Texas Instruments

Field Programmable Gate Array

European Applications Seminar

1994

ASIC Products

FPGA APPLICATIONS SEMINAR

Easy Logic Integration

Texas Instruments

Agenda

- ***PLD/FPGA Architecture Analysis***
- ***How To Replace TTL With FPGA***
- ***How To Replace PLD With FPGA***
- ***Applications Benchmarks***
- ***FPGA Design With VHDL***
- ***FPGA Products And Development Systems***

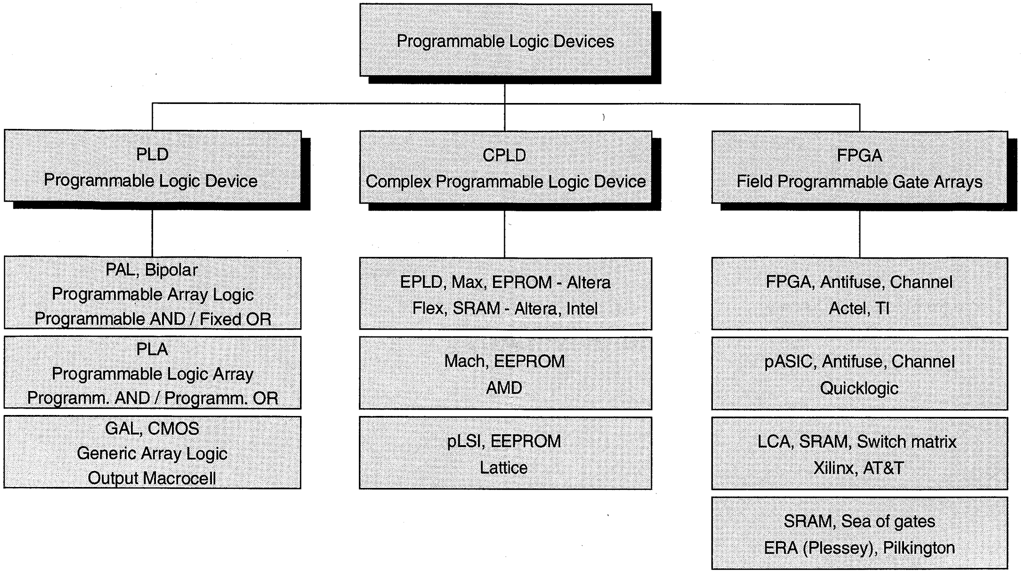
Agenda

- ☞ ■ *PLD/FPGA Architecture Analysis*
- ■ *How To Replace TTL With FPGA*
- ■ *How To Replace PLD With FPGA*
- ■ *Applications Benchmarks*
- ■ *FPGA Design With VHDL*
- ■ *FPGA Products And Development Systems*

FPGA Application Examples

COMPUTER:	PC ADD-ON BOARD FOR NETWORKING
	WORKSTATION PERIPHERALS (DMA, DRAM CONTROLLER)
INDUSTRIAL:	CONTROLLER FOR MACHINE TOOLS, CRANES, SCALES, PARKING TIMER, LIFTS
	IMAGE PROCESSING (SCANNER, ROBOTS, ...)
TELECOM:	PABX, CENTRAL UNITS
	LINE CARDS
INSTRUMENTATION:	MEDICAL EQUIPMENTS
	SOPHISTICATED METERS
AUTOMOTIVE:	PROTOTYPING TWO WIRES BUS SYSTEM FOR CARS/TRUCKS
	CONTROLLERS FOR WIPERS, LIGHTS
CONSUMER:	PROFESSIONAL VCR
SPACE:	AIRCRAFT CONTROLLER

PLD / FPGA Architectures


Programmable Logic Devices

PLD
Programmable Logic Device

CPLD
Complex Programmable Logic Device

FPGA
Field Programmable Gate Arrays

PAL, Bipolar
Programmable Array Logic
Programmable AND / Fixed OR

PLA
Programmable Logic Array
Programm. AND / Programm. OR

GAL, CMOS
Generic Array Logic
Output Macrocell

EPLD, Max, EPROM - Altera
Flex, SRAM - Altera, Intel

Mach, EEPROM
AMD

pLSI, EEPROM
Lattice

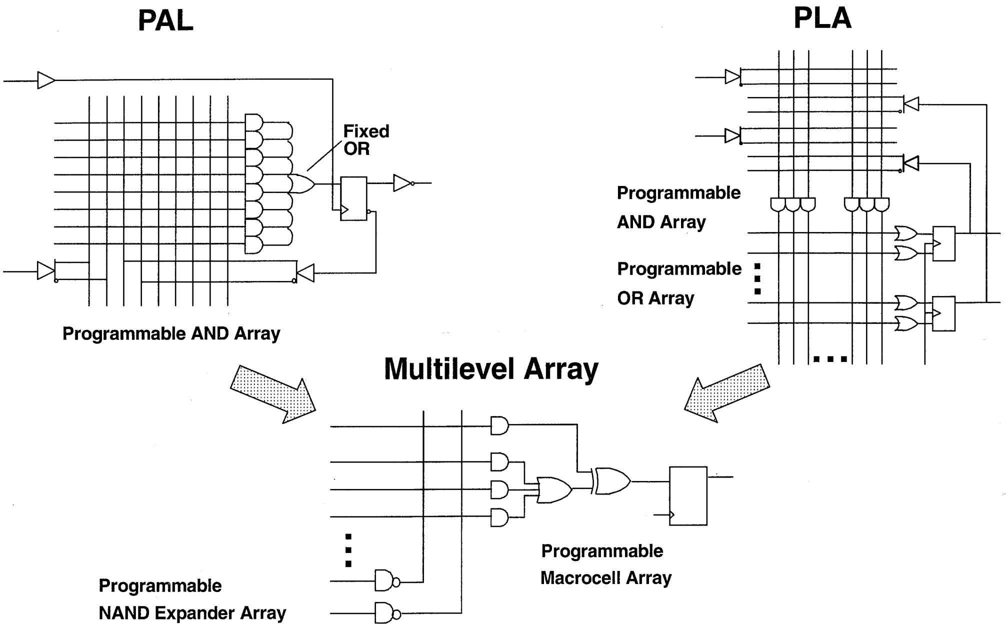
FPGA, Antifuse, Channel
Actel, TI

pASIC, Antifuse, Channel
Quicklogic

LCA, SRAM, Switch matrix
Xilinx, AT&T

SRAM, Sea of gates
ERA (Plessey), Pilkington

PAL/PLA Evolution



PAL

Fixed OR

Programmable AND Array

PLA

Programmable AND Array

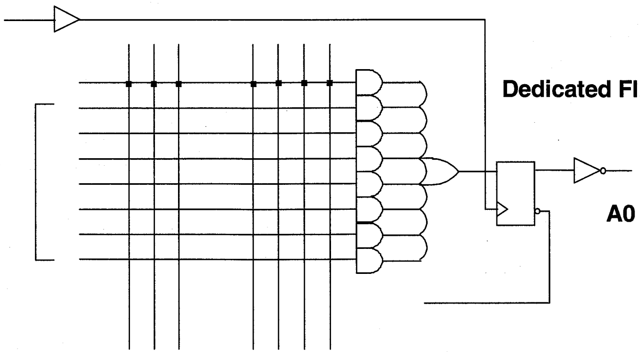
Programmable OR Array

Multilevel Array

Programmable Macrocell Array

Programmable NAND Expander Array

Macrocell Utilisation



1 product term is used

7 product terms waisted

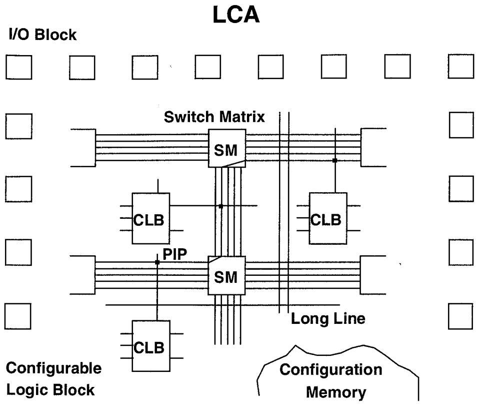
Dedicated Flipflop waisted

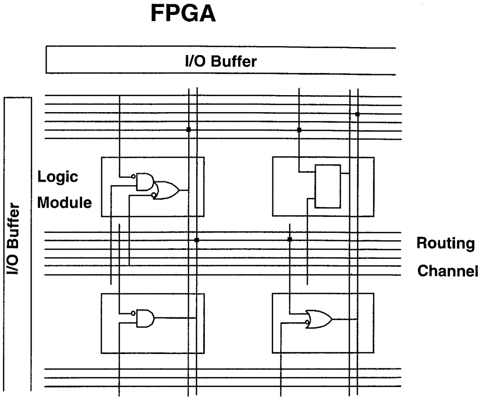
A0 & A1 & ...& An

A0 ▪ ▪ ▪ An

A typical address decoder

LCA vs FPGA



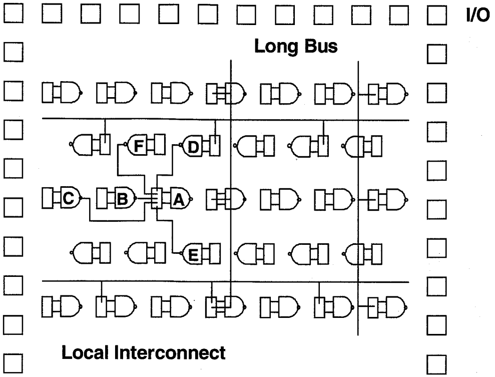


- Volatile SRAM cells, 2-chip solution
- XC3090: 9000 nominal gates in 320 Configurable Logic Blocks, big building block
- Complex to use switch matrix
- Routing time in hours / days

- Non-volatile antifuses, 1-chip solution
- TPC1280: 8000 gates in 1232 logic modules, fine granularity
- Simple to use routing channels
- Routing time in minutes / one hour

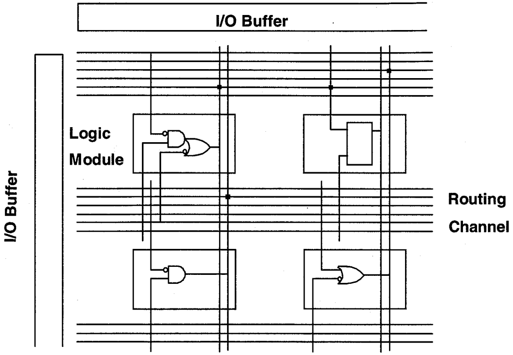
Sea of Gates vs Channeled Architecture

Sea of Gates



- High nominal but low usable gate count
- Limited and complex routing
- Low gate utilisation
- Difficult place & route

FPGA Channeled Gate Array

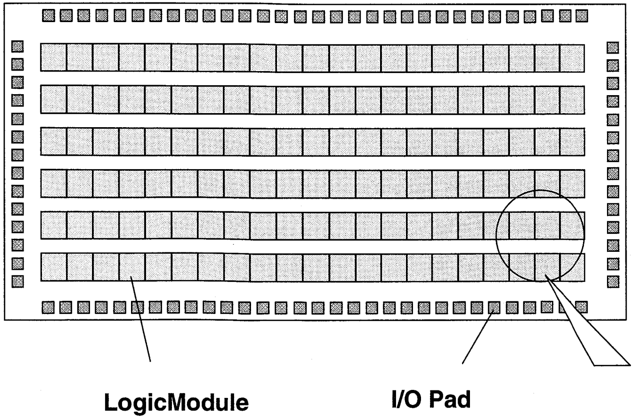


- Medium nominal but high usable gate count
- Efficient routing channels
- 85% - 95% gate utilisation
- Fast, automatic place & route

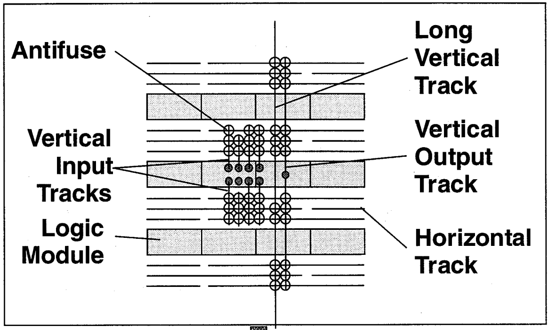
What is FPGA?

- A family of Field Programmable Gate Arrays which combines the complexity and flexibility of real gate arrays with the user programmability of PALs.
- A one chip solution using "Antifuses" as high-density, non-volatile and reliable programming elements.
- A channeled gate array architecture with efficient logic modules, simple and easy to use horizontal and vertical routing tracks.
- Allow 85-95% gate utilization, fast and 100% automatic Place & Route.
- FPGA "Actionprobe" allows in-circuit debug and 100% observability of all internal nodes.

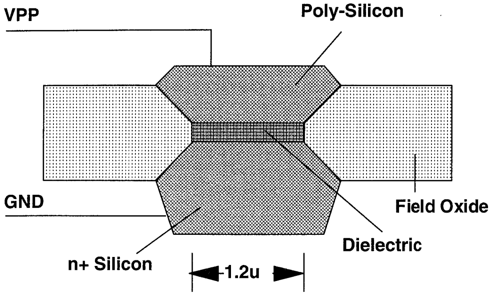
FPGA Architecture



- Array architecture similar to gate array
- Rows of logic modules interconnected by horizontal and vertical routing tracks
- I/O pads on periphery
- Programmable antifuse elements used to connect routing tracks



Antifuse

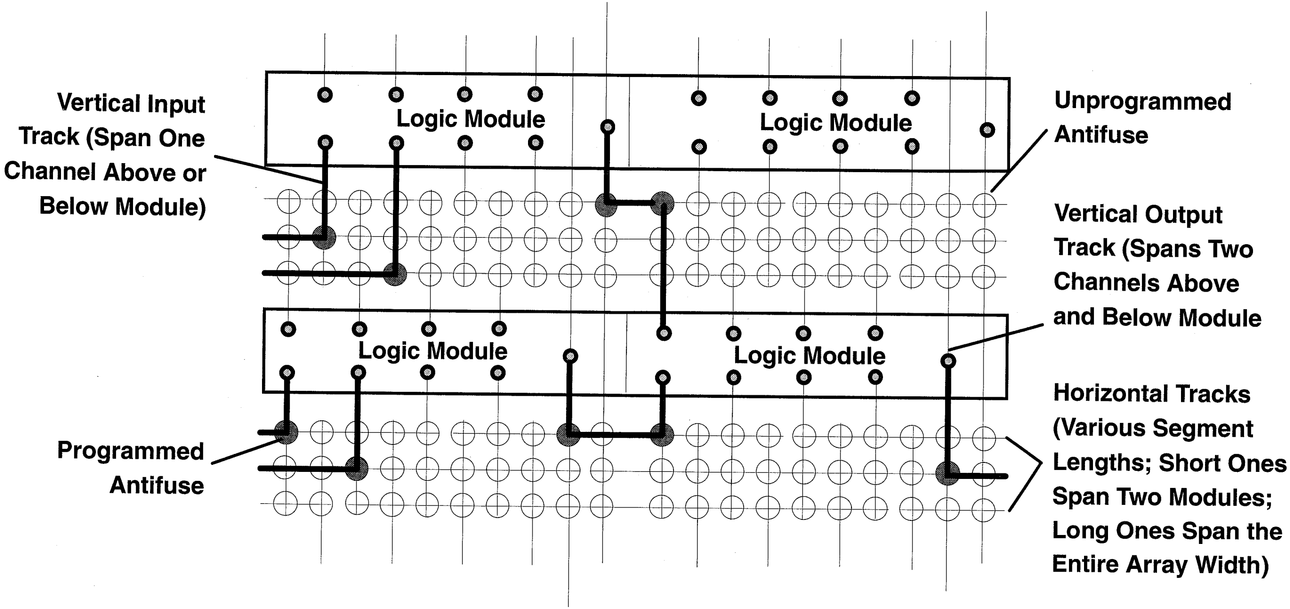


Antifuse

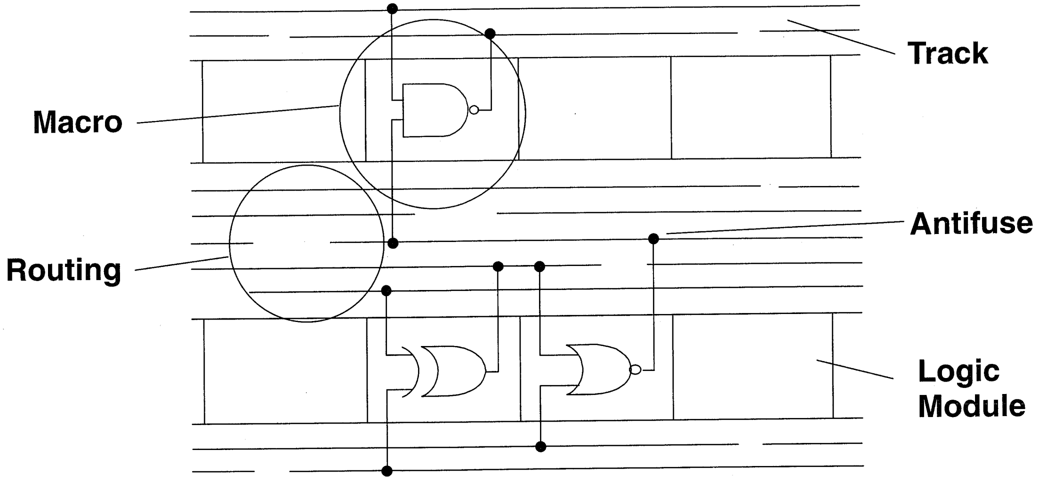
Parameter	Value
Programming Voltage VPP	21V
Programming Time	< 5ms
Programming Current	< 10mA
On Resistance	~ 500 Ohms
Off Resistance	> 100 MOhms
Geometrie	1.2u x 1.6u

- PAL fuses are normally connecting. FPGA Antifuses are non-connecting
- Antifuses consist of two conducting layers, separated from each other by a thin dielectric
- During programming, electrical pulses are applied across the antifuse that breaks down the dielectric and creates a conducting path
- The antifuse size is 1/20th the size of an SRAM size and 1/10 of an EPROM cell

Routing Interconnects


Vertical Input Track (Span One Channel Above or Below Module)
Logic Module
Logic Module
Logic Module
Logic Module
Unprogrammed Antifuse
Vertical Output Track (Spans Two Channels Above and Below Module
Horizontal Tracks (Various Segment Lengths; Short Ones Span Two Modules; Long Ones Span the Entire Array Width)
Programmed Antifuse

Architecture Summary

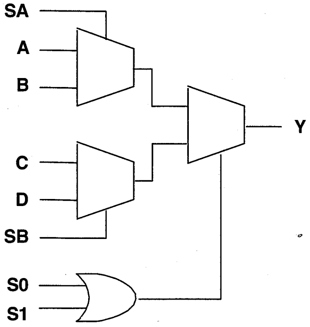


Easy Logic Integration

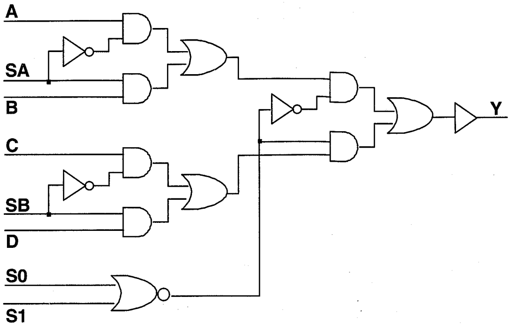
FPGA

TPC10 Logic Module

Block Diagram



Logic Diagram

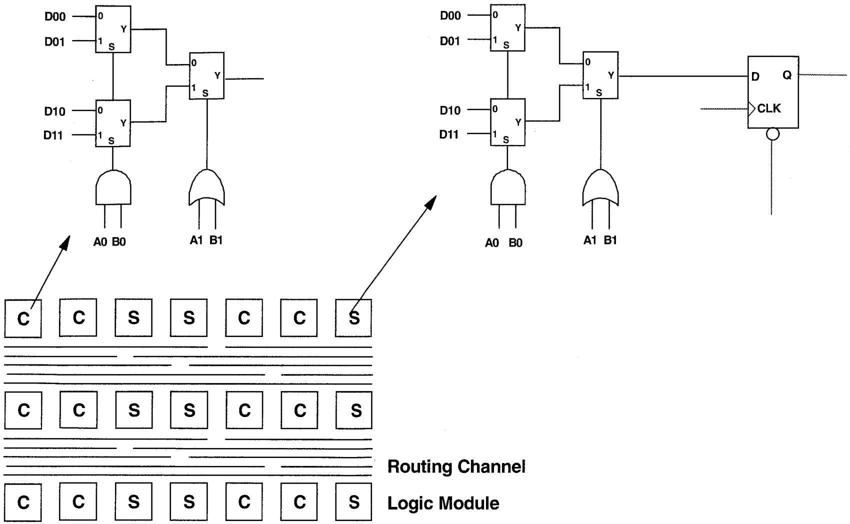


- TPC10 series has only combinatorial logic modules
- Every logic module consists of 8 inputs and one output
- Emulates the function of three 2-input multiplexors and one OR gate

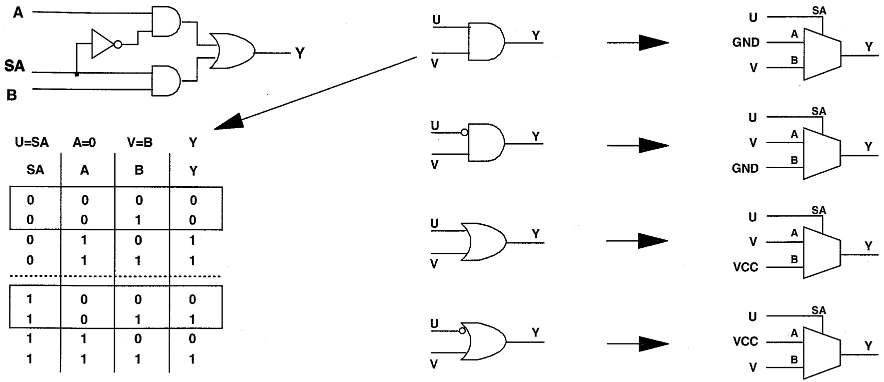
TPC12 Logic Modules

Combinatorial Module

Sequential Module



FPGA Hard Macros



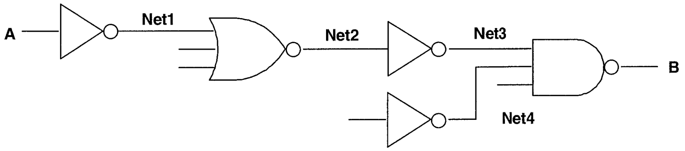
U=SA	A=0	V=B	Y
SA	A	B	Y
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

- FPGA hard macros are built from logic modules by programming the appropriate antifuses to connect the module inputs to GND or VCC
- The logic module architecture is capable of building complex gates or flip-flops with a minimum of resources

FPGA Simplifies Logic

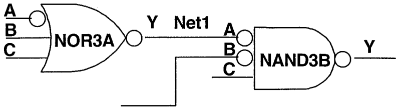
Standard Logic

Example1: 5 Gates, 4 Nets

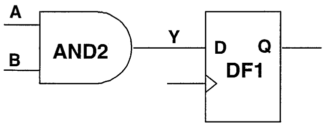


Equivalent FPGA Logic

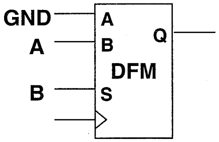
2 Gates, 1 Net



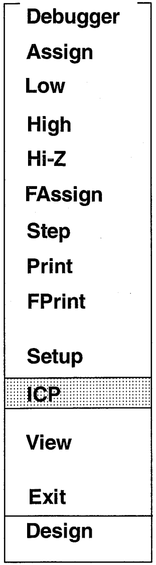
Example 2: 1 Gate + 1 Flipflop

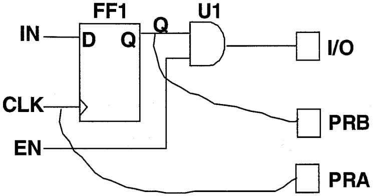


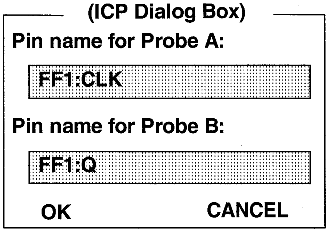
1 Multiplexed D-Flipflop



In-Circuit Probe





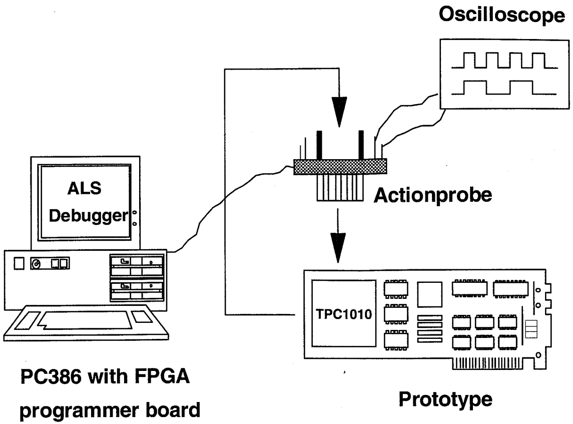

(ICP Dialog Box)

Pin name for Probe A:

FF1:CLK

Pin name for Probe B:

FF1:Q

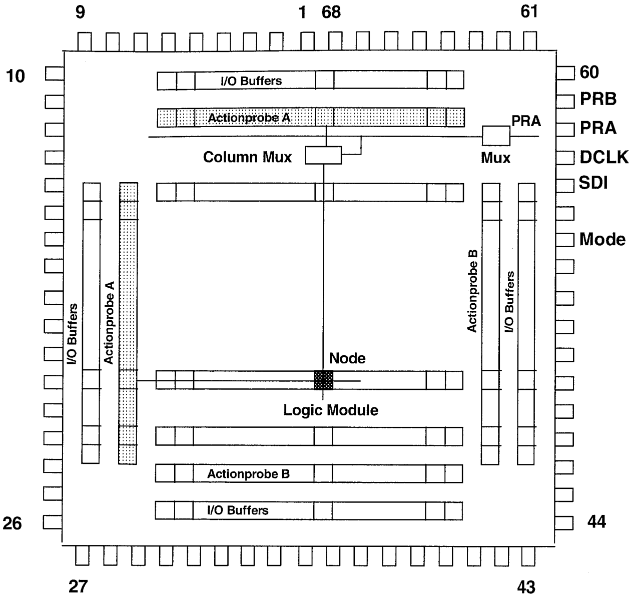


1) Remove FPGA from prototype

2) Plug FPGA in Actionprobe socket

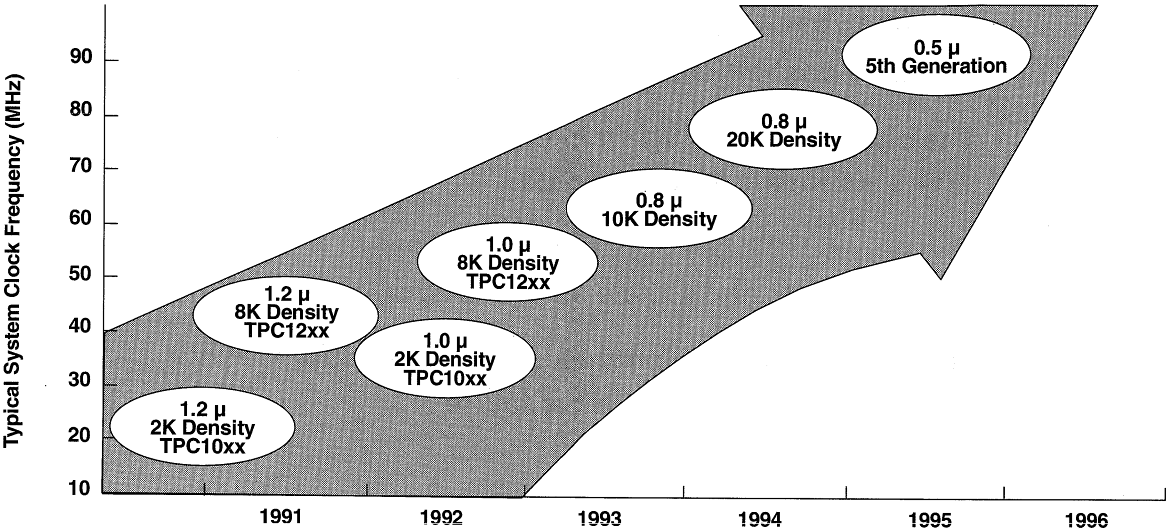
3) ~~Plug Actionprobe in FPGA socket on prototype~~

Actionprobe



- Actionprobe allows 100% observability of any internal node in the array
- The device is switched into the Probe Mode (Mode)
- The node address is shifted to the device through the Serial Data Input (SDI) and loaded into the row and column registers
- The node is then connected to the pin PRA or PRB. DCLK is the Data Clock for SDI

Product Roadmap



Easy Logic Integration FPGA

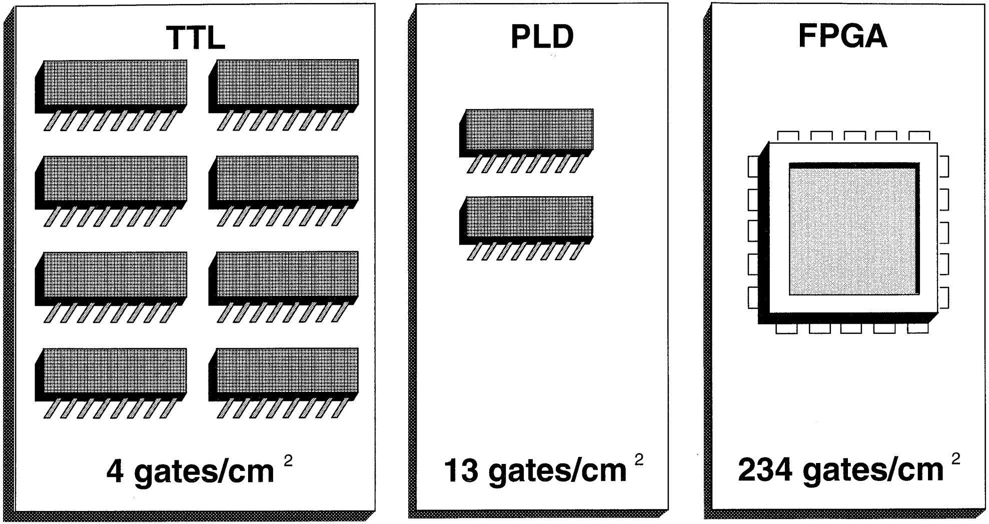
Summary

- TI FPGAs have fine grained logic modules which facilitate cost-efficient utilization of silicon resources.
- The channeled routing architecture allows fast and 100% automatic Place & Route.
- The non-volatile antifuses provide reliable interconnects and enable an one chip solution.
- The small size of the antifuses allows development of highest density FPGAs in the future .

Agenda

- *PLD/FPGA Architecture Analysis*
- ☞ *How To Replace TTL With FPGA*
- *How To Replace PLD With FPGA*
- *Applications Benchmarks*
- *FPGA Design With VHDL*
- *FPGA Products And Development Systems*

Why Use FPGA?



TTL

4 gates/cm ^{2}

PLD

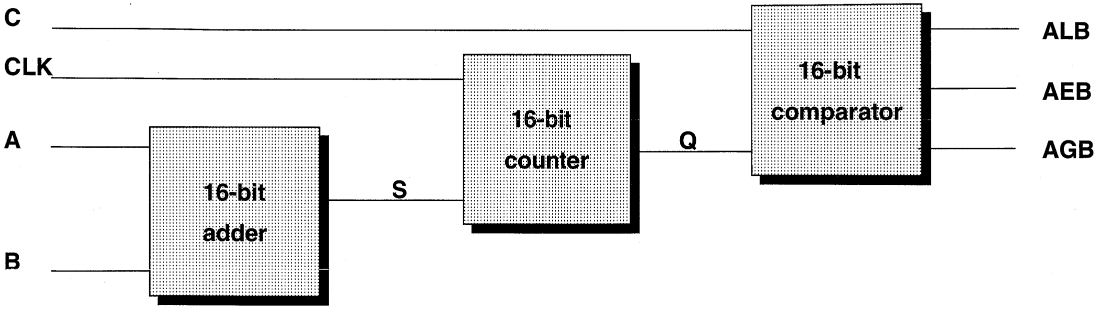
13 gates/cm ^{2}

FPGA

234 gates/cm ^{2}

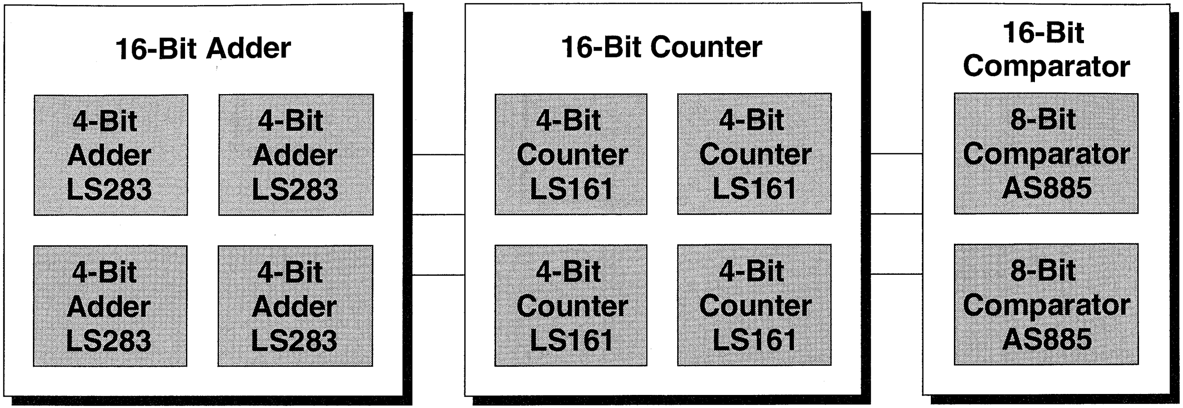
Source: In-Stat Forum, May 1990

A Case Study



- Two 16-bit numbers A and B are added.
- The sum S is loaded to a preloadable 16-bit counter.
- A 16-bit comparator compares the count Q with a 16-bit number C.
- The outputs are ALB (A<B), AEB (A=B) and AGB (A>B).

A TTL Solution


16-Bit Adder

4-Bit Adder LS283

4-Bit Adder LS283

4-Bit Adder LS283

4-Bit Adder LS283

16-Bit Counter

4-Bit Counter LS161

4-Bit Counter LS161

4-Bit Counter LS161

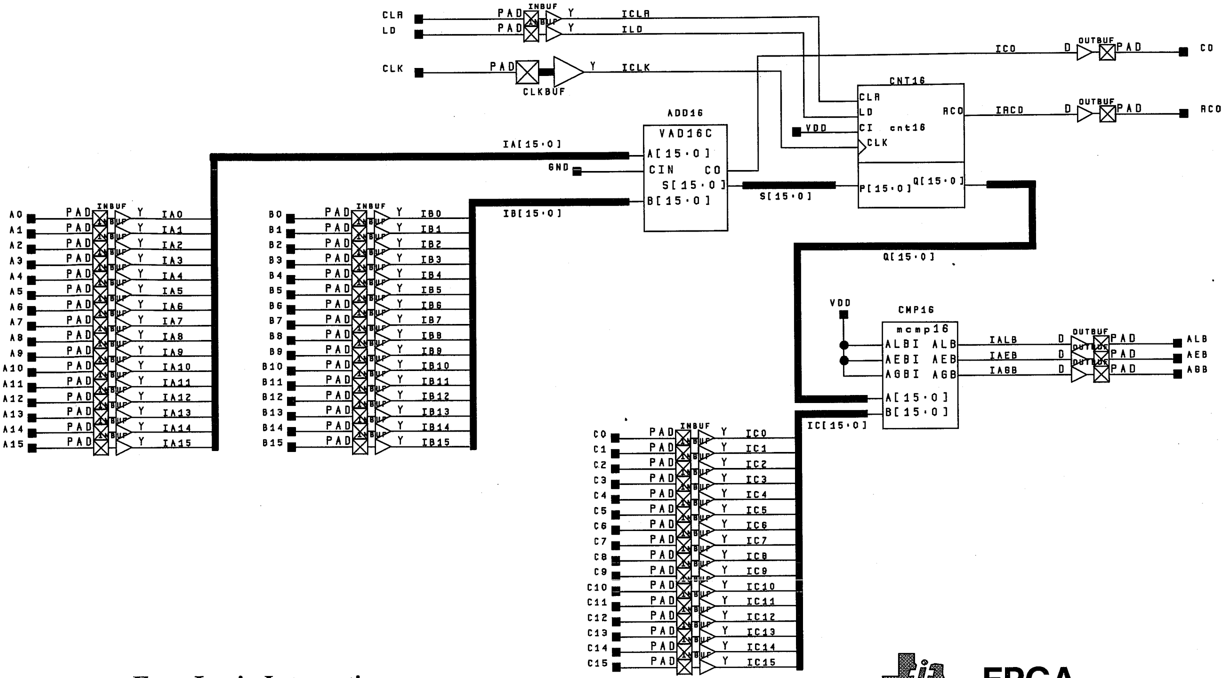
4-Bit Counter LS161

16-Bit Comparator

8-Bit Comparator AS885

8-Bit Comparator AS885

FPGA Solution



Easy Logic Integration

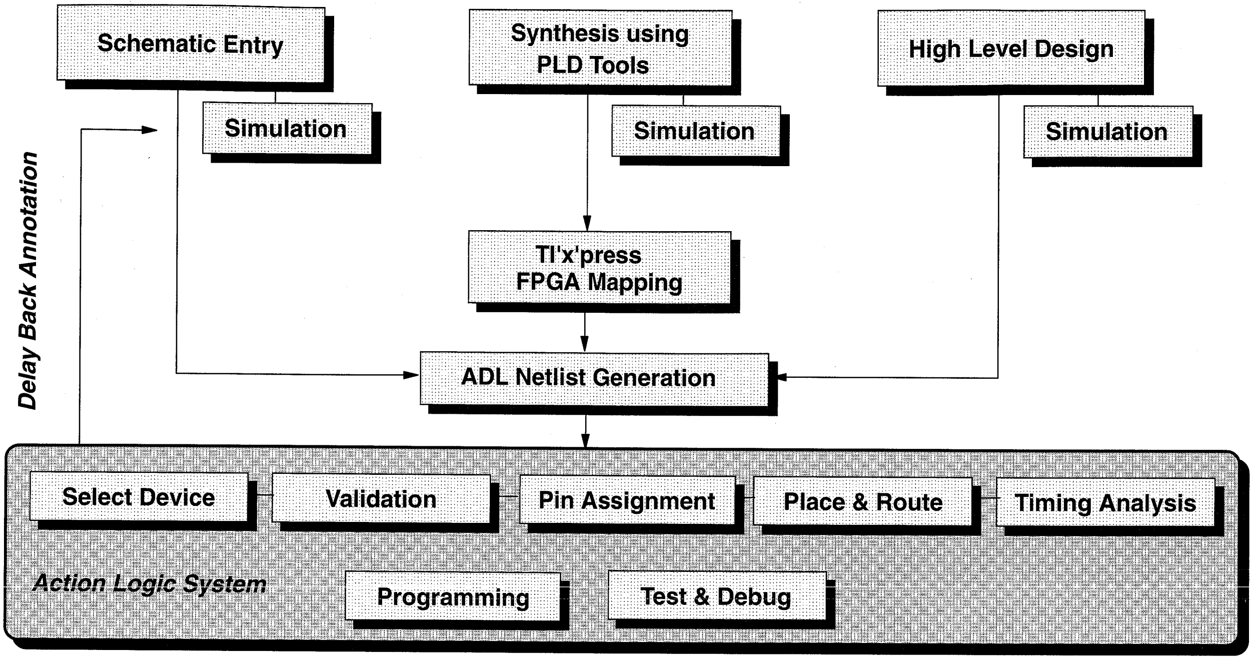
FPGA

TTL versus FPGA Analysis

	TTL			TPC1225A		
	TTL Devices	tpd	Icc	% Device used	tpd	Icc
16-Bit Adder	4xLS283	95 ns	130 mA	21%	64 ns	80 mA
16-Bit Counter	4xLS161	130 ns	128 mA	16%	45 ns	
16-Bit Comparator	2xAS855	35 ns	420 mA	16%	114 ns	
Soldered Pins	176			85 (56 Used)		
Icc Total	687 mA			80 mA		
Device Cost	$11.00 (10 Packages)			$40.00 (1 Package) 53% used		
PCB Cost	$20.00			$5.00		
Total Cost	$31.00			$45.00		

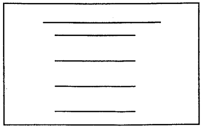
- *Reduce board space*
- *Reduce power consumption*
- *Reduce test effort*
- *Improve reliability*
- *Reduce cost*

FPGA Design Flows



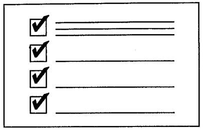
Action Logic System

Select Device



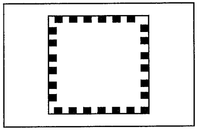
- ■ Select device
- ■ Select package

Validation



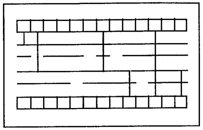
- ■ Design rule check
- ■ Utilisation statistics
- ■ Fanout warning

Pin Assignment



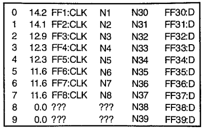
- ■ Automatic
- ■ Manual (optional)

Place & Route



- ■ 100% Automatic
- ■ Manual placement (optional)

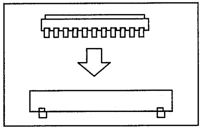
Static Timing Analyser



```
0   14.2 FF1:CLK   N1    N30   FF30:D
1   14.1 FF2:CLK   N2    N31   FF31:D
2   12.9 FF3:CLK   N3    N32   FF32:D
3   12.3 FF4:CLK   N4    N33   FF33:D
4   12.3 FF5:CLK   N5    N34   FF34:D
5   11.6 FF6:CLK   N6    N35   FF35:D
6   11.6 FF7:CLK   N7    N36   FF36:D
7   11.6 FF8:CLK   N8    N37   FF37:D
8    0.0 ???       ???   N38   FF38:D
9    0.0 ???       ???   N39   FF39:D
```

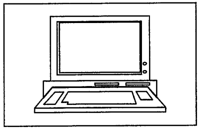
- ■ Static analysis of path delays
- ■ Voltage/temperature derating capability

Programming



- ■ Activator 2 and 2S
- ■ DataIO for TPC10

Test & Debug



- ■ In the programmer
- ■ In-circuit (Actionprobe)

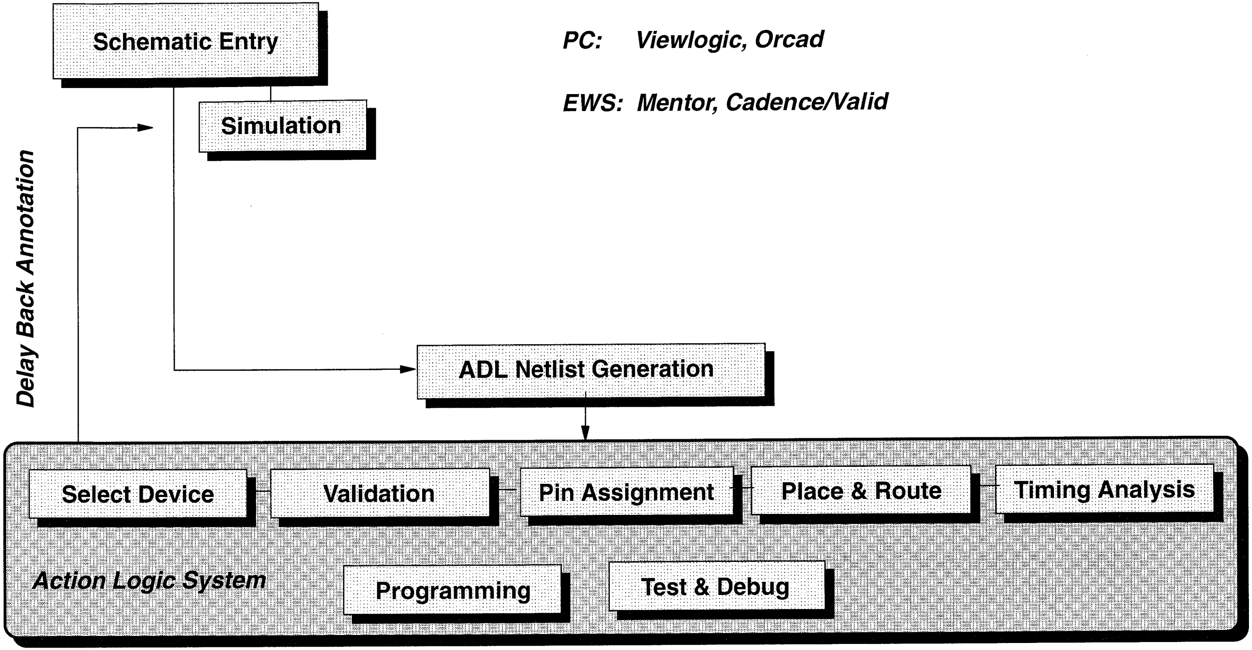
How To Replace TTL With FPGA

- Many TTL functions can be replaced directly by an equivalent soft macro from the library. For example, the TA161 is an equivalent soft macro for the TTL 4-bit counter 161.

- If there is no direct replacement, another soft macro may be used to realize the function:
  For example, the soft macros FADD16 (16-bit adder) or VADD16C (very fast 16-bit adder) can be used to replace four TTL 4-bit adders HC283.

- In other cases, users can create user macros using soft macros or hard macros from the library.
  For example, a 16-bit comparator can be built by cascading two 8-bit comparator soft macros MCMPC8.

TTL To FPGA Design Flow



PC: Viewlogic, Orcad

EWS: Mentor, Cadence/Valid

TTL Soft Macros 1

Soft Macro	Function	Logic Levels	S-Module	C-Module
TA00	2-input NAND	1	0	1
TA02	2-input NOR	1	0	1
TA04	Inverter	1	0	1
TA07	Buffer	1	0	1
TA08	2-input AND	1	0	1
TA10	3-input NAND	1	0	1
TA11	3-input AND	1	0	1
TA20	4-input NAND	1	0	2
TA21	4-input AND	1	0	1
TA27	3-input NOR	1	0	1
TA32	2-input OR	1	0	1
TA40	4-input NAND	1	0	2
TA42	4 to 10 decoder	1	0	10
TA51	AND-OR-Invert	1	0	2
TA54	4-wide AND-OR-Invert	2	0	5
TA55	2-wide AND-OR-Invert	2	0	3
TA86	Quad 2-input XOR	1	0	1
TA138	3-to-8 decoder	2	0	12
TA139	Dual 2-to-4 decoder	1	0	4

TTL Soft Macros 2

Soft Macro	Function	Logic Levels	S-Module	C-Module
TA150	16-to-1 multiplexor	3	0	6
TA151	8-to-1 multiplexor	3	0	5
TA153	4-to-1 multiplexor	2	0	2
TA154	4-to-16 decoder	2	0	22
TA157	2-to-1 multiplexor	1	0	1
TA160	4-bit decade counter	4	4	12
TA161	4-bit binary counter	3	4	10
TA164	8-bit shift register	1	8	0
TA169	4-bit updown counter	6	4	14
TA174	Hex D-flipflops	1	6	0
TA175	Quad D-flipflops	1	4	0
TA190	BCD updown counter	7	4	31
TA191	4-bit updown counter	7	4	30
TA194	4-bit shift register	1	4	4
TA195	4-bit shift register	1	4	1
TA269	8-bit updown counter	8	8	28
TA273	Octal register	1	8	0
TA280	Parity generator	4	0	9
TA377	Octal register	1	8	0
TA688	8-bit comparator	3	0	9

Complex Soft Macros 1

Function	Description	Name	Logic Levels	S-Modules	C-Modules
Counters	4-bit counter	CNT4A	4	4	8
	4-bit counter (CI, CO inverted)	CNT4B	4	4	7
	4-bit updown counter	UDCNT4A	5	4	13
	very fast 16-bit counter	VCNT16C	1	34	31
	2-bit down counter, prescaler	VCNT2CP	1	5	2
	2-bit down counter, MSB	VCNT2CU	1	2	3
	4-bit down counter, middle bits	VCNT4C	1	4	8
	4-bit down counter, LSB	VCNT4CL	1	4	7
Decoders	2-to-4 decoder	DEC2X4	1		4
	2-to-4 decoder, active low outputs	DEC2X4A	1		4
	3-to-8 decoder	DEC3X8	1		8
	3-to-8 decoder, active low outputs	DEC3X8A	1		8
	4-to-16 decoder	DEC4X16A	2		20
	2-to-4 decoder with enable	DECE2X4	1		4
	2-to-4 decoder, active low outputs	DECE2X4A	1		4
	3-to-8 decoder with enable	DECE3X8	2		11
	3-to-8 decoder, active low outputs	DECE3X8A	2		11

Complex Soft Macros 2

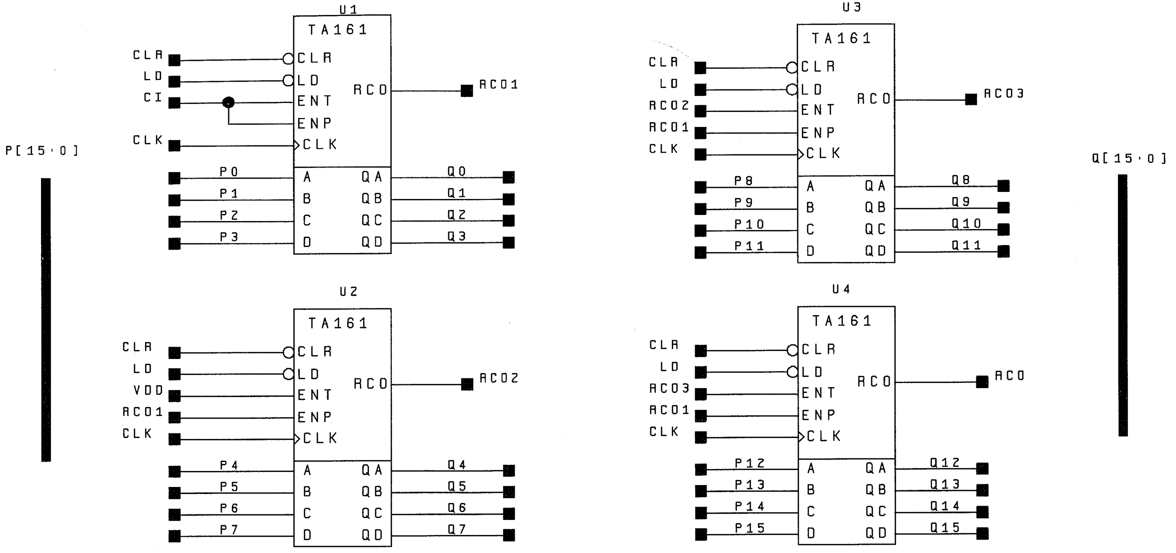
Function	Description	Macro Name	Logic Levels	S-Modules	C-Modules
Registers	octal latch	DLC8A	1	8	
	octal latch with enable	DLE8	1	8	
	octal latch with mux	DLM8	1	8	
	4-bit shift register	SREG4A	1	4	
	8-bit shift register	SREG8A	1	8	
Adders	8-bit adder	FADD8	3		44
	9-bit adder	FADD9	3		49
	10-bit adder	FADD10	3		56
	12-bit adder	FADD12	4		69
	16-bit adder	FADD16	5		97
	2-bit sum generator	SUMX1A	2		5
	very fast 16-bit adder	VADD16C	3		97
Comparators	4-bit identity comparator	ICMP4	2		5
	8-bit identity comparator	ICMP8	3		5
	2-bit magnitude comparator with enable	MCMPC2	3		9
	4-bit magnitude comparator with enable	MCMPC4	4		18
	8-bit magnitude comparator with enable	MCMPC8	6		36
Multiplexors	8-to-1 multiplexor	MX8	2		3
	8-to-1 multiplexor with active low outputs	MX8A	2		3
	16-to-1 multiplexor	MX16	2		5

How To Create A User Soft Macro

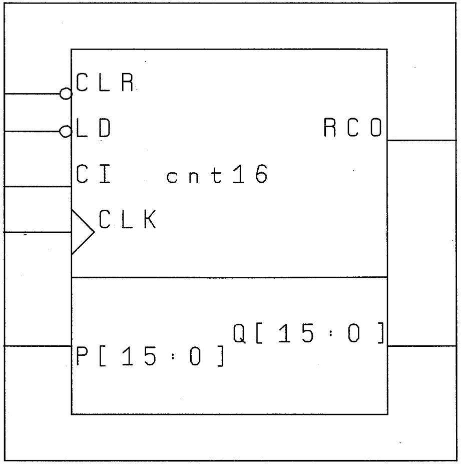
To create a 16-bit user soft macro you do the following steps:

- Open a schematic window with your CAE tool (e.g. Viewdraw I Schematic)
- Use the 4-bit counter soft macro TA161 from the library to design a 16-bit counter as if you were designing with a LS161
- Write the file to create a netlist. Viewlogic saves this file under a /SCH sub-directory
- Open a symbol window (e.g. Viewdraw I Symbol) and draw a symbol for the 16-bit counter
- Write the symbol file with the same name as the schematic file name. Viewlogic saves this file under a /SYM sub-directory
- You can use your user soft macro in a similar way to a soft macro from the library

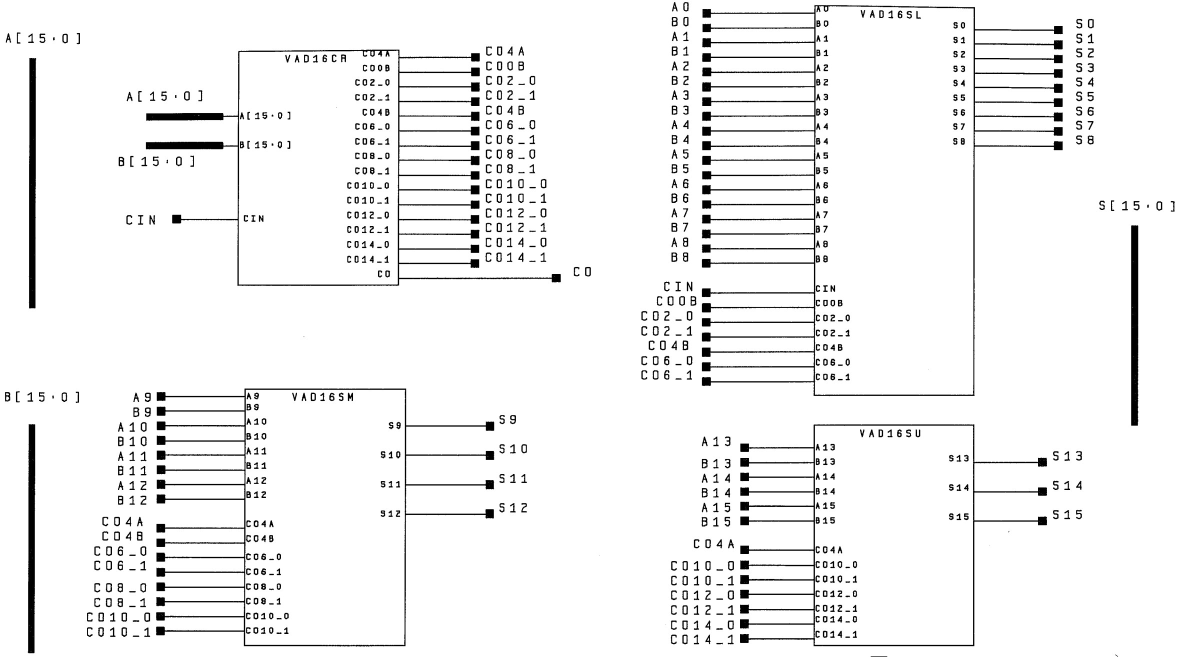
16-Bit Counter With 4 TA161



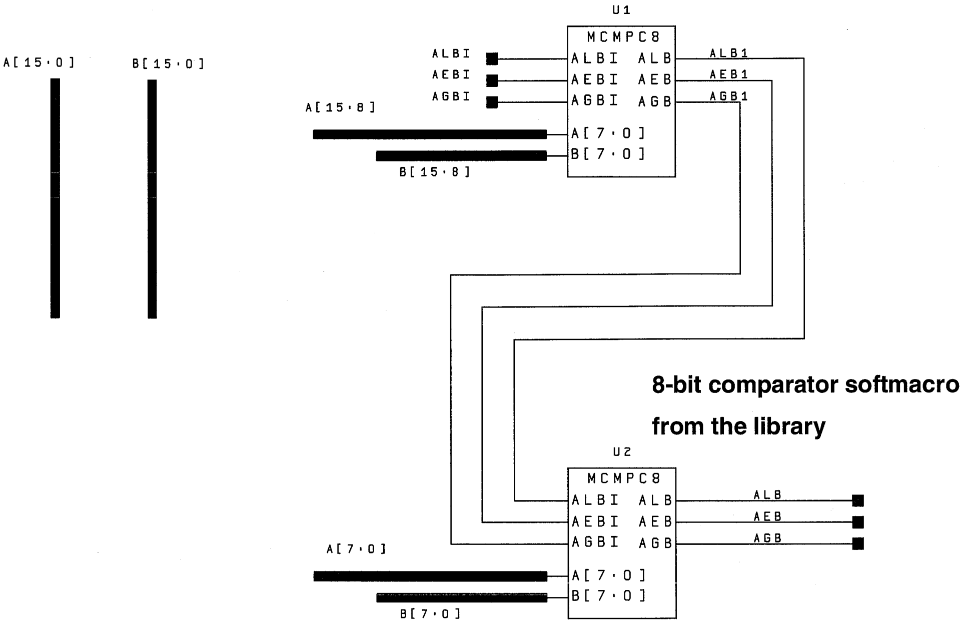
16-Bit Counter User Symbol



16-Bit Adder Soft Macro

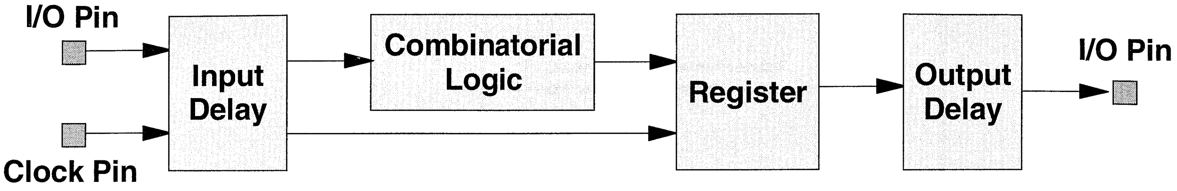


16-Bit Comparator



8-bit comparator softmacro from the library

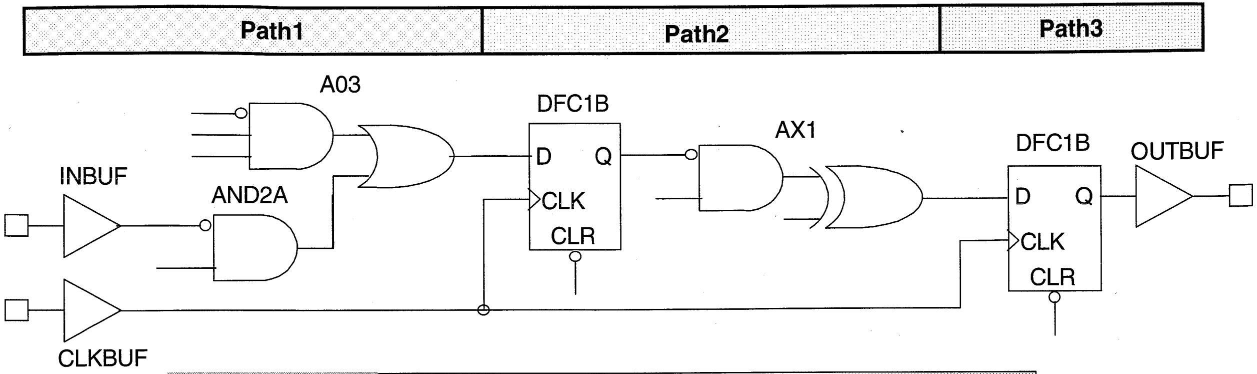
FPGA Timing Analysis



Propagation delay time (tpd) depends on:

- Levels of combinatorial logic
- Length of routing tracks used
- Number of antifuse elements used
- Number of logic modules being driven (FANOUT)

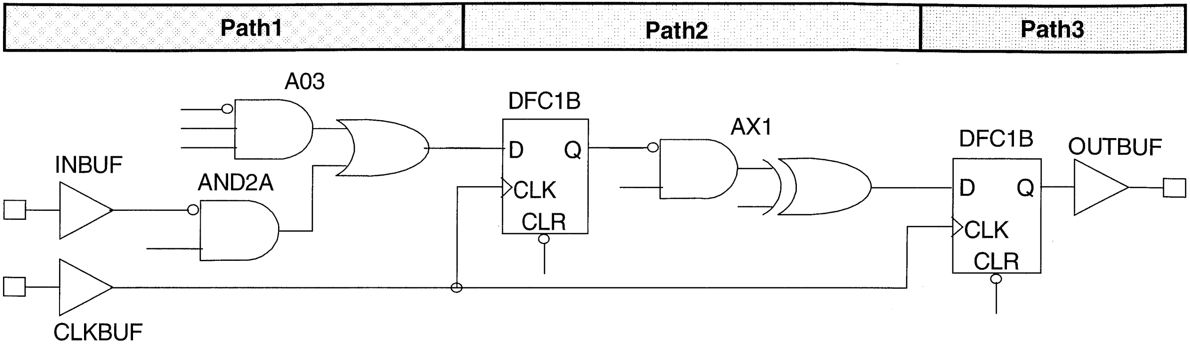
Combinatorial Logic Timing



Array Modules Used	
TPC10	2 Combinatorial
TPC12	2 Combinatorial

Path 1 Delay =	INBUF +	AND2A +	A03 +	$\text{DFC1B}_{\text{tsu}}$		
TPC10 =	10.6 +	9.7 +	9.7 +	3.9	=	33.6 ns
TPC12 =	9.2 +	8.0 +	0.0 +	0.4	=	17.6 ns

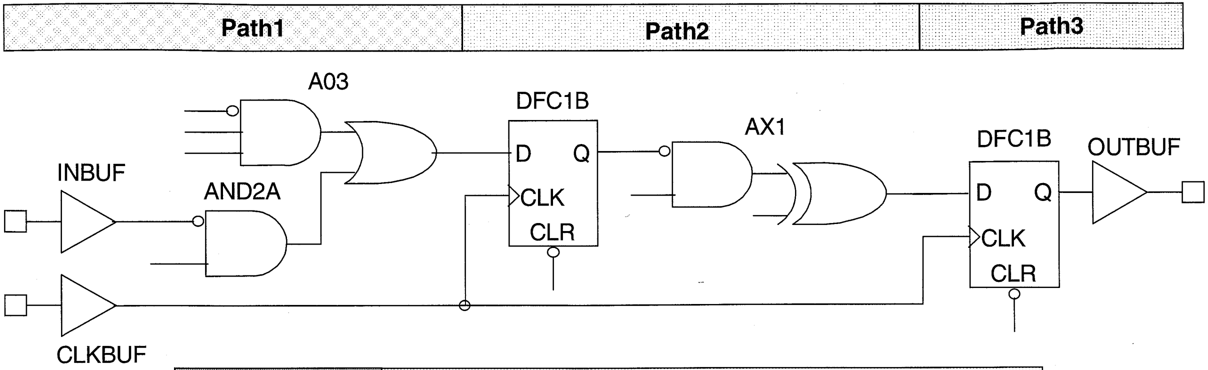
Sequential Logic Timing



Array Modules Used		
	TPC10	3 Combinatorial
	TPC12	1 Combinatorial and 1 Sequential

Path 2 Delay =	DFC1B	+	AX1	+	DFC1B_{tsu}		
TPC10 =	9.7	+	9.7	+	3.9	=	23.3 ns
TPC12 =	8.0	+	8.0	+	0.4	=	16.4 ns

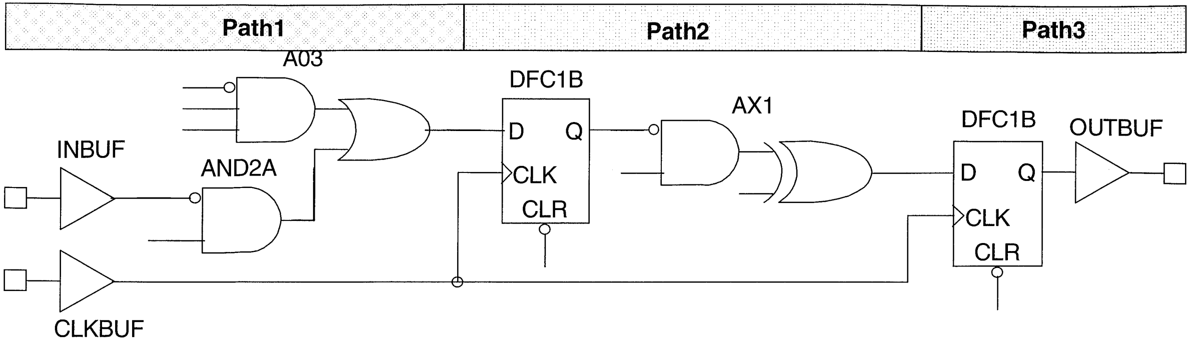
I/O Module Timing



Array Modules Used		
	TPC10	2 Combinatorial
	TPC12	1 Sequential

Path 3 Delay = DFC1B + OUTBUF
TPC10 = 9.7 + 11.2 = 20.9 ns
TPC12 = 8.0 + 11.6 = 19.6 ns

Maximum Frequency



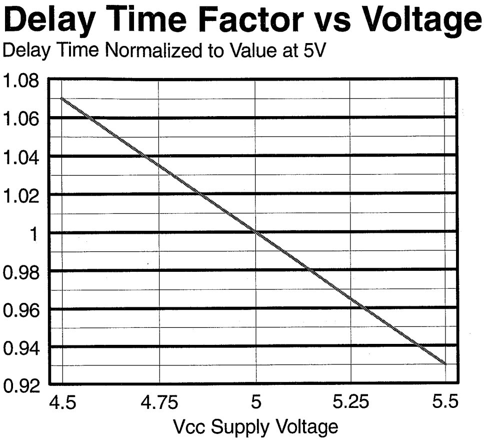
Total Array Modules Used	
TPC10	7 Combinatorial
TPC12	3 Combinatorial and 2 Sequential

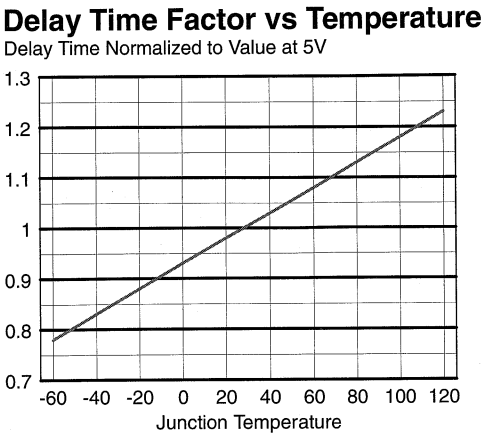
Maximum Frequency

TPC10 = 29.8 MHz

TPC12 = 51.0 MHz

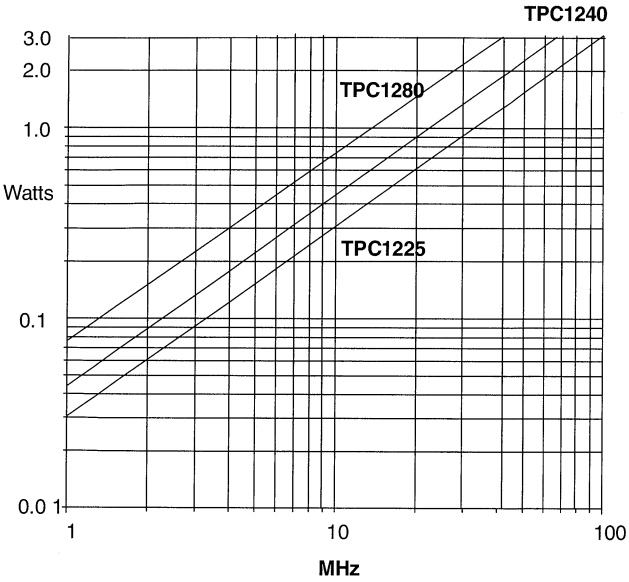
Timing Derating Factor





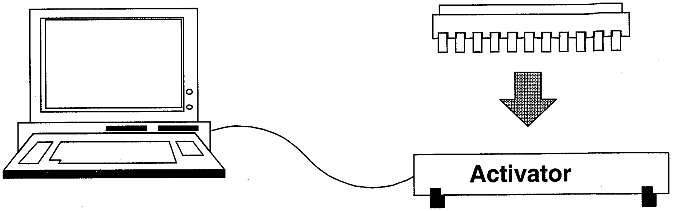
- Back-annotation timing simulation works with typical delays.
- Worst-case timing simulation can be made by using a delay derating factor.
- Worst case Timing Derating Factor (min voltage, max temperature, worst case process):
  C Suffix: 1.40 **I Suffix: 1.50** **M Suffix: 1.60**

Power Estimates TPC12 Series



- Maximum static power for commercial device
  Static power = 10mA x 5.25V = 52.5mW
- Active power depends on the frequency and the utilization of the modules
- The power graphs provide a simple guideline for estimating power

FPGA Device Programming



Blank Check

- Check for correct device
- Verifies each antifuse is unprogrammed
- Displays fuse checksum if programmed

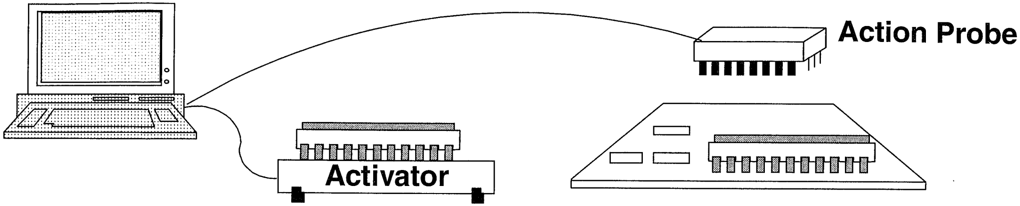
Device Programming

- Programming pulses applied sequentially
- Programming time between 5 and 20 minutes
- Dynamically verifies each programmed fuse

Design Protection

- Security fuse
- Probe fuse

Test & Debug



In the Activator:

- Debug allows functional testing of the programmed FPGA by applying input test vectors and monitoring the outputs
- Allows testing of all internal nodes

In the prototype board:

- Actionprobe adapter allows testing of the programmed FPGA under real operation (in-circuit-test)
- Allows display of all internal nodes on an oscilloscope or a logic analyzer

Kit1e - FPGA Design Kit up to 2500 Gates

Description	Part Number
Viewlogic Viewdraw for schematic capture	TPC-ALS-VL-005
'Designer' ALS package for TPC10, TPC12, TPC14 series up to 2500 gates	TPC-ALS-DS-VL-PC
Viewlogic Viewsim simulator up to 3k gates	TPC-ALS-017
Single socket programmer for TPC10, TPC12, TPC12 (Activator 2S)	TPC-ALS-DS-P2S-PC

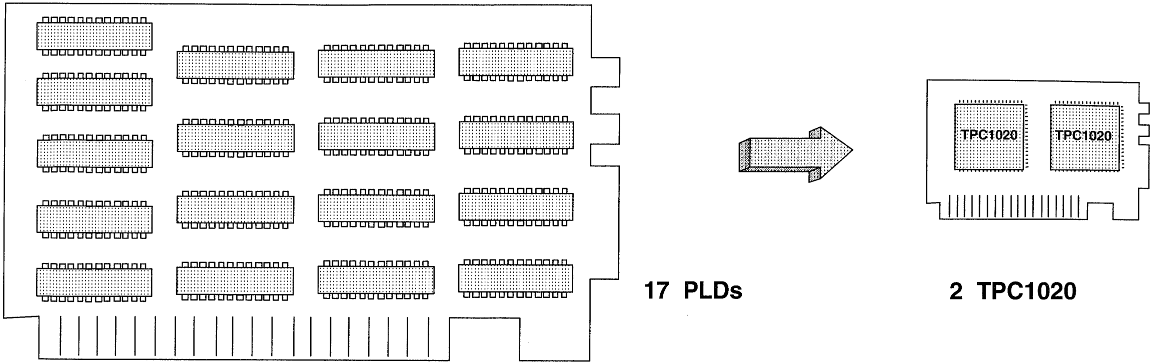
Summary

- Standard TTL functions can be replaced easily by FPGA soft macros. A TTL board can be converted and programmed into a FPGA in a week.
- FPGA's offer greater flexibility for design modification. TTL design iterations require a new PCB each time.
- FPGA's offer considerable power reduction
- FPGA's offer more logic integration at lower cost.

Agenda

- PLD/FPGA Architecture Analysis
- How To Replace TTL With FPGA
- ☞ How To Replace PLD With FPGA
- Applications Benchmarks
- FPGA Design With VHDL
- FPGA Products And Development Systems

Consolidating PLD with FPGA

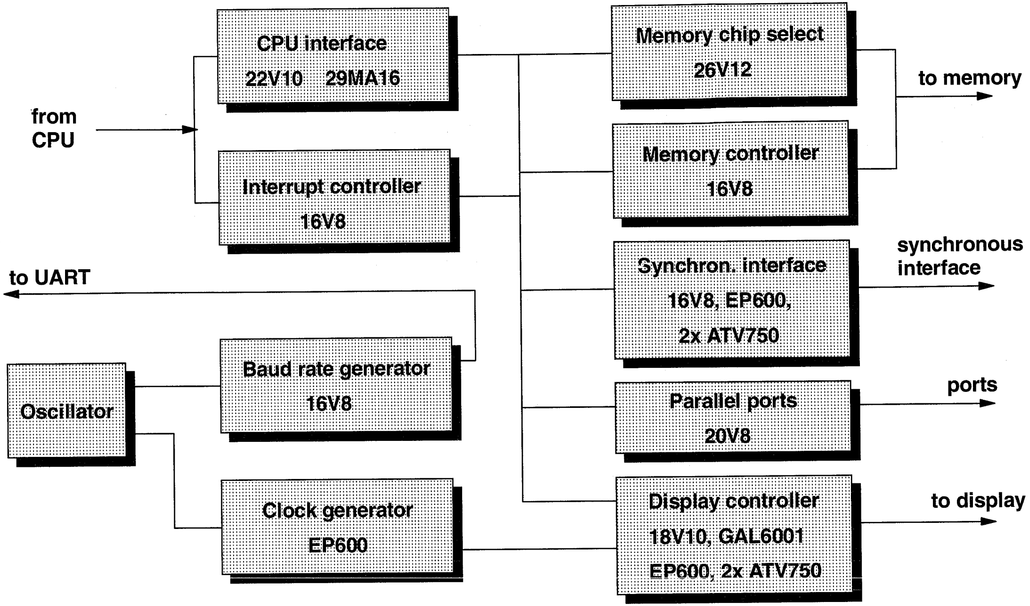


17 PLDs

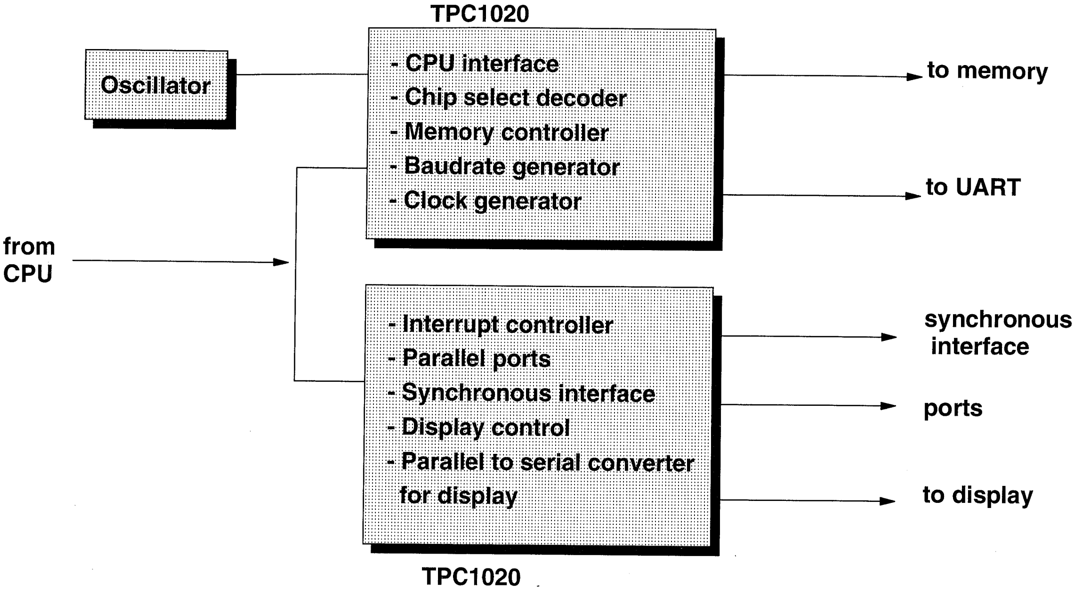
2 TPC1020

4	GAL16V8	1	PAL22V10	3	EP600
1	GAL18V10	1	PAL29MA16	4	ATV750
1	GAL20V8	1	PAL26V12		
1	GAL6001				

PLD Solution



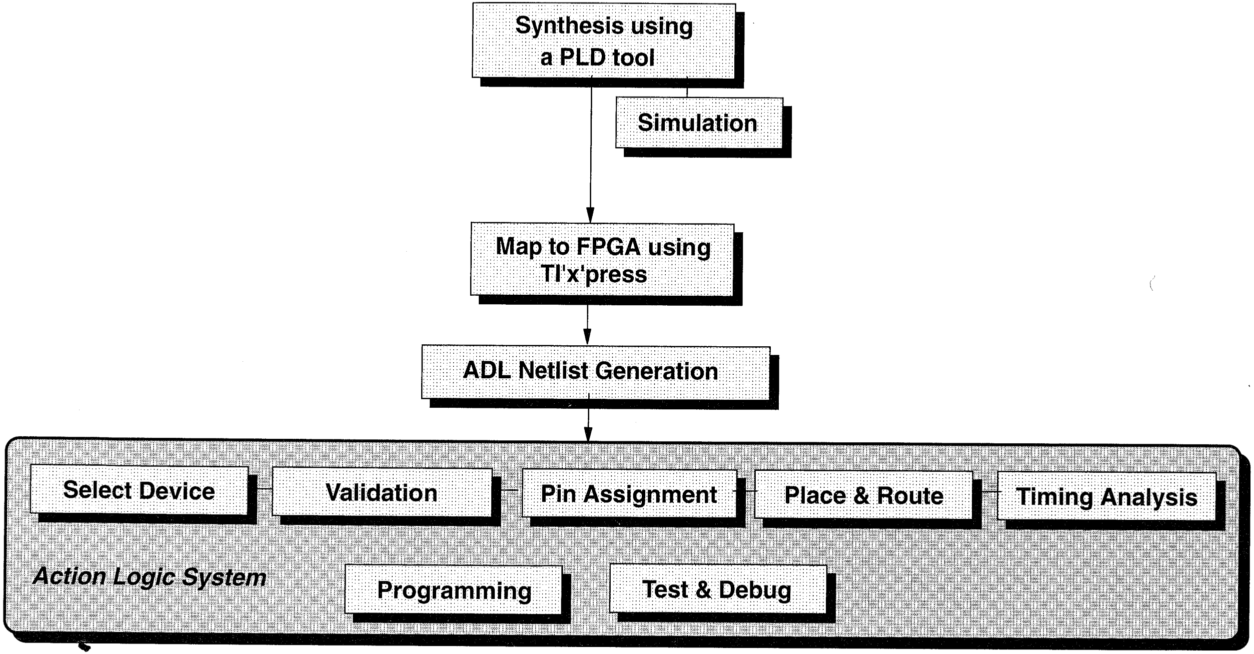
FPGA Alternative



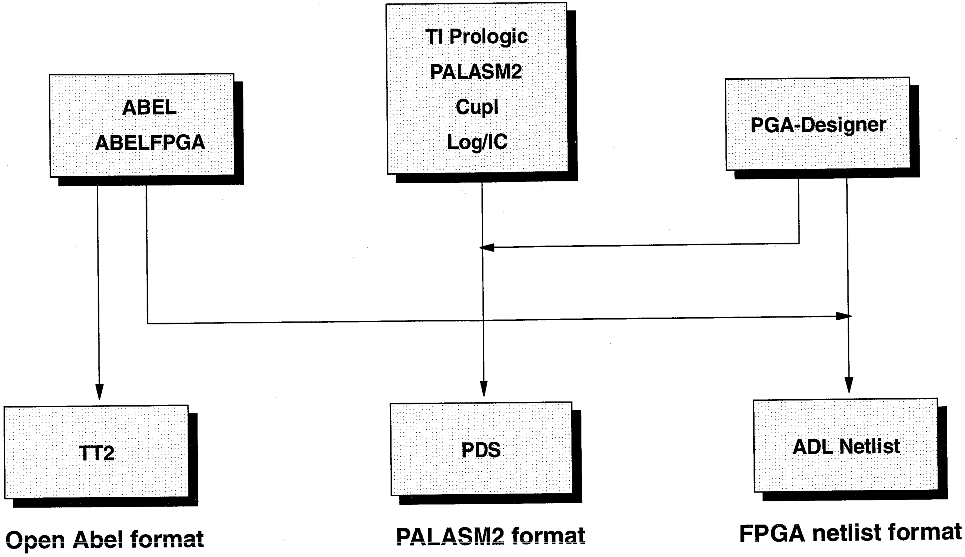
How To Replace PLD's With FPGA's

- PLD designs are normally described as Boolean equations, truth table or state machine using the syntax of a PLD design tool
- All PLD tools can generate reduced equations in PALASM2 format (.PDS)
- The reduced equations are optimized for the PAL AND-OR structure
- FPGA has architecturally-specific gates and flip-flops which yield efficient utilization of device resources
- The PLD equations are remapped and optimized for the FPGA hard macros with synthesis software

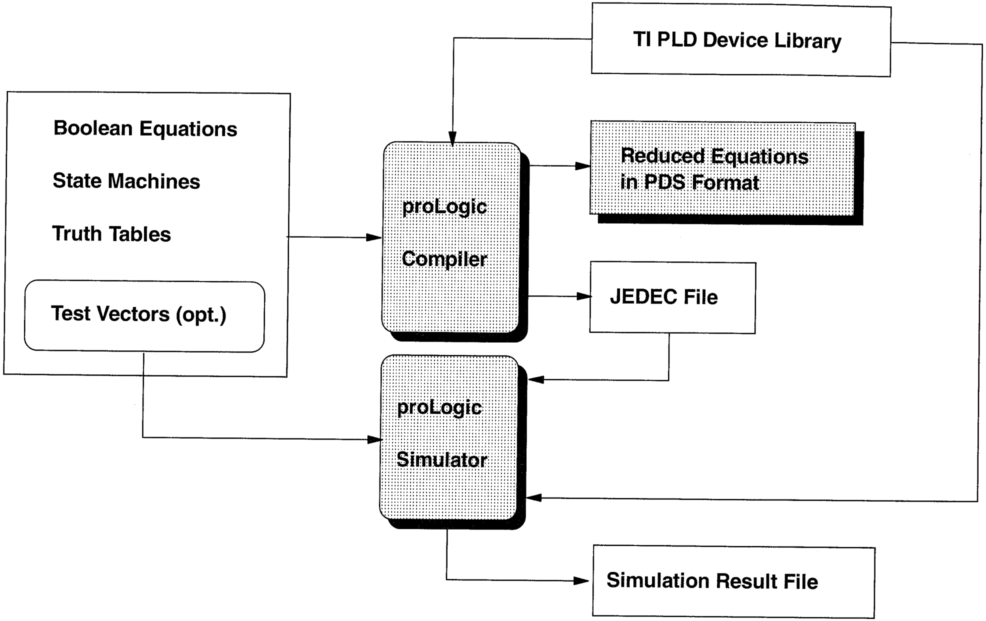
PLD To FPGA Design Flow



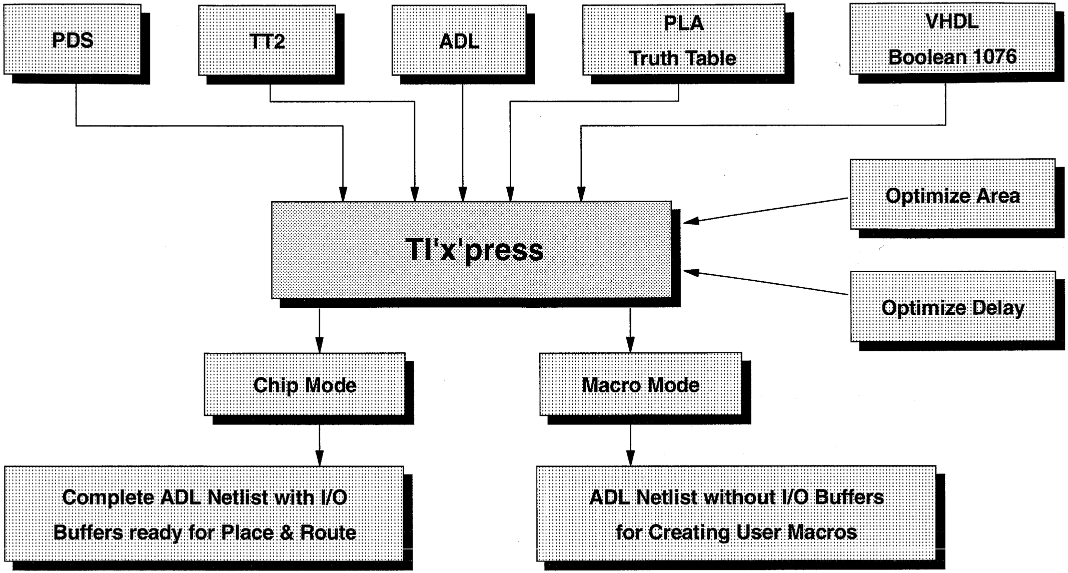
PLD Synthesis Tools



TI Prologic - PLD Synthesis Tool



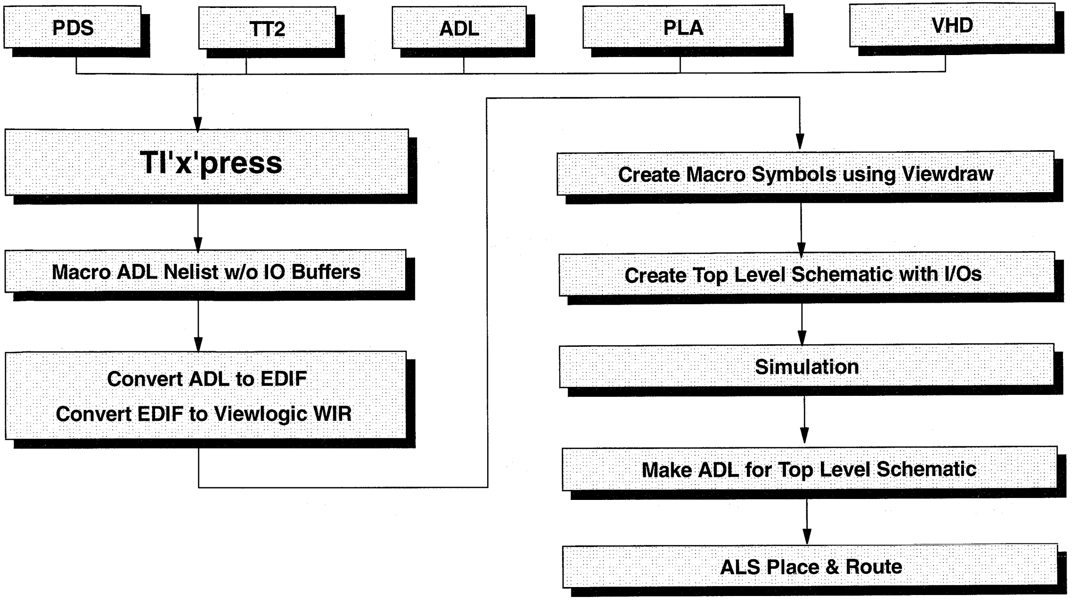
TI'x'press - FPGA Synthesis Tool



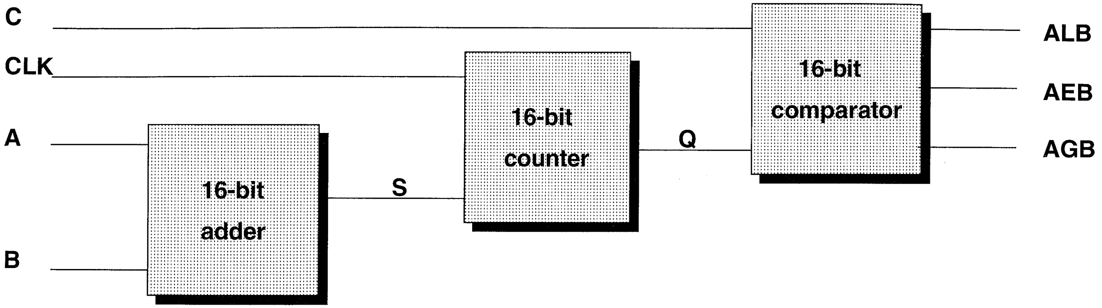
TI'x'press Features

- TI'x'press is a FPGA synthesis tool which can
  - use as inputs logic equations from ABEL, LOG/iC, PALASM, Cuple, PGADesigner, Prologic, or a FPGA netlist in ADL format ,
  - to optimize the logic for speed or area,
  - and map the logic to FPGA hard macros, and generate an ADL netlist.
- TI'x'press supports a subset of VHDL - the Boolean 1076 VHDL.
- TI'x'press allows two compilation modes
  - Chip mode synthesizes the whole design and adds input/output buffers in ADL netlist which can be used directly for Place & Route
  - Macro mode synthesizes portions of a design without input/output buffers which can be used in a CAD schematic capture environment as user soft macros (mixed mode entry.)
- TI'x'press integration language allows the interconnection of design blocks in text file without the need for a schematic editor.

TI'x'press Synthesis Flow - Macro Mode

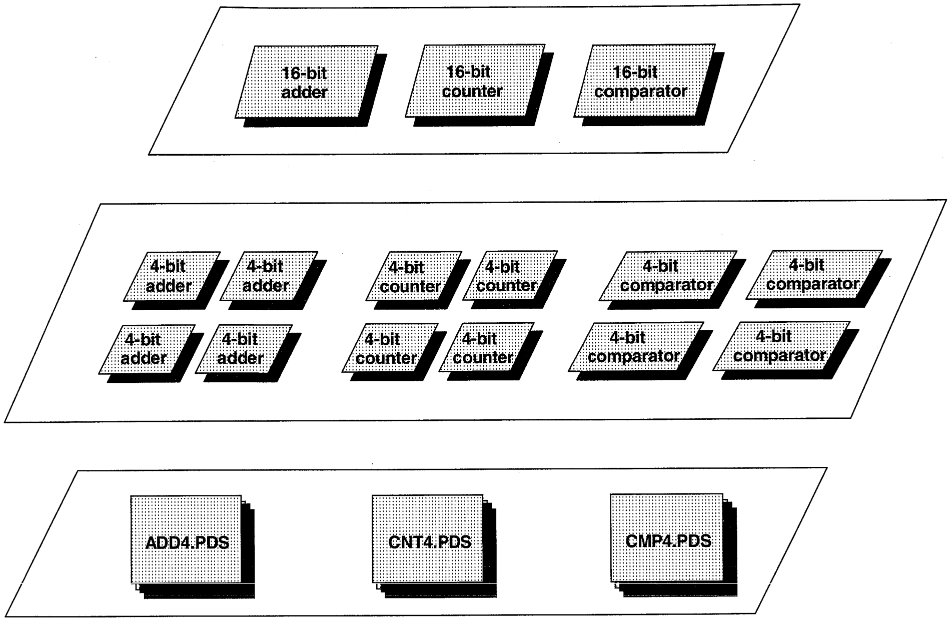

PDS
TT2
ADL
PLA
VHD
TI'x'press
Macro ADL Nelist w/o IO Buffers
Convert ADL to EDIF
Convert EDIF to Viewlogic WIR
Create Macro Symbols using Viewdraw
Create Top Level Schematic with I/Os
Simulation
Make ADL for Top Level Schematic
ALS Place & Route

A Case Study



- Two 16-bit numbers A and B are added.
- The sum S is loaded to a preloadable 16-bit counter.
- A 16-bit comparator compares the count Q with a 16-bit number C.
- The outputs are ALB (A<B), AEB (A=B) and AGB (A>B).

Design Hierarchy



Top level
includes softmacros
and I/O buffers

Second level
includes symbols
for softmacros

Equation level
PDS files
Source files

4-Bit Counter (TA161) Prologic File

```
title { Filename: cnt161l.pld
        Function: 4-bit loadable counter with asynchronous
                  clear, equivalent to TA161
        Designer: Dung Tu
        Date:     27.Dec.92
        Note:     22V10 is specified as device to produce a PDS file and
                  for simulation. The design is later converted to FPGA
                  using ALES.
}
include p22v10;
include XOR.H;
define  CLK = pin1;
define  CLR = pin2;   /* active low CLR */
define   LD = pin3;   /* active low LD  */
define  ENP = pin4;   /* carry look ahead parallel input */
define  ENT = pin5;   /* carry look ahead serial input */
/* load inputs */
define  P0 = pin6;
define  P1 = pin7;
define  P2 = pin8;
define  P3 = pin9;
```

```
/* counter outputs */
define  Q0 = pin14;
define  Q1 = pin15;
define  Q2 = pin16;
define  Q3 = pin17;
define  RCO = pin22;   /* carry-out for cascading */
/* 4-bit counter equations */
Q0.d =  !LD & P0 | LD & (Q0.q % (ENT & ENP));
Q1.d =  !LD & P1 | LD & (Q1.q % (Q0.q & ENT & ENP));
Q2.d =  !LD & P2 | LD & (Q2.q % (Q1.q & Q0.q & ENT & ENP));
Q3.d =  !LD & P3 | LD & (Q3.q % (Q2.q & Q1.q & Q0.q & ENT & ENP));
RCO  =  Q3.q & Q2.q & Q1.q & Q0.q & ENT;
Q0.RSTF = !CLR;
Q1.RSTF = !CLR;
Q2.RSTF = !CLR;
Q3.RSTF = !CLR;
```

4-Bit Adder Prologic File

```
title { add4pl   4-bit full adder using Prologic syntax
      Author:  Dung Tu Texas Instruments
      Date:    5.May.93'
}

include p22v10;
include xor.h;

define A3 = pin1;
define A2 = pin2;
define A1 = pin3;
define A0 = pin4;

define B3 = pin5;
define B2 = pin6;
define B1 = pin7;
define B0 = pin8;

define CIN = pin9;

define S3 = pin10;
define S2 = pin11;
define S1 = pin13;
define S0 = pin14;

define CO = pin15;
```

```
/* equations */

S0 = A0 % B0 % CIN;

S1 = A1 % B1 %
   ((A0 & B0)
    | (A0 | B0) & CIN);

S2 = A2 % B2 %
   ((A1 & B1)
   | (A1 | B1) & (A0 & B0)
   | (A1 | B1) & (A0 | B0) & CIN);

S3 = A3 % B3%
  ((A2 & B2)
   | (A2 | B2) & (A1 & B1)
   | (A2 | B2) & (A1 | B1) & (A0 & B0)
   | (A2 | B2) & (A1 | B1) & (A0 | B0) & CIN);

CO = (A3 & B3)
   | (A3 | B3) & (A2 & B2)
   | (A3 | B3) & (A2 | B2) & (A1 & B1)
   | (A3 | B3) & (A2 | B2) & (A1 | B1) & (A0 & B0)
   | (A3 | B3) & (A2 | B2) & (A1 | B1) & (A0 | B0) & CIN;
```

4-Bit Comparator Prologic File

```
title { Filename:   lcmp4l.pld
        Function:   4-bit comparator
        Designer:   Dung Tu
        Date:       5.May.93
        Note:       22v10 is specified as device to produce a
                    PDS file and for simulation. The design is
                    later converted to FPGA using ALES.
}

include p22v10;
include compare.h;

define A0 = pin1;
define A1 = pin2;
define A2 = pin3;
define A3 = pin4;

define B0 = pin9;
define B1 = pin10;
define B2 = pin11;
define B3 = pin13;

define AEBI = pin21;
define AGBI = pin22;
define ALBI = pin23;
```

```
define AEB = pin17;
define AGB = pin18;
define ALB = pin19;

define A = (A0, A1, A2, A3);
define B = (B0, B1, B2, B3);

AEB = (A == B) & AEBI;
AGB = (A > B) & AGBI;
ALB = (A < B) & ALBI;
•
```

4-Bit Adder - Sum Bit S3

```
S3 = (A3 * /B3 * /B2 * /B1 * /B0 * /CIN
   + /A3 * B3 * /B2 * /B1 * /B0 * /CIN
   + A3 * /A2 * /B3 * /B1 * /B0 * /CIN
   + /A3 * /A2 * B3 * /B1 * /B0 * /CIN
   + A3 * /A1 * /B3 * /B2 * /B0 * /CIN
   + /A3 * /A1 * B3 * /B2 * /B0 * /CIN
   + A3 * /A2 * /A1 * /B3 * /B0 * /CIN
   + /A3 * /A2 * /A1 * B3 * /B0 * /CIN
   + A3 * /A0 * /B3 * /B2 * /B1 * /CIN
   + /A3 * /A0 * B3 * /B2 * /B1 * /CIN
   + A3 * /A2 * /A0 * /B3 * /B1 * /CIN
   + /A3 * /A2 * /A0 * B3 * /B1 * /CIN
   + A3 * /A1 * /A0 * /B3 * /B2 * /CIN
   + /A3 * /A1 * /A0 * B3 * /B2 * /CIN
   + A3 * /A2 * /A1 * /A0 * /B3 * /CIN
   + /A3 * /A2 * /A1 * /A0 * B3 * /CIN
   + /A3 * /B3 * B2 * B1 * B0 * CIN
   + A3 * B3 * B2 * B1 * B0 * CIN
   + /A3 * A2 * /B3 * B1 * B0 * CIN
   + A3 * A2 * B3 * B1 * B0 * CIN
   + /A3 * A1 * /B3 * B2 * B0 * CIN
   + A3 * A1 * B3 * B2 * B0 * CIN
   + /A3 * A2 * A1 * /B3 * B0 * CIN
   + A3 * A2 * A1 * B3 * B0 * CIN
   + /A3 * A0 * /B3 * B2 * B1 * CIN
   + A3 * A0 * B3 * B2 * B1 * CIN
   + /A3 * A2 * A0 * /B3 * B1 * CIN
   + A3 * A2 * A0 * B3 * B1 * CIN
   + /A3 * A1 * A0 * /B3 * B2 * CIN
   + A3 * A1 * A0 * B3 * B2 * CIN
   + /A3 * A2 * A1 * A0 * /B3 * CIN
```

```
+ A3 * A2 * A1 * A0 * B3 * CIN
+ A3 * /A0 * /B3 * /B2 * /B1 * /B0
+ /A3 * /A0 * B3 * /B2 * /B1 * /B0
+ A3 * /A2 * /A0 * /B3 * /B1 * /B0
+ /A3 * /A2 * /A0 * B3 * /B1 * /B0
+ A3 * /A1 * /A0 * /B3 * /B2 * /B0
+ /A3 * /A1 * /A0 * B3 * /B2 * /B0
+ A3 * /A2 * /A1 * /A0 * /B3 * /B0
+ /A3 * /A2 * /A1 * /A0 * B3 * /B0
+ /A3 * A0 * /B3 * B2 * B1 * B0
+ A3 * A0 * B3 * B2 * B1 * B0
+ /A3 * A2 * A0 * /B3 * B1 * B0
+ A3 * A2 * A0 * B3 * B1 * B0
+ /A3 * A1 * A0 * /B3 * B2 * B0
+ A3 * A1 * A0 * B3 * B2 * B0
+ /A3 * A2 * A1 * A0 * /B3 * B0
+ A3 * A2 * A1 * A0 * B3 * B0
+ A3 * /A1 * /B3 * /B2 * /B1
+ /A3 * /A1 * B3 * /B2 * /B1
+ A3 * /A2 * /A1 * /B3 * /B1
+ /A3 * /A2 * /A1 * B3 * /B1
+ /A3 * A1 * /B3 * B2 * B1
+ A3 * A1 * B3 * B2 * B1
+ /A3 * A2 * A1 * /B3 * B1
+ A3 * A2 * A1 * B3 * B1
+ A3 * /A2 * /B3 * /B2
+ /A3 * /A2 * B3 * /B2
+ /A3 * A2 * /B3 * B2
+ A3 * A2 * B3 * B2);
```

Test Vector Block

```
test_vectors{ CLK  CLR  Q3  Q2  Q1  Q0 ;   /* count */

               C    0    L   L   L   L ;   /* clear */
               C    1    L   L   L   H ;   /* 1 */
               C    1    L   L   H   L ;   /* 2 */
               C    1    L   L   H   H ;   /* 3 */
               C    1    L   H   L   L ;   /* 4 */
               C    1    L   H   L   H ;   /* 5 */
               C    1    L   H   H   L ;   /* 6 */
               C    1    L   H   H   H ;   /* 7 */
               C    1    H   L   L   L ;   /* 8 */
               C    1    H   L   L   H ;   /* 9 */
               C    1    H   L   H   L ;   /* 10 */
               C    1    H   L   H   H ;   /* 11 */
               C    1    H   H   L   L ;   /* 12 */
               C    1    H   H   L   H ;   /* 13 */
               C    1    H   H   H   L ;   /* 14 */
               C    1    H   H   H   H ;   /* 15 */
               C    1    L   L   L   L ;   /* 0 */
            }
```

- Test vectors are used by the proSim simulator to verify that the logic functions defined for the PLD are correct.

- Each test_vectors block defines a sequence of input and output patterns which are used to test the logic function.

- The first line of the table defines the test variables:

```
CLK  CLR  Q3  Q2  Q1  Q0;
```

- Each other line is a test vector:

```
C   0   L   L   L   L;
```

proLogic Source File With Test Vectors

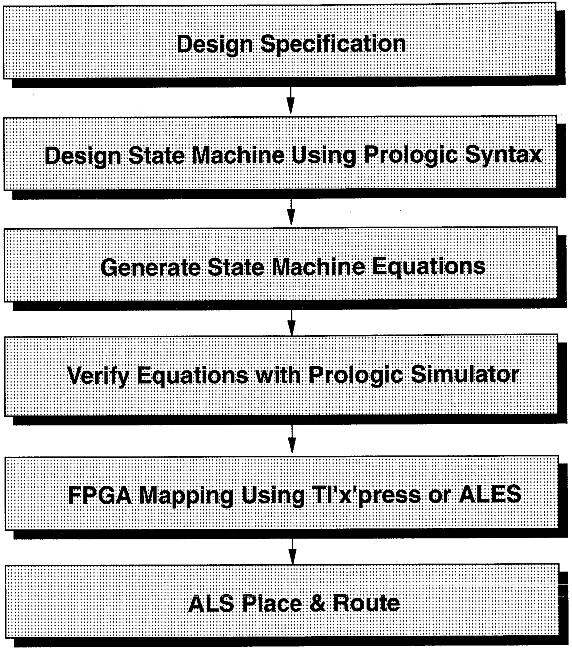
```
title {    Filename:    CNT4L.pld
           Function:    Simple 4-bit binary counter
           Designer:     Dung Tu
           Date:         1.Sep.92
           Note:         22V10 is specified as device to produce a
                        PDS file and for simulation. The design is
                        later converted to FPGA using ALES.
}

include p22v10;    /* Specify a PLD device type                        */
include XOR.H;     /* To implement the "%" operator as Exclusive OR
*/

/* Input pins */
define  CLK = pin1;
define  CLR = pin2;

/* Output pins */
define  Q0 = pin14;
define  Q1 = pin15;
define  Q2 = pin16;
define  Q3 = pin17;

/* Specify register outputs of the 22V10 as active high               */
Q0 = q;          Q1 = q;          Q2 = q;          Q3 = q;

/* The generic format for the n-th bit (Qn) of an n-bit counter is:   */
/* Qn  = Qn    %  ( Qn-1 &  Qn-2  &  ... &  Q0 )
/* proLogic uses "Q.d" for a flipflop input and "Q.q" for the output */
/* Equations for a 4-bit counter:                                     */

Q0.d  =  !Q0.q   &   CLR;
Q1.d  =  (Q1.q   %   Q0.q) & CLR;
Q2.d  =  (Q2.q   %   (Q1.q & Q0.q))   &   CLR;
Q3.d  =  (Q3.q   %   (Q2.q & Q1.q & Q0.q))   &   CLR;
```

```
/* Test vectors for proLogic functional simulator */

test_vectors {  CLK  CLR  Q3  Q2   Q1   Q0 ;  /* count */

                 C    0    L    L    L    L ;  /* clear */

                 C    1    L    L    L    H ;  /* 1 */
                 C    1    L    L    H    L ;  /* 2 */
                 C    1    L    L    H    H ;  /* 3 */
                 C    1    L    H    L    L ;  /* 4 */
                 C    1    L    H    L    H ;  /* 5 */
                 C    1    L    H    H    L ;  /* 6 */
                 C    1    L    H    H    H ;  /* 7 */
                 C    1    H    L    L    L ;  /* 8 */
                 C    1    H    L    L    H ;  /* 9 */
                 C    1    H    L    H    L ;  /* 10 */
                 C    1    H    L    H    H ;  /* 11 */
                 C    1    H    H    L    L ;  /* 12 */
                 C    1    H    H    L    H ;  /* 13 */
                 C    1    H    H    H    L ;  /* 14 */
                 C    1    H    H    H    H ;  /* 15 */
                 C    1    L    L    L    L ;  /* 0 */

}
```

Test Result File

```
proLogic Simulator
Texas Instruments V2.0
Copyright (C) 1991 Prologic Systems
Architecture Description: \prologic\p22v10.lxa
JEDEC Fuse Information:  cnt4l.jed
JEDEC Test Vectors:      cnt4l.jed

V01   C0NN NNNN NNNN NLLL LNNN NNNN
V02   C1NN NNNN NNNN NHLL LNNN NNNN
V03   C1NN NNNN NNNN NLHL LNNN NNNN
V04   C1NN NNNN NNNN NHHL LNNN NNNN
V05   C1NN NNNN NNNN NLLH LNNN NNNN
V06   C1NN NNNN NNNN NHLH LNNN NNNN
V07   C1NN NNNN NNNN NLHH LNNN NNNN
V08   C1NN NNNN NNNN NHHH LNNN NNNN
V09   C1NN NNNN NNNN NLLL HNNN NNNN
V10   C1NN NNNN NNNN NHLL HNNN NNNN
V11   C1NN NNNN NNNN NLHL HNNN NNNN
V12   C1NN NNNN NNNN NHHL HNNN NNNN
V13   C1NN NNNN NNNN NLLH HNNN NNNN
V14   C1NN NNNN NNNN NHLH HNNN NNNN
V15   C1NN NNNN NNNN NLHH HNNN NNNN
V16   C1NN NNNN NNNN NHHH HNNN NNNN
V17   C1NN NNNN NNNN NLLL LNNN NNNN

No errors detected with 17 Test Vectors.
```

State Machine Synthesis With proLogic



Design Specification

Design State Machine Using Prologic Syntax

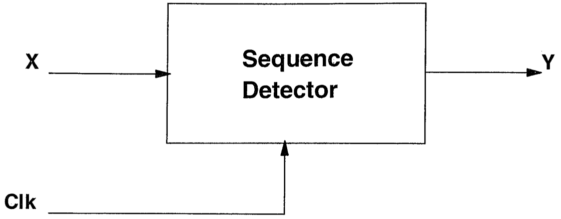
Generate State Machine Equations

Verify Equations with Prologic Simulator

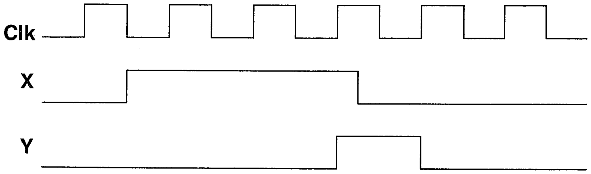
FPGA Mapping Using TI'x'press or ALES

ALS Place & Route

Bit Sequence Detector

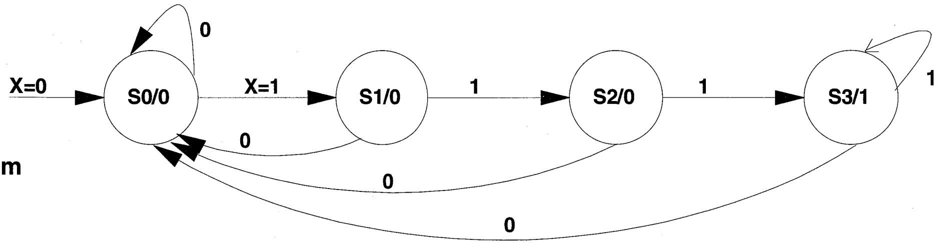


A Bit-Sequence Detector monitors an input signal X and outputs a signal Y=1 when a defined sequence of bits has been transmitted through the input of the system.



In this example the output Y becomes High when the input bit sequence is 111.

State Diagram: Graphical vs Syntax



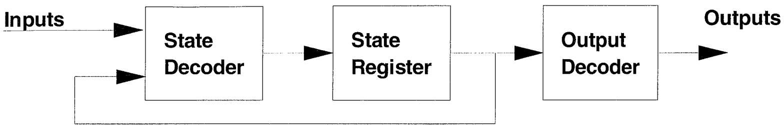
Graphical
State Diagram

State Diagram in
proLogic syntax

```
state_diagram Q1, Q0  {
      state s0 = 00 {
           Y = 0;
              if (X)        s1;
              else          s0;
      }
      state s1 = 01 {
             Y = 0;
              if (X)        s2;
              else          s0;
      }
state s2 = 10 {
       Y = 0;
        if (X)        s3;
        else          s0;
}
state s3 = 11 {
       Y = 1;
        if (X)        s3;
        else          s0;
}
}
```

Moore vs Mealy State Machine

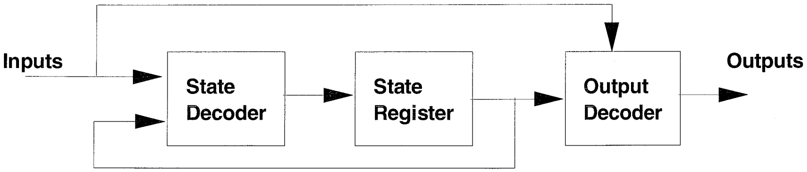
Moore State Machine



- Outputs depend only on the state.
- Changes in inputs must first change the machine state before they can change the outputs.

Synchronous state machine

Mealy State Machine



- Outputs depend on the state and the inputs.
- Changes in inputs affect the outputs immediately.

Asynchronous state machine

Moore Sequence Detector

```
title { Filename:    mooreseq.pld
        Function:    Sequence detector as Moore state machine
        Designer:    Dung Tu
        Date:        8.Sep.92
        Note:        22V10 is specified as device to produce a
                     PDS file and for simulation. The design is
                     later converted to FPGA using ALES.
}
include p22v10;
include dff;

/* Inputs */
define  CLK = pin1;
define    X = pin2;

/* State variables */
define  Q0 = pin14;
define  Q1 = pin15;

/* Output */
define   Y = pin18;

/* Registers active high */
Q0 = q;  Q1 = q;
```

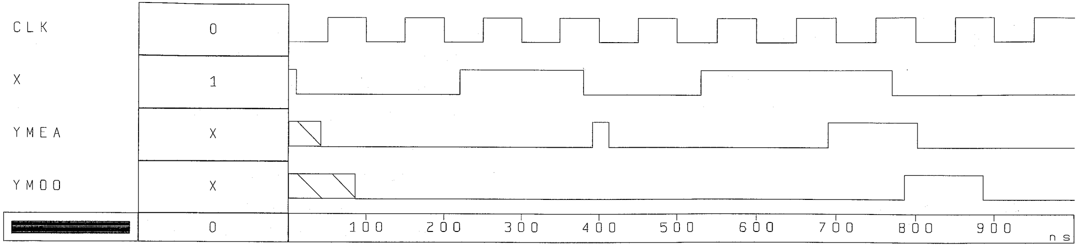
```
state_diagram Q1, Q0  {
    state s0 = 00 {
            Y = 0;
          if (X) s1;
      else      s0;
    }
    state s1 = 01 {
        Y = 0;
      if (X)    s2;
      else      s0;
    }
    state s2 = 10 {
        Y = 0;
      if (X)    s3;
      else      s0;
    }
    state s3 = 11 {
        Y = 1;
      if (X)    s3;
      else      s0;
    }
}
test_vectors { CLK   X   Y
;
        C   1   L;
        C   1   L;
        C   0   L;
        C   1   L;
        C   1   L;
        C   1   H;
        C   0   L;

}
```

Mealy Sequence Detector

```
title {     Filename:     mealyseq.pld
           Function:     Sequence detector as Mealy state machine
                 Designer:     Dung Tu
           Date:       8.Sep.92
                 Note:           22V10 is specified as device to produce a
                                 PDS file and for simulation. The design is
                                 later converted to FPGA using ALES.
}
include p22v10;
include dff;              /* for state machine synthesis */

/* Inputs */
define  CLK = pin1;
define    X = pin2;

/* State variables */
define  Q0 = pin14;
define  Q1 = pin15;

/* Output */
define   Y = pin18;

/* Registers active high */
Q0 = q;  Q1 = q;
```

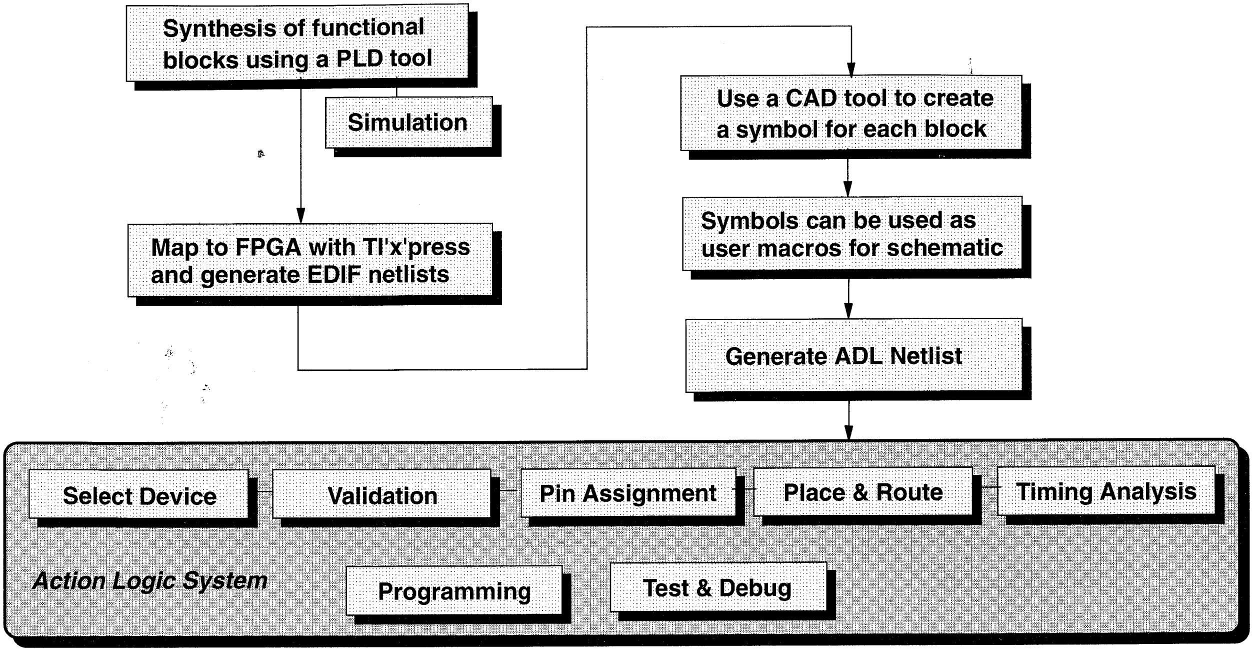
```
state_diagram Q1, Q0 {
    state s0 = 00
      if (X) { Y = 0;    s1; }
      else   { Y = 0;    s0; }
    state s1 = 01
      if (X) { Y = 0;    s2; }
      else   { Y = 0;    s0; }
    state s2 = 10
      if (X) { Y = 1;    s2; }
      else   { Y = 0;    s0; }
    state s3 = 11
        s0;
}
test_vectors { CLK  X   Y;
          C   0   L;
          C   1   L;
          C   1   H;
          C   0   L;
          C   1   L;
          C   1   H;
          C   1   H;
          C   0   L;
}
```

Waveform Mealy vs Moore



- Compared to the Moore machine, the output waveform of the Mealy machine is shifted to the left by one clock period.
- The falling edge of the output pulse is triggered by the falling edge of the input signal and not by the clock edge.
- To build a synchronous system, the input of a Mealy machine must be synchronized with a flip-flop.

Equations And Schematic Mixed Mode Entry



Synthesis of functional blocks using a PLD tool

Simulation

Map to FPGA with TI'x'press and generate EDIF netlists

Use a CAD tool to create a symbol for each block

Symbols can be used as user macros for schematic

Generate ADL Netlist

Select Device

Validation

Pin Assignment

Place & Route

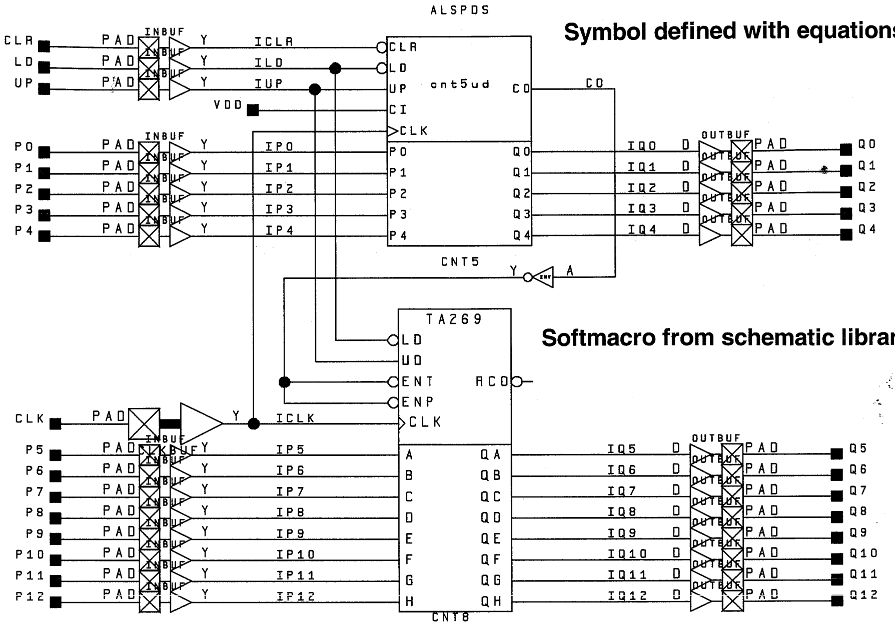
Timing Analysis

Action Logic System

Programming

Test & Debug

Combine Logic Synthesis With Schematic



Easy Logic Integration

FPGA

Summary

- FPGA architecture enables better use of silicon resource compared to GAL's and macrocell-based complex PLD's.

- Structure allows implementation of a wider range of functions.

- Synthesis enables designers to use familiar PLD design entry techniques to target TI FPGA's.

- PLD consolidation saves space, power and cost.

Agenda

- *PLD/FPGA Architecture Analysis*
- *How To Replace TTL With FPGA*
- *How To Replace PLD With FPGA*
- ☞ *Applications Benchmarks*
- *FPGA Design With VHDL*
- *FPGA Products And Development Systems*

PREP

- PREP (Programmable Electronics Performance Corporation) is a non-profit consortium of 13 vendors of programmable logic hardware and software.
- PREP develops standard benchmarks which allow a fair analysis and comparison of the performance and functional abilities of programmable logic devices.
- The first PREP Benchmark Suite Version 1.2 which was announced on March 29, 1993 in Santa Clara is composed of nine commonly used applications which are neutral to all architectures and vendors.

PREP Benchmarks

1. Data Path

   Eight 4-to-1 multiplexers drive an 8-bit register. This register drives a parallel loade 8-bit shift register.

2. Timer / Counter

   Two 8-bit words from a common data bus drive a 2-to-1 multiplexer whose output loads an 8-bit binary counter. The counter outputs are compared against an 8-bit register which is loaded from the data bus.

3. Small State Machine

   A state machine with 8 states, whose outputs are a function of the current state and the 8 input data bits.

4. Large State Machine

   A state machine with 16 states, 40 transitions and 8 inputs.

5. Arithmetic

   A 4-by-4 unsigned multiplier drives an 8-bit accumulator. The accumulator has a control input that passes either the multiplier data to the accumulator register or the accumulator's adder data to the register.

6. 16-bit accumulator

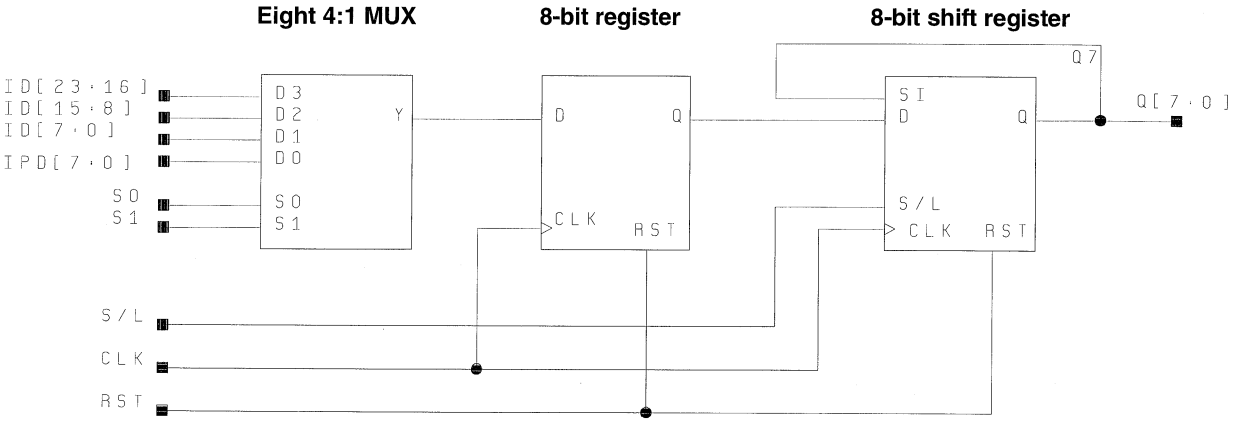
7. 16-bit loadable binary counter

8. 16-bit pre-scaled binary counter

9. Memory map

   The map decodes address spaces ranging in size from 4k down to one. Addresses that do not correspond to any space assert a bus error.

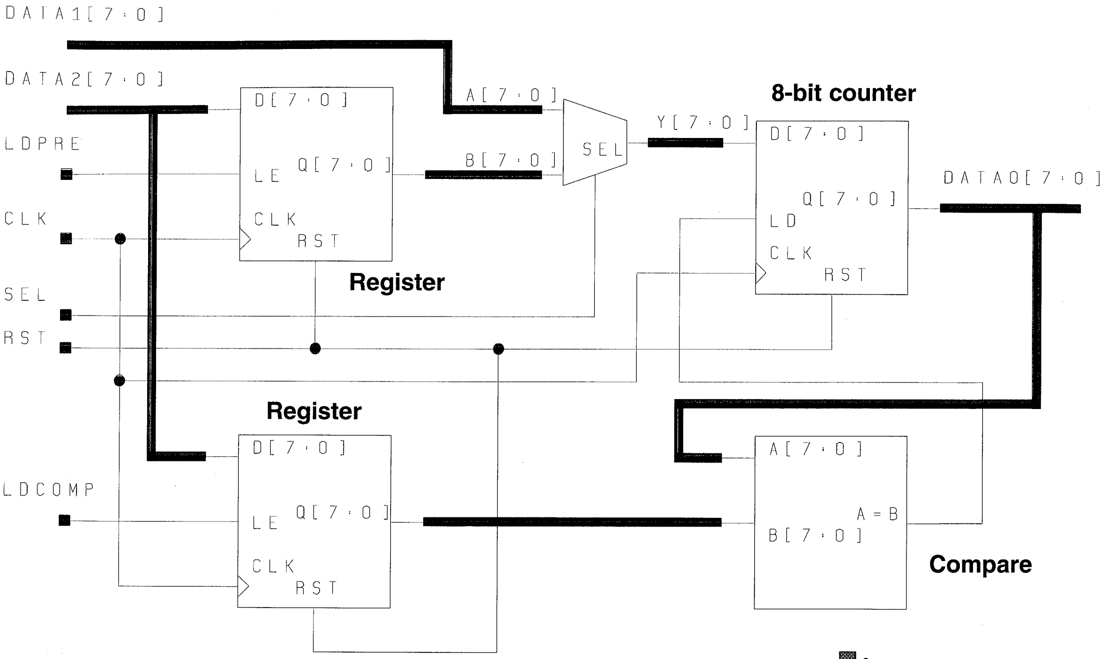
Benchmark #1 : Data Path



Data Path - Capacity / Performance

Device	Number of circuits	Worst-case speed (MHz)	Optimized for capacity	Optimized for speed
TPC1280-1	48	29	X	
TPC1280-1	39	55		X
TPC1240	21	66		X
TPC1240A-2	27	49	X	
TPC1240	27	37	X	
TPC1240A-2	21	95		X
TPC1225-1	18	44	X	
TPC1225-1	14	85		X

Benchmark #2 : Timer / Counter



Easy Logic Integration

FPGA

Timer / Counter : Capacity / Performance

Device	Number of circuits	Worst-case speed [MHz]
TPC1225-1	9	28
TPC1240A-2	14	35
TPC1240	14	26
TPC1280-1	20	19

Benchmark #3 : Small State Machine

Current State	Next State	IN[7:0] hex	OUT[7:0] hex
START	SA	3C	82
START	START	/3C	00
SA	SC	2A	40
SA	SB	1F	20
SA	SA	/1F * /2A	04
SB	SE	AA	11
SB	SF	/AA	30
SC	SD	XX	08
SD	SG	XX	80
SE	START	XX	40
SF	SG	XX	02
SG	START	XX	01

Small State Machine : Capacity / Performance

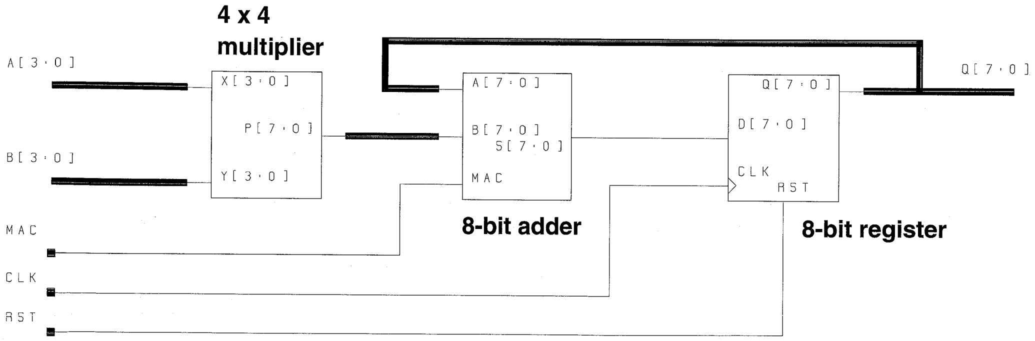
Device	Number of circuits	Worst-case speed [MHz]
TPC1225-1	15	32
TPC1240	23	28
TPC1240A-2	23	37
TPC1280-1	42	26

Benchmark #4 : Large State Machine

- 16 states
- 40 transitions
- 10 inputs
- 8 outputs

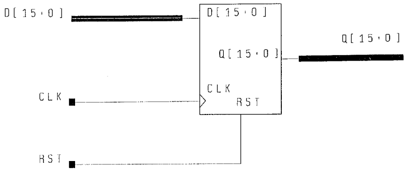
Device	Number of circuits	Worst-case speed [MHz]
TPC1225-1	6	21
TPC1240	10	17
TPC1240A-2	10	22
TPC1280-1	18	16

Benchmark #5 : Arithmetic Circuit



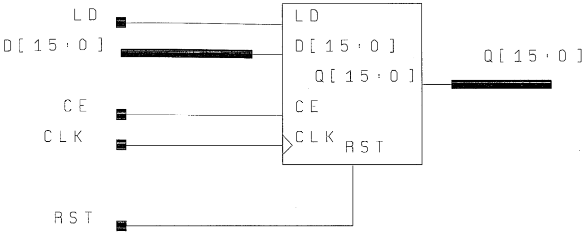
Device	Number of circuits	Worst case speed [MHz]
TPC1225-1	6	13
TPC1240A-2	9	14
TPC1240	9	11
TPC1280-1	16	11

Benchmark #6 : 16-Bit Accumulator



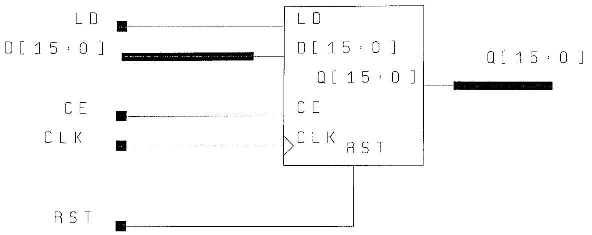
Device	Number of circuits	Worst-case speed [MHz]	Optimized for capacity	Optimized for speed
TPC1225-1	5	23	X	
TPC1240	8	21	X	
TPC1240A-2	8	27	X	
TPC1280-1	15	18	X	
TPC1225-1	4	32		X
TPC1240	7	24		X
TPC1240A-2	7	32		X
TPC1280-1	13	21		X

Benchmark #7 : 16-Bit Up-Counter



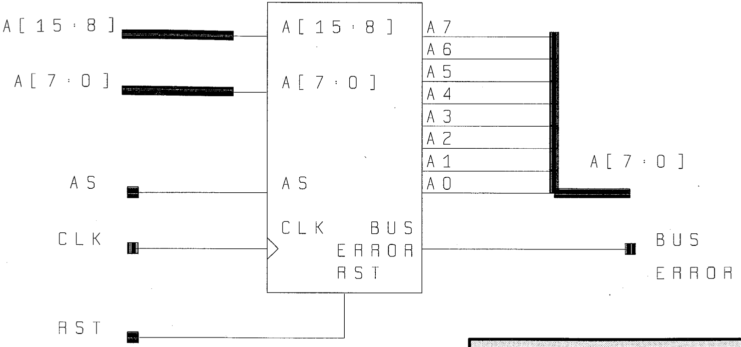
Device	Number of circuits	Worst-case speed [MHz]
TPC1225-1	9	46
TPC1240	13	39
TPC1240A-2	13	52
TPC1280-1	25	33

Benchmark #8 : 16-Bit Prescaled Counter



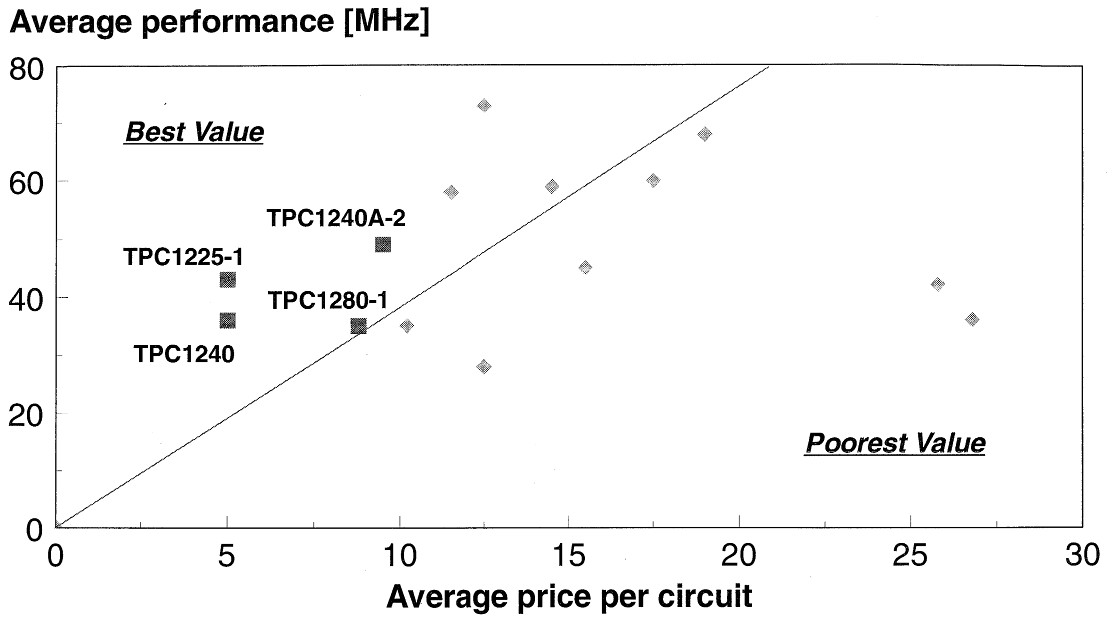
Device	Number of circuits	Worst-case speed [MHz]
TPC1225-1	6	85
TPC1240	10	66
TPC1240A-2	10	95
TPC1280-1	18	75

Benchmark #9 : Memory Map



Device	Number of circuits	Worst-case speed [MHz]
TPC1225-1	20	33
TPC1240	31	37
TPC1240A-2	31	37
TPC1280-1	56	26

FPGA Performance vs Cost



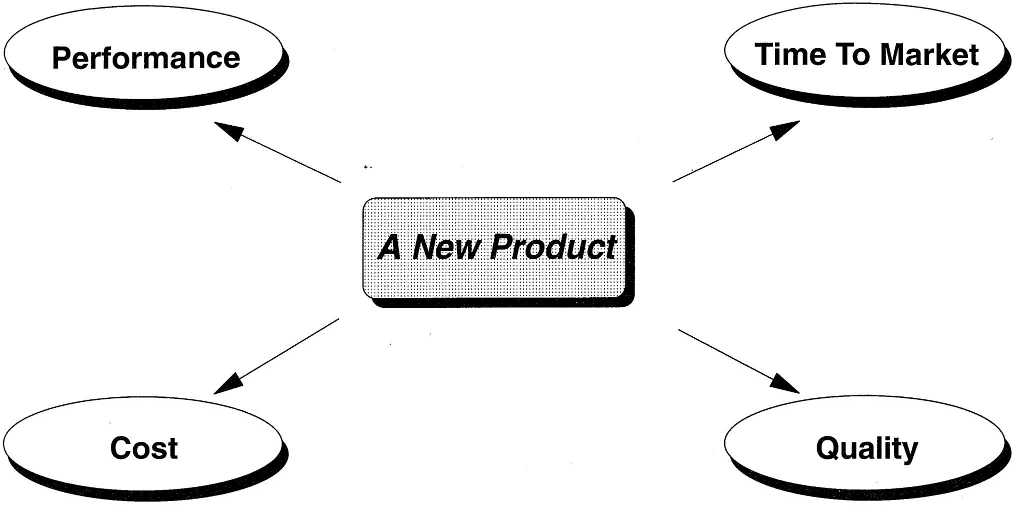
Summary

- PREP Benchmarks are useful for understanding and comparing the performance and functional abilities of programmable logic products.
- The best product for a user is the product which fits the application in terms of performance, density, power consumption and cost.
- Antifuse based FPGAs from Texas Instruments are production proven and offer an execellent cost / performance benefit for many applications.

Agenda

- *PLD/FPGA Architecture Analysis*
- *How To Replace TTL With FPGA*
- *How To Replace PLD With FPGA*
- *Applications Benchmarks*
- ☞ *FPGA Design With VHDL*
- *FPGA Products And Development Systems*

Success Factors



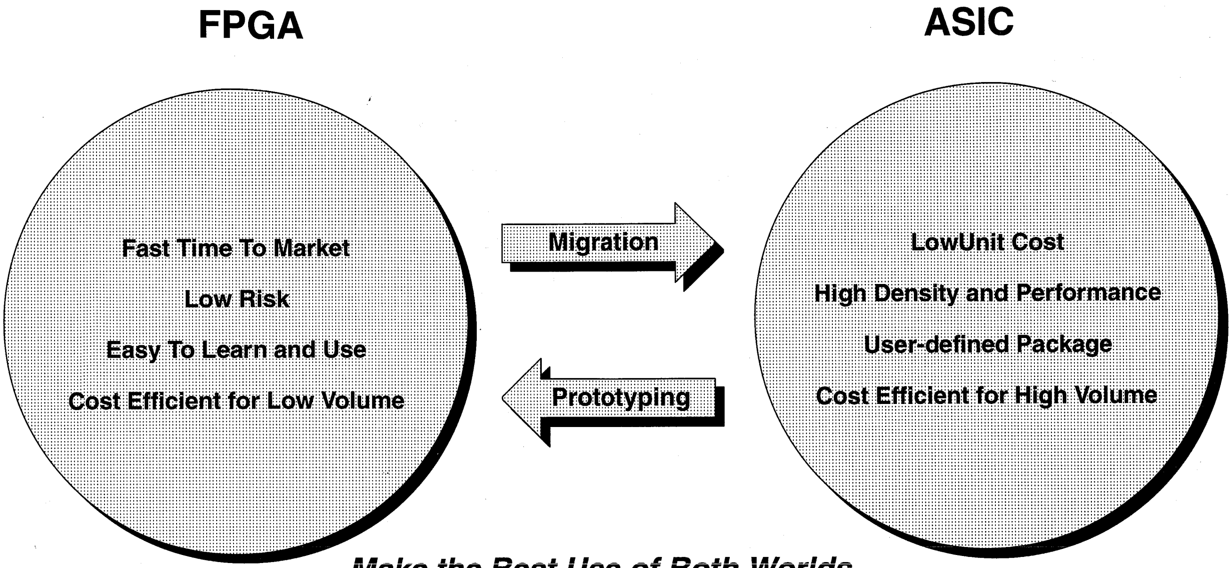
The Designer's Pressures

- Decreasing product life cycle.
- Increasing time-to-market pressure .
- Increasing design complexity and performance.
- Increasing product differentiation pressure.
- Increasing cost pressure.
- Increasing quality / reliability pressure.
- Increasing global competition.

Design Methodology Trends

- Trend to high complexity application specific products.
- Trend to high level design languages to master the complexity and reduce the development time.
- Trend to technology independent design methodology to use the best appropriate silicon for each phase of the product.
- Trend to automatic test synthesis to reduce the development time and improve the quality.
- Trend to board and system level simulation.

Why Technology Independent?



FPGA

Fast Time To Market

Low Risk

Easy To Learn and Use

Cost Efficient for Low Volume

Migration

Prototyping

ASIC

LowUnit Cost

High Density and Performance

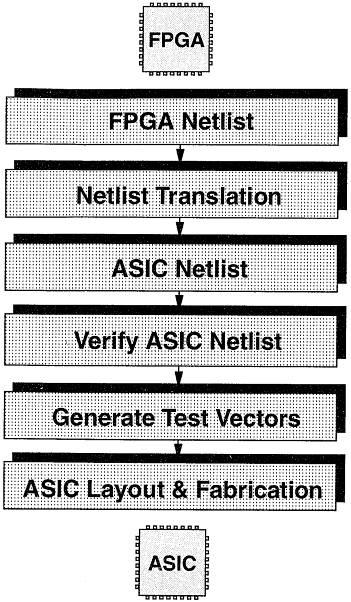
User-defined Package

Cost Efficient for High Volume

Make the Best Use of Both Worlds

Migration Options

Low Level Netlist Migration



FPGA

FPGA Netlist

Netlist Translation

ASIC Netlist

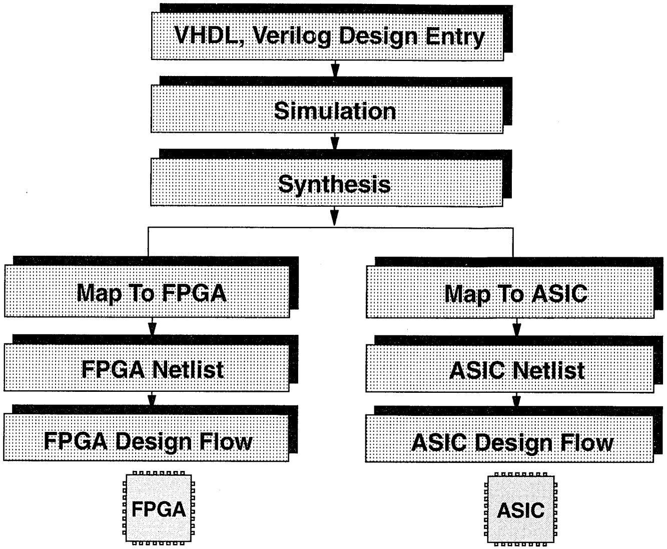
Verify ASIC Netlist

Generate Test Vectors

ASIC Layout & Fabrication

ASIC

High Level Design Migration



VHDL, Verilog Design Entry

Simulation

Synthesis

Map To FPGA

FPGA Netlist

FPGA Design Flow

FPGA

Map To ASIC

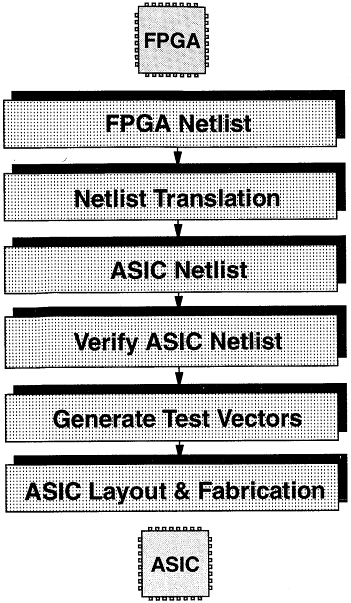
ASIC Netlist

ASIC Design Flow

ASIC

Low Level Netlist Migration

Low Level Netlist Migration



FPGA

FPGA Netlist

Netlist Translation

ASIC Netlist

Verify ASIC Netlist

Generate Test Vectors

ASIC Layout & Fabrication

ASIC

- ***Advantages:***
  - Straight forward FPGA implementation (without bothering about the ASIC)
  - Low cost FPGA tool
  - Familiar schematic entry approach

- ***Disadvantages:***
  - Could cause timing problem with the ASIC (asynchronous signals)
  - Potential testability problem with the ASIC

Designing for Migration

- Rule for automatic scan chain insertion:
  - Design should be synchronous as possible
  - Clock should not be divided or gated with other signals
  - All asynchronous preset and clear signals should be controllable from external pins
  - All register or latches should be sensitive to the same clock edge
- TI Migration Checklist:
  - Address testability issues to ensure high fault coverage

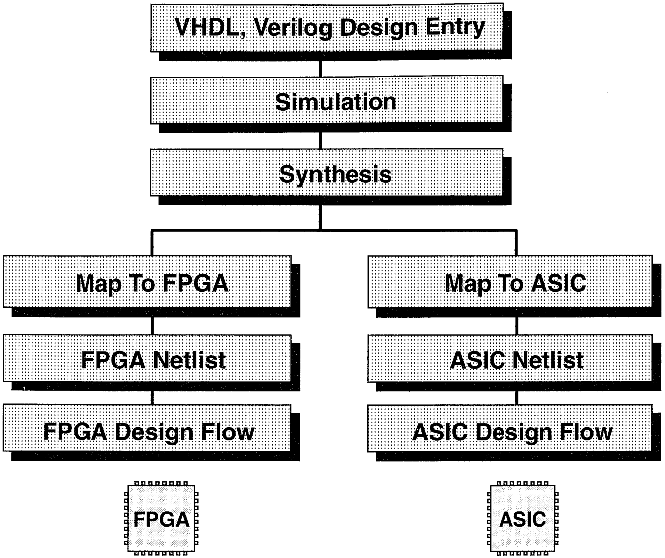
☞ ***Always keep ASIC testability issues in mind***

What is VHDL?

- Very High Speed IC (VHSIC) Hardware Description Language.
- A design language, a simulation language, a documentation language.
- Supports all levels of abstraction: behavioural, RTL, gate, switch.
- Initiated 1985 by Intermetrics, IBM and Texas Instruments.
- Standardized 1987 by IEEE-1076 and revised 1992.
- Widely supported by EDA vendors and ASIC suppliers.
- Strongly typed language with a wide set of constructs.

High Level Migration

High Level Design Migration



VHDL, Verilog Design Entry

Simulation

Synthesis

Map To FPGA

FPGA Netlist

FPGA Design Flow

FPGA

Map To ASIC

ASIC Netlist

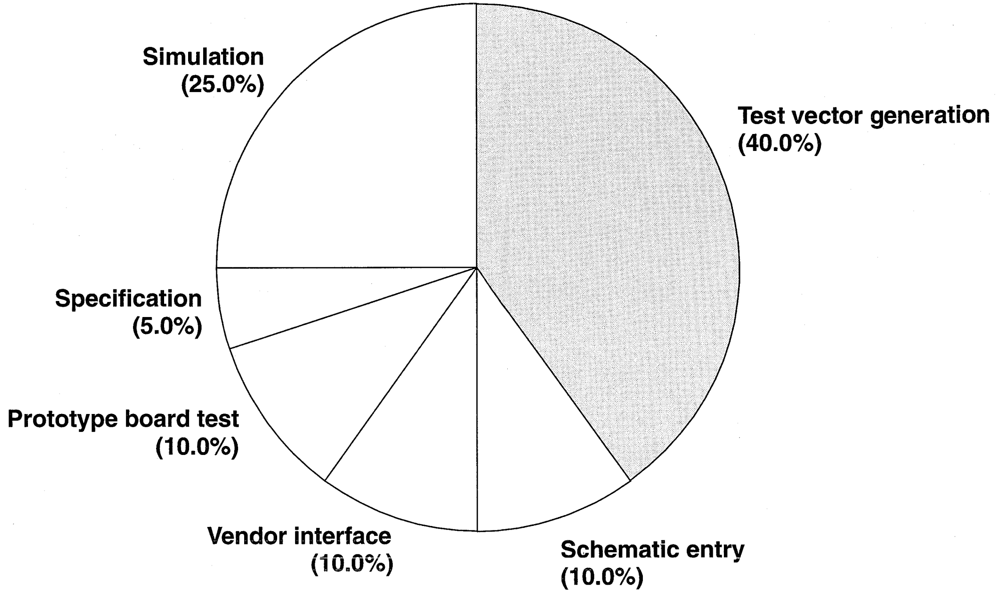
ASIC Design Flow

ASIC

- ***Advantages:***
  - When learned - increased design productivity
  - Fast design modification
  - Synthesis tool optimizes utilization of silicon resources
  - Automatic test pattern generation with test synthesis tool

- ***Disadvantages:***
  - Expensive synthesis tools
  - High level design is new to designers

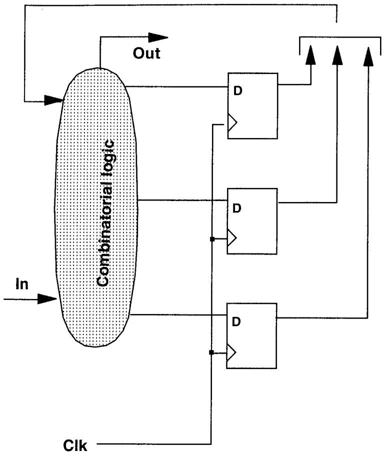
Why Automatic Test Synthesis?



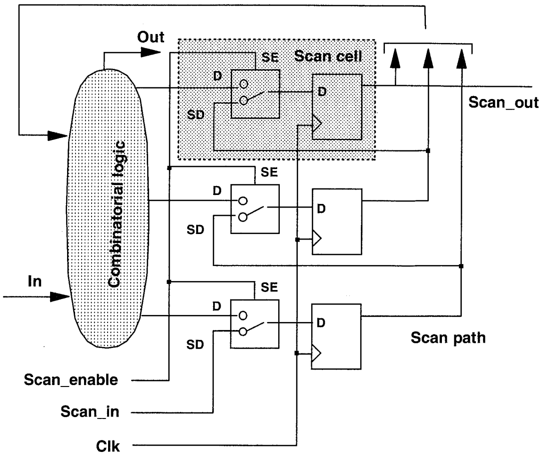
Source: Dataweek

Scan Cell - Scan Path

Synchronous Model



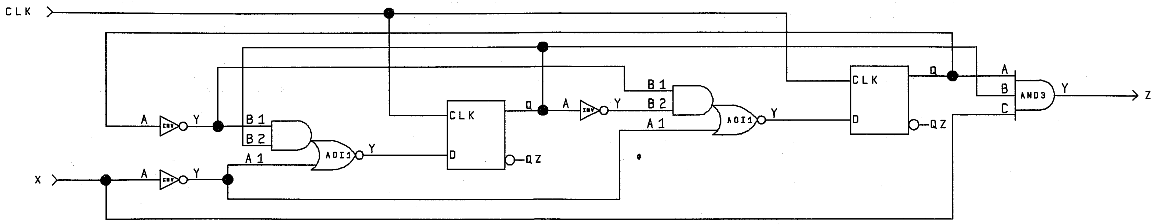
With Scan Path



Scan_enable=H -> Test mode

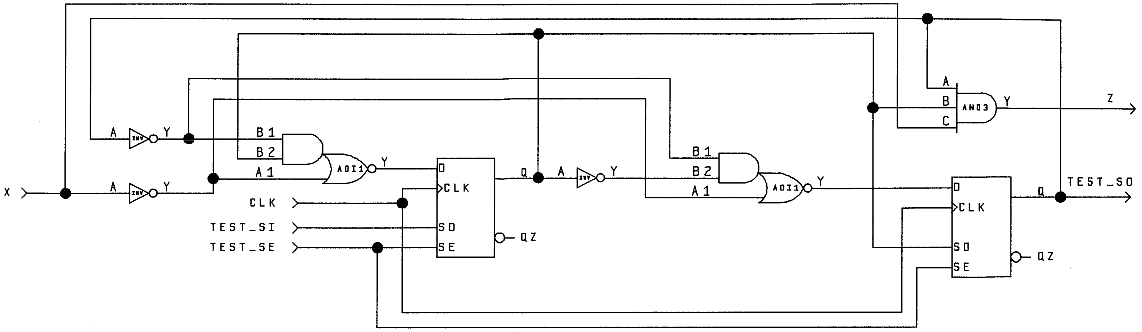
Scan_enable=L -> Normal mode

Circuit Without Test Scan



design:	**MEASEQ**	**designer:**	**Dung Tu**	**date:**	**9/18/91**
technology:	**TGC100.db**	**company:**	**Texas Instruments**	**sheet:**	**1 of 1**

TGC100 Schematic With Test Scan



design: MEASEQ	**designer:** Dung Tu	**date:** 9/18/91
technology: TGC100.db	**company:** Texas Instruments	**sheet:** 1 of 1

ATPG Test Patterns

SIGNAL DEFINITION

1. Signal Names:

test_se	# scan_mode_control port
CLK	# system clock ; active rising-edge
test_si	# scan_in port
test_so	# scan_out port
X	# primary input
Z	# primary output

2. Signal Sets:

Set 1: SS1

test_se	1	I	# scan_mode_control port
CLK	2	I	# system clock ; active rising-edge
test_si	3	I	# scan_in port
test_so	4	O	# scan_out port
X	5	I	# primary input
Z	6	O	# primary output

Set 2: SS2

CLK	1	I	# system clock ; active rising-edge
test_si	2	I	# scan_in port

Set 3: SS3

CLK	1	I	# system clock ; active rising-edge
test_so	2	O	# scan_out port

Set 4: SS4

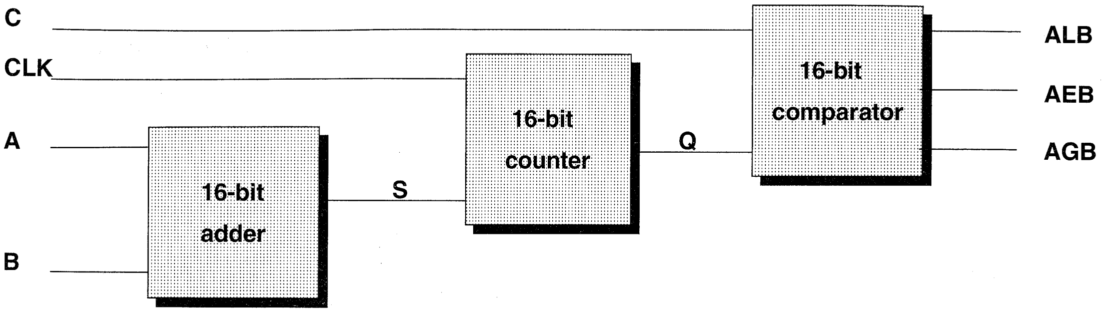
CLK	1	I	# system clock ; active rising-edge
test_si	2	I	# scan_in port
test_so	3	O	# scan_out port

TEST VECTORS

```
0.0 U D U X U X SS1
500.0 U U U X U X SS1
1000.0 D D SS2
1500.0 U D SS2

2000.0 U D N H U X SS1
2500.0 U U N X U X SS1
3000.0 D L SS3


3500.0 U D U X U X SS1
4000.0 U U U X U X SS1
4500.0 D U SS2
5000.0 U U SS2

5500.0 D D U X U X SS1
5600.0 D D U X U H SS1

5700.0 D U U X U X SS1
6200.0 U D U H U X SS1
6700.0 U U U X U X SS1
7200.0 D D H SS4
7700.0 U D X SS4


8200.0 D D D X U X SS1
8300.0 D D D X U L SS1

8400.0 D U D X U X SS1
8900.0 U D D H U X SS1
9400.0 U U D X U X SS1
9900.0 D D H SS4
10400.0 U D X SS4
```

```
10900.0 D D U X U X SS1
11000.0 D D U X U L SS1

11100.0 D U U X U X SS1
11600.0 U D U L U X SS1
12100.0 U U U X U X SS1
12600.0 D U H SS4
13100.0 U U X SS4


13600.0 D D U X D X SS1
13700.0 D D U X D L SS1

13800.0 D U U X D X SS1
14300.0 U D D L D X SS1
14800.0 U U D X D X SS1
15300.0 D U L SS4
15800.0 U U X SS4


16300.0 D D U X U X SS1
16400.0 D D U X U L SS1

16500.0 D U U X U X SS1
17000.0 U D N H U X SS1
17500.0 U U N X U X SS1
18000.0 D L SS3
```

A Case Study



- Two 16-bit numbers A and B are added.
- The sum S is loaded to a preloadable 16-bit counter.
- A 16-bit comparator compares the count Q with a 16-bit number C.
- The outputs are ALB (A<B), AEB (A=B) and AGB (A>B).

16-Bit Comparator VHDL Code

```
entity CMP  is
        port( A, B                      : in INTEGER range 0 to 255;
               ALBI, AEBI, AGBI         : in BOOLEAN;
               ALB, AEB, AGB            : out BOOLEAN
              );
 end CMP;
architecture BEHAVIOR of CMP is
begin
      ALB <= ALBI and (A < B);
      AEB <= AEBI and (A = B);
      AGB <= AGBI and (A > B);
end BEHAVIOR;
```

entity
defined the interface between
a design and its environment

architecture
describes the operation of a design
defines the relationship between
inputs and outputs

16-Bit Counter VHDL Code

```
entity COUNT is
  port( Q      : buffer INTEGER range 0 to 65535;
        P      : in INTEGER range 0 to 65535;
        RCO  : out BIT;
        CLK  : in BIT;
        CLR  : in BIT;
        CI     : in BIT;
        LD     : in BIT );
end COUNT;
```

```
architecture BEHAVIOR of COUNT is
begin
  process ( CLK, CLR, LD, CI, P, Q )
  begin
    if Q = 65535 then
      RCO <= '1';
    else
      RCO <= '0';
    end if;
    if CLR = '0' then
      Q <= 0;
    elsif (CLK'event and CLK = '1') then
      if LD = '0' then
        Q <= P;
      elsif CI = '1' then
        if Q = 65535 then
          Q <= 0;
        else
          Q <= Q + 1;
        end if;
      end if;
    end if;
  end process;
end;
```

16-Bit Adder VHDL Code

```
entity ADD is
 port( A, B  : in INTEGER range 0 to 65535;
     S      : buffer INTEGER range 0 to 65535;
     CO     : out BIT);
end ADD;
architecture  BEHAVIOR of ADD is
begin
 process (A, B, S)
 begin
  S <= A + B;
  if (S >= 65535) then
   CO <= '1';
  else
   CO <= '0';
  end if;
 end process;
end BEHAVIOR;
```

VHDL Design Integration

```
use work.SYNOPSYS.all;
use work.ATTRIBUTES.all;
entity DEMO is
port (  CLK, CLR, LD    : in BIT;
        A, B, C         : in INTEGER range 0 to 65535;
        ALB, AEB, AGB   : out BOOLEAN;
        RCO, CO         : out BIT );
end DEMO;
architecture STRUCTURAL of DEMO is
component ADD
port(  A, B            : in INTEGER range 0 to 65535;
       S           : buffer INTEGER range 0 to 65535;
       CO            : out BIT);
end component;
component CMP
port(  A, B            : in INTEGER range 0 to 65535;
       ALBI, AEBI, AGBI : in BOOLEAN;
       ALB, AEB, AGB   : out BOOLEAN );
end component;
component COUNT
port(  Q             : buffer INTEGER range 0 to 65535;
       P           : in INTEGER range 0 to 65535;
       CLK, CLR, CI, LD : in BIT;
       RCO           : out BIT);
end component;
```

```
signal SI        : INTEGER range 0 to 65535;
signal QI        : INTEGER range 0 to 65535;
signal VDDTIE     : BIT ;
signal VDDTIEBOOL : BOOLEAN ;
signal N1, N2     : BIT;
signal N3, N4, N5 : BOOLEAN;
begin
  VDDTIE <= '1';
  VDDTIEBOOL <= TRUE;
  ADD16 : ADD port map (A, B, SI, N1);
  CNT16 : COUNT port map (QI, SI, CLK, CLR, VDDTIE, LD, N2);
  CMP16 : CMP port map (QI, C, VDDTIEBOOL, VDDTIEBOOL, VDDTIEBOOL, N3,
N4, N5);
  CO <= N1;
  RCO <= N2;
  ALB <= N3;
  AEB <= N4;
  AGB <= N5;
end STRUCTURAL;
```

Recommended Reference Books

VHDL : Douglas Perry
McGraw Hill : $39.95

VHDL : Hardware Description and Design : Roger Lipsett, Carl Schaefer & Carl Ussary
Kluwer Academic Publishers : $59.95

Chip-Level Modelling with VHDL : James Armstrong
Prentice Hall : $39.00

Hardware Design and Simulation : Larry Augustin, David Luckham and Benoit Gennart
Kluwer Academic Publishers : $69.95

The VHDL Handbook : David Coelho
Kluwer Academic Publishers : $70.00

IEEE Standard VHDL Language Reference Manual

Summary

- VHDL is becoming a useful tool for hardware design and simulation.
- VHDL synthesis tools for FPGAs are now available for PC as well as workstations.
- Designing with high level language enables a seamless migration from one technology to another.
- High level languages increase productivity and design quality.
- TI is one major supplier for both FPGAs and ASICs.

Agenda

- *PLD/FPGA Architecture Analysis*
- *How To Replace TTL With FPGA*
- *How To Replace PLD With FPGA*
- *Applications Benchmarks*
- *FPGA Design With VHDL*
- ☞ *FPGA Products And Development Systems*

TPC10 Devices

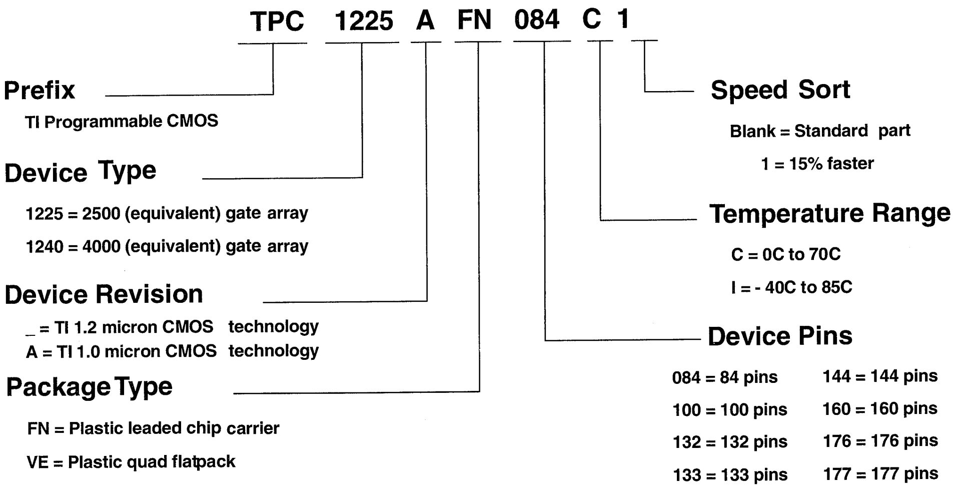
Device	TPC1010A	TPC1020A	TPC1010B	TPC1020B
Gate array equivalent gates	1200	2000	1200	2000
TTL equivalent packages	34	53	34	53
CMOS Process	1.2um	1.2um	1.0um	1.0um
Logic Modules	295	547	295	547
Flip-Flops (maximum)	130	273	130	273
Antifuse	112k	186k	112k	186k
User I/Os (maximum)	57	69	57	69
Speed grade -1 : 15 %faster -2 : 25% faster	-1	-1	-1 -2	-1 -2
Packages	44 PLCC 68 PLCC 100 PQFP	44 PLCC 68 PLCC 84 PLCC 100 PQFP	44 PLCC 68 PLCC 100 PQFP	44 PLCC 68 PLCC 84 PLCC 100 PQFP

TPC12 Devices

Device	TPC1225A	TPC1240	TPC1240A	TPC1280	TPC1280A
Gate array equivalent gates TTL equivalent packages	2500 70	4000 105	4000 105	8000 210	8000 210
CMOS Process	1.0um	1.2um	1.0um	1.2um	1.0um
Logic Modules	451	684	684	1232	1232
Flip-Flops (maximum)	382	565	565	998	998
Antifuse	250k	400k	400K	750k	750k
Signal I/Os (maximum)	82	104	104	140	140
Speed grade -1 : 15% faster -2 : 25% faster	-1	-1	-1 -2	-1	-1 -2
Packages	84 PLCC 100 PQFP	132 CPGA 144 PQFP	84 PLCC 132 CPGA 144 PQFP	176 CPGA 160 PQFP	176 CPGA 160 PQFP

TPC Product Nomenclature

Commercial and Industrial



TPC 1225 A FN 084 C 1

Prefix

TI Programmable CMOS

Device Type

1225 = 2500 (equivalent) gate array

1240 = 4000 (equivalent) gate array

Device Revision

_ = TI 1.2 micron CMOS technology
A = TI 1.0 micron CMOS technology

Package Type

FN = Plastic leaded chip carrier

VE = Plastic quad flatpack

Speed Sort

Blank = Standard part

1 = 15% faster

Temperature Range

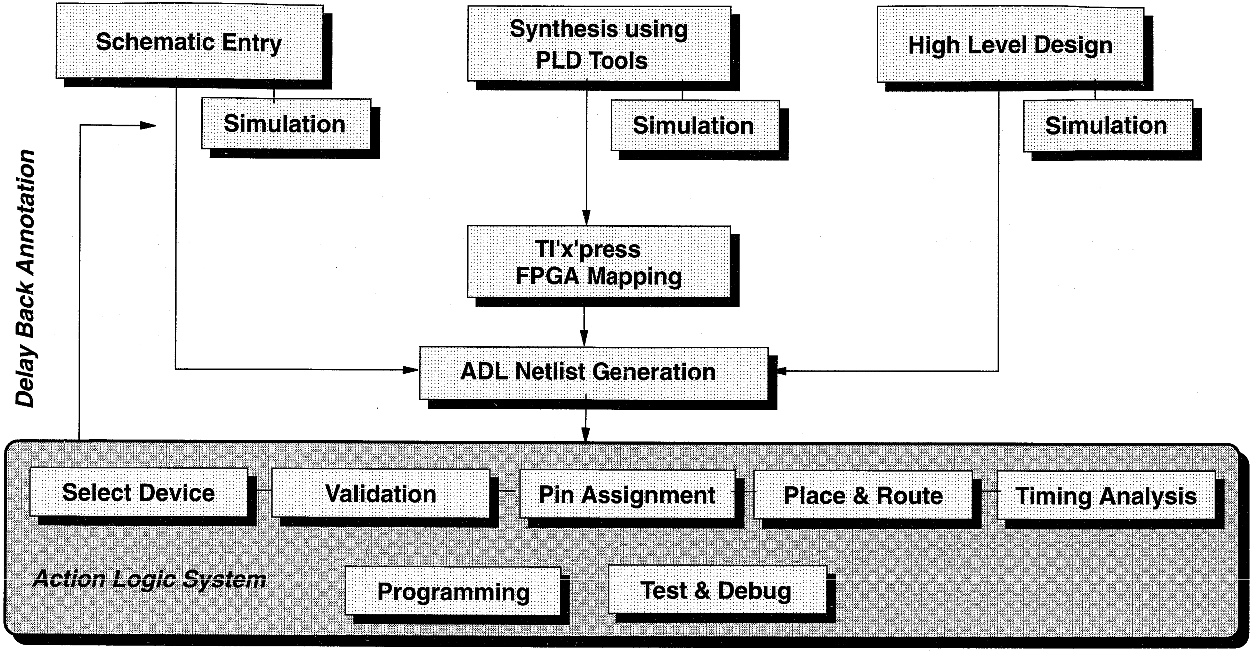
C = 0C to 70C

I = - 40C to 85C

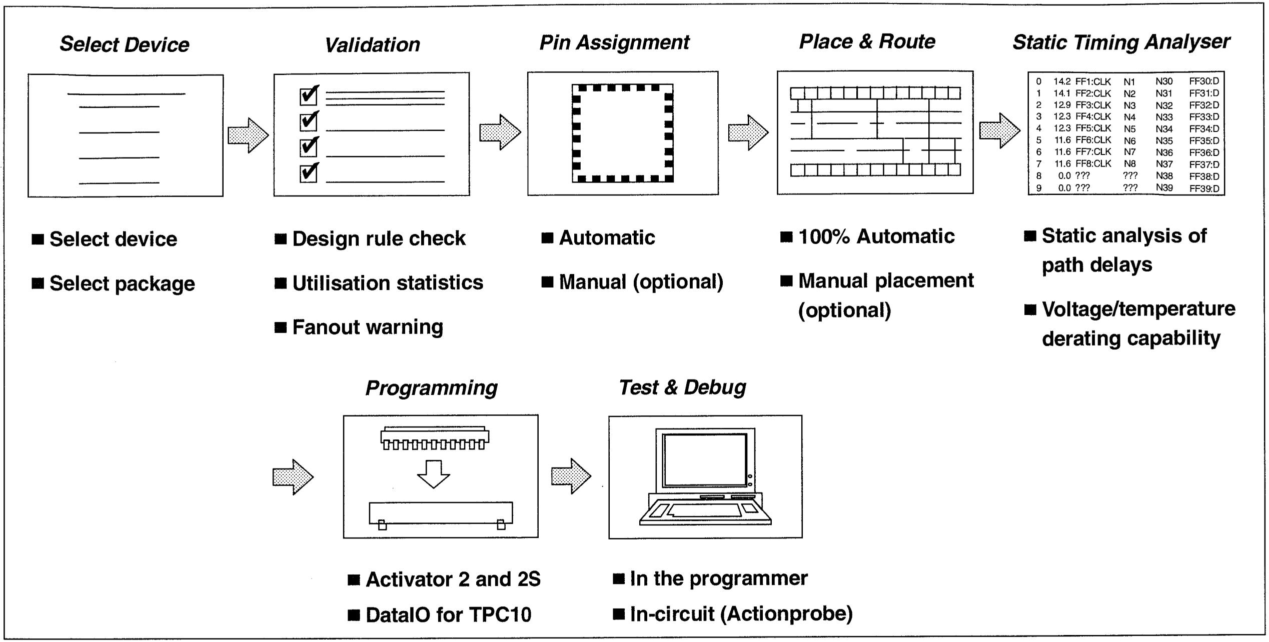
Device Pins

084 = 84 pins	144 = 144 pins
100 = 100 pins	160 = 160 pins
132 = 132 pins	176 = 176 pins
133 = 133 pins	177 = 177 pins

FPGA Design Flows



Action Logic System



Select Device

- Select device
- Select package

Validation

- Design rule check
- Utilisation statistics
- Fanout warning

Pin Assignment

- Automatic
- Manual (optional)

Place & Route

- 100% Automatic
- Manual placement (optional)

Static Timing Analyser

```
0   14.2 FF1:CLK   N1    N30   FF30:D
1   14.1 FF2:CLK   N2    N31   FF31:D
2   12.9 FF3:CLK   N3    N32   FF32:D
3   12.3 FF4:CLK   N4    N33   FF33:D
4   12.3 FF5:CLK   N5    N34   FF34:D
5   11.6 FF6:CLK   N6    N35   FF35:D
6   11.6 FF7:CLK   N7    N36   FF36:D
7   11.6 FF8:CLK   N8    N37   FF37:D
8    0.0 ???       ???   N38   FF38:D
9    0.0 ???       ???   N39   FF39:D
```

- Static analysis of path delays
- Voltage/temperature derating capability

Programming

- Activator 2 and 2S
- DataIO for TPC10

Test & Debug

- In the programmer
- In-circuit (Actionprobe)

TI-ALS System Options

TPC- ALS-DS Series : for devices up to 2500 gates

TPC-ALS-DA Series : for devices greater than 2500 gates

TI-ALS CAE Platforms

Hardware Platform	Library/CAE Host Environment	Max Density 2500 gates	Max Density 10000 gates	Part Number
PC	Viewlogic	X		TPC-ALS-DS-PC-VL
PC	Viewlogic		X	TPC-ALS-DA-PC-VL
PC	Orcad	X		TPC-ALS-DS-PC-OR
PC	Orcad		X	TPC-ALS-DA-PC-OR
Sun	Cadence		X	TPC-ALS-DA-SN-CD
Sun	Mentor 8.2		X	TPC-ALS-DA-SN-MG
Sun	Valid		X	TPC-ALS-245
Sun	Viewlogic		X	TPC-ALS-DA-SN-VL
HP700	Mentor 8.2		X	TPC-ALS-DA-HP7-MG
DN4000/HP400	Mentor 7.0		X	TPC-ALS-235

***Note:** To get the Part Number for the annual maintenance, please add a -SP suffix to the Part Number of the corresponding software packages*

TI-ALS Programming Configurations

Hardware Platform	CAE Host Environment	Programmer for One Device	Programmer for Four Devices	Part Number
PC	Viewlogic/Orcad	X		TPC-ALS-DS-P2S-PC
PC	Viewlogic/Orcad		X	TPC-ALS-219
Sun	Cadence/Mentor/ Valid/Viewlogic	X		TPC-ALS-DS-P2S-SN
Sun	Cadence/Mentor/ Valid/Viewlogic		X	TPC-ALS-249
HP700	Mentor	X		TPC-ALS-DS-P2S-HP7
HP700	Mentor		X	TPC-ALS-DS-P2-HP7
HP400	Mentor	X		TPC-ALS-DS-P2S-HP4
HP400	Mentor		X	TPC-ALS-DS-P2-HP4
DN Series	Mentor		X	TPC-ALS-239

Note: Programming units are compatible with both high (10000 gates) and low (2500 gates) density systems

Software Options

Hardware Platform	Library/CAE Environmeent	Description	Part Numnber
PC	Viewlogic	High Density Viewsim Simulator	TPC-ALS-016
PC	Viewlogic	Low Density Viewsim Simulator	TPC-ALS-017
PC	Viewlogic	Schematic Capture, Viewdraw	TPC-ALS-VL-005
PC	Viewlogic	Schematic Generation, Viewgen	TPC-ALS-VL-001
PC	TI'x'press	Boolean Synthesis and FPGA Optimization	TPC-ALS-PLDSYN
PC	TI	Universal Actionprobe for Test & Debug	TPC-ALS-218
Sun	Synopsys	TI-Synopsys Libraries fro Design Compiler	TPC-ALS-SYN-S4
Sun	TI	Universal Actionprobe for Test & Debug	TPC-ALS-218
HP700	Synopsys	TI-Synopsys Libraries fro Design Compiler	TPC-ALS-SYN-HP
HP700	TI	Universal Actionprobe for Test & Debug	TPC-ALS-218
DN4000/HP400	Synopsys	TI-Synopsys Libraries fro Design Compiler	TPC-ALS-SYN-DN
DN4000/HP400	TI	Universal Actionprobe for Test & Debug	TPC-ALS-218

TI FPGA Integrator Series

- ✓ Schematic Capture - Viewlogic Viewdraw
- ✓ Simulation - Viewlogic Viewsim
- ✓ PLD Equation Entry - TI Prologic
- ✓ Logic Synthesis - TI'x'press
- ✓ Device Implementation - TI-ALS-DS
- ✓ Floorplanning - TI Module Mover
- ✓ Logic Technology Selector - TI Logic Integration Software
- ✓ Programmer - Activator 2S

TI FPGA Integrator Series

	FPGA DELUXE KIT	PLD PLUS KIT	PLD KIT	TI FPGA STARTER KIT
Schematic Capture	✓			
Simulation	✓			
PLD Equation Entry	✓	✓	✓	
Logic Synthesis	✓	✓	✓	
Device Implementation: - Place & Route - Viewlogic or OrCAD Libraries - Timing Analysis - Programming Software	✓	✓	✓	✓
Floorplanning	✓	✓	✓	✓
Logic Selector Software	✓	✓	✓	✓
Single Socket Programmer	✓	✓		
$1000 Credit Towards Exemplar VHDL Upgrade	✓	✓	✓	
1 Year Maintenance	✓	✓	✓	✓
	$4,495	$1,995	$1,295	$ 695

TI FPGA Integrator Series

FPGA Deluxe Kit	PLD Plus Kit	PLD Kit	FPGA Starter Kit
TPC-ALS-INT-FPGA-DX	TPC-ALS-INT-PLDP-VL	TPC-ALS-INT-PLD-VL	TPC-ALS-INT-START-VL
Viewlogic Schematic Capture (TPC-ALS-VL-005)			
Viewlogic 3k Gates Simulator (TPC-ALS-017)			
Prologic - PLD Equation Entry	Prologic - PLD Equation Entry	Prologic - PLD Equation Entry	
TI'x'PRESS - Logic Synthesis (TPC-ALS-PLDSYN)	TI'x'PRESS - Logic Synthesis (TPC-ALS-PLDSYN)	TI'x'PRESS - Logic Synthesis (TPC-ALS-PLDSYN)	
TI-ALS with Viewlogic Libraries (TPC-ALS-DS-PC-VL)	TI-ALS with Viewlogic Libraries (TPC-ALS-DS-PC-VL)	TI-ALS with Viewlogic Libraries (TPC-ALS-DS-PC-VL)	TI-ALS with Viewlogic Libraries (TPC-ALS-DS-PC-VL)
TI Module Mover	TI Module Mover	TI Module Mover	TI Module Mover
TI Logic Integration Software	TI Logic Integration Software	TI Logic Integration Software	TI Logic Integration Software
Programmer Activator 2S (TPC-ALS-DS-P2S-PC)	Programmer Activator 2S (TPC-ALS-DS-P2S-PC)		
$ 1000 Credit towards Exemplar VHDL Upgrade	$ 1000 Credit towards Exemplar VHDL Upgrade	$ 1000 Credit towards Exemplar VHDL Upgrade	
1 Year Maintenance	1 Year Maintenance	1 Year Maintenance	1 Year Maintenance
$ 4,495	$ 1,995	$ 1,295	$ 695

TEXAS INSTRUMENTS