDAC10, so this part of the circuit showed seven module delays through the comparator, plus three more modules to the TRIG signal for a total of ten module delays. Actually, the AND gate was combined with the latch when I compiled the design, so there were nine module delays.

Typical module delays, including interconnect, range from <6 nsec to approximately 11 nsec for a fan-out of eight. Long tracks can push the delay to approximately 35 nsec. Using 10 nsec as a round number, the delay from a magnitude-compare input change to the trigger-signal output was 90 nsec. Because I hadn't specified any of the nets in the trigger-level compare circuit as being critical, long tracks could show up anywhere, even inside the soft macros. Because I had two clock cycles of 500 nsec each before I needed valid data for the nominal design condition, I wasn't concerned. Even if every module had a long track connection, the delay would be about 315 nsec. Of course, the as-routed simulation or the static timer would show just how long it takes to get through a given path.

When I simulated the circuit at 20 MHz and stopped it to view the data one 50-nsec clock cycle after a data bit had changed going into the trigger-level compare circuit, I would get a circuit-not-yet-stable warning. The simulator was still giving me the correct results at that instant, but it was also warning me that even if all clocks and external inputs freeze in their present state, some outputs have yet to reach their final state.

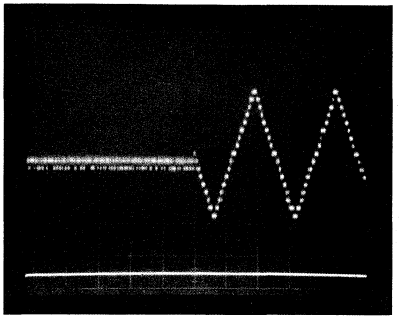


Fig 5—The waveform illustrates the record and playback circuit capturing a signal that changes from ground to a 1-kHz, 40-mV p-p triangle wave. The waveform was photographed during playback at twice the record speed using 0.2 msec/div and 10 mV/div. You can clearly see the 2.5-mV quantization levels. The pulse on the lower trace marks the capture-trigger location.

The simulation indicated that my FPGA would work at 12.5 MHz. If I needed the circuit to work at 20 MHz, I'd need to go back and start assigning fast and medium criticality to the appropriate nets or change the design.

With simulation complete, I spent the second half of the day laying out the circuit for the prototype board.

Days 22 to 24

Finally I got to the point where I was ready to freeze the pinouts. I had left them floating so that the place-and-route tools would have maximum flexibility to place and

The analog-circuit performance

The dc offset of the circuit from analog input to analog output is −5 to −7.5 mV. The dc gain error is within ±1 bit (2.5 mV) over the ±5V range, although the component specifications indicate you shouldn't expect better than ±2 bits. For better dc accuracy, you could trim the offset with an op amp on the input. Transition noise is approximately ±0.75 bit. I haven't been able to characterize the ac accuracy of the system to 12-bit accuracy.

The circuit is useful for examining signals to about 20 kHz with its 167-kHz sample rate. Filtering to avoid aliasing is a necessity if the circuit is used to examine signals with frequencies beyond 80 kHz. The 32k-word RAM provides 0.197 sec of storage with a 2-MHz clock speed. By slowing the clock speed to 12 kHz the circuit can sample at 1 kHz for more than 32 secs. During playback, you can increase the clock speed to 12 MHz for a flicker-free display on an analog oscilloscope.

EDN·SPECIAL PROJECT

route the design efficiently with a minimum of long tracks.

I had no more time for improvements. The only changes I could allow now were to fix bugs if I found them. As I transferred the FPGA pinout list to the full-circuit schematic, I discovered I hadn't brought out two signals—ARMED and TRIGD (triggered). I added the output pads, reran the place-and-route software, and got the signals. I used up 92% of the logic modules and 63 of the 69 I/O pins available.

On day 24, I assembled the prototype circuit. The prototype board was complete, except for an empty socket where the FPGA belongs. I created the fuse file for programming a chip, which took only a few minutes. Normally, you'd have the unit that you use to program the chip connected to your computer. When you're ready to program a device, you just put it in the programmer and run the software. I didn't have a device programmer, so I went to Actel to program my first chip. I didn't measure the time required to program a part, but I estimate it takes about 10 minutes.

