

ALTERA

ALTERA

ALTERA

ALTERA

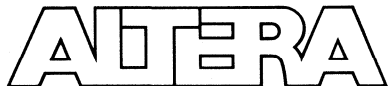
# EPLD HANDBOOK

Copyright © 1984, 1985 ALTERA Corporation.

#### PATENT PENDING

The following are trademarks of Altera Corporation: EP300, EP310, EP600, EP900, EP1200, EP1210, EP1800, PLDS2, PLS2, PLCAD, PLE. ASAP is a trademark of Altera Corporation. TURBO-BIT is a trademark of Altera Corporation. PAL is a trademark of Monolithic Memories Inc. CROSSTALK is a trademark of Microstuf Inc. PC-CAPS is a trademark of Personal CAD Systems Inc. Dash-2 is a trademark of FutureNet Corporation. MS-DOS is a trademark of MicroSoft Corporation. WordStar is a trademark of MicroPro Corporation. ABEL is a trademark of Data I/O Corporation. CUPL is a trademark of Assisted Technology Inc. A+PLUS design elements and Mnemonics are Altera Corporation copyright. CHMOS is a trademark of Intel Corporation, IBM is a registered trademark of International Business Machines, Inc., A+PLUS, NetMap and LogicMap are trademarks of Altera Corporation. ALTERA does not assume any liability arising out of the application or use of any product described herein; neither does it convey any license under its patent rights nor the rights of others. ALTERA reserves the right to make changes at any time in order to improve reliability, function or design and to supply the best product possible.

ALTERA cannot assume any responsibility for any circuits shown or represent that they are free from patent infringement.



**ALTERA CORPORATION**  
**3525 MONROE STREET,**  
**SANTA CLARA, CA 95051**  
**(408) 984-2800**

# *the Logical Alternative*

## **HOW TO USE THIS HANDBOOK**

This handbook is organized in sections progressing from overview to detail. However, it is not necessary that it be used in this manner. If you are already familiar with Altera EPLDs and the associated design support versus other semicustom logic alternatives, you can go right to the product data in Section 2. On the other hand, if you're venturing into EPLDs or semicustom logic for the first time, you may choose to acquaint yourself with Section 1 which includes some brief history and a discussion of the technology issues and alternatives in the application specific market. Application information is contained in Section 3.

To place an order, go directly to the Appendices in Section 4 for ordering information, package outlines, or distributor locations.

If this Handbook doesn't answer your questions, please call on our toll-free numbers, 1-800-821-8124 (outside California) or 1-800-654-4236 (inside California), and we will help you directly.



# ALTERA

# ALTERA

# ALTERA

# ALTERA

---

## **EPLD: Erasable Programmable Logic Devices**

**Page 5**

---

■ INTRODUCTION TO ALTERA	7
■ THE EPLD CONCEPT	8
■ CMOS EPROM TECHNOLOGY	10
■ DEVELOPMENT SYSTEMS & SOFTWARE SUPPORT	12

---

## **DATA SHEETS**

**Page 17**

---

■ PRODUCT LIST	18
■ SELECTION GUIDE	19
■ DATA SHEETS	20

---

## **APPLICATION INFORMATION**

**Page 83**

---

■ APPLICATION NOTES	84
■ APPLICATION BRIEFS	148

---

## **APPENDICES**

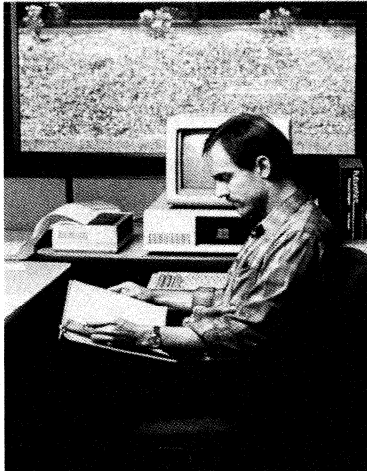
**Page 227**

---

■ GLOSSARY	229
■ TECHNICAL REFERENCES	233
■ PRODUCT COMPATIBILITY	233
■ PROGRAM SUPPORT	234
■ ORDERING INFORMATION	235
■ PACKAGING INFORMATION	236
■ QUALITY INFORMATION	238
■ ALTERA DISTRIBUTORS	241
■ ALTERA SALES REPRESENTATIVES	244



# Introduction to Altera



## ALTERA CORPORATION

### BACKGROUND

Altera was founded in June of 1983 on the premise that the broad based market of semiconductor logic users needed a new alternative for large scale custom logic. The founders of Altera believed that the problems of development cost, lead time, and flexibility associated with traditional vendor mask-customized solutions could be solved through the use of programmable standard components. This has led to the development and marketing of Altera's family of CMOS EPLD products. By combining CMOS and erasable technology with advanced design tools, Altera's EPLDs are targeted to replace TTL/74HC/CMOS 4000 random logic and low density gate arrays, reducing system cost, product size, and power requirements.

### PRODUCT LINES

Altera products fall into three groups:

1. Erasable Programmable Logic Devices (EPLDs)
2. Programmable Logic Development Systems
3. Software Development Support Tools

## EPLD COMPONENTS

The EPLD family of components includes five initial devices that range in density from 300 to 1800 gates and with performance from 25 to 90 ns. All use low power CHMOS erasable technology and include between 8 and 48 flipflops. EPLDs have general system applicability by allowing the replacement of many low density logic functions with single packages. This saves physical space while at the same time reduces power requirements and system failure rates.

## DEVELOPMENT SYSTEMS

Altera Development System products are self contained IBM PC compatible development systems for designing with and programming Altera EPLD components. Each development system contains everything the first time user needs to make an IBM PC (or compatible) be a logic development system. Each system comes with programming hardware, fully documented software, and sample devices to allow the new user to design quickly and easily.

## SOFTWARE SUPPORT TOOLS

Software support tool products include A+PLUS (Altera's proprietary automatic logic compiler), schematic symbol libraries, a logic simulator, interfaced schematic capture packages from FutureNet and P-CAD (Altera is a value-added remarketer) and other development tools. This group of support tools is a growing family of products designed to make the designer's task quicker and easier to perform.

Altera's products are sold by manufacturers representatives and major electronic distributors throughout the world. Intel Corporation is a second source for many of the Altera products.

## PHILOSOPHY AND DIRECTION

Altera believes that advanced development tools are a critical factor for the advancement of programmable semiconductor technology. In order that component utility and ultimately designer efficiency be maximized, the semiconductor and the software must be developed in parallel and concert. If one is subordinated to the other, the total product combination will suffer in performance and utility. In line with this belief, Altera has invested in approximately equal sized R&D staffs in the areas of Software Development and Integrated Circuit Design.

One clear benefit of this philosophy to the user is that of immediate support for new products. When Altera introduces a new product to the market place,

there will always be software and hardware support in place the day of introduction. This level of support and service is unique in the industry and is an integral part of Altera's philosophy.

Altera, as a Company, is focused entirely on the application of CMOS erasable technology to the user programmable logic market. Altera's goal is to serve its customers with superior quality in both products and services. By focusing exclusively on this market area, Altera expects to be able to provide a higher level of technical support as well as demonstrate a greater rate of technical innovation. It is the Company's intent to be recognized as the technology and market leader in this product area.

## The EPLD Concept

### THE EPLD CONCEPT

Logic is the key proprietary ingredient in electronic system design. The ability to rapidly complete the logic engineering of a system, and quickly and cost effectively bring the product to market, is critical. This can be a deciding factor in the ultimate success or failure of a product and a company. To achieve performance in the marketplace, more and more manufacturers have sought higher levels of integration (functional density) for the electronic components in their products. This has led to various forms of custom chips that require lengthy development lead times and sizable design costs. The concept of EPLDs is to provide the benefits of large scale integration *without* the drawbacks of custom chips.

### WHY INTEGRATE?

The benefits of large scale integration are becoming more widely known. A few of the most significant benefits are:

**LOWER MANUFACTURING COSTS:** The use of customized LSI circuits instead of standard SSI/MSI (small and medium scale) components reduces required printed circuit board space, thereby significantly reducing board costs. Through integration, savings can also be realized in the individual costs of the passive components, reduced assembly and inventory costs.

**LOWER POWER:** In this age of need for energy efficiency, integration can reduce the power consumption of systems by orders of magnitude. Additional benefits are less expensive power supplies and elimination of cooling fans.

**HIGHER RELIABILITY:** LSI circuits have been shown to have statistically higher reliability than equivalent systems comprised of many low density standard

components. When an LSI circuit implementation is replacing electro-mechanical functions, system lifetimes will be greatly extended since there are no moving parts to wear out. Required maintenance is lower and system reliability is higher.

### PROPRIETARY DESIGN PROTECTION:

An LSI circuit design solution creates barriers to market entry for competitors. It makes "reverse engineering" of your design significantly more difficult for potential imitators.

**ADDITIONAL FEATURES:** At the same time the benefits of the previous items are occurring, LSI circuit implementation often allows the incorporation of optional and special features at a very low incremental cost.

### ALTERNATIVES FOR ACHIEVING

### LARGE SCALE LOGIC INTEGRATION

As end users of semiconductors were compelled to move to higher and higher levels of integration for the reasons just discussed, chip designers found it increasingly difficult to define larger and larger common "building blocks" of logic. These difficulties led to the emergence of the *user-defined* Application-Specific Integrated Circuit (ASIC). Over the past five years, this has been one of the fastest growing segments of the semiconductor business, indicating increasing use of customized LSI in end products.

The options presently available for application-specific logic are:

- Full Custom
- Standard Cell Library
- Gate Array
- User Programmable

Each of these options will be briefly described along with the benefits and tradeoffs.

**FULL CUSTOM:** The full custom alternative supplies the highest efficiency in chip area and performance. These circuits, typically handcrafted, can be tailored to give the best functional performance with the highest level of integration, the smallest silicon area, use the lowest power, and be produced for the least cost at high production volumes.

Offsetting these optimal characteristics, however, are long development lead times (long time to market: 1 to 3 years), and large development costs (\$50,000-\$250,000). These drawbacks have led to the popularity of Standard Cell Library and Gate Array custom semiconductors.

**STANDARD CELL LIBRARY:** The standard cell library approach represents an integrated circuit which is composed of predesigned and precharacterized cells chosen from a computer "data base" library of cells. A specific function is realized by choosing, placing, and interconnecting these cells.



Some of the specific advantages and disadvantages of a standard cell library approach are as follows. Advantages are high integration capability with gate densities up to 20,000 gates; medium design time (3-6 months); and low production cost when produced in significant volumes. Some of the disadvantages of cell libraries are that all mask layers are customized; the technology is the highest cost form of semicustom design, and results in a non-standardized pin out, requiring unique packaging, bonding, and test hardware for each circuit. Also, no second sourcing is presently available, and the process technology typically lags the state-of-the-art due to library maintenance and characterization required for each wafer fabrication process upgrade.

**GATE ARRAYS:** Gate arrays are integrated circuits that contain a regular, usually square, matrix of predefined logic gates. Custom functions are realized by means of a unique customized interconnection of the gates and the input and output structures. The custom interconnection varies from one layer metal to three layer metal, with a multitude of techniques used to design the connections.

The advantages of gate arrays are that they are simpler and cheaper to design than full custom or cell library circuits, and can be produced with less lead time. The development costs and lead times remain significant, however.

A comparison of these mask customized alternatives is shown below:

ALTERNATIVE	% OF WAFER PRE-PROCESSED	DEVELOPMENT COST/CHIP (K-\$)	PROTOTYPE LEADTIME (MONTHS)
FULL CUSTOM	0%	\$50-250	9-18
STANDARD CELL	0%	\$30-90	3-6
GATE ARRAY	80-90%	\$10-40	1½-5

None of these three mask customized solutions have been wholly satisfactory for system designers and manufacturers due to several problems:

Development lead times are relatively long, requiring from 6 to 20 weeks for the fastest solution.

Design costs are significant, varying from \$10K to \$40K as a minimum.

Inventory is dedicated which is expensive and prohibits adequate second sourcing.

Semiconductor distributors have difficulty participating in this business — thus limiting widespread use.

None of the solutions mentioned address the fundamental issue that engineering is inherently an interactive process. Design changes in midstream are not allowed due to lead time and inventory constraints.

As a result of these restrictions, many designers are still reluctant to switch from TTL standard logic to application-specific logic.

**USER PROGRAMMABLE LOGIC:** Attempts to eliminate these restrictions have led to an increasing interest in programmable logic devices. The concept of user programmable logic is to provide the designer with the benefits of custom LSI chips from standard products. The benefits of such parts include off-the-shelf availability, minimal design costs, multiple sourcing from distributors and manufacturers, and flexible interchangeable inventory.

## BIPOLAR FUSE TECHNOLOGY

In the past all programmable logic products were implemented using bipolar fuse technology. These products eliminated the lead time and development cost penalties of the mask customized solutions previously mentioned, but brought with them their own inherent limitations:

Bipolar, with its high power dissipation, cannot provide the integration density required.

Fuse programming does not allow complete testing at the factory and is inefficient in silicon utilization.

The devices can only be programmed once; therefore, mistakes in development result in scrap, a significant penalty with high density parts.

The programming software and development tools are primitive and tedious to use.

## CMOS ERASABLE TECHNOLOGY

Altera was the first supplier to overcome the problems of programmable logic when it introduced its EPLD line of user-programmable logic devices incorporating CMOS floating-gate technology.

Altera EPLDs are manufactured with high-speed complementary metal oxide semiconductor (CHMOS\*) technology. Compared to bipolar fuse technology, CMOS provides lower power dissipation and a cooler operating temperature which enables designers to pack a greater number of logic functions onto a chip. The EPLD family, introduced in July 1984, provides low power consumption, integration densities from 300 to 1800 gates, and full testability.

Altera's EPLDs use an EPROM programming mechanism. This technology, used in MOS memories since the early 1970's, brings further advantages. It enables the devices to be reprogrammed in the event of any design changes. The fact that programming can be erased permits thorough testing during the manufacturing process and flexibility in the hands of the user. Overall, it puts a much greater degree of control in the hands of the system designer.

\* Trademark of Intel Corporation

EPLDs resolve the earlier limitations of user-programmable logic *and* the limitations of mask customized logic. Large scale custom logic can now be created without a long development cycle or significant cost of design. Minor design changes or adjustments can be made quickly and efficiently without inventory jeopardy. These conclusions are demonstrated in the comparison below.

ALTERNATIVE	% OF WAFER PRE-PROCESSED	DEVELOPMENT COST/CHIP (K-\$)	PROTOTYPE LEADTIME (MONTHS)
FULL CUSTOM	0%	\$50-250	9-18
STANDARD CELL	0%	\$30-90	3-6
GATE ARRAY	80-90%	\$10-40	1½-5
USER PROGRAMMABLE EPLDs	100%	0	OFF THE SHELF

## EASY TO USE

For user-programmable circuits to reach the broad base of existing SSI/MSI TTL users, the programming and design tools must meet three criteria: (1) be low cost, (2) easy to learn and use, and (3) have personal availability and access. Today, the most widely available source of computing power is the personal computer. By creating development tools that fit the personal computer environment, all three of these criteria can be met.

## SUMMARY

Altera, with its EPLD products and development system support tools, has addressed the limitations of gate arrays and fuse programmable logic. The benefits to the system designer are:

- no lead times
- low design costs
- multiple sourcing from distributors and manufacturers
- ease of design changes
- multiple programming, if necessary
- low power dissipation from CMOS technology
- high density products that maximize function, integration, and quality
- maximum flexibility in each chip, that comes from programmable architecture, and the ability to erase and reprogram
- a self contained low-cost sophisticated development system.

EPLDs are now a cost-effective solution to the problem of large scale logic integration. EPLDs are the simplest form of high density application-specific logic to implement. As such, they will be a key ingredient to boosting electronic engineering productivity over the next decade.

# CMOS EPROM Technology

## CMOS EPROM TECHNOLOGY

To achieve the goals established for EPLD products, Altera performed a thorough evaluation of semiconductor technology. This resulted in the selection of CMOS and floating gate (EPROM) technologies. CMOS was chosen for its density, reliability, testability, and low power. Each of these technologies are described in more detail below.

## WHAT IS CMOS?

Complementary metal oxide semiconductor (CMOS) technology simply involves the fabrication of n and p channel MOS transistors on the same substrate and is analogous to using both npn and pnp transistors in a bipolar circuit. CMOS typically uses only enhancement-mode transistors, in contrast to NMOS which may employ both enhancement and depletion device types.

## LOW POWER

The use of enhancement-only device types give CMOS its automatic power-down capability since use of both n and p channel MOSFETs in the basic inverter gate means one of the transistors is always off. Thus, there is no current path from power supply to ground and both active pull up and pull down paths exist for the load. This first feature gives CMOS its extremely low power dissipation in the standby mode (only leakage current flow), and the second means that CMOS devices can be very fast — faster than NMOS. (CMOS rise and fall times are short and symmetrical, while NMOS has a short fall time but a significantly longer rise time because of its high resistance load, dictated by D.C. pull up/pull down considerations.)

## NOISE IMMUNITY

The simplicity of the inverter configuration and the transistor structure gives CMOS a wide margin of power supply. One outcome of the resulting increased logic swing is that it gives the technology excellent noise immunity — typically 45% of  $V_{CC}$ , which is better than that of either TTL or NMOS.

## RELIABILITY

CMOS device junctions average about 20°C cooler than their NMOS counterparts. This gives CMOS up to more than 10 times the reliability edge with respect to failure rate. Additionally, high temperature operation (greater than 150°C) is possible with CMOS for short times because of this.

Altera manufactures its EPLDs using CHMOS\* n-well technology. This high speed CMOS technology provides lower power dissipation and a cooler operating temperature, enabling designers to pack a greater number of logic functions onto a chip. Altera's EPLDs use an EPROM-type programming mechanism. This technology, used in MOS memories since the mid-1970's, brings further advantages.

## EPROM (FLOATING GATE TECHNOLOGY)

Altera uses EPROM transistors in its EPLDs to perform all of the various customizing functions. By using the EPROM transistors as switches throughout the circuits, both architecture and individual connections can be configured.

## HOW EPROMs WORK

The EPROM transistor is built with the same structure as a standard NMOS silicon gate transistor, with the addition of a second polysilicon gate electrode that is electrically isolated from all other elements of the circuit by means of oxide regions which surround it on all sides.

In the unprogrammed state the floating gate is uncharged and does not influence transistor action. The transistor acts like a normal transistor whose gate oxide thickness is the sum of the thicknesses of the gate

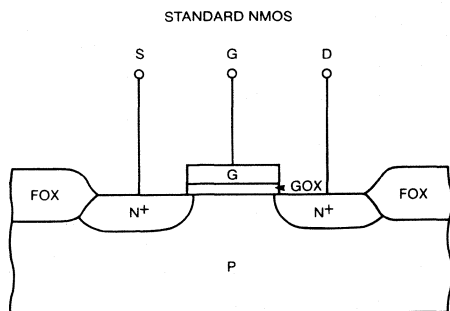
oxide and the interpoly oxide. In this condition, the transistor is turned on if  $V_{CC}$  is applied to the gate electrode.

To program the transistor, high voltages are applied to the gate and drain electrodes. This turns the transistor on very hard, with the result that energetic electrons are able to surmount the potential barrier and move through the gate oxide to the floating gate. There they accumulate until the effect saturates. Upon removal of the applied voltages, there is a net negative change left on the floating gate.

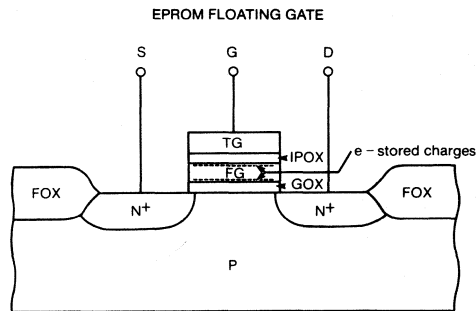
The effect of the stored charge is to increase the threshold (turn-on voltage of the transistor), such that when  $V_{CC}$  is applied to the gate, the transistor remains off. This charge is very stable and under normal operating conditions it will not dissipate for many years.

## ADVANTAGES FOR LOGIC USERS

The advantages that CMOS and EPROM give the user of EPLDs are numerous. For instance, the lower power consumption makes battery and non-cooled applications possible. It also allows the production of higher density logic devices since the devices do not reach power dissipation limits. EPROM transistors, as well as being programmable, are also erasable. This permits users to re-use and re-program devices ultimately saving money. The erasability saves in other ways too: because Altera can program all transistors and test them fully before shipment, 100% programming yield is assured to users. This was previously unknown with bipolar devices. Finally, EPLDs are secure. The logic that is programmed cannot be seen even if the chip is opened since the stored charges are invisible and captured on a buried layer of polysilicon. These are just some of the advantages that EPLDs offer by using the combination of CMOS and EPROM floating gate technologies.



TG = TOP POLY-SILICON GATE  
IPOX = INTER-POLY OXIDE  
FG = FLOATING POLY-SILICON GATE  
GOX = GATE OXIDE  
FOX = ISOLATION (FIELD) OXIDE



$N^+$  =  $N^+$  SOURCE AND DRAIN  
P = P-TYPE SUBSTRATE  
S = SOURCE  
G = GATE  
D = DRAIN

# Development Systems and Software Support

## INTRODUCTION

Programmable Logic Devices (PLDs) have seen a rapid, even dramatic rise in their market over the past few years. This has been due to a combination of factors acting favorably together: availability of a greater variety of devices, improved support tools, multiple sources, and increased design sophistication in the marketplace. The net result has been the emergence of a new technology which will see universal application over the next few years.

The bulk of this phenomenal growth has been in Field Programmable Logic Devices (FPLDs). Programmable by the user via a variety of technologies, instant turnaround has proven an invaluable aid in reducing time to market for the end-user.

Occupying an expanding niche between traditional TTL and LSI, FPLD capabilities have increased as older device families have been refined and new device concepts have been introduced. Lower power, higher performance, more flexibility are being built into new devices. The system designer, receptive to and educated in the use of such devices, reaps these benefits.

The challenge for those developing FPLD support tools and systems is to simplify the education process and provide an efficient translation of abstract design ideas into device programs. With the increased complexity of FPLDs, the user requires assistance at all stages of the design process from design conception through to test generation. Simple coding assemblers are no longer enough in the FPLD arena just as they have fallen out of favor for the bulk of microprocessor programming applications. As FPLDs push into the domain previously reserved for LSI and large-scale gate arrays, the demand for higher-level support tools, both hardware and software, will increase. Given the complexity of the design process and the proliferation of devices and features available, this is no small task.

EPLDs (Erasable Programmable Logic Devices) emerged on the scene in 1984 with the introduction of Altera's EP300 component. Utilizing CMOS EPROM technology to retain programming information and a flexible Macrocell architecture allows one device to replace all members of earlier 20 pin PAL families. Low

CMOS power is an additional attraction for designers, as well as the reprogrammable nature of the EPROM technology.

Soon after, Altera introduced the EP1200/EP1210 equipped with 28 flip-flops. The EP1200/EP1210 represented an integration level and speed-power product unmatched in previous devices. With the EP1200/EP1210, EPLDs were no longer considered just another PAL technology but a true contender for large-scale applications previously reserved for gate arrays.

Altera's EP600 device with 16 macrocells, programmable flip-flop (D/T/JK/SR) types, and asynchronous or synchronous clocking of each flip-flop has enabled the system designer to implement a variety of random logic and flip-flops all on a single chip. The "Programmable" part of EPLD now designates not only the combinatorial logic portion, but the sequential (storage) element portion also, as well as the logic's clocking scheme! The breadth of applications for these devices thus expands dramatically.

Subsequent devices have already and will continue to build on these trend-setting concepts. Altera is committed to be the industry leader in innovation and being sensitive to customer needs.

This range of capability represents an expanding design support problem, however. Each device, while having basic features in common, has its own special design requirements in terms of design tools. Solving these specific problems in the context of the user's overall design needs is the key.

## FORCES DRIVING EPLD SUPPORT

The evolution of support tools for EPLDs has been driven by many factors, both direct and indirect. The goal has always been to allow the system designer to more efficiently turn his design concept into a programmed device, transforming his idea into silicon. As the efficiency of this process increases, the technology is more widely accepted.

Indirectly, EPLD support has experienced and benefitted from the general improvements in both design methodology and CAD/CAE technology (both hardware and software) enjoyed by the design world at large. EPLDs fit well into a structured design methodology given their regular, array based architectures. Their primitive elements map well onto established TTL primitives. As such, they have been able to exploit many of the common tools developed.

Evolution of device complexity and capability required a like improvement in supporting tools. Coding a bipolar PROM via a specification of 0's and 1's may be adequate, but to effectively design with and program a complex device such as Altera's EP1200/EP1210 requires a much more global view of overall design considerations. Schematic capture or other high-level descriptions of logic are then required to efficiently deal with this level of design complexity. Simple means to

implement device simulation and programming are also essential.

## INTRODUCTION TO A+PLUS SOFTWARE

As in the area of component architecture, Altera is committed to provide effective, flexible solutions to the CAD/CAE problems the system designer faces. Particularly in those areas which require extensive device-specific knowledge, Altera has developed an effective set of PC-based CAD tools to handle the problems of design entry and programming. This package, known as A+PLUS (Altera Programmable Logic User System) provides a powerful design support tool that matches the continually expanding capabilities of Altera programmable logic devices.

Within A+PLUS, the user may enter a design via graphic schematic diagrams, text-based netlist entries, state machine descriptions, or Boolean equations. Once the design is entered, Altera's fully automated integration process, called the Altera Design Processor (ADP), translates the design into an industry-standard (JEDEC) PLD programming file. The ADP can autonomously, or with user input, make all device pin assignments and apply sophisticated logic reduction algorithms to the design to insure optimal EPLD resource utilization. A utilization report informs the user how the design was implemented.

The JEDEC file output from the ADP can be read by Altera's LogicMap II program in order to control the programmer hardware supplied by Altera. In this way, the original designer's concept is translated into an actual EPLD device for use.

## DESIGN ENTRY

To begin with, the PLD user today may choose to enter his design in a number of ways affording various levels of abstraction. The trade-off in speed of entry versus microscopic design control is one the user must wrestle with.

## BOOLEAN EQUATION ENTRY

Boolean equations or expressions provide what some would consider the "purest" method of design entry. This method provides very quick entry for simple constructs but also assumes the user has a good grasp of his design at an equation level, which may or may not be true depending on how structured the engineer's design process is. In addition, complex structures may be difficult or cumbersome to specify at this level.

Altera's approach has been to provide Boolean equation entry capability within the overall scope of its other design entry methods. In either NetMap or schematic editor (to be described below), the user may operate in the Boolean equation domain for portions of his design, and at the gate or logic primitive level for the

remainder. Boolean entry is therefore not a strait-jacket, but another tool in the designer's repertoire.

## NETLIST CAPTURE DESIGN ENTRY

Altera's NetMap program emulates schematic capture efficiently without the need for complex graphics capability on the PC. It is well-suited to the user who has done a hand-drawn schematic or other manual design and wishes to enter it for ADP processing.

In using NetMap, the user is led through a process of selecting components and specifying interconnections by the program as he systematically traverses his design tree until all elements are specified. During the input process, NetMap makes continuous connectivity and validity checks to insure a logically correct and complete design. A pictorial display of entry progress gives the designer vivid feedback on his entry. As such, NetMap represents a very effective entry method. Boolean equation entry is also available for sections of logic within NetMap at the user's discretion. The user can operate in either or both worlds as he enters his design and has control over his level of abstraction: equation or gate.

In addition, editing of designs may be done. In this case, the NetMap module simply reads in the ADF (Altera Design File) specification of the design and prompts the user for changes.

## SCHEMATIC CAPTURE DESIGN ENTRY

Optional graphic schematic capture packages allow the user to enter a design into the system in a user-friendly, Mouse and menu-driven environment. The design so-entered may include both schematic elements and Boolean equation elements. Schematic capture entry allows highly interactive creation and revision of a design, as well as hard copy printout of schematics on either a dot matrix, electrostatic or pen plotter.

Altera supports both FutureNet's DASH-2 schematic editor, as well as P-CAD's PC-CAPS package. Both of these systems are PC-based and provide Mouse-input as mentioned previously. Menus on-screen provide a quick method to acquire system skills as well as an easy method of control on an on-going basis.

DASH-2 utilizes high-resolution graphics provided by a custom graphics adapter board and standard monochrome display. PC-CAPS provides medium-resolution color graphics through the use of a standard IBM color graphics adapter board and standard color monitor. High resolution color graphics is also optionally available.

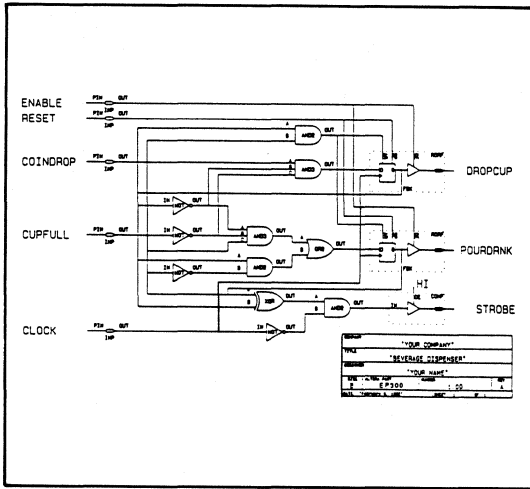
The basic functional blocks used in design entry in both systems are the Altera Design Primitives which constitute the design symbol library. An extensive set of primitives are provided for all basic logic elements such

as NAND, NOR, etc., as well as a full set of flip-flop types including D, T, JK and SR. The user can thus operate within the context of familiar TTL gate equivalents.

In addition, the user may specify a section of logic in Boolean equation form by drawing a box with the appropriate interface connections to the rest of the circuit and typing the equations defining the box's behavior into the system. The user can therefore mix entry methods as he feels appropriate.

The output of the schematic capture packages is either a pinlist or component list representing the design. These lists are then further converted into Altera Design Files (ADFs) and processed by the ADP.

A sample schematic is shown below.



## STATE MACHINE DESIGN ENTRY

State Machine entry provides the user the most abstract and highest level method of specifying his design. As such, it tends to be an efficient entry method but one which tends to insulate the user (for better or worse) from the gate-level logic. A+PLUS provides several methods for specifying designs as state machines. In all cases, the resulting designs are automatically processed by the ADP including logic minimization.

The user can specify states and conditional branches, as well as specify inputs and outputs to the state machines.

State machine descriptions may also be input directly into the Design File in the form of state tables. This allows the user to specify a state machine in terms of inputs, outputs and state transitions in tabular form.

In summary, the user has a variety of formats to work from. Compatibility with existing methodologies, speed of entry, and accuracy of representation are

considerations which will determine ultimately how the user performs this front-end task.

## FITTING

The translation of a design from its original input format (be it equation, schematic, or other) to a device-specific programming map is typically called fitting. Early PAL assemblers provided no support for software-assisted logic minimization: the user in essence had to determine the minimal set of equations to implement his logic. This could be quite time consuming and was early-on recognized as a mechanical task well suited to software algorithms.

Altera's ADP package utilizes logic minimization schemes such as DeMorgan's Inversion to aid the designer in minimizing his logic and optimally fitting into a given device. This process becomes more complex as devices become more sophisticated. Variable product term distribution, multiple flip-flop types, local / global feedback all complicate the process as they add capability.

Fitting is a process which is obviously sensitive to device architecture. A very compelling reason for vendor-supplied development tools such as Altera's A+PLUS is the need for close interaction between tools and architecture. Much as optimizing compilers must have knowledge of the target computer architecture, fitters must have knowledge of their devices. Tools developed in parallel with device development will arrive in a timely fashion at device introduction rather than lagging by substantial periods of time.

Altera's answer to the fitting problem is its ADP package. It performs three major functions.

First, regardless of the type of design entry method used, it translates the design into internal logic equations. At this stage, most design syntax errors are detected and reported to the user.

Second, the ADP performs sophisticated Boolean logic reductions on the translated design in order to maximize utilization of the EPLD's resources. The user is given complete control over the application of minimization and inversion techniques.

Finally, the ADP matches requirements of a specific design with the known resources of an Altera part. The actual fitting process results in a utilization report and a JEDEC standard programming file. The JEDEC file is used to directly program the target EPLD using Altera-supplied hardware, or general-purpose third party programmers.

In essence, the fitter is where the design and PLD meet. As such, close coupling is required. General-purpose tools are often found lacking or very late in addressing this essential step.

## PROGRAMMING

The LogicMap II program provides a graphical

user interface between the JEDEC Standard File output by the ADP and the Altera Logic Programmer hardware.

LogicMap II allows a user to edit a JEDEC Standard File and controls programming of the target Altera EPLD based upon it. In addition, it can read the contents of a programmed part (if the Security Bit is not set) to allow program verification. This capability also allows generating a JEDEC file from a previously programmed device for easy pattern duplication.

During the editing of JEDEC files, LogicMap II operates in a hierarchy of four levels. Information is displayed through a series of easy-to-use windows. The levels range from a local, detailed view of the device program, to a global block view. The combination allows rapid scoping in and out of the user's view.

LogicMap II provides an easy to use interface to the Altera Logic Programmer hardware. LogicMap II also provides control over two special features available in Altera EPLDs: the Verify Protect feature, implemented via the so-called Security Bit which prevents the part from being interrogated or inadvertently programmed, and the Turbo-Bit which allows the user to trade-off speed and power characteristics of a part.

In summary, LogicMap II provides a smooth interface between programming file and device. Altera's emphasis on maximum designer productivity carries through to this last part of the design implementation.

## SIMULATION

Design checking before system integration from components has taken on new importance with more complex designs. Granted, with EPLDs the user does have the luxury of relatively painlessly reprogramming his device if an error in the design should be found. However, by employing simulation at an early stage this type of wasted effort can be prevented.

Logic simulation, until relatively recently, has been relegated to large, complex programs running on mainframe computers. Simulation now is available on PCs, such as PCAD's PC-LOGS.

PC-LOGS is a program which allows the user to model the behavior of a circuit designed with Altera's Design Primitives and the PC-CAPS editor. The simulation program provides either graphical or tabular display of the response of the circuit to a user-supplied input pattern. The user can observe the design's response before committing it to hardware.

PC-LOGS is an interactive, event-driven, 12 state simulator. Being event driven, the program keeps track of and updates circuit node states only upon a node changing state. This method considerably shortens simulation time as opposed to blindly evaluating all nodes every time an event occurs. Input stimulus to the circuit can also be easily supplied in a variety of formats. Fault injection (stuck-at-zero, stuck-at-one) is also supported.

PC-LOGS requires no additional hardware beyond the PC to run. As a result, it is a very cost-effective arrangement.

## TEST CONSIDERATIONS

As in the simulation arena, the problem of testing FPLDs is in some sense a subset of system test issues. Concerns at the system level are likely to be mirrored at the FPLD component level.

It must be mentioned that FPLD technologies do have an impact on the need for application-specific test generation. By nature of bipolar fuse technology and the fact that these devices are not reprogrammable, the manufacturer cannot by definition test every logic element before shipment. As such, programming yield is less than 100% and may typically be in the 90% range. Erasable technologies, however, allow the programming and erasure of all elements, and thus the devices are generically testable. The manufacturer can thus guarantee essentially 100% programming yield for the customer.

Besides the obvious savings in paid-for but unshippable components, the user of a generically testable device has substantially reduced his concern with respect to hidden faults residing in the final device due to an improperly blown fuse. Remaining "holes" in the fault coverage of final test programs are more easily constricted when the individual component has been exhaustively checked.

Aside from the above concerns, PLDs hold no special problems for the user during test. Reliability of the programming elements has proven good for all technologies. The user is not likely to modify the method of or extent of his final system test due to the inclusion of PLDs.

## CONCLUSION

Existing EPLDs provide a flexible level of integration positioned above traditional TTL and below high-end gate arrays and LSI devices. The emphasis on rapid design turnaround will drive system designers more and more toward the flexibility of EPLD devices.

Future EPLD devices will incorporate new architectures which will allow the rapid design of complex devices with the equivalent of several thousand gates on a single chip. The result will be the integration of complex subsystems on single chips much as large-scale gate arrays permit today.

Portions of the design process such as schematic entry are best left to general-purpose tools supplied by third parties. Tool selection in these areas is based upon global user design concerns, not specific PLD issues. In the areas of fitting and programming, however, PLD device-specific tools are needed, and these must be provided early in the device life at introduction. Altera is committed to supplying this capability.







---

**DATA SHEETS****PAGE NO.**

---

Numerical Listing of Products .....	18
Product Selection Guide .....	19
EP300/EP310 .....	20
EP600 .....	29
EP1200/1210 .....	41
EP900 .....	53
EP1800 .....	65
PLDS2, PLE4, PLE5 .....	67
PLS2 .....	70
PLCAD1, PLE1, PLE2 .....	73
PLCAD2, PLE10, PLE20 .....	75
PLSIM1 .....	77
PLDSMA1, PLDSMA2, PLDSMA3 .....	79
A+DESIGN PRIMITIVES .....	80

PRODUCT	PAGE NO.
EP300 DC	20
EP300 DM	20
EP300-2 DC	20
EP310 DC	20
EP310 DM	20
EP310-2 DC	20
EP310-1 DC	20
EP600 DC	29
EP600 JC	29
EP600 DM	29
EP600 JM	29
EP600-3 DC	29
EP600-3 JC	29
EP600-2 DC	29
EP600-2 JC	29
EP600-1 DC	29
EP600-1 JC	29
EP900 DC	53
EP900 JC	53
EP900 JM	53
EP900-2 DC	53
EP900-2 JC	53
EP900-1 DC	53
EP900-1 JC	53
EP1200 DC	41
EP1200 JC	41
EP1210 DC	41
EP1210 JC	41
EP1210 JM	41
EP1210-2 DC	41
EP1210-2 JC	41
EP1210-1 DC	41
EP1210-1 JC	41
EP1800 JC	65
EP1800 JM	65
PLDS2	67
PLE4	67
PLE5	67
PLS2	70
PLCAD1	73
PLE1	73
PLE2	73
PLCAD2	75
PLE10	75
PLE20	75
PLSIM1	77
PLDSMA1	79
PLDSMA2	79
PLDSMA3	79
A+DESIGN PRIMITIVES	80

QUANTITATIVE DATA	PRODUCT TYPE						
	EP300	EP310	EP600	EP900	EP1200	EP1210	EP1800
Pins	20	20	24	40	40	40	68
Inputs	18	18	20	38	36	36	64
Outputs	8	8	16	24	24	24	48
Macrocells	8	8	16	24	28	28	48
Product Terms	74	74	160	240	236	236	480
<sup>1</sup> Equivalent Gates	352	352	716	1096	1176	1176	2140
Standby Current (mA@5v)	25	20	.01	.01	3	3	.01
Active Current (mA@5v, 10MHz)	25	20	15	30	35	25	60
Max Clock Speed (MHz)	18.2	40	40	33.3	12	20	25

### FEATURES

Input Latches					•	•	
Programmable I/O	•	•	•	•	•	•	•
TRI State Outputs	•	•	•	•	•	•	•
Programmable Output Polarity	•	•	•	•	•	•	•
Programmable Registers			•	•			•
Buried Registers					•	•	
Register By-pass	•	•	•	•	•	•	•
Register Pre Load	•	•	•	•	•	•	•
Register Reset	•	•	•	•	•	•	•
Programmable Clock					•	•	
Synchronous Clocking	•	•	•	•	•	•	•
Asynchronous Clocking			•	•			•
Product Term Sharing					•	•	
Variable Product Term Dist					•	•	
Global-Local Busing					•	•	•
Security Bit	•	•	•	•	•	•	•
Erasable (U.V.)	•	•	•	•	•	•	•
Ceramic Dip Package	•	•	•	•	•	•	
Ceramic J Lead Package			•	•	•	•	•

<sup>1</sup> See AB12, Page 225

# ALTERA

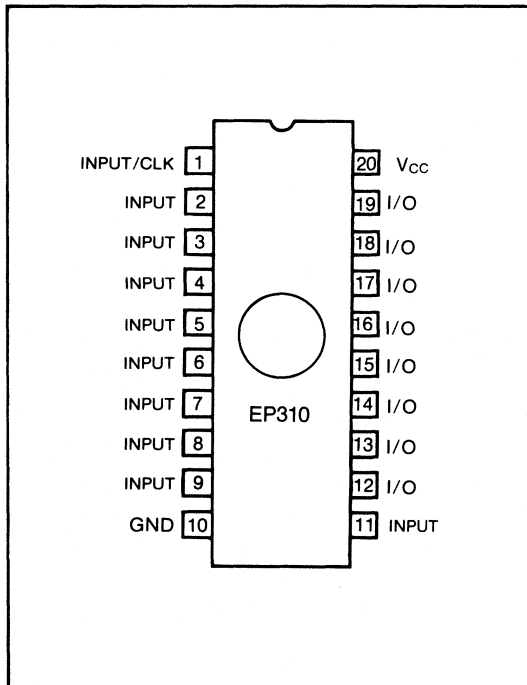
## EP300 EP310

## EPLD ERASABLE PROGRAMMABLE LOGIC DEVICE

### FEATURES

- Programmable replacement for conventional fixed logic.
- EPROM technology allows reprogrammability, ensures high programming yield and ease of use.
- Second generation programmable logic architecture allows up to 18 inputs and 8 outputs.
- Each output is *User Programmable* for combinatorial or registered operation, in active high or low mode.
- Each output also has an independently *User Programmable* feedback path.
- 100% generically testable—provides 100% programming yield.
- Programmable "Security Bit" allows total protection of proprietary designs.
- Advanced software support featuring Schematic Capture, Interactive Netlist, Boolean Equation and State Machine design entry.
- **Advanced CHMOS II-E circuitry for systems requiring low power, high performance speeds, and immunity to noise (EP310).**

### CONNECTION DIAGRAM

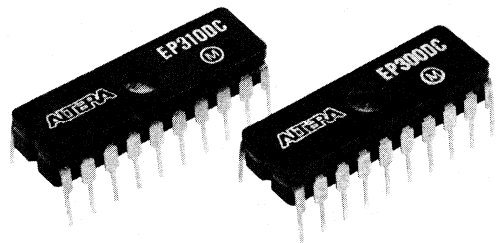


### GENERAL DESCRIPTION

The ALTERA EP300/EP310 combine the power, flexibility, and density advantages of CMOS, EPROM technology with second generation programmable logic array architecture. This combination defines a new capability in electrically programmable logic. The EP300/EP310 utilizes the familiar sum-of-products architecture which allows users to program complex custom logic functions quickly and easily. Up to 18 inputs and 8 outputs are provided, with eight product terms and a separate Output Enable term for each output.

A unique feature of the EP300/EP310 is the ability to program each output architecture on an individual basis. This gives the user the flexibility to assign either combinatorial or registered output, in either active high or active low mode, to each output pin. In addition, the feedback path can be programmed independently of the output to be either combinatorial, registered, or I/O. Other advantages include: 100% generic testing (all devices are 100% tested at the factory). The device can be erased with ultraviolet light. Design changes are no longer costly, nor is there a need for post programming testing.

Programming the EP300/EP310 is accomplished with the use of Altera's A+PLUS development software which supports four different design entry methods. Once the circuit has been entered, the A+PLUS software performs automatic translation into logical equations, boolean minimization, and design fitting directly into an EP300/EP310.



## FUNCTIONAL DESCRIPTION

A block diagram of the EP300/EP310, along with logic diagrams of the I/O Architecture Control function and the Logic Array Macrocell are shown in figures 2 through 4. The EP300/EP310 is organized in the familiar sum-of-products format with a total of 74 product terms and 36 input lines.

At each intersecting point in the logic array, there exists an EPROM type programmable connection. Initially, all connections are made. This means that both the true and complement of all inputs are connected to each product term. Connections are opened during the programming process. Therefore any product term can be connected to the true or complement of any input. When both the true and complement connections of any input are left intact, a logical false results on the output of the AND gate. If both the true and complement connections of any input are programmed open, then a logical "don't care" results for that input. If all inputs for a product term are programmed open, then a logical true results on the output of the AND gate.

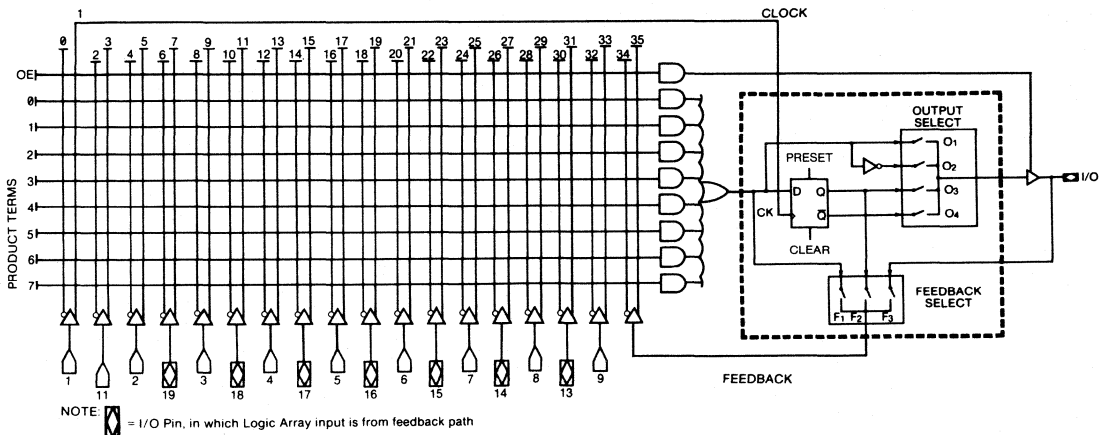
A dramatic improvement in the flexibility of programmable logic is achieved in the ALTERA EP300/EP310 through programmable I/O architecture. Each output can be combinatorial (i.e. direct output of the OR gate) or registered (i.e. output through a D type flip-flop). Both types of output can also be inverted. Independent of the output mode, the feedback can be programmed to be combinatorial, registered, I/O (i.e. directly from the pin), or none. These features enable the user to optimize the device for precise application requirements.

To improve functionality, the ALTERA EP300/EP310 has additional Synchronous Preset and Asynchronous Clear product terms. These terms are connected to all D-type Flip-Flops. When the Synchronous Preset product term is asserted (HIGH), the output register will be loaded with a HIGH on the next LOW-to-HIGH clock transition. When the Asynchronous Clear product term is asserted (HIGH), the output register will immediately be loaded with a LOW (independent of the clock). An Asynchronous Clear overrides a Synchronous Preset request. On power-up, the EP300/EP310 performs the Clear function automatically.

The EP300/EP310 is manufactured using a CMOS EPROM process. This advanced process, along with built in test features, allows 100% pre-test of each programmable connection at the factory.

Using EPROM technology provides a logic chip concept which up until now was unheard of — *Reprogrammable Custom Logic*. In the past, regardless of whether the user chose full custom, standard cell, gate array, PROM's, FPLA's, or fuse programmable logic devices, once the custom pattern was integrated into a chip, it was not alterable. In contrast when using the EP300/EP310 the same chip can be erased and reprogrammed to correct programming mistakes, updates, or design changes.

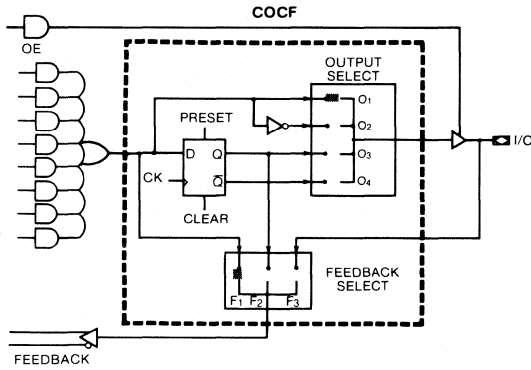
FIG. 2 EP300/EP310 MACROCELL



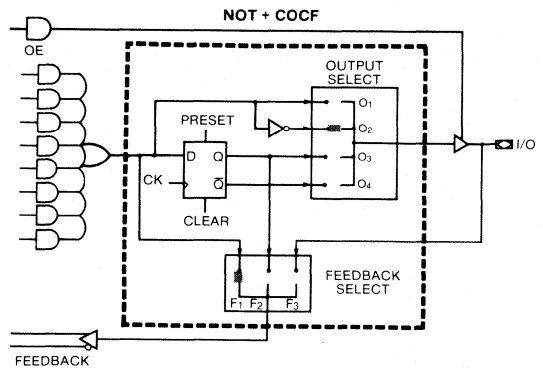
THIS DIAGRAM SHOWS ONE OF THE EIGHT MACROCELLS WITHIN THE EP300/EP310.

### FIG. 3 I/O ARCHITECTURE OPTIONS

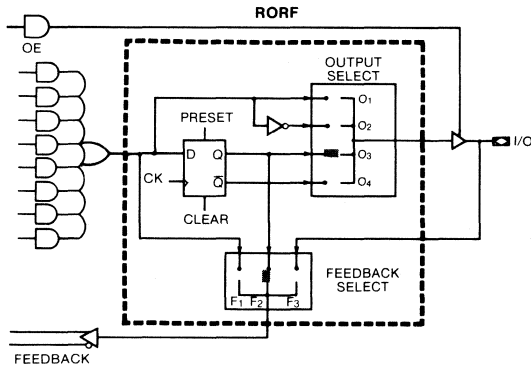
(a) Combinatorial output high. Combinatorial feedback



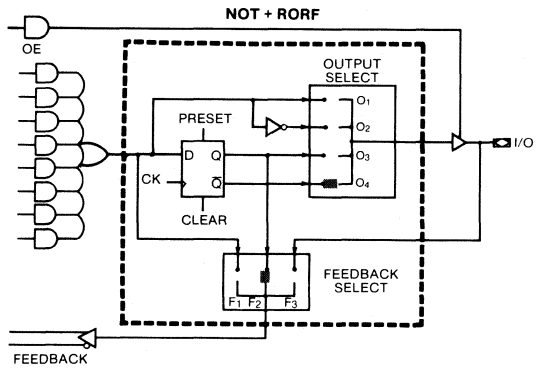
(b) Combinatorial output low. Combinatorial feedback



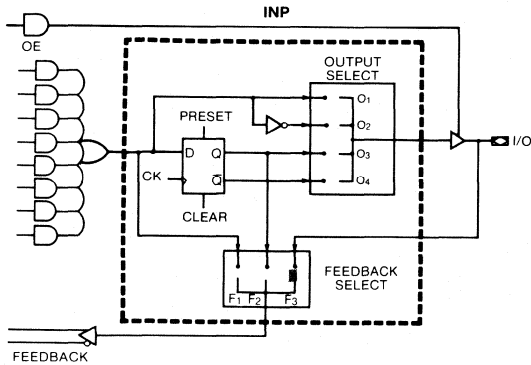
(c) Registered output high. Registered feedback



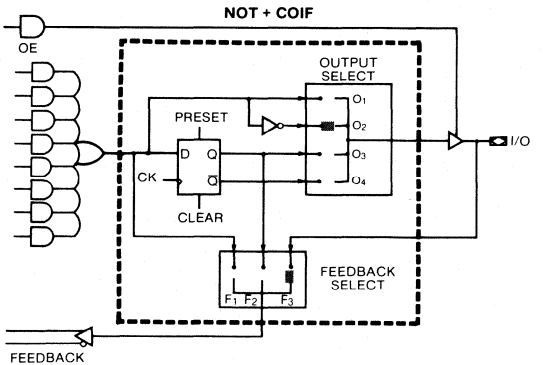
(d) Registered output low. Registered feedback



(e) Output used for additional input

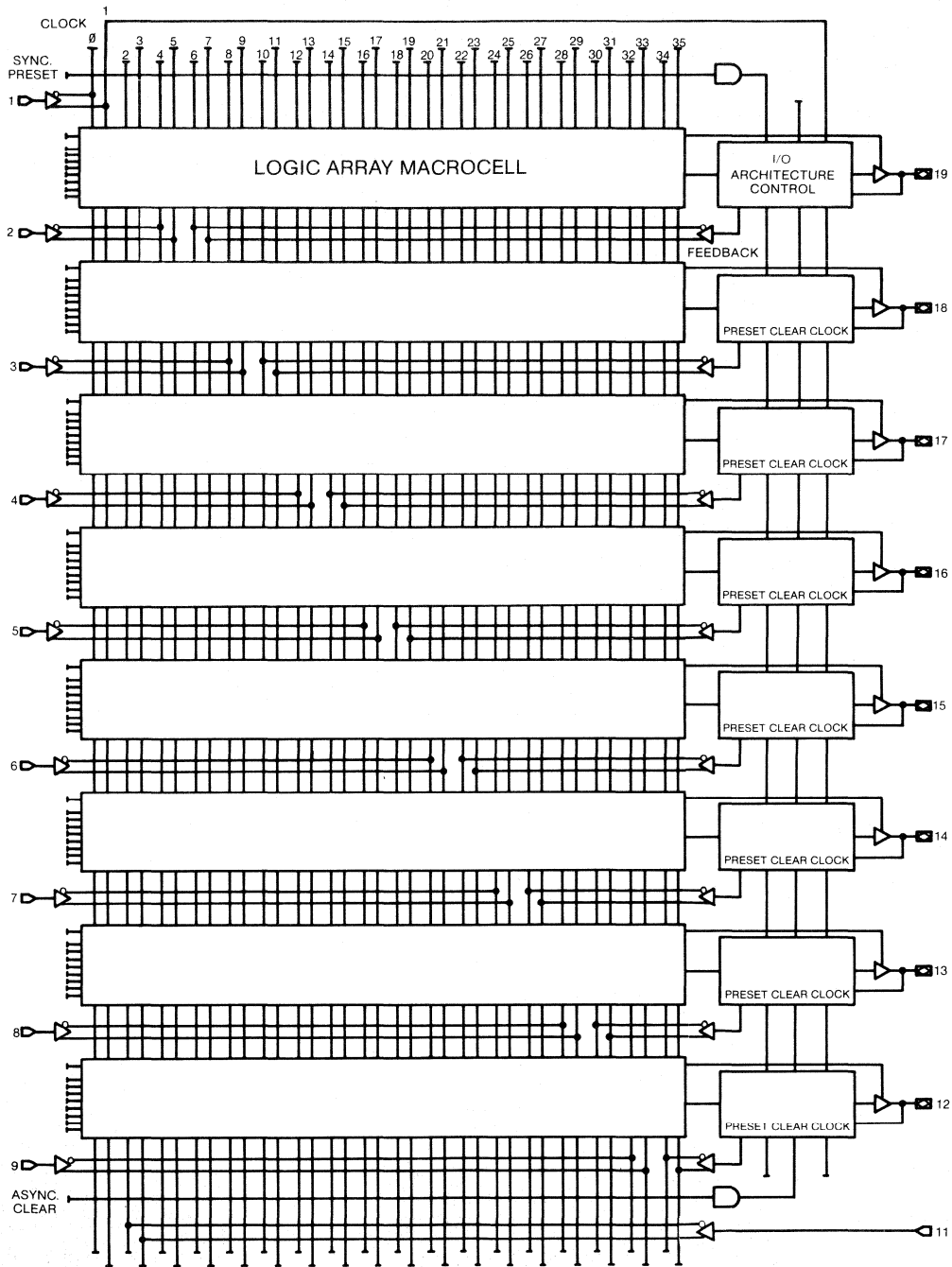


(f) Bidirectional I/O



Note: Not all architecture options are shown. Any combination of one switch from the O group and one switch from the F group is valid.

FIG. 4 EP300 AND EP310 BLOCK DIAGRAM



## ABSOLUTE MAXIMUM RATINGS

SYMBOL	PARAMETER	CONDITIONS	EP310, EP310-1, EP310-2	EP300, EP300-2, EP300 M	UNITS
V <sub>CC</sub>	Supply voltage	With respect to	-2.0 to 7.0	-0.3 to 6	V
V <sub>PP</sub>	Supply voltage		-2.0 to 13.5	-0.3 to 22	V
V <sub>I</sub>	Input voltage	GND note (3)	-2.0 to 7.0	-0.3 to 6	V
V <sub>O</sub>	Output voltage		-2.0 to 7.0	-0.3 to 6	V
T <sub>stg</sub>	Storage temperature		-65 to +150	-55 to +125	°C
I <sub>O</sub>	DC Output Current	per pin	±25	±25	mA
P <sub>D</sub>	Power dissipation		240	240	mW
I <sub>CC</sub>	DC supply current	V <sub>CC</sub> or GND	65	65	mA

## RECOMMENDED OPERATING CONDITIONS

SYMBOL	PARAMETER	CONDITIONS	MIN	MAX	UNIT
V <sub>CC</sub>	Supply voltage		4.75	5.25	V
V <sub>I</sub>	INPUT voltage		0	V <sub>CC</sub>	V
V <sub>O</sub>	OUTPUT voltage		0	V <sub>CC</sub>	V
T <sub>A</sub>	Operating temperature	Commercial	0	70	°C
T <sub>A</sub>	Operating temperature	Military	-55	125	°C
T <sub>R</sub>	INPUT rise time			500	ns
T <sub>F</sub>	INPUT fall time			500	ns
T <sub>RVCC</sub>	V <sub>CC</sub> rise time			10	ms

## DC CHARACTERISTICS

EP300/310 (T<sub>A</sub> = 0°C to 70°C, V<sub>CC</sub> = 5V ± 5%) EP300 M (T<sub>A</sub> = -55 to 125°C, V<sub>CC</sub> = 5V ± 5%) note (1)

SYMBOL	PARAMETER	CONDITIONS	EP300, EP300-2, EP310, EP310-1, EP310-2			EP300 M			UNIT
			MIN	TYP	MAX	MIN	TYP	MAX	
I <sub>I</sub>	Input leakage current	V <sub>IN</sub> = V <sub>CC</sub> or GND	-10		10	-10		10	μA
I <sub>OZ</sub>	Output leakage current	V <sub>OUT</sub> = V <sub>CC</sub> or GND	-50		50	-50		50	μA
I <sub>CC1</sub>	V <sub>CC</sub> supply current (standby)	V <sub>IN</sub> = V <sub>CC</sub> or GND		20	35		25	35	mA
V <sub>IL</sub>	Input LOW voltage		-0.3		0.8	-0.3		0.8	V
V <sub>IH</sub>	Input HIGH voltage		2.0		V <sub>CC</sub> +0.3	2.0		V <sub>CC</sub> +0.3	V
V <sub>OL</sub>	Output LOW voltage	I <sub>OL</sub> = +4.0mA			0.45			0.45	V
V <sub>OH</sub>	TTL Output HIGH voltage	I <sub>OH</sub> = -4.0mA	2.4			2.4			V
V <sub>OH</sub>	CMOS Output HIGH voltage	I <sub>OH</sub> = -2.0mA	3.84			3.84			V

## CAPACITANCE

Note (4)

SYMBOL	PARAMETER	CONDITIONS	MIN	TYP	MAX	UNIT
C <sub>IN</sub>	Input Capacitance	V <sub>IN</sub> = 0V f = 1.0 MHz			6	pF
C <sub>OUT</sub>	Output Capacitance	V <sub>OUT</sub> = 0V f = 1.0 MHz			12	pF
C <sub>CLK</sub>	Clock Pin Capacitance	V <sub>OUT</sub> = 0V f = 1.0 MHz			13	pF



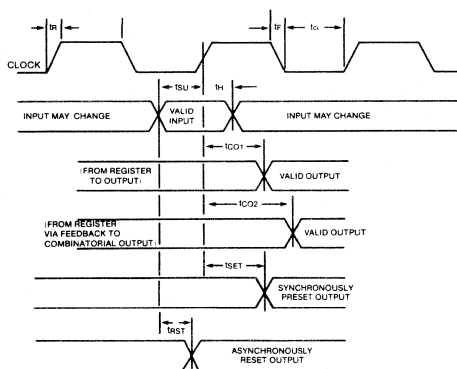
# AC CHARACTERISTICS

EP300/310 ( $T_A = 0^\circ\text{C}$  to  $70^\circ\text{C}$ ,  $V_{CC} = 5V \pm 5\%$ ) EP300 M ( $T_A = -55$  to  $125^\circ\text{C}$ ,  $V_{CC} = 5V \pm 5\%$ )

SYMBOL	PARAMETER	CONDITIONS	EP310-1		EP310-2		EP310		EP300-2		EP300		EP300 M		UNIT
			MIN	MAX	MIN	MAX	MIN	MAX	MIN	MAX	MIN	MAX	MIN	MAX	
$t_{PD}$	Input or I/O input to non-registered output			25		35		50		65	25	90	25	90	ns
$t_{pZx}$	Input or I/O input to output enable	$C_1 = 30\text{pF}$		25		35		50		65		90		90	ns
$t_{pxz}$	Input or I/O input to output disable	$C_1 = 5\text{pF}$ note 2		25		35		50		65		90		90	ns
$t_{SU}$	Input or I/O input setup time		15		28		32		47		62		62	ns	
$t_H$	Input or I/O input hold time		0		0		0		0		0		0	ns	
$t_{CH}$	Clock high time		12		15		20		25		30		30	ns	
$t_{CL}$	Clock low time		12		15		20		25		30		30	ns	
$t_{CO1}$	Clock to output delay time			18		22		28		33		38		38	ns
$t_{P1}$	Minimum clock period (register output feedback to register input via internal path)			25		30		42		55		75		75	ns
$f_1$	Maximum frequency ( $1/t_{P1}$ )	$C_1 = 30\text{pF}$	40.0		33.3		21.3		18.2		13.3		13.3	MHz	
$t_{P2}$	Minimum clock period ( $t_{SU} + t_{CO1}$ )			33		50		60		80		100		100	ns
$f_2$	Maximum frequency ( $1/t_{P2}$ )		30.0		20.0		16.6		12.5		10.0		10.0	MHz	
$t_{set}$	Synchronous preset input set-up time		15		28		35		47		62		62	ns	
$t_{ctr}$	Asynchronous output reset time delay			30		40		55		65	20	90	20	90	ns
$t_{CO2}$	Registered feedback through PLA to output. Relative to external clock			35		55		70		75		100		100	ns
$I_{CC2}$	$V_{CC}$ supply current (active)	No load, $f = 10\text{MHz}$		40		40		40		40		40		40	mA

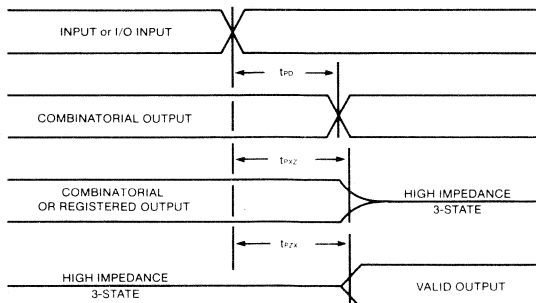
1. Typical values are for  $T_A = 25^\circ\text{C}$ ,  $V_{CC} = 5V$
2. Sample tested only for an output change of 500 mV.
3. Minimum DC input is  $-0.3V$ . During transitions, the inputs may undershoot to  $-2.0V$  for periods less than 20ns.
4. Sample tested only

**FIG. 5 SWITCHING WAVEFORMS**

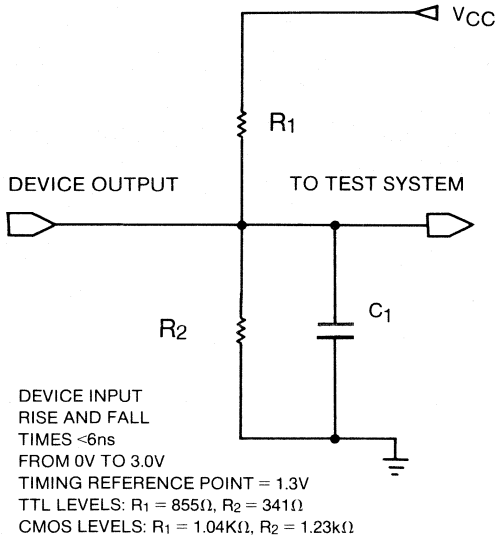


Notes:  
 $t_r$  &  $t_f = 6\text{ns}$   
 $t_{CL}$  &  $t_{CH}$  measured at 0.3V and 2.7V  
 all other timing at 1.3V  
 Input voltage levels at 0V and 3V

**FIG. 6 SWITCHING WAVEFORMS**



**FIG. 7 AC TEST CONDITIONS**



**PROGRAM ERASURE**

The erasure characteristics of the EP300/EP310 are such that erasure of the programmed connections begins to occur upon exposure to light with wavelengths shorter than approximately 4000 Angstroms. It is important to note that sunlight and certain fluorescent lighting could erase a programmed EP300/EP310 since they have wavelengths in the range of 3000 to 4000 Angstroms. Extrapolated results suggest that constant exposure to room level fluorescent lighting could erase an EP300/EP310 in approximately 3 years while it would take approximately 1 week to cause erasure when exposed to direct sunlight. As a consequence if the EP300/EP310 is to be exposed to these types of lighting conditions for extended periods of time then opaque labels should be placed over the EP300/EP310 window to prevent unintentional erasure.

The recommended erasure procedure for the EP300/EP310 is exposure to shortwave ultraviolet light which has a wavelength of 2537 Angstroms. The integrated exposure dose for erasure should be a minimum of 15 W sec/cm<sup>2</sup>. The erasure time with this dosage is approximately 15 to 20 minutes using an ultraviolet lamp with a 12000  $\mu\text{W}/\text{cm}^2$  power rating. The EP300/EP310 should be placed within 1 inch of the lamp tubes during erasure. The maximum integrated exposure dose that an EP300/EP310 can be exposed to without damage is 7000 Wsec/cm<sup>2</sup>. This is approximately one week at 12000  $\mu\text{W}/\text{cm}^2$ . Exposure of the EP300/EP310 to high intensity UV light for long periods may cause permanent damage.

**FUNCTIONAL TESTING**

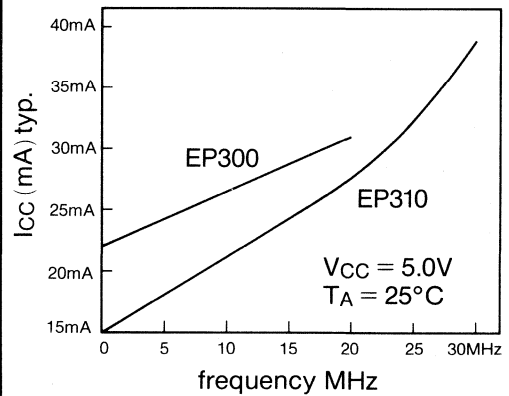
The EP300/EP310 is fully functionally tested and guaranteed through complete testing of each programmable EPROM bit and all internal logic elements.

As a result traditional problems associated with the programming yield of fusible programmable logic circuits are avoided.

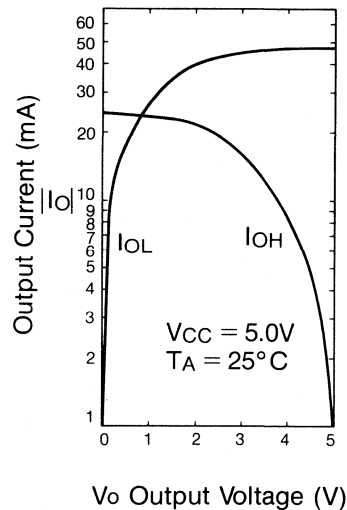
Additionally, to assist rapid testing of the EP300/EP310, a special test pre-conditioning mode is available.

This mode is entered by raising pin 9 to  $V_{HH}$  and controlling data pre-conditioning with pin 1 (see figure 11). The pre-conditioning mode permits any of the states that the EP300/EP310 could attain to be reached directly without the need for extensive input sequences to attain the desired state.

**FIG. 8  $I_{CC}$  vs.  $f_{max}$**



**FIG. 9 OUTPUT DRIVE CURRENTS**



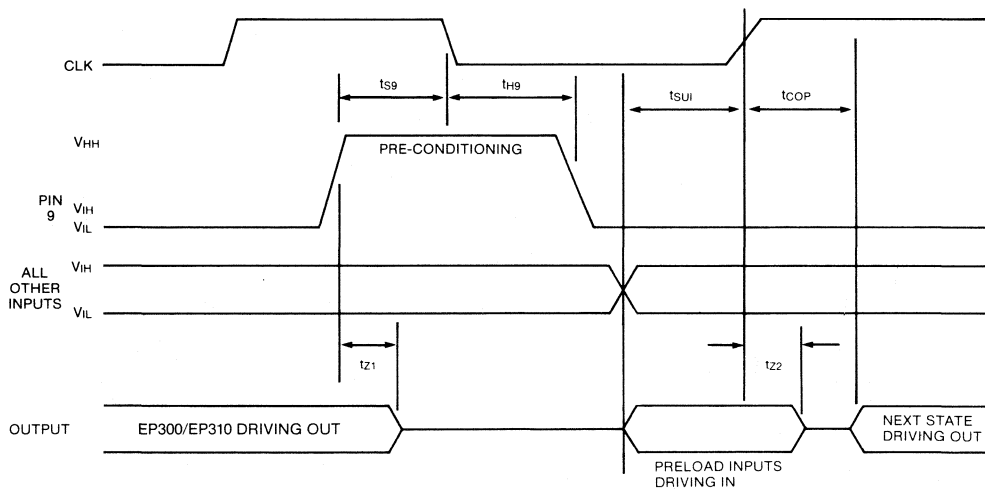
# PRECONDITIONING

## A.C. CHARACTERISTICS

( $T_A = 0^\circ\text{C}$  to  $70^\circ\text{C}$ ,  $V_{CC} = 5V \pm 5\%$ )  $V_{HH} = 21V \pm 0.5V$  for EP300 Series and  $V_{HH} = 12.5V \pm 0.5V$  for EP310 Series

SYMBOL	PARAMETER	CONDITIONS	MIN	TYP	MAX	UNIT
$t_{S9}$	Setup time of Pin 9 going to $V_{HH}$ with respect to Clock falling edge			100		ns
$t_{H9}$	Hold time of Pin 9 (going from $V_{HH}$ to $V_{IL}$ or $V_{IH}$ ) with respect to Clock falling edge			0		ns
$t_{SUI}$	Setup time of all preload inputs with respect to Clock rising edge			100		ns
$t_{COP}$	Output delay time (Register to output) after Clock rising edge			100		ns
$t_{Z1}$	Output 3-state delay time after assertion of Pre-load (Pin 9 = $V_{HH}$ )			200		ns
$t_{Z2}$	3-state delay time of external device driving in after clock rising edge			15		ns

**FIG. 10 PRECONDITIONING WAVEFORM**



## DESIGN SECURITY

The EP300/EP310 contain a programmable design security feature that controls the access to the data programmed into the device. If this programmable feature is used a proprietary design implemented in the device cannot be copied nor retrieved. This enables a high level of design control to be obtained since programmed data within EPROM cells is invisible. The bit that controls this function, along with all other program data, may be reset simply by erasing the device.

## PROGRAMMING DEVELOPMENT TOOLS

The EP300/EP310 are supported by an advanced programming development system that facilitates accurate and rapid product development.

The software system, known as A+PLUS, is the Altera Programmable Logic User System which supports multiple design entry techniques that include:

- Schematic diagram entry . . . PC-CAPS, DASH-2
- Interactive netlist entry . . . . . NetMap
- Boolean equation entry . . . . . Altera Design File

The typical development environment used for this software would be an IBM Personal Computer or equivalent machines with the following configuration:

- Dual floppy disk drive or hard disk drive
- MS-DOS operating system version 2.0 or later release
- 512k bytes of main memory
- Altera device programming card and unit

The output of A+PLUS is a data file in a standard JEDEC format. The EP300/EP310 can then be programmed using the programming card.

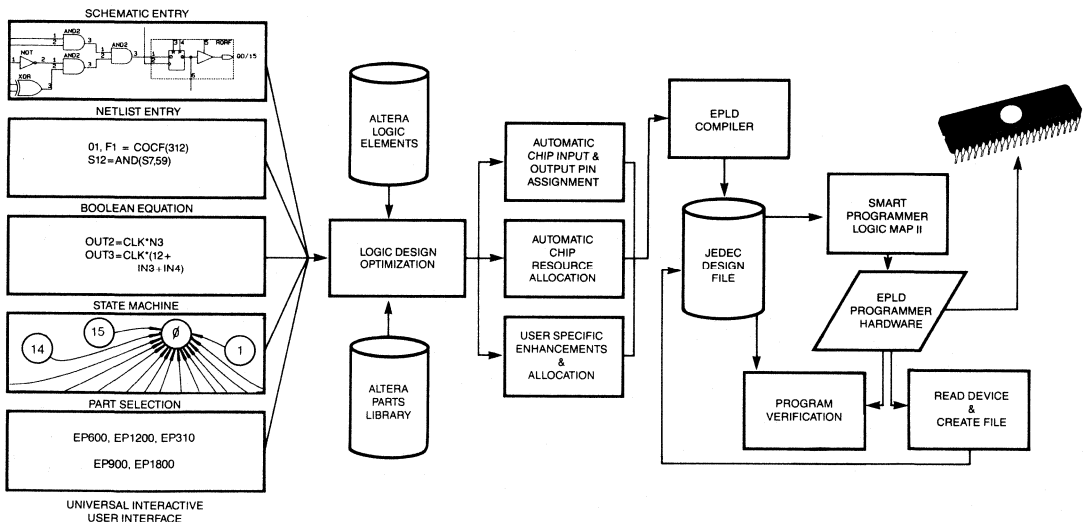
The EP300/EP310 is also supported by ABEL (Advanced Boolean Expression Language) which is available from DATA I/O and by CUPL (Compiler for Universal Programmable Logic) from Assisted Technology. All of these development programs produce standard JEDEC data files that are compatible with a variety of programming hardware including ALTERA LogicMap; Stag Microsystem's PPZ and ZL30 products and DATA I/O's LogicPak with programming/test adapter 303A-009.

## DESIGN RECOMMENDATIONS

Operation of devices described herein with conditions above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this data sheet is not implied. Exposure to absolute maximum rating condition for extended periods may affect device reliability. These devices contain circuitry to protect the input against damage to high static voltages or electric fields; however, it is advised that normal precautions be taken to avoid application of any voltage higher than maximum rated voltages to this high-impedance circuit. For proper operation, it is recommended that input and output pins be constrained to the range  $GND \leq (V_{in} \text{ or } V_{out}) \leq V_{cc}$ . Unused inputs must always be tied to an appropriate logic level (e.g. either Vcc or GND). A power supply decoupling capacitor of at least  $0.1\mu F$  must be connected directly between Vcc pin and GND.

The typical Icc versus frequency data shown in fig. 8 is derived from the performance data of an 8-bit binary counter application.

FIG. 11



## FEATURES

- High density (over 600 gates) replacement for TTL and 74HC.
- Advanced CHMOS EPROM technology, allows erase and reprogram.
- High speed,  $t_{pd} = 25$  ns.
- "Zero Power" (typically  $10\mu A$  standby).
- **Asynchronous clocking of all registers or banked register operation from 2 synchronous clocks.**
- Sixteen Macrocells with configurable I/O architecture allowing 20 inputs and 16 outputs.
- **Programmable registers providing D, T, SR or JK flipflops with individual Clear control.**
- 100% generically testable—provides 100% programming yield.
- Programmable "Security Bit" allows total protection of proprietary designs.
- Advanced software support featuring Schematic Capture, Interactive Netlist, Boolean Equation and State Machine design entry.
- Space saving 24 pin, 300 mil, dual in-line package and 28 pin J-leaded chip carrier.

## GENERAL DESCRIPTION

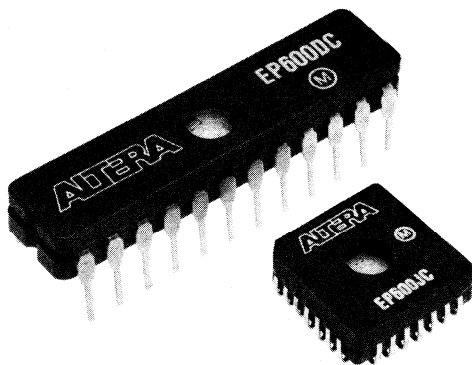
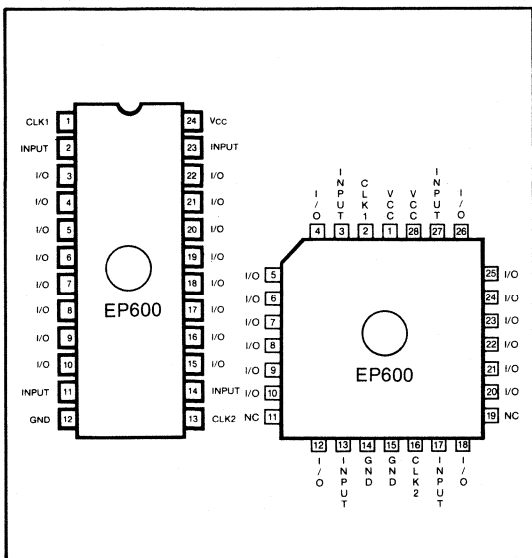
The ALTERA EP600 Programmable Logic Device is capable of implementing over 600 equivalent gates of SSI and MSI logic functions all in a space saving 24 pin, DIP, 300 mil package or a 28 pin J-leaded chip carrier.

The EP600 uses familiar sum-of-products logic providing a programmable AND with fixed OR structure. The device accommodates both combinatorial and sequential logic functions with up to 20 inputs and 16 outputs. The EP600 includes an ALTERA proprietary programmable I/O architecture providing individual selection of either combinatorial or registered output and feedback signals, active high or low.

A unique feature of the EP600 is the ability to program D, T, SR, or JK flipflop operation individually for each output without sacrificing product terms. In addition, each register can be individually clocked from any of the input or feedback paths available in the AND array. These features allow a variety of logic functions to be simultaneously implemented.

The CHMOS EPROM technology reduces the power consumption to less than 20% of equivalent bipolar devices without sacrificing speed performance. Other advantages include: 100% generic testing (all devices are 100% tested at the factory). The device can be erased with ultraviolet light. Design changes are no longer costly, nor is there a need for post programming testing.

## CONNECTION DIAGRAM



Programming the EP600 is accomplished with the use of Altera's A+PLUS development software which supports four different design entry methods. Once the circuit has been entered, the A+PLUS software performs automatic translation into logical equations, boolean minimization, and design fitting directly into an EP600.

## FUNCTIONAL DESCRIPTION

The EP600 is an Erasable Programmable Logic Device (EPLD) which uses a CMOS EPROM technology to configure connections in a programmable AND logic array. The device also contains a revolutionary programmable I/O architecture which provides advanced functional capability for user programmable logic.

Externally, the EP600 provides 4 dedicated data inputs, 2 synchronous clock inputs, and 16 I/O pins which may be configured for input, output, or bi-directional operation.

Figure 1 and 2 shows the EP600 basic Macrocell and the complete block diagram. The internal architecture is organized with familiar sum-of-products (AND-OR) structure. Inputs to the programmable AND array come from true and complement signals of the four dedicated data inputs and sixteen I/O architecture control blocks. The 40 input AND array encompasses 160 product terms which are distributed among 16 available Macrocells. Each EP600 product term repre-

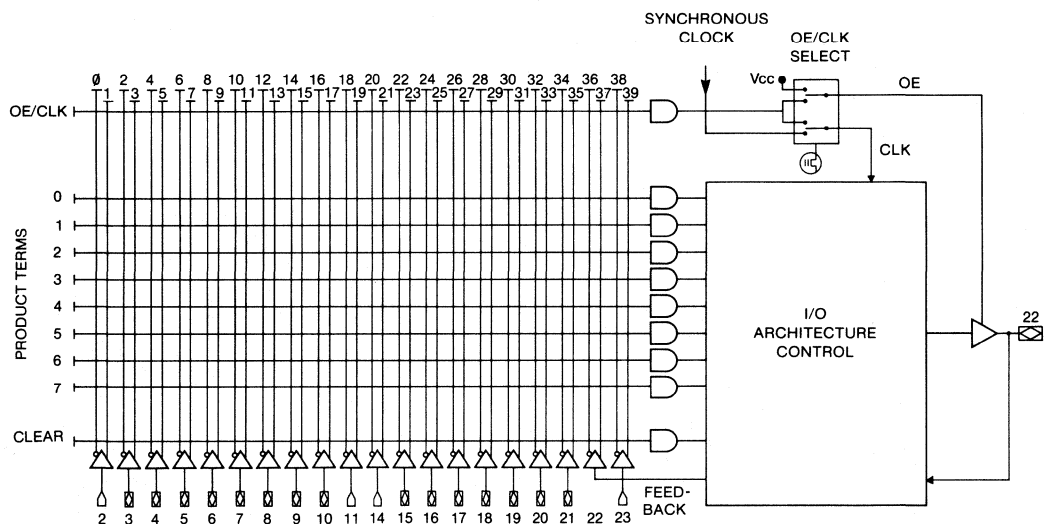
sents a 40 input AND gate.

Each Macrocell contains ten product terms. Eight product terms are dedicated for logic implementation. One product term is used for Clear control of the Macrocell internal register. The remaining product term is used for Output Enable/Asynchronous Clock implementation.

At the intersection point of an input signal and a product term there exists an EPROM connection. In the erased state, all connections are made. This means both the true and complement of all inputs are connected to each product term. Connections are opened during the programming process. Therefore, any product term may be connected to the true or complement of any array input signal. When both the true and complement of any signal is left intact, a logical false results on the output of the AND gate. If both the true and complement connections are open, then a logical "don't care" results for that input. If all inputs for the product term are programmed open, then a logical true results on the output of the AND gate.

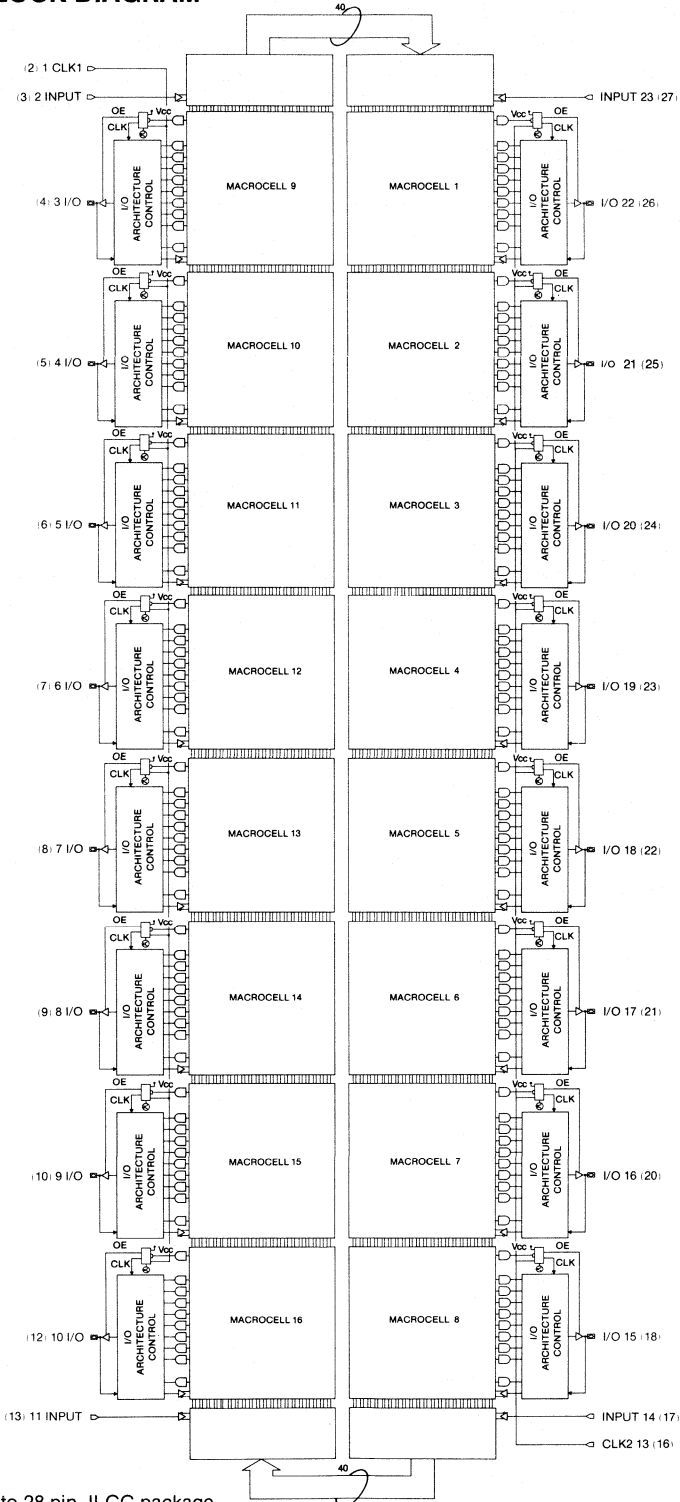
Two dedicated clock inputs provide synchronous clock signals to the EP600 internal registers. Each of the clock signals controls a bank of eight registers. CLK1 controls registers associated with Macrocells 9-16. CLK2 controls registers associated with Macrocells 1-8. The EP600 advanced I/O architecture allows the number of synchronous registers to be user defined, from one to sixteen. Both dedicated clock inputs are positive edge triggered.

FIG. 1 LOGIC ARRAY MACROCELL



NOTE:  = I/O pin, in which Logic Array input is from feedback path.

FIG. 2 EP600 BLOCK DIAGRAM



Pin #'s in ( ) pertain to 28 pin JLCC package

## I/O ARCHITECTURE

The EP600 Input/Output Architecture provides each Macrocell with over 50 possible I/O configurations. Each I/O can be configured for combinatorial or registered output, with programmable output polarity. Four different types of registers (D, T, JK, SR), can be implemented into every I/O without any additional logic requirements. I/O feedback selection can also be programmed for registered or input (pin) feedback. Another benefit of the EP600 I/O architecture is its ability to individually clock each internal register from asynchronous clock signals.

### OE/CLK Selection

Figure 3 shows the two modes of operation which are provided by the OE/CLK Select Multiplexer. The operation of this multiplexer is controlled by a single EPROM bit and may be individually configured for each EP600 I/O pin. In Mode 0, the three-state output buffer is controlled by a single product term. If the output of the AND gate is a logical true then the output buffer is enabled. If a logical false resides on the output of the AND gate then the output buffer is seen as high impedance. In this mode the Macrocell flipflop may be clocked by its respective synchronous clock input. After erasure, OE/CLK Select Mux is configured as Mode 0.

In Mode 1, the Output Enable buffer is always enabled. The Macrocell flipflop now may be triggered from an asynchronous clock signal generated by the OE/CLK multiplexable product term. This mode allows individual clocking of flipflops from any available signal in the AND array. Because both true and complement

signals reside in the AND array, the flipflop may be configured for positive or negative edge trigger operation. With the clock now controlled by a product term, gated clock structures are also possible.

### OUTPUT/FEEDBACK Selection

Figure 4 shows the EP600 basic output configurations. Along with combinatorial output, four register types are available. Each Macrocell I/O may be independently configured. All registers have individual Asynchronous Clear control from a dedicated product term. When the product term is asserted to a logical one, the Macrocell register will immediately be loaded with a logical zero independently of the clock. On power up, the EP600 performs the Clear function automatically.

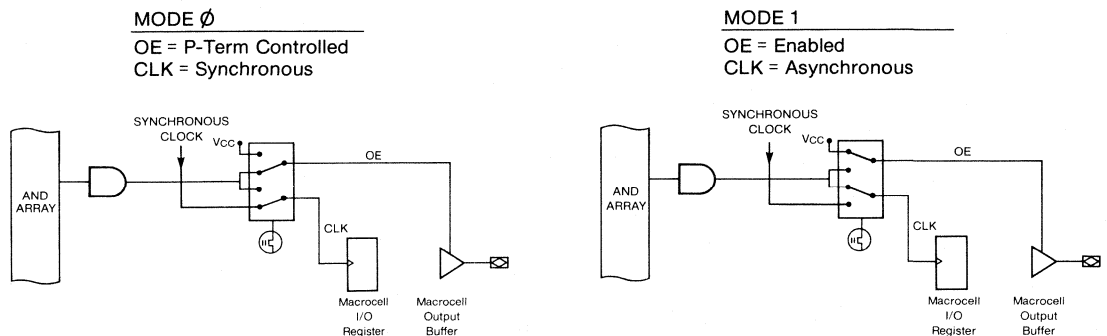
When the D or T register is selected, eight product terms are ORed together and made available to the register input. The Invert Select EPROM bit determines output polarity. The Feedback Select Multiplexer enables registered, I/O (pin) or no feedback to the AND array.

If the JK or SR registers are selected, the eight product terms are shared among two OR gates. The allocation of product terms for each register input is optimized by the A+PLUS development software. The Invert Select EPROM bits configures output polarity. The Feedback Select Multiplexer enables registered or no feedback to the AND array.

Any I/O pin may be configured as a dedicated input by selecting no output and pin feedback. No output is obtained by disabling the Macrocell output buffer.

In the erased state, the I/O is configured for combinatorial active low output with input (pin) feedback.

**FIG. 3 OE/CLK SELECT MUX**



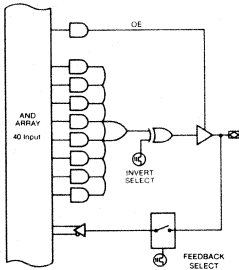
The register is clocked by the synchronous clock signal which is common to 7 other Macrocells. The output is enabled by the logic from the product term.

The output is permanently enabled and the register is clocked via the product term. This allows for gated clocks that may be generated from elsewhere in the EP600.



**FIG. 4 I/O CONFIGURATIONS**

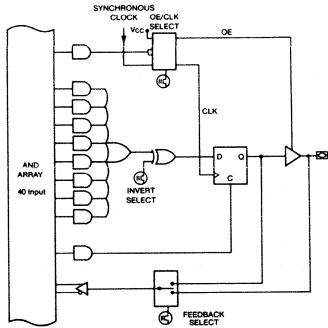
**COMBINATORIAL**



**I/O SELECTION**

OUTPUT/POLARITY	FEEDBACK
Combinatorial/High	Pin, None
Combinatorial/Low	Pin, None
None	Pin

**D-TYPE FLIP-FLOP**



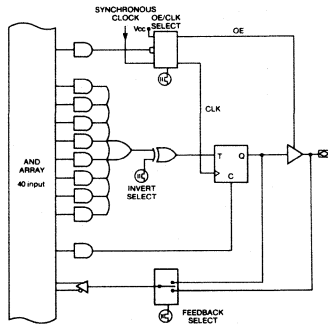
**I/O SELECTION**

OUTPUT/POLARITY	FEEDBACK
D-Register/High	D-Register, Pin, None
D-Register/Low	D-Register, Pin, None
None	D-Registered
None	Pin

**FUNCTION TABLE**

D	Q <sub>n</sub>	Q <sub>n+1</sub>
0	0	0
0	1	0
1	0	1
1	1	1

**TOGGLE FLIP-FLOP**



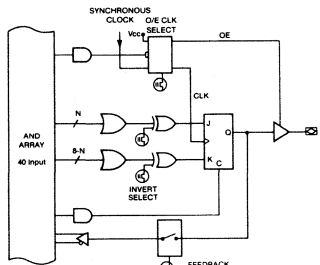
**I/O SELECTION**

OUTPUT/POLARITY	FEEDBACK
T-Register / High	T-Register, Pin, None
T-Register / Low	T-Register, Pin, None
None	T-Registered
None	Pin

**FUNCTION TABLE**

T	Q <sub>n</sub>	Q <sub>n+1</sub>
0	0	0
0	1	1
1	0	1
1	1	0

**JK FLIP-FLOP**



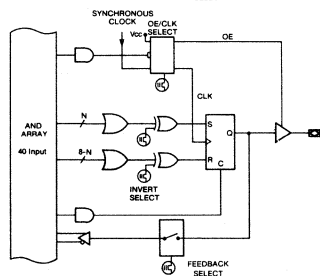
**I/O SELECTION**

OUTPUT/POLARITY	FEEDBACK
JK Register/High	JK Register, None
JK Register/Low	JK Register, None
None	JK Register

**FUNCTION TABLE**

J	K	Q <sub>n</sub>	Q <sub>n+1</sub>
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

**SR FLIP-FLOP**



**I/O SELECTION**

OUTPUT/POLARITY	FEEDBACK
SR Register/High	SR Register, None
SR Register/Low	SR Register, None
None	SR Register

**FUNCTION TABLE**

S	R	Q <sub>n</sub>	Q <sub>n+1</sub>
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1

**ABSOLUTE MAXIMUM RATINGS**

EP600, EP600-1, EP600-2, EP600-3, EP600 DM, EP600 JM

Note: See Design Recommendations

SYMBOL	PARAMETER	CONDITIONS	MIN	MAX	UNIT
V <sub>CC</sub>	Supply voltage	With respect to GND note (3)	-2.0	7.0	V
V <sub>PP</sub>	Programming supply voltage		-2.0	13.5	V
V <sub>I</sub>	DC INPUT voltage		-2.0	7.0	V
I <sub>MAX</sub>	DC V <sub>CC</sub> or GND current			+100.0	mA
I <sub>OUT</sub>	DC OUTPUT Current, per pin		-25	+25	mA
P <sub>D</sub>	Power Dissipation			250	mW
T <sub>STG</sub>	Storage temperature	No bias	-65	+150	°C
T <sub>AMB</sub>	Ambient temperature	Under bias	-10	+85	°C

**RECOMMENDED OPERATING CONDITIONS**

SYMBOL	PARAMETER	CONDITIONS	MIN	MAX	UNIT
V <sub>CC</sub>	Supply voltage		4.75	5.25	V
V <sub>I</sub>	INPUT voltage		0	V <sub>CC</sub>	V
V <sub>O</sub>	OUTPUT voltage		0	V <sub>CC</sub>	V
T <sub>A</sub>	Operating temperature	Commercial	0	70	°C
T <sub>A</sub>	Operating temperature	Military	-55	125	°C
T <sub>R</sub>	INPUT rise time			500	ns
T <sub>F</sub>	INPUT fall time			500	ns
T <sub>RVCC</sub>	V <sub>CC</sub> rise time			10	ms

**DC OPERATING CHARACTERISTICS**

Note (1)

SYMBOL	PARAMETER	CONDITIONS	MIN	TYP	MAX	UNIT
V <sub>IH</sub>	HIGH level input voltage		2.0		V <sub>CC</sub> +0.3	V
V <sub>IL</sub>	LOW level input voltage		-0.3		0.8	V
V <sub>OH</sub>	HIGH level TTL output voltage	I <sub>OH</sub> = -4.0mA DC	2.4			V
V <sub>OH</sub>	HIGH level CMOS output voltage	I <sub>OH</sub> + -2.0 mA DC	3.84			V
V <sub>OL</sub>	LOW level output voltage	I <sub>OL</sub> = 4.0mA DC			.45	V
I <sub>I</sub>	Input leakage current	V <sub>I</sub> = V <sub>CC</sub> or GND			±10.0	μA
I <sub>OZ</sub>	3-state output off-state current	V <sub>O</sub> = V <sub>CC</sub> or GND			±10.0	μA
I <sub>CC1</sub>	V <sub>CC</sub> supply current (standby)	V <sub>I</sub> = V <sub>CC</sub> or GND I <sub>O</sub> = 0		10	100	μA
I <sub>CC2</sub>	V <sub>CC</sub> supply current (active)	No load f = 1.0 MHz		1.5	4.0	mA

**CAPACITANCE**

Note (4)

SYMBOL	PARAMETER	CONDITIONS	MIN	TYP	MAX	UNIT
C <sub>IN</sub>	Input Capacitance	V <sub>IN</sub> = 0V f = 1.0 MHz			6	pF
C <sub>OUT</sub>	Output Capacitance	V <sub>OUT</sub> = 0V f = 1.0 MHz			12	pF
C <sub>CLK</sub>	Clock Pin Capacitance	V <sub>OUT</sub> = 0V f = 1.0 MHz			13	pF

## AC CHARACTERISTICS

EP600, -1, -2, -3: ( $T_A = 0^\circ\text{C}$  to  $70^\circ\text{C}$ ,  $V_{CC} = 5V \pm 5\%$ )EP600 DM/JM: ( $T_A = -55^\circ\text{C}$  to  $125^\circ\text{C}$ ,  $V_{CC} = 5V \pm 5\%$ )

See note (5)

SYMBOL	PARAMETER	CONDITIONS	EP600-1		EP600-2		EP600-3		EP600		EP600 DM EP600 JM		UNIT
			MIN	MAX	MIN	MAX	MIN	MAX	MIN	MAX	MIN	MAX	
$t_{PD}$	Input or I/O input to non-registered output			25		35		45		55		55	ns
$t_{PZX}$	Input or I/O input to output enable	$C_1 = 50\text{pF}$		25		35		45		55		55	ns
$t_{PXZ}$	Input or I/O input to output disable	$C_1 = 5\text{pF}$ note (2)		25		35		45		55		55	ns
$t_{CLR}$	Asynchronous output clear time	$C_1 = 50\text{pF}$		25		35		45		55		55	ns

## SYNCHRONOUS CLOCK MODE

SYMBOL	PARAMETER	CONDITIONS	EP600-1		EP600-2		EP600-3		EP600		EP600 DM EP600 JM		UNIT
			MIN	MAX	MIN	MAX	MIN	MAX	MIN	MAX	MIN	MAX	
$t_{SU}$	Input or I/O input setup time		18		28		30		35		35		ns
$t_H$	Input or I/O input hold time		0		0		0		0		0		ns
$t_{CH}$	Clock high time		12.5		15		17.5		22.5		22.5		ns
$t_{CL}$	Clock low time		12.5		15		17.5		22.5		22.5		ns
$t_{CO1}$	Clock to output delay			15		22		25		30		30	ns
$t_{P1}$	Minimum clock period (register output feedback to register input - internal path)			25		30		35		45		45	ns
$f_1$	Maximum frequency ( $1/t_{P1}$ )		40		33.3		28.6		22.2		22.2		MHz
$t_{P2}$	Minimum clock period ( $t_{SU} + t_{CO1}$ )			33		50		55		65		65	ns
$f_2$	Maximum frequency ( $1/t_{P2}$ )		30		20		18.2		15.4		15.4		MHz
$t_{CO2}$	Registered feedback through PLA to output. Relative to external clock			35		40		45		55		55	ns

## ASYNCHRONOUS CLOCK MODE

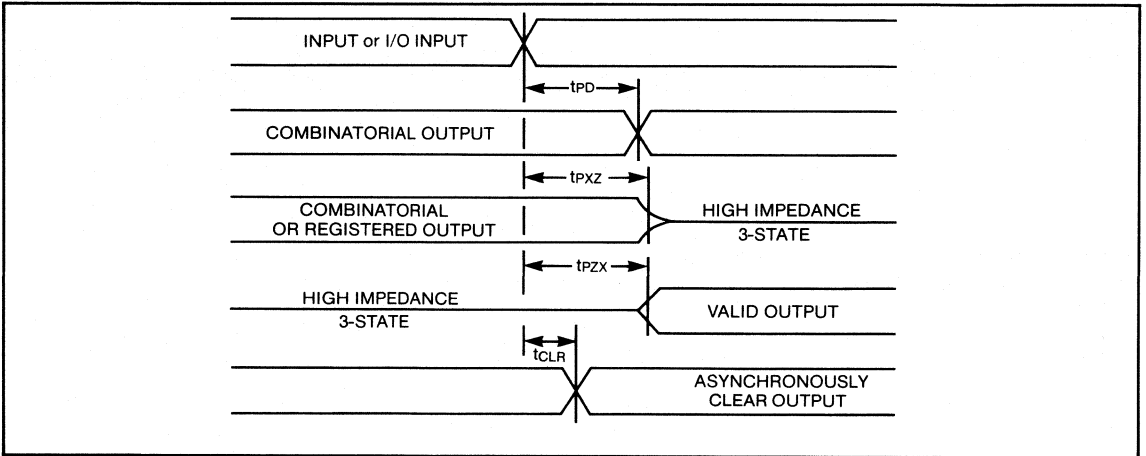
SYMBOL	PARAMETER	CONDITIONS	EP600-1		EP600-2		EP600-3		EP600		EP600 DM EP600 JM		UNIT
			MIN	MAX	MIN	MAX	MIN	MAX	MIN	MAX	MIN	MAX	
$t_{ASU}$	Input or I/O input setup time		5		5		5		5		5		ns
$t_{AH}$	Input or I/O input hold time		10		10		10		10		10		ns
$t_{ACH}$	Clock high time		12.5		15		17.5		22.5		22.5		ns
$t_{ACL}$	Clock low time		12.5		15		17.5		22.5		22.5		ns
$t_{ACO1}$	Clock to output delay			30		40		50		60		60	ns
$t_{AP1}$	Minimum clock period (register output feedback to register input - internal path)			25		30		35		45		45	ns
$f_{A1}$	Maximum frequency ( $1/t_{AP1}$ )		40		33.3		28.6		22.2		22.2		MHz
$t_{AP2}$	Minimum clock period ( $t_{ASU} + t_{ACO1}$ )			35		45		55		65		65	ns
$f_{A2}$	Maximum frequency ( $1/t_{AP2}$ )		28.6		22.2		18.2		15.4		15.4		MHz
$t_{ACO2}$	Registered feedback through PLA to output. Relative to an asynchronous input pin			50		70		90		110		110	ns

## Notes:

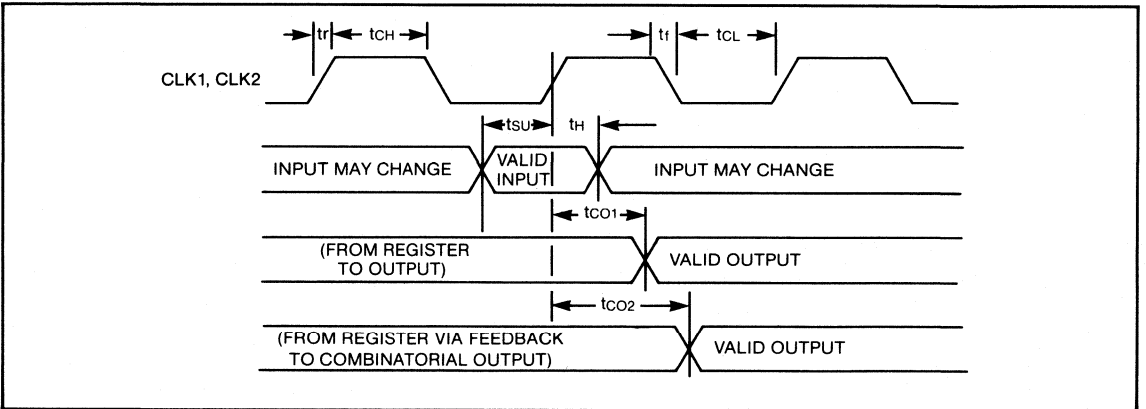
1. Typical values are for  $T_A = 25^\circ$ ,  $V_{CC} = 5V$ .
2. Sample tested only for an output change of 500mV.
3. Minimum DC input is -0.3V. During transitions, the inputs may undershoot to -2.0V for periods less than 20ns.
4. Capacitance measured at  $25^\circ\text{C}$ . Sample tested only. Clock Pin Capacitance for dedicated clock inputs only.
5. All AC values tested with TURBO-BIT™ not programmed.

**FIG. 5 SWITCHING WAVEFORMS**

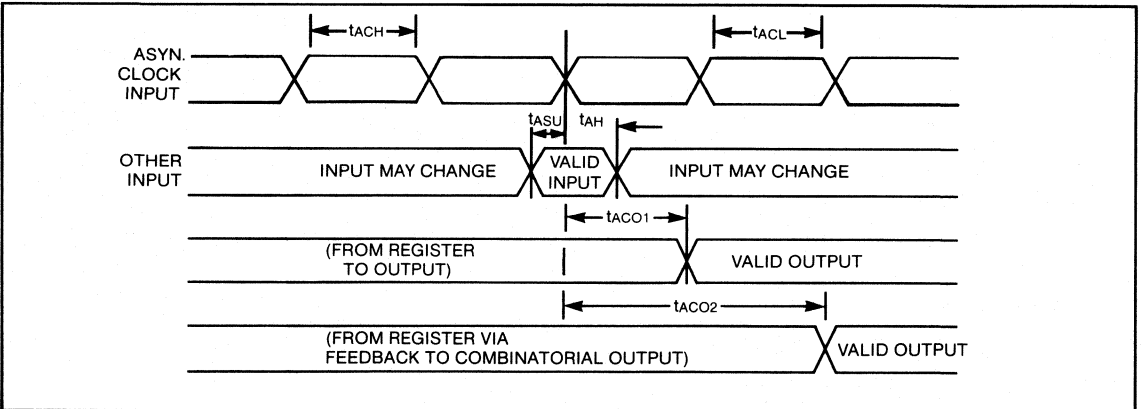
**COMBINATORIAL MODE**



**SYNCHRONOUS CLOCK MODE**



**ASYNCHRONOUS CLOCK MODE**



Notes:  $t_r$  &  $t_f = 6\text{ns}$   
 $t_{CL}$  &  $t_{CH}$  measured at 0.3V and 2.7V  
 all other timing at 1.3V  
 Input voltage levels at 0V and 3V

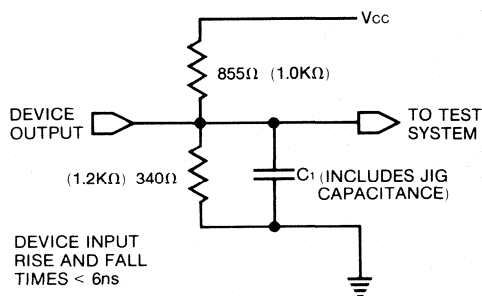
## PROGRAM ERASURE

The erasure characteristics of the EP600 are such that erasure of the programmed connections begins to occur upon exposure to light with wavelengths shorter than approximately 4000 Angstroms. It is important to note that sunlight and certain fluorescent lighting could erase a programmed EP600 since they have wavelengths in the range of 3000 to 4000 Angstroms. Extrapolated results suggest that constant exposure to room level fluorescent lighting could erase an EP600 in approximately 3 years while it would take approximately 1 week to cause erasure when exposed to direct sunlight. As a consequence, if the EP600 is to be exposed to these types of lighting conditions for extended periods of time, opaque labels should be placed over the EP600 window to prevent unintentional erasure.

The recommended erasure procedure for the EP600 is exposure to shortwave ultraviolet light which has a wavelength of 2537 Angstroms. The integrated exposure dose for erasure should be a minimum of 15Wsec/cm<sup>2</sup>. The erasure time with this dosage is approximately 15 to 20 minutes using an ultraviolet lamp with a 12000  $\mu$ W/cm<sup>2</sup> power rating. The EP600 should be placed within 1 inch of the lamp tubes during erasure. The maximum integrated exposure dose for an EP600 without damage is 7000 Wsec/cm<sup>2</sup>. This is approximately one week at 12000  $\mu$ W/cm<sup>2</sup>. Exposure of the EP600 to high intensity UV light for long periods may cause permanent damage.

The EP600 may be erased and re-programmed as many times as needed using the recommended erasure exposure levels.

**FIG. 6 AC TEST CONDITIONS**



## FUNCTIONAL TESTING

The EP600 is fully functionally tested and guaranteed through complete testing of each programmable EPROM bit and all internal logic elements thus ensuring 100% programming yield.

As a result, traditional problems associated with fuse-programmed circuits are eliminated. The erasable nature of the EP600 allows test program patterns to be used and then erased. This facility to use application-independent, general purpose tests is called generic testing and is unique among user-defined LSI logic devices.

To enable functional evaluation of counter and state-machine applications, the EP600 contains register pre-load circuitry. This can be activated by interrupting the normal clocked sequence and applying V<sub>HH</sub> on pin 11 to engage the pre-load state. Under these conditions the flip-flops in the EP600 can be set to any logical condition and then return to normal operation. This process simplifies the input sequences necessary to evaluate counter and state-machine operations. If the inverted output path is selected, (active low output) a "1" at the I/O pad will result in preloading a "1" into the register. If an active high output path is selected, then a "1" on the I/O pad will result in preloading a "0" into the register. The preload waveforms are shown in Figure 7.

## DESIGN SECURITY

The EP600 contains a programmable design security feature that controls the access to the data programmed into the device. If this programmable feature is used, a proprietary design implemented in the device cannot be copied nor retrieved. This enables a high level of design control to be obtained since programmed data within EPROM cells is invisible. The bit that controls this function, along with all other program data, may be reset simply by erasing the device.

## LATCH-UP

The EP600 input, I/O, and clock pins have been carefully designed to resist latch-up which is inherent in CMOS structures. Each of the EP600 pins will not latch-up with currents up to 100 mA and voltages from -1V to V<sub>CC</sub>+1V. Additionally, the programming pin is designed to resist latch-up to the 13.5 volt maximum device limit.

## PRECONDITIONING MODE

## A.C. CHARACTERISTICS

( $T_A = 0^\circ$  to  $70^\circ\text{C}$ ,  $V_{CC} = 5V \pm 5\%$ ,  $V_{HH} = 12V \pm 0.5V$ )

SYMBOL	PARAMETER	CONDITIONS	MIN	TYP	MAX	UNIT
$t_{pz}$	Output 3-state delay time after assertion of Preload (Pin 11 = $V_{HH}$ )			100		ns
$t_{ph}$	Hold time of all preload inputs, with respect to Clock rising edge			15		ns
$t_{psu}$	Setup time of all preload inputs, with respect to Clock rising edge			100		ns
$t_{plh}$	Hold time for Preconditioning input			50		ns
$t_{co1}$	Output delay time (Register to output) after Clock rising edge			50		ns
$t_{su}$	Setup time for all other inputs with respect to Clock rising edge			50		ns
$t_h$	Hold time for all other inputs with respect to Clock rising edge			0		ns

FIG. 7 PRECONDITIONING WAVEFORM

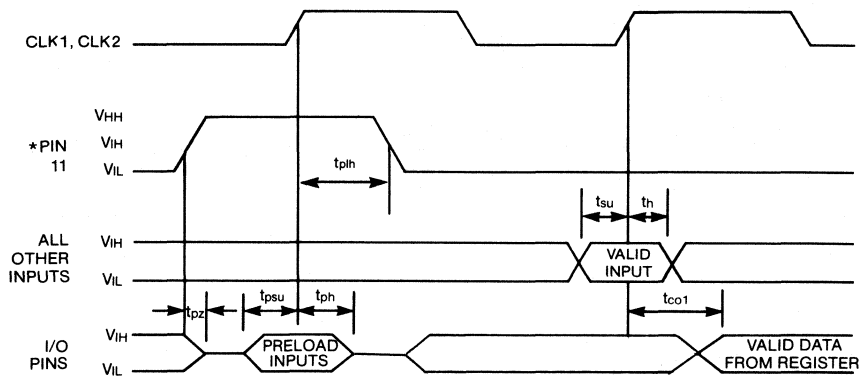
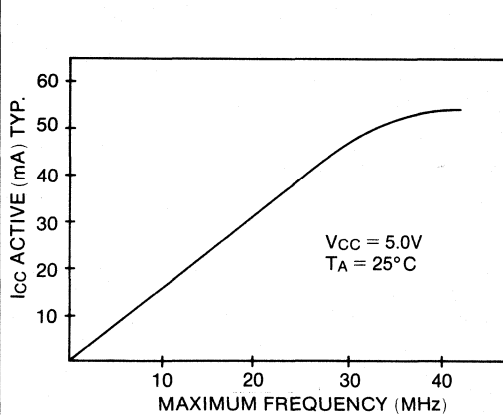
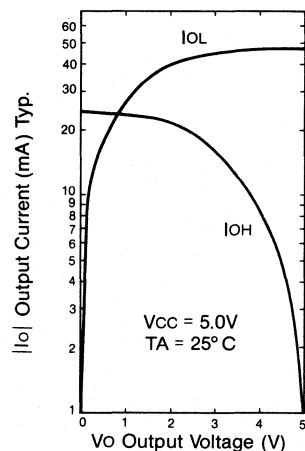
FIG. 8  $I_{CC}$  vs.  $f_{max}$ 

FIG. 9 OUTPUT DRIVE CURRENTS



## PROGRAMMING DEVELOPMENT TOOLS

The EP600 is supported by an advanced programming development system that facilitates accurate and rapid product development.

The software system, known as A+PLUS, is the **Altera Programmable Logic User System** which supports multiple design entry techniques that include:

- Schematic diagram entry ... PC-CAPS or DASH-2
- Interactive netlist entry ..... NetMap
- Boolean equation entry ..... Altera Design File

The typical development environment used for this software would be an IBM Personal Computer and equivalent machines with the following configuration:

- Dual floppy disk drive or hard disk drive
- MS-DOS operating system version 2.0 or later release
- 512K memory
- Altera device programming card and unit

The output of A+PLUS is a data file in a standard JEDEC format. The EP600 can then be programmed using the Altera programming card.

All of the EP600 architecture features are supported by Altera symbol primitives. The symbol library interfaces with each of the A+PLUS design entry methods. These primitives are in the form of Input, Basic Logic Gates, Equations, and I/O architectures.

When using the primitive symbols, the desired output polarity is obtained by appropriately connecting inverters to the inputs of the I/O resources. Figure 11 shows how to achieve active low outputs. For combinatorial output or D flipflops, the I/O primitive input is driven by an inverter. The feedback (if used) also becomes active low. By placing another inverter in the feedback, path on active high signal is obtained. For the JK or SR flipflops, simply reverse the input connections.

In this case, a Set now becomes a Reset and a Reset now becomes a Set thus achieving an active low output. After the EP600 powers up, the JK or SR flipflop output must first be preset to a logical one for proper active low operation.

To specify asynchronous clock operation, the CLK input into a flipflop must be driven by the Asynchronous Clock Buffer primitive. Figure 12 illustrates asynchronous clock connections.

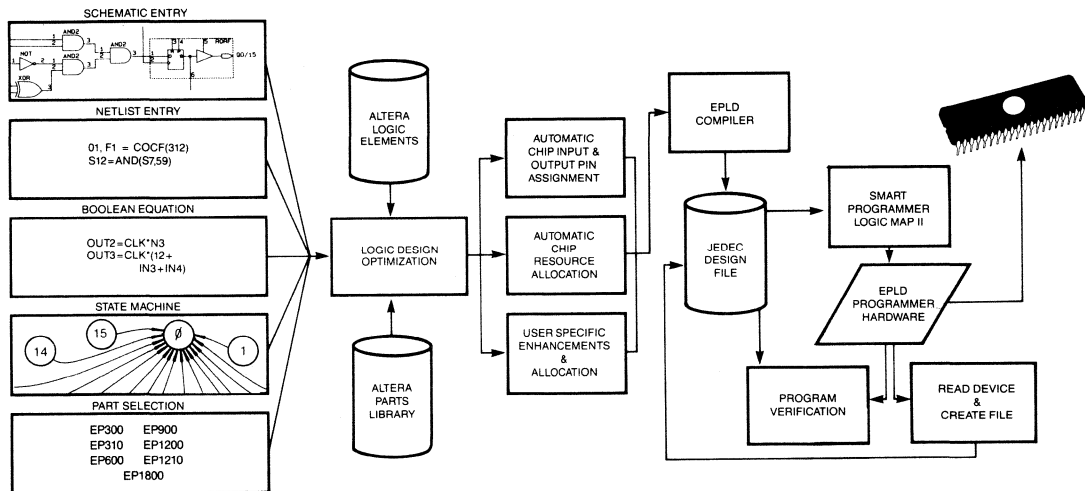
## DESIGN RECOMMENDATIONS

Operation of devices described herein with conditions above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this data sheet is not implied. Exposure to absolute maximum rating condition for extended periods may affect device reliability. These devices contain circuitry to protect the input against damage to high static voltages or electric fields; however, it is advised that normal precautions be taken to avoid application of any voltage higher than maximum rated voltages to this high-impedance circuit.

For proper operation, it is recommended that input and output pins be constrained to the range  $GND \leq (V_{in} \text{ or } V_{out}) \leq V_{CC}$ . Unused inputs must always be tied to an appropriate logic level (e.g. either VCC or GND). A power supply decoupling capacitor of at least  $0.2\mu F$  must be connected directly between VCC pin and GND.

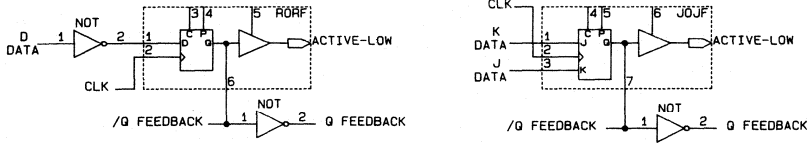
The typical  $I_{CC}$  versus frequency data shown in Fig.8 is derived from the performance data of a 16-bit binary counter application. The EP600 contains a programmable option to control the autonomous power down feature that enables the low standby current

**FIG. 10 ALTERA PROGRAMMABLE LOGIC USER SYSTEM**

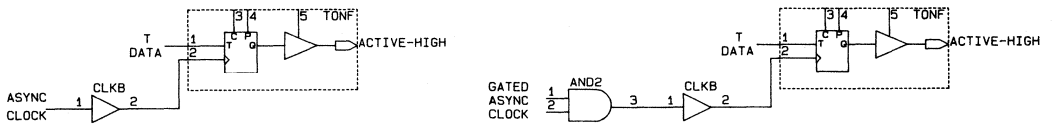


UNIVERSAL INTERACTIVE  
USER INTERFACE

**FIG. 11 ACTIVE-LOW OUTPUTS**



**FIG. 12 ASYNCHRONOUS CLOCKING**



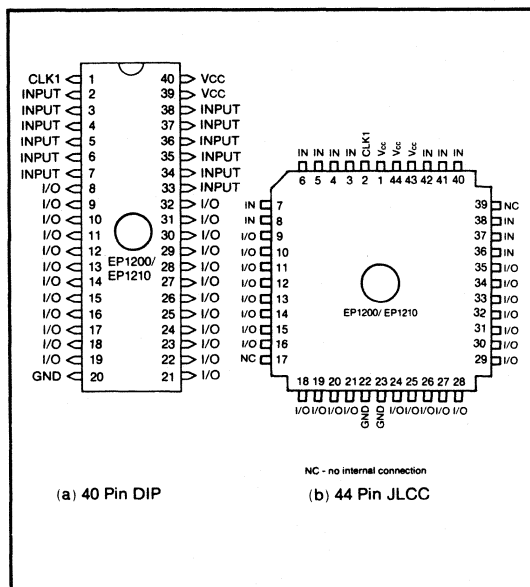
performance of the device. This option to disable the low standby power is controlled by TURBO-BIT™ EPROM locations within the device that can be set using LogicMap II. When the bits are programmed the low standby power mode is disabled. In this case the speed performance of the EP600 is typically accelerated by less than 5%. Performance data shown in the AC Characteristics section of this datasheet is for low power operation with the TURBO-BIT™ not programmed.



### FEATURES

- High Density (over 1200 gates) replacement for TTL and 74HC.
- Advanced CHMOS EPROM technology allows for erasability and reprogrammability.
- Low power: 15 mW typical standby power dissipation.
- Programmable Macrocell & I/O Architecture: up to 36 inputs or 24 outputs, 28 Macrocells including 4 buried state registers.
- Programmable latch feature allows latching of all inputs.
- Programmable clock system for input latches and output registers.
- Product term sharing and local bus architecture for optimized array performance.
- 100% generically testable — provides 100% programming yield.
- Programmable "Security Bit" allows total protection of proprietary designs.
- Advanced software support featuring Schematic Capture, Interactive Netlist, State Machine and Boolean Equation design entry methods.
- Package options include 40 pin DIP and 44 pin J-Leaded Chip Carrier.

### CONNECTION DIAGRAM



### GENERAL DESCRIPTION

The Altera EP1200/EP1210 is an LSI logic circuit that can be programmed to provide logic replacement for conventional SSI and MSI logic circuits.

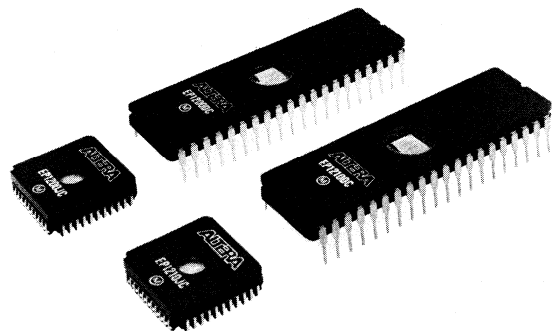
The EP1200/EP1210 contains CMOS EPROM (floating-gate) elements that control the logical operation of the device. The device can typically provide equivalent performance to 1200 gates of SSI and MSI logic. The EPROM technology enables the logic designer to rapidly program the device and make design changes after erasing for just a few minutes. The same technology also permits 100% factory testing of all elements within the device.

The CMOS technology reduces power consumption to less than 10% of equivalent bipolar devices without sacrificing speed performance.

To implement general purpose logic the EP1200/EP1210 contains the familiar sum-of-product PLA structure with a programmable AND and fixed OR array. The design uses a range of OR gate widths to accommodate logical functions without the overhead of unnecessary product-terms nor the speed penalties of programmable OR structures.

A segmented PLA design that provides local and global connectivity also optimizes the performance of the EP1200/EP1210.

The EP1200/EP1210 contains innovative architectural features that provide significant I/O flexibility and maximize performance within a conventional dual-in-line package or a J-leaded chip carrier package for increased footprint efficiency.



## FUNCTIONAL DESCRIPTION

The EP1200/EP1210 is an LSI erasable programmable logic device (EPLD) which uses EPROM technology to configure connections within a programmable logic array. The device has a programmable I/O architecture that provides options to change inputs, outputs and logical function of the device.

The internal architecture is based on 28 Macrocells each of which contains a PLA and a programmable I/O block that can be programmed to create many different logic structures. This powerful I/O architecture can be configured to support both active-high, active-low, 3-state, open-drain and bi-directional data ports or act as an input, all on a 4-bit wide basis.

All inputs to the circuit may be latched, including the 12 dedicated input pins.

The Macrocells share a common programmable clock system that controls clocking of all registers and input latches. The device contains 8 modes of clock operation that allow logic transitions to take place on either rising or falling edges of the clock signals.

The primary logic array of the EP1200/EP1210 is segmented into two symmetrical halves that communicate via global bus signals. The main arrays contain some 15104 programmable elements representing 236 product terms each containing 64 input signals.

Macrocells in each half of the circuit are grouped together for architecture programming. These banks of four Macrocells can be further programmed on an individual Macrocell basis to generate active high or active low outputs of the logic function from the PLA.

The circuit further contains four Macrocells whose outputs are only fed back into the array to create buried-state functions. The feedback path may be either the registered or combinatorial result of the PLA output. The use of buried state Macrocells provides maximum equivalent logic density without demanding higher pin-count packages which consume valuable board space.

## I/O ARCHITECTURE

The Input/Output architecture of the EP1200/EP1210 Macrocells can be programmed using both static and dynamic controls. The static controls remain fixed after the device is programmed whereas the dynamic controls may change state as a result of the signals applied to the device.

The static controls set the inversion logic, register by-pass and input feedback multiplexers. In the latter two cases these controls operate on four Macrocells as a bank. The buried-state registers have simpler controls which determine if the feedback is to be registered or combinatorial.

The dynamic controls consist of a programmable input latch-enable, as well as register clear and output-

enable product terms. The latch-enable function is common throughout the EP1200/EP1210 and is programmed by the clock control block but may also be driven by input signals applied to pin 1 (see clock modes Table 1). The register clear and output-enable controls are logically controlled by single product terms (the logic AND of programmed variables in the array). These terms have control over banks of four Macrocells.

The output-enable control may be used to generate architecture types that include bi-directional, 3-state, open-drain or input only structures.

## OUTPUT/FEEDBACK SELECTION

The EP1200/EP1210 Input/Output Architecture allows each group of Macrocells to be programmed for combinatorial or registered operation, with individual control over output polarity. In addition, the designer may configure the feedback path for combinatorial, registered, input (pin), and latched input feedback. All Macrocell groups have Asynchronous Clear control from a dedicated product term. When the product term is asserted to a logical "1", the registers within the respective Macrocell group will immediately be loaded with a logical "0" independently of the clock. On power up, the EP1200/EP1210 performs the Clear function automatically.

Figure 2 shows the basic output configurations for the EP1200/EP1210. In a combinatorial mode, the output is controlled via the group dedicated Output Enable product term. The Invert Select EPROM bit controls output polarity. The Feedback Select Multiplexer enables registered feedback, pin or latched pin feedback, or no feedback.

In a registered mode, 4 to 16 product terms are ORed together and made available to the D-type flipflop. The Output Enable product term allows registered or no output. The Invert Select EPROM bit determines output polarity. The Feedback Select Multiplexer can be configured for registered feedback, pin or latched pin feedback, or no feedback.

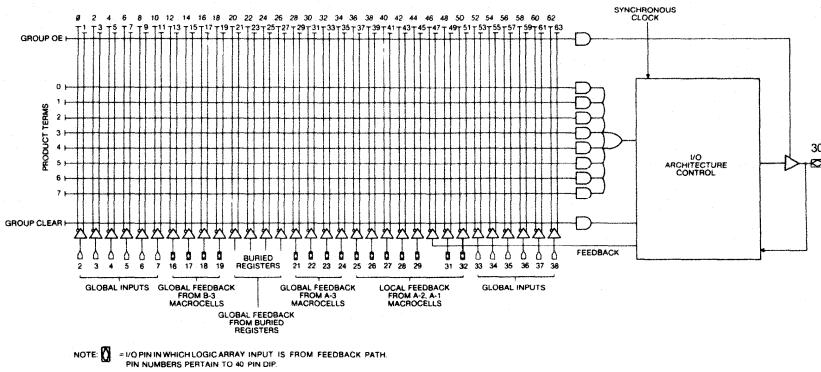
Any I/O group can be configured as a dedicated input group by selecting no output and pin feedback.

In the erased state, the EP1200/EP1210 I/O is configured for active low combinatorial output and latched pin feedback.

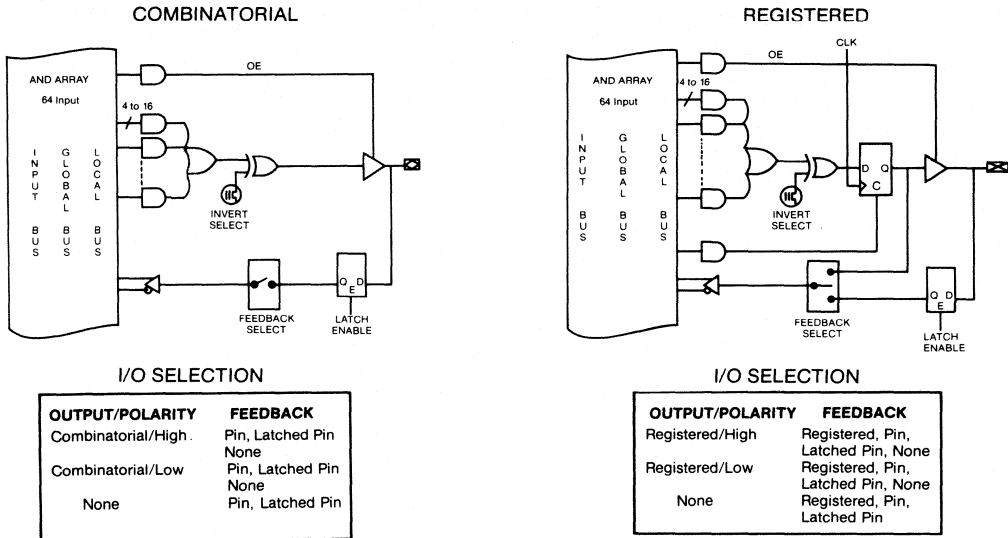
## SHARED PRODUCT TERMS

Macrocells 9, 11, 12, 17, 18, 19 and 20 have the facility to share a total of 16 additional product terms. The sharing takes place between pairs of adjacent macrocells. This capability enables, for example, Macrocells 9 and 10 to expand to 16 and 8 effective product terms respectively and for Macrocells 11 and 12 both to expand to 12 effective product terms. This facility is primarily of use in state machine and counter applications where common product-terms are frequently required among output functions.

**FIG. 1 LOGIC ARRAY MACROCELL  
(FOR OUTPUT TAKEN FROM "A" HALF ONLY)**

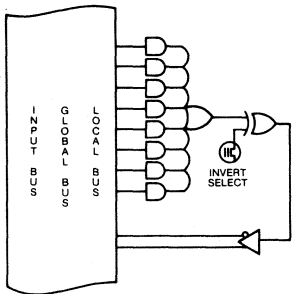


**FIG. 2 MACROCELL CONFIGURATIONS  
A. I/O MACROCELLS**

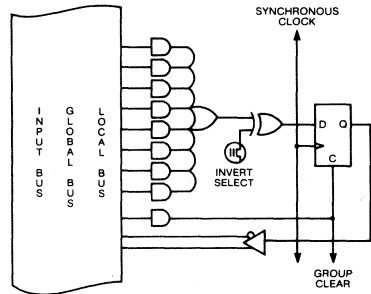


**B. BURIED MACROCELLS**

**NO OUTPUT, COMBINATORIAL FEEDBACK**



**NO OUTPUT, REGISTERED FEEDBACK**



**BUS STRUCTURE**

The two identical halves of the EP1200/EP1210 communicate via a series of busses. The local bus structure that is used for communication within each half of the chip contains 16 conductors that carry the TRUE and COMPLEMENT of 8 local Macrocells.

The global bus is comprised of 48 conductors that span the entire chip which carry the TRUE and COMPLEMENT of primary inputs (pins 2 through 7 and 33 through 38), signals from 4 Buried Registers, as well as the global outputs of 8 Macrocells in groups A-3 and B-3.

**MACRO — BUS INTERFACE**

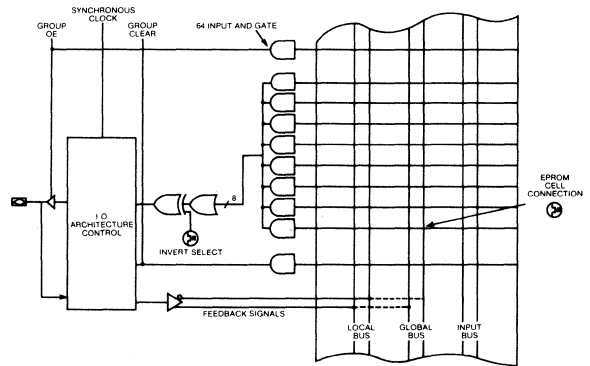
The Macrocells within an EP1200/EP1210 are interconnected to other Macrocells and inputs to the device via three internal data buses.

The product-terms span the entire bus structure that is adjacent to their Macrocell so that they may produce a logical AND of any of the variables (or their complements) that are present on the buses.

Macrocells all have the ability to return data to the local or global bus. Feedback data may originate from the output of the Macrocell or from the I/O pin. Feedback to the global bus communicates throughout the part. Macrocells that feedback to the local bus communicate to only half the EP1200/EP1210. Connections to and from the signal busses are made with

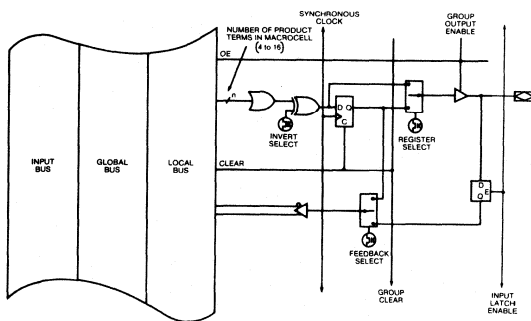
**FIG. 3 MACROCELL BUS STRUCTURE**

At each intersecting point in the logic array there exists an EPROM-type programmable connection. Initially, all connections are complete. This means that both the true and complement of all inputs are connected to each product-term. Connections are opened during the programming process. Therefore any product term can be connected to the true or complement of any input. When both the true and complement connections of any input are left intact, a logical false results on the output of the AND gate. If both the true and complement connections of any input are programmed open, then a logical "don't care" results for that input. If all inputs for a product term are programmed open, then a logical true results on the output of the AND gate.

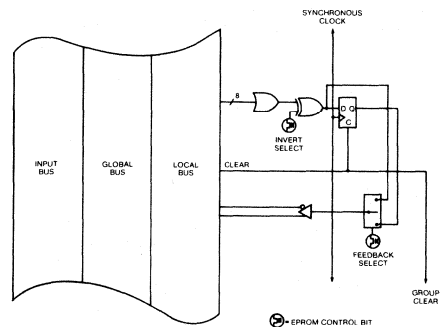


**FIG. 4**

**A. I/O MACROCELL**



**B. BURIED MACROCELL**



EPROM switches that provide the reprogrammable logic capability of the circuit.

Macrocells in groups A-3 and B-3 and the buried registers all have global bus connections while Macrocells in groups A-1, A-2, and B-1, B-2 have local bus connections. Figure 3 illustrates the local and global bus connections. Advanced features of the ALTERA development system will, if desired, automatically select an appropriate Macrocell to meet both the logic requirements and the connection to an appropriate signal bus to achieve the interconnection to other Macrocells.

### CLOCK MODE CONTROL















The EP1200/EP1210 contains two internal clock data paths that drive the input latches (transparent 7475 type) and the output registers. These clocks may be

programmed into one of eight operating modes. Input latches may be enabled on either the high or low level of CLK1 (pin1). Once latched, the input signal keeps its value until the next transition of the chosen clock. Output registers can be programmed to be positive or negative edge-triggered with respect to CLK1 or CLK2. Table 1 shows the operation of each programming mode.

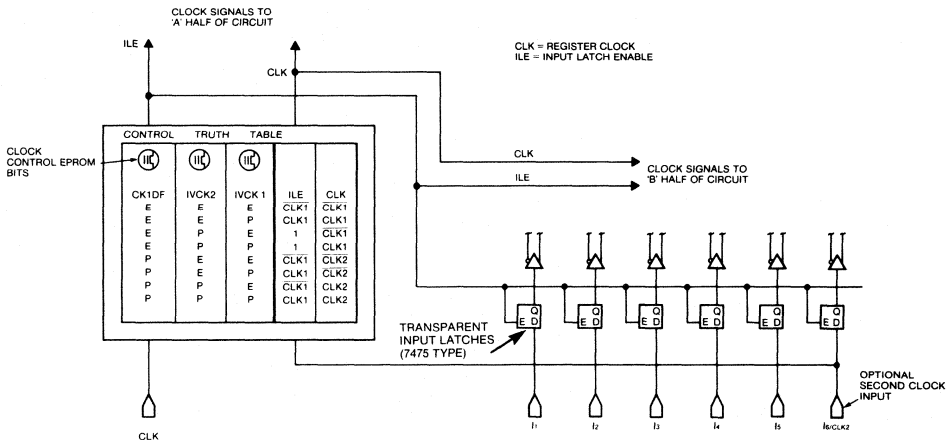
In the erased state, the EP1200/EP1210 clocking operation is set for mode 0. This means inputs are latched on a high level of CLK1. The high to low transition of CLK1 causes the input latches to become disabled, allowing input values to propagate into the logic array without being latched. In addition, CLK1 drives the output registers which are negative edge-triggered.

Care is required when using any of the two-clock modes to ensure that timing hazards are not created.

**TABLE 1 CLOCK PROGRAMMING**

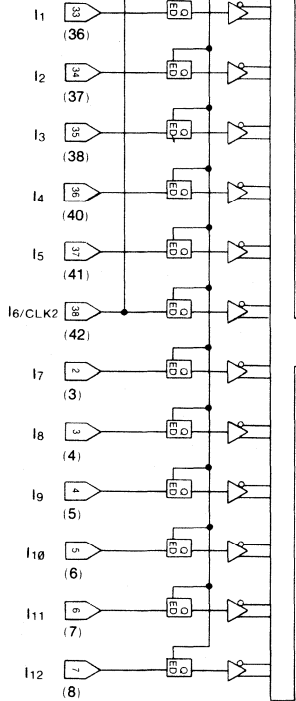
PROGRAMMED MODE	INPUT SIGNALS ARE PASSED (a) AND DATA IS LATCHED (b) WHEN:	OUTPUT REGISTERS CHANGE STATE WHEN:	CLOCK CONFIGURATION
0	CLK1 (PIN1) 	CLK1 (PIN1) 	1 CLOCK
1	CLK1 (PIN1) 	CLK1 (PIN1) 	1 CLOCK
2	INPUTS NOT LATCHED	CLK1 (PIN1) 	1 CLOCK
3	INPUTS NOT LATCHED	CLK1 (PIN1) 	1 CLOCK
4	CLK1 (PIN1) 	CLK2 (PIN38) 	2 CLOCKS
5	CLK1 (PIN1) 	CLK2 (PIN38) 	2 CLOCKS
6	CLK1 (PIN1) 	CLK2 (PIN38) 	2 CLOCKS
7	CLK1 (PIN1) 	CLK2 (PIN38) 	2 CLOCKS

**FIG. 5 PROGRAMMABLE CLOCK CONTROL SYSTEM**



MODE	CK1DF	IVCK2	IVCK1	ILE	CLK
0	E	E	E	CLK1	CLK1
1	E	E	P	CLK1	CLK1
2	E	P	E	1	CLK1
3	E	P	P	CLK1	CLK1
4	P	E	E	CLK1	CLK2
5	P	E	P	CLK1	CLK2
6	P	P	E	CLK1	CLK2
7	P	P	P	CLK1	CLK2

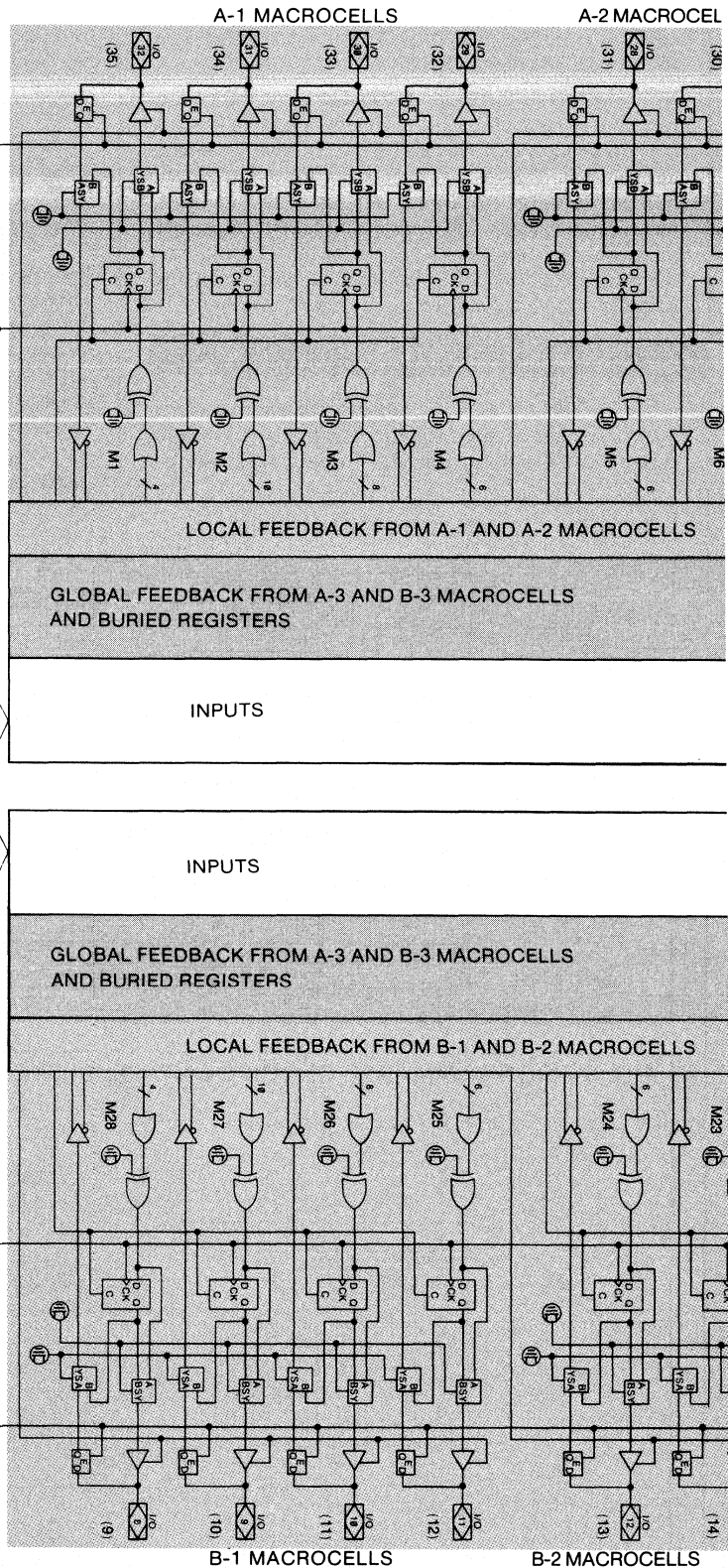
CLK1 (2) E Erased  
P Programmed



VCC (39)

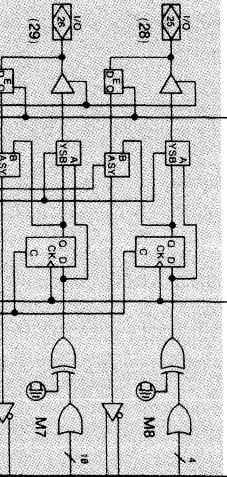
VCC/Vpp (44)

GND (22)

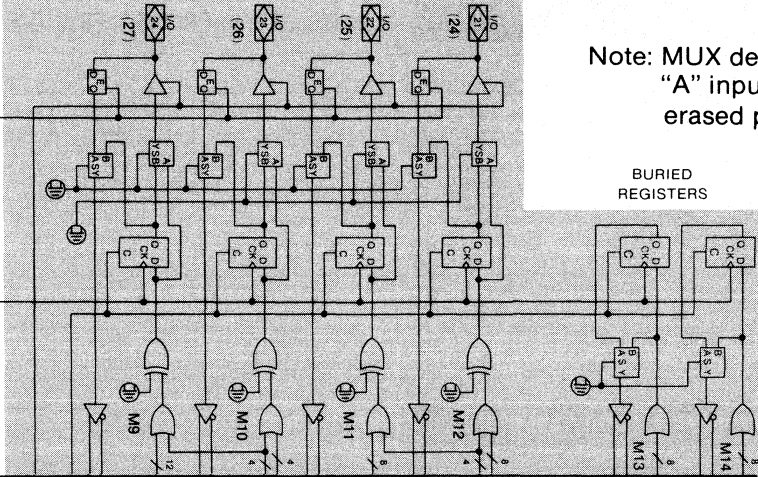


PIN NUMBERS IN ( ) PERTAIN TO 44 PIN JLC.

A-2 MACROCELLS

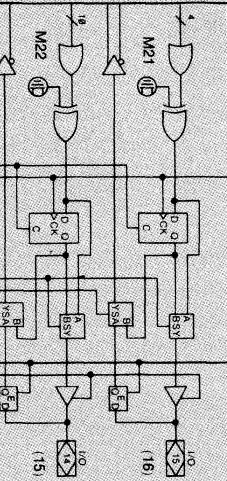
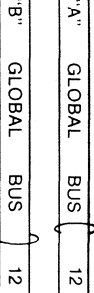
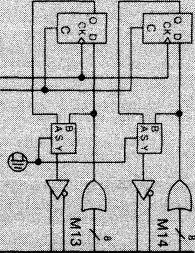


A-3 MACROCELLS

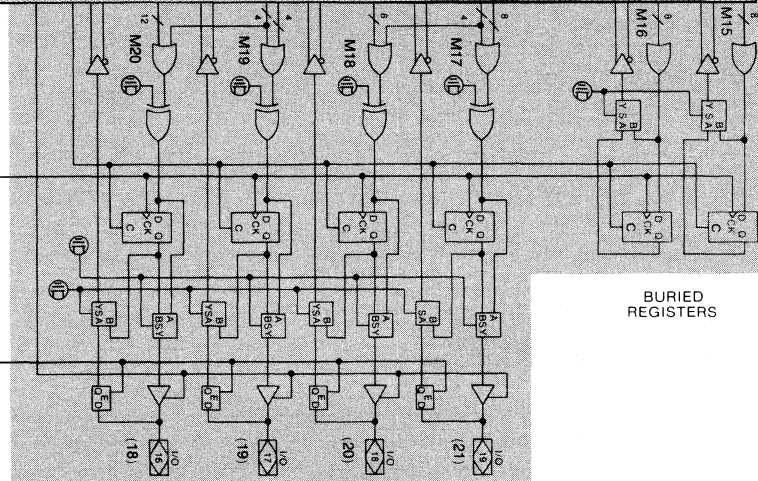


Note: MUX defaults to "A" input for an erased part.

BURIED REGISTERS

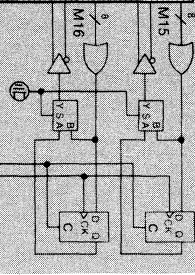


B-2 MACROCELLS



B-3 MACROCELLS

BURIED REGISTERS



## ABSOLUTE MAXIMUM RATINGS

## EP1200, EP1210, EP1210-1, EP1210-2, EP1210 JM

Note: See Design Recommendations

SYMBOL	PARAMETER	CONDITIONS	MIN	MAX	UNIT
V <sub>CC1200</sub>	EP1200 Supply Voltage	With respect to GND note (1)	-0.5	6.0	V
V <sub>CC1210</sub>	EP1210 Supply Voltage		-2.0	7.0	V
V <sub>PP1200</sub>	EP1200 Programming Supply Voltage		-0.5	21.5	V
V <sub>PP1210</sub>	EP1210 Programming Supply Voltage		-2.0	13.5	V
V <sub>I</sub>	DC Input Voltage		-0.5	V <sub>CC</sub> + 0.5	V
I <sub>CC MAX</sub>	DC V <sub>CC</sub> or GND current			+150	mA
I <sub>OUT</sub>	DC Output current, per pin		-25	+25	mA
P <sub>D</sub>	Power Dissipation			380	mW
T <sub>STG</sub>	Storage Temperature	No bias	-65	150	°C
T <sub>AMB</sub>	Ambient Temperature	Under bias note (2)	-10 (-65)	+85 (+135)	°C

## RECOMMENDED OPERATING CONDITIONS

SYMBOL	PARAMETER	CONDITIONS	MIN	MAX	UNIT
V <sub>CC</sub>	Supply Voltage		4.75	5.25	V
V <sub>I</sub>	Input Voltage		0	V <sub>CC</sub>	V
V <sub>O</sub>	Output Voltage		0	V <sub>CC</sub>	V
T <sub>A</sub>	Operating Temperature	note (2)	0 (-55)	70 (+125)	°C
t <sub>r</sub>	Input Rise time			500	ns
t <sub>f</sub>	Input Fall time			500	ns
t <sub>rVCC</sub>	V <sub>CC</sub> Rise time			10	ms

## DC OPERATING CHARACTERISTICS

T<sub>A</sub> = 0° to 70°C, V<sub>CC</sub> = 5.0V ±5%

SYMBOL	PARAMETER	CONDITIONS	MIN	TYP	MAX	UNIT
V <sub>IH</sub>	HIGH level input voltage		2.0		V <sub>CC</sub> +0.3	V
V <sub>IL</sub>	LOW level input voltage		-0.3		0.8	V
V <sub>O<sub>H</sub></sub>	HIGH level TTL output voltage	I <sub>OH</sub> = -4mA DC	2.4			V
V <sub>O<sub>H</sub></sub>	HIGH level CMOS output voltage	I <sub>OH</sub> = -2mA DC	3.84			V
V <sub>O<sub>L</sub></sub>	LOW level output voltage	I <sub>OL</sub> = 4mA DC			0.45	V
I <sub>I</sub>	Input leakage current	V <sub>I</sub> = V <sub>CC</sub> or GND note (2)			±10 (±20)	μA
I <sub>OZ</sub>	3-state output off-state current	V <sub>O</sub> = V <sub>CC</sub> or GND note (2)			±10 (±20)	μA
I <sub>CC1</sub>	V <sub>CC</sub> Supply Current (Standby - Low Power Mode)	V <sub>I</sub> = V <sub>CC</sub> or GND I <sub>O</sub> = 0 note (2,3)		3.0	6 (9)	mA
I <sub>CC2</sub>	V <sub>CC</sub> Supply Current (Active - Low Power Mode)	V <sub>I</sub> = V <sub>CC</sub> or GND No Load, f = 1 MHz note (2,3)		5.5	10 (13)	mA
I <sub>CC3</sub>	V <sub>CC</sub> Supply Current (Active - Turbo Mode)	V <sub>I</sub> = V <sub>CC</sub> or GND No Load, f = 1 MHz note (2,3)		35	65 (75)	mA

## CAPACITANCE

note (4)

SYMBOL	PARAMETER	CONDITIONS	MIN	MAX	UNIT
C <sub>IN</sub>	Input Capacitance	V <sub>IN</sub> = 0V, f = 1.0 MHz		30	pF
C <sub>OUT</sub>	Output Capacitance	V <sub>OUT</sub> = 0V, f = 1.0 MHz		40	pF
C <sub>CLK</sub>	Clock Pin Capacitance	V <sub>OUT</sub> = 0V, f = 1.0 MHz		30	pF



# AC CHARACTERISTICS

EP1200, EP1210, EP1210-1, EP1210-2:  $T_A = 0^\circ\text{C}$  to  $70^\circ\text{C}$ ,  $V_{CC} = 5V \pm 5\%$

EP1210JM:  $T_A = -55^\circ\text{C}$  to  $125^\circ\text{C}$ .

EP1200/EP1210

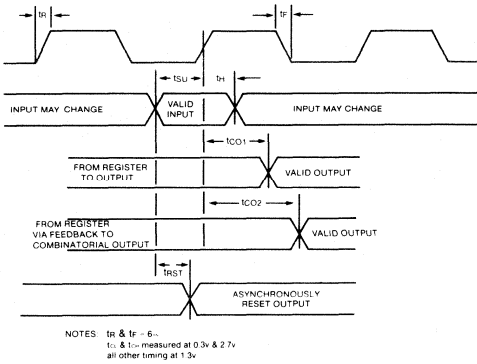
SYMBOL	PARAMETER	CONDITIONS	EP1210-1		EP1210-2		EP1210		EP1200		EP1210JM		UNIT
			MIN	MAX	MIN	MAX	MIN	MAX	MIN	MAX	MIN	MAX	
$t_{PD}$	Non-registered Input or I/O Input to non-registered output	note (5)		50		65		90		100		90	ns
$t_{PZX}$	Non-registered Input or I/O Input to output enable	$C_1 = 30\text{pF}$		50		65		90		100		90	ns
$t_{PXZ}$	Non-registered Input or I/O Input to output disable	$C_1 = 5\text{pF}$ note (6)		50		65		90		100		90	ns
$t_{SU}$	Non-registered Input or I/O Input to output register set-up		37		47		65		70		65		ns
$t_H$	Non-registered Input or I/O Input to output register hold		0		0		0		0		0		ns
$t_{CH}$	Clock high time		20		25		30		35		30		ns
$t_{CL}$	Clock low time	$C_1 = 30\text{pF}$	20		25		30		35		30		ns
$t_{CO1}$	Clock to output delay			30		36		45		40		45	ns
$t_{P1}$	Minimum clock period (register output feedback to register Input — Internal path)			50		58		75		84		75	ns
$f_1$	Maximum frequency ( $1/t_{P1}$ )		20		17		13		12		13		MHz
$t_{P2}$	Minimum clock period ( $t_{SU} + t_{CO1}$ )			67		83		110		110		110	ns
$f_2$	Maximum frequency ( $1/t_{P2}$ )		15		12		9		9		9		MHz
$t_{CLR}$	Asynchronous Clear time			65		75		100		110		100	ns
$t_{CO2}$	Registered feedback through PLA to output. Relative to external clock			65		75		100		110		100	ns
$t_{ILS}$	Set up time for latching inputs		0		0		0		0		0		ns
$t_{ILH}$	Hold time for latching inputs		15		20		25		30		25		ns
$t_{C1C2}$	Minimum clock 1 to Clock 2 delay			40		50		65		70		65	ns
$t_{LDFS}$	Input latch to D-FF setup time	Mode 0,1	40		50		65		70		65		ns
$t_{DFILS}$	D-FF to Input latch setup time		25		30		35		40		35		ns
$t_{P3}$	Minimum period for a 2-clock system ( $t_{C1C2} + t_{CO1}$ )			65		86		110		110		110	ns
$f_3$	Maximum frequency ( $1/t_{P3}$ )		15		11		9		9		9		MHz

**Notes:**

1. Minimum DC input is  $-0.3V$ . During transitions, the inputs may undershoot to  $-2.0V$  for periods less than 20 ns.
2. Figures in ( ) pertain to extended temperature version, EP1210JM.
3. Typical values measured with device programmed as a 26 bit counter.
4. Capacitance measured at  $25^\circ\text{C}$ . Sample tested only.
5. All AC values measured with Turbo bit programmed.
6. Sample tested only for an output change of 500 mV.

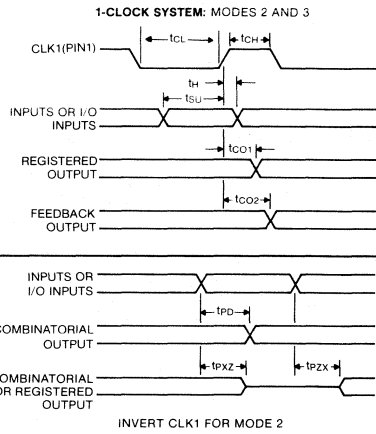
**FIG. 6 SWITCHING WAVEFORMS**

**A.**



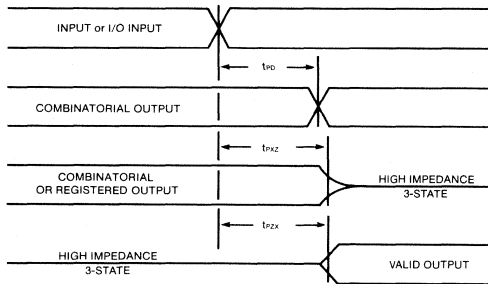
**FIG. 6 SWITCHING WAVEFORMS**

**B.**



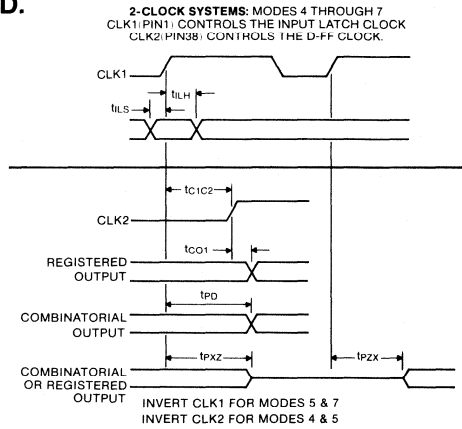
**FIG. 6 SWITCHING WAVEFORMS**

**C.**



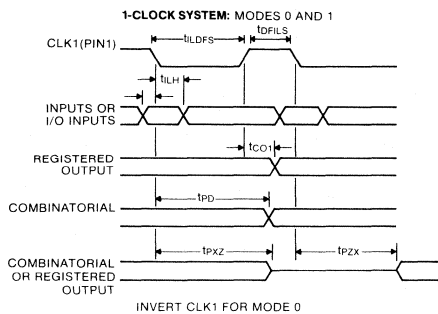
**FIG. 6 SWITCHING WAVEFORMS**

**D.**



**FIG. 6 SWITCHING WAVEFORMS**

**E.**



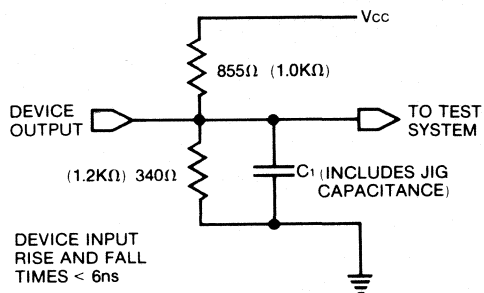
**PROGRAM ERASURE**

The erasure characteristics of the EP1200/EP1210 are such that erasure of the programmed connections begins to occur upon exposure to light with wavelengths shorter than approximately 4000 Angstroms. It is important to note that sunlight and certain fluorescent lighting could erase a programmed EP1200/EP1210 since they have wavelengths in the range of 3000 to 4000 Angstroms. Extrapolated results suggest that constant exposure to room level fluorescent lighting could erase an EP1200/EP1210 in approximately 3 years while it would take approximately 1 week to cause erasure when exposed to direct sunlight. As a consequence, if the EP1200/EP1210 is to be exposed to these types of lighting conditions for extended periods of time, opaque labels should be placed over the EP1200/EP1210 window to prevent unintentional erasure.

The recommended erasure procedure for the EP1200/EP1210 is exposure to shortwave ultraviolet light which has a wavelength of 2537 Angstroms. The integrated exposure dose for erasure should be a minimum of 15Wsec/cm<sup>2</sup>. The erasure time with this dosage is approximately 15 to 20 minutes using an ultraviolet lamp with a 12000 μW/cm<sup>2</sup> power rating. The EP1200/EP1210 should be placed within 1 inch of the lamp tubes during erasure. The maximum integrated exposure dose for an EP1200/EP1210 without damage is 7000 Wsec/cm<sup>2</sup>. This is approximately one week at 12000 μW/cm<sup>2</sup>. Exposure of the EP1200/EP1210 to high intensity UV light for long periods may cause permanent damage.

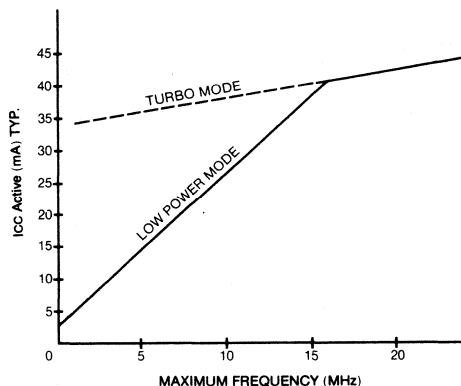
The EP1200/EP1210 may be erased and reprogrammed as many times as needed using the recommended erasure exposure levels.

**FIG. 7 AC TEST CONDITIONS**



( ) - CMOS Level Test Conditions

**FIG. 8 ICC vs. FMAX**



**FUNCTIONAL TESTING**

The EP1200/EP1210 is fully functionally tested and guaranteed through complete testing of each programmable EPROM bit and all internal logic elements thus ensuring 100% programming yield.

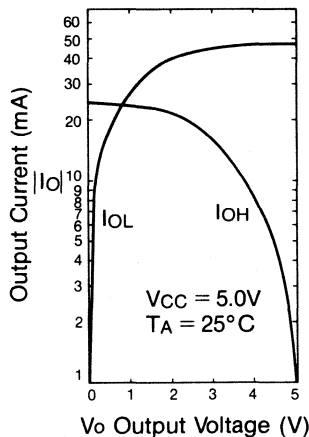
As a result, traditional problems associated with fuse-programmed circuits are eliminated. The erasable nature of the EP1200/EP1210 allows test program patterns to be used and then erased. This facility to use application-independent, general purpose tests is called generic testing and is unique among user-defined LSI logic devices.

To enable functional evaluation of counter and state-machine applications, the EP1200/EP1210 contains register pre-load circuitry. This can be activated by interrupting the normal clocked sequence and applying V<sub>HH</sub> on pin 2 (Pin 3 for 44 pin package) to engage the pre-load state. Under these conditions the flip-flops in the EP1200/EP1210 can be set to any logical condition and then return to normal operation. Presenting a "1" or "0" at the I/O pad will result in preloading a "1" or "0" (respectively) into the corresponding register. This process simplifies the input sequences necessary to evaluate counter and state-machine operations.

**DESIGN SECURITY**

The EP1200/EP1210 contains a programmable design security feature that controls the access to the data programmed into the device. If this programmable feature is used, a proprietary design implemented in the device cannot be copied nor retrieved. This enables a high level of design control to be obtained since programmed data within EPROM cells is invisible. The bit that controls this function, along with all other program data, may be reset simply by erasing the device.

**FIG. 9**



## LATCH-UP

The EP1200/EP1210 input, I/O, and clock pins have been carefully designed to resist latch-up which is inherent in CMOS structures. Each of the EP1200/EP1210 pins will not latch-up with currents up to 100 mA and voltages from -1V to  $V_{CC} + 1V$ . Additionally, the programming pin is designed to resist latch-up to the  $V_{PP} + 1V$  maximum device limit.

## DESIGN RECOMMENDATIONS

Operation of devices described herein with conditions above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this data sheet is not implied. Exposure to absolute maximum rating condition for extended periods may affect device reliability. These devices contain circuitry to protect the input against damage to high static voltages or electric fields; however, it is advised that normal precautions be taken to avoid application of any voltage higher than maximum rated voltages to this high-impedance circuit.

The typical  $I_{CC}$  versus frequency data shown in Fig. 8 is derived from the performance data of a 26-bit binary counter application. The EP1200/EP1210 contains a programmable option to control the autonomous power down feature that enables the low standby current performance of the device. This option to disable the low standby power is controlled by a TURBO-BIT™ EPROM location within the device that can be set using LogicMap II. When the bit is programmed the low standby power mode is disabled. In this case the speed

performance of the EP1200/EP1210 is typically accelerated by 10%. Performance data shown in the AC Characteristics section of this datasheet is for operation with the TURBO-BIT™ programmed.

For proper operation, it is recommended that input and output pins be constrained to the range  $GND \leq (V_{in} \text{ or } V_{out}) \leq V_{CC}$ . Unused inputs must always be tied to an appropriate logic level (e.g. either  $V_{CC}$  or GND). A power supply decoupling capacitor of at least  $0.2\mu F$  must be connected directly between  $V_{CC}$  pin and GND.

## PROGRAMMING DEVELOPMENT TOOLS

The EP1200/EP1210 is supported by an advanced programming development system that facilitates accurate and rapid product development.

The software system, known as A+PLUS, is the **Altera Programmable Logic User System** which supports multiple design entry techniques that include:

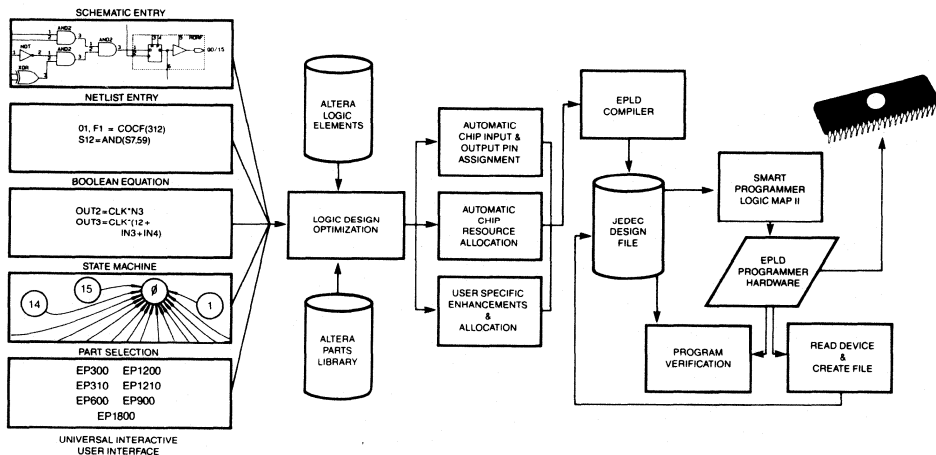
- Schematic diagram entry ... PC-CAPS or DASH-2
- Interactive netlist entry ..... NetMap
- Boolean equation entry ..... Altera Design File

The typical development environment used for this software would be an IBM Personal Computer and equivalent machines with the following configuration:

- Dual floppy disk drive or hard disk drive
- MS-DOS operating system version 2.0 or later release
- 512K memory
- Altera device programming card and unit

The output of A+PLUS is a data file in a standard JEDEC format. The EP1200/EP1210 can then be programmed using the Altera programming card.

**FIG. 10 ALTERA PROGRAMMABLE LOGIC USER SYSTEM**



#### FEATURES

- High density (over 900 gates) replacement for TTL and 74HC.
- Advanced CHMOS EPROM technology allows erasability and reprogrammability.
- High speed,  $t_{pd} = 30ns$ .
- "Zero Power" (typically  $10\mu A$  standby)
- **Asynchronous clocking of all registers or banked register operation from 2 synchronous clocks.**
- 24 Macrocells with configurable I/O architecture allowing 36 inputs and 24 outputs.
- **Programmable registers providing D, T, SR or JK flipflops with individual Asynchronous Clear control.**
- 100% generically testable—provides 100% programming yield.
- Programmable "Security Bit" allows total protection of proprietary designs.
- Advanced software support featuring Schematic Capture, Interactive Netlist, Boolean Equation and State Machine design entry methods.
- Package options include both a 40 pin, 600 mil DIP and a 44 pin J-leaded chip carrier.

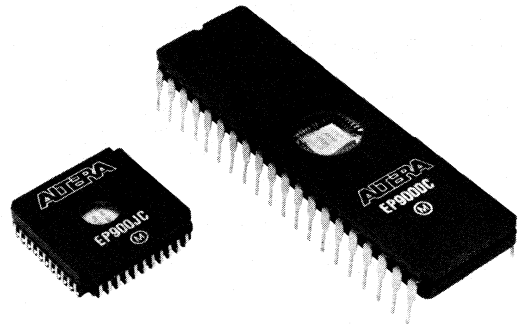
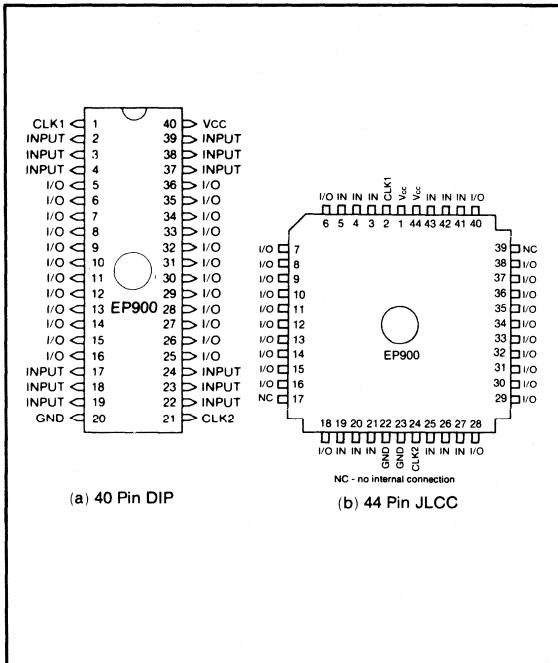
#### GENERAL DESCRIPTION

The ALTERA EP900 Erasable Programmable Logic Device may be used to implement over 900 equivalent gates of SSI and MSI logic, accommodating up to 36 inputs and 24 outputs all within a 40 pin DIP or 44 pin J-leaded chip carrier.

Each of the 24 Macrocells contains a programmable AND, fixed OR PLA structure which yields 8 product terms for logic implementation, and single product terms for Output Enable and Asynchronous Clear control functions.

The ALTERA proprietary programmable I/O architecture allows the EP900 user to program output and feedback paths for both combinatorial or registered operation, active high or active low.

#### CONNECTION DIAGRAM



#### PRELIMINARY DATA

NOTICE: THIS IS NOT A FINAL SPECIFICATION. SOME PARAMETRIC LIMITS ARE SUBJECT TO CHANGE.

For increased flexibility, the EP900 also includes programmable registers. Each of the 24 internal registers may be programmed to be a D, T, SR or JK flipflop. In addition, each register may be clocked asynchronously on an individual basis or synchronously on a banked register basis.

In addition to density and flexibility, the performance characteristics allow the EP900 to be used in the widest possible range of applications. The CHMOS EPROM technology reduces active power consumption to less than 20% of equivalent bipolar devices without a sacrifice in speed performance. This technology also facilitates 100% generic testability as well as UV erasability. As a result, designs and design modifications may be quickly implemented upon a given EP900 without the need for post programming testing.

Programming the EP900 is accomplished by using the ALTERA A+PLUS development software which supports schematic capture, netlist, state machine and Boolean equation design entry methods. Once the design is entered, A+PLUS automatically performs translation into logical equations, Boolean minimization, and design fitting directly to an EP900. The device may then be programmed to achieve customized working silicon within minutes at the designer's own desktop.

Figure 1 shows the basic EP900 Macrocell while figure 2 shows the complete EP900 block diagram. The internal architecture is organized with the familiar sum of products (AND-OR) structure. Inputs to the programmable AND array (running vertically in Figure 1) come from the true and complement forms of: 1) the 12 dedicated data inputs and; 2) the 24 feedback signals originating from each of the 24 I/O architecture control blocks. The 72 input AND array encompasses 240 product terms, distributed equally among the EP900's 24 Macrocells. Each product term (running horizontally in Figure 1) represents a 72 input AND gate.

At the intersection point between an AND array input and a product term is an EPROM control cell. In the erased state, all cell connections are made. This means both the true and complement of all array inputs are connected to each product term. During the programming process, selected connections are opened. Therefore, any product term may be connected to the true or complement of any array input signal. When both the true and complement of an array input signal is left connected, a logical false results on the output of the AND gate. If both the true and complement of any array input signal are programmed open, then a logical "don't care" results for that input. If all 72 inputs for a given product term are programmed open, then a logical true results on the output of the corresponding AND gate. Two dedicated clock inputs (these two clock signals are not available in the AND array) provide the clock signals used for synchronous clocking of the EP900 internal registers. Each of these two clock signals is positive edge triggered and has control over a bank of 12 registers. "CLK1" controls Macrocells 13-24, while "CLK2" controls Macrocells 1-12. The EP900 advanced I/O architecture allows any number of the 24 internal registers to be user-defined for synchronous or asynchronous clock modes.

**FUNCTIONAL DESCRIPTION**

The EP900 is an Erasable Programmable Logic Device (EPLD) in which CMOS EPROM technology is utilized in order to configure connections in a programmable AND logic array. EPROM connections are also used to construct a revolutionary programmable I/O architecture which provides advanced functional capability for user programmable logic.

Externally, the EP900 provides 12 dedicated data inputs, 2 synchronous clock inputs and 24 I/O pins which may be configured for input, output or bi-directional operation.

**FIG. 1 LOGIC ARRAY MACROCELL**

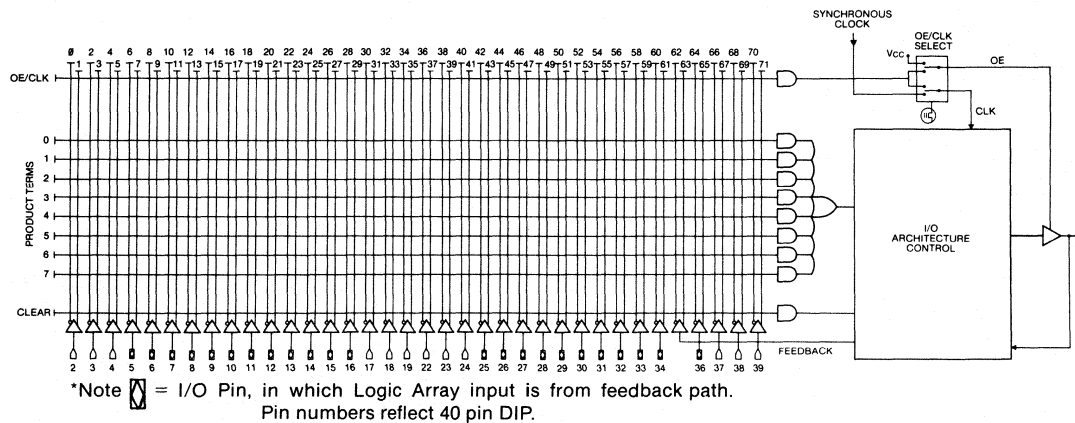
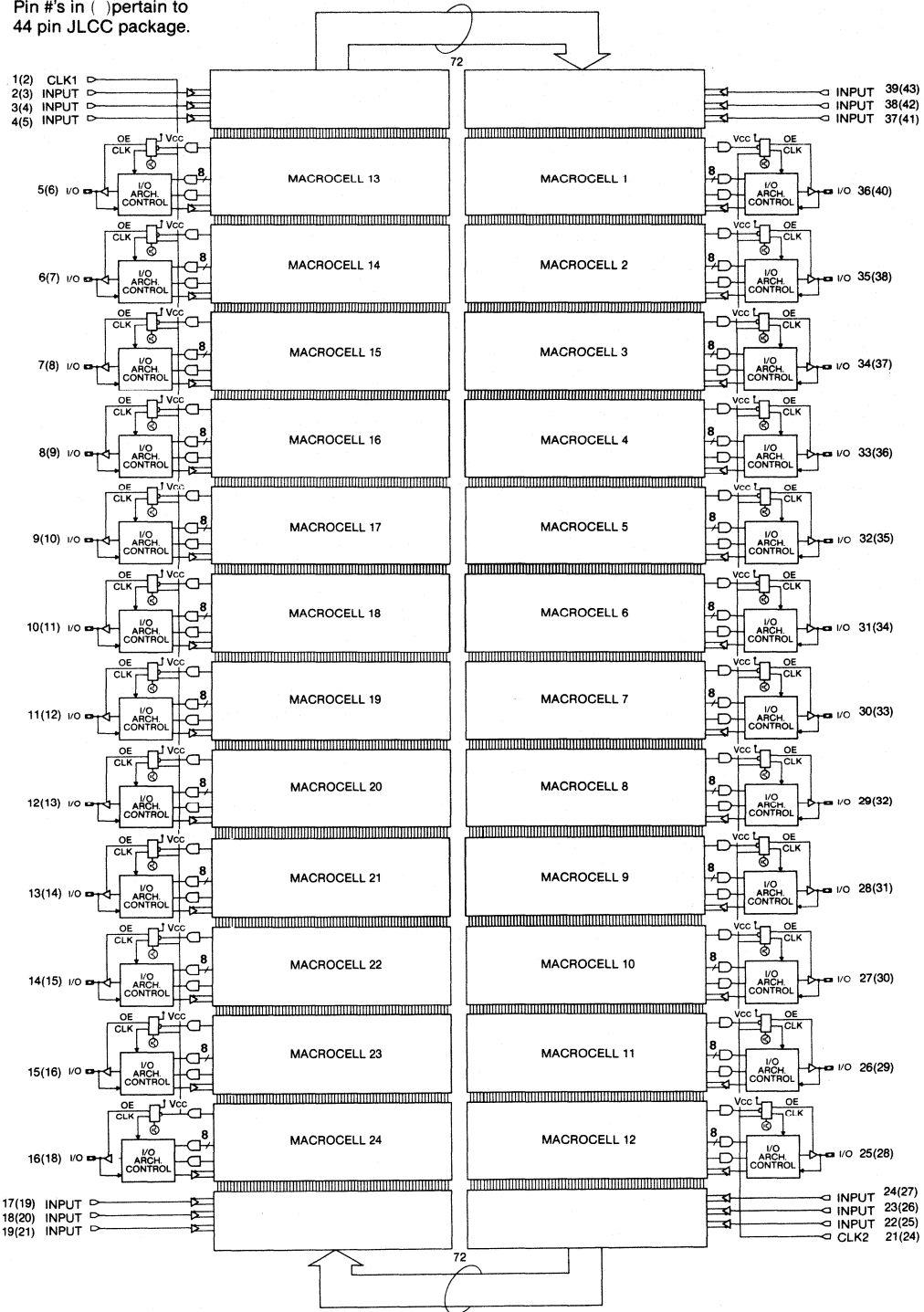


FIG. 2 EP900 BLOCK DIAGRAM

Pin #'s in ( ) pertain to 44 pin JLCC package.



## I/O ARCHITECTURE

The EP900 Input/Output Architecture provides each Macrocell with over 50 programmable I/O configurations. Each I/O can be configured for combinatorial or registered output, with programmable output polarity. Four different register types (D, T, JK, SR) may be implemented into every I/O without additional logic requirements. I/O feedback selection can also be programmed for registered or input (from the pin) feedback. Another characteristic of the EP900 I/O architecture is the ability to individually clock each internal register from asynchronous clock signals.

### OE/CLK Selection

Figure 3 shows the two modes of operation which are provided by the OE/CLK Select Multiplexer. The operation of this multiplexer is controlled by a single EPROM control bit and may be individually configured at each of the 24 I/O pins. In Mode 0, the three-state output buffer is controlled by the OE/CLK product term. (Recall that a single product term is equivalent to a 72 input AND gate.) If the output of the AND gate is a logical true, then the output buffer is enabled. If a logical false resides on the output of the AND gate, then the output buffer is seen as a high impedance node. In this mode the Macrocell flipflop is clocked by its respective synchronous clock input signal (CLK1 or CLK2). After erasure, the OE/CLK Select Mux is configured as Mode 0.

In Mode 1, the Output Enable buffer is tied to VCC (output is always enabled). The Macrocell flipflop may now be triggered from an asynchronous clock signal generated by the OE/CLK product term. This mode allows for individual clocking of flipflops from any of the 72 available AND array input signals. With both true and complement signals in the AND array, the flipflop may be configured to trigger on a rising or falling edge. In addition, this product term controlled clock config-

uration allows for the implementation of gated clock structures.

Figure 4 shows the basic output configurations available in the EP900. Along with combinatorial output, four register types are available. Each Macrocell may be individually configured. All registers have an individual Asynchronous Clear function which is controlled by a dedicated product term. When this product term yields a logical "1," the Macrocell register will immediately be loaded with a logical "0" independently of the clock. Upon power up of the EP900, the Clear function is performed automatically.

In the Combinatorial configuration, eight product terms are ORed together to acquire the output signal. The Invert Select EPROM bit controls output polarity and the Output Enable buffer is product term controlled. The Feedback Select Multiplexer allows the user to choose I/O (pin) feedback or no feedback to the AND array.

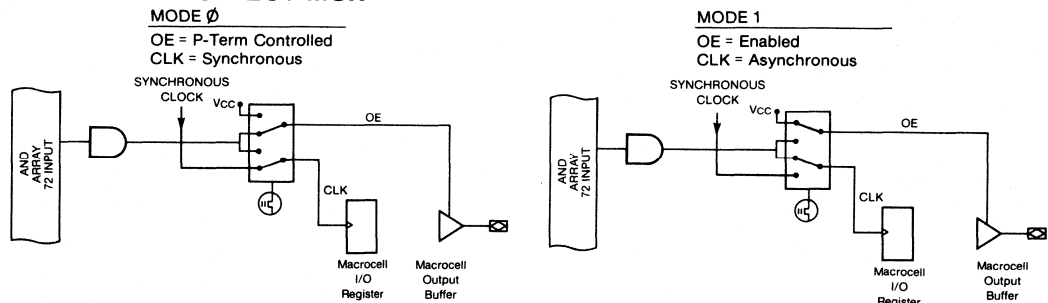
When the D or T register is selected, eight product terms are ORed together and made available to the register input. The Invert Select EPROM bit controls output polarity. The OE/CLK Select Multiplexer is used to configure the mode of operation (Mode 0 or Mode 1... see Figure 3). The Feedback Select Multiplexer allows the user to choose registered, I/O (pin) or no feedback to the AND array.

If the JK or SR register is selected, eight product terms are shared between two OR gates whose outputs feed the two primary register inputs. The allocation of product terms for each register input is optimized by the A+PLUS development software. The Invert Select EPROM bits control output polarity while the OE/CLK Select Multiplexer allows the mode of operation to be Mode 0 or Mode 1. The Feedback Select Multiplexer allows the user to choose registered or no feedback to the AND array.

Any I/O pin may be configured as a dedicated input by selecting no output with I/O (pin) feedback.

In the erased state, the I/O architecture is configured for combinatorial active low output with I/O (pin) feedback.

**FIG. 3 OE/CLK SELECT MUX**



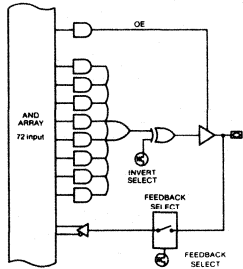
The register is clocked by the synchronous clock signal which is common to 11 other Macrocells. The output is enabled by the logic from the product term.

The output is permanently enabled and the register is clocked via the product term. This allows for gated clocks that may be generated from elsewhere in the EP900.



FIG. 4 I/O CONFIGURATIONS

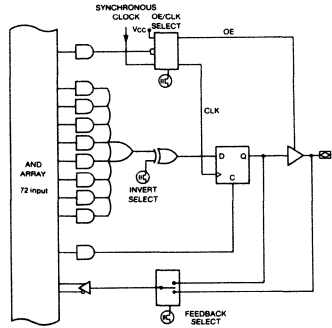
COMBINATORIAL



I/O SELECTION

OUTPUT/POLARITY	FEEDBACK
Combinatorial/High	Pin, None
Combinatorial/Low	Pin, None
None	Pin

D-TYPE FLIP-FLOP



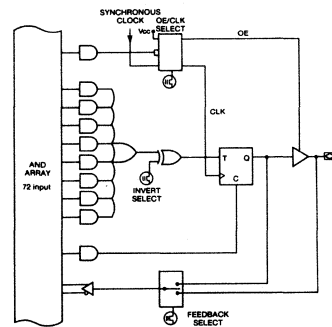
I/O SELECTION

OUTPUT/POLARITY	FEEDBACK
D-Register/High	D-Register, Pin, None
D-Register/Low	D-Register, Pin, None
None	D-Registered
None	Pin

FUNCTION TABLE

D	Q <sub>n</sub>	Q <sub>n+1</sub>
0	0	0
0	1	0
1	0	1
1	1	1

TOGGLE FLIP-FLOP



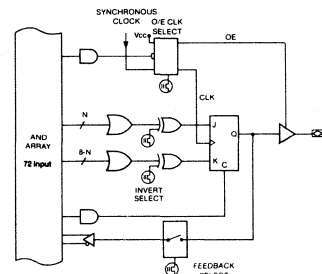
I/O SELECTION

OUTPUT/POLARITY	FEEDBACK
T-Register / High	T-Register, Pin, None
T-Register / Low	T-Register, Pin, None
None	T-Register
None	Pin

FUNCTION TABLE

T	Q <sub>n</sub>	Q <sub>n+1</sub>
0	0	0
0	1	1
1	0	1
1	1	0

JK FLIP-FLOP



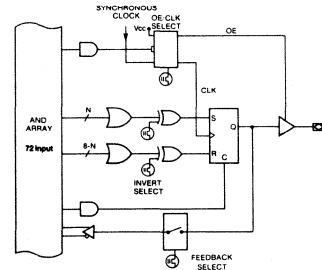
I/O SELECTION

OUTPUT/POLARITY	FEEDBACK
JK Register/High	JK Register, None
JK Register/Low	JK Register, None
None	JK Register

FUNCTION TABLE

J	K	Q <sub>n</sub>	Q <sub>n+1</sub>
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

SR FLIP-FLOP



I/O SELECTION

OUTPUT/POLARITY	FEEDBACK
SR Register/High	SR Register, None
SR Register/Low	SR Register, None
None	SR Register

FUNCTION TABLE

S	R	Q <sub>n</sub>	Q <sub>n+1</sub>
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1

**ABSOLUTE MAXIMUM RATINGS**

EP900, EP900-1, EP900-2, EP900-3, EP900 JM

Note: See Design Recommendations

SYMBOL	PARAMETER	CONDITIONS	MIN	MAX	UNIT
V <sub>CC</sub>	Supply voltage	With respect to GND note (3)	-2.0	7.0	V
V <sub>PP</sub>	Programming supply voltage		-2.0	13.5	V
V <sub>I</sub>	DC INPUT voltage		-2.0	7.0	V
I <sub>CCMAX</sub>	DC V <sub>CC</sub> or GND current			+150	mA
I <sub>OUT</sub>	DC OUTPUT Current, per pin		-25	+25	mA
P <sub>D</sub>	Power Dissipation			380	mW
T <sub>STG</sub>	Storage temperature	No bias	-65	+150	°C
T <sub>AMB</sub>	Ambient temperature	Under bias	-10	+85	°C

**RECOMMENDED OPERATING CONDITIONS**

SYMBOL	PARAMETER	CONDITIONS	MIN	MAX	UNIT
V <sub>CC</sub>	Supply voltage		4.75	5.25	V
V <sub>I</sub>	INPUT voltage		0	V <sub>CC</sub>	V
V <sub>O</sub>	OUTPUT voltage		0	V <sub>CC</sub>	V
T <sub>A</sub>	Operating temperature	For EP900,-1,-2,-3	0	70	°C
T <sub>A</sub>	Operating temperature	For EP900JM	-55	125	°C
T <sub>R</sub>	INPUT rise time			500	ns
T <sub>F</sub>	INPUT fall time			500	ns
T <sub>RVCC</sub>	V <sub>CC</sub> rise time			10	ms

**DC OPERATING CHARACTERISTICS**

Note (1)

SYMBOL	PARAMETER	CONDITIONS	MIN	TYP	MAX	UNIT
V <sub>IH</sub>	HIGH level input voltage		2.0		V <sub>CC</sub> +0.3	V
V <sub>IL</sub>	LOW level input voltage		-0.3		0.8	V
V <sub>OH</sub>	HIGH level TTL output voltage	I <sub>OH</sub> = -4.0mA DC	2.4			V
V <sub>OH</sub>	HIGH level CMOS output voltage	I <sub>OH</sub> = -2mA DC	3.84			V
V <sub>OL</sub>	LOW level TTL output voltage	I <sub>OL</sub> = 4.0mA DC			.45	V
I <sub>I</sub>	Input leakage current	V <sub>I</sub> = V <sub>CC</sub> or GND			±10.0	μA
I <sub>OZ</sub>	3-state output off-state current	V <sub>O</sub> = V <sub>CC</sub> or GND			±10.0	μA
I <sub>CC1</sub>	V <sub>CC</sub> supply current (standby)	V <sub>I</sub> = V <sub>CC</sub> or GND I <sub>O</sub> = 0		10	100	μA
I <sub>CC2</sub>	V <sub>CC</sub> supply current (active)	No load f = 1 MHz		3	6	mA

**CAPACITANCE**

Note (4)

SYMBOL	PARAMETER	CONDITIONS	MIN	MAX	UNIT
C <sub>IN</sub>	Input Capacitance	V <sub>IN</sub> = 0V f = 1.0 MHz		30	pF
C <sub>OUT</sub>	Output Capacitance	V <sub>OUT</sub> = 0V f = 1.0 MHz		30	pF
C <sub>CLK</sub>	Clock Pin Capacitance	V <sub>IN</sub> = 0V f = 1.0 MHz		30	pF

## AC CHARACTERISTICS

( $V_{CC} = 5V \pm 5\%$ ,  $T_A = 0^\circ\text{C}$  to  $70^\circ\text{C}$  for EP900,-1,-2,-3)  
 ( $V_{CC} = 5V \pm 5\%$ ,  $T_A = -55^\circ\text{C}$  to  $125^\circ\text{C}$  for EP900 JM)

See note (5)

SYMBOL	PARAMETER	CONDITIONS	EP900-1		EP900-2		EP900-3		EP900		EP900 JM		UNIT
			MIN	MAX	MIN	MAX	MIN	MAX	MIN	MAX	MIN	MAX	
$t_{PD}$	Input or I/O input to non-registered output			30		40		50		60		60	ns
$t_{PZX}$	Input or I/O input to output enable	$C_1 = 50\text{pF}$		30		40		50		60		60	ns
$t_{PXZ}$	Input or I/O input to output disable	$C_1 = 5\text{pF}$ note (2)		30		40		50		60		60	ns
$t_{CLR}$	Asynchronous output clear time	$C_1 = 50\text{pF}$		30		40		50		60		60	ns

## SYNCHRONOUS CLOCK MODE

SYMBOL	PARAMETER	CONDITIONS	EP900-1		EP900-2		EP900-3		EP900		EP900 JM		UNIT
			MIN	MAX	MIN	MAX	MIN	MAX	MIN	MAX	MIN	MAX	
$t_{SU}$	Input or I/O input setup time		23		33		35		40		40		ns
$t_H$	Input or I/O input hold time		0		0		0		0		0		ns
$t_{CH}$	Clock high time		15		20		25		25		25		ns
$t_{CL}$	Clock low time		15		20		25		25		25		ns
$t_{CO1}$	Clock to output delay			15		22		25		30		30	ns
$t_{P1}$	Minimum clock period (register output feedback to register input - internal path)			30		35		40		50		50	ns
$f_1$	Maximum frequency ( $1/t_{P1}$ )		33.3		28.6		25		20		20		MHz
$t_{P2}$	Minimum clock period ( $t_{SU} + t_{CO1}$ )			38		55		60		70		70	ns
$f_2$	Maximum frequency ( $1/t_{P2}$ )		26.3		18.2		16.7		14.3		14.3		MHz
$t_{CO2}$	Registered feedback through PLA to output. Relative to external clock			40		45		50		60		60	ns

## ASYNCHRONOUS CLOCK MODE

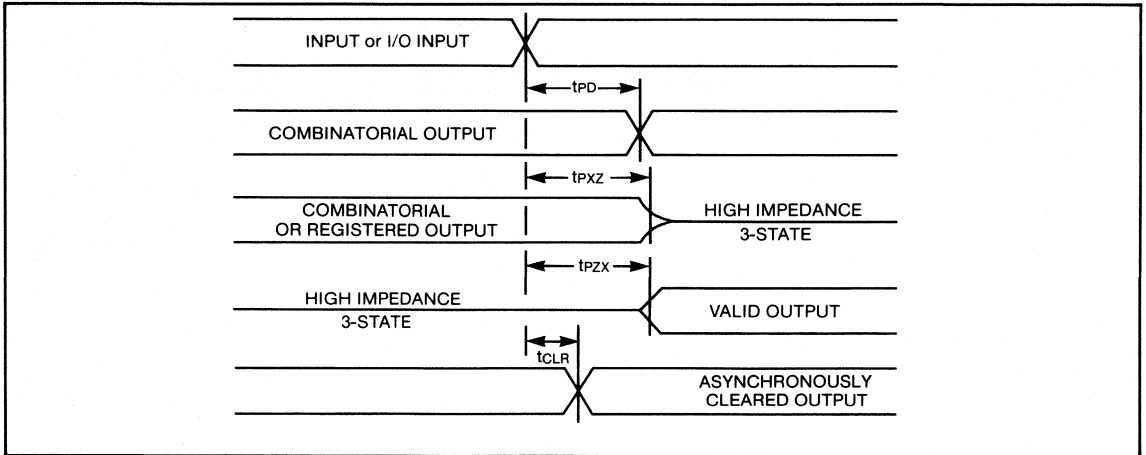
SYMBOL	PARAMETER	CONDITIONS	EP900-1		EP900-2		EP900-3		EP900		EP900 JM		UNIT
			MIN	MAX	MIN	MAX	MIN	MAX	MIN	MAX	MIN	MAX	
$t_{ASU}$	Input or I/O input setup time		5		5		5		5		5		ns
$t_{AH}$	Input or I/O input hold time		10		10		10		10		10		ns
$t_{ACH}$	Clock high time		15		20		25		25		25		ns
$t_{ACL}$	Clock low time		15		20		25		25		25		ns
$t_{AC01}$	Clock to output delay			35		45		55		65		65	ns
$t_{AP1}$	Minimum clock period (register output feedback to register input - internal path)			30		35		40		50		50	ns
$f_{A1}$	Maximum frequency ( $1/t_{AP1}$ )		33.3		28.6		25		20		20		MHz
$t_{AP2}$	Minimum clock period ( $t_{ASU} + t_{AC01}$ )			40		50		60		70		70	ns
$f_{A2}$	Maximum frequency ( $1/t_{AP2}$ )		25		20		16.7		14.3		14.3		MHz
$t_{AC02}$	Registered feedback through PLA to output. Relative to an asynchronous input pin			60		80		100		120		120	ns

## Notes:

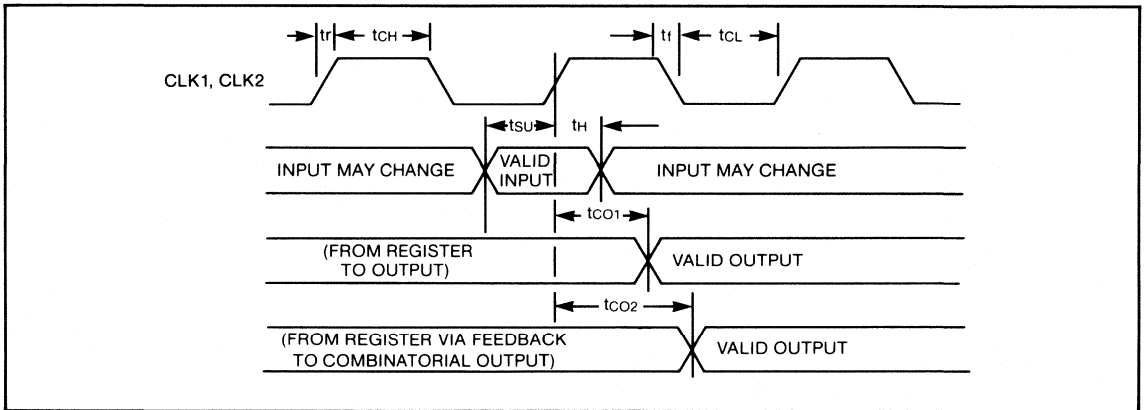
1. Typical values are for  $T_A = 25^\circ\text{C}$ ,  $V_{CC} = 5V$ .
2. Sample tested only for an output change of 500mV.
3. Minimum DC input is -0.3V. During transitions, the inputs may undershoot to -2.0V for periods less than 20ns.
4. Capacitance measured at  $25^\circ\text{C}$ . Sample tested only. Clock Pin Capacitance for dedicated clock inputs only.
5. All AC values tested with TURBO-BIT™ not programmed.

**FIG. 5 SWITCHING WAVEFORMS**

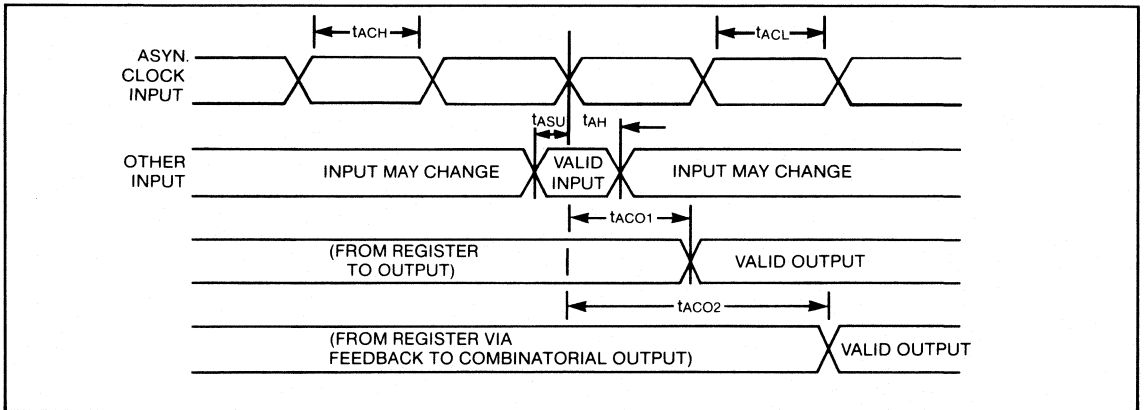
**COMBINATORIAL MODE**



**SYNCHRONOUS CLOCK MODE**



**ASYNCHRONOUS CLOCK MODE**



Notes:  $t_r$  &  $t_f = 6ns$   
 $t_{CL}$  &  $t_{CH}$  measured at 0.3V and 2.7V  
 all other timing at 1.3V  
 Input voltage levels at 0V and 3V

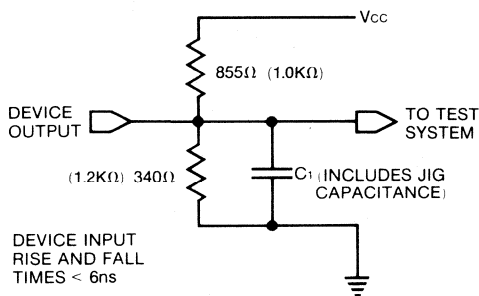
## PROGRAM ERASURE

The erasure characteristics of the EP900 are such that erasure of the programmed connections begins to occur upon exposure to light with wavelengths shorter than approximately 4000 Angstroms. It is important to note that sunlight and certain fluorescent lighting could erase a programmed EP900 since they have wavelengths in the range of 3000 to 4000 Angstroms. Extrapolated results suggest that constant exposure to room level fluorescent lighting could erase an EP900 in approximately 3 years while it would take approximately 1 week to cause erasure when exposed to direct sunlight. As a consequence, if the EP900 is to be exposed to these types of lighting conditions for extended periods of time, opaque labels should be placed over the EP900 window to prevent unintentional erasure.

The recommended erasure procedure for the EP900 is exposure to shortwave ultraviolet light which has a wavelength of 2537 Angstroms. The integrated exposure dose for erasure should be a minimum of 15Wsec/cm<sup>2</sup>. The erasure time with this dosage is approximately 15 to 20 minutes using an ultraviolet lamp with a 12000  $\mu$ W/cm<sup>2</sup> power rating. The EP900 should be placed within 1 inch of the lamp tubes during erasure. The maximum integrated exposure dose for an EP900 without damage is 7000 Wsec/cm<sup>2</sup>. This is approximately one week at 12000  $\mu$ W/cm<sup>2</sup>. Exposure of the EP900 to high intensity UV light for long periods may cause permanent damage.

The EP900 may be erased and re-programmed as many times as needed using the recommended erasure exposure levels.

**FIG. 6 AC TEST CONDITIONS**



( ) - CMOS Level Test Conditions

## FUNCTIONAL TESTING

The EP900 is fully functionally tested and guaranteed through complete testing of each programmable EPROM bit and all internal logic elements thus ensuring 100% programming yield.

As a result, traditional problems associated with fuse-programmed circuits are eliminated. The erasable nature of the EP900 allows test programming patterns to be used and then erased. This facility to use application-independent, general purpose tests is called generic testing and is unique among user-defined LSI logic devices.

To enable functional evaluation of counter and state-machine applications, the EP900 contains register pre-load circuitry. This can be activated by interrupting the normal clocked sequence and applying V<sub>HH</sub> on pin 17 (see note below) to engage the pre-load state. Under these conditions the flip-flops in the EP900 can be set to any logical condition and then return to normal operation. This process simplifies the input sequences necessary to evaluate counter and state-machine operations. If the inverted output path is selected, (active low output) a "1" at the I/O pad will result in preloading a "1" into the register. If an active high output path is selected, then a "1" on the I/O pad will result in preloading a "0" into the register. The preload waveforms are shown in Figure 7.

## DESIGN SECURITY

The EP900 contains a programmable design security feature that controls access to the data programmed into the device. If this programmable feature is used, a proprietary design implemented in the device cannot be copied nor retrieved. This enables a high level of design control to be obtained since programmed data within EPROM cells is invisible. The bit that controls this function, along with all other program data, may be reset simply by erasing the device.

## LATCH-UP

The EP900 input, I/O, and clock pins have been carefully designed to resist latch-up which is inherent in CMOS structures. Each of the EP900 pins will not latch-up with currents up to 100 mA and voltages from -1V to VCC+1V. Additionally, the programming pin is designed to resist latch-up to the 13.5 volt maximum device limit.

Note: Pin 17 engages preload state for EP900 in 40 pin DIP. For the 44 pin JLCC versions, use Pin 19.

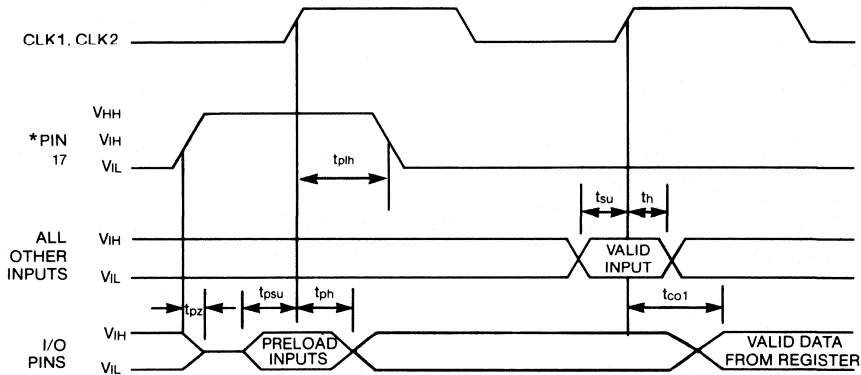
**PRECONDITIONING MODE**

**A.C. CHARACTERISTICS**

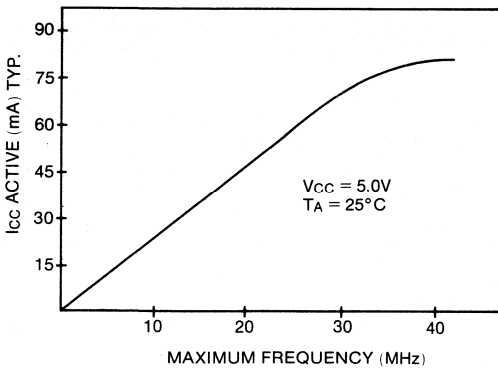
( $T_A = 0^\circ$  to  $70^\circ\text{C}$ ,  $V_{CC} = 5V \pm 5\%$ ,  $V_{HH} = 12V \pm 0.5V$ )

SYMBOL	PARAMETER	CONDITIONS	MIN	TYP	MAX	UNIT
$t_{pz}$	Output 3-state delay time after assertion of Preload (Pin 17 = $V_{HH}$ )*			100		ns
$t_{ph}$	Hold time of all preload inputs, with respect to Clock rising edge			15		ns
$t_{psu}$	Setup time of all preload inputs, with respect to Clock rising edge			100		ns
$t_{plh}$	Hold time for Preconditioning input			50		ns
$t_{co1}$	Output delay time (Register to output) after Clock rising edge			50		ns
$t_{su}$	Setup time for all other inputs with respect to Clock rising edge			50		ns
$t_h$	Hold time for all other inputs with respect to Clock rising edge			0		ns

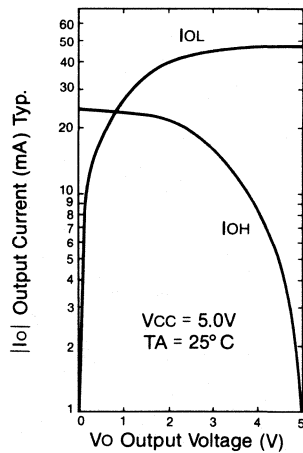
**FIG. 7 PRECONDITIONING WAVEFORM**



**FIG. 8 ICC vs. FMAX**



**FIG. 9 OUTPUT DRIVE CURRENTS**



\*Note: Pin 17 applies to 40 pin DIP EP900.  
For 44 pin JLCC EP900, Pin 19 =  $V_{HH}$ .

## PROGRAMMING DEVELOPMENT TOOLS

The EP900 is supported by an advanced programming development system that facilitates accurate and rapid product development.

The software system, known as A+PLUS, is the **Altera Programmable Logic User System** which supports multiple design entry techniques that include:

- Schematic diagram entry .... PCCAPS or DASH 2
- Interactive netlist entry ..... NetMap
- Boolean equation entry ..... Altera Design File

The typical development environment used for this software would be an IBM Personal Computer and equivalent machines with the following configuration:

- Dual floppy disk drive or hard disk drive
- MS-DOS operating system version 2.0 or later release
- 512K memory
- Altera device programming card and unit

The output of A+PLUS is a data file in a standard JEDEC format. The EP900 can then be programmed using the Altera programming card.

All of the EP900 architecture features are supported by Altera symbol primitives. The symbol library interfaces with each of the A+PLUS design entry methods. These primitives are in the form of Inputs, Basic Logic Gates, Equations, and I/O architectures.

When using the primitive symbols, the desired output polarity is obtained by appropriately connecting inverters to the inputs of the I/O resources. Figure 11 shows how to achieve active low outputs. For combinatorial output or D flipflops, the primitive input is driven by an inverter. The feedback (if used) also becomes active low. By placing another inverter in the feedback

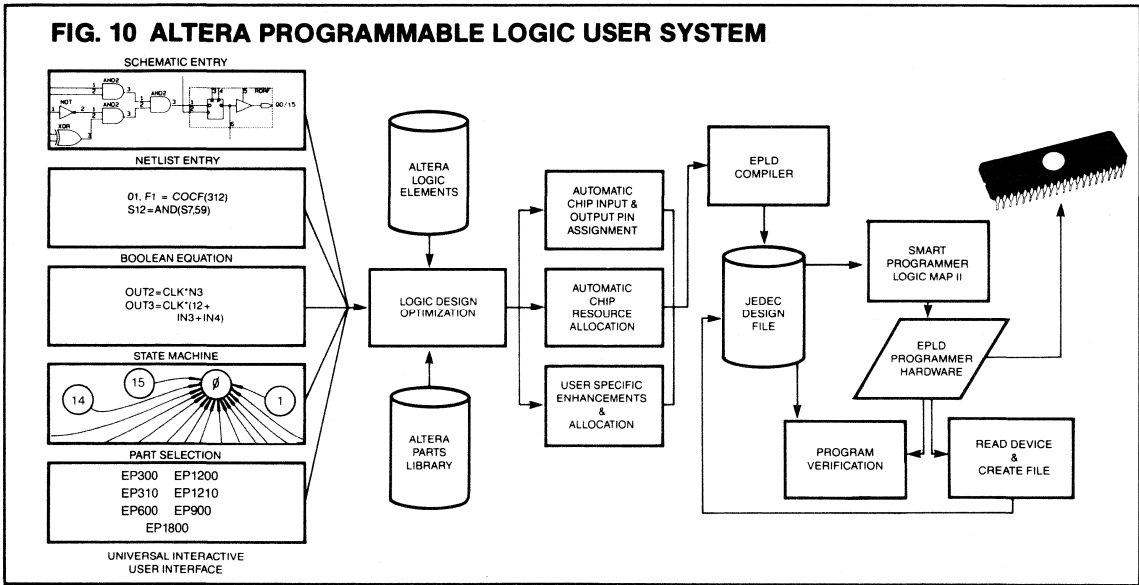
path an active high signal is obtained. For the JK or SR flipflops, simply reverse the input connections. In this case, a Set now becomes a Reset and a Reset now becomes a Set thus achieving an active low output. After the EP900 powers up, the JK or SR flipflop output must first be preset to a logical one for proper active low operation.

To specify asynchronous clock operation, the CLK input into a flipflop must be driven by the Asynchronous Clock Buffer primitive. Figure 12 illustrates asynchronous clock connections.

## DESIGN RECOMMENDATIONS

Operation of devices described herein with conditions above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this data sheet is not implied. Exposure to absolute maximum rating condition for extended periods may affect device reliability. These devices contain circuitry to protect the input against damage to high static voltages or electric fields; however, it is advised that normal precautions be taken to avoid application of any voltage higher than maximum rated voltages to this high-impedance circuit.

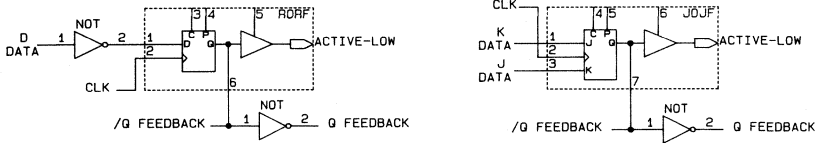
For proper operation, it is recommended that input and output pins be constrained to the range  $GND \leq (V_{in} \text{ or } V_{out}) \leq V_{CC}$ . Unused inputs must always be tied to an appropriate logic level (e.g. either VCC or GND). A power supply decoupling capacitor of at least 0.2μF must be connected directly between VCC pin and GND.



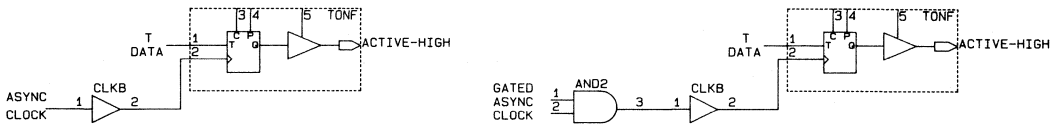
The typical I<sub>CC</sub> versus frequency data shown in Fig. 8 is derived from the performance data of a 24-bit binary counter application. The EP900 contains a programmable option to control the autonomous power down feature that enables the low standby current performance of the device. This option to disable the low standby power is controlled by TURBO-BIT™ EPROM locations within the device that can be set

using LogicMap II. When the bits are programmed the low standby power mode is disabled. In this case the speed performance of the EP900 is typically accelerated by less than 5%. Performance data shown in the AC Characteristics section of this datasheet is for low power operation with the TURBO-BIT™ not programmed.

**FIG. 11 ACTIVE-LOW OUTPUTS**



**FIG. 12 ASYNCHRONOUS CLOCKING**

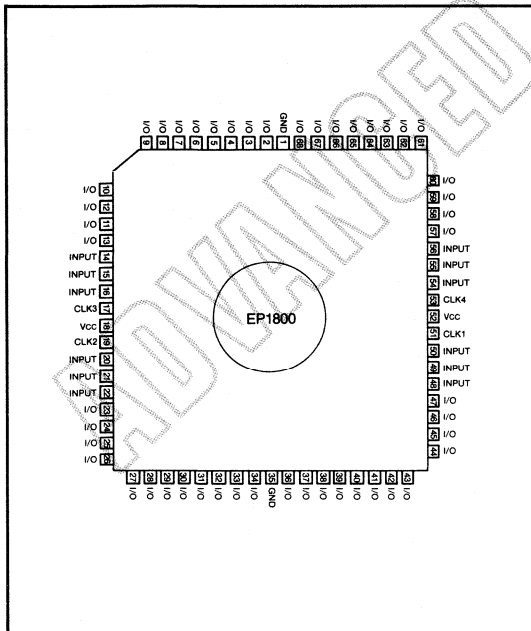




#### FEATURES

- Erasable, User Programmable LSI circuit capable of implementing 1800 equivalent gates of conventional and custom logic.
- "Zero Power" (typically 10 uA standby).
- Forty-eight Macrocells with configurable I/O architecture allowing 64 inputs or 48 outputs.
- **Individual clocking of all registers or banked register operation from 4 synchronous clocks.**
- Programmable registers providing D, T, JK, or SR flipflops with individual Reset control.
- **Dual Feedback signals allowing I/O pins to be used for buried logic and dedicated input.**
- 100% generically testable — provides 100% programmable yield.
- Advanced software support featuring 4 different design entry methods, logic minimization, and design fitting.
- **Advanced CHMOS\* II-E circuitry for systems requiring low power, high performance speeds, and immunity to noise.**
- Packaged in a 68 pin J-Lead Chip Carrier (0.96 Sq in) providing maximum foot-print efficiency.

#### CONNECTION DIAGRAM

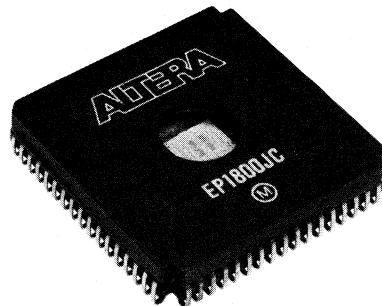


#### GENERAL DESCRIPTION

The ALTERA EP1800 is a CMOS LSI logic device using EPROM technology to implement custom logic functions in minutes, with no penalty for error. The EP1800 is the largest member of Altera's initial family of Erasable Programmable Logic Devices (EPLDs).

The EP1800 provides 12 dedicated inputs, 48 programmable I/O pins, and 4 synchronous clock inputs to accommodate a variety of independent logic functions. Internally, the device uses familiar sum-of-products logic providing a programmable AND, fixed OR structure. The EP1800 houses 48 Macrocells each containing 10 product terms and an I/O select block. Eight product terms are used for logic requirements, one product term for Output Enable/Asynchronous Clock implementation, and the remaining product term for flipflop Reset control. Each I/O can be individually configured for combinatorial or sequential logic functions.

The EP1800 is portioned into four quadrants, each containing 12 Macrocells and an 88 input AND array. Each of the EP1800 48 registers may be programmed for D, T, JK, or SR operation. Sixteen Macrocells provide two independent feedback paths allowing the Macrocell to be used as an internal register and dedicated input pin. Each flipflop may be clocked from one of the four dedicated synchronous clocks or may be independently clocked from a Macrocell product term.



#### ADVANCED INFORMATION

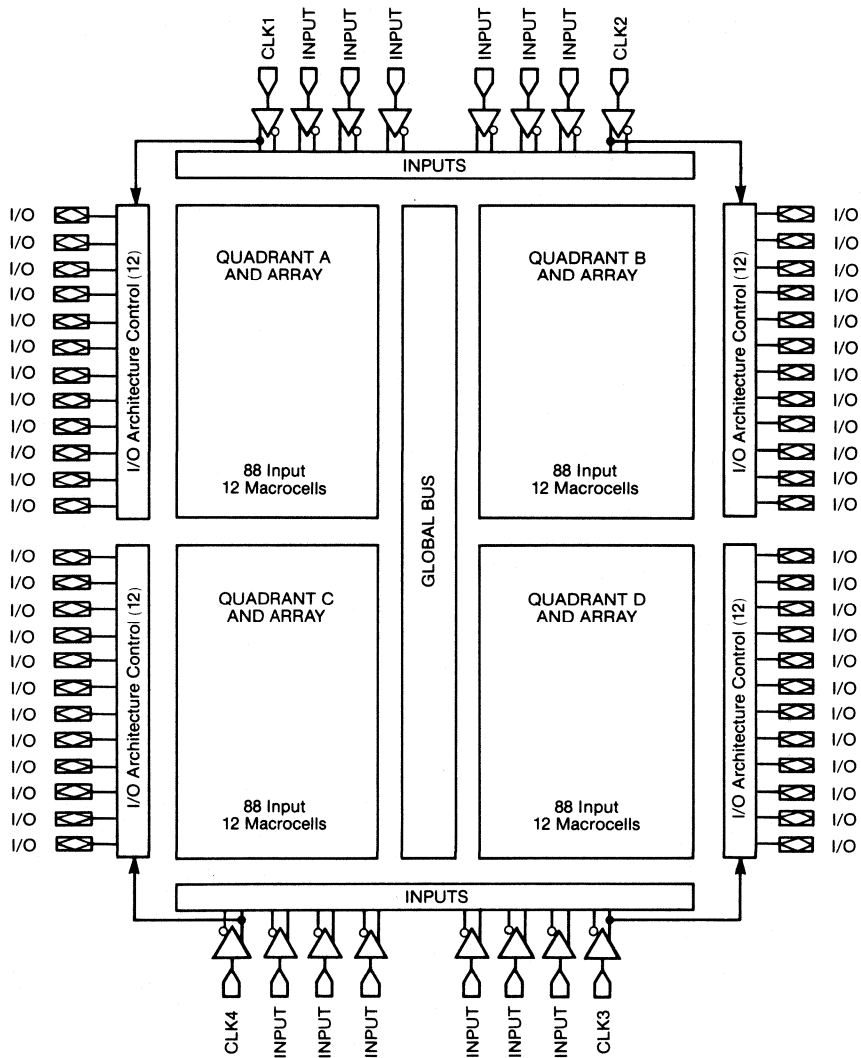
\*CHMOS is a trademark and patented process of Intel Corporation.

REV. 1.0

The CHMOS EPROM technology reduces the power consumption to less than 20% of equivalent bipolar devices without sacrificing speed performance. Other advantages include: 100% generic testing (all devices are 100% tested at the factory). The device can be erased with ultraviolet light. Design changes are no longer costly or time consuming, nor is there the need for post programming testing.

Programming the EP1800 is accomplished with the use of Altera's A+PLUS development software and hardware. Once a circuit design has been entered by any of four design entry methods, A+PLUS performs automatic translation to logical equations, boolean minimization, and design fitting directly into an EP1800.

EP1800 BLOCK DIAGRAM



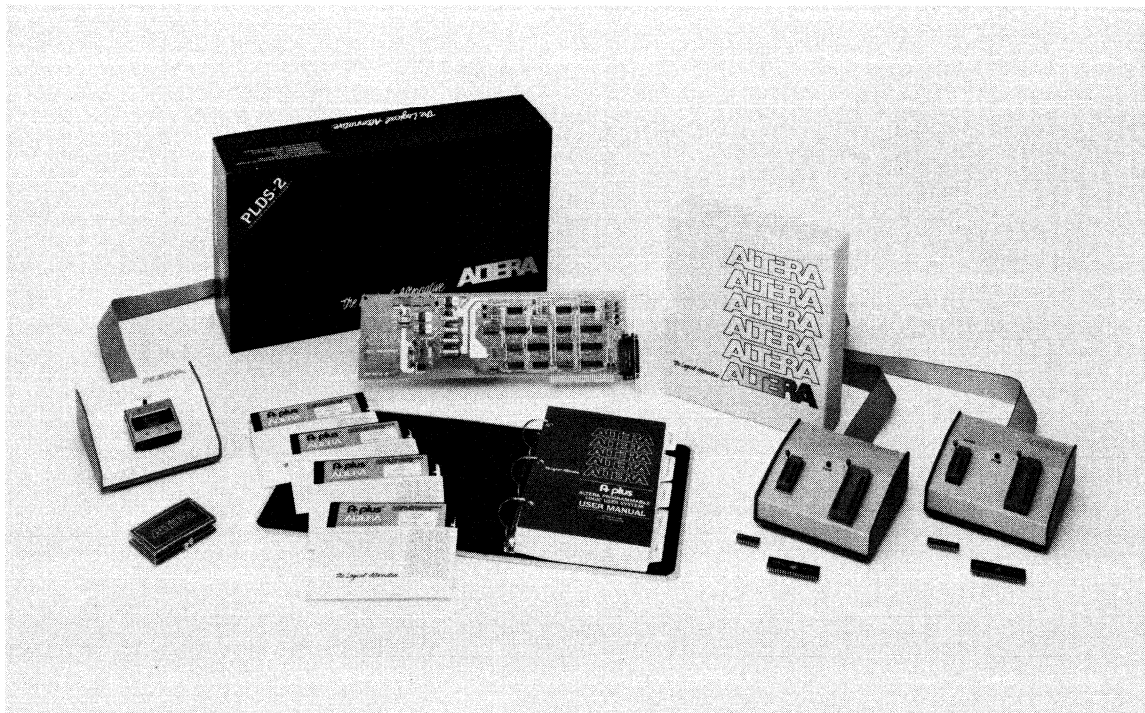
NOTE: See EP900 DATA SHEET for Macrocell and I/O Configurations

### FEATURES

- Development system for Altera EPLDs
- Hardware for device programming included
- Sample EPLDs included for immediate evaluation
- Boolean equation design entry
- Interactive netlist entry
- Automatic pin assignments
- Boolean equation minimization
- Sophisticated code optimizes logic design
- Optional schematic design interfaces
- Optional state-machine design interface
- Runs on PC compatible computers

### GENERAL DESCRIPTION

The Altera PLDS2 is a complete hardware and software development system that enables the circuit designer to develop optimized code used to program the target ALTERA EPLDs. The system allows a wide variety of design input techniques to be used that suit the particular logic design task. These include schematic capture, netlist entry, Boolean equation entry and state machine entry. The designer is not restricted to just one entry method but may 'mix-and-match' methods to best meet the needs of the overall logic design. The system contains an expansion card for PC compatible computers with cable connections to an external programming unit that is used for device programming. The programming hardware is fully software controlled via the LogicMap interface program. The design processing software includes options that will minimize the logic implementation, select pin assignments automatically and configure the best architecture for the required logic function.



## Design Processor

The A+PLUS design processor translates design information from a variety of sources into programming object code for the EPLD. The design processor has the ability to work with multiple netlists or Boolean equations in the Altera Design File format. Interfaces are available that connect to popular schematic capture packages such as PC-CAPs and DASH-2 (see PLCAD1 and PLCAD2 for details of these options).

Logic minimization of designs is provided by the processor in several phases. These include Boolean minimization using a sophisticated generalized consensus algorithm which yields superior results to other heuristic reduction techniques. De Morgan's theorem inversion can be applied automatically to equations by the design processor. The processor contains algorithms based on artificial intelligence techniques to select candidate equations that will best be represented by a complemented AND-OR function. This significantly reduces product-term demands that can be generated from complex logic designs. The Altera EPLDs that contain programmable flip-flops are supported by algorithms that further optimize the architectural configuration to best suit the logic function designed.

The design processor is capable of working with all Altera devices and permits rapid evolution of designs from low-level integration to high-integration EPLDs with minimal design changes of the source design data. This is achieved using iterative design fitting algorithms to enable the basic design to be transported from one

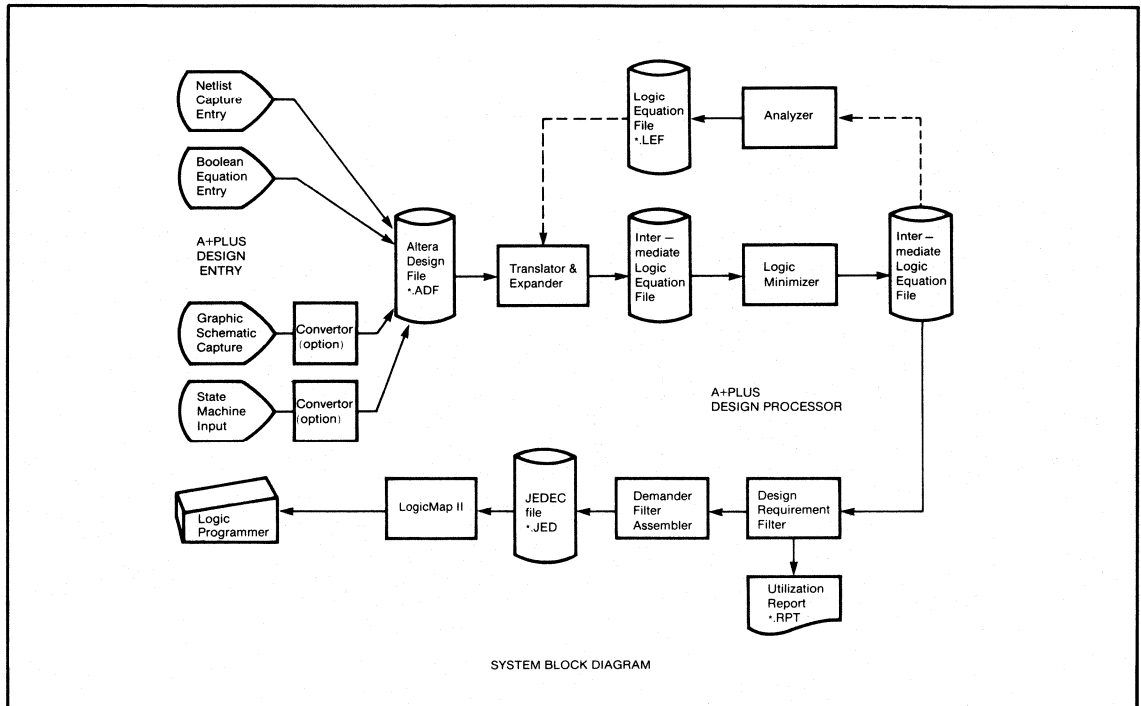
EPLD to another. The fitting process encompasses all EPLD architectural attributes such as variable product-terms, programmable flip-flops, local and global busses and I/O grouping.

## Boolean design entry

The A+PLUS design processor compiles Boolean equation designs that are written in a high-level design language. The source for the design may be created using any convenient text processor. The language supports free-form entry of all syntactical elements. Boolean equations need not be entered in sum-of-products form since the design processor will expand equations automatically. The multi-pass design processor/compiler has the ability to support intermediate equations. This feature permits significant reduction in the size of the Boolean equation source code and allows the designer to define the logic in the most natural conceptual manner.

## NetMap

NetMap is an interactive tool that allows the designer to enter a circuit schematic in netlist form. The program speeds the traditional process of entering a netlist from schematics that have been hand drawn or drafted by prompting the designer for schematic primitives and node names. The program fully checks the network for complete connectivity to ensure the design is self consistent before the A+PLUS design processing compiler is used. Changes and corrections can be easily implemented using the powerful edit functions built into this menu-driven design entry tool.



## LogicMap II

LogicMap II is the interface software that programs EPLDs from JEDEC files created by the design processor. The program uses the Altera Super Adaptive Programming algorithm ASAP™ which significantly reduces device programming times. LogicMap II calibrates the programming environment and the programming hardware when initiated. In addition the program allows the designer to review the JEDEC object code generated by the design processor in a structured manner. The program is fully menu driven and provides views of the device object code through a series of hierarchical windows. This permits low-level observation and editing of the design, viewed from a perspective similar to the logic diagram of the device in the datasheet. Individual EPROM bits within each Macrocell PLA structure may be examined or changed if desired.

## HARDWARE

The programming hardware is comprised of a software configured expansion card that occupies a single slot in the computer. Programming signals are transmitted to an external programming unit via a 30 inch ribbon cable and connector. The programming unit contains both 20 pin and 40 pin zero-insertion-force sockets for easy device insertion. All programming waveforms and voltages are derived by the expansion card from the internal computer supplies so that no additional power source is necessary. A programming indicator lamp on the programming unit shows when the unit is active.

## Contents

**User manual:** 400 pages in slip-case and binder

**Programmer:** Software controlled programming card  
Programming unit and cable  
Supports EP300, EP310, EP1200  
and EP1210

**EPLDs:** Two sample devices for evaluation

Floppy disks containing the following programs:

**Installation** software installation programs  
**A+PLUS** design processing system  
**A+PLUS** design processor/compiler  
**NetMap** interactive netlist entry program  
**LogicMap II** bit-map level programming software

- Options:**
- programming unit for EP600/EP900 (PLE4)
  - programming unit for EP1800 (PLE5)
  - adapter for 28 pin JLCC (EP600)
  - adapter for 44 pin JLCC (EP1200/EP1210)
  - adapter for 44 pin JLCC (EP900)
  - An A+PLUS user manual with accompanying demonstration disk may be purchased as a separate item for evaluation
  - Interface to PC-CAPS schematic capture system
  - Interface to DASH-2 schematic capture system
  - State-machine design entry interface

Minimum computer configuration:

IBM PC and compatible machines  
Monochrome display  
512k bytes of main memory  
Dual floppy-disk drives  
MS-DOS or PC-DOS versions 2.0 or later releases  
Full-card slot for programming card

Recommended computer configuration:

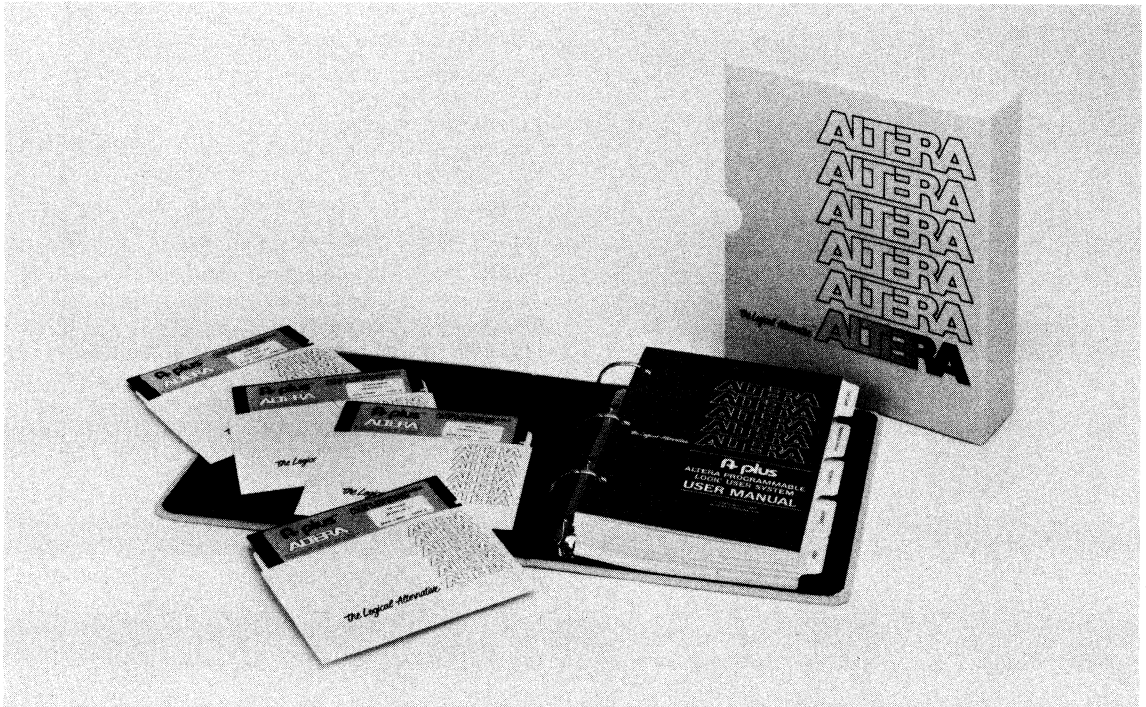
IBM XT or AT and compatible machines  
Color graphics or Enhanced graphics display  
640k bytes of main memory  
10M byte hard disk drive and floppy-disk drive  
MS-DOS or PC-DOS versions 2.0 or later releases  
Full-card slot for programming card

## FEATURES

- Complete support for all Altera EPLDS
- Cuts development time
- Boolean equation design entry
- Interactive netlist entry
- Automatic pin assignments
- Boolean equation minimization
- Sophisticated code optimizes logic design
- Optional schematic design interfaces
- Optional state-machine design interface
- Generates industry standard JEDEC object code
- Open architecture for customization
- Runs on PC compatible computers

## GENERAL DESCRIPTION

The Altera PLS2 is a suite of programs that assist the circuit designer develop optimized code used to program the target ALTERA EPLDs. The programs allow a wide variety of design input techniques to be used that suit the particular logic design task. These include schematic capture, netlist entry, Boolean equation entry and state machine entry. The designer is not restricted to just one entry method but may 'mix-and-match' methods to best meet the needs of the overall logic design. The software includes options that will minimize the logic implementation, select pin assignments automatically and configure the best architecture for the required logic function.



## Design Processor

The A+PLUS design processor translates design information from a variety of sources into programming object code for the EPLD. The design processor has the ability to work with multiple netlists or Boolean equations in the Altera Design File format. Interfaces are available that connect to popular schematic capture packages such as PC-CAPs and DASH-2 (see PLCAD1 and PLCAD2 for details of these options).

Logic minimization of designs is provided by the processor in several phases. These include Boolean minimization using a sophisticated generalized consensus algorithm which yields superior results to other heuristic reduction techniques. De Morgan's theorem inversion can be applied automatically to equations by the design processor. The processor contains algorithms based on artificial intelligence techniques to select candidate equations that will best be represented by a complemented AND-OR function. This significantly reduces product-term demands that can be generated from complex logic designs. The Altera EPLDs that contain programmable flip-flops are supported by algorithms that further optimize the architectural configuration to best suit the logic function designed.

The design processor is capable of working with all Altera devices and permits rapid evolution of designs from low-level integration to high-integration EPLDs with minimal design changes of the source design data. This is achieved using iterative design fitting algorithms to enable the basic design to be transported from one

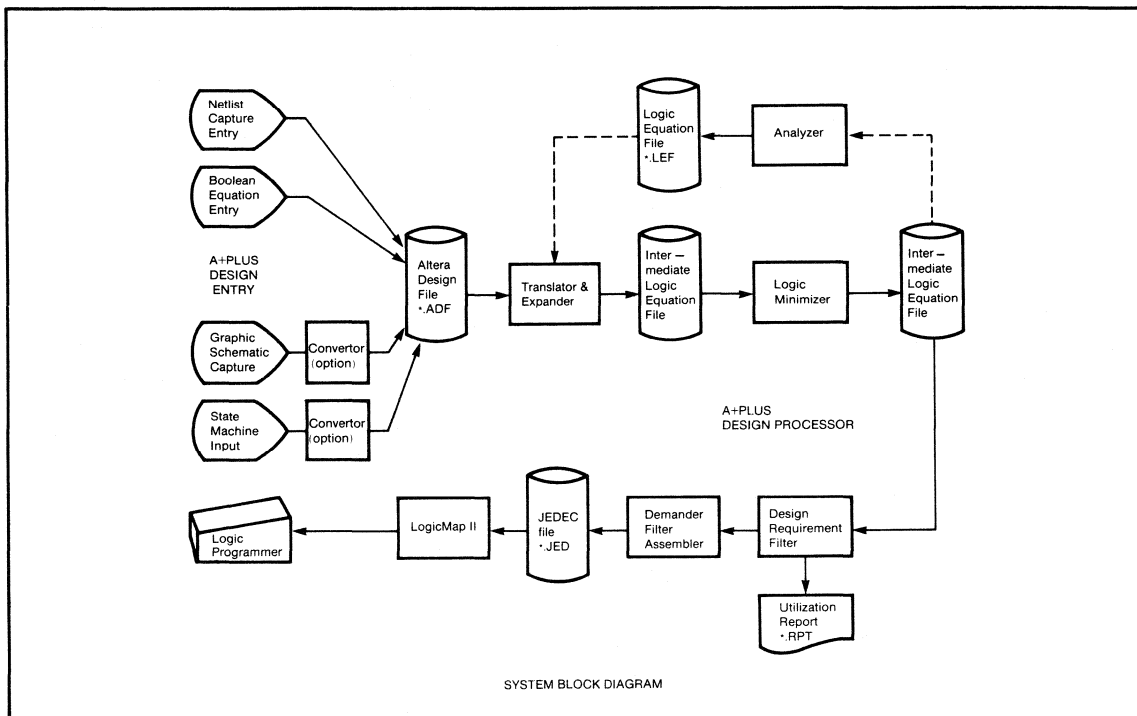
EPLD to another. The fitting process encompasses all EPLD architectural attributes such as variable product-terms, programmable flip-flops, local and global busses and I/O grouping.

## Boolean design entry

The A+PLUS design processor compiles Boolean equation designs that are written in a high-level design language. The source for the design may be created using any convenient text processor. The language supports free-form entry of all syntactical elements. Boolean equations need not be entered in sum-of-products form since the design processor will expand equations automatically. The multi-pass design processor/compiler has the ability to support intermediate equations. This feature permits significant reduction in the size of the Boolean equation source code and allows the designer to define the logic in the most natural conceptual manner.

## NetMap

NetMap is an interactive tool that allows the designer to enter a circuit schematic in netlist form. The program speeds the traditional process of entering a netlist from schematics that have been hand drawn or drafted by prompting the designer for schematic primitives and node names. The program fully checks the network for complete connectivity to ensure the design is self consistent before the A+PLUS design processing compiler is used. Changes and corrections can be easily implemented using the powerful edit functions built into this menu-driven design entry tool.



## LogicMap II

LogicMap II is a basic tool that allows the designer to review the JEDEC object code generated by the design processor in a structured manner. The program is fully menu driven and provides views of the device object code through a series of hierarchical windows. This permits low-level observation and editing of the design, viewed from a perspective similar to the logic diagram of the device in the datasheet. Individual EPROM bits within each Macrocell PLA structure may be examined or changed if desired. LogicMap II is also used as the software link between the JEDEC format object code generated by the A+PLUS design processor and the device programming hardware. This hardware is available in the complete PLDS2 development system but is not part of the PLS2 software package.

## Contents

**User manual:** 400 pages in slip-case and binder

Floppy disks containing the following programs:

**Installation** software installation programs  
**A+PLUS** design processing system  
**A+PLUS** design processor/compiler  
**NetMap** interactive netlist entry program  
**LogicMap II** bit-map level programming software

**Options:**

- An A+PLUS user manual with accompanying demonstration disk may be purchased as a separate item for evaluation.
- Interface to PC-CAPS schematic capture system
- Interface to DASH-2 schematic capture system
- State-machine design entry interface

Minimum computer configuration:

IBM PC and compatible machines  
 Monochrome display  
 512k bytes of main memory  
 Dual floppy-disk drives  
 MS-DOS or PC-DOS versions 2.0 or later releases

Recommended computer configuration:

IBM XT or AT and compatible machines  
 Color graphics or Enhanced graphics display  
 640k bytes of main memory  
 10M byte hard disk drive and floppy-disk drive  
 MS-DOS or PC-DOS versions 2.0 or later releases



### FEATURES

- Fully Integrated Development System featuring Schematic Design Entry using the PC-CAPS Mouse-driven Schematic Capture Package from P-CAD.
- Complete Schematic Symbol Library supporting all EPLD architectures as well as standard logic functions.
- Component list Converter program provides 100% compatibility with A+PLUS Software.
- Operates on IBM XT, AT or equivalent machines (using MS-DOS 2.0 Operating System) providing convenient design environment.
- Software utilities allow print/plot capability for creation of hardcopy schematics.
- Capability for Boolean Equation Entry directly within schematic drawing.
- Provides compatibility with PLSIM1 logic simulation package.

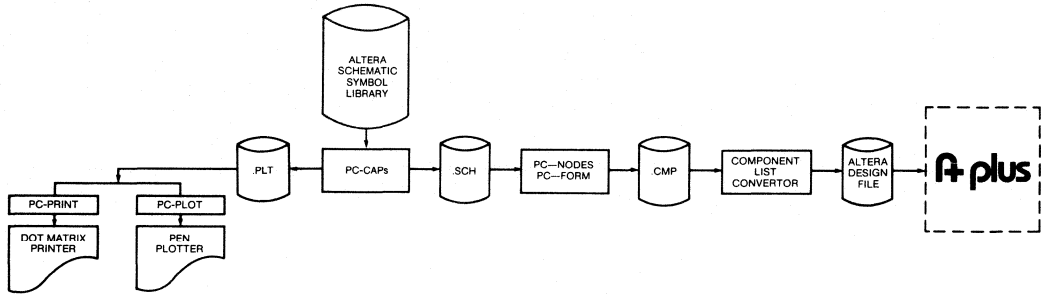
### GENERAL DESCRIPTION

PLCAD1 combines the PC-CAPS Schematic Capture package from P-CAD with the Altera Programmable Logic Development Software (A+PLUS) providing rapid development of EPLD's. Using PLCAD1, a logic design can be entered in the form of a schematic drawing and within minutes, can be transformed into a working EPLD. The schematic drawing is created by using PC-CAPS in conjunction with the Altera Schematic Symbol Library. Upon completion of the schematic a Component List representation of the design is created, which in turn, is then converted to an Altera Design File. The A+PLUS Software then processes the design for optimum performance within the target EPLD.

PLCAD1 comes complete with the PC-CAPS Schematic Capture package, the Altera Programmable Logic Development System (PLDS2), and the Interface Software. The interface includes the converter program which transforms a PC-CAPS component list to an Altera Design File and the Schematic Symbol Library which allows the PC-CAPS user to access symbol primitives used to implement EPLD logic designs.



**FIG. 1. FLOWCHART for PLCAD1**

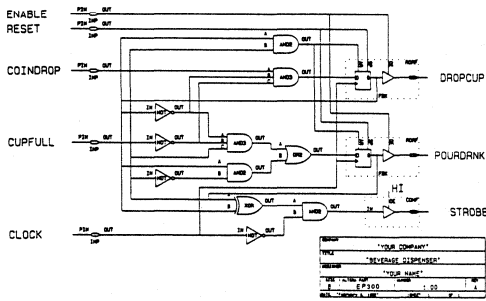


PLE1 is for existing users of A+PLUS Software who wish to enter EPLD logic designs in a schematic fashion. PLE1 consists of the PC-CAPS Schematic Capture package and the Interface software (Component List convertor and Schematic Symbol Library) required to link PC-CAPS with the existing A+PLUS Software.

PLE2 is for users who have the PC-CAPS Schematic Capture package and the A+PLUS Development Software and need to link these two systems together. PLE2 consists of the Component List Converter program and the Schematic Symbol Library.

Logic simulation capability is also made available for PLCAD1 users via PLSIM1. The PLSIM1 package includes the PC-LOGS Logic Simulator which enables logic within any schematic design created with PC-CAPS to be fully simulated.

**FIG. 2. PC-CAPS SCHEMATIC CAPTURE**



**MINIMUM PC CONFIGURATION FOR PLCAD1, PLE1, PLE2**

- IBM XT, AT or equivalent machines
- 640K bytes of main memory
- Color Monitor
- 10M byte hard disk drive with floppy disk drive
- Rev. 2.0 or later DOS Operating System

**CONTENTS**

**PLCAD1**

Altera PLDS2

- A+PLUS Software
- Programming Card and Unit
- (2) EPLD Components
- User Manual

P-CAD PC-CAPS Schematic Capture Package

- PC-CAPS Software
- Software Utility Programs including PC-PRINT and PC-PLOTS
- Hardware Security Device
- Optical Mouse and Mouse tablet

Interface Software

- Component List Converter
- Schematic Symbol Library

**PLE1**

P-CAD PC-CAPS Schematic Capture Package

- PC-CAPS Software
- Software Utility Programs including PC-PRINT and PC-PLOTS
- Hardware Security Device
- Optical Mouse and Mouse tablet

Interface Software

- Component List Converter
- Schematic Symbol Library

**PLE2**

Interface Software

- Component List Converter
- Schematic Symbol Library

**ORDERING INFORMATION**

Order by Product Number: PLCAD1, PLE1, PLE2

### FEATURES

- Fully Integrated Development System featuring Schematic Design Entry using FutureNet DASH-2 Schematic Designer — Mouse driven.
- Complete Schematic Symbol Library supporting all EPLD architecture features.
- Pinlist Converter — 100% compatible with A+PLUS Software.
- Operates on IBM Personal Computer or Equivalent Machine using MS-DOS Operating System.
- Hard Copy Print Out or Plots of Schematic Drawings.
- Allows Boolean Equation Entry to be Directly Integrated into Schematic Drawings.
- Interface to TTL Library.

### GENERAL DESCRIPTION

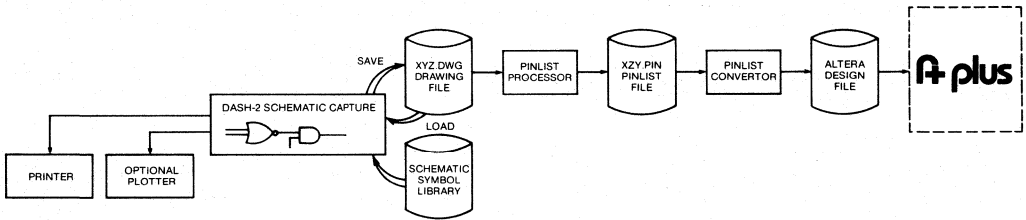
PLCAD2 combines the DASH-2 Schematic Designer from FutureNet with the Altera Programmable Logic Development Software (A+PLUS) to allow schematic design entry for EPLDs. With the use of DASH-2 and A+PLUS, logic designs can be created via

schematic drawings and within minutes a working EPLD can be produced. The schematic drawing is first constructed with the DASH-2 Schematic Designer and Altera Schematic Symbol Library. The Pinlist representation of the design is then converted to an Altera Design File which can then be processed by the A+PLUS software, (see PLS2 Data Sheet for details). The PLCAD2 comes complete with the DASH-2 Schematic Designer, Programmable Logic Development System (PLDS2), and the Interface Software. This interface includes a Pinlist Converter which translates the DASH-2 pinlist to an Altera Design File and the Schematic Symbol Library which includes symbol primitives used to implement the EPLD logic functions. Schematic designs for Altera EPLDs can only be created from these symbol primitives.

PLE10 is for existing users of the A+PLUS software who wish to enter designs with schematic representation. PLE10 contains the DASH-2 Schematic Designer and Interface Software (Pinlist Converter and Schematic Symbol Library) required to link DASH-2 with their existing A+PLUS Development Software.



**FIG. 1. FLOWCHART FOR PLCAD2**

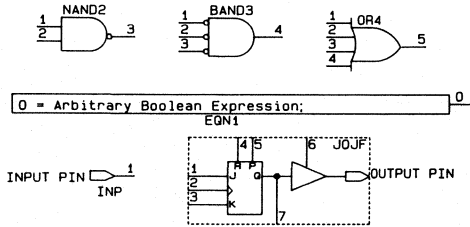


PLE20 is for users who have the FutureNet Schematic Designer and the A+PLUS Development Software. PLE20 includes the Pinlist Converter and Schematic Symbol Library. PLE20 will interface DASH-1, 2, or 3 with the A+PLUS Development Software.

A TTL Library consisting of over 100 SSI and MSI TTL devices has been constructed with the DASH-2 Schematic Designer and Altera Symbol Primitives. Users can integrate the TTL devices directly into their DASH-2 drawings. The TTL Library is available to all users who purchase the A+PLUS Software Maintenance Agreement (PLDSMA1).

**FIG. 2**

The Schematic Symbol Library contains over 60 logic primitives which interface directly with the DASH-2 Schematic Designer. These primitives are in the form of Input, Basic Logic Gates, Equations, and I/O Architectures.



**MINIMUM PC CONFIGURATION REQUIRED FOR PLCAD2, PLE10, PLE20**

- IBM PC, XT, AT or equivalent
- 512K bytes of main memory
- Monochrome monitor and card
- 1 (one) double sided floppy disk drive with hard disk
- Rev. 2.0 or later DOS Operating System.

**PRODUCT DESCRIPTION**

**PLCAD2**

**Altera PLDS2**

- A+PLUS Software
- Programming Card and Box
- (2) EPLD Components
- User Manual with Slip Case and Binder
- EPLD Worksheets

**FutureNet DASH-2 Schematic Designer**

- DASH-2 Software
- Graphics Card
- Mouse

**Interface Software**

- Pinlist Converter
- Schematic Symbol Library

**PLE10**

**FutureNet DASH-2 Schematic Designer**

- DASH-2 Software
- Graphics Card
- Mouse

**Interface Software**

- Pinlist Converter
- Schematic Symbol Library

**PLE20**

**Interface Software**

- Pinlist Converter
- Schematic Symbol Library

**ORDERING INFORMATION**

Order by Product Number: PLCAD2, PLE10, PLE20

### FEATURES

- Provides logic simulation capability for PLCAD1 users.
- Complete simulation models allow simulation of entire EPLD family.
- Twelve state event-driven simulation format yielding faster simulation without compromising accuracy.
- Modeling capability of both internal and external nodes provides thorough analysis of entire EPLD structure.
- Spike and Stability Analysis for detection and display of glitches or unstable circuit nodes.
- Operates on IBM PC/XT/AT (using MS-DOS Operating System) providing convenient design environment.
- Spooled storage as well as on-screen display capability of output formats which include timing waveforms and formatted state listings.
- Post processing utilities allow printout capability for timing waveforms and state listings.

### GENERAL DESCRIPTION

PLSIM1 provides PLCAD1 users with interactive simulation capability for checking critical logic within EPLD designs. This capability is provided by P-CAD's, PC-LOGS, the event-driven (only logic nodes where a state change is to occur are evaluated at any given time) logic simulator used in conjunction with the Altera Simulation Models Library.

Twelve logic states are used to track and display the status of both internal and external circuit nodes. These states are comprised of all three logic levels (0, 1, and Unknown) and four signal strengths (Supply, Driving, Resistive and High Impedance). As a result, fast and accurate simulation of all EPLD's is achieved.

PLSIM1 allows users to easily define input stimuli. Flipflops can be loaded with initial conditions and both synchronous and asynchronous input signals may be established. Input waveforms are defined in terms of simulator time steps with options for high or low state definition, pulse duration, and pulse repetition. For instance, a clock signal may be defined as a repetitive pattern of low for 20 time steps and high for 10 time



steps. An unlimited number of variations are thus possible.

To efficiently continue through a series of interactive sessions, input stimuli are saved and may be reloaded. In addition, simulation may repeatedly be terminated and re-initialized to permit changes to input waveforms, probing and display format. Execution can be set to occur in either continuous or step mode for variable lengths of time. Inputs, outputs and internal nodes can be probed to monitor logic signals just as with a logic analyzer. Thus, complete simulation of your design can progress over an extended period of time without tedious repetition.

The outputs resulting from input stimuli can be evaluated with spike and stability analysis to determine possible glitches and unstable circuit nodes. Spikes and unstable nodes can be set up as simulation breakpoint conditions causing a halt in simulation as well as a display of the relevant cause.

To enhance data analysis as well as documentation capability, PLSIM1 offers various display options and post-processing features. PLSIM1 can be used to display up to 16 timing waveforms simultaneously with automatic node naming. Each waveform physically depicts all three logic levels (0, 1, and Unknown). All output formats may be stored to disk and processed to create printed timing diagrams and formatted state listings for permanent documentation.

The contents of the PLSIM1 logic simulation package consists of the PC-LOGS logic simulator, the POST-SIM post-processing software, the Altera Simulation Models Library for complete simulation of the entire EPLD family, and a comprehensive User Manual.

### MINIMUM PC CONFIGURATION

- IBM XT, AT or equivalent
- 640K bytes of main memory
- Color Monitor
- 1 (one) double sided floppy disk drive with hard disk
- Rev. 2.0 or later DOS Operating System.

### CONTENTS

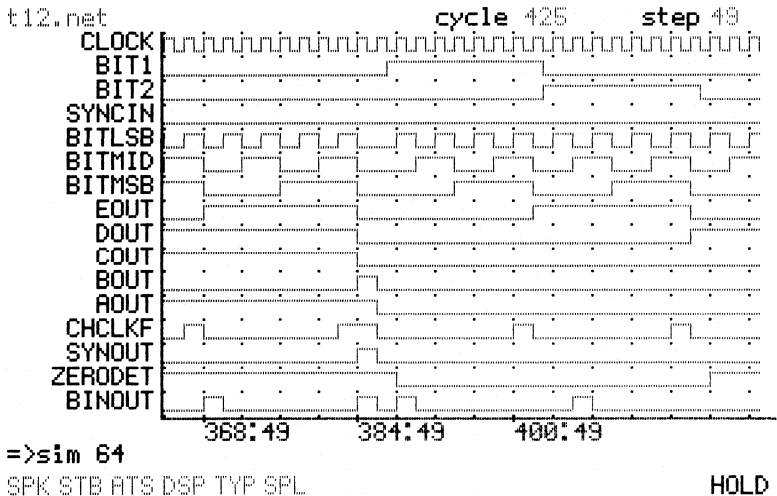
- P-CAD PC-LOGS Logic Simulator
- Altera Simulator Models Library
- POST-SIM Post-processing software
- User Manual

### ORDERING INFORMATION

Order by Product Number: PLSIM1

### OUTPUT WAVEFORMS

This waveform was produced by PLSIM1 for a sample EPLD design. Such waveforms can be seen in full color display or saved as a file to be printed or plotted for permanent documentation.



### FEATURES

- Automatic update for new EPLDs
- Applications support Hotline
- Modem access to design library
- Discounts on hardware enhancements
- Design assistance with modem design transfer support

### GENERAL DESCRIPTION

These software maintenance and support products provide the development system user access to the latest revisions of software. Development software is subject to periodic upgrade to provide new features and to support new EPLDs. These products provide a simple approach to ensure that your development system is fully up to date and ensures you are able to use the latest EPLD technology. The support includes access to applications assistance for design work with Hot-line access to applications engineers. Designs can be transferred to Altera via a 24 hour auto-answer dial-up modem service.

An extensive design library can also be accessed via this modem link.

#### TTL and Application Design library

The Altera TTL library which contains over 100 logic designs. These are intended as practical examples of small circuits implemented in EPLDs. The designs in the library contain many techniques that may be useful when developing more complex circuits.

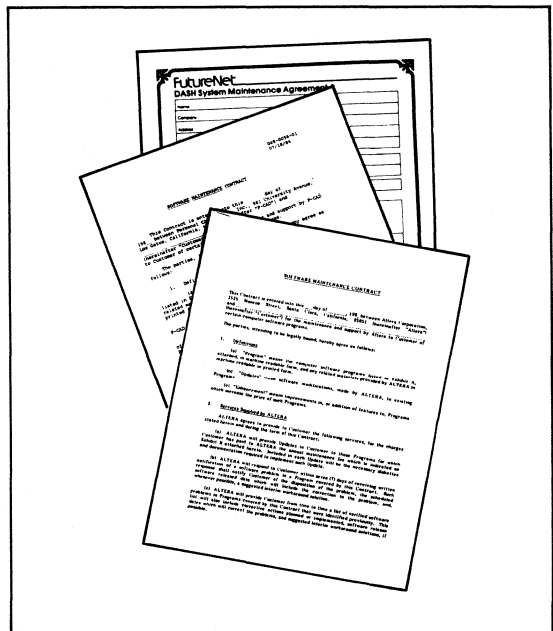
The complete library contains schematic drawings, and their Boolean algebra equivalents in the Altera Design File format (ADF). These designs and all those published in application notes and application briefs (including those in this handbook) form the body of the library. Subscribers to the A+PLUS upgrade and maintenance package have free access to this design database.

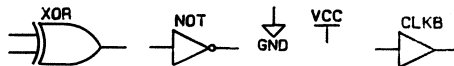
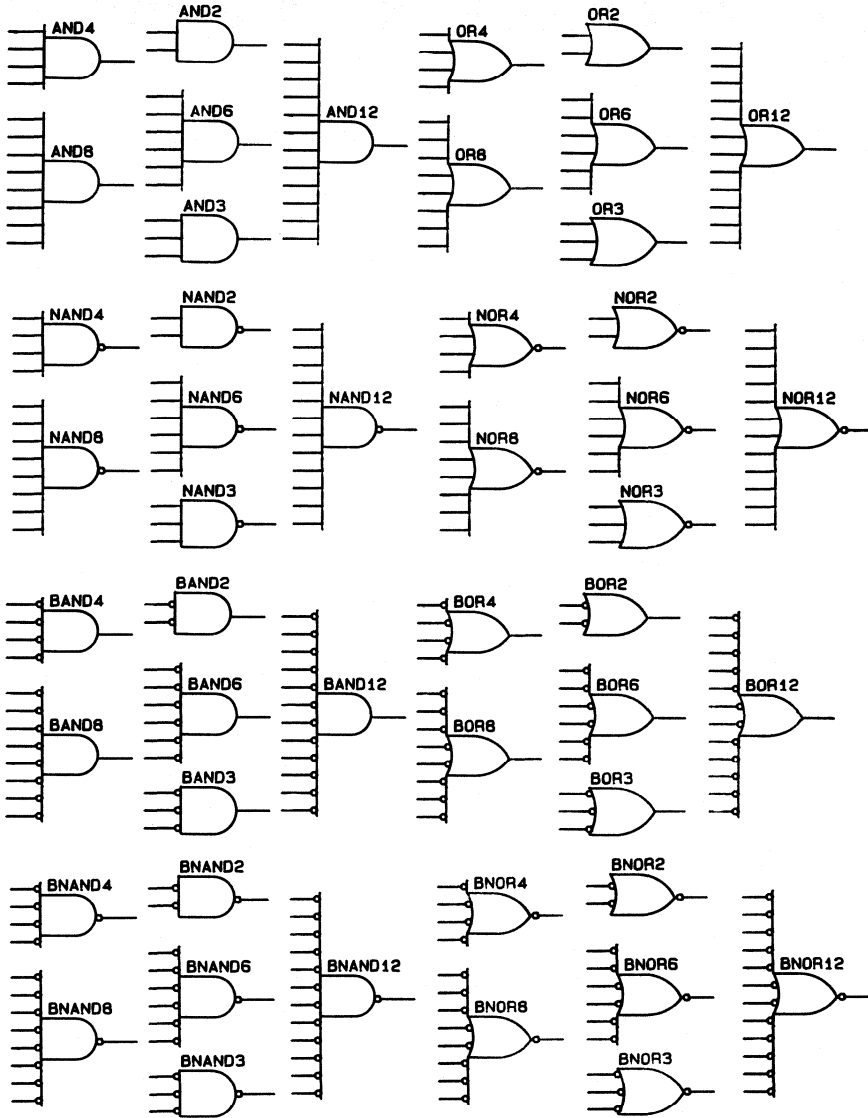
#### Options:

- PLDSMA1 Upgrade and maintenance for Altera PLDS2 and PLS2 software.
- PLDSMA2 Maintenance for PCAD PC-CAPS schematic capture software.
- PLDSMA3 Maintenance for FutureNet DASH-2 schematic capture software.

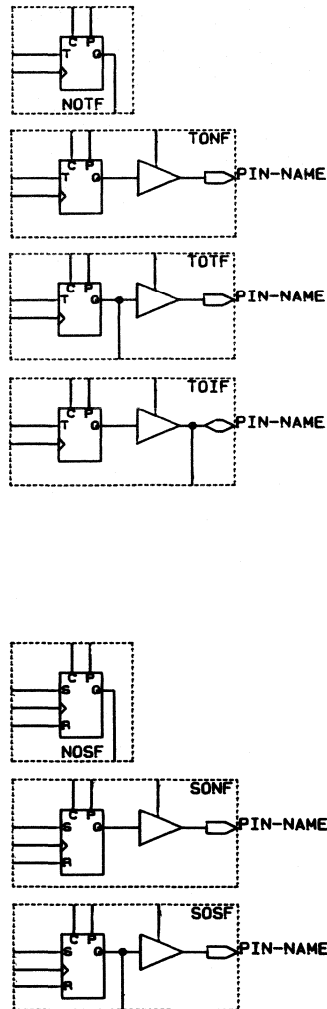
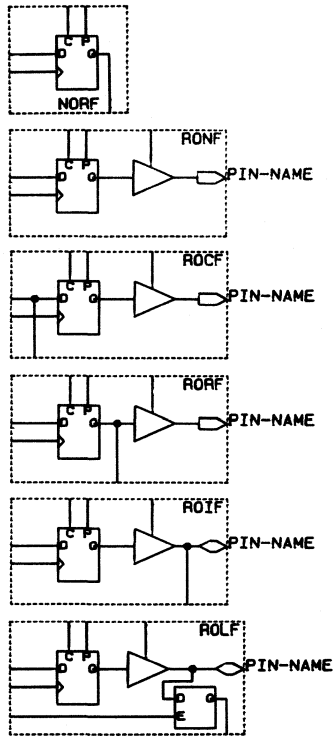
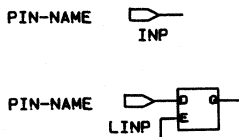
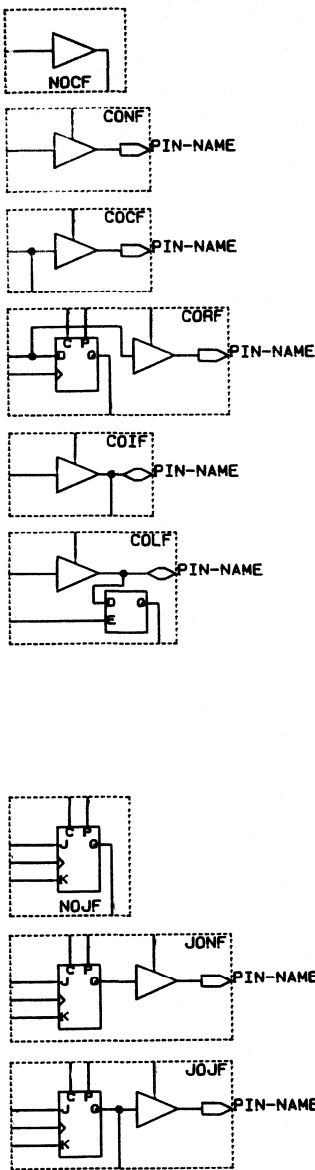
### Contents:

- User manual:** Modem and Applications library access instructions.
- Access:** 1200 Baud Dial-Up Auto-Answer modem. Password access only.
- Protocol:** Crosstalk XVI error-free protocol for both binary and text file transfers.
- Host system:** AT&T 6300 with 640k bytes of RAM and 10M byte hard-disk.
- Duration:** 12 months, renewable.
- Recommended computer configuration:**
  - IBM XT or AT and compatible machines
  - 1200 Baud modem (212A)
  - Crosstalk XVI software
  - Color graphics or Enhanced graphics display
  - 640k bytes of main memory
  - 10M byte hard disk drive and floppy-disk drive
  - MS-DOS or PC-DOS versions 2.0 or later releases
  - Full-card slot for programming card









0 - Arbitrary Boolean Expression; EGN1

01 - Arbitrary Boolean Expression; 01  
 02 = : 02  
 03 = : 03  
 04 = : 04  
 05 = : 05  
 06 = : 06  
 07 = : 07  
 08 = : 08

COMPANY				ALTERA CORP.							
TITLE								A+PLUS DESIGN PRIMITIVES			
DESIGNER				EPLD				ALL			
SIZE	CODE	NUMBER	REV								
C	-	1.00	I								
DATE				AUG 29, 1985				SHEET 1 OF 1			

EGN8





---

**APPLICATION INFORMATION****PAGE NO.**

---

AN 1	Introduction to EPLDs .....	84
AN 2	Replacing 20 pin PALs .....	98
AN 3	Memory Interfacing with EPLDs .....	108
AN 4	EPLD Simulation .....	116
AN 5	NetMap Design Entry .....	128
AN 6	High Speed Custom UART .....	140
AB 1	TTL Library for EPLD Design .....	148
AB 2	Boolean Library for EPLD Design .....	157
AB 3	Manchester Encoder/Decoder .....	190
AB 4	T1 Serial Transmitter .....	192
AB 5	Single Chip Signature Analyzer .....	198
AB 6	EP1200 as a 6502/6800/68000 Peripheral .....	201
AB 7	EP1200 as an 8085/8086 Peripheral .....	203
AB 8	Efficient Counter Design with Toggle Flip-Flops ...	205
AB 9	Designing Asynchronous Latches with EPLDs .....	208
AB 10	Counter Design Using the EP1200 .....	211
AB 11	16 Bit Up/Down Counter with Left Right Shift Register .....	219
AB 12	Equivalent Gate Counting .....	223

### INTRODUCTION

This Application Note provides an introduction to EPLD's (erasable-programmable logic devices) via the ALTERA EP300. A typical application of the EP300 is shown in Figure 17. The EP300's flexible architecture (Figure 5) makes it ideal for the design of both combinatorial and sequential logic.

User programmable logic is very effective for developing custom circuits quickly. Also it is invaluable in the fast prototyping of circuits that will be mass produced. The EP300 EPLD allows the designer to create a design and integrate it on a chip the same day. In addition, the erasable nature of the EP300 lets the designer realize circuit changes with the same chip.

#### Notation: AND-Arrays

Schematics of PLD's use a shorthand notation called an "AND-array" to represent several large AND gates which have common inputs. Figure 1 shows three different representations of the same logic function. Circuit I is presented in normal logic notation, Circuit II has been modified to a sum-of-products implementation, and Circuit III is written with the AND-array notation. A dot at an intersection of the 2 by 8 grid in Circuit III represents a connection between a vertical wire and an input of

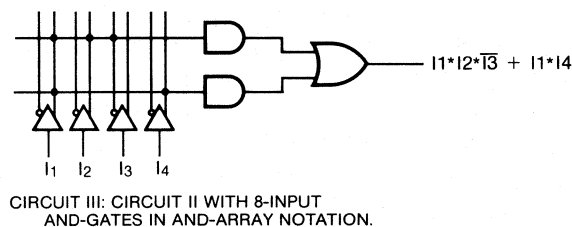
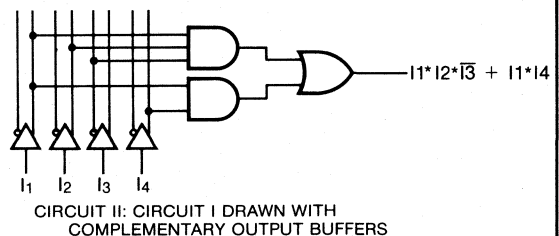
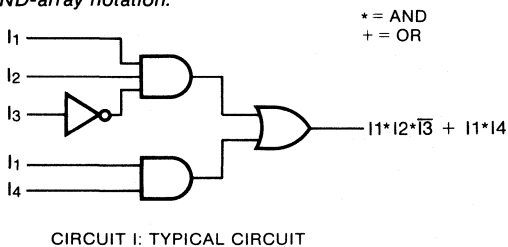
one of the 8-input AND gates on the right. No dot implies that there is no connection and that the AND gate's unused input floats high to logic 1.

A 2 by 8 AND-array, like the one in Circuit III of Figure 1, can realize any Boolean function of four variables provided only two product terms are required when it is expressed in sum-of-products form. The outputs of the two AND gates in Figure 1 are called "product terms" or "p-terms."

Figure 2 shows how AND-arrays can realize the five basic functions AND, OR, XOR, NAND, and NOR if the functions are first converted into sum-of-products form. Notice how the unused product terms of the AND and NOR in Figure 2 are set to an expression that always evaluates to zero. Note also that NAND and NOR use DeMorgan's theorem in order to realize a function that has its final output inverted. (Since the EP300 features output inversion, DeMorgan's theorem does not always have to be applied. It is used here for illustrative purposes only.)

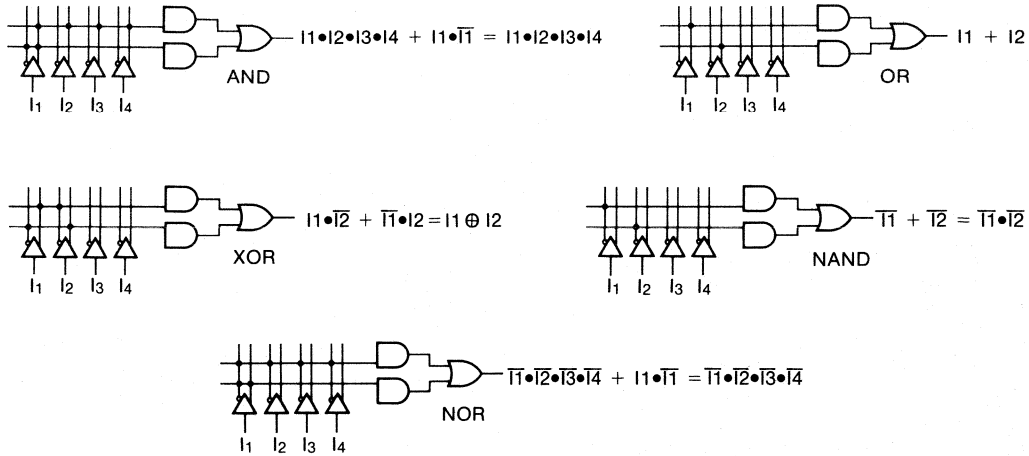
Any Boolean expression—no matter how complex—may be written in sum-of-products form. This powerful concept is best understood by transforming an arbitrary expression into its truth table, then converting the table back into an expression—but into one that is in SOP form. Figure 3 shows an example of this procedure.

**Figure 1:**  
AND-array notation.



**Figure 2:**

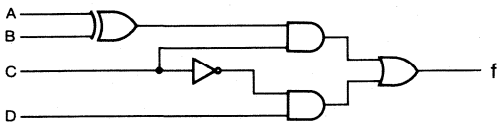
Basic Boolean functions implemented with AND-arrays.



**Figure 3:**

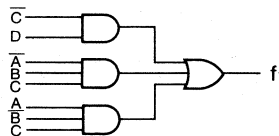
Example of the procedure used to show that any combinational network has an equivalent two level form. The network is first converted into its truth table. Next, a sum-of-minterms expression that has the same truth table is derived and then

reduced until it has as few product terms as possible. The resulting two-level structure is the minimal sum-of-products form of the original network. This procedure can be repeated for any combinational network — no matter how complex.



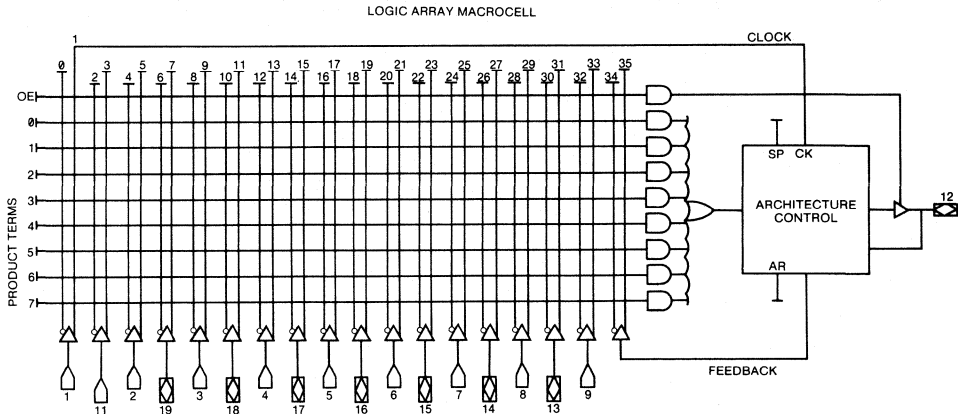
A	B	C	D	f = f1
0	0	0	0	0
0	0	0	1	1
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	1
0	1	1	0	1
0	1	1	1	1
1	0	0	0	0
1	0	0	1	1
1	0	1	0	1
1	0	1	1	1
1	1	0	0	0
1	1	0	1	1
1	1	1	0	0
1	1	1	1	0

$$\begin{aligned}
 f' &= \bar{A} \cdot \bar{B} \cdot \bar{C} \cdot D + \bar{A} \cdot \bar{B} \cdot C \cdot \bar{D} + \bar{A} \cdot B \cdot C \cdot \bar{D} + \bar{A} \cdot B \cdot C \cdot D \\
 &+ A \cdot \bar{B} \cdot \bar{C} \cdot D + A \cdot \bar{B} \cdot C \cdot \bar{D} + A \cdot \bar{B} \cdot C \cdot D + A \cdot B \cdot \bar{C} \cdot D \\
 &= \bar{A} \cdot \bar{C} \cdot D \cdot (\bar{B} + B) + \bar{A} \cdot B \cdot C \cdot (\bar{D} + D) \\
 &+ A \cdot \bar{C} \cdot D \cdot (\bar{B} + B) + A \cdot \bar{B} \cdot C \cdot (\bar{D} + D) \\
 &= \bar{A} \cdot \bar{C} \cdot D + \bar{A} \cdot B \cdot C + A \cdot \bar{C} \cdot D + A \cdot \bar{B} \cdot C \\
 &= \bar{C} \cdot D \cdot (\bar{A} + A) + \bar{A} \cdot B \cdot C + A \cdot \bar{B} \cdot C \\
 f' &= \bar{C} \cdot D + \bar{A} \cdot B \cdot C + A \cdot \bar{B} \cdot C
 \end{aligned}$$



**Figure 4:**

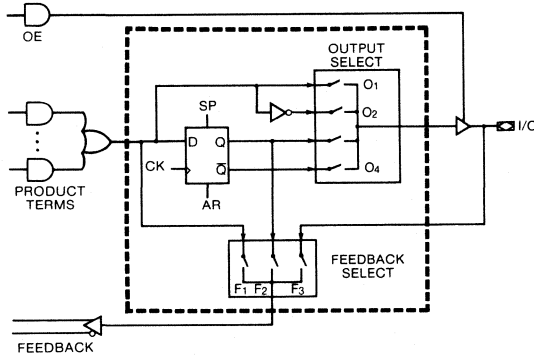
One of eight EP300 logic array Macrocells. The inputs of each AND-array come directly from the EP300's ten dedicated input pins and from the eight architecture control blocks via feedback lines.



NOTE: □ = 1/O Pin, in which Logic Array input is from feedback path

**Figure 5:**

I/O Architecture Control block of Macrocell. Zero or one output select switches may be closed; zero or one feedback select switches may be closed.



O <sub>1</sub>	O <sub>2</sub>	O <sub>3</sub>	O <sub>4</sub>	F <sub>1</sub>	F <sub>2</sub>	F <sub>3</sub>	OUTPUT	FEEDBACK
I	O	O	O	I	O	O	Combinatorial/High	Comb.
I	O	O	O	O	I	O	"	Reg.
I	O	O	O	O	O	I	"	I/O
O	I	O	O	I	O	O	Combinatorial/Low	Comb.
O	I	O	O	O	I	O	"	Reg.
O	I	O	O	O	O	I	"	I/O
O	O	I	O	I	O	O	Registered/High	Comb.
O	O	I	O	O	I	O	"	Reg.
O	O	I	O	O	O	I	"	I/O
O	O	O	I	I	O	O	Registered/Low	Comb.
O	O	O	I	O	I	O	"	Reg.
O	O	O	I	O	O	I	"	I/O
n	n	n	n	n	n	n	any of above	none
O	O	O	O	m	m	m	none	any of above

I = Connection Closed  
 O = Connection Open  
 nnnn = Any Legal Output Configuration  
 mmm = " " " Input " "

**The EP300 Macrocell**

Figure 4 shows a Macrocell—one of eight such structures in an EP300. Each Macrocell has a 36-input/9-output programmable AND-array, an eight input OR gate, a tri-state buffer driving an I/O pin, and a programmable Architecture Control block.

The inputs of each Macrocell's AND-array come from the EP300's ten input pins and eight feedback paths (with one feedback path coming from each of the Macrocell's Architecture Control blocks.) Since both the inverse and complement signals are available, the total number of AND-array inputs is 36.

Eight of the AND-array's product terms are OR'ed together to form an input to the Macrocell's Architecture Control Block. The ninth product term is used to enable the Macrocell's tri-state buffer. This means that the Architecture Control block's input can be any Boolean function of 36 inputs that can be represented with eight or fewer product terms, and the enable line can be any function that can be expressed in one product term.

The Macrocell Architecture Control block shown in Figure 5 consists of a D-type flip-flop and two programmable multiplexers. One multiplexer selects the input of the Macrocell's tri-state buffer, while the other selects the source of the Macrocell's feedback signal. The feedback multiplexer can select one of three sources; the output multiplexer can select one of four sources. Together, the multiplexers can configure a Macrocell and its associated I/O pin to act as a combinatorial output, a registered output, an input, or possibly a combinatorial output with registered feedback, or a registered output with combinatorial feedback, etc.

The ability of each Macrocell to have a different configuration makes a group of Macrocells a powerful design tool. Such a group is shown in Figure 6, the block diagram of the EP300.

**DESIGN EXAMPLE:**

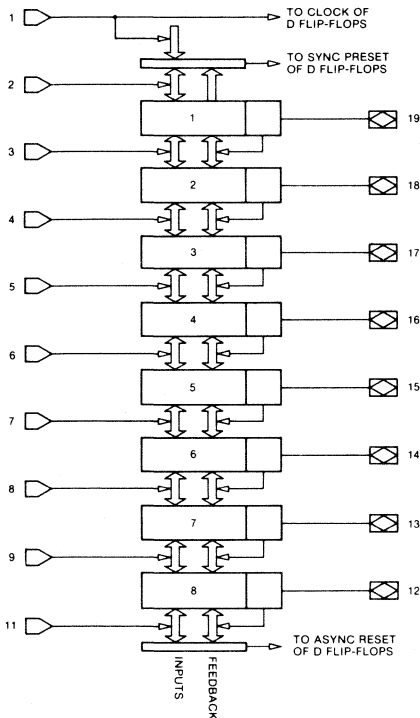
**BOOLEAN FUNCTIONS**

Figure 7 shows the pinout of an EP300 that was programmed to implement the Boolean functions in Figure 2. In this example each function uses one Macrocell.

The EP300 is programmed with the aid of the ALTERA logic design program A+PLUS. A+PLUS converts the schematic representation of a design into an equivalent sum-of-products form that will fit in the AND-OR array of an EP300. The schematic in Figure 8

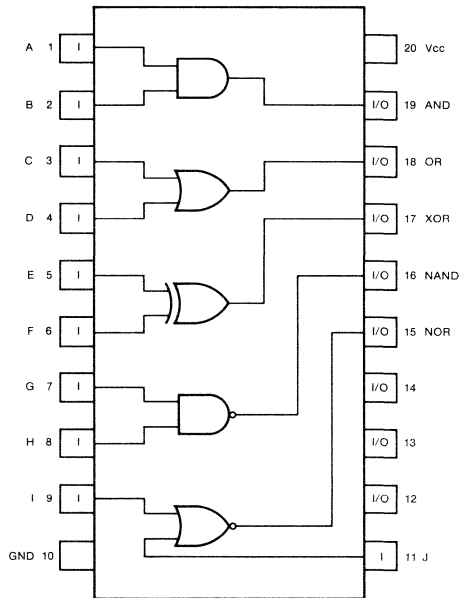
**Figure 6:**

*Block diagram of the EP300 with eight Macrocells like the one shown in figure 4. Pins 1-9 and 11 are dedicated inputs. Pin 1 also clocks the Macrocells' D flip-flops.*



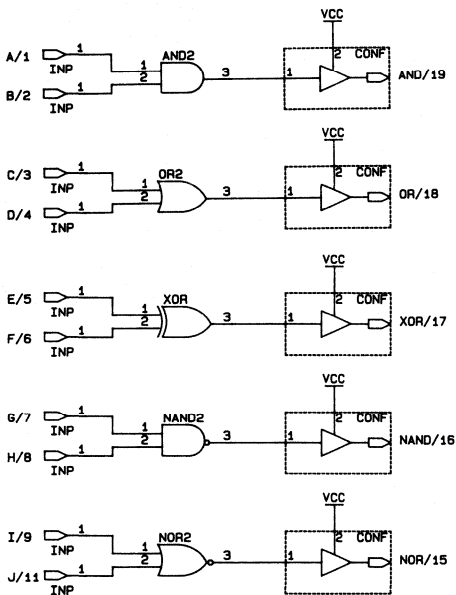
**Figure 7:**

*Implementing the Boolean functions of Figure 2 in an EP300. Only five of eight Macrocells are used.*



**Figure 8:**

The schematic used by A+PLUS to implement the EP300 in Figure 7. Each input pin (INP) is assigned to a physical pin of the EP300 by following the pin name with a slash and a pin number. The output pins (CONF, or Combinatorial Output/No Feedback primitive) are assigned to EP300 output pins in the same way.



is a reproduction of the actual source schematic used by A+PLUS to implement the design in Figure 7.

The five boxed primitives in the schematic represent five EP300 macrocells whose output architecture multiplexers have selected Combinatorial Output and whose feedback multiplexers have selected No Feedback—hence the output primitives' acronym, CONF. The output macrocells have been assigned to specific physical I/O pins by following the pin name in each macrocell with a slash and a pin number. The input pins on the left of the schematic have been assigned to physical input pins in the same manner.

A+PLUS converts the schematic level design into its equivalent sum-of-products form, the form that will fit into the EP300's programmable AND-OR array (similar to Figure 2.)

From this SOP form of the design, A+PLUS produces a JEDEC Standard file for logic device programmers (JEDEC council JC-42.1 standard N-3.) The JEDEC file is used by a logic device programmer when it programs an ALTERA EPLD. An EP300 programmed with the JEDEC file that A+PLUS produced from the schematic in Figure 8 will operate just as described in the schematic. Compare Figures 2 and 8 with Figure 9, the schematic of an EP300 programmed with the JEDEC file from this example.

**Figure 9:**

Characteristics of delay, set-reset, toggle, and JK flip-flops.

OUTPUT STATE CHANGE	FLIP-FLOP TYPE/REQUIRED INPUTS			
	D	SR	T	JK
0 → 0	0	0-	0	0-
0 → 1	1	10	1	1-
1 → 0	0	01	1	-1
1 → 1	1	-0	0	-0

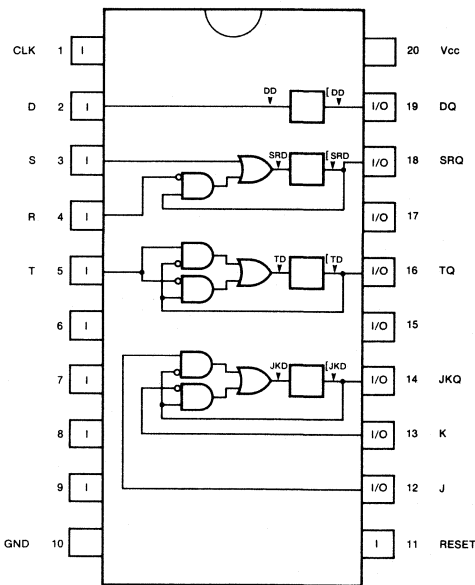
**Figure 10:**

Characteristic equations of delay, set-reset, toggle, and JK flip-flops.

$$\begin{aligned}
 Q^+ &= D && \text{(D flip-flop)} \\
 Q^+ &= S + R'Q && \text{(SR flip-flop)} \\
 Q^+ &= T \oplus Q && \text{(T flip-flop)} \\
 Q^+ &= JQ' + K'Q && \text{(JK flip-flop)}
 \end{aligned}$$

**Figure 11:**

Schematic of EP300 programmed to emulate D, SR, T, and JK flip-flops.

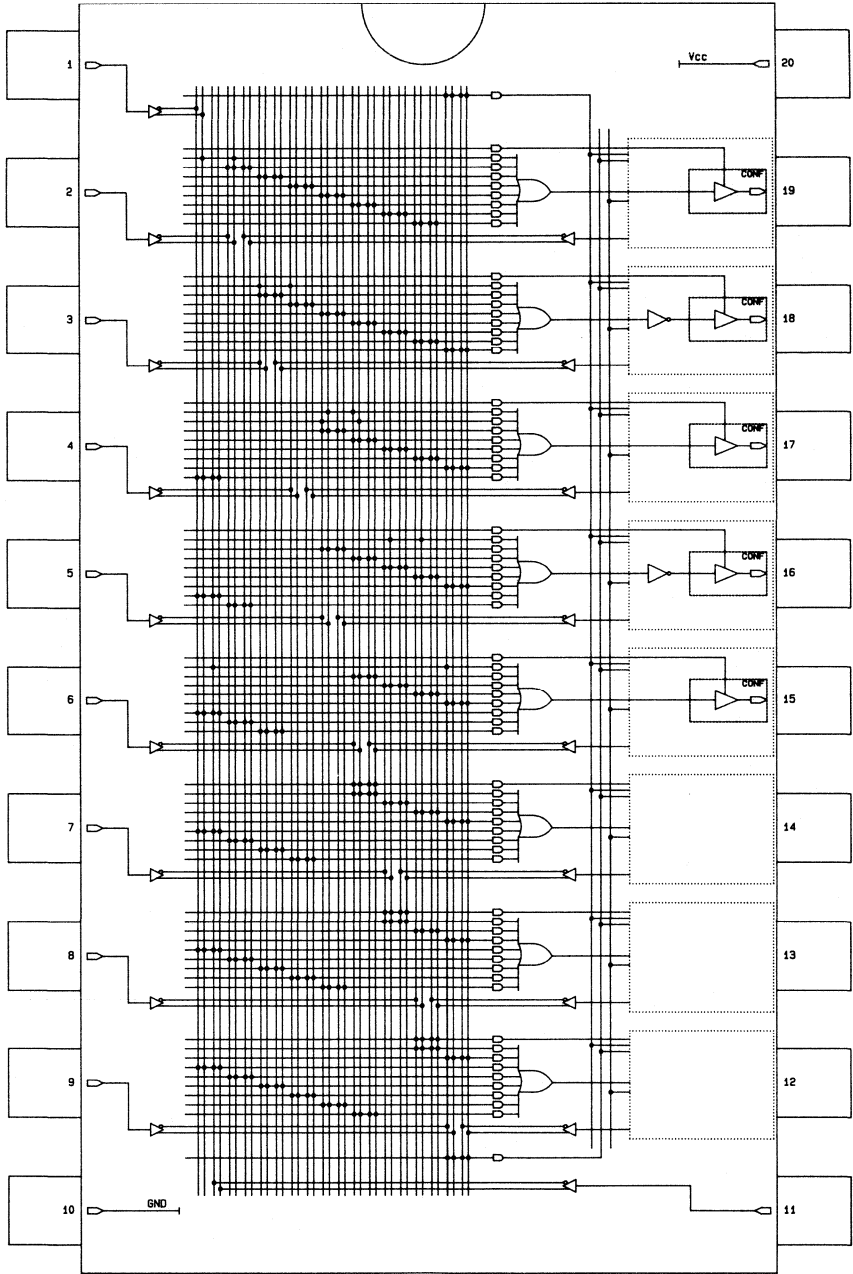




**Figure 12:**

Schematic of an EP300 programmed to function as the circuit in Figure 8. The JEDEC programming file produced by A+PLUS instructs that the Macrocells on pins 15, 17, and 19 be programmed for non-inverted combinatorial output with Output Select switch 0<sub>1</sub> (see Figure 5.) The remaining two Macrocells on pins 16 and 18 are programmed for inverted combinatorial output with switch 0<sub>2</sub>. Each dot in the AND-array represents a connection between a vertical input

line and one input of the 36-input AND gate to its right. The unused inputs of the AND gates float high, allowing 36-input AND gates to function with fewer than 36 inputs. AND gates that are completely unused are disabled by programming them with an equation that always evaluates to FALSE. These unused AND gates form a diagonal pattern that can be seen in the chip's AND-array.



**DESIGN EXAMPLE:**

**FLIP-FLOPS**

By feeding the output of an EP300 Macrocell's D-type flip-flop back to its AND-array, the Macrocell can be made to emulate any other type of flip-flop. This approach is useful to a logic designer implementing a circuit that uses flip-flops other than D type.

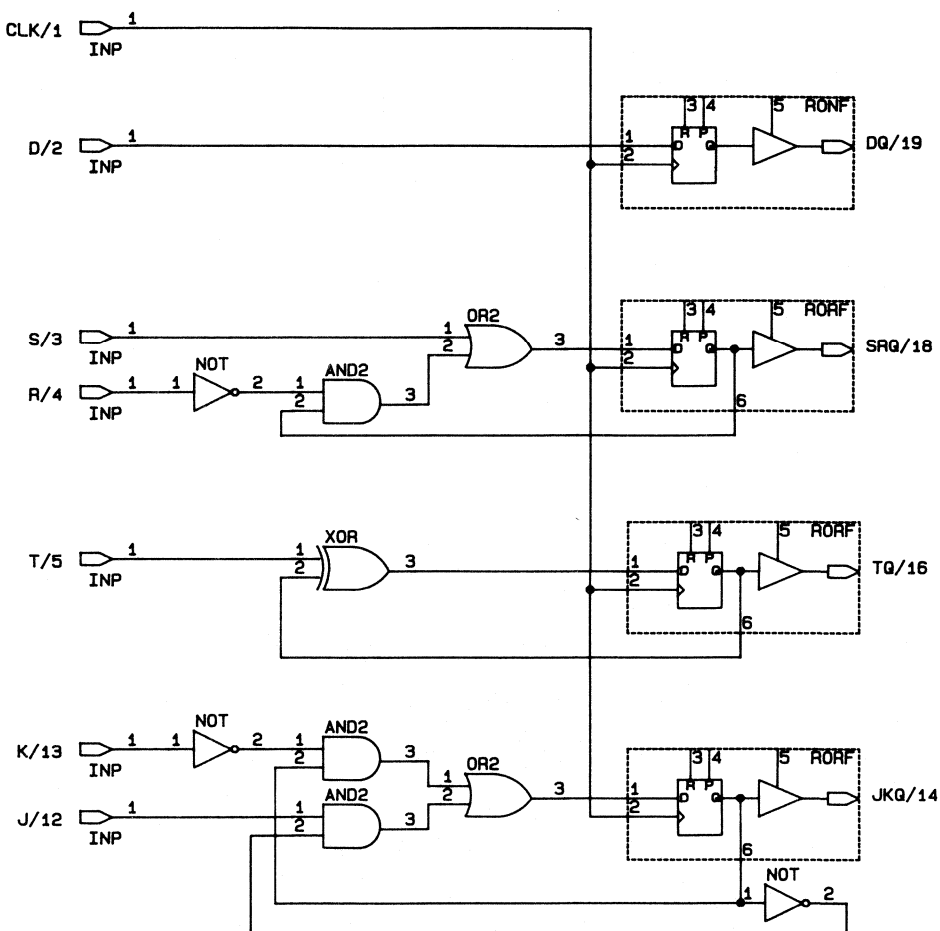
Figure 10 shows the characteristics of four types of flip-flops: D, Set-Reset, Toggle, and JK. The table shows that D flip-flop's output will change from a one to a zero on the next clock pulse only if the D input is a zero. For a JK flip-flop to make the same state change,

its K input must be a one. (The dash under the J entry in the table indicates "don't care," meaning the J input can be either zero or one.)

The four circuits in Figure 12 contain D-type flip-flops and the combinatorial logic necessary for the D flip-flops to emulate the flip-flops described by characteristic equations in Figure 11. The schematic in Figure 12 may be converted into A+PLUS design primitives, as in the previous example, and the resulting design run through A+PLUS to produce a JEDEC file for programming EP300's.

**Figure 13:**

*Schematic for implementing the design in Figure 12. The RORF Macrocell primitives direct A+PLUS to select registered output and registered feedback (switches 0<sub>3</sub> and F<sub>2</sub> in Figure 5.) Unconnected inputs on the top sides of the RORF's default to reasonable values: the reset and preset inputs default to FALSE, the output enables default to TRUE.*



## STATE MACHINES

Single flip-flops have limited power, but groups of flip-flops taken with a generous amount of combinatorial logic can implement powerful *state machines*.

A four-bit synchronous binary up-counter is an example of a state machine. Between clock pulses, the four outputs of such a counter are in one of sixteen states, as shown in Figure 14. The outputs of the flip-flops are called the *state variables* because they unambiguously define the counter's states. In a state machine, no two states can have the same combination of state variables.

### Example

The counter mentioned above is implemented in an EP300 using A+PLUS and the schematic in Figure 16. Typical of most state machines, the counter consists of inputs driving a block of combinatorial logic, flip-flops storing the current state, and outputs. The outputs of the flip-flops are the state variables and, in this example, the outputs of the counter circuit as well. The CARRYOUT combinatorial output is a function of both the current state and the inputs. Note that both registered and combinatorial outputs are used since individual Macrocells of an EP300 can have different output configurations.

**Figure 14:**

The outputs of a four-bit binary counter form a set of state variables because they unambiguously define which state the counter is in.

STATE	D	C	B	A
S0	0	0	0	0
S1	0	0	0	1
S2	0	0	1	0
S3	0	0	1	1
S4	0	1	0	0
S5	0	1	0	1
S6	0	1	1	0
S7	0	1	1	1
S8	1	0	0	0
S9	1	0	0	1
S10	1	0	1	0
S11	1	0	1	1
S12	1	1	0	0
S13	1	1	0	1
S14	1	1	1	0
S15	1	1	1	1

## Design of a State Machine

The design of a state machine requires the following steps:

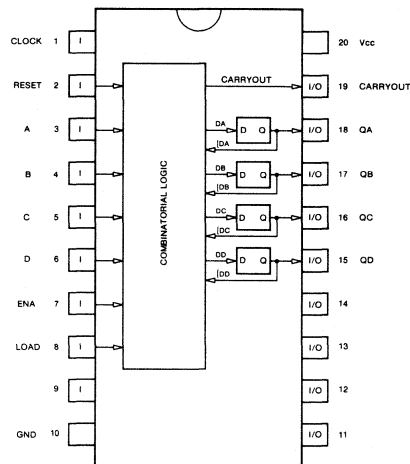
1. Draw a state diagram,
2. Derive the next-state table,
3. Assign state variables to the states, and
4. Derive the flip-flop excitation equations.

A logic designer traditionally does all these steps, but A+PLUS can help reduce errors in step 4 by deriving the flip-flop excitation equations.

Also, steps 1 through 3 usually are carried out without considering whether the state machine will be implemented using discrete logic, an EP300, or any other PLD. In the following example, however, Step 3 shows an exception to the rule. By considering the final implementation when assigning state variables, a designer can change a design that ordinarily does not fit in an EP300 into one that does.

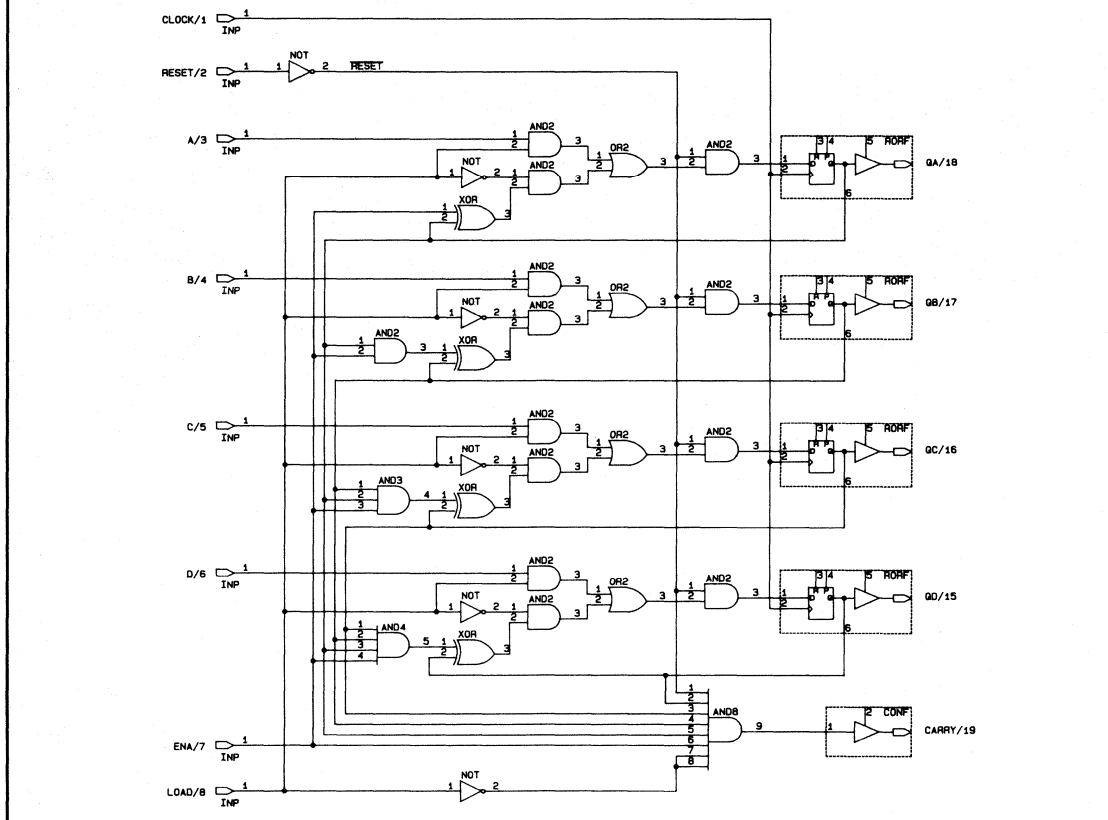
**Figure 15:**

Example of a state machine. Four Macrocells store the current state and provide binary outputs; a fifth provides a carry in.



**Figure 16:**

A+PLUS source schematic for a four-bit binary counter.



### Design Example: Counter plus 7-Segment Display

A seven-segment LED display can be connected to a 74160 synchronous 4-bit decade counter through a 7448 decoder-driver. The resulting circuit, when clocked, will repeatedly display the digits 0 through 9.

Consider implementing a circuit like this with a single EP300. To show the Ep300's power, change the specifications of the circuit to include a mode selector which allows the counter to count from 0 to 5, count from 0 to 9, or to count in hexadecimal. When connected to a seven segment common-cathode LED display, the counter will sequentially show the states in Figure 18. The inverses of the required outputs are shown in Figure 19. Following the design methodology presented above, these steps result.

**Step 1: Draw a state diagram.** Figure 20 shows the state diagram of the counter. The counter increments when the enable input ENA is true and RESET is false. If ENA is false, the counter holds its state regardless of the clock—except when RESET is high, in which case the counter goes directly to state S0.

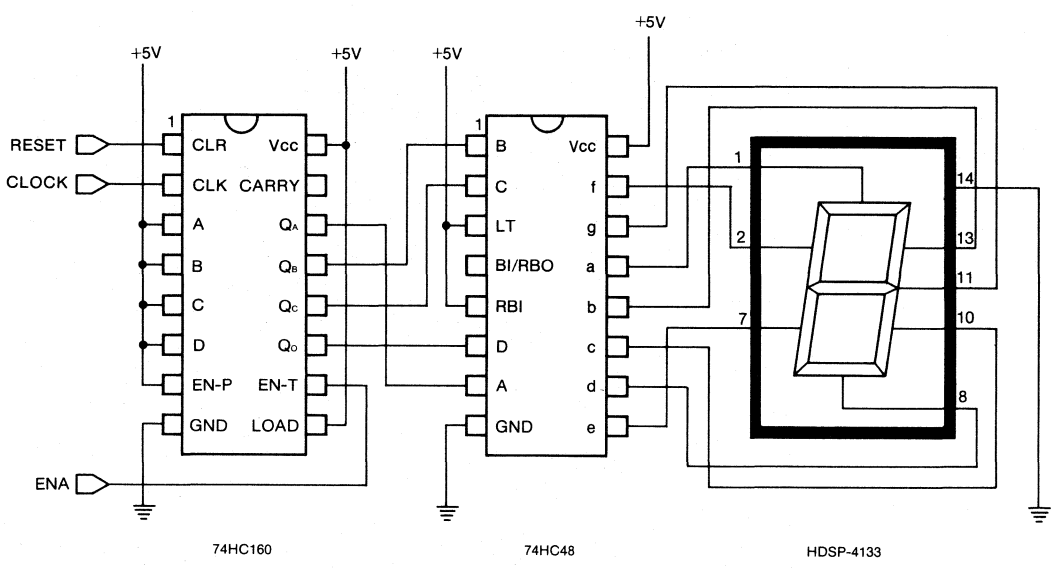
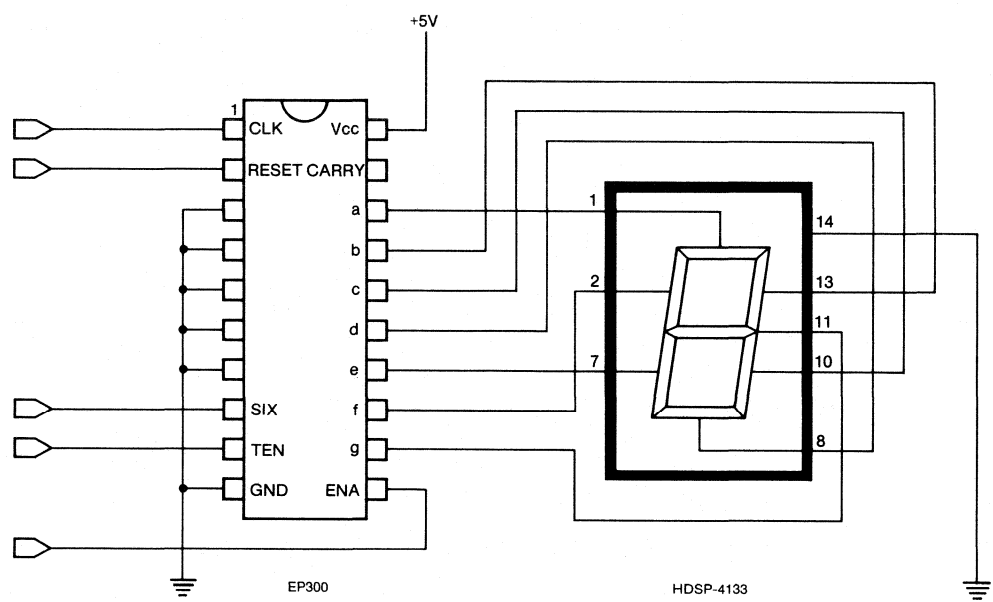
The inputs TEN and SIX cause the counter to cycle with a period shorter than 16 states. Note that since the D flip-flops of the EP300 are positive edge triggered, all state transitions occur on the rising edge of the clock pulse.

**Step 2: Derive the next state table.** With the state diagram completed, the next-state table can be derived as shown in the three rightmost columns of Figure 21. Although the next-state table contains the same information as the state diagram, it will be easier to derive the flip-flop excitation equations from the table.

**Step 3: Assign state variables to the states.** The state variables of the 16-state machine will *not* form the pattern in Figure 14. Instead, they will form the pattern in Figure 19 by defining the state variables to be the outputs themselves. With this done, the counter will not really "count," but will assume successive states which, when interpreted with a seven segment display, appear to be counting. This approach is taken in order to replace both a four-bit counter and a decoder with one EP300.

**Figure 17:**

Example of integrating many circuit functions into an EP300. The counter-decoder pair in the upper circuit can be replaced by one EP300. The flexibility of the EP300 allows logic-low inputs to be changed to logic-high, and lets the designer implement additional functions, such as counter mode selection.



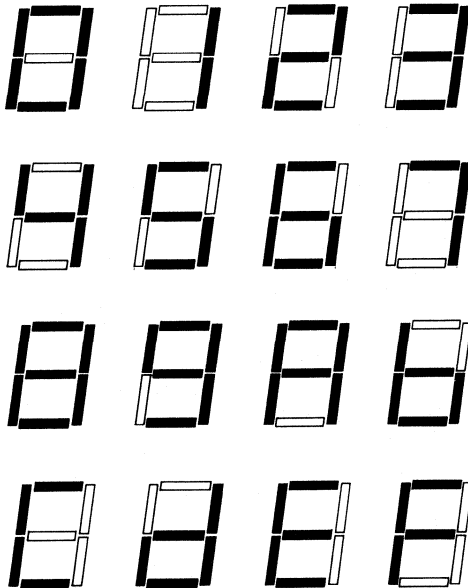
Consider implementing a circuit like this with a single EP300. To show the EP300's power, change the specifications of the circuit to include a mode selector which allows the counter to count from 0 to 5, to count from 0 to 9, or to count in hexadecimal. When connected to a seven segment common-cathode LED display, the counter will sequentially show the states in Figure 20. The inverses of the required outputs are shown in Figure 21. Following the design methodology presented above, these steps result.

**Step 1: Draw a state diagram.** Figure 22 shows the state diagram of the counter. The counter increments when the enable input ENA is true and RESET is false. If ENA is false, the counter holds its state regardless of the clock—except when RESET is high, in which case the counter goes directly to state S0. The inputs TEN and SIX cause the counter to cycle with a period shorter than 16 states. Note that since the D flip-flops of the EP300 are positive edge triggered, all state transitions occur on the rising edge of the clock pulse.

**Step 2: Derive the next state table.** With the state diagram completed, the next-state table can be derived as shown in the three rightmost columns of Figure 23. Although the next-state table contains the same information as the state diagram, it will be easier to derive the flip-flop excitation equations from the table.

**Figure 18:**

Sequential outputs of the LED display in the counter-decoder circuit at the bottom of Figure 17.



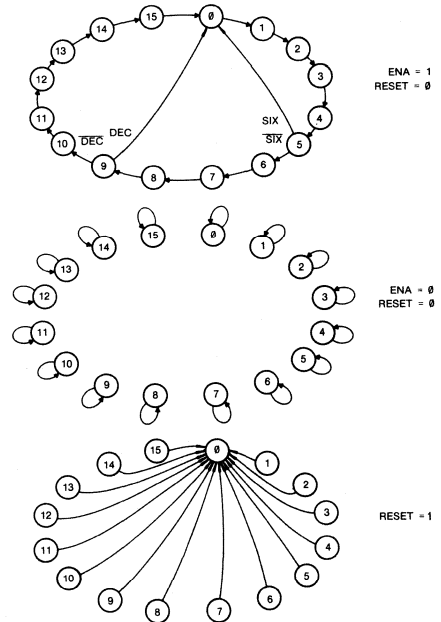
**Figure 19:**

Outputs and state variables of the sixteen-state counter-decoder. To simplify the design (which uses common-cathode displays) the outputs and state variables are defined as inverses of each other.

STATE	OUTPUTS / STATE VARIABLES						
	Q1	Q2	Q3	Q4	Q5	Q6	Q7
S0	0	0	0	0	0	0	1
S1	1	0	0	1	1	1	1
S2	0	0	1	0	0	1	0
S3	0	0	0	0	1	1	0
S4	1	0	0	1	1	0	0
S5	0	1	0	0	1	0	0
S6	0	1	0	0	0	0	0
S7	0	0	0	1	1	1	1
S8	0	0	0	0	0	0	0
S9	0	0	0	0	1	0	0
S10	0	0	0	1	0	0	0
S11	1	1	0	0	0	0	0
S12	0	1	1	0	0	0	1
S13	1	0	0	0	0	1	0
S14	0	1	1	0	0	0	0
S15	0	1	1	1	0	0	0

**Figure 20:**

State Diagram of the lower circuit in Figure 17.



**Step 4: Derive the flip-flop excitation equations.** The flip-flop excitation equations for the counter are shown in the Altera Design File on the following page.

There is no need to use Karnaugh maps to reduce the resulting equations since A+PLUS contains a boolean expander-minimizer.

**Figure 21:**

Next-state table of seven-segment counter/decoder. This table was taken directly from Figures 19 and 20.

OUTPUTS / STATE VARIABLES								TEN=0,SIX=0	TEN=1, SIX=0	TEN=0,SIX=1
STATE	Q1	Q2	Q3	Q4	Q5	Q6	Q7			
S0	0	0	0	0	0	0	1	S1	S1	S1
S1	1	0	0	1	1	1	1	S2	S2	S2
S2	0	0	1	0	0	1	0	S3	S3	S3
S3	0	0	0	0	1	1	0	S4	S4	S4
S4	1	0	0	1	1	0	0	S5	S5	S5
S5	0	1	0	0	1	0	0	S6	S6	S0
S6	0	1	0	0	0	0	0	S7	S7	S7
S7	0	0	0	1	1	1	1	S8	S8	S8
S8	0	0	0	0	0	0	0	S9	S9	S9
S9	0	0	0	0	1	0	0	S10	S0	S10
S10	0	0	0	1	0	0	0	S11	S11	S11
S11	1	1	0	0	0	0	0	S12	S12	S12
S12	0	1	1	0	0	0	1	S13	S13	S13
S13	1	0	0	0	0	1	0	S14	S14	S14
S14	0	1	1	0	0	0	0	S15	S15	S15
S15	0	1	1	1	0	0	0	S0	S0	S0

## CONCLUSION

The EP300 EPLD finds its applications under a wide range of topics. It can be used to implement state machine designs as well as to implement combinatorial logic, and its programmable architecture allows even more complex circuits to fit in one or a few EP300's.

The clever application of design "tricks" can change a design which is optimized for discrete logic into one optimized for EP300 Macrocells. Different types of flip-flops can be emulated, and state machines that ordinarily do not fit in an EP300 can be reworked to fit with ease. The logic designer using discrete TTL or CMOS devices must consider optimizing a design at the gate level, but designers using EP300's optimize at the Macrocell level, allowing more time for creative effort and experimentation.

This Application Note has presented only an introduction to the capabilities of the EP300. With its flexible Macrocell architecture, the EP300 ranks among the most powerful 20-pin PLD's and with its EPROM fabrication technology, the EP300 is the first erasable PLD of its kind.

## REFERENCES

1. D. L. Dietmeyer, *Logic Design of Digital Systems*. Boston: Allyn and Bacon, 1978.
2. C. H. Roth, Jr., *Fundamentals of Logic Design*. St. Paul, Minn.: West Publishing Co., 1979
3. National Semiconductor Corporation, *MM54HC/-74HC High Speed microCMOS Logic Family Data-book*. Santa Clara, Calif.
4. Altera Corporation, *Altera EP300 Programmable Logic Design System User Manual*. Santa Clara, Calif.

Figure 22:

ALTERA CORPORATION  
 12-07-84  
 SIX/TEN/HEX DISPLAY & CNTR  
 12  
 EP300  
 15.0MHz  
 PART:EP300

INPUTS:  
 CLOCK @ 1, RST @ 2, SIX @ 8, TEN @ 9, ENABLE @ 11

OUTPUTS:  
 OUTG @ 12, OUTF @ 13, OUTE @ 14, OUTD @ 15, OUTC @ 16, OUTB @ 17, OUTA @ 18, CARRY @ 19

NETWORK:  
 %INPUTS%  
 CLK = INP(CLOCK)  
 RESET = INP(RST)  
 MODE6 = INP(SIX)  
 MODE10 = INP(TEN)  
 ENA = INP(ENABLE)

%OUTPUTS%  
 CARRY = CONF(CARRYD,)  
 OUTA, A = RORF(SEGA, CLK, , , )  
 OUTB, B = RORF(SEGB, CLK, , , )  
 OUTC, C = RORF(SEGC, CLK, , , )  
 OUTD, D = RORF(SEGD, CLK, , , )  
 OUTE, E = RORF(SEGE, CLK, , , )  
 OUTF, F = RORF(SEGF, CLK, , , )  
 OUTG, G = RORF(SEGG, CLK, , , )

EQUATIONS:  
 %STATE VARIABLE DEFINITIONS%  
 S0 = /A \* /B \* /C \* /D \* /E \* /F \* G; %0%  
 S1 = A \* /B \* /C \* D \* E \* F \* G; %1%  
 S2 = /A \* /B \* C \* /D \* /E \* F \* /G; %2%  
 S3 = /A \* /B \* /C \* /D \* E \* F \* /G; %3%  
 S4 = A \* /B \* /C \* D \* E \* /F \* /G; %4%  
 S5 = /A \* B \* /C \* /D \* E \* /F \* /G; %5%  
 S6 = /A \* B \* /C \* /D \* /E \* /F \* /G; %6%  
 %S7 = /A \* /B \* /C \* D \* E \* F \* G; %7% %STATE 7 IS NOT USED%  
 S8 = /A \* /B \* /C \* /D \* /E \* /F \* /G; %8%  
 S9 = /A \* /B \* /C \* /D \* E \* /F \* /G; %9%  
 S10 = /A \* /B \* /C \* D \* /E \* /F \* /G; %A%  
 S11 = A \* B \* /C \* /D \* /E \* /F \* /G; %B%  
 S12 = /A \* B \* C \* /D \* /E \* /F \* G; %C%  
 S13 = A \* /B \* /C \* /D \* /E \* F \* /G; %D%  
 S14 = /A \* B \* C \* /D \* /E \* /F \* /G; %E%  
 S15 = /A \* B \* C \* D \* /E \* /F \* /G; %F%

MODE16 = /MODE6 \* /MODE10;

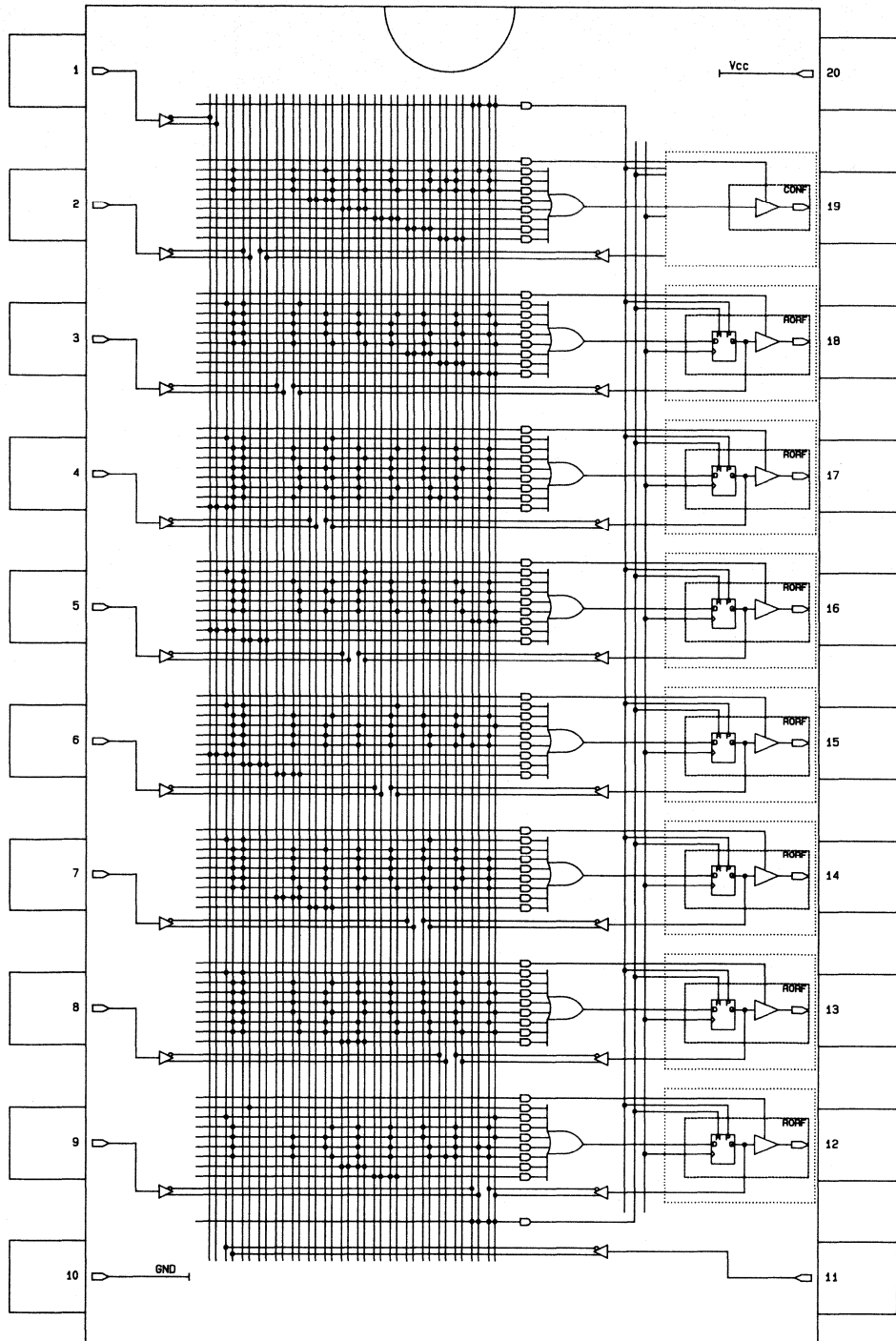
SEGA = /RESET\*( /ENA\*A + ENA\*(S0 + S3 + S10 + S12));  
 SEGB = /RESET\*( /ENA\*B + ENA\*(S4 + /MODE6\*S5 + S10 + S11 + S13 + S14));  
 SEGC = /RESET\*( /ENA\*C + ENA\*(S1 + S11 + S13 + S14));  
 SEGD = /RESET\*( /ENA\*D + ENA\*(S0 + S3 + S6 + /MODE10\*S9 + S14));  
 SEGE = /RESET\*( /ENA\*E + ENA\*(S0 + S2 + S3 + S4 + S6 + S8));  
 SEGF = /RESET\*( /ENA\*F + ENA\*(S0 + S1 + S2 + S4 + S6 + S12));  
 SEGG = RESET + /ENA\*G + ENA\*(S0 + MODE6\*S5 + S6 + MODE10\*S9 + S11);

CARRYD = ENA\*(MODE6\*S5 + MODE10\*S9 + MODE16\*S15);

ENDS



Figure 23:

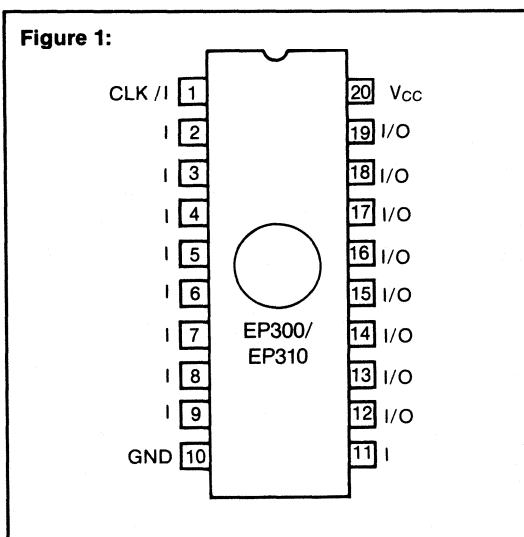


## INTRODUCTION

The ALTERA EP300/EP310 architecture can be configured to directly replace all functions that may be implemented with the 20 pin PAL™ family. The EP300/EP310 offers I/O control, making functional as well as pin to pin compatibility easily achievable. The EP300/EP310 is manufactured using a n-well CMOS technology, thus eliminating the high power requirements demanded by conventional fuse-programmable bipolar logic devices. In addition, the EP300/EP310 features UV erasability making it the first "reprogrammable" logic device. By offering this greater flexibility, a single EP300/EP310 can replace many types of logic devices that otherwise need to be purchased and inventoried.

## FUNCTIONAL OVERVIEW

The EP300/EP310, like PALs, implements sum of products logic using a programmable AND array whose outputs feed a fixed OR array. Shown below is the EP300/EP310 pin diagram. There are ten input pins which provide both true and complement signals into the AND array. Also, there are eight bidirectional pins which can be programmed as input, output, or bidirectional I/O.



Internally, the EP300/EP310 is divided into eight macrocells. Each macrocell can sum (logically OR) eight product terms. The result is directed to an Architecture Control Block which produces either

combinatorial or registered outputs, and an internal feedback path to the AND array. The feedback path may also be combinatorial or registered producing both true and complement signals. In addition, each macrocell contains a ninth product term dedicated to a tri-state buffer which drives the I/O pin. A 36 input AND array is generated from the ten input pins and eight feedback paths. Since the AND array is shared by all macrocells, each input and feedback signal is available to all 74 EP300/EP310 product terms. Other features include Synchronous Preset and Asynchronous Clear capability to all output registers, along with automatic power up clear.

**TABLE 1 EP300/EP310-PAL COMPARISON**

*Features of the EP300/EP310 and commercially available 20 pin PALs. All numbers represent maximum values. Some numbers will change depending on the specific application.*

FEATURE	EP300/	
	EP310	PAL-20
Array Logic	AND-OR	AND-OR
Current Consumption	40 mA	180 mA
Inputs	17	16
Outputs	8	8
Array Input Lines	36	32
Product Terms	74	64
D Flip-Flops	8	8
Tri-State Outputs	8	8
Preset Control	YES	NO
Reset Control	YES	NO
Reprogrammable	YES	NO

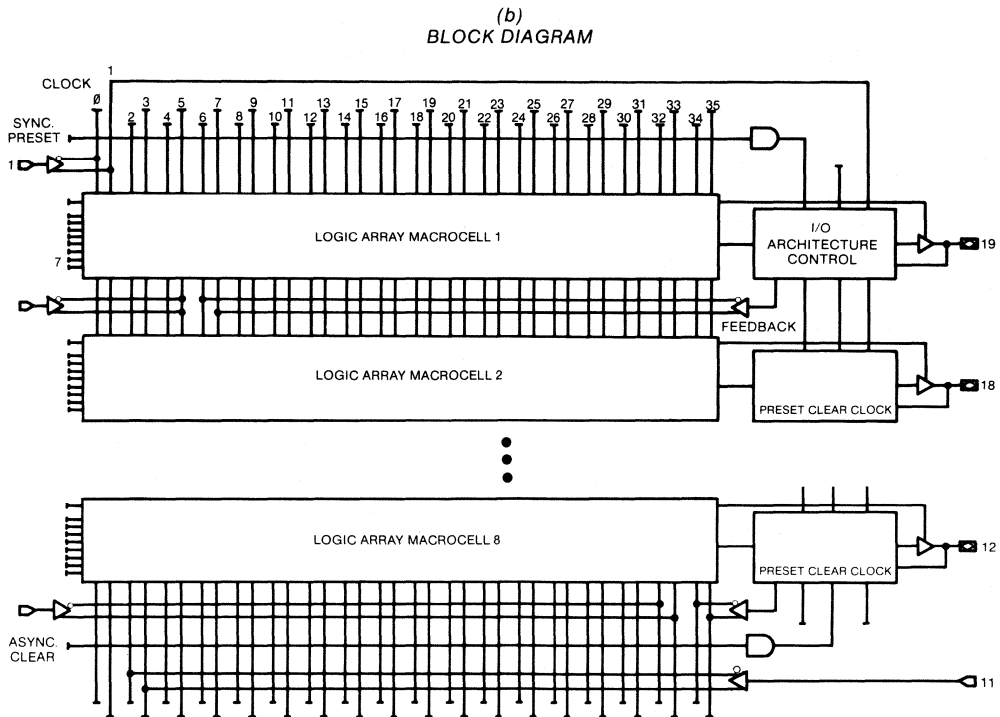
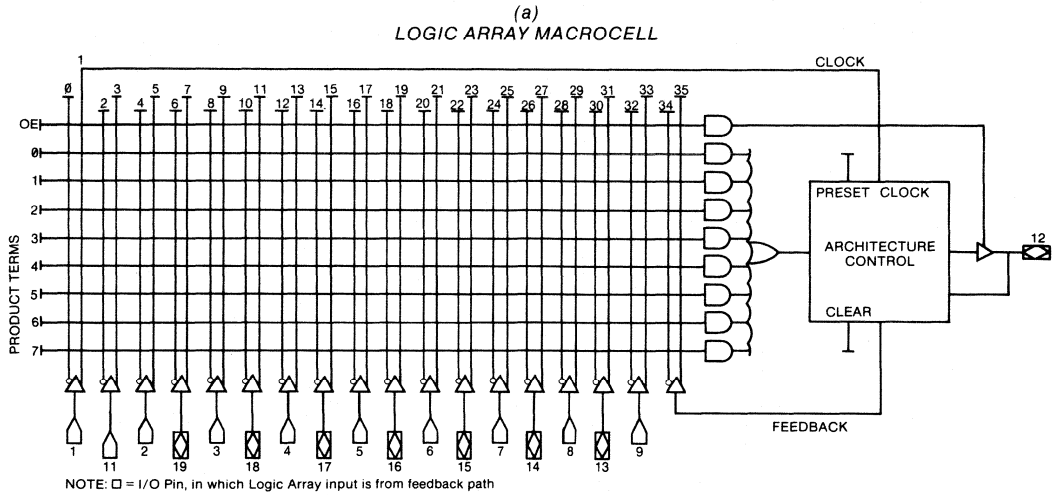
Clocking internal registers is accomplished through pin number one. The true signal is connected to all internal registers, see Figure 2. The registers are positive edge triggered meaning data transitions occur on the rising edge of the clock signal. If no clocking is required, pin one may be used as an additional input with both the true and complement signal available to the AND array.

## CONFIGURING THE EP300/EP310

Every macrocell in the EP300/EP310 contains its own I/O control (Figure 3). Any output can be configured as combinatorial (directly from OR gate), or registered (output through D-flipflop), programmed in either active high or low mode. Independent of output mode, the feedback into the AND array can be combinatorial, registered, I/O (directly from pin), or none.

**Figure 2: EP300/EP310 Macrocell and Block Diagram**

EP300/EP310 logic-array macrocell(a). Each cell includes 36 input lines and 9 product terms. Eight terms feed an 8-input OR gate, the remaining serves as an output enable control. Eight cells compose the complete EP300/EP310 logic array (b). Clock, Synchronous Preset and Asynchronous Clear are available to all output registers.

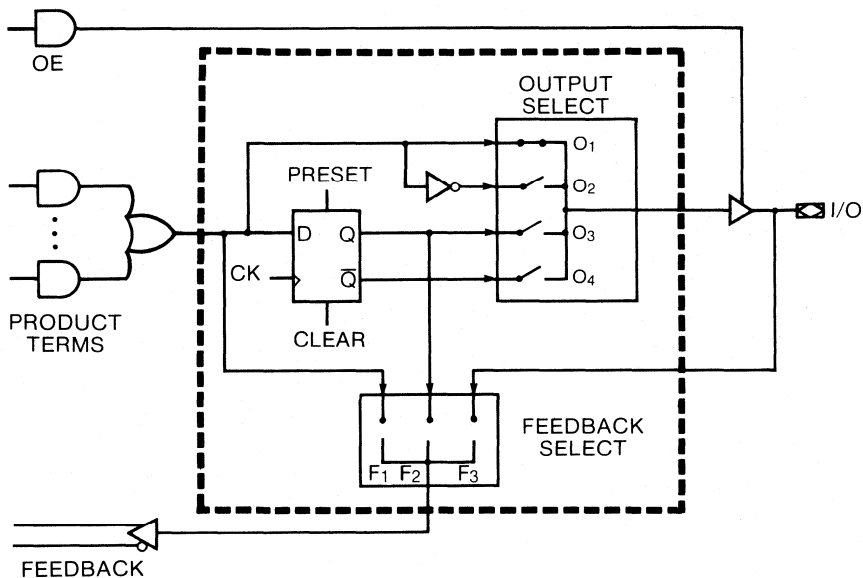


Combinatorial outputs are obtained by closing switches O1 and O2. Registered outputs can be selected by activating switch O3 or O4. Combinatorial feedback requires closing switch F1. Registered feedback is directed through F2. Closing F3 configures the macrocell as an external input or bidirectional I/O. For example, to obtain a registered output, registered feedback, active low output would require closing switches O4

and F2. Multiplexers or entire macrocells are disabled by leaving all respective switches open. To use an I/O pin as a dedicated input, the output is disabled by leaving switches O1-O4 open and closing F3 which connects the pin to the AND array. Figure 4 shows how to configure the EP300/EP310 to reproduce any common PAL output architecture.

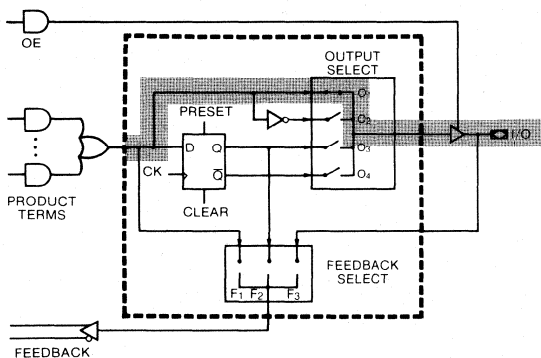
**Figure 3: EP300/EP310 I/O Architecture**

Output and feedback multiplexers provide up to 20 different operating conditions for each Sum term. The output multiplexer selects inverted or noninverted, registered or combinatorial outputs. The feedback multiplexer lets you apply either combinatorial, registered, or bidirectional I/O feedback.

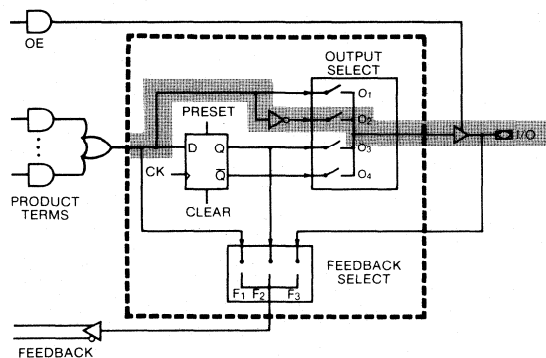


**Figure 4: I/O Configurations**

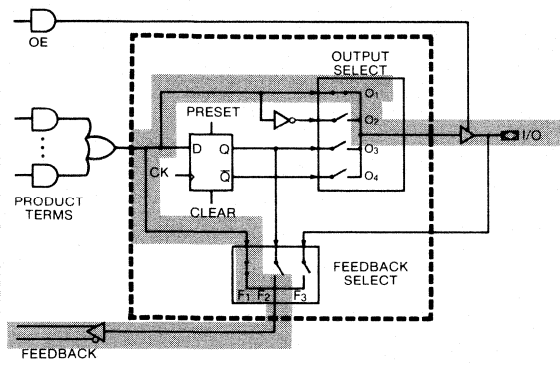
(a) Combinatorial output high. No feedback



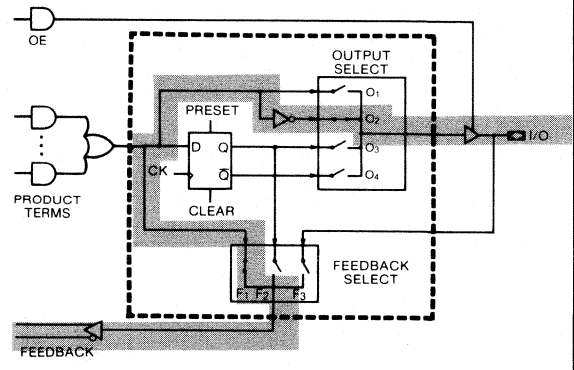
(b) Combinatorial output low. No feedback



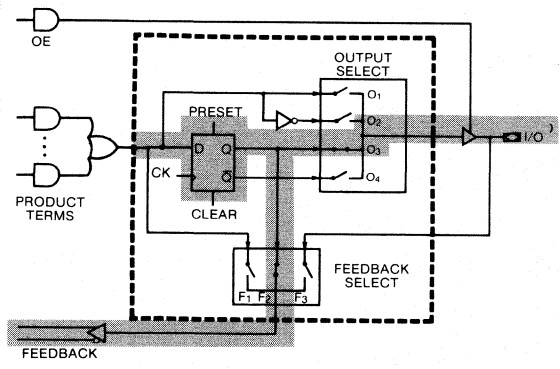
(c) Combinatorial output high. Combinatorial feedback



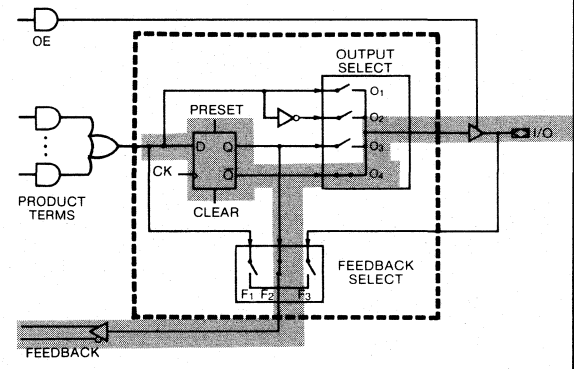
(d) Combinatorial output low. Combinatorial feedback



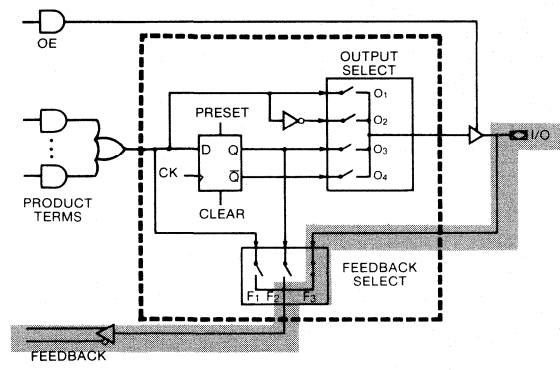
(e) Registered output high. Registered feedback



(f) Registered output low. Registered feedback



(g) Output used for additional input



(h) Bidirectional I/O

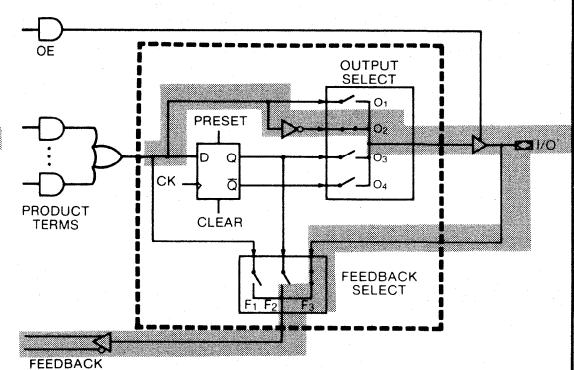


TABLE 2

## EP300/EP310 MULTIPLEXER SELECTION

PAL PART NUMBER	EP300/EP310		EP300/EP310 I/O CONTROL							OUTPUT	FEEDBACK	
	PIN NUMBER	MACROCELL NUMBER	O1	O2	O3	O4	F1	F2	F3			
10H8	12-19	1-8	●	○	○	○	○	○	○	○	Comb/High	—
10L8	12-19	1-8	○	●	○	○	○	○	○	○	Comb/Low	—
12H6	12	8	○	○	○	○	○	○	○	●	—	I/O
	13-18	2-7	●	○	○	○	○	○	○	○	Comb/High	—
	19	1	○	○	○	○	○	○	○	○	—	I/O
12L6	12	8	○	○	○	○	○	○	○	●	—	I/O
	13-18	2-7	○	●	○	○	○	○	○	○	Comb/Low	—
	19	1	○	○	○	○	○	○	○	○	—	I/O
14H4	12-13	7-8	○	○	○	○	○	○	○	●	—	I/O
	14-17	3-6	●	○	○	○	○	○	○	○	Comb/High	—
	18-19	1-2	○	○	○	○	○	○	○	●	—	I/O
14L4	12-13	7-8	○	○	○	○	○	○	○	○	—	I/O
	14-17	3-6	○	●	○	○	○	○	○	○	Comb/Low	—
	18-19	1-2	○	○	○	○	○	○	○	●	—	I/O
16C1	12-14	6-8	○	○	○	○	○	○	○	●	—	I/O
	15	5	○	●	○	○	○	○	○	○	Comb/Low	*
	16	4	●	○	○	○	○	○	○	○	Comb/High	*
	17-19	1-3	○	○	○	○	○	○	○	○	—	I/O
16H2	12-14	6-8	○	○	○	○	○	○	○	○	—	I/O
	15-16	4-5	●	○	○	○	○	○	○	○	Comb/High	—
	17-19	1-3	○	○	○	○	○	○	○	○	—	I/O
16L2	12-14	6-8	○	○	○	○	○	○	○	○	—	I/O
	15-16	4-5	○	●	○	○	○	○	○	○	Comb/Low	—
	17-19	1-3	○	○	○	○	○	○	○	○	—	I/O
16H8	12	8	●	○	○	○	○	○	○	○	Comb/High/Z	—
	13-18	2-7	●	○	○	○	○	○	○	○	Comb/High/Z	Comb
	19	1	●	○	○	○	○	○	○	○	Comb/High/Z	—
16L8	12	8	○	●	○	○	○	○	○	○	Comb/Low/Z	—
	13-18	2-7	○	●	○	○	○	○	○	○	Comb/Low/Z	Comb
	19	1	○	●	○	○	○	○	○	○	Comb/Low/Z	—
16R4	12-13	7-8	○	●	○	○	○	○	○	○	Comb/Low/Z	Comb
	14-17	3-6	○	○	○	○	○	○	○	○	Reg/Low/Z	Reg
	18-19	1-2	○	○	○	○	○	○	○	○	Comb/Low/Z	Comb
16R6	12	8	○	●	○	○	○	○	○	○	Comb/Low/Z	Comb
	13-18	2-7	○	○	○	○	○	○	○	○	Reg/Low/Z	Reg
	19	1	○	○	○	○	○	○	○	○	Comb/Low/Z	Comb
16R8	12-19	1-8	○	○	○	●	○	○	○	Reg/Low/Z	Reg	
16P8	12	8	E	E	○	○	○	○	○	○	Comb/Option/Z	—
	13-18	2-7	E	E	○	○	○	○	○	○	Comb/Option/Z	Comb
	19	1	E	E	○	○	○	○	○	○	Comb/Option/Z	—
16RP4	12-13	7-8	E	E	○	○	○	○	○	○	Comb/Option/Z	Comb
	14-17	3-6	○	○	E	E	○	○	○	○	Reg/Option/Z	Reg
	18-19	1-2	E	E	○	○	○	○	○	○	Comb/Option/Z	Comb
16RP6	12	8	E	E	○	○	○	○	○	○	Comb/Option/Z	Comb
	13-18	2-7	○	○	E	E	○	○	○	○	Reg/Option/Z	Reg
	19	1	E	E	○	○	○	○	○	○	Comb/Option/Z	Comb
16RP8	12-19	1-8	○	○	E	E	○	○	○	Reg/Option/Z	Reg	

E = Connection Optional  
● = Connection Closed

○ = Connection Open  
Z = High Impedence

\* = Maximum of Seven Product Terms

Proper I/O selection for each macrocell, represented above by external pin number, allows the EP300/EP310 to be functional as well as pin to pin compatible with all 18 commonly used 20 pin PALs.

The EP300/EP310 can be programmed to directly replace any logic function or I/O configuration that can be implemented with the 20 pin PAL family, (excluding the PAL 16X4 AND 16A4). Table 2 shows how to configure the EP300/EP310 I/O to obtain functional as well as pin to pin compatibility with all 18 commonly used PALs. For each PAL, listed by part number, the table lists the correct I/O connections required by each macrocell. It also indicates the output and feedback characteristics of each I/O pin. Macrocell 1 is always associated with pin 19, macrocell 2 with pin 18,...., and macrocell 8 with pin 12. The column labeled OUTPUT explains the type of output used by the respective pin(s), either combinatorial or registered, and if it is active high or low. Active high outputs correspond to true outputs of the EP300/EP310—switches O1 or O3. Active low outputs are associated with complement outputs—switches O2 or O4. If the column contains "OPTION" then the PAL can be programmed either active high or low. Here the corresponding I/O switches are marked with an "E" meaning either connection may be made (depending on the particular function desired). Only one switch in each multiplexer may be closed. A "Z" indicates that those outputs have an optional three state output. The column labeled FEEDBACK identifies if the macrocell feedback, if any, is from combinatorial or registered logic or if the pin(s) are used as a dedicated input. Though bidirectional I/O configurations are not shown, it simply requires closing switch F3 and the desired output switch. When using the macrocells as bidirectional I/O, the feedback switches F1 and F2 must be left open and output must tristate when the pin is used as an input.

As an example, suppose we wish to replace the PAL 16R4 with a low power, reprogrammable EP300/EP310. From the table, we notice that macrocells 1, 2, 7 and 8 require combinatorial active low outputs and combinatorial feedback. This requires closing switch O2 and F1. Macrocells 3, 4, 5 and 6 need registered

active low outputs with registered feedback, therefore, we close switch O4 and F2. The table also tells us that three state outputs may be used. The EP300/EP310 can replace any three state logic offered by a 20 pin PAL.

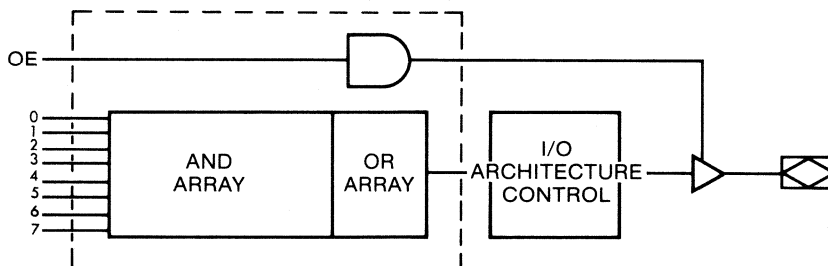
## OUTPUT ENABLE

Every output of the EP300/EP310 has an available three state output control. A separate product term is contained in each macrocell specifically for Output Enable functions. When the product term is asserted (HIGH) the output associated with that macrocell will be enabled. Output enable functions are implemented directly from the AND array. Therefore, it can be programmed active high or low or be conditionally asserted from any of the selected inputs or feedback lines. For combinatorial outputs, the EP300/EP310 and PAL have similar three state output implementation. Each dedicate a product term to control the three state output buffer. The only difference is the PAL uses one of its eight product terms to control the buffer. The result is a seven input OR gate for these outputs. The EP300/EP310 provides an additional product term, thus leaving its eight input OR gate intact.

Registered output PALs have their Output Enable function hard wired to pin 11 allowing only active low operation. If desired, the EP300/EP310 Output Enable can remain exactly compatible with the PAL. An example is shown in Figure 6. The complement signal of pin 11 is connected to each of the EP300/EP310 Output Enable product terms thus providing an active low operation. To finish our example for the 16R4, we would connect the complement signal of pin 11 to each Output Enable product term in macrocells 3, 4, 5 and 6 (registered outputs). For combinatorial outputs (macrocells 1, 2, 7 and 8) we directly reproduce the Output Enable control by examining the respective product term connections.

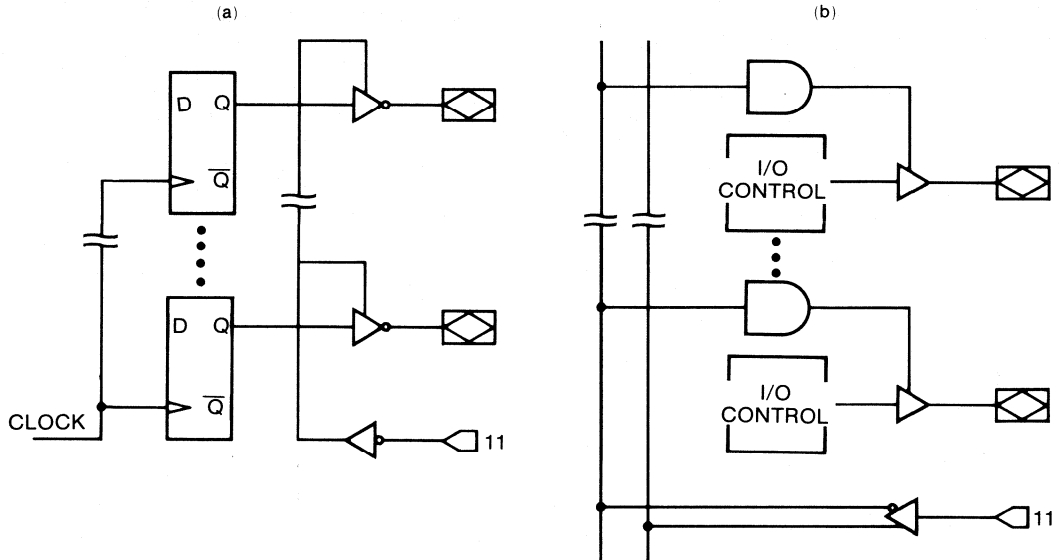
**Figure 5: EP300/EP310 Output-Enable Feature**

*Each macrocell of the EP300/EP310 contains its own Output Enable product term. Output enable functions are implemented directly from the AND array. When the product term is asserted low, the output pin is seen as high impedance.*



**Figure 6: Output Enable Example**

Output Enable control for registered output PAL (a). Duplicate function using EP300/EP310 (b). Both are configured active low.

**ADDITIONAL FEATURES**

The EP300/EP310 not only will directly replace a PAL, it also provides features not offered by the 20 pin family. Such features are Synchronous Preset and Asynchronous Clear. When the Synchronous Preset product term is asserted (HIGH), all the output registers are Preset (loaded with a HIGH) on the next rising edge of the clock. The Preset command can be asserted through one input, active high or low, several inputs, feedback lines or a combination of input and feedback signals. Several examples of a Preset implementation are given in Figure 7. For each case the variables are logically ANDed with the true signal of the clock. When the Asynchronous Clear product term is asserted (HIGH), all output registers will immediately be set to a LOW (independent of clock). Like the Preset, the Clear can be asserted by various inputs, feedbacks, or a combination of both. Examples of Clear are shown in Figure 8. An Asynchronous Clear always overrides a Synchronous Preset. On power up, the EP300/EP310 performs the Clear function (sets all registers to LOW) automatically. In addition, other key features of the EP300/EP310 are reduced power consumption and the ability to reprogram a given device.

**CONCLUSION**

With its AND/OR array, user defined I/O architecture, three state outputs, and registered Preset and Clear capability, the EP300/EP310 can directly replace the 20 pin PAL family and many other types of conventional and programmable logic devices. The advantage of CMOS low power, "reprogrammable" technology will help decrease development and production cost and at the same time increase device and final product reliability. Packaged in a space saving SLIMDIP (300 mil) 20 pin package the EP300/EP310 will prove to be a more effective solution for many applications.

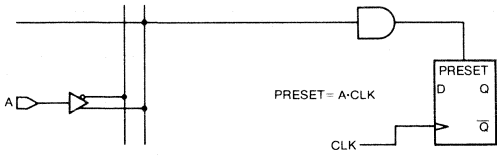
**REFERENCES**

1. ADVANCED MICRO DEVICES, INC,  
*Programmable Array Logic Handbook*.  
Sunnyvale, Calif., 1983
2. ALTERA CORPORATION,  
*EP300 Data Sheet, Rev. 2.0*.  
Santa Clara, Calif., 1984
3. MONOLITHIC MEMORIES, INC,  
*PAL Handbook, 3rd Edition*,  
Santa Clara, Calif., 1983

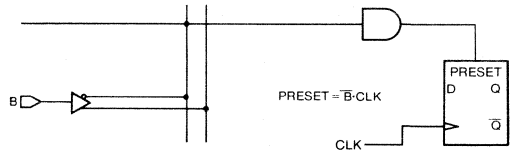


**Figure 7: Synchronous Preset Examples**

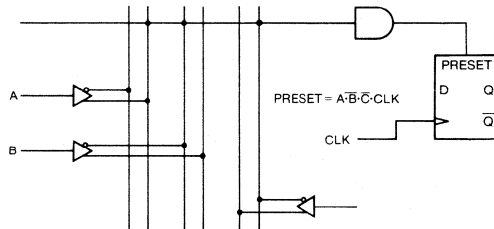
(a) One variable, active high preset command



(b) One variable, active low preset command

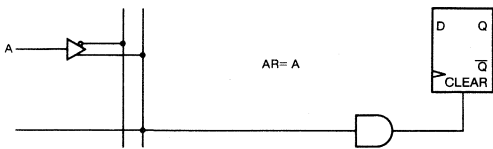


(c) Conditional logic preset command

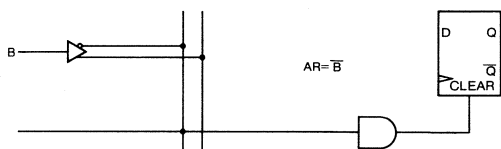


**Figure 8: Asynchronous Clear Examples**

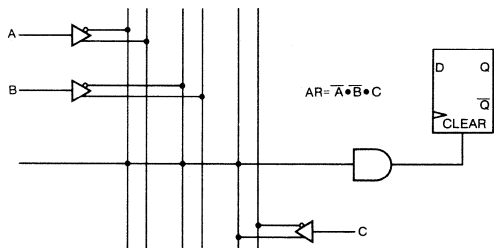
(a) One variable, active high clear command



(b) One variable, active low clear command

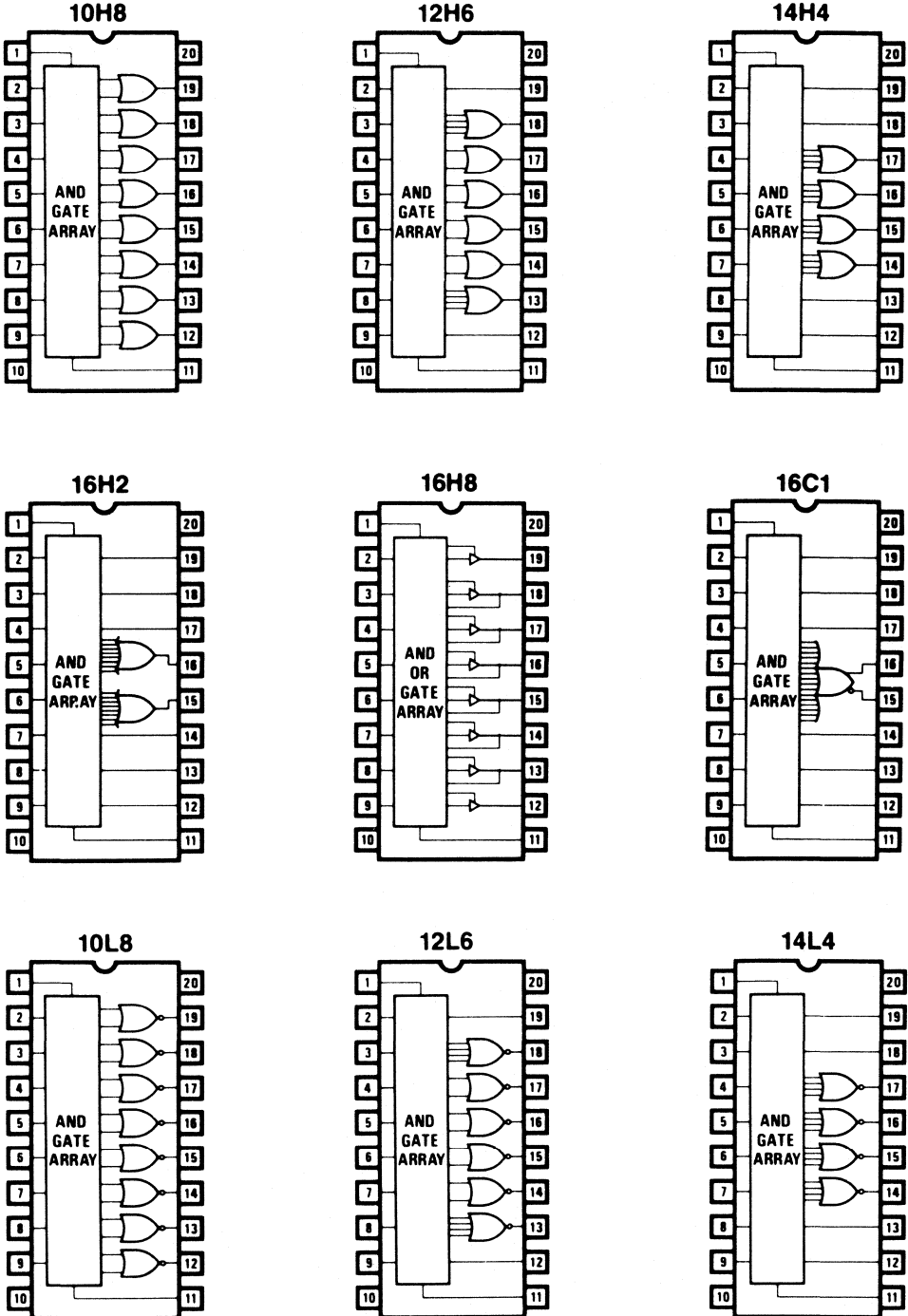


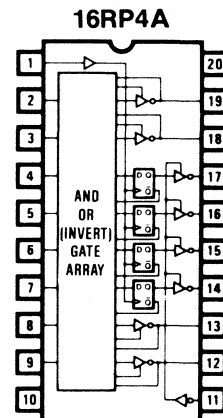
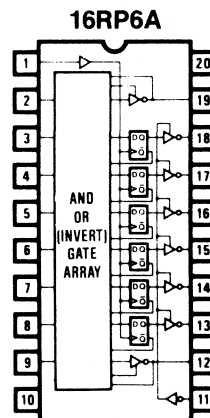
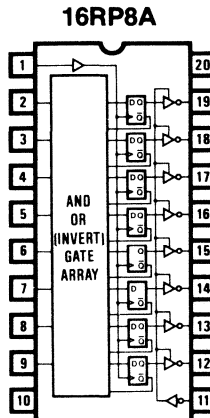
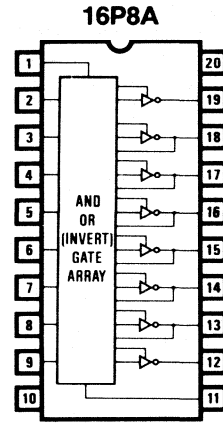
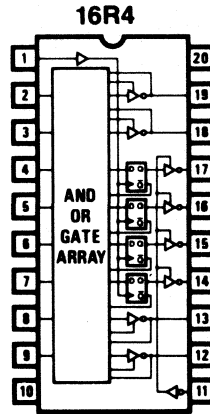
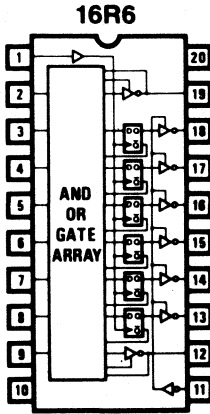
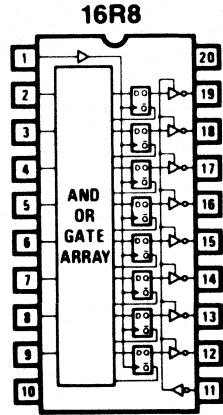
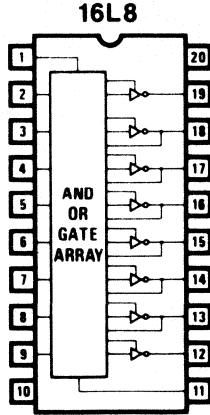
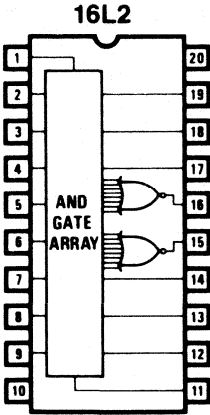
(c) Conditional logic clear command



**Figure 9: Logic Symbols**

The EP300/EP310 can replace all 18 commonly used 20 pin PAL members.





### INTRODUCTION

Microcomputer address decoding for memory and peripheral devices, traditionally done with discrete logic or ROM's, is being done more effectively with user programmable EPLD's (erasable-programmable logic devices.) The advantages of programmable logic include faster decode times and decreased PC board space. Disadvantages, such as increased power consumption, are eliminated with the use of CMOS based EPLD's.

This Application Note presents address decoding and memory control solutions using Altera CMOS EPLD's. The Intel 8088 and the Motorola 68000, examples of microprocessors with synchronous and asynchronous data busses respectively, are connected to various types of memory and peripheral devices. Features of the Altera programmable logic design program A+PLUS are reviewed as well.

### ADDRESS DECODING WITH PLD'S

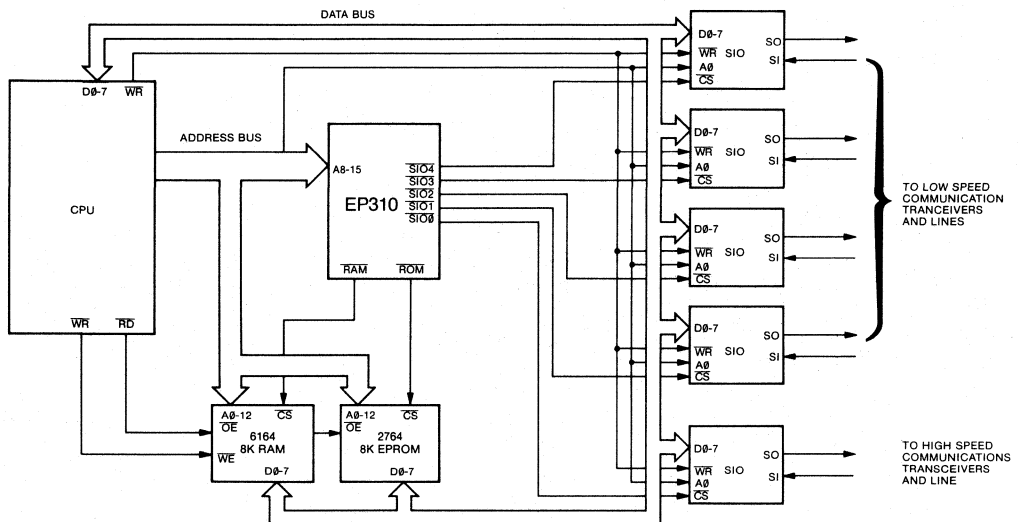
The circuit in Figure 1 shows an EP310 EPLD providing chip-select signals in a high-speed serial data line multiplexer. Control comes from an 8088 microprocessor whose program is stored in a 2764 EPROM. A 6164 static RAM stores data packets as they are being assembled and stores the 8088's stack, while five SIO (serial input/output) peripheral chips provide an interface between the 8088 CPU and the serial data lines.

The EP310 EPLD in Figure 1 decodes the address lines of the 8088 into chip select signals for the CPU's peripherals (RAM, ROM and SIO). The EP310's outputs follow the memory map in Figure 2, where the accompanying Boolean equations describe the memory map for a 16-bit address bus.

The EP310 is programmed with the aid of either

**Figure 1.**

*Block diagram of the serial data multiplexer using an EP310. The ROM, RAM and SIO's are controlled by an EP310 EPLD that is programmed to generate chip select pulses from the CPU's address lines A8-A15. Programming is done with either Alterans or A+PLUS software packages using the equations in Figure 2.*



the Altera computer design programs: Alterans or A+PLUS. Alterans and A+PLUS transform the equations in Figure 2 into a data file understood by logic device programmers (JEDEC council JC-42.1 standard N-3.) When the JEDEC file is used by a logic device programmer to program an Altera EPLD, the programmed chip will act just as described in the original specification—which in this case is the set of equations in Figure 2.

More complicated memory maps can be imple-

mented by changing the equations in Figure 2. The BASIC program in Figure 9 helps here by deriving equations for complicated memory maps.

### WAIT STATE GENERATION WITH PLD'S

It is inefficient to run a microprocessor at the speed of its slowest peripheral chip. Static RAM's, for example, can access data many times faster than

**Figure 2.**

Memory map of the EP310 address decoder in Figure 1 along with the equations used to program the EP310. Both Altera software packages Alterans and A+PLUS can convert Boolean

equations into a JEDEC standard file for logic programmers. The equations can be derived intuitively, or the BASIC program in Figure 9 can be used.

SIGNAL NAME	LOW ADDRESS	HIGH ADDRESS	EQUATION
ROM	0000	1FFF	$= \overline{A15} \bullet \overline{A14} \bullet \overline{A13}$
RAM	2000	3FFF	$= \overline{A15} \bullet \overline{A14} \bullet A13$
SIO0	8000	80FF	$= \overline{A15} \bullet \overline{A14} \bullet \overline{A13} \bullet \overline{A12} \bullet \overline{A11} \bullet \overline{A10} \bullet \overline{A9} \bullet \overline{A8}$
SIO1	8100	81FF	$= \overline{A15} \bullet \overline{A14} \bullet \overline{A13} \bullet \overline{A12} \bullet \overline{A11} \bullet \overline{A10} \bullet \overline{A9} \bullet \overline{A8}$
SIO2	8200	82FF	$= \overline{A15} \bullet \overline{A14} \bullet \overline{A13} \bullet \overline{A12} \bullet \overline{A11} \bullet \overline{A10} \bullet A9 \bullet \overline{A8}$
SIO3	8300	83FF	$= \overline{A15} \bullet \overline{A14} \bullet \overline{A13} \bullet \overline{A12} \bullet \overline{A11} \bullet \overline{A10} \bullet A9 \bullet \overline{A8}$
SIO4	8400	84FF	$= \overline{A15} \bullet \overline{A14} \bullet \overline{A13} \bullet \overline{A12} \bullet \overline{A11} \bullet A10 \bullet A9 \bullet \overline{A8}$

**Figure 3.**

Wait state generator and address decoder on a chip. The EP310 communicates with the 68000 over the signals on the left, and with peripherals over the chip select signals on the right. The address decoder half of the chip supports two modes of operation: normal operation (MODE1) and a special mode enabled upon CPU RESET (MODE 0). MODE 0 places a

copy of ROM in the first 64K of RAM, allowing the 68000 to jump to its boot routine. The first 64K of RAM, which is hidden by ROM in MODE 0, is moved to an unused area of the memory map thereby allowing the boot routine to access it immediately after RESET. MODE 1 operation is set by accessing a memory location from 70000 hex to 7FFFF hex.

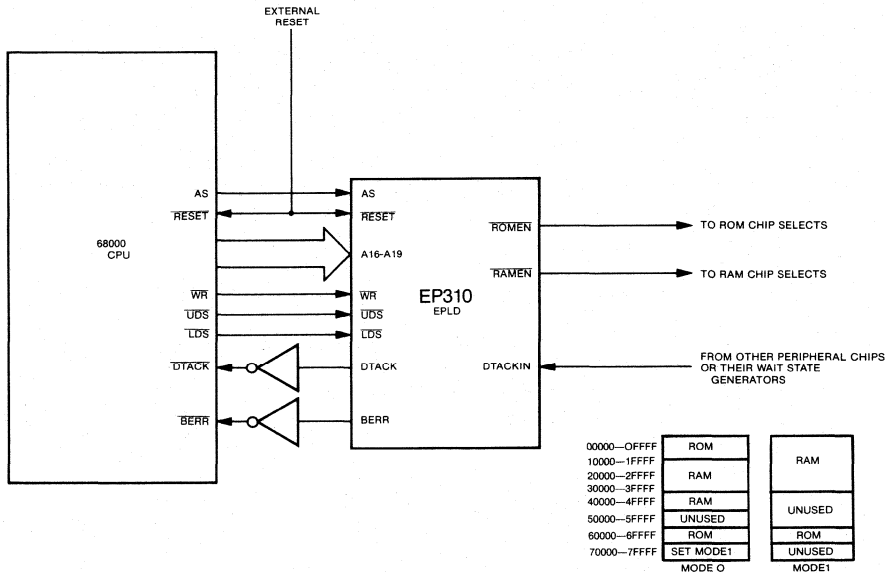
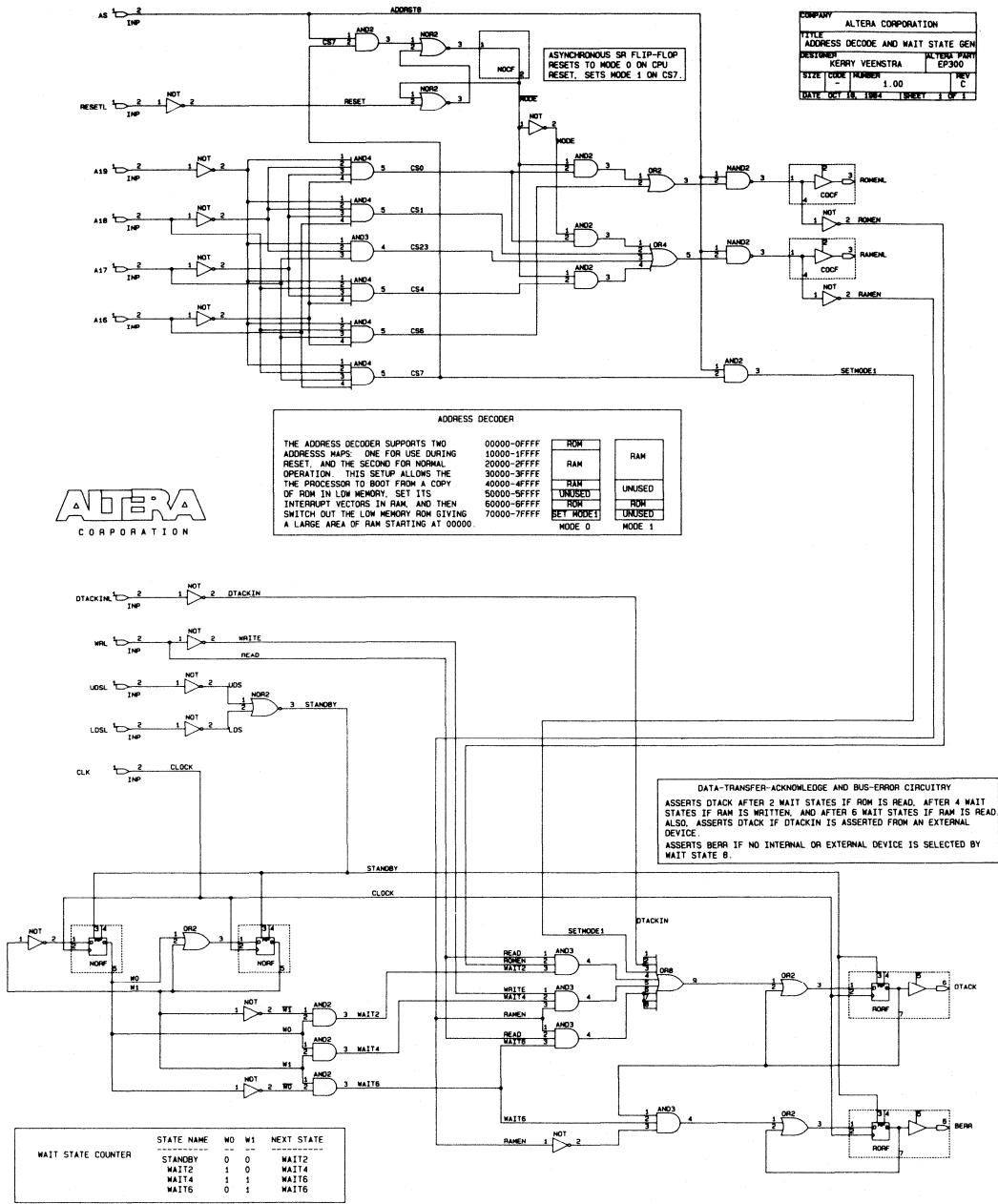


Figure 4.

This is a reproduction of the actual source schematic used by A+PLUS to program the EP310 wait state generator in Figure 3. A+PLUS converts the netlist of the schematic into an equivalent two-level form that fits in the EP310's AND-OR array architecture, then derives from it a JEDEC standard file for a logic device programmer. The primitives in the schematic fall into three groups: input pins, logic gates and output macrocells. The four character codes in the output macrocells

(RORF, NORF, etc.) tell A+PLUS the configuration of the macrocell. For example RORF means Registered-Output / Registered-Feedback, COCF means Combinatorial-Output / Combinatorial-Feedback. A+PLUS converts the four character mnemonics into output architecture programming information in the EP310 JEDEC file. The NOCF (No-Output / Combinatorial-Feedback) macrocell at the top of the schematic is used as a delay element in the NOR gate SR flip-flop.



UART's, but a well tuned microprocessor design does not run its static RAM at the speed of its slowest UART. Instead, the microprocessor is clocked at its top speed and a wait state generator slows the microprocessor's bus cycles when the UART is accessed.

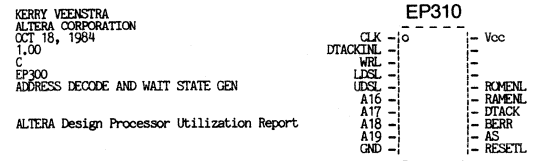
Wait states in a 68000 circuit are easy to generate by delaying the DTACK (data transfer acknowledge) signal. Two or three flip-flops can count the number of clock pulses after the assertion of either of the 68000's data select lines, LDS and UDS, and assert DTACK once a programmed count is reached. The programmed terminal count can depend on the speed of the selected peripheral device, allowing different numbers of wait states for peripherals of different speeds.

Figure 3 shows an EP310 programmed as such a wait state generator. The EP310 provides two chip-select signals, one for a 256K bank of dynamic RAM and the other for a 64K bank of ROM. The DTACK signal is asserted one clock pulse after ROM is selected, providing the ROM with two 68000 wait states, while RAM is given four or six wait states depending on the status of the WR signal. The result is a 68000 circuit that will work with inexpensive RAM's, but will also perform well when accessing fast program ROM.

As an element of added security, the 68000's bus error line (BERR) is asserted if neither RAM nor ROM is selected and no device asserts the DTACKIN signal by wait state 8. This feature is useful in industrial control applications to signal a controller fault since a CPU accesses non-existent resources only when it is operating improperly.

Figure 5.

A+PLUS produced this Design Processor Utilization Report when it converted Figure 4 into a JEDEC file for a logic programmer. The report gives the pinout that a chip will have when it is programmed with the associated JEDEC file. Other information in the report can be used to help partition several smaller designs into a larger Altera chip.



\*\*INPUTS\*\*

Name	Pin	Resource	MCell	PTerms	MCell	Feeds:	OE	Reset	Preset
CLK	1	CKR	-	-	-	-	-	-	-
DTACKIN	2	INP	-	-	6	-	-	-	-
WRL	3	INP	-	-	6	-	-	-	-
LDSL	4	INP	-	-	-	-	1	-	-
UDSL	5	INP	-	-	-	-	1	-	-
A16	6	INP	-	-	1	-	-	-	-
					4				
					5				
					6				
A17	7	INP	-	-	1	-	-	-	-
					4				
					5				
					6				
A18	8	INP	-	-	1	-	-	-	-
					4				
					5				
					6				
A19	9	INP	-	-	1	-	-	-	-
					4				
					5				
					6				
RESETL	11	INP	-	-	1	-	-	-	-
AS	12	INP	8	0/ 8	1	-	-	-	-
					4				
					5				
					6				

\*\*OUTPUTS\*\*

Name	Pin	Resource	MCell	PTerms	MCell	Feeds:	OE	Reset	Preset
BERR	13	RORF	7	2/ 8	7	-	-	-	-
DTACK	14	RORF	6	6/ 8	6	-	-	-	-
					7				
RAMENL	15	COCF	5	4/ 8	6	-	-	-	-
					7				
ROMENL	16	COCF	4	2/ 8	6	-	-	-	-

\*\*BURIED REGISTERS\*\*

Name	Pin	Resource	MCell	PTerms	MCell	Feeds:	OE	Reset	Preset
	19	NOCF	1	2/ 8	1	-	-	-	-
					4				
					5				
	18	NORF	2	1/ 8	3	-	-	-	-
					6				
					7				
	17	NORF	3	2/ 8	2	-	-	-	-
					3				
					6				
					7				

\*\*UNUSED RESOURCES\*\*

Name	Pin	Resource	MCell	PTerms
All Resources used				

\*\*PART UTILIZATION\*\*

100%	Pins
87%	MacroCells
25%	PTerms

**A+PLUS SCHEMATIC CAPTURE**

A+PLUS can convert a schematic netlist into a JEDEC device programming file without a designer first converting the design to Boolean equations. A possible implementation of the EP310 wait state generator is shown in schematic form in Figure 4. The schematic is created from Altera design primitives, which include input pins (INP), logic gates (such as AND2, OR4), and output macrocells (NOCF, COCF, NORF and RORF). A+PLUS reads the netlist generated by this schematic, converts the logic gates into Boolean equations, reduces each equation to its minimal sum-of-products form (the type of equation that fits in the target chip's AND-OR array), and finally fits the design into the target chip, writing a documentation file of the resulting pinout and a JEDEC file for programming a chip. (Figure 5.)

Changes can be made to a schematic or to its netlist, and A+PLUS can be used again to create corresponding documentation and JEDEC files. If inexpensive EPROM were desired in the application above, the wait-state counter in Figure 4 might be changed to give more wait states for the slower EPROM accesses. Simply delete the AND2 and NOT

gates that generate the signal WAIT2, and connect the dangling input which results to either of the signals WAIT4 or WAIT6. The speed of RAM accesses may be changed in a similar manner—without changing any other part of the design.

## DYNAMIC RAM CONTROL

Dynamic RAM circuits must generate several control signals: RAS, CAS, and WR for the DRAM's themselves, MAS (memory address select) for the address multiplexer, and a DTACK or READY line to acknowledge the data transfer to the CPU. DRAM controllers connected to some 16-bit microprocessors must allow 8-bit data transfers over either half of a 16-bit data bus (e.g. 8086 and 68000.) In addition, DRAM's must be refreshed—either once each bus cycle or in burst mode—a function that can be provided by either the DRAM controller, a DMA, or the CPU.

A DRAM controller that handles all of the above is presented schematic form in Figure 6. As with all DRAM controllers, an EP310 programmed as Figure 6 must generate each bus cycle signals that

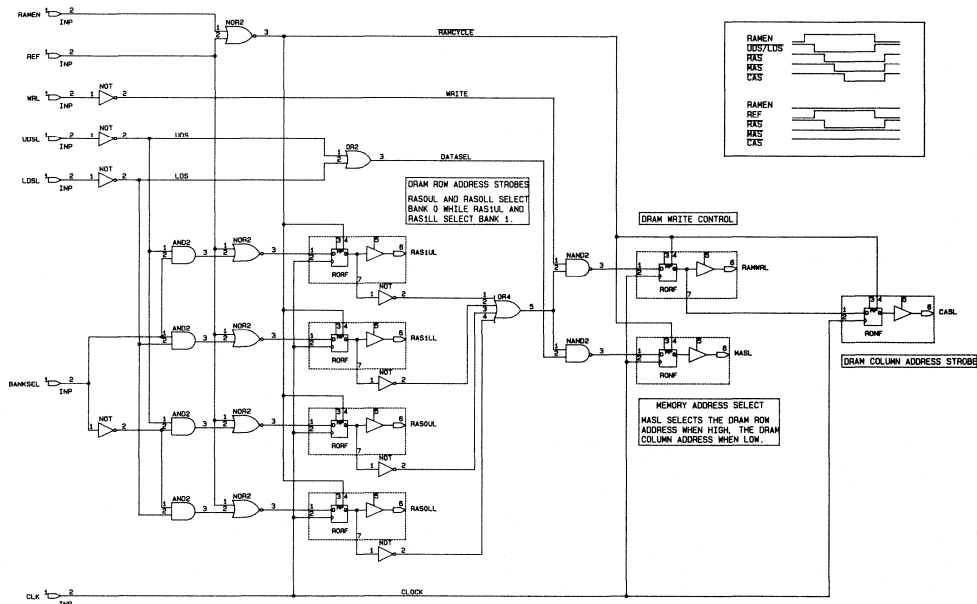
perform the following actions:

1. Place one half of the CPU address on the DRAM's address lines (negate MAS.)
2. Assert the Row Address Strobe.
3. (a) Place the second half of the CPU address on the DRAM's address lines (assert MAS.)  
(b) Assert the DRAM's WR line if the bus transfer is a write operation.
4. Assert the Column Address Strobe.
5. Wait for the bus cycle to terminate, then negate all signals.

An EP310 programmed as Figure 6 remains in a state with all signals negated (number 1 above) until a bus cycle is started with RAMEN and either UDS or LDS. The EP310 then cycles through the actions above at a rate set by the clock, stopping at number 5 until the bus cycle ends (RAMEN, UDS and LDS negate), then returns to number 1 to await the next bus access.

**Figure 6.**

*DRAM controller schematic used by A+PLUS to program an EP300. To operate properly, the controller's clock frequently must be exactly twice that of the 68000 connected to it.*



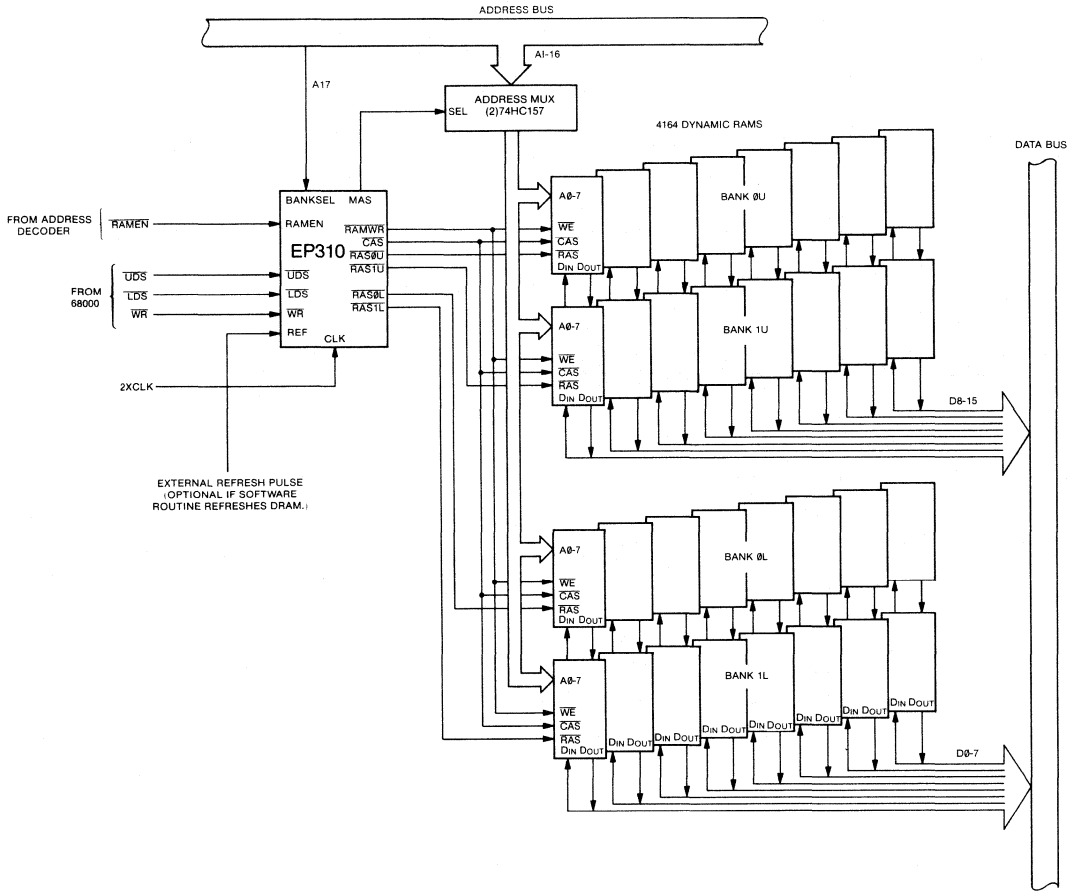
**ALTERA**  
CORPORATION

COMPANY	ALTERA CORPORATION		
TITLE	DYNAMIC RAM CONTROLLER		
DESIGNER	KERRY VEENSTRA	ALTERA PART	EP300
STYLE CODE NUMBER	1.00	REV	B
DATE	07/16/1997	THRESH	1 OF 1



**Figure 7.**

Typical use of EP310 DRAM controller. Since the address multiplexer is separate from the controller, 256K DRAM's may be used with the same controller by expanding the multiplexer to 9 bits.



To allow byte-wide transfers over a 16-bit bus, Figure 6 asserts two RAS signals for each bank of RAM's: one for the high byte, and the other for the low byte. Which RAS is asserted in step 2 above depends on whether UDS or LDS or both lines are asserted to start a bus cycle. Supporting two banks doubles the number of RAS signals to four.

Usually the RAS signals of only one bank are asserted, thereby allowing only one bank's DRAM's to drive the data bus when the CAS line is asserted in step 4. During RAS-only refresh, however, all four RAS lines are asserted. To avoid bus contention the DRAM controller does not continue beyond step 2 during a refresh operation.

If the CPU or a DMA is used to refresh the DRAM, the REF line can be tied to GND. The DRAM controller will still function as before, but will not be used to refresh the DRAM.

Figure 7 shows how the DRAM controller would be connected between a 68000 and a 256K byte bank of 64K DRAM. The RAMEN signal comes from an external address decoder, such as Figure 4, or can be connected to address line A18 of the 68000 if only 512K of the 68000's address space is used. If 256K DRAM's are used, the address multiplexer in Figure 7 must be expanded to 9 bits.

The actual pinout of the DRAM controller is in the A+PLUS utilization report in Figure 8.

## CONCLUSIONS

EPLD's in microprocessor applications provide functions that logic designers typically use, and also provide functions that designers would not normally include in a design if only MSI integrated circuits were available. Memory maps that vary depending on a circuit's mode of operation can simplify complex microprocessor design, and when coupled to wait state generators, can increase a circuit's speed.

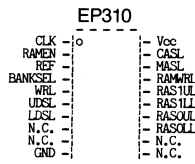
The Altera computer program A+PLUS provides a high-level environment for programmable logic design. A+PLUS converts designs in schematic form into EPLD descriptions for logic device programmers.

**Figure 8.**

*Design Processor Utilization Report for the DRAM controller in Figure 6.*

KERRY VEENSTRA  
ALTERA CORPORATION  
OCT 18, 1984  
1.00  
E  
EP300  
DYNAMIC RAM CONTROLLER

ALTERA Design Processor Utilization Report



**\*\*INPUTS\*\***

Name	Pin	Resource	MCell	PTerms	MCell	Feeds: OE	Reset	Preset
CLK	1	CKR	-	-	-	-	-	-
RAMEN	2	INP	-	-	-	-	-	-1
REF	3	INP	-	-	4	-	-	-1
BANKSEL	4	INP	-	-	4	-	-	-
WRL	5	INP	-	-	3	-	-	-
UDSL	6	INP	-	-	3	-	-	-
LDSL	7	INP	-	-	3	-	-	-

**\*\*OUTPUTS\*\***

Name	Pin	Resource	MCell	PTerms	MCell	Feeds: OE	Reset	Preset
RASOLL	13	RORF	7	2/ 8	3	-	-	-
RASOUL	14	RORF	6	2/ 8	3	-	-	-
RAS1LL	15	RORF	5	2/ 8	3	-	-	-
RAS1UL	16	RORF	4	2/ 8	3	-	-	-
RAMWRL	17	RORF	3	2/ 8	1	-	-	-
MASL	18	RONF	2	2/ 8	-	-	-	-
CASL	19	RONF	1	1/ 8	-	-	-	-

**\*\*BURIED REGISTERS\*\***

Name	Pin	Resource	MCell	PTerms	MCell	Feeds: OE	Reset	Preset

**\*\*UNUSED RESOURCES\*\***

Name	Pin	Resource	MCell	PTerms
-	8	-	-	-
-	9	-	-	-
-	11	-	-	-
-	12	-	8	8

**\*\*PART UTILIZATION\*\***

77% Pins  
57% MacroCells  
20% PTerms

Figure 9.

Listing of BASIC program that calculates Boolean equations from memory maps. The program accepts input for the width of the address bus (in bits) and for the upper and lower ranges of decoded addresses (in hexadecimal.)

```

5 ' COPYRIGHT (C) 1984 ALTERA CORPORATION
10 DEFDBL A-Z
20 FALSE=0 : TRUE=NOT FALSE
30 PRENOT$="/" : POSTNOT$="" : PROD$="*" : SUM$="+"
40 PRENOT=TRUE : POSTNOT=FALSE ' SELECT LEADING OR TRAILING INVERSION
50 ' GET SIZE OF ADDRESS SPACE
60 INPUT "NUMBER OF ADDRESS BITS"; N%
70 IF N%<1 OR N%>32 THEN 60
80 DIM STACK(N%)
90 MIN=0
100 MAX=1 : FOR N1%=1 TO N% : MAX=MAX*2 : NEXT N1% : MAX=MAX-1
110 L=MIN : U=MAX
120 PTNUM=0 ' SET FIRST P-TERM SO "=" IS PRINTED
130 ' PROMPT FOR BEGINNING AND ENDING ADDRESSES
140 P$="FIRST ADDRESS (HEX)" : GOSUB 350 : FIRST=D
150 P$="LAST ADDRESS (HEX)" : GOSUB 350 : LAST=D
160 GOSUB 200 ' PRINT EQUATION
170 PRINT "NUMBER OF PRODUCT TERMS: ";PTNUM
180 GOTO 110
190 ' RECURSIVE ROUTINE:
200 IF NOT (L>FIRST AND U<LAST) THEN 240
210 PTNUM=PTNUM+1 : IF PTNUM=1 THEN PRINT "=" : ELSE PRINT SUM$+" " ;
220 GOSUB 400 : PRINT T$
230 GOTO 330
240 IF (L>LAST AND U>LAST) OR (L<FIRST AND U<FIRST) THEN 330
250 M1=L+INT((U-L)/2)
260 STACK[I%]=U : I%=I%+1
270 U=M1 : GOSUB 200
280 I%=I%-1 : U=STACK[I%]
290 M2=L+INT((U-L+1)/2)
300 STACK[I%]=L : I%=I%+1
310 L=M2 : GOSUB 200
320 I%=I%-1 : L=STACK[I%]
330 RETURN
340 ' GET HEX NUMBER AND CONVERT TO DECIMAL
350 PRINT P$ ; : INPUT H$
360 IF LEN(H$)=0 THEN 350
370 D=0
380 WHILE LEN(H$)<>0
390 C$=LEFT$(H$,1)
400 H$=RIGHT$(H$,LEN(H$)-1)
410 IF "0"<=C$ AND C$<="9" THEN D=D*16 : D=D+ASC(C$)-ASC("0") : GOTO 450
420 IF "A"<=C$ AND C$<="F" THEN D=D*16 : D=D+ASC(C$)-ASC("A")+10 : GOTO 450
430 IF "a"<=C$ AND C$<="f" THEN D=D*16 : D=D+ASC(C$)-ASC("a")+10 : GOTO 450
440 PRINT "INVALID HEX NUMBER--TRY AGAIN." : GOTO 350
450 WEND
460 RETURN
470 ' CALCULATE PRODUCT TERM THAT DECODES ADDRESSES L THROUGH U
480 IF L=MIN AND U=MAX THEN T$="TRUE" : GOTO 600
490 T$="" : D=U : K%=N%-1
500 J=(MAX+1)/2
510 WHILE J>=U-L+1
520 IF T$<>"" THEN T$=T$+PROD$ ' APPEND PRODUCT CHARACTER
530 IF D<J AND PRENOT THEN T$=T$+PRENOT$
540 T$=T$+"A"+RIGHT$(STR$(K%),LEN(STR$(K%))-1)
550 IF D<J AND POSTNOT THEN T$=T$+POSTNOT$
560 IF D>=J THEN D=D-J
570 J=J/2
580 K%=K%-1
590 WEND
600 RETURN

```

## INTRODUCTION

The advanced technology of Altera Erasable Programmable Logic Devices (EPLDs), combined with the proprietary A+PLUS™ development software, has brought high density CMOS logic devices and schematic capture design support to the programmable logic world. Complex logic designs can now be easily integrated onto a single EPLD. However, this sophisticated combination of silicon and software has created the need for timing simulation models.

This application note will discuss timing delays which exist in Altera EPLDs. The major focus is to present the internal delay paths inherent in every EPLD, show their relation to the data sheet AC specifications, and present simulation values for these parameters. Users will be able to model and simulate their own logic designs once they understand how logic designs are actually implemented into EPLDs via the A+PLUS software; they also will know the timing parameters that model the EPLD internal delay paths.

It is assumed the user is familiar with the EPLD internal architecture and has a device data sheet which can be used for reference.

## THE EPLD STRUCTURE

Altera EPLD's are designed with a programmable AND/fixed OR PLA structure whose output connects to a programmable I/O architecture. With these two ingredients the most complex logic functions can be easily realized within a single Altera EPLD. Logic implemented in an Altera EPLD will obey the timing restrictions associated with PLA structures as well as the delays characteristic of the proprietary I/O architecture.

Delay effects for EPLD structures are generally classified with two specifications: maximum propagation delay,  $t_{PD}$ , and maximum operating frequency,  $f_{max}$ . Propagation delay ( $t_{PD}$ ) is the maximum time allowed for any external input to propagate through the EPLD and appear at any output. This parameter is defined for combinatorial logic only. Maximum operating frequency ( $f_{max}$ ), is associated with registered functions. It is the maximum frequency at which a circuit can operate. The remaining AC characteristics can be defined from these two parameters.

When designing with EPLDs, the term "gate delay" is not a useful measure. Within the EPLD AND array exist product terms. A product term is simply an n-input AND gate, where n is the number of input

connections. Depending on the logic implemented, a single product term may represent one to several gate equivalents.

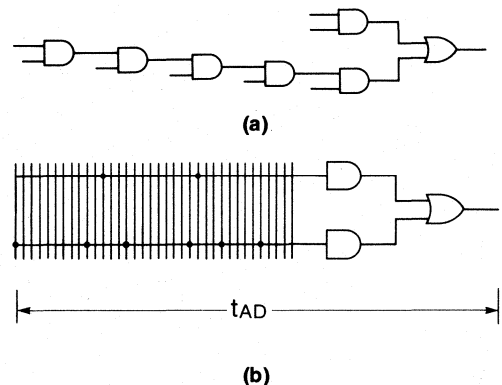
EPLD structures use two-stage logic functions, AND/OR, which have a two-stage logic delay. This delay is known as the Array Delay. In the model we will use, the number of connections made to each product term does not significantly change the delay associated with the AND/OR array.

Altera EPLD's are characterized with worst-case delay patterns. Therefore, the number of connections made in the AND array will not degrade the timing specifications presented in the EPLD data sheets nor the values presented in this application note.

Consider the circuit shown in Figure 1. The OR gate is connected to two signal paths. The upper connection represents one gate equivalent and the lower circuit path represents five gate equivalents. The same circuit transposed into an EPLD internal structure is also shown. The upper product term has two connections (one gate) and the lower term has six connections (five gates). Since the array is a fixed delay, it takes the same amount of time for each signal to propagate through the AND/OR array. Therefore, a gate delay has no significant value in an EPLD structure. For proper timing analysis of EPLD's the array delay ( $t_{AD}$ ) must be used.

**Figure 1.**

*EPLD's use Array Delays rather than Gate Delays. The inputs to the OR gate in Figure (a) represent a one-gate and five-gate equivalent circuit path. However, the delay associated with each path is the same when incorporated into an EPLD (Figure b). The Array Delay ( $t_{AD}$ ), is a fixed delay and is independent of the number of gates implemented in the AND array.*

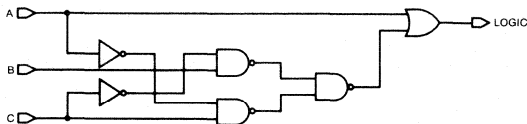


**Figure 2.**

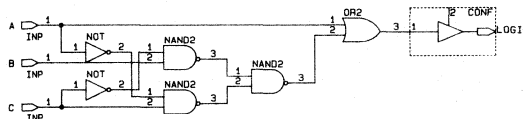
Logic implementation procedure using Altera A+PLUS software. Design to be implemented (a), Schematic using Altera library symbol primitives (b), translation to

Boolean equations (c), sum-of-products expansion (d), Final equation after Boolean reduction (e), Design "fitting" into EPLD Macrocells.

**Example 1.**



(a) Circuit Schematic



(b) Circuit Schematic Using Altera Symbols

$$\text{LOGIC} = A + \overline{(\overline{B} * \overline{C}) * (\overline{C} * \overline{A})}$$

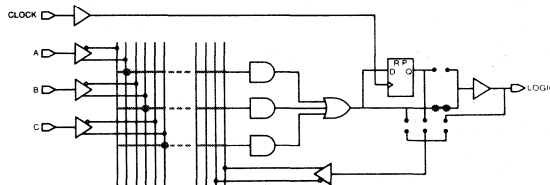
(c) Boolean Representation

$$\begin{aligned} \text{LOGIC} = & A + B * \overline{B} * \overline{C} * \overline{C} + B * \overline{B} * \overline{C} * A + B * \overline{B} * C * \overline{C} + B * \overline{B} * C * A + B * A * \overline{C} * \overline{C} + B * A * \overline{C} * A \\ & B * A * C * \overline{C} + B * A * C * A + C * \overline{B} * \overline{C} * \overline{C} + C * \overline{B} * \overline{C} * A + C * \overline{B} * C * \overline{C} + C * \overline{B} * C * A \\ & C * \overline{A} * \overline{C} * \overline{C} + C * \overline{A} * \overline{C} * A + C * \overline{A} * C * \overline{C} + C * \overline{A} * C * A \end{aligned}$$

(d) Expanded Equation

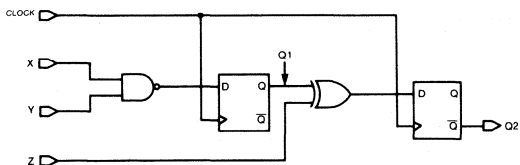
$$\text{LOGIC} = A + B + C$$

(e) Reduced Equation

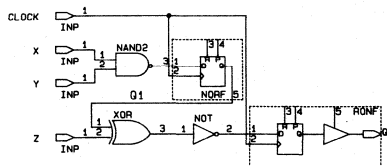


(f) Design Fit

**Example 2.**



(a) Circuit schematic



(b) Circuit Schematic Using Altera Symbols

$$\begin{aligned} Q1 &= \overline{X * Y} \\ Q2 &= \overline{Q1 \oplus Z} \end{aligned}$$

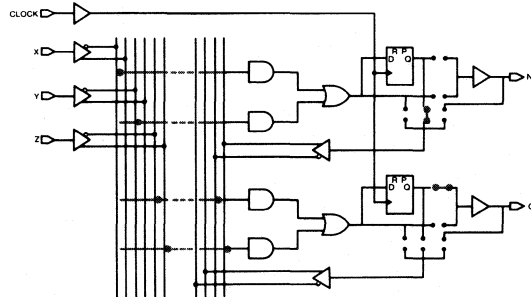
(c) Boolean Representation

$$\begin{aligned} Q1 &= \overline{X} + \overline{Y} \\ Q2 &= \overline{Z} * \overline{Q1} + Z * Q1 \end{aligned}$$

(d) Expanded Equation

$$\begin{aligned} Q1 &= \overline{X} + \overline{Y} \\ Q2 &= \overline{Z} * \overline{Q1} + Z * Q1 \end{aligned}$$

(e) Reduced Expression



(f) Design Fit

Logic designs are implemented into Altera EPLD's through the A+PLUS software package. Before presenting circuit timing simulations the reader must first understand how logic designs are translated and "fitted" into an EPLD.

A+PLUS supports four different design entry methods: Schematic Capture, Netlist Capture, Boolean Equations, and State Machine entry. Once a design has been constructed by any of the above entry methods or by a combination of them, the result is entered into the A+PLUS Design Processor (ADP). The Design Processor translates the design to appropriate Boolean equations and expands these equations to sum-of-product form (the logical structure of an Altera EPLD). Using Boolean reduction theorems the ADP then reduces the expanded equations. Once in reduced form, the ADP "fits" the logic expressions into the EPLD by selecting correct connections via EPROM bits. To illustrate this process two examples are presented in Figure 2. Example 1 shows a simple combinatorial logic design, while Example 2 represents a registered application.

The circuit schematics are shown in Figure (a) of each example. Figure (b) represents the circuit redrawn with the Altera library primitives. The design can now be translated to appropriate Boolean equations, as shown in Figure (c). Any logic function, no matter how complex, can always be represented by a Boolean equation. The expanded sum-of-products representation is given in Figure (d). Figure (e) shows the reduced logical expression after Boolean reduction theorems have been applied. Finally, the ADP fits the design by choosing proper connections via EPROM bits located both in the AND array and in the I/O architecture for each EPLD Macrocell, as shown in Figure (f).

By using Altera library symbols, functional and timing requirements can be more easily recognized. Each time an Altera I/O primitive appears, another of the EPLD internal Macrocells will be used. The method by which logic is distributed among Macrocells is a governing factor for timing simulation. For example, to feed a signal back into the AND array requires an I/O primitive; thus the function will use an EPLD Macrocell. Depending on how the Macrocell is configured (combinatorial or registered) will determine the internal delay paths. Note, the logic schematic in Example 1 used one EPLD Macrocell where the schematic in Example 2 used two Macrocells.

## EPLD DELAY ELEMENTS

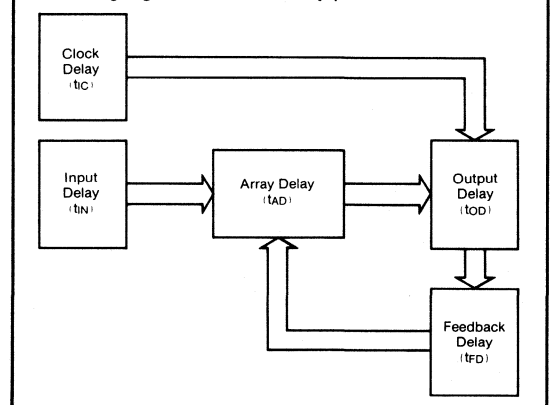
The internal architecture of an Altera EPLD is segmented into five functional blocks: dedicated data inputs, dedicated clock inputs, programmable AND-fixed OR array, output, and feedback control functions. These same blocks also classify the major internal delay

paths, see Figure 3.

The five functional delay blocks describe internal delays associated with each EPLD Macrocell. Within each Macrocell there are seven possible signal paths which may be used. Therefore, each Macrocell may be modeled with seven characteristic delays. Figure 4 shows the EPLD Macrocell and its respective internal delay paths. Depending on the logic implemented, not all delay paths will be used at any given time. For example, if a Macrocell has been configured with combinatorial output and no feedback, then the only parameters required to simulate the internal timing would be the input delay, array delay, and output delay. All input signals would have the same delay, independent of which Macrocell they are connected to

**Figure 3.**

*An EPLD can be modeled by five functional blocks which highlight the internal delay parameters.*

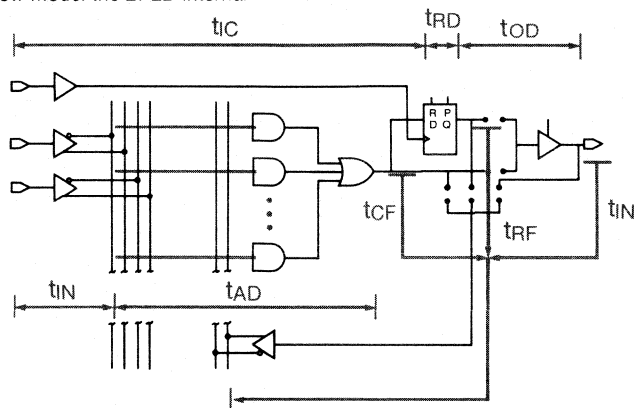


and where in the AND array they make their connection. All gating logic required to implement the function would be incorporated into the AND array and modeled with the array delay. Finally, the output delay would be added to the sum of the input and array delay to model the entire timing path. The following lists give a detailed description of each internal delay path which will be used for circuit timing simulation.

- Input Delay — The delay associated with the input pad and input buffers which direct the true and complement data input signals into the AND array.  
( $t_{in}$ )
- Array Delay — The delay for a signal to propagate through the AND array (along a product term) and appear at the output of the OR gate. This delay assumes worst case loading on any given product terms.  
( $t_{ad}$ )
- Clock Delay — The time required for the external clock signal to reach and trigger an internal register.  
( $t_c$ )

**Figure 4.**

Internal delay paths for an EPLD Macrocell. The seven delay elements shown below model the EPLD internal architecture.



**Output Delay** — The delay associated with the output buffers and output pad. For registered applications  $t_{OD}$  is measured from the output of the flipflop to the external pin. For combinatorial outputs,  $t_{OD}$  is measured from the output of the OR gate to the external pin. Due to the internal layout of the EPLD,  $t_{OD}$  is the same for both output configurations.

( $t_{OD}$ )

**Register Delay** — The delay of any flipflop. The flipflop delay is measured from the appropriate clock trigger, rising or falling edge, to the time at which valid data is present at the flipflop output.

( $t_{RD}$ )

**Register Feedback Delay** — This is the delay from the time a valid signal appears on the output of the flipflop to the time the signal both true and complement is fed back into the AND array.

( $t_{RF}$ )

**Combinatorial Feedback Delay** — The internal delay of a signal measured from the output of an OR gate to the time when both true and complement signals appear back in the AND array.

( $t_{CF}$ )

## DATA SHEET SPECIFICATIONS

The data sheet for each Altera EPLD references the timing parameters which characterize the AC operating specifications. These parameters are measured values, derived from extensive device characterization and guaranteed by 100 percent testing. The data sheet shows both typical and worst-case values. Five parameters ( $t_{PD}$ ,  $t_{SU}$ ,  $t_{CO1}$ ,  $t_{CO2}$ , and  $t_{P1}$ ) list the AC characteristics which reference and describe the EPLD internal delays. These parameters, described below in

detail, may be represented by the EPLD internal delay elements as shown in Figure 5.

1. **Propagation Delay ( $t_{PD}$ )** is defined as non-registered input or I/O input to non-registered output. The propagation delay is the time required for any external input to propagate through any combinatorial logic and appear at the EPLD external output pin. This delay is the sum of input delay ( $t_{IN}$ ), array delay ( $t_{AD}$ ), and output delay ( $t_{OD}$ ).

2. **Set-up time ( $t_{SU}$ )** is defined as non-registered input or I/O input to internal register set-up. The set-up time is the time required for the input data to settle before the triggering edge of the clock. For an Altera EPLD, the set-up time is measured relative to the external clock signal. This accounts for any combinatorial logic through which the signal must pass before reaching the internal register. The set-up time is the sum of the input delay ( $t_{IN}$ ) and array delay ( $t_{AD}$ ), minus the internal delay of the clock ( $t_{IC}$ ).

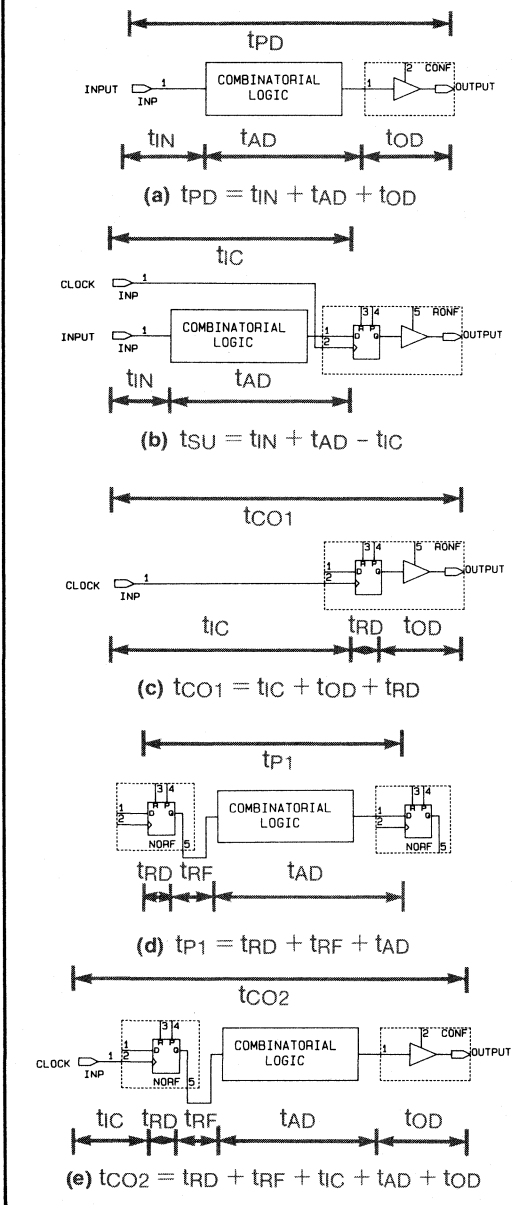
3. **Clock to Output Delay ( $t_{CO1}$ )** is defined as the time required for a signal to pass through the internal flipflop and appear at the EPLD external pin, relative to the triggering edge of the clock. Clock to output delay is the sum of the register delay ( $t_{RD}$ ), clock delay ( $t_{IC}$ ), and output delay ( $t_{OD}$ ).

4. **Minimum internal clock period ( $t_{P1}$ )** is defined as registered feedback to registered input via an internal path. This is the maximum clock frequency at which an EPLD can operate when registered inputs are dependent only on an internal logic, not on any external input. The minimum clock period accounts for any combinatorial logic through which the feedback signal must pass before reaching the input of the flipflop. Minimum clock period is the sum of the registered feedback delay ( $t_{RF}$ ), array delay ( $t_{AD}$ ), and the register delay ( $t_{RD}$ ).

5. **Registered Feedback to Combinatorial Output ( $t_{CO2}$ )** is defined as the delay for a signal to propagate from an input of a flipflop through the AND/OR array and appear at an external pin, relative to the external clock.

**Figure 5.**

The EPLD data sheet AC specifications are modeled by the internal delay paths.



This delay accounts for any combinatorial logic through which the registered feedback signal propagates before being output to the external pin.  $t_{CO2}$  is the sum of the register delay, ( $t_{RD}$ ), register feedback delay ( $t_{RF}$ ), clock delay ( $t_{IC}$ ), array delay ( $t_{AD}$ ), and output delay ( $t_{OD}$ ).

## EPLD SIMULATION

When analyzing a logic design, the user must first express the circuit with the Altera primitive symbols that are provided with the A+PLUS development software. The I/O primitive symbols are shown in Figure 13. Altera EPLD's will only accept circuits designed with these symbol primitives. Altera primitives clearly represent all delay paths which exist in the EPLD's.

Next, the user must partition the circuit into the five functional blocks which represent the EPLD internal architecture: clock inputs, data inputs, gates, feedback, and output configurations. After partitioning, the user highlights the appropriate delay paths and begins tracing the desired signal from input to output accounting for each delay parameter and its effect on the overall timing characteristics. **Note that individual gates do not add any additional delay to the path.**

Table 1, the simulation data table, shows each of the internal delay parameters for EP300, EP310 and EP1200 logic devices. It also shows the relation of the modeled parameters to the data sheet AC specifications. Both the model results and the actual data sheet values are given. Note that in some cases the modeled result differs slightly from the data sheet specification. This discrepancy is the result of the different guardbands used to ensure compliance with the specification when tested during the manufacture of the product. The model does not account for the extended guardbanding of some data sheet specifications. Even though the data sheet shows max/min numbers, the model results should be considered "typical worst-case" numbers. Also note, that in the actual device, rise time and fall time delays are different, as are the delays due to different programming patterns. No differentiation between rise and fall times are made in this analysis.

## EXAMPLES

Consider the circuit presented in Figure 6. This combinatorial design uses no clock or feedback. The timing restrictions are dependent only on the input, array, and output delay elements. This simple example will help illustrate circuit partitioning into the EPLD internal delay paths.

As Figure 6 shows, the circuit consists of three inputs, gating logic and one output. The circuit is therefore partitioned into three delay paths: input delay, array delay, and output delay. Notice that all the gating logic is incorporated into the AND/OR array. The maximum time allowed for any input to propagate through the entire circuit (from input pin to output pin) is also known as the propagation delay,  $t_{PD}$ . The worst case  $t_{PD}$  is given in Tables 1 and in the respective EPLD data sheet. Propagation delay always applies when combinatorial output logic is used. Thus:

$$t_{LOGIC} = t_{IN} + t_{AD} + t_{OD} = t_{PD}$$

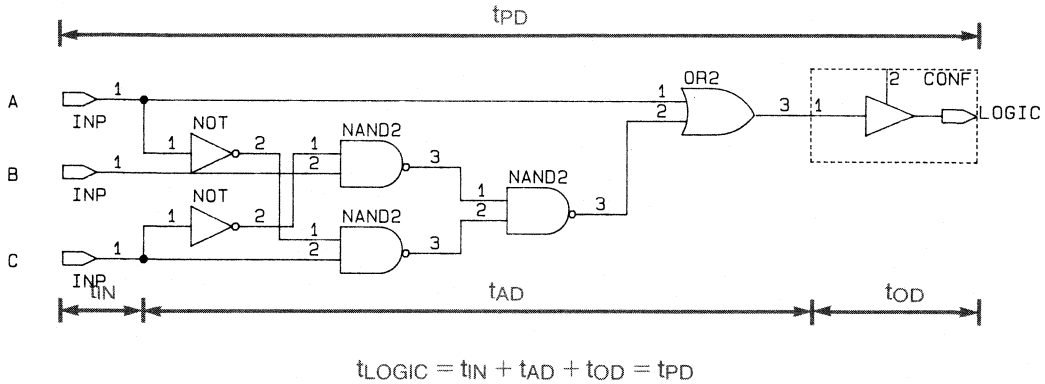


To illustrate the delay effects in an asynchronous design using combinatorial feedback, an R-S latch will be analyzed. Figure 7 presents such a latch designed with NOR gates. The circuit is also represented with the

Altera symbol primitives and partitioned into its appropriate delay paths, i.e., input delay, array delay, output delay and feedback delay.

**Figure 6.**

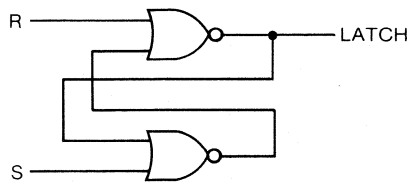
The maximum delay for any input to propagate through the circuit shown below is the sum of the internal input delay ( $t_{IN}$ ), array delay ( $t_{AD}$ ), and output delay ( $t_{OD}$ ). The total delay is also known as the propagation delay ( $t_{PD}$ ).



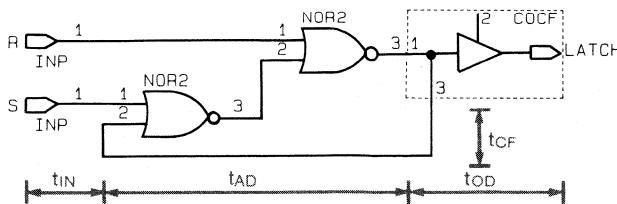
**Figure 7.**

An asynchronous R-S latch is shown in Figure (a). Representing the circuit with the Altera symbol primitives, Figure b, shows the internal delay paths. To avoid

unwanted output glitches, the input pulse width must be greater than  $t_{CF} + t_{AD}$



(a)



(b)

As in any combinatorial design, the maximum delay from input to output is the propagation delay,  $t_{PD}$ . If the output of the latch is a logical zero ( $Q=L$ ), and a logical one is applied to the Set input ( $S=H$ ), the output of the latch will go to a logical one ( $Q=H$ ) within one propagation delay, or  $t_{Set} \leq t_{PD}$ . The same principle is true if the latch is reset ( $R=H, S=L$ ).

To avoid any unwanted output glitches, the asynchronous set-up time of the latch must be observed. The pulse width of either Reset or Set inputs must be long enough to ensure that the feedback signal can return from the output primitive and reensure the latched condition. The input pulse width must be greater than the time required for the feedback signal to return to the AND array and propagate through the gating logic. Otherwise, an H-L or L-H glitch may occur at the output. Thus, Input Pulse Width  $> t_{AD} + t_{CF}$ .

The next example evaluates a synchronous design presented in Figure 8. This circuit uses three inputs, one clock (common to both D-flipflops), gating logic, and one output. Note that the output is active low. The design is first represented with the Altera primitive symbols and partitioned to highlight the delay paths. There are six inherent delays with this design: clock delay, input delay, array delay, register delay, register feedback delay, and output delay. The active low output is accomplished by driving the output primitive with an inverter. All gating logic, including the inverter, is incorporated into the AND/OR array.

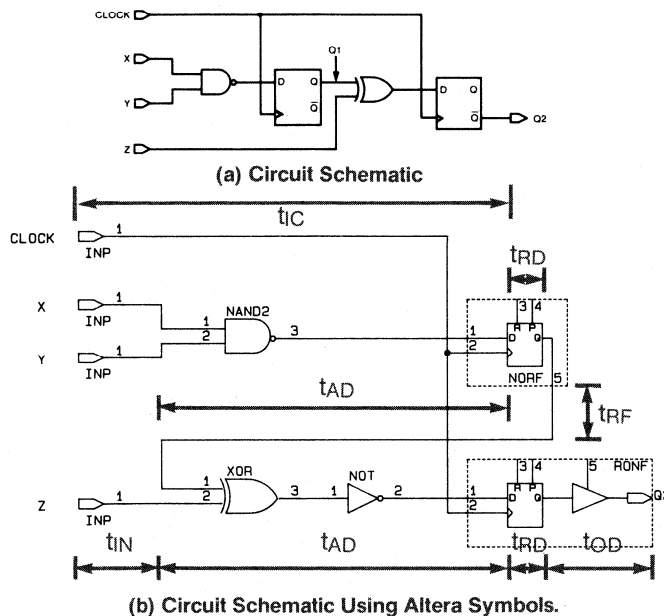
First, consider the clock to output delay, characterized by the  $t_{CO1}$  specification described in the data sheet. Refer to the output register of Figure 8 (lower D-flipflop). Once the triggering edge activates the flip-flop, the data residing on the D input will appear at the output after passing through the register and three-state output buffer. The triggering edge of the clock has been delayed. Therefore, with respect to the clock seen at the external pin, an additional delay must be added. Thus, the maximum amount of delay before data reaches the output is the sum of the register delay ( $t_{RD}$ ), and output delay ( $t_{OD}$ ), and clock delay ( $t_{C}$ ). The timing diagram in Figure 9 illustrates these delays.

The set-up time is the time required for the input data to be stable before the triggering edge of the clock. The set-up time for the EPLD internal flipflops has been modeled from the input, array, and clock delay paths. To illustrate the model, consider the timing diagram shown in Figure 10. The upper waveform represents the clock signal seen at the pin. Due to the internal clock delay,  $t_{C}$ , this signal has been shifted when examined at the internal D-flipflop. The external data signal is also delayed by an input and array delay before it reaches the flipflop. The set-up time is the sum of the input delay and array delay minus the internal delay of the clock. In the actual device the set-up time for the internal registers is extremely small. The data sheet  $t_{SU}$  specification accounts for the EPLD internal delay. As long as the set-up time is obeyed at the external inputs, the circuit will function properly.

**Figure 8.**

When the circuit schematic in Figure (a) is represented with Altera symbol primitives, Figure (b), six internal delay paths are shown: clock delay ( $t_{C}$ ), register delay

( $t_{RD}$ ), feedback delay ( $t_{RF}$ ), input delay ( $t_{IN}$ ), array delay ( $t_{AD}$ ), and output delay ( $t_{OD}$ ). These internal delay elements are used to simulate the circuit.



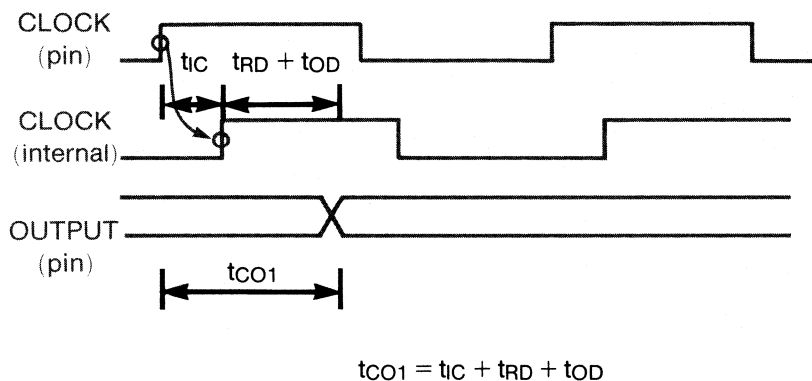
The maximum internal clock frequency, data sheet specification  $tp_1$ , is governed by the internal feedback path. Again, referring to the circuit in Figure 8, the time required for a signal to pass from the upper register to the lower register determines the minimum internal clock period at which the circuit could operate. Once data is triggered in the top register, the signal passes through a register delay ( $t_{RD}$ ), feedback delay ( $t_{RF}$ ) and array delay ( $t_{AD}$ ), before reaching the bottom register. The sum of these delays specifies the maximum internal clock frequency. It was previously shown that the register set-up time accounts for the EPLD internal delays. This is also true for signals which are

generated internally; the register set-up time accounts for the internal delays. Therefore, circuits which are dependent only on internal signals can operate at the minimum clock period specified by the data sheet parameter  $tp_1$ .

Most circuit functions are dependent on both external and internal signals. When this is the case, the maximum clock frequency is dependent on the input delay, array delay, register delay, output delay, feedback delay, and clock delay. The data sheet specification  $tp_2$  specifies the minimum clock period when the next state of any internal register is dependent on both external and internal data signals.

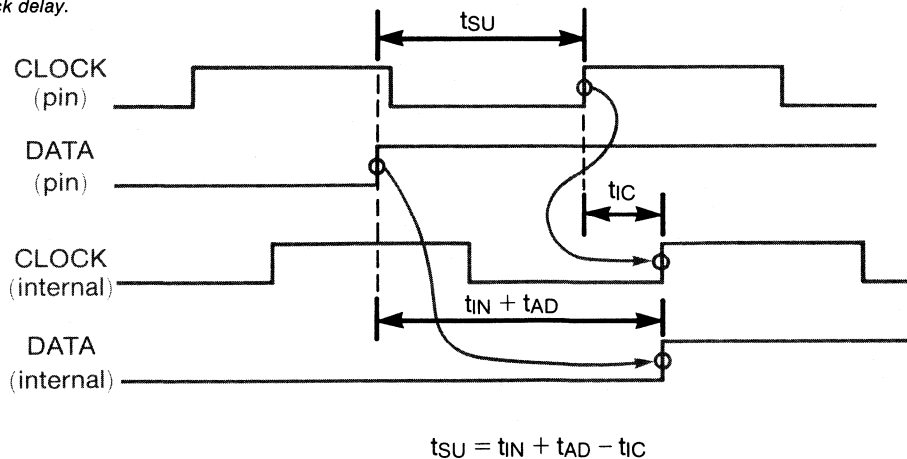
**Figure 9.**

*Clock to output delay,  $t_{CO1}$ , is the sum of the clock delay, register delay, and output delay.*



**Figure 10.**

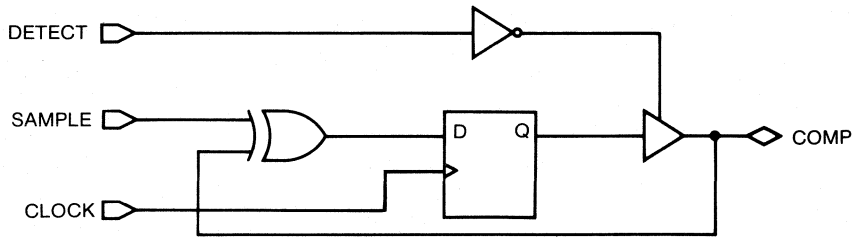
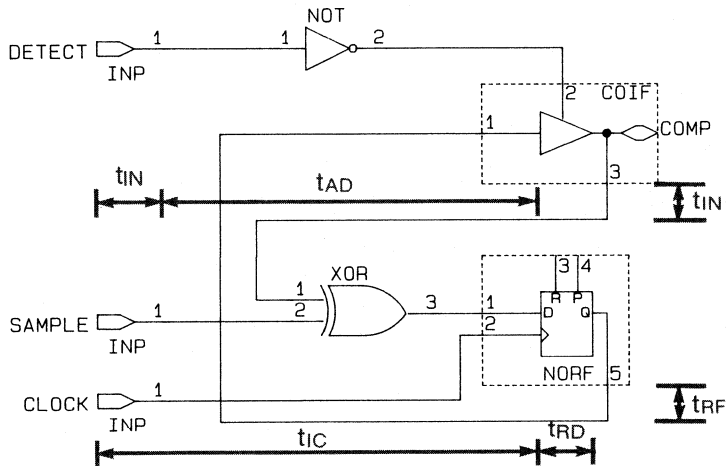
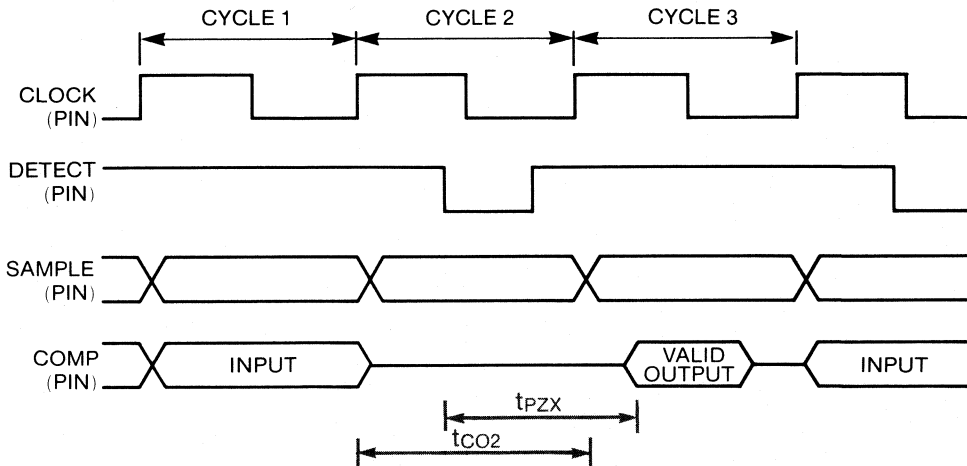
*The set-up time for an EPLD internal register is the sum of the input delay and array delay minus the internal clock delay.*



**Figure 11.**

The circuit shown below compares two input signals (*SAMPLE* and *COMP*) and stores the result in the internal register. The delay associated to output the data is characterized by the data sheet  $t_{CO2}$  specification.  $t_{CO2}$  is the sum of the clock delay ( $t_{IC}$ ),

registered delay ( $t_{RD}$ ), feedback delay ( $t_{RF}$ ), array delay ( $t_{AD}$ ), and output delay ( $t_{OD}$ ). Note, output enable is controlled by the input *DETECT*. The delay associated to enable or disable the output,  $t_{PZX}$ , is given in the EPLD data sheet.

**(a) Circuit Schematic****(b) Circuit schematic represented with Altera primitives****(c) Timing diagram**

The last example analyzes the circuit in Figure 11. This circuit uses two dedicated inputs, (DETECT and SAMPLE) and one bidirectional pin (COMP). The I/O direction is controlled by the input DETECT. If DETECT is a logical one, DETECT=H, then the output is seen as a high impedance node, and the signal COMP may be used as a dedicated input signal into the EPLD AND/OR array. If DETECT is a logical zero, DETECT=L, then the output is driven by the internal EPLD register. During the first clock cycle, as shown in the timing diagram provided in Figure 11, the inputs are compared by means of the exclusive-OR gate. The result is stored in the register. In the next clock cycle, the stored result is output from the register, and feedback through the AND/OR array to the three-state output buffer to which COMP is connected. On the third clock cycle, the output is enabled by the input signal DETECT. Once the signal has been successfully transmitted, the output buffer is again disabled thus allowing another comparison to begin.

During the first clock cycle, the data set-up time for the internal register has been satisfied. The second clock cycle characterizes the data sheet AC specification  $t_{CO2}$ . Figure 11 also shows the circuit represented with the Altera primitive symbols highlighting the internal delay paths.  $t_{CO2}$  is the delay from an internal register to a combinatorial output, referenced from the triggering edge of the external clock signal. Therefore it is dependent on the clock delay (the triggering edge as seen at the register is delayed by  $t_{IC}$ ), register and feedback delay, array delay (the signal must pass through the AND/OR array to reach another I/O primitive, and the output delay.

In this example, the data reaches the output well before the output is enabled. Also notice that the third clock cycle will stimulate another output transition from invalid data. The output buffer is disabled well before this erroneous data can reach the output pin.

The I/O primitive COIF (combinatorial output, input feedback) is used for bidirectional operation in this example. The input signal is characterized by a normal input delay,  $t_{IN}$ . This primitive can also be used as a COCF (combinatorial output, combinatorial feedback) structure. Figure 12 shows both the COIF and COCF I/O primitives. The output delay is the same for each. The feedback delay is longer for the COIF primitive because it is connected after the three-state buffer. The difference in feedback delay is simply  $t_{IN} - t_{CF}$ . The EP1200 I/O pins do not allow COCF architectures but do allow the COIF architecture. Therefore, by sacrificing a few nanoseconds, combinatorial feedback can be easily achieved in the EP1200 I/O architecture.

## CONCLUSION

To implement logic designs into Altera EPLDs users must use the Altera symbol primitives. These primitives not only allow design entry but also highlight the internal delay paths associated with EPLDs. By knowing these internal delay parameters and their relation to the data sheet specifications users can now simulate their own designs ensuring that both functional and timing requirements are satisfied. The simulation data table, provided in this application note, is used together with the EPLD data sheet to reference all AC specifications.

**Figure 12.**

The symbol primitives COIF and COCF both provide combinatorial feedback. Due to the positioning of the feedback path the COIF feedback delay will be longer. The difference is  $t_{IN} - t_{CF}$ .

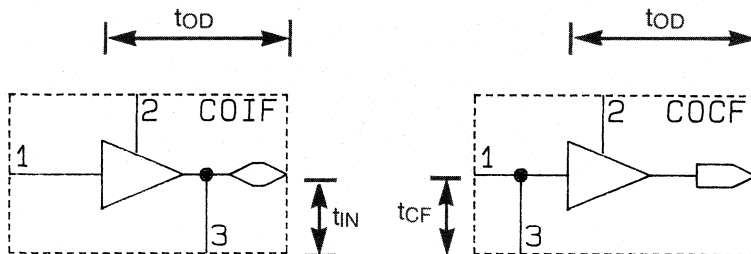


Table 1. EPLD SIMULATION DATA

PART NUMBER	MODEL PARAMETER	DELAY (ns)
EP300	tAD	61
	tRD	9
	tOD	13
	tIN	16
	tIC	15
	tRF	4
	tCF	4
EP300-2	tAD	43
	tRD	8
	tOD	12
	tIN	12
	tIC	11
	tRF	4
	tCF	4
EP310	tAD	31
	tRD	8
	tOD	10
	tIN	10
	tIC	8
	tRF	4
	tCF	4
EP310-2	tAD	21
	tRD	7
	tOD	8
	tIN	8
	tIC	5
	tRF	3
	tCF	3
EP1210-1	tAD	38
	tRD	7
	tOD	8
	tIN	10
	tIC	8
	tRF	5
	tCF	5
EP1210-2	tAD	48
	tRD	7
	tOD	11
	tIN	12
	tIC	11
	tRF	5
	tCF	5
EP1210	tAD	67
	tRD	7
	tOD	11
	tIN	16
	tIC	16
	tRF	5
	tCF	5

PART NUMBER	DATA SHEET PARAMETER	MODEL RESULTS (ns)	DATA SHEET (ns)
EP300	tPD	90	90
	tSU	62	62
	tCO1	37	38
	tCO2	102	100
	tP1	74	75
EP300-2	tPD	67	65
	tSU	44	47
	tCO1	31	33
	tCO2	78	75
	tP1	55	55
EP310	tPD	51	50
	tSU	33	32
	tCO1	26	28
	tCO2	61	70
	tP1	43	42
EP310-2	tPD	37	35
	tSU	24	28
	tCO1	20	22
	tCO2	44	55
	tP1	31	30
EP1210-1	tPD	56	50
	tSU	40	37
	tCO1	23	30
	tCO2	66	65
	tP1	50	50
EP1210-2	tPD	71	65
	tSU	49	47
	tCO1	29	36
	tCO2	82	75
	tP1	60	58
EP1210	tPD	94	90
	tSU	67	65
	tCO1	34	45
	tCO2	106	100
	tP1	79	75

**Parameter Relationships**

$$t_{PD} = t_{IN} + t_{AD} + t_{OD}$$

$$t_{SU} = t_{IN} + t_{AD} - t_{IC}$$

$$t_{CO1} = t_{IC} + t_{RD} + t_{OD}$$

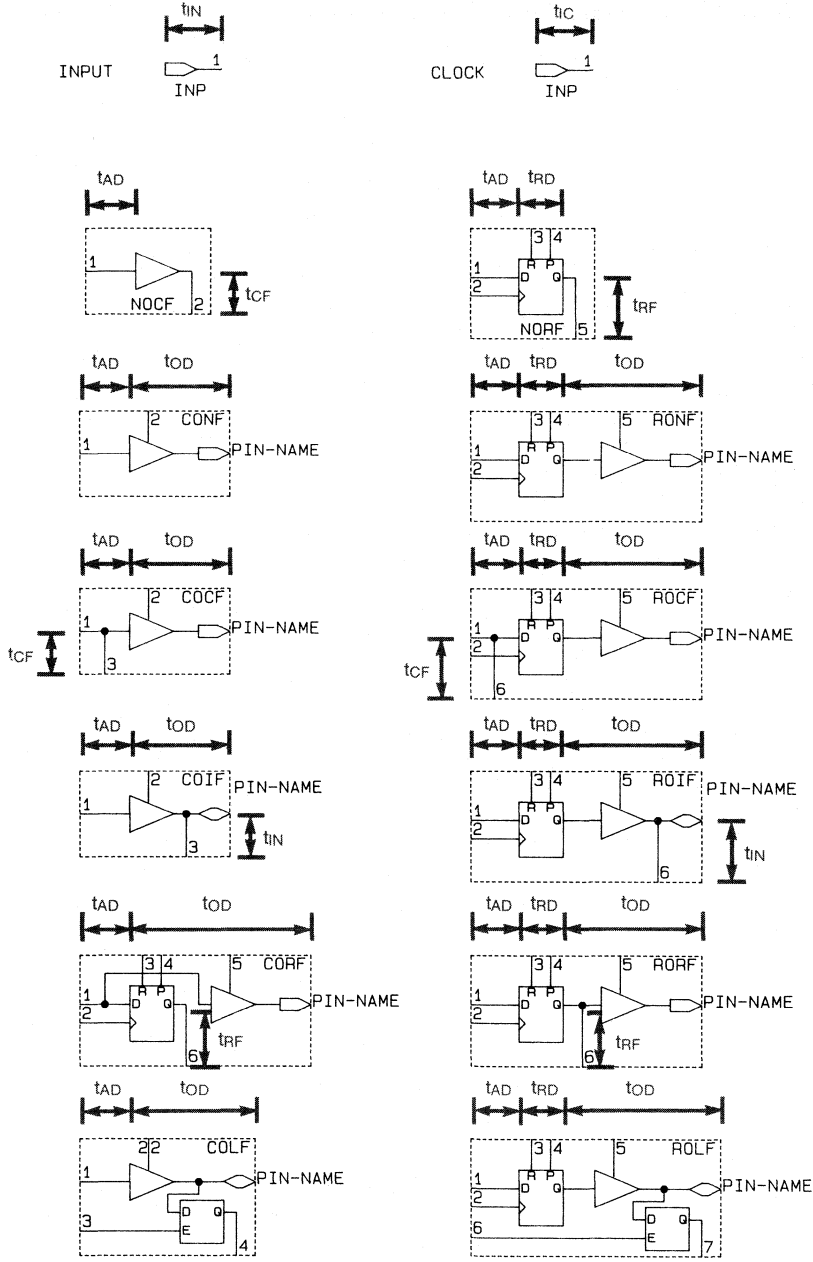
$$t_{CO2} = t_{IC} + t_{RD} + t_{RF} + t_{AD} + t_{OD}$$

$$t_{P1} = t_{RD} + t_{RF} + t_{AD}$$

Note: Typical delays only.

**Figure 13. A+PLUS DESIGN LIBRARY PRIMITIVES**

Below are the input and I/O symbol primitives supplied with the A+PLUS software. These library elements must be used for design entry into Altera EPLD's. The internal delay paths associated with each input and I/O architecture is shown.



### INTRODUCTION

As the demand for application specific integrated circuits continues to grow, the need for programmable logic devices becomes increasingly evident. However, the devices alone do not enable the logic designer to take advantage of user programmable logic. Of equal importance is the need for development tools which can be used in conjunction with programmable logic devices.

Altera has achieved both of these goals by introducing its family of CMOS-based EPLD's (Erasable Programmable Logic Devices) as well as the Altera PLDS2 (Programmable Logic Development System). This application note will outline how to enter a design, ultimately producing "working silicon" within minutes at your own desktop.

### GETTING STARTED

#### \*\*READER'S NOTE

The information within this application note is based on the following versions of ALTERA software:

NetMap ..... Version 3.0  
A+PLUS..... Version 3.0  
LogicMap II ..... Version 3.0

The first and most important step (**this is an absolute MUST!**) in using your PLDS2 is the creation of a hand-drawn logic schematic for the design you wish to implement. In doing so, the user should follow the set of guidelines in figure 1.

(For additional help, see the list of **Design Recommendations** on page 133 in this Application Note.)

For a complete description of each primitive see the Altera User Manual (Reference Manual, Section 4).

Figure 2 shows the list of Altera Design Primitives available to the NetMap user. Your hand-drawn schematic should use elements exclusively from this list.

Figure 1 - Guidelines for Hand-drawn schematic

1. Use elements exclusively from the Altera Design Primitives Library shown in figure 2.
2. All inputs to the chip must be inputs to an "INPUT PRIMITIVE."
3. All outputs from the chip are outputs from an "I/O PRIMITIVE."
4. No "VCC" or "GND" connections to any "LOGIC PRIMITIVES."
5. Asynchronous clock signals must pass through an "Asynchronous Clock Primitive" (CLKB) before connection to the clock input of a register.
6. Assign exactly one name to each unique signal in the circuit.
7. Limit pin-names to eight characters or less.
8. Observe the clock configurations available in the target EPLD.

Figure 2 - Altera Design Primitives

TYPE OF PRIMITIVE	DESCRIPTION	APPLICABLE PARTS
<b>INPUT PRIMITIVES</b>		
IMP	Normal pin Input	All
LIMP	Latched pin Input	EP1200, EP1210
<b>LOGIC PRIMITIVES</b>		
AND	Logical AND function (up to 12 inputs)	All
NAND	Logical NAND function (up to 12 inputs)	All
OR	Logical OR function (up to 12 inputs)	All
NOR	Logical NOR function (up to 12 inputs)	All
NOT	Inversion function	All
XOR	Exclusive OR function	All
<b>I/O PRIMITIVES</b>		
COCF	Combinatorial Output, Combinatorial Feedback	EP300, EP310
COBF	Combinatorial Output, D-Registered Feedback	EP300, EP310, EP1200, EP1210
COF	Combinatorial Output, No Feedback	All
COIF	Combinatorial Output, I/O Feedback	All
COLF	Combinatorial Output, Latched Feedback	EP1200, EP1210
ROCF	D-Registered Output, Combinatorial Feedback	EP300, EP310
ROBF	D-Registered Output, D-Registered Feedback	All
ROF	D-Registered Output, No Feedback	All
ROIF	D-Registered Output, I/O Feedback	All
ROLF	D-Registered Output, Latched Feedback	EP1200, EP1210
TOTF	T-Registered Output, T-Registered Feedback	EP600, EP900
TONF	T-Registered Output, No Feedback	EP600, EP900
TOIF	T-Registered Output, I/O Feedback	EP600, EP900
JOIF	JK-Registered Output, JK-Registered Feedback	EP600, EP900
JONF	JK-Registered Output, No Feedback	EP600, EP900
SOSF	SR-Registered Output, SR-Registered Feedback	EP600, EP900
SONF	SR-Registered Output, No Feedback	EP600, EP900
NOCF	No Output, Combinatorial Feedback	All
NOBF	No Output, D-Registered Feedback	All
NOIF	No Output, T-Registered Feedback	EP600, EP900
NOJF	No Output, JK-Registered Feedback	EP600, EP900
NOSF	No Output, SR-Registered Feedback	EP600, EP900
<b>EQUATION PRIMITIVES</b>		
EQN	Arbitrary Boolean Expression	All
<b>ASYNCHRONOUS CLOCK PRIMITIVES</b>		
CLKB	Asynchronous Clock Buffer	EP600, EP900

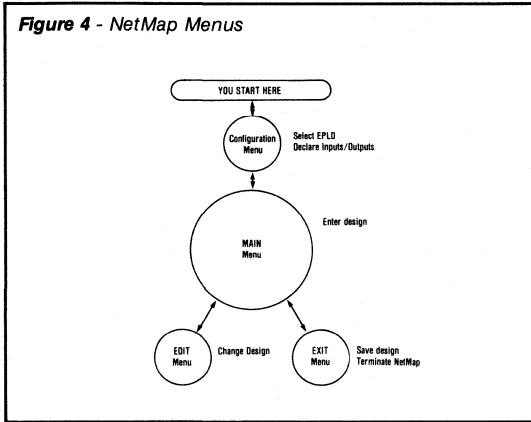
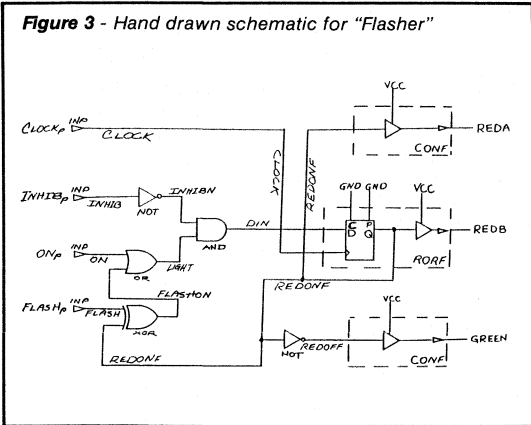
### DESIGN EXAMPLE

Figure 3 shows the hand-drawn schematic for the design example called "FLASHER." Notice that it adheres to all of the guidelines set forth in figure 1.

If your hand-drawn schematic does not follow



these guidelines, make the necessary adjustments before going on with this design example.



### ENTERING THE DESIGN

NetMap, the Netlist Design Entry Method, is the program in which the user may enter the hand-drawn schematic design.

#### STOP!

Before you can make use of the NetMap program or any of the Altera software, it is NECESSARY to install the software into your computer system. To do this, consult the User Guide within the Altera User Manual concerning "Software Installation."

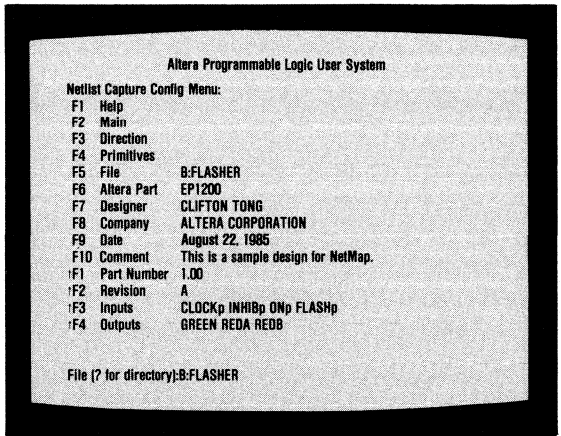
**IF INSTALLATION IS COMPLETE, PROCEED.**

The NetMap program contains various menus, each with specific functions. Figure 4 shows these menus and what you should use them for.

For detailed descriptions concerning various menu options in NetMap as well as all Altera software, see the appendix at the back of this Application Note.

For a description of any error message that you may come across while in NetMap, consult Appendix A in the Altera User Manual.

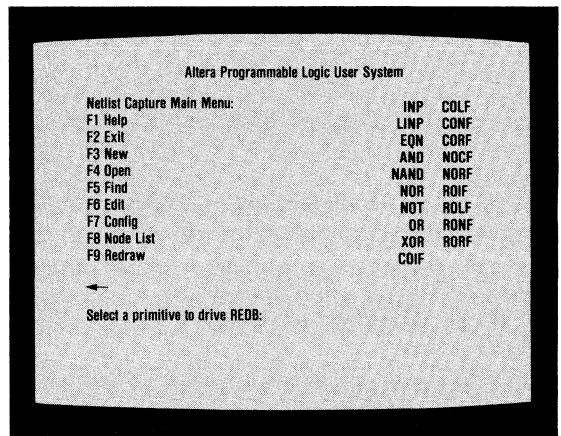
**Figure 5 - NetMap Configuration Menu**



Once invoked, NetMap prompts the user for a filename for the design followed by various information concerning the design. This information is entered within the NetMap Configuration Menu. Figure 5 shows the Configuration Menu entries you should now make for the "FLASHER" design example.

The Main Menu (see figure 6) is the next menu that appears. This is where the design will actually be entered.

**Figure 6 - NetMap Main Menu**



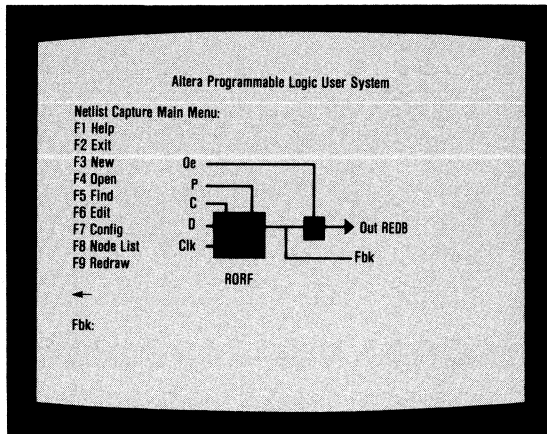
The first prompt is:  
**Press a function key:**  
You should respond by pressing F4, which locates the first "Open" node.  
The next prompt is:  
**Select a primitive to drive REDB:**  
In other words, NetMap is asking for a primitive

that has the signal "REDB" as an output. By inspection of our hand-drawn schematic we see that the appropriate response is:

**RORF (enter)**

The RORF primitive is displayed as shown in figure 7 with REDB connected to the OUT signal.

**Figure 7 - Display of a Primitive within NetMap Main Menu**



Now you are prompted with the remaining signals:

For **Fbk**: enter **REDONF (enter)**

For **OE**: enter **VCC (enter)**

For **P**: enter **GND (enter)**

For **C**: enter **GND (enter)**

For **D**: enter **DIN (enter)**

For **Clk**: enter **CLOCK (enter)**

### **\*\*WARNING\*\***

The NetMap module will distinguish between upper and lower case signal names. In other words, the same signal name typed once in upper case and once in lower case represent DIFFERENT SIGNALS. As a result, it is recommended for users to pay close attention to the signal names in this sample session design session and type them EXACTLY as shown.

As you enter the connections they are displayed in the viewing area. The next prompt is:

**Select a primitive to drive CLOCK:**

enter: **INP (enter)**

The INP primitive is now shown and you are prompted for the proper connection:

For **In**: enter **CLOCKp (enter)**

The following prompts and user entries are needed to complete the entry of this design example:

PROMPT	USER ENTRY
Select a primitive to drive DIN:	<b>AND (enter)</b>
In1:	<b>INHIBN (enter)</b>
In2:	<b>LIGHT (enter)</b>
In3:	<b>(F4)</b>

Select a primitive to drive LIGHT:**OR (enter)**

In1: **ON (enter)**

In2: **FLASHON (enter)**

In3: **(F4)**

Select a primitive to drive FLASHON:**OR (enter)**

In1: **FLASH (enter)**

In2: **REDONF (enter)**

In3: **(F4)**

Select a primitive to drive FLASH:**INP (enter)**

In: **FLASHp (enter)**

Select a primitive to drive ON: **INP (enter)**

In: **ONp (enter)**

Select a primitive to drive INHIBN:**NOT (enter)**

In: **INHIB (enter)**

Select a primitive to drive INHIB:**INP (enter)**

In: **INHIBp (enter)**

Select a primitive to drive REDA:**CONF (enter)**

OE: **VCC (enter)**

In: **REDONF (enter)**

Select a primitive to drive GREEN:**CONF (enter)**

OE: **VCC (enter)**

In: **REDOFF (enter)**

Select a primitive to drive REDOFF:**NOT (enter)**

In: **REDONF (enter)**

At this point the following message appears:

**All nodes have been connected in both directions.**

**Press a function key:**

### **STOP!**

Normally we would save our file and exit at this point. However, in order to show how to EDIT the design in NetMap, we will not exit NetMap at this time.

## **CHANGING THE DESIGN**

Figure 8 is the design (same as figure 3) we have just entered in NetMap. Figure 9 shows the changes we are going to make to our original "FLASHER" design.

**CHANGE #1:** Remove the inverter and have the signal known as INHIB feed the AND gate directly.

**CHANGE #2:** Trigger the register on the falling edge of the register clock rather than the rising edge. This is achieved by inserting an inverter along the signal known as CLOCK and having its inverted signal, known as CLOCKLO, feed the register.

**CHANGE #3:** Change the signal name of DIN to REDON. This is only a name change and does not affect the circuit.

Figure 8 - Handrawn schematic for "Flasher"

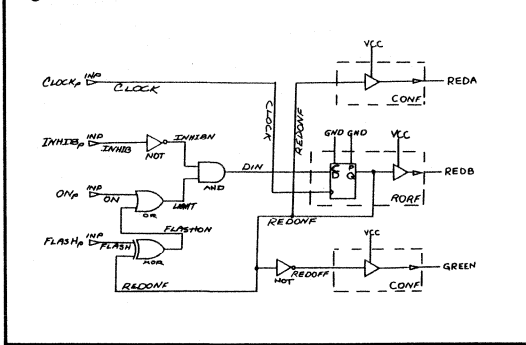
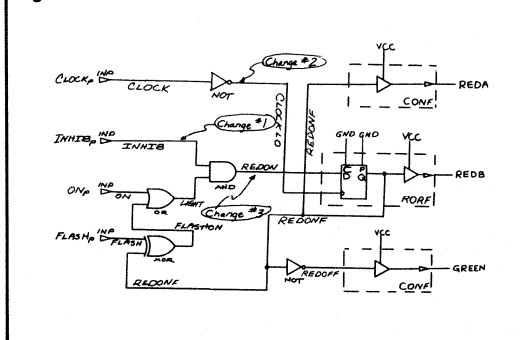


Figure 9 - Altered "Flasher"



The Edit Menu (shown in figure 10) will be used to make the desired changes to the circuit.

The following user prompts and entries will put each of the above changes into our existing circuit:

PROMPT	USER ENTRY
--------	------------

FOR CHANGE #1:

Press a function key:	<b>(F5)</b> (FIND)
Node:	<b>INHIBN</b> (enter)
Press a function key:	<b>(F6)</b> (Edit)
Press a function key:	<b>(F7)</b> (Delete Prim.)
Press a function key:	<b>(F4)</b> (Open Node)

Select a primitive to drive INHIBN:	<b>(F6)</b> (Edit)
Press a function key:	<b>(F3)</b> (Reconnect)
New Node:	<b>INHIB</b> (enter)

Press a function key:	<b>(F2)</b> (Main Menu)
-----------------------	-------------------------

Press a function key:	<b>(F4)</b> (Open Node)
-----------------------	-------------------------

FOR CHANGE #2:

Press a function key:	<b>(F5)</b> (Find)
Node:	<b>CLOCK</b> (enter)

Press a function key:	<b>(rt. arrow key)</b>
Press a function key:	<b>(rt. arrow key)</b>

**NOTE:** After using the Find function (F5) hitting an arrow key when one of the nodenames is highlighted (in this case CLOCK) will always scan the circuit in the direction of the arrow key hit.

Press a function key:	<b>(F6)</b> (Edit)
Press a function key:	<b>(F3)</b> (Reconnect)
New Node:	<b>CLOCKLO</b> (enter)
Press a function key:	<b>(F2)</b> (Main Menu)
Press a function key:	<b>(F4)</b> (Open Node)

Select a primitive to drive	<b>NOT</b> (enter)
CLOCKLO:	<b>CLOCK</b> (enter)
In:	

FOR CHANGE #3:

Press a function key:	<b>(F5)</b> (Find)
Node:	<b>DIN</b> (enter)

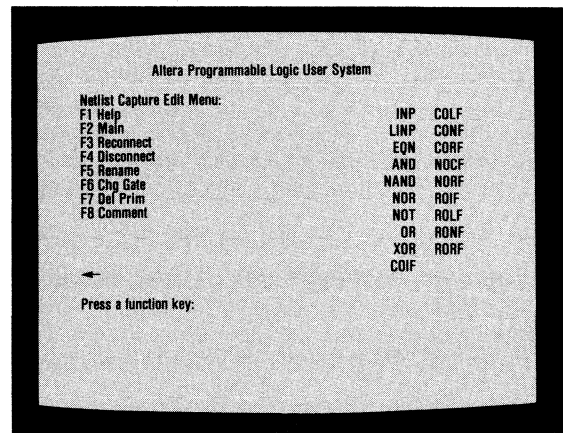
Press a function key:	<b>(F6)</b> (Edit)
Press a function key:	<b>(F5)</b> (Rename)

New Node:	<b>REDON</b> (enter)
-----------	----------------------

Press a function key:	<b>(F2)</b> (Main Menu)
-----------------------	-------------------------

Press a function key:	<b>(F4)</b> (Open Node)
-----------------------	-------------------------

Figure 10 - NetMap Edit Menu



Now that we have made the desired changes we can now save and exit from NetMap. Remember that we should interpret the message...

**All nodes have been connected in both directions.**

**Press a function key:**

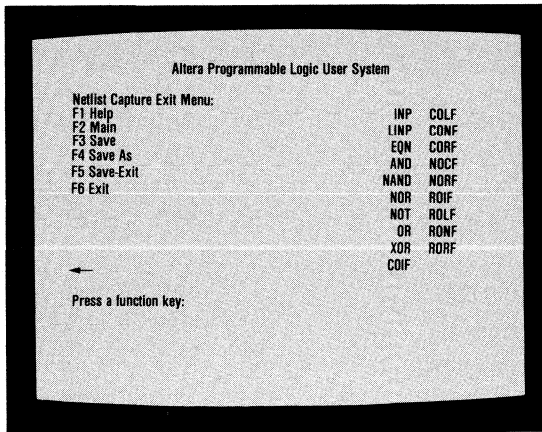
as our circuit is completed without any unconnected nodes. Respond with the following in order to save the design and exit from NetMap:  
enter: **(F2)** (Exit)

The Exit Menu (shown in figure 11) appears with the following prompt:

**Press a function key:**

enter: **(F5)** (saves file and exits NetMap)

**Figure 11 - NetMap Exit Menu**



This will cause the file FLASHER.ADF (shown in figure 12) to be written to disk as well as return the user to DOS.

**YOU HAVE NOW CAPTURED A DESIGN WITH NETMAP!!**

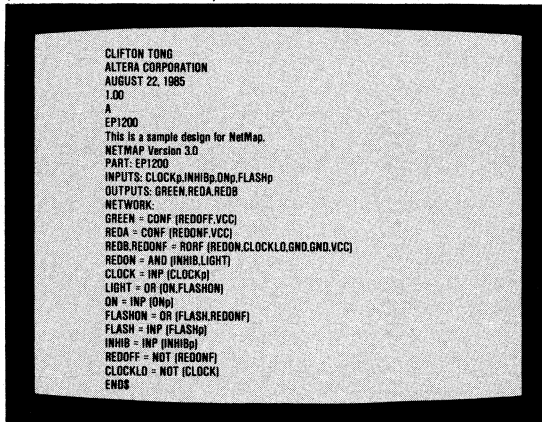
**FROM NETMAP TO WORKING SILICON**

At this point you are literally 5 minutes away from having a fully functional chip!

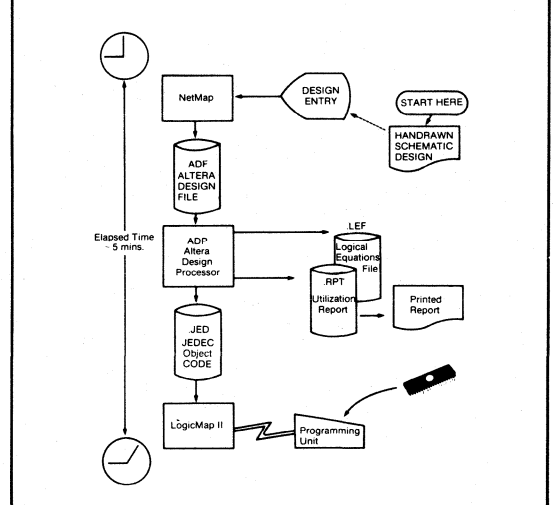
Figure 13 shows the finishing touches that need to be made in PLDS2.

Now that the design is entered, it may be used as the input to the Altera Design Processor which automatically performs Boolean minimization and design fitting as well as create automatic design documentation and the industry standard JEDEC file. At this point, Logicmap II takes this JEDEC file, interfaces with the programming hardware, and produces a working EPLD!

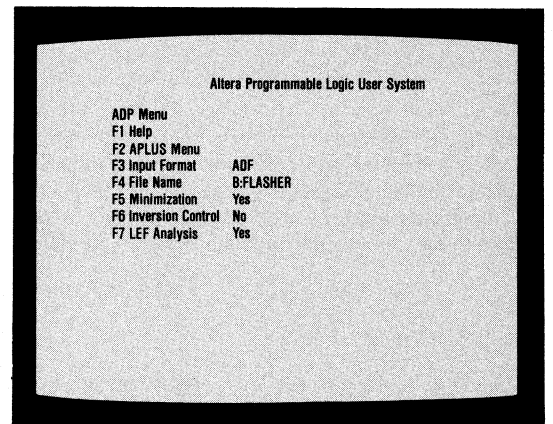
**Figure 12 - Printout of the netlist created in NetMap (filename: Flasher.ADF)**



**Figure 13 - The Design Cycle**



**Figure 14 - The ADP Menu (see Appendix in this Application Note for description of prompts)**



**Figure 15 - Display of a successful compilation in ADP**

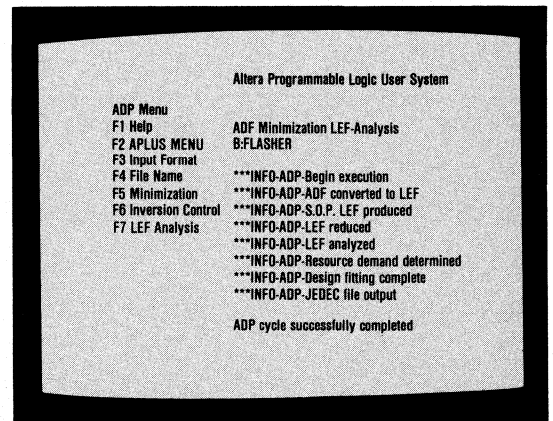
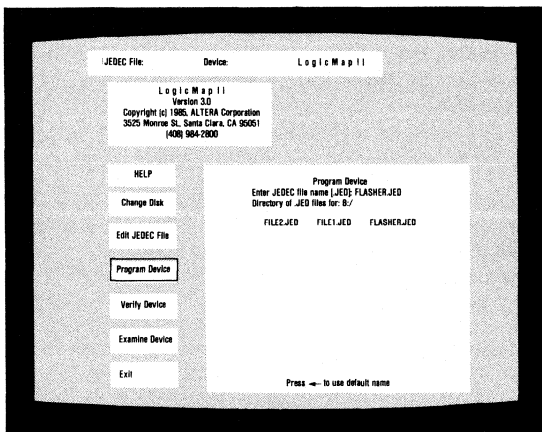


Figure 14 shows how the prompts in the Altera Design Processor (ADP) Menu should be answered and figure 15 shows what should appear on your screen following a successful run through ADP.

For a description of each of the messages in figure 15 as well as error message details, consult User Guide, section 7 and Appendix A in the Altera User Manual.

Figure 16 shows an example of the LogicMap II display when the "Program Device" option is selected.

Figure 16 - LogicMap II System Level Window



## CONCLUSION

In order to take full advantage of the vast benefits of EPLD technology, the logic designer must equip himself with PLDS2, the most advanced development system that allows for accurate and rapid product development. Using PLDS2, the design is captured, processed for optimum performance, and programmed to the target EPLD...all within minutes at the ultimate design environment...the logic designer's desktop.

## SYSTEM REQUIREMENTS

The PLDS2 recommended minimum system configuration is as follows:

- IBM PC, XT, AT or Equivalent
- 512K bytes of main memory
- Monochrome monitor (IBM)
- 2 (two) double-sided floppy disk drives
- Rev. 2.0 or later DOS operating system

## THE CONTENTS OF PLDS2

- A+PLUS Software
- NetMap
- LogicMap II
- Programming Card
- Programming Unit
- (2) EPLD Components
- User Manual

## DESIGN RECOMMENDATIONS

### 1. Obey architectural specifications of the EPLD.

Different EPLD's have different architectural characteristics. (For example, the EP300 has both a Synchronous Preset as well as an Asynchronous Clear control while the EP1200 has strictly the Asynchronous Clear control.) As a result, it may be beneficial to examine the device datasheets as well as portions of the user manual in order to gain an in-depth understanding of the device.

### 2. Inverted outputs may be implemented.

Simply invert (by inserting a "NOT" primitive) the signal going into the primary input of the I/O primitive.

### 3. Logic Primitives cannot feed back to themselves without first going through an I/O Primitive.

If Combinatorial Feedback is needed use the appropriate I/O Primitive (COCF and ROCF are available for the EP300; NOCF is available for the EP300 and EP1200). Latches may NOT be constructed within a macrocell using Logic Primitives only.

### 4. Pin Assignments may be made to any input or output signal name.

A pin number is assigned to a signal name by entering (@) and the specific pin number after the pin name. This may also speed the fitting process.

Example: Inputs: Onp@9,CLOCKp@ 1  
Outputs: REDA@13,REDB@18

### 5. VCC and GND are only used to feed P,C and Oe secondary inputs to I/O Primitives. They cannot be used as inputs in any other case.

### 6. P,C and Oe inputs to I/O Primitives do not require connections...when left unconnected, the following default connections are supplied:

P,C connected to: GND i.e., Disabled  
Oe connected to: VCC i.e., Enabled

### 7. The fan-out of each primitive is unlimited.

The user may apply an unlimited number of Logic Primitives within a design. The output of any primitive may act as an input to as many other primitives as desired.

## APPENDIX

This appendix gives a brief description of each menu encountered in each software module as well as a description of the functions available within each menu. For more details consult the Altera User Manual.

## NETMAP

## THE CONFIGURATION MENU

FUNCTION	DESCRIPTION	COMMENT
F1 Help	Gives help text for all functions in this menu.	Alternate to user manual.
F2 Main F3 Direction	Goes to the Main Menu provided a file has been selected (see F5 File). Toggles the direction of design entry as indicated by the arrow on the lower left side of the display. Default is outputs to inputs (right to left).	Use right to left.
F4 Primitives	Displays/hides the list of Altera Design Primitives in the upper right side of the display.	Keep in display mode.
F5 File	The DOS filename you wish this design to be saved and re-accessed as. May include a DOS path before the filename. DO NOT use a file extension (the file will be assigned the extension .ADF). A "?" will give a directory of all files on the current directory with the .ADF file extension.	DO NOT use a file extension. Ex: B:FLASHER C:/APLUS/FLASHER
F6 EPLD	Prompts for the Altera EPLD number for which you wish to target this design for.	Any member of the Altera EPLD family. Ex: EP1200
F7 Designer	Prompts for the name of the circuit designer.	Ex: I.C. DESIGNER
F8 Company	Prompts for the name of the company producing the circuit.	Ex: ALTERA CORPORATION
F9 Date	Prompts for the starting or completion date.	Any date is legal.
F10 Comment	Prompts for a one-line comment for this design.	Any text is legal.
†F1 Part Number	Prompts for the company's inventory part number.	Ex: #ABC-123
†F2 Revision	Prompts for the version identification.	Ex: version no.1
†F3 Inputs	Prompts for the input pin-names (each is separated by a comma or blank space). Use characters with an upper limit of 8 per pin-name. (To make pin assignments see "Design Recommendations.")	Use a "p" at the end of the name to designate a pin name. Ex: CLOCKp@1,G0p@3
†F4 Outputs	Prompts for the output or I/O pin-names (each is separated by a comma or blank space). Use characters with an upper limit of 8 per pin-name. (To make pin assignments see "Design Recommendations.")	Use a "p" at the end of the name to designate a pin name. Ex: Q1p@32,Q2p@33

## THE MAIN MENU

FUNCTION	DESCRIPTION	COMMENT
F1 Help	Gives help text for all functions in this menu as well as information describing each of the Altera design primitives.	You should read this before entering the design.
F2 Exit	Goes to the Exit Menu. Used for ending a NetMap session.	Use only when you're ready to end a session.
F3 New	Prompts for a device not yet associated with any other part of the circuit. The designer may choose to connect it to other nodes in the already existing circuit (by using common node names) or keep this "NEW" part of the circuit completely separated (by choosing unique node names).	Use this when you want to enter logic which is unrelated to the "current" part of the design you are working on.
F4 Open	Finds the most recently entered open node and prompts the user to make appropriate connections.	Use this to "pick up" where you left off.
F5 Find	Prompts for a node name to be found.	Use this to locate a node. Then use the arrow keys to scan the circuit.
F6 Edit	Goes to the Edit Menu for making changes to the current design.	In most cases, you need to name ALL signals associated with the primitive you are working on BEFORE editing.
F7 Config	Goes to the Configuration Menu.	You can make changes to your entries in the Config Menu at any time.
F8 Node List	Gives a list of all nodes that currently exist in the design. Nodes which appear in boldface print have not been completely connected.	Use this to see what nodes exist in the circuit.
F9 Redraw	Repaints the screen.	Hardly ever used.

## THE EDIT MENU

FUNCTION	DESCRIPTION	COMMENT
F1 Help	Gives help text for all functions in this menu.	MAKE SURE YOU READ THE HELP TEXT CONTAINED IN THIS HELP MENU.
F2 Main	Returns to Main Menu.	
F3 Reconnect	Prompts for a new node. The current node (see definition below) is disconnected from the primitive and the new node is connected in its place.	This is the most often used edit function. Think of it as picking up one end of a wire on your proto-board and plugging it in elsewhere.
F4 Disconnect	Disconnects only nodes that are not required.	Use this only on the third through twelfth input signals to a logic gate.
F5 Rename	Prompts for a new node name. The current node (see definition below) is changed to the new node everywhere in the circuit including the viewing area. This IS a global change.	This is a name change only.
F6 Chg Gate	Prompts for a new Logic Primitive then substitutes it for the one in the viewing area. The gate inputs remain intact.	This function may only be used on AND, NAND, OR and NOR gates.
F7 Del Prim	Removes the primitive in the viewing area from the circuit. If the nodes connected to this primitive are not connected elsewhere in the circuit, they are also deleted. If the nodes connected to this primitive are also connected elsewhere in the circuit, those connections will remain intact.	Use this as a "RESET" key when you want to go back a couple steps.
F8 Comment	Displays two percentage symbols (%%) within which the user may enter any comment associated with the viewing area. As a result, when the same viewing area appears in the Main Menu during design entry, or design review, the comment will also appear.	Use this for documentation purposes or to identify functions on your schematic.
<b>**NOTE**</b>	<p><b>DEFINITION OF "CURRENT NODE"</b></p> <p>The current node is that node which is highlighted in the viewing area. When a node name is entered to be connected to a primitive signal (e.g., C,P,Oe, Out,In1,...) that node name appears highlighted in the viewing area and becomes the "current node."</p>	To "force" a node to be the current node use the "Find" function (F5 in Main Menu) and type the name of the node. At that point, using the arrow keys allows you to scan the circuit.



## THE EXIT MENU

FUNCTION	DESCRIPTION	COMMENT
F1 Help	Gives help text for all functions in this menu.	
F2 Main	Returns to Main Menu.	
F3 Save	Saves the design file with an assigned file extension of .ADF. The file will be saved on the drive specified and with the name specified in the File function of the Configuration Menu.	
F4 Save As	Prompts for a filename and saves the design file with the specified filename.	
F5 Save-Exit	Saves the current design file then terminates NetMap. If the design file is not complete, i.e., open nodes still exist (an open node is a node that is not connected to a device in BOTH the input and output directions), the user will be informed and prompted on whether to continue with this function.	USE THIS 99% OF THE TIME.
F6 Exit	Terminates NetMap without saving the design file. If you attempt to use this function without saving your design file you will be prompted on whether to continue with this function.	Only use this when you DO NOT want to save the design.

## THE APLUS MENU

<b>FUNCTION</b>	<b>DESCRIPTION</b>	<b>COMMENT</b>
F1 Help	Gives help text on all functions in this menu.	
F2 Exit	Returns to DOS.	
F3 NetMap	Invokes NetMap module	For entering a design
F4 ADP	Invokes Altera Design Processor.	For processing a design.
F5 LogicMap II	Invokes LogicMap II module.	For programming devices.
F6 Directory	Prompts for a pattern which may be a drive name or path name with any matching pattern including an asterisk (*). A list of files matching the pattern will be displayed.	
F7 Rename File	Prompts for an old and a new filename.	
F8 Copy File	Prompts for an old and a new filename. The old file is copied under the new filename.	
F9 Delete File	Prompts for a file and then deletes it.	
F10 DOS Command	Allows user to execute any DOS function.	

## THE ADP MENU

FUNCTION	DESCRIPTION	COMMENT
F1 Help	Gives help text for all functions in this menu.	
F2 APLUS	Returns to the APLUS Menu.	
F3 Inp Fmt	Prompts for the type of file to be processed. You may enter a question mark (?) to see the choices that you have. Respond by hitting: A for Altera Design File (ADF) P for FutureNet Pinlist (PIN) C for PCAD Component List (CMP)	Respond by hitting "A" for all designs entered with NetMap.
F4 File Name	Prompts for the name of the design file you wish to process. If you enter a (?), a list of files based on the Inp Fmt is displayed. If you then enter a drive name or a directory name, the corresponding list of files will be displayed. When entering the name of the file, do NOT use a file extension.	Type your file name as saved in NetMap. DO NOT use a file extension. EX: B:FLASHER C:/APLUS/FLASHER
F5 Minimization	Allows user to request/reject reduction of the Boolean logic in his design by responding with a Y/N keystroke.  If you answer YES (Y) to this prompt, your design will be minimized.  If you answer NO (N) to this prompt, no minimization will be performed on your design during processing time.	Choose "YES" 99% of the time for minimization.
F6 Inversion Control	If you answer NO (N) to this prompt, ADP will not only perform Boolean minimization but it will also perform de Morgan's inversion in simple cases where the "de Morganized" form requires fewer product terms than the non-inverted, minimized form. This optimizes the number of product terms used to implement your design.  If you answer YES (Y) to this prompt, ADP goes through the same process as if you had answered NO to this prompt, but it also prompts the user as to whether he would like to perform de Morgan's inversion for each of the previously minimized equations. The user will see the number of product terms used in a given equation as well as whether the equation has been "de Morganized" in the ADP's first pass at minimization. (If the left hand side of the equation is complemented then it has already been de Morganized to achieve the fewest number of product terms utilized. Otherwise, the non-complemented left side indicates the ADP did not perform deMorgan's inversion.) Upon inspection of the number of product terms as well as the left hand side of the equation, the user is then prompted as to whether he would like to "de Morganize" the equation in its current status. In effect, the user may choose to "re-de Morganize" equations with an inverted left side or "de Morganize" equations that have non-complemented left sides.	Choose "NO" 99% of the time for inversion control. This prompt will only occur when you answer YES to minimization.
F7 LEF Analysis	Allows user to output a file (with the name: <filename>. LEF) with minimized equations.	This LEF file is in the ADF format.

### INTRODUCTION

A large class of applications for digital logic involve serial data transmission and reception. Altera's new high density Erasable Programmable Logic Devices (EPLDs) provide the ability to integrate serial data communication logic with other processing or control logic, reducing system chip count and cost. In addition, they provide the advantages of low power and high speed: for example the EP310 is capable of a clock rate over 40 MHz giving a baud rate of over 10Mbaud.

This application note focuses on the conversion of serial data to and from parallel data using EPLDs. The designs presented should be thought of as 'building blocks' that could be modified and combined with other logic specific to your application needs.

Examples where these circuits will be of use include single chip remote data acquisition systems, where part of the EPLD is used for the serial communications and another part controls the analog to digital conversion of the input signal. A stepping motor controller or a remote display are examples in which a serial receiver would be combined with other logic in a single EPLD. Other applications include interfacing hardware to a communications system such as a Local Area Network (LAN). An internal state machine in the EPLD would handle the protocol and handshake requirements of the buss and other logic could provide an interface to the hardware being connected.

### SERIAL TRANSMITTERS

Serial transmission involves converting a parallel data word to a serial bit stream, and appending to it synchronization and error detection information.

The synchronization information used here consists of a START bit preceding the data, and one or more STOP bits trailing. Many forms of error detection have been devised, the most simple is to append a parity bit to each word. This application note will discuss parity generation and testing in a later section.

There are several ways to approach the design of a serial transmitter. Serial Transmission may be performed with a digital 'switch' or multiplexer. The switch is set at a logic high while waiting to transmit. Upon initiation of a transmission, the switch selects a logical low to generate a start bit, then scans the parallel inputs one at a time at the desired baud rate, finally ending up at a logic high where it awaits the next transmission. Figure 1 shows this concept of transmission, while Figure 2 shows a block diagram of an actual implementation of this concept.

Although the multiplexer design is simple in concept, it tends to use up more of the EPLD resources than is desirable, thus leaving less for other logic that could

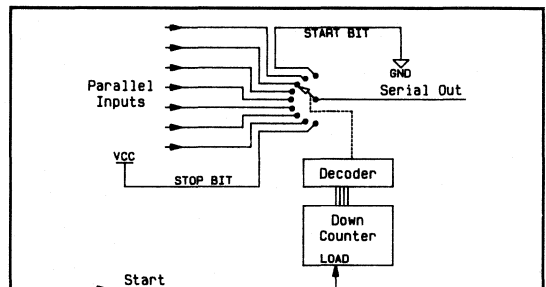


Figure 1.

Transmission by sequentially routing the parallel data to the serial line. See Figure 3 for an actual implementation of the switch and decoder portion of this arrangement.

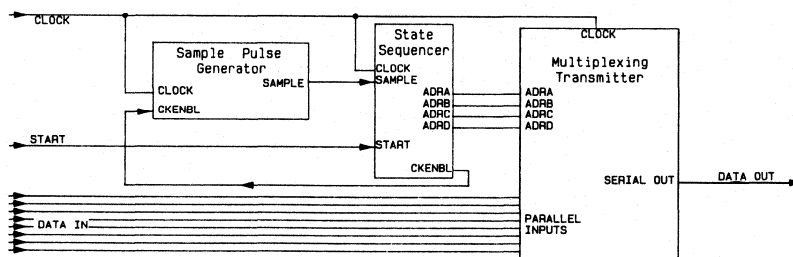
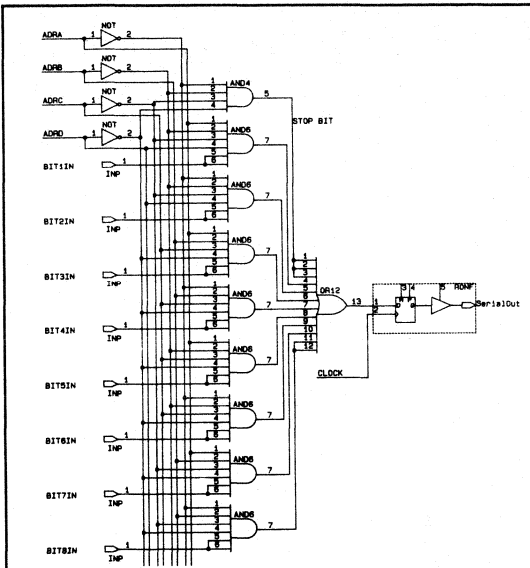


Figure 2.

Block diagram of the 8 bit multiplexing type Transmitter shown in Figures 3, 4, and 5.

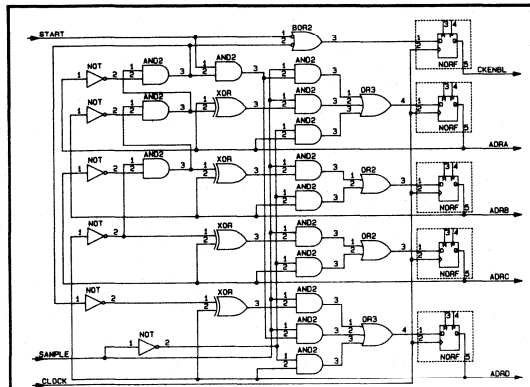


**Figure 3.**

A multiplexing transmitter has the advantage of only requiring one output register, but the inputs must be stable for the duration of the transmission.

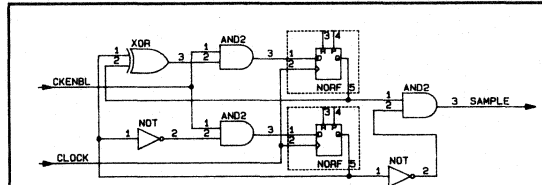
be implemented with the device. As the width of the data word increases, the size of the resulting logic equations increases, and at some point would become too large to fit the resources of the chip.

An alternative approach is to use a shift register with additional control logic. A shift register has the advantage that each register is fed only from its nearest neighbor. The size of the resulting logic equation no longer increases as more bits are added to the data



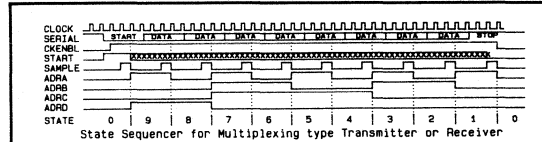
**Figure 4.**

A down counter is used for keeping track of which bit is accessed at any given time in the multiplexing transmitter or demultiplexing receiver. When the transmission sequence is initiated, or a start bit is detected in the case of the receiver, this counter is set to a binary 9 (for 8 bits of data, plus the start bit) and then counts down to zero, the idle state. The value of the counter is decoded in Figures 3 and 13 for the routing of the data.



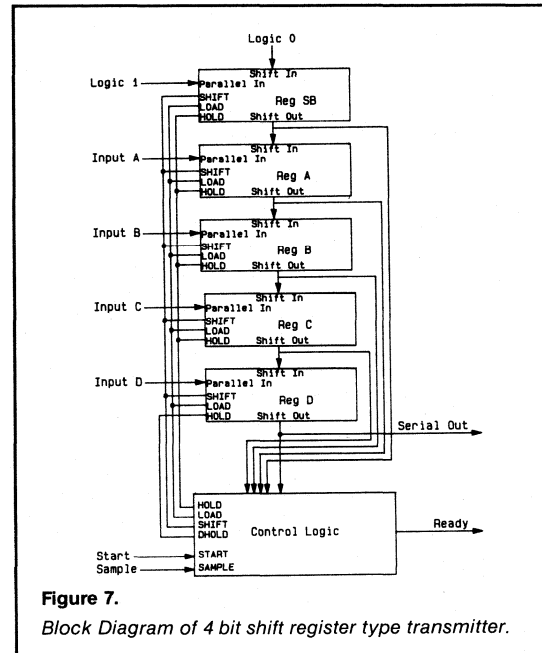
**Figure 5.**

Circuit for generation of the sample pulse required by the demultiplexing receiver. This pulse is timed such that the serial input line will be sampled near the middle of each bit period to reduce errors.



**Figure 6.**

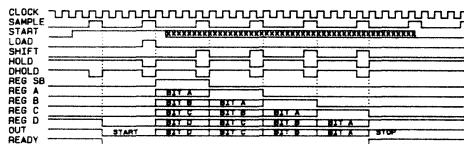
Timing diagram for the 8 bit multiplexing circuits. State 0 is the idle state, state 9 is the start bit, and states 8 to 1 are the data transfer states. Note that the SERIAL data shown here represents data being input to the receiver. Data being output from the transmitter would be aligned with the state transitions on this diagram.



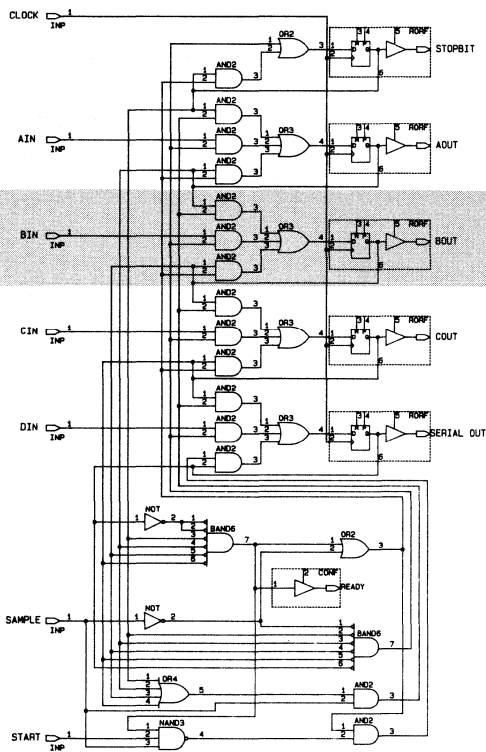
**Figure 7.**

Block Diagram of 4 bit shift register type transmitter.

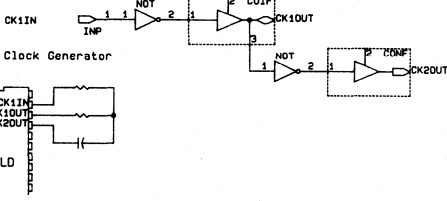
width. The design shown in Figures 7, 8, and 9 treats the shift register as a state machine with the current state being a function of the position of the stop bit in the register. The operation sequence is as follows: When the circuit shown in Figure 9 is idle, waiting for the start command as asserted by a TRUE on the START input, all of the flip flops in the shift register (figure 9) are at a



**Figure 8.**  
Timing Diagram for Transmitter shown in Figure 9.



**Figure 9.**  
Schematic of 4 bit shift register type transmitter. The word width may be increased by insertion of more register 'stages' identical to the logic within the box.



**Figure 10.**  
An astable multivibrator may be built from two macro-cells to create a truly stand alone integrated design.

logic zero, with the exception of the serial output register which is at a logic ONE. When the START input is asserted, the serial output register is reset to logic zero on the next cycle of the SAMPLE signal. This has the effect of initiating the start bit in the data stream. Four clock cycles later, the next time SAMPLE is asserted, the flipflops in the shift register are loaded with the parallel data. The last flip flop, STOPBIT, is set to a logic one. Each time thereafter that SAMPLE gets asserted, (one quarter the clock rate), all of the data including the final stopbit is shifted DOWN, and a logical zero is shifted in from the TOP. This zero propagates DOWN until the logic detects that the whole register with the exception of SERIAL OUT is at zero. When this occurs, the transmission is complete, the high stop-bit is residing on the serial output line, and no more shifting is permitted until the sequence is started again.

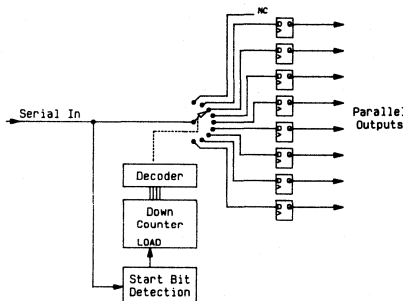
### SERIAL RECEIVERS

Reception of the serial data may be accomplished by use of a demultiplexer in a manner similar to transmission by multiplexing. In this case the sequence is initiated by detection of a start bit (logic Low) on the serial data line. The demultiplexing switch (see Figure 11) then routes the data from the serial input to the parallel outputs at the baud rate until the expected number of bits have been read.

A sample pulse running at one quarter the clock rate is timed to insure that the receiver samples the incoming data near the middle of the bit time, thus reducing the possibility of errors due to transmission line effects and differing clock rates.

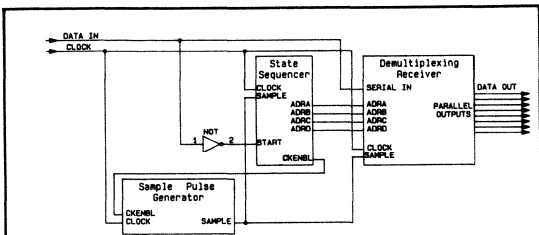
A shift register type receiver is an effective alternative and is shown in Figures 14 and 15. The operation of this design is as follows:

In idle mode, while waiting for a stop bit on the serial input line, the registers A to D hold whatever data was

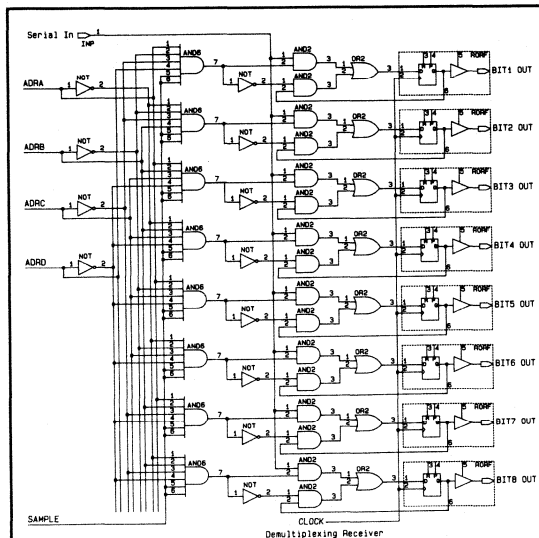


**Figure 11.**  
Serial data may be received by routing it sequentially to the appropriate parallel output registers, or 'demultiplexing,' at the baud rate. This is the complement to the transmitter shown in Figure 1.

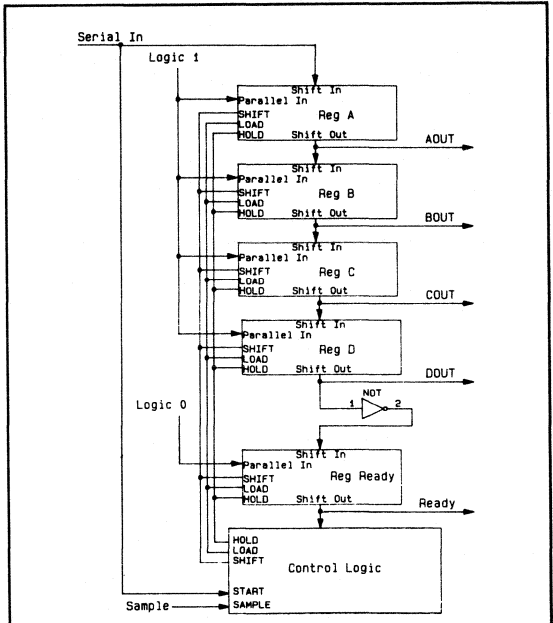
previously received, the READY register is asserted, and the timing counter PHASE 1, PHASE 2 is idle in the state with both registers at a logical Low. Detection of a logic Low on the serial input line is assumed to be a start bit. When this start bit is received, the timing counter is allowed to begin counting, the READY register is brought Low, and all other registers in the circuit are set to a logic High. Two clock pulses later the start bit, still on the serial line, is shifted into register A. Data from the serial input line is shifted in every 4 clock cycles thereafter, until the start bit propagates down to the READY register. When this occurs, the circuit again becomes idle, awaiting a new start bit, but with the newly received data available on the parallel outputs.



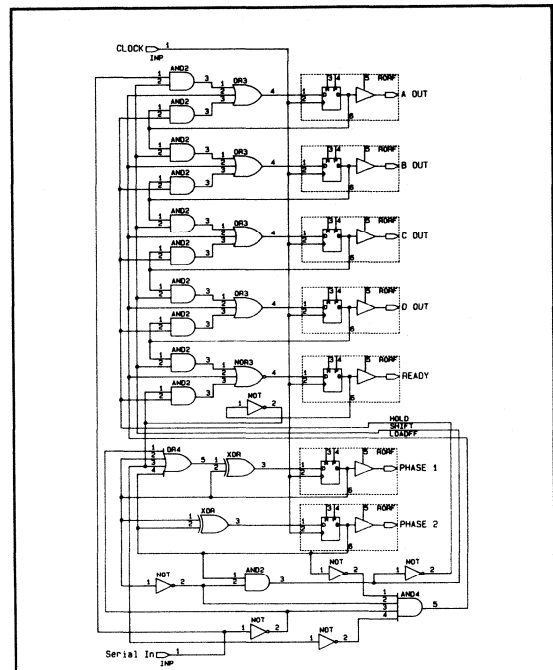
**Figure 12.**  
Block Diagram of 8 bit Demultiplexing type receiver. The schematics for the State Sequencer and the Sample Pulse Generator are shown in Figures 4 and 5.



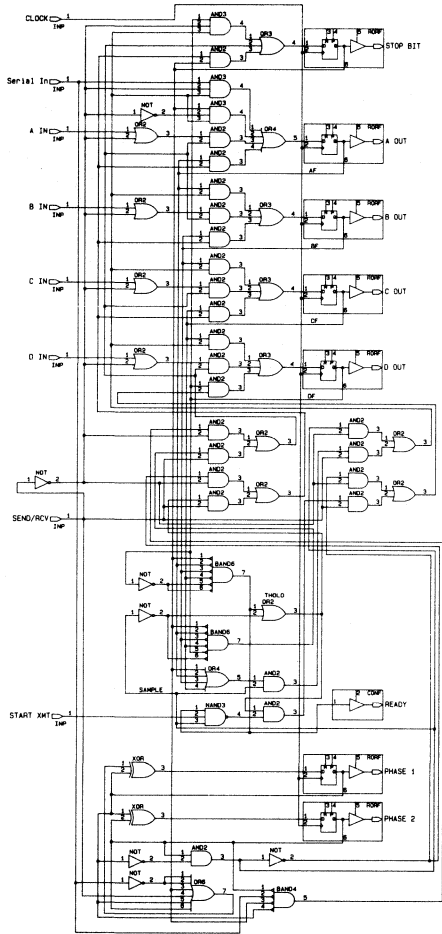
**Figure 13.**  
Shows a schematic for the demultiplexing type receiver. The control lines ADRA to ADRD come from the Controller shown in Figure 4.



**Figure 14.**  
Block Diagram of the 4 bit shift register type receiver.



**Figure 15.**  
Schematic diagram for shift register type receiver. Note that the timing circuit uses the serial input signal to initiate the counting sequence. This is to set the sample pulse near the middle of the 'bit time' to avoid errors due to data skew, ringing, and other transmission effects.



**Figure 16.**

Shift register transmitter and receiver are combined into a transceiver. It would also be possible to use the multiplexer transmitter or demultiplexer receiver for their respective halves of this circuit.

## SERIAL TRANSCEIVERS

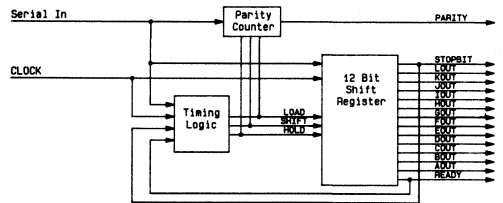
One of the significant benefits the designer has when using schematic capture as an input method for logic designs is that he may combine different functional logic blocks together by the simple use of an AND—OR multiplexer. One may then let the design processor worry about combining the resulting boolean equations and removal of redundant terms. This technique was used for the transceiver shown in Figure 16. Note the multiplexers near the center of the figure. The receiver control logic is at the bottom of the drawing, with the transmitter control logic immediately above. The top half of the drawing is the shift register used for both

receiver and transmitter, as selected by the SEND/RCV input line.

## ADDING PARITY

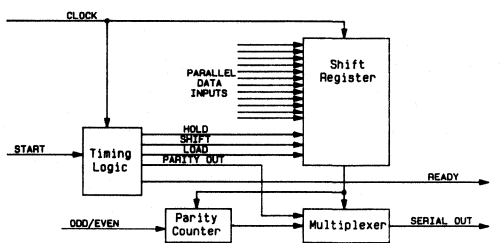
The logic required to generate parity of a parallel data word is not well suited to the sum of products form required by EPLDs because the resulting equations become very long for more than three bits of data. By generating the parity from the serial data the complexity of the required logic is much reduced and is no longer a function of the word width.

Parity by definition is simply a bit that tells if the data has an even or odd number of ones (or zeros). A modulo-2 counter that counts the ones in the serial bit stream is the obvious choice and a simple design to incorporate. Checking the parity at the receiving end is done by including the incoming parity bit in the count. This can be done because the data word, including the parity bit, will always have the same overall parity (unless a transmission error exists). Figures 19 and 20 show a 12 bit receiver and transmitter that incorporate parity. Note the increase in the complexity of the transmitter due to the insertion of the parity bit into the serial data stream just prior to the stop bit.



**Figure 17.**

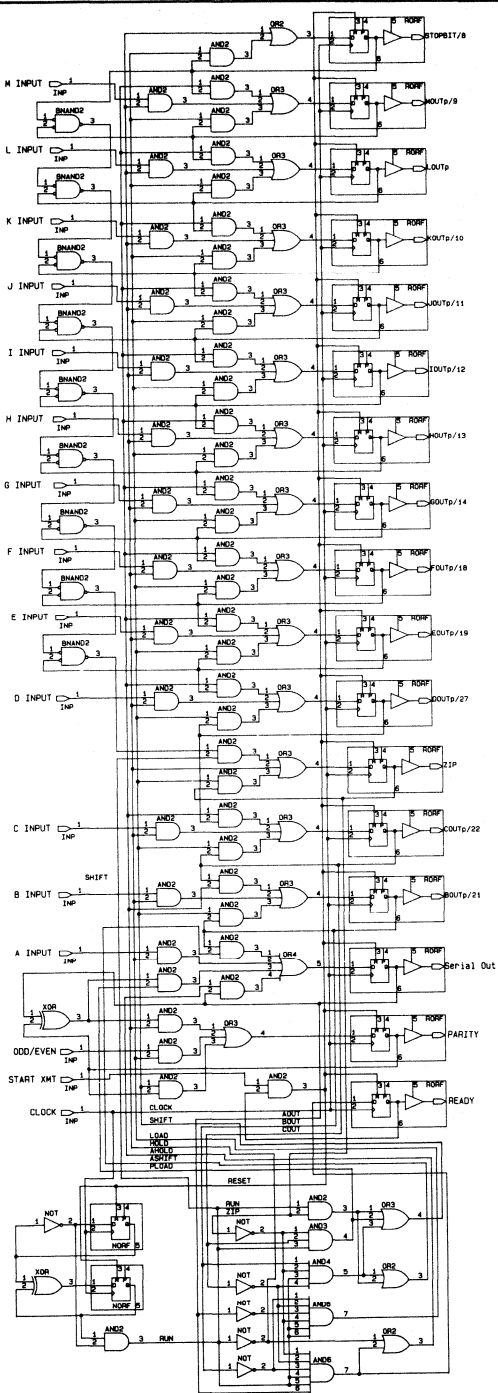
Block diagram of a receiver with parity check. The parity of the incoming data should always be the same, if you include the incoming parity bit.



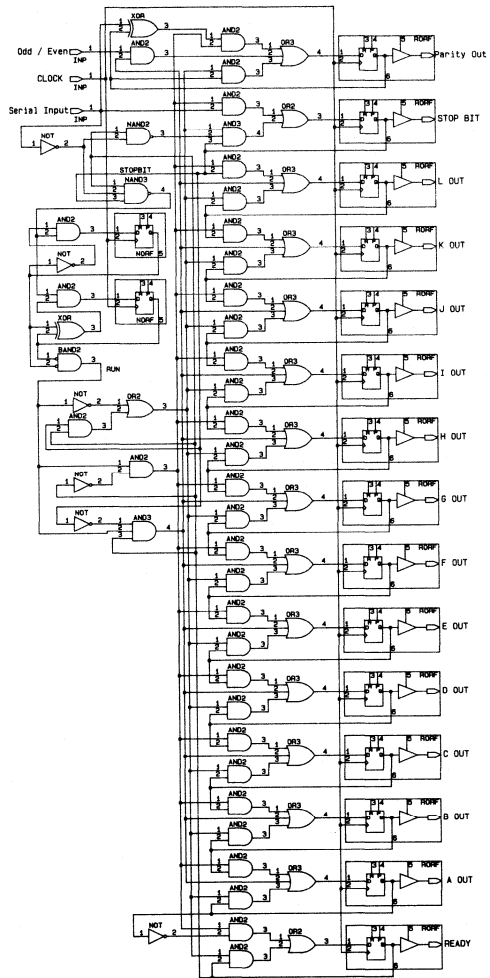
**Figure 18.**

Block diagram of a transmitter with parity. The parity is calculated from the serial data stream, then inserted at the appropriate time prior to the stop bit.





**Figure 19.**  
 Schematic of a 12 bit transmitter with parity. The register 'ZIP' asserts when the upper three quarters of the shift register is clear, this reduces the size of the resulting equations that detect when to insert the parity bit and when to stop transmitting.



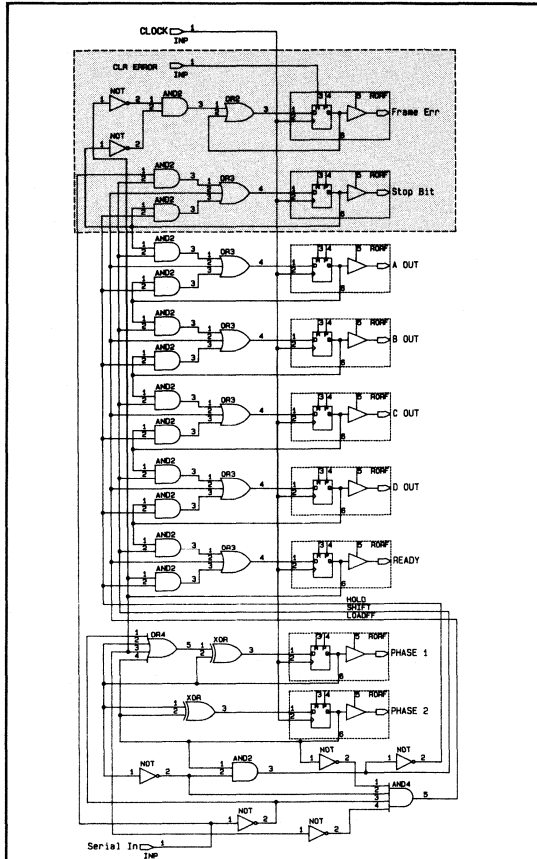
**Figure 20.**  
 Schematic of a 12 bit receiver with parity checking. The parity check is done with a modulo 2 counter that counts incoming ones including the incoming parity bit. Presetting this counter switches from odd to even check.

**BELLS AND WHISTLES**

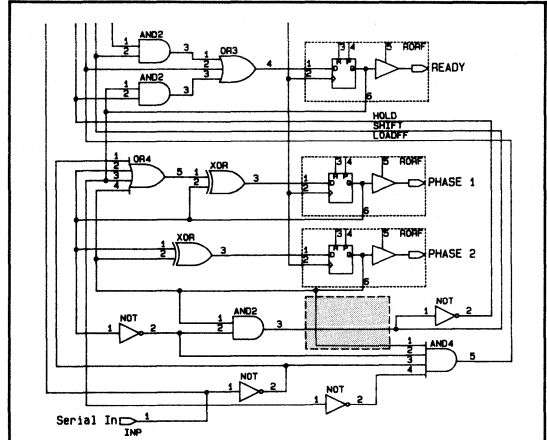
A number of enhancements may be added to these designs, dependent upon the requirements of the final system. Some possibilities are:

- Double buffering of the parallel data
- Cyclic Redundancy Checking
- Error Correction
- Latching of parity error bit
- Frame error detection
- Overrun detection

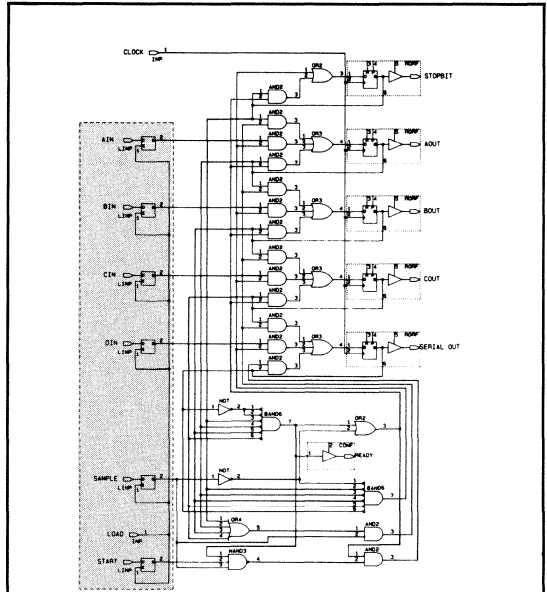
Examples of some of these are shown in the figures on this page.



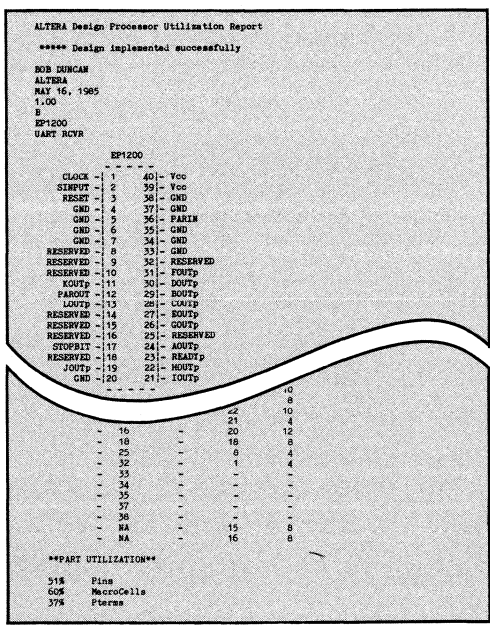
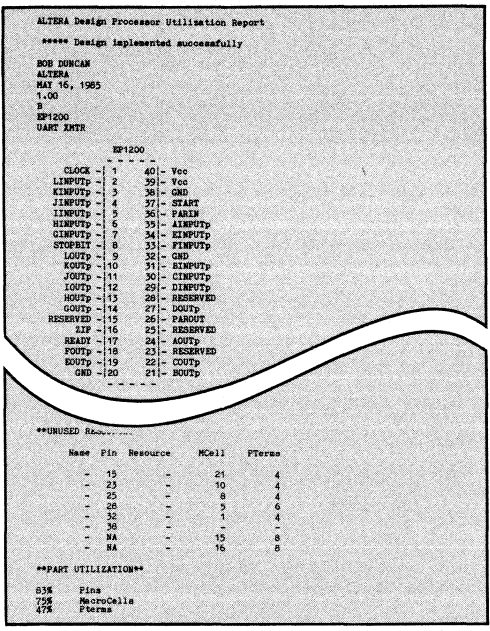
**Figure 22.** Frame error detection may be obtained by the inclusion of an additional stage in the receiver shift register. The additional stage, marked as STOPBIT, is at the input end of the register and thus gets loaded with the stop bit following the reception of the serial data. If this flip flop is not at a logic HIGH when the READY signal becomes asserted, the Frame Error flip flop gets set and remains so until reset by an external signal. See Figure 15 for the original circuit.



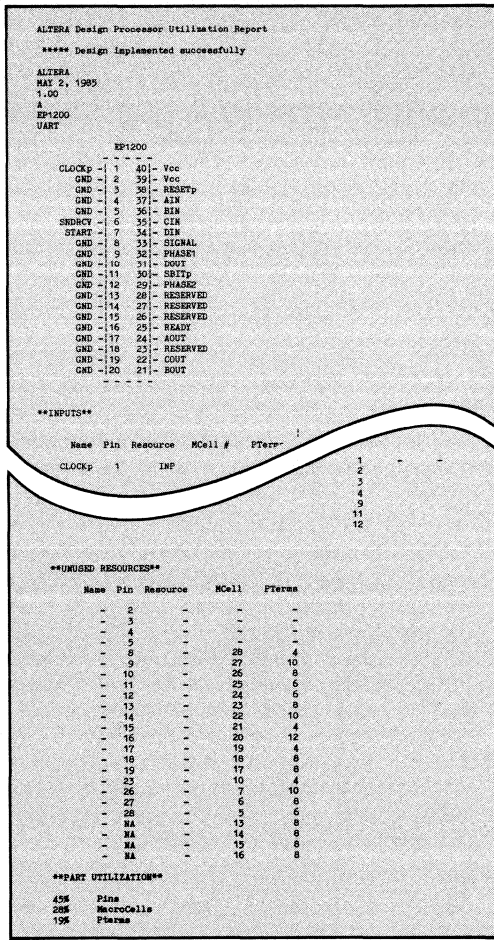
**Figure 21.** Removal of one NOT gate in the receiver timing logic allows automatic recovery from false start bit detection. See Figure 15 for the original circuit. The removal of this NOT gate has the effect of delaying the pulse that sets all of the flip flops in the shift register to initiate the receive sequence. The flip flops now are not set unless the serial input line is still at a logic LOW in the center of the bit time that started the counters. If this condition is not met, the data in the register is preserved and the system returns to its idle mode of waiting for a start bit.



**Figure 23.** The EP1200 programmable input latches may be used for buffering the parallel inputs to the transmitter, simplifying the timing requirements for logic feeding these inputs.



**Figure 24.** The Altera Programmable Logic User System (A+PLUS) generates a Utilization Report showing how the design was fitted to the EPLD. The above is a partial copy of the utilization reports generated for the 12bit UART transmitter (top) and for the 12 bit UART receiver (bottom). A+PLUS also generates a JEDEC file that would be used to electrically program the chip, completing the design cycle from concept to silicon.



**Figure 25.** Partial utilization report for integrating the 4 bit UART transceiver in an Altera EP1200. 15 macrocells with associated I/O pins are available as well as 6 dedicated input pins and 4 buried macrocells, for use by other logic in the system.

## CONCLUSION

User programmable logic has entered a new era with the introduction of the high density Altera EP1200 and EP600 EPLDs (Eraseable Programmable Logic Devices). Using these chips, engineers may design close to the system level, integrating individual functional blocks to form the overall design. The serial transmitter and receiver are examples of such functional blocks. By combining these blocks with others (counters, phase detectors, state machines, etc.) it is often possible to integrate an entire sub-system on one chip.

This TTL library consists of over 100 designs of standard 74- and 93-series TTL parts.

This library provides many examples of applications for Altera EPLDs, illustrating a wide variety of designs: counters, flip-flops, decoders, multiplexers, shift registers, adders, etc. In addition, this library serves as a valuable design tool since individual part designs can be used as building blocks for larger designs.

Following are brief descriptions of the contents of files included in the library.

**TEXT (.TXT):**

The text file contains a short description of the design, including a comparison between the design and the TTL part it represents. Most text files also include a list of various AC delay parameters and operating requirements found during evaluation. **NOTE:** The numbers listed are the result of a one-time evaluation of the design and do not necessarily represent worst case or typical numbers. Please refer to Application Note 4 for timing analysis of Altera EPLDs. The text file also includes information such as the part number of the chip used for evaluation and the number of macrocells required for the design.

**DRAWING (.DWG):**

This file contains a complete FutureNet schematic of the design.

**AREA (.ARA):**

This file is similar to the .DWG file except that input and output pins are excluded. It is useful if you want to include the TTL part into your design. With FutureNet, you can 'call up' the .ARA file and insert it into your design by making the appropriate connections. For added convenience, input and output connections of the .ARA files have been labeled.

**NOTE:** The RORF and COCF output primitives of the .DWG files have been converted to NORF and NOCF primitives in the .ARA files. If you wish to use these as outputs, they must be converted to their original form.

**UTILIZATION REPORT (.RPT):**

The Utilization Report indicates how the resources of an EPLD have been used by a particular design. The report includes a listing of pin assignments as well as a section which shows the percentage of used pins, macrocells, and product terms.

**ALTERA DESIGN FILE (.ADF):**

One of the most convenient ways to create a design is to enter Boolean equations directly into an Altera Design File. This ADF is then entered into the A+PLUS Altera Design Processor (ADP) which performs logic minimization (if requested) on the Boolean equations. The ADP next fits the design into the target Altera EPLD, producing a Utilization Report that shows which of the part's resources were consumed by the design and how. This fitted design is then transformed into a JEDEC Standard File, which is used to program an Altera part.

This library contains many examples of ADFs. All of the Boolean equations in these ADFs are minimized equations.

A few conventions adopted by the ADF files in the TTL library are worth noting:

-Input and output pin names are appended with a 'p' to differentiate them from internal signal names.

-Feedback signal names have been appended with an 'f.'

-If active LOW outputs are required, the right hand side of the Boolean equation must be inverted. In the ADFs in this library, equations of this type are marked by the comment '% Inverted Equation %.'

To illustrate the contents of the TTL library, the 74393 modulo-16 counter is used as an example.

Figure 1 shows the 74393.TXT file. Operating characteristics of the design and data comparing the library design and the TTL part it simulates are listed in the description section of the file. Next, the comments section shows the delay parameters found during evaluation of an Altera EP300 programmed with the 74393 design. Remember, these figures are the result of

a one-time evaluation of the part. They should not be used as reference numbers for timing analysis. Also included in this portion of the ADF is the part number of the chip used during evaluation as well as the number of macrocells utilized by the design. It is important to refer to this text file when you use a design; it will aid you in understanding the operation of the part.

**Fig. 1 74393**

74393

Description: The 74393 is a modulo-16 counter. Each output is synchronous with the rising edge of the clock. The Reset input, when set HIGH, forces all outputs to the LOW state. The TTL 74393 package contains two of these counters. Also, the TTL model is a ripple counter with the first stage synchronous with the falling edge of the clock.

Comments: EP300 AC CHARACTERISTICS:  $V_{CC} = 5V$ , Temp = 25°C.

SYMBOL	DELAY PARAMETER	APPROXIMATE DELAY
t <sub>PHL</sub>	Clock to Q <sub>n</sub>	24 ns
t <sub>PHL</sub>		21 ns
t <sub>PHL</sub>	Reset to Q <sub>n</sub>	65 ns

AC testing conducted with load conditions as shown in Fig. 9 of the EP300 data sheet.

EP300 OPERATING REQUIREMENTS:

SYMBOL	DELAY PARAMETER	APPROXIMATE DELAY
t <sub>rec</sub>	Reset to Clock	96 ns

Test part number: EP300

Number of macrocells required: 4

The device has been tested and shown to implement the required function.

Date: March 4, 1985

**Fig. 2 74393 Modulo-16 Counter**

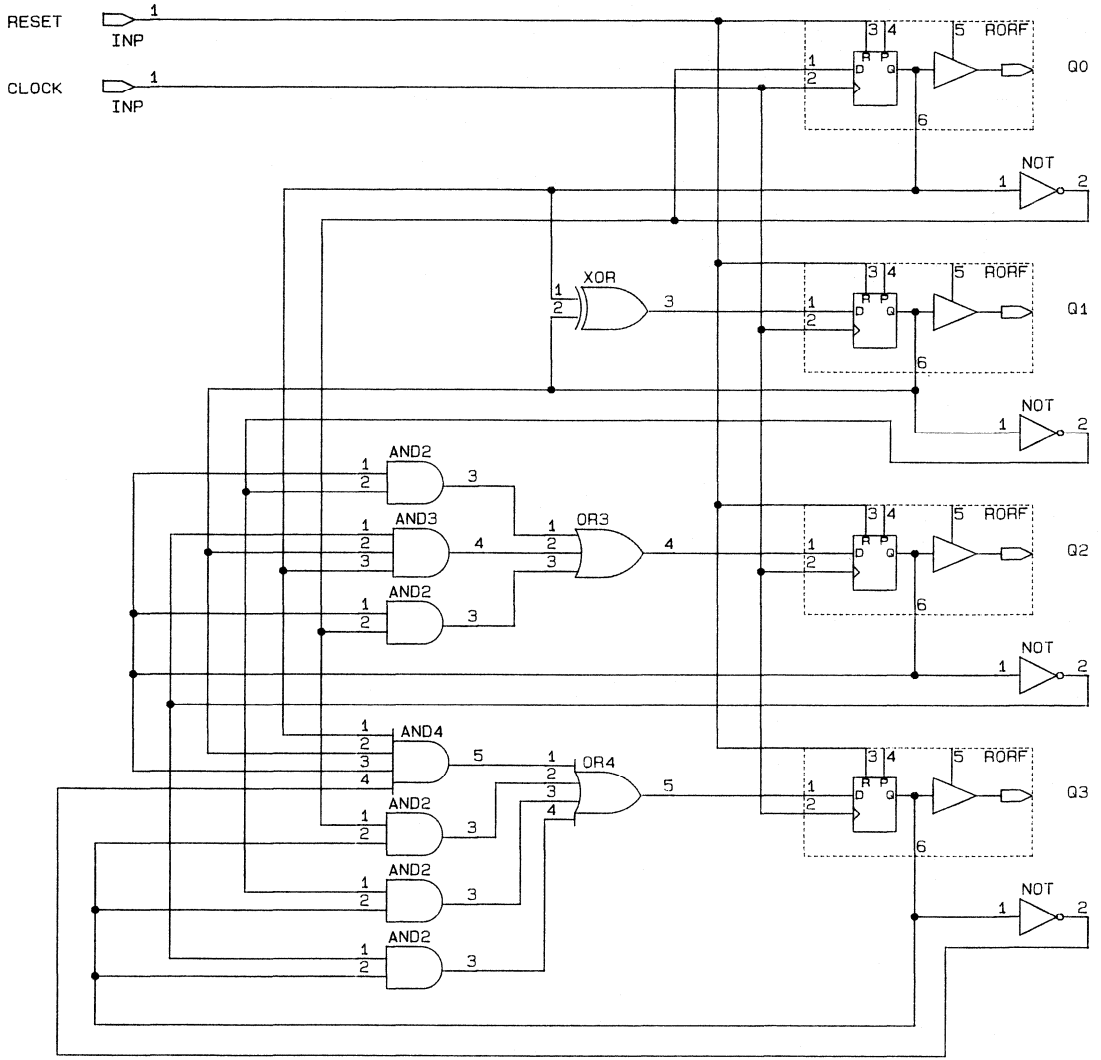


Figure 2 shows the 74393.DWG DASH-2 schematic file. It is a complete schematic of the design, including input and output primitives and title block. From this .DWG file, the DASH-2 Pin List Processor generates a pinlist. Altera's A+PLUS Design Processor then converts the pinlist file into an Altera Design File from which, in turn, a JEDEC Standard File is created.

Schematic capture is a convenient method of entering and processing designs. You can quickly and easily revise existing drawings and generate hard copy printouts of the entered design. Refer to the Schematic Capture sections of the A+PLUS User Guide for further notes and design considerations.

Fig. 3

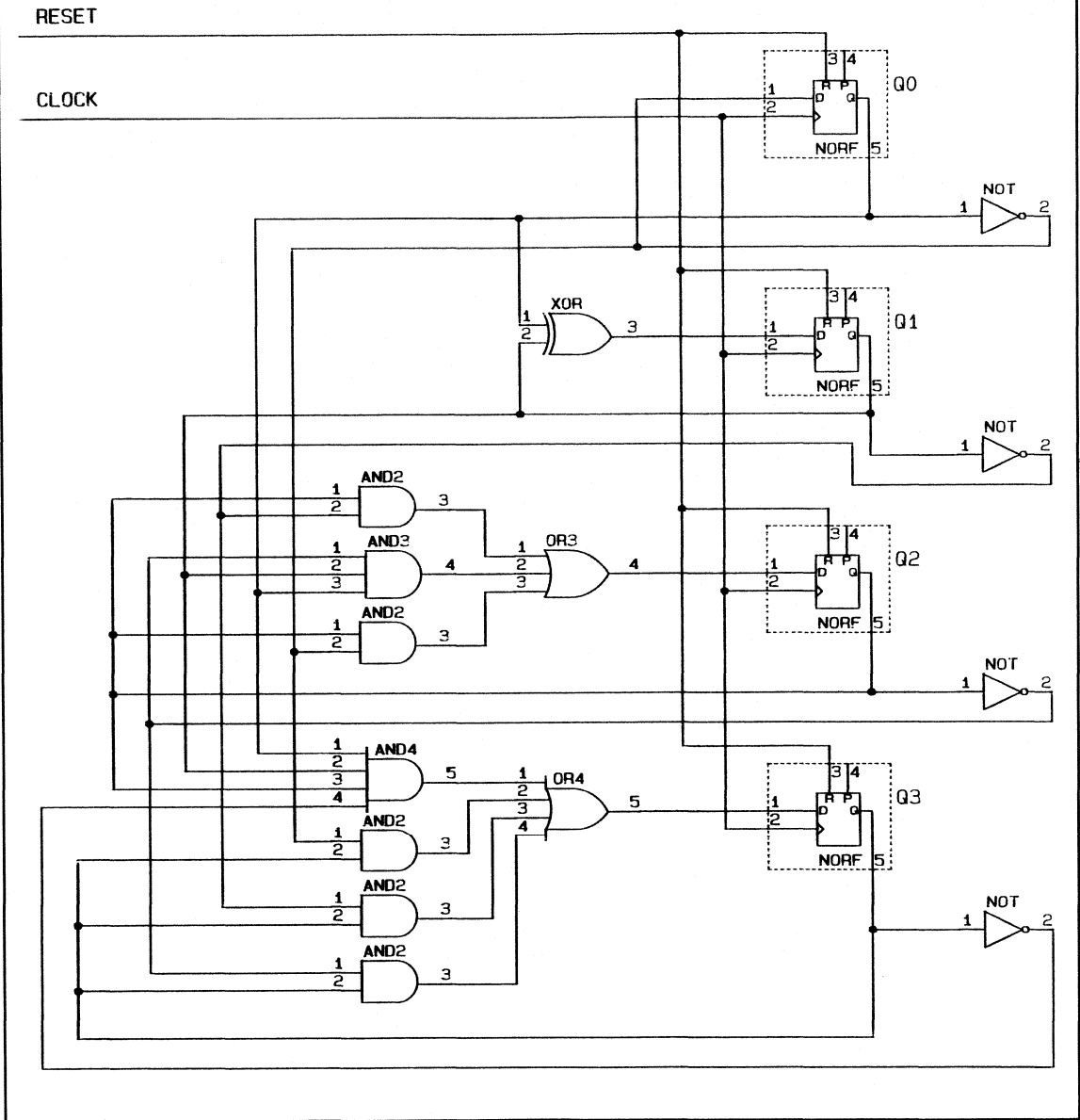


Figure 3 shows the 74393.ARA DASH-2 area file. Comparing it to the .DWG schematic, you will notice that the input and output primitives have been removed, the RORF output primitives having been replaced by NORFs. This characteristic allows you to use the area files as building blocks for larger designs. However, if you wish to use this 74393.ARA schematic in a design and desired Q0, Q1, Q2, Q3 as outputs, you must convert the NORFs back to RORFs.

The following simple example demonstrates this schematic design technique. Figure 4 consists of two area files: 74393.ARA modulo-16 counter and 7449.ARA BCD to 7-segment display decoder. With these two area files a single complete schematic consisting of both the counter and the decoder can be easily produced: input and output primitives must be added, a title block must be entered, and the appropriate connections between the two area files must be made. The complete schematic is shown in Figure 5.

Fig. 4

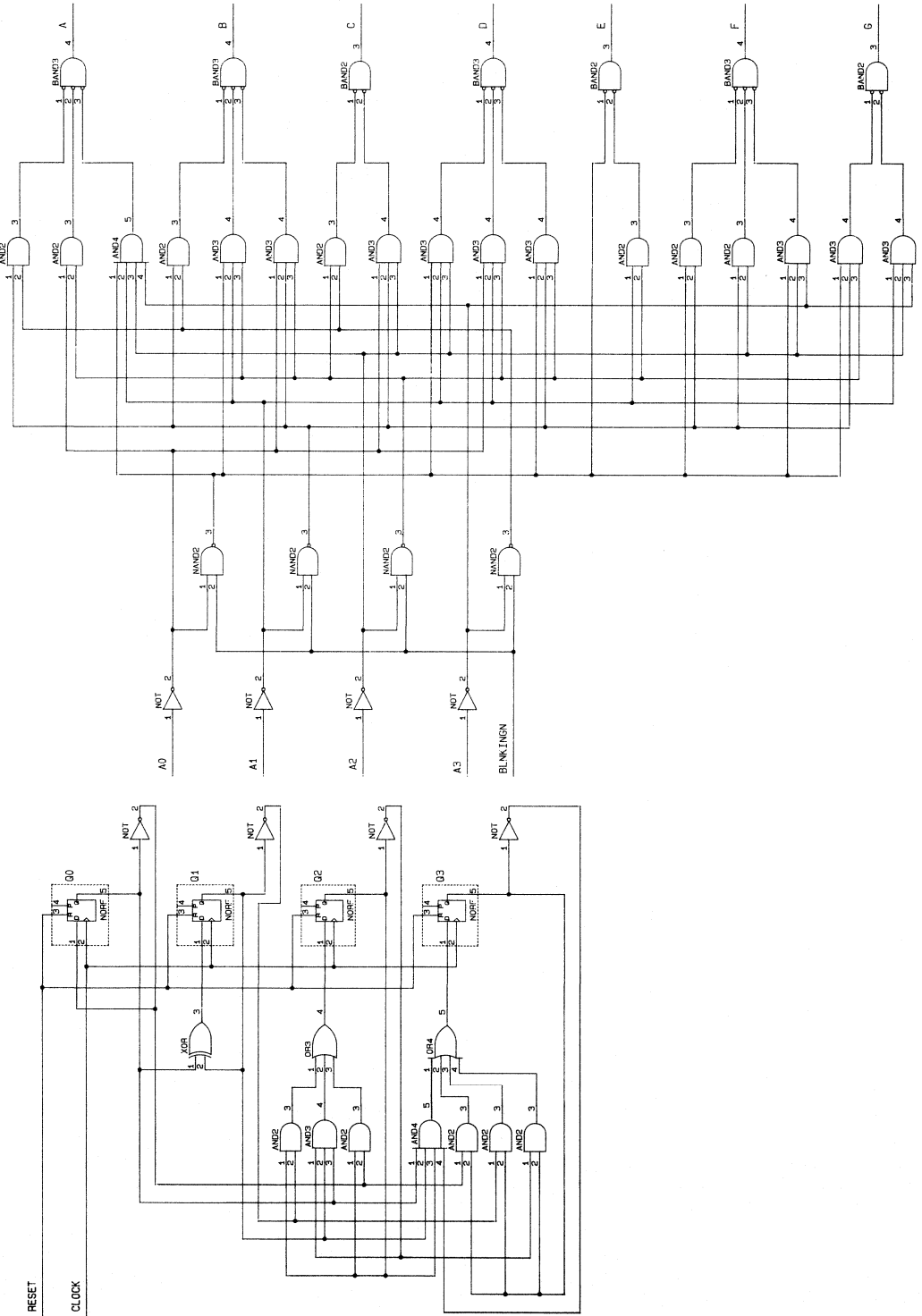