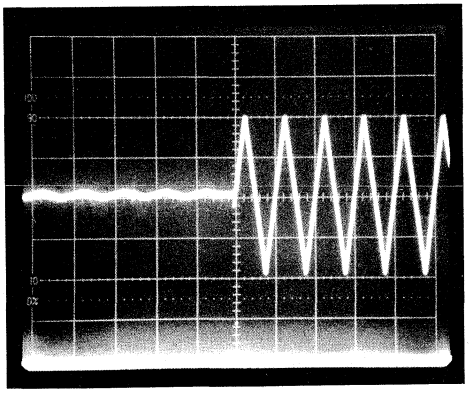
I plugged the part in the socket and powered up the prototype. I brought my power supplies and function generator with me to Actel and borrowed a scope. The circuit showed signs of life, but I couldn't get a good trigger from the display-trigger signal (DISP_TRIG). Gradually I came to the conclusion that the problem was the scope and not DSIP_TRIG. I asked for another scope and found that the circuit could perform all the basic operations. I don't know whether I was more surprised by my first FPGA design working, or that I handwired the prototype correctly.

I went back to my office for further testing, where I discovered a bug. As I turned the optical rotary encoder to adjust horizontal trigger position or trigger level, it occasionally jumped, rather than scrolling smoothly. The problem happened perhaps once in a hundred increments. The cause was a simple mistake: I had not synchronized the signal coming from the rotary encoder before using it in the circuit.

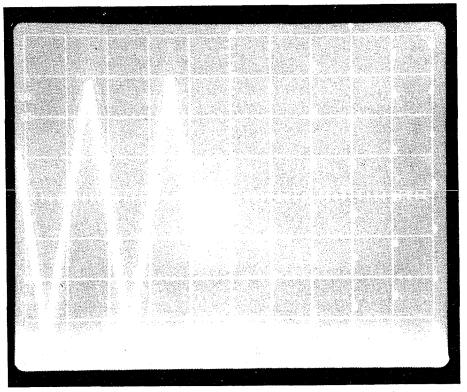
The rotary-encoder decode logic sets the count-up or -down signal correctly and then enables the counter for one clock cycle each time both rotary-encoder inputs are low. As the initial circuit was designed, the count-enable signal was set up for a random length of time before the clock. Most of the time the counter counted, sometimes it didn't, and other times it jumped because part of the count logic had sufficient setup time, and part of it didn't.

I ran the problem in simulation and it behaved just like the real thing. As I reduced the setup time below 5 nsec, jumps occurred on some count transitions. As setup time dropped below 2.5 nsec, the circuit just didn't count.

Adding a flip-flop to synchronize the signal solved the problem. If I were concerned about metastability, I would have added a second flip-flop. The effects of metastability in this application were not catastrophic and should be very infrequent with the long clock cycles.



(a)



(b)

Fig 6—The record-playback circuit provides 2.5-mV resolution on a ±5V signal. The two scope photos here show the voltage range and resolution. In (a) you can see the circuit has captured an 8V p-p, 500-Hz triangle waveform when I switched out an attenuator on the function generator. The waveform is being played back at twice the record speed (1 msec/div and 2V/div). The same waveform in (b) is being played back with the oscilloscope settings changed to 20 mV/div and 0.5 msec/div.

EDN·SPECIAL PROJECT

Incidentally, I did not incorporate any debounce circuits for the switches because I previously concluded they were not necessary for this design. When switching the clock frequency, I assumed a reset was necessary.

Days 25 to 28

I added the flip-flop required to synchronize the encoder inputs to the schematic, then placed and routed the design. On this place-and-route run, all I/O pins were fixed. I resimulated all FPGA functions with special attention to make sure the bug was fixed. I also verified all other functions to make sure the place-and-route changes did not adversely affect other functions. Simulation indicated the FPGA should fully function at 12.5 MHz.

I spent the remainder of the day working over the analog portion of the circuit to get the best performance I could.

I went back to Actel to program a new chip. The problem was solved; the circuit now appears to work properly.

I spent more time on the analog. Digital is either right or it's wrong—analog can always get better. By the end of the day I decided I had done all I could for the analog and decided to take the weekend off.

By Sunday afternoon I couldn't stay away, so I spent an hour using the circuit to capture signals. I found a bug. The circuit was supposed to capture signals with 25% pretrigger data and 75% post trigger data. About half the time it worked correctly, and the other half of the time it captured signals with 75% pretrigger data. I couldn't believe I didn't notice this problem earlier.

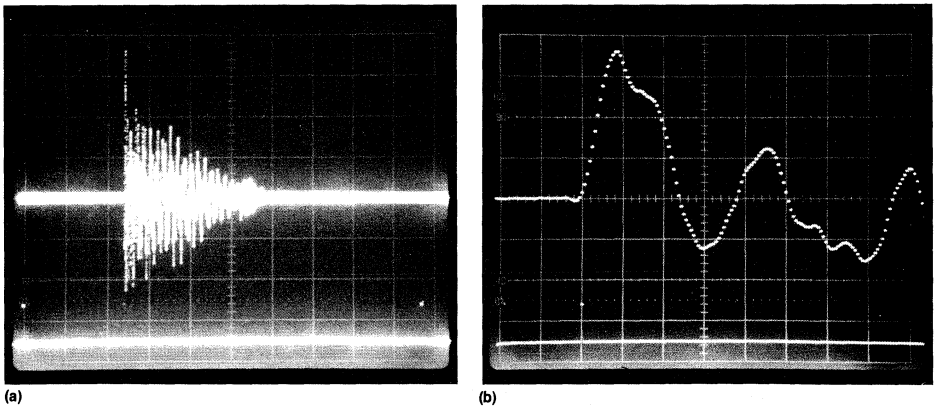
I was sure the error must be in how I computed the memory-trigger match signal (&MEM_TM), but it took me a while to see the problem. A simple logic error. The cases I tested earlier in simulation all worked properly. I added new test cases, and the bug showed up in simulation. I fixed the error by adding an XOR gate and reverifying the circuit in simulation.

I carefully tested the hardware one more time to make sure I couldn't find any more bugs. I went through the steps of placing, routing, and verifying that all simulations ran correctly. This time I sent my fuse files for programming the FPGA to Actel by modem. They programmed the part and mailed it back.

Day 29

The FPGA arrived. I plugged it in, and the circuit was fully operational. The project was finished. **Figs 5, 6,** and **7** show the circuit in operation.

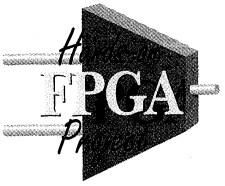
The circuit was designed for recording with a 2-MHz clock rate. The clock-speed limit was set by the analog circuitry performing conversion. During playback, the circuit could run much faster. At 16 MHz, all playback functions appeared to work properly. Simulation indicated that the circuit was operating on the ragged edge at 16 MHz. At 20 MHz, some of the counters were



(a) (b)

Fig 7—The 32k-word record length provides 197 msec of data at 6-µsec intervals. The signal in (a) is an acoustic noise amplified from a microphone. The upper trace is the full record length. The lower trace shows the data beginning and end markers. The circuit captures the signal with 25% pretrigger data. Photo (b), 20 msec/div, 0.5V/div, shows the same waveform played back at 100 µsec/div. The capture trigger location is visible on the lower trace.

EDN·SPECIAL PROJECT



occasionally causing errors because the ripple carry had insufficient setup time, just as simulation indicated.

Hindsight is 20/20

The first step in simulation, and perhaps the most difficult and important one, is to make a list of all the cases that require testing. Simulation can only find problems when you look for them. My faulty logic for determining when to stop the counter to acquire 25% pretrigger data was tested using two different start addresses. Both of them worked correctly. The counter was free running and could start on any of its 32,768 values, so I didn't think it practical to test them all. Had I given more thought to the problem, I might have tested a few more critical cases to verify the logic and find the problem in simulation.

My error in not synchronizing the optical encoder inputs was a careless design oversight. Although the problem can be found with simulation—I verified it with simulation after I found it—this type of problem could probably have been avoided by taking more care in the design process. Anytime you have asynchronous signals coming into a synchronous system, they demand plenty of careful consideration to make sure they won't have undesirable effects.

I did not make the same kind of careful schematic check I normally do before having a circuit built. I thought simulation would provide a more thorough job finding errors than my going over the schematic a few more times. I also felt the time pressure to finish the project.

I think simulation did help me make a more thorough check of the logic than I could have done without it. Simulation however, should not be a substitute for carefully checking your schematic to identify potential sources of trouble. Once you've identified potential problem areas, simulation can help you test them.

Although I had hoped to be able to report a fully functioning circuit on my first silicon, reality turned out different. In retrospect, my experience on the project probably points out the strong points of FPGAs. I don't know how many days or weeks I would have had to spend on simulation to find the two bugs that slipped through. The problems were easy to find in real silicon and didn't take much longer to fix than when you find them in simulation.

I wouldn't want to push the approach of finding your mistakes in silicon too far. Simulation provides a better way to test a design over the full operating temperature and voltage ranges plus manufacturing process variations. I think of finding mistakes in silicon as a fall-back position after you've done the best you can in simulation.

The realities of schedules that don't allow weeks to simulate a design as completely as you'd like may force you into a corner if it is vital that first silicon be final silicon. I used as much time as I had for simulation—about four and a half days, which included learning to use the software, and then went on to try the real device. It would not have been worth another week of simulation to find the two problems I found in silicon. I'd have a different perspective if I'd designed a mistake into a mask-programmed gate array and spent $10,000 and lost a few weeks before I found my mistake.

Had I made a design mistake that left the FPGA with serious functional problems, I could have used the diagnostic probe capability on the FPGA. The diagnostic probes let you look at any two nodes in the FPGA with an oscilloscope or logic analyzer.

I started this project with no experience designing FPGAs and none in digital simulation. I wanted to see if designing a circuit using an FPGA was really simple enough for a designer familiar with 7400-series TTL design to jump into expecting to produce the first cir-

The analog-circuit performance

For more information on the FPGA and design tools used on the project, circle the appropriate numbers on the Information Retrieval Service card or use EDN's Express Request service. When you contact any of the following manufacturers directly, please let them know you read about their products in EDN.

Actel Corp
955 E Arques Ave
Sunnyvale, CA 94086
(408) 739-1010
FAX (408) 739-1540

Analog Devices
Box 9106
Norwood, MA 02062
(617) 329-4700
FAX (617) 326-8703

Hewlett-Packard Co
19310 Pruneridge Ave
Cupertino, CA 95014
(800) 752-0900

Linear Technology Corp
1630 McCarthy Blvd
Milpitas, CA 95035
(800) 637-5545
(408) 434-0507

NEC Electronics Inc
Box 7241
Mountain View, CA 94039
(800) 632-3531
TLX 3715792

Viewlogic Systems
293 Boston Post Rd W
Marlboro, MA 01752
(508) 480-0881
FAX (508) 480-0882

EDN-SPECIAL PROJECT

HANDS-ON FPGA PROJECT

cuit in a reasonable amount of time.

My conclusion based on the products I used on this project is that migrating to FPGAs is a step any design engineer should be able to make. You always have to stretch yourself when learning to use new tools, but the jump shouldn't consume great quantities of time while you come up to speed. In the course of this project, I've covered all the problems I had that were worth mentioning. There weren't many. In the end, my biggest problems were the normal system design issues of deciding what the circuit should do. Once I knew what I wanted to do, designing the circuit was relatively easy.

My biggest surprise was how well the software worked. I had a few problems, but frankly I expected more. My dread of simulation turned out to be unfounded. I actually look forward to using simulation on my next project. It's more enjoyable to find mistakes in simulation than in hardware.

For this project I chose a circuit that would not require high clock rates. As a result, I didn't spend any time refining the design to make it run faster. Had I needed the circuit to run faster, I'd have needed more time to refine the schematic and criticality file, and I'd perform more simulation runs.

EDN

Acknowledgment

I'd like to thank the following companies for providing products for this project: Actel, Analog Devices, Hewlett-Packard, Linear Technology Corp, and Viewlogic.

Technical Editor Doug Conner is based in California. You can reach him at (805) 461-9669.

Technical Support Services

Section 12: Technical Support Services

Technical Support Services

Actel Technical Support

Actel's application engineers have developed many ways to meet your needs. Our services provide technical assistance to all of our customers worldwide. Customers can obtain technical assistance by means of the Technical Support Hotline, the Bulletin Board Service (BBS), Customer Training, International email, and our recently improved fax-back system, Action Facts and Fax Broadcast.

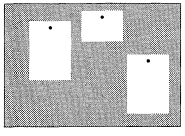
Technical Support Hotline

The Technical Support Hotline provides technical information on Actel hardware and software products. Questions regarding software authorization, availability, and pricing are handled through Actel customer service. All hotline calls are answered

by our Technical Message Center. The center retrieves information such as the caller's name, the company name, the phone number, and the caller's question, and then issues a case number. The center then fowards the information to a queue where the application engineers receive the data and return the customer call. Our goal is to return all customer inquiries within one hour. The hotline's phone hours are from 7:00 a.m. to 5:30 p.m. PST. In addition to answering the Technical Support Hotline calls, our applications engineers develop other ways of assisting our customers, such as writing application notes, and user guides, creating design examples, and evaluating software.

The Technical Support Hotline number is 800-262-1060.

Electronic Bulletin Board Service (BBS)



Actel currently offers information access and transfer via a 24-hour worldwide bulletin board. Customers can download information such as new soft macros and software bug fixes. In addition, our applications engineers use

the BBS to help solve design problems. Sometimes a customer's issue can not be solved by the technical hotline. In these cases, our applications engineers may request the customer to upload their design to the BBS. All files uploaded to the Actel BBS are automatically stored in a private directory. This way the applications engineers can examine the software first hand and make the necessary adjustments.

The telephone numbers for the BBS is 408-739-6397 and 408-738-5717 with a baud rate of 9600 and 14400 respectively. To connect to the BBS by modem, the following equipment and configuration are required:

- Baud rate of 9600 or 14400
- Data format: 8 data bits, 1 stop bit, no parity

The following file transfer protocols are supported:

- Xmodem
- Ymodem
- Zmodem
- ASCII

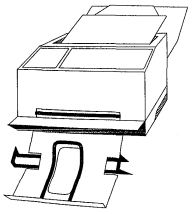
First-time callers need to establish an account. Once connected to the BBS, the caller is then prompted for his/her name, company, phone number and so on. After the account is established, the customer calls the Technical Support Hotline 800-262-1060 and requests file transfer class. Our Technical Message Center verifies the name and company and then upgrades the account's security level.

Customer Training

Actel offers an introductory two-day course covering all aspects of designing an Actel FPGA. The class covers a discussion of the architecture of ACT™ 1, ACT 2 and ACT 3 families design methodologies, a brief look at the Viewlogic® schematic capture and simulation tools, and a thorough examination of the Actel FPGA design

software. The course is a mixture of lectures and lab exercises. Classes are offered to a minimum of three persons at a time. Customers can register for our training course by calling the Technical Support Hotline and asking about training class registration.

Action Facts



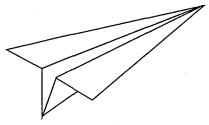
The Action Facts system is a 24-hour fax-back service. Customers can obtain a list of current software bugs and workarounds, application notes, design hints, package pinouts, and much more information. Simply call the toll free number and request that a catalogue be faxed to your office. Review the document and call the number again to request up to five documents per call. The Action Facts Catalogue is frequently updated.

The Action Facts phone number is 800-262-1062.

Fax Broadcast

Actel has a new support mechanism that allows our customer to receive automatic information regarding latest software bugs and workarounds, new application notes, new products and much more. This information, if you choose to participate, will be automatically faxed to your fax machine when it comes available. When you call the Technical Support Hotline be sure to tell them you want to be a member of Fax Broadcast.

International E-mail Address



Actel has recently introduced a new method for customers to communicate with our applications engineers. Customers can e-mail their technical questions to our e-mail address and receive answers back either by e-mail, fax, or phone call. In addition, customers can quickly e-mail their design files to receive assistance. The e-mail account is monitored several times throughout the day, and our goal for responding is two working days after receiving the message.

Technical support's e-mail address is

tech@actel.com

Internet World Wide Web

Actel is currently establishing a Web site. This should be operational by April 2, 1995. The Web address will be:

http://www.actel.com

Action Facts Catalog

Actel Fax On Demand - Index of Available Documents

Doc #	Description	Pages	Date
0000 CATALOG			
0000	Action Facts Catalog	5	06-06-95
0001	New Documents Index	1	06-06-95
1000 SALES & MARKETING			
1001	Actel Sales Offices	3	03-22-95
1002	Eastern Western Area Sales Reps	10	03-22-95
1003	Domestic Distributors	18	03-22-95
1004	International Representatives	11	03-22-95
1007	The Hidden Cost of Reprogrammability	2	07-18-94
1008	TI - Silicon Products Q&A	15	05-25-95
1100 RELEASE NOTES			
1101	ALS 2.3.1	14	03-08-95
1102	Cadence/Concept Rapidsim -Sun	8	07-20-94
1103	Synopsys/ALS 2.2 Release Notes	9	07-21-94
1104	ALS 2.3	9	09-14-94
1105	ProSeries Version 5.1	8	09-14-94
1106	Cadence/Concept Rapidsim Patch	5	03-06-95
1107	ALS 2.3.2	14	03-08-95
1200 ERRATA			
1201	ALS 2.2S1 Update - PC	12	07-18-94
1202	BC Eratta for Designer Advantage	1	07-18-94
1203	Converting Designs with ALS and Designer	1	04-14-95
1204	Actel/Cadence Installation Upate	1	05-25-95
1300 LATEST ISSUES			
1303	ALS 2.3 Installation Notes -PC	3	09-22-94
1304	Designer's Digest - Dec. 15, 1994	3	12-14-94
1305	Designer's Digest - Jan. 15, 1995	2	01-17-95
1306	Designer's Digest - Feb. 17, 1995	2	02-17-95
1307	Designer's Digest - March 17, 1995	2	03-17-95
1308	Designer Digest April 15, 1995	2	04-14-95
1309	Designer's Digest, May 15, 1995	2	05-15-95
2000 HARDWARE DOCUMENTS			
2001	A1010/A1020 Binning Circuit Description	1	07-09-94
2002	Act 2 Typical and Worst Case VOH & VOL	4	07-09-94
2003	Activator 2 Test Procedure	5	07-09-94
2004	External Control of Probe Pins	7	02-16-95

Actel Fax On Demand - Index of Available Documents

Doc #	Description	Pages	Date
2005	Lead Form & Trim Recommendation	3	07-09-94
2006	A1280 Capacitive Loading vs Slew	1	07-09-94
2007	Programming Failure Guidelines	4	07-09-94
2008	Programming ACTEL Die	6	12-15-94
2009	Single Event Upset & Latchup Susceptibilities of A1280	8	07-09-94
2010	Activator Cables	3	07-09-94
2012	Using Act1 Security Fuses	1	07-18-94
2013	Troubleshooting APS2	3	07-18-94
2014	Weight of Actel Packages	1	07-18-94
2015	Answer to commonly asked Device questions	6	02-28-95
2016	APS2 Installation Guide	12	09-27-94
2017	Actel Calibration	1	10-27-94
2018	Actel Security Fuse Configuration	3	11-14-94
2019	Security Fuse Verification	1	03-08-95
2020	Reliablity Report	16	05-25-95
2100	RADIATION TEST DOCUMENTS		
2101	Rad Testing-Single Event Upset A1020	10	07-09-94
2102	Rad Testing-Total Dose A1020	5	07-09-94
2103	Rad Testing Total Dose A1020a	5	07-09-94
2104	Rad Testing-Single Event Upset A1280	8	07-09-94
2105	Rad Testing - Total Dose A1020B	13	12-30-94
2106	Rad Testing - Total Dose A1280A	13	01-23-95
2200	SOCKET INFORMATION		
2201	Socket Recommendation for Actel Packages	2	11-15-94
2202	Yamaichi VQ80 socket	1	11-08-94
2203	Yamaichi PQ100 socket	1	11-09-94
2204	Yamaichi PQ144 socket	1	12-15-94
2205	Yamaichi PQ160 socket	1	11-09-94
2206	Yamaichi PQ208 socket	2	11-09-94
2207	Yamaichi CQ132/172/196 socket	2	02-21-95
3100	ACT1 DIAGRAMS		
3101	Act1 44, 68, 84 Pin PLCC Mechanical Description	1	07-21-94
3102	84-Pin CQFP Mechanical Description	4	05-25-95
3103	80-Pin VQFP Mechanical Description	1	01-02-95
3104	100-Pin PQFP Mechanical Description	1	07-21-94
3105	84-Pin CPGA Mechanical Description	1	07-21-94
3106	A1020B 80-Pin VQFP Pin-outs	1	07-18-94
3107	A1010B/A1020B 100-Pin PQFP Pin-outs	1	07-18-94
3108	Act1 44,68,84-Pin PLCC Pin-outs	1	07-18-94
3109	A1010B 80-Pin VQFP Pin-Outs	1	07-18-94
3200	ACT2 DIAGRAMS		

Actel Fax On Demand - Index of Available Documents

Doc #	Description	Pages	Date
3201	84-PLCC Mechanical Description	1	07-21-94
3202	100-Pin CPGA Mechanical Description	1	07-21-94
3203	100-Pin PQFP Mechanical Description	1	07-21-94
3204	132-Pin CPGA Mechanical Description	1	07-21-94
3205	144-Pin PQFP Mechanical Description	1	07-21-94
3206	160-Pin PQFP Mechanical Description	1	07-21-94
3207	172-Pin CQFP Mechanical Description	1	07-21-94
3208	176-Pin CPGA Mechanical Description	1	07-21-94
3209	1225, 1240 84-PLCC Pin-Outs	1	07-18-94
3210	1280 84-PLCC Pin-outs	1	07-18-94
3211	Act2 100-Pin CPGA Pin-outs	1	07-18-94
3213	Act2 100-Pin PQFP Pin-outs	1	07-18-94
3215	Act2 132-Pin CPGA Pin-outs	1	07-18-94
3217	Act2 144-Pin PQFP Pin-outs	1	07-18-94
3219	Act2 160-Pin PQFP Pin-outs	1	07-18-94
3221	Act2 172-Pin CQFP Pin-outs	1	07-18-94
3223	Act2 176-Pin CPGA Pin-outs	1	07-18-94
3225	Act2 General Floorplan	1	07-18-94
3226	A1280 Logic Module Floorplan	1	07-09-94
3227	A1240 Logic Module Floorplan	1	07-09-94
3228	A1225 Logic Module Floorplan	1	07-09-94
3229	A1225A VQ100L Pin-outs	1	11-17-94
3230	A1280A TQ176 Pin-outs	1	11-17-94
3231	A1240A TQ176 Pin-outs	1	11-17-94
3232	176 TQFP Mechical Description	1	11-17-94
3233	100L VQFP Mechanical Description	1	11-17-94
3300 ACT3 DIAGRAMS			
3301	133-Pin CPGA Mechanical Description	1	07-21-94
3302	160-Pin PQFP Mechanical Description	2	05-25-95
3303	207-Pin CPGA Mechanical Description	1	07-21-94
3304	208-Pin PQFP Mechanical Description	1	07-21-94
3305	175-Pin CPGA Mechanical Description	1	07-21-94
3306	257-Pin CPGA Mechanical Description	1	07-21-94
3307	84-PLCC Mechanical Description	1	07-21-94
3308	100-Pin PQFP Mechanical Description	1	07-21-94
3309	100-Pin CPGA Mechanical Description	1	07-21-94
3310	Act3 100-Pin CPGA Pin-outs	1	07-18-94
3311	Act3 100-Pin PQFP Pin-outs	1	07-18-94
3312	Act3 133-Pin CPGA Pin-outs	1	07-18-94
3315	A1425 160-Pin PQFP Pin-outs	1	07-18-94
3317	Act3 175-Pin CPGA Pin-outs	1	07-18-94
3318	Act3 207-Pin CPGA Pin-outs	1	07-18-94
3320	1460A 208-Pin PQFP Pin-outs	1	07-18-94
3326	Act3 84-Pin PLCC Pin-outs	1	07-18-94
3327	A1415A VQ100L Pin-outs	1	11-17-94
3328	A1425A VQ100L Pin-outs	1	11-17-94
3329	A1440A VQ100L Pin-outs	1	11-17-94
3330	A1440A TQ176L Pin-outs	1	11-17-94

Actel Fax On Demand - Index of Available Documents

Doc #	Description	Pages	Date
3331	A1460A TQ176 Pin-outs	1	11-17-94
3332	A1460A BGA225 Pin-outs	1	11-17-94
3333	A14100A BGA313 Pin-outs	2	01-02-95
3334	A14100A RQ208 Pin-outs	1	11-17-94
3335	BGA225 Mechanical Description	1	11-17-94
3336	BGA313 Mechanical Description	1	11-17-94
3337	176TQFP Mechanical Description	1	11-17-94
3338	100L VQFP Mechanical Description	1	11-17-94
3339	208 RQFP Mechanical Description	1	03-07-95
3340	A14100 PG257 Pin-outs	2	03-07-95
3341	1425A CQ125 Die & Pkg Connector	4	05-25-95
3342	196LD CQFP Outline Drawing	4	05-25-95
4000 ALS SOFTWARE INFO			
4001	Checksum & Silicon Signature Analysis	2	07-18-94
4002	ALS EDIF Reader & Writer	7	11-09-94
4003	Invoking APS2 with different I/O address in DFW	2	07-21-94
4004	Network License Installation - Sun	10	09-26-94
4005	Network License Installation - HP	12	09-26-94
,0 VIEWLOGIC DOCUMENTS			
5101	Changing Paper Tray on Printer - Procedure	1	07-09-94
5102	Viewlogic Plotting Projects	3	07-18-94
5103	Creating Custom Fonts for Schematic Capture & Plottin	4	07-09-94
5104	Powerview 5.1 Library List	5	07-09-94
5105	VHDL Designer 2.2.1 Release Notes	2	07-09-94
5106	Creating/Changing Paper Size and Margins for Plotting	2	07-18-94
5107	Modifying the Pen Plotter Color Mappings	1	07-18-94
5108	Creating Custom Title Sheets	1	07-18-94
5109	Using Bus Structures	8	07-29-94
5110	Updating from Workview V.4.0 to V.4.1	6	03-08-95
5111	Using ProLib to set your Actel Libraries in ProSeries 6.0	6	03-22-95
5112	Using "Set Actel Libs" in ProSeries 6.4	3	03-22-95
5113	Creating Designs w/Procapture in ProSeries 6.0	2	03-22-95
5200 ORCAD DOCUMENTS			
5201	How to create a user-defined soft macro	4	07-21-94
5203	Creating an ACT 1 Template Design	2	03-17-95
5204	Creating an ACT 2 Template Design	2	03-17-95
5205	Creating an ACT 3 Template Design	1	03-17-95
5300 MENTOR GRAPHICS DOCS			
5301	Mentor V7 to V8 Design Conversion	2	07-18-94
5302	Actel Properties in Mentor Graphics	2	02-28-95

Actel Fax On Demand - Index of Available Documents

Doc #	Description	Pages	Date
5303	Board Level Post-Layout Sim. using ALS & Quicksim II	2	07-18-94
5305	Establishing symbolic link for HP700	1	07-22-94
5306	Integrating ACTgen Blocks w/ Mentor	11	09-26-94
5307	Integrating ACTmap Blocks w/ Mentor	8	09-26-94
5400 CADENCE DOCUMENTS			
5401	Using postactmap_chip in ACTMAP/Concept/RapidSIM	4	09-15-94
5402	Verilog Backannotation Simulation using SDF Annotatic	5	09-15-94
5403	Cadence/Concept Rapidsim Patch	5	02-28-95
5600 SYNOPSYS DOCUMENTS			
5601	Answers to most frequently asked questions	11	11-09-94
5602	Actgen/Synopsys Design Flow	6	07-18-94
5603	Integrating Synopsys Designs with Mentor Graphics	6	11-11-94
5604	Regenerating Actel DesignWare Lib. - Synopsys 3.2	1	11-16-94
5700 DATA I/O DOCUMENTS			
5701	Programming Actel FPGAs Using Unisite or the 3900 P	5	01-02-95
5702	Actel Devices/Data I/O Programmer Cross-Reference Li	2	12-30-94
6000 DESIGN INFORMATION			
6001	Using ACT3 Family I/O Macros	17	07-09-94
6002	Simultaneously Switching Output Limits for Actel FPGA	2	07-18-94
6003	Power-on Reset Circuit	2	03-08-95
6004	Implementing Three-State & Bidirectional Buses w/ mult	5	07-28-94
6005	Oscillators for Actel FPGAs	2	07-28-94
6006	Hints and Tips for Better Actel Designs I	1	07-28-94
6009	Introduction to FPGA System Design	4	07-28-94
6010	Estimating Actel FPGA Device Capacity	2	07-28-94
6011	Three-Stating Act Device I/O Pins for Board-level Testin	2	07-28-94
6012	List of Legal Macros for HCLKBUF in ALS2.3	1	09-26-94
6013	ALS Timer Tutorial	12	03-08-95
6014	Setup & Hold Time Analysis using the Actel Timer	4	03-22-95
7000 TRAINING CLASS INFO			
7001	Training Class Agenda	1	07-26-94
7002	Training Class Schedule	1	02-28-95
7003	Training Class Registration	1	07-26-94
7004	Map to Actel and Hotel from SJ Airport	1	07-18-94
7005	Training Class Form	1	07-26-94

Notes:

Actel Corporation
955 East Arques Avenue
Sunnyvale, CA 94086
Tel: (408) 739-1010

Actel Europe Ltd.
Intec 2, Unit 22 Wade Road
Basingstoke Hants RG24 8NE
England
Tel: (256) 29.209

5172035-0

Actel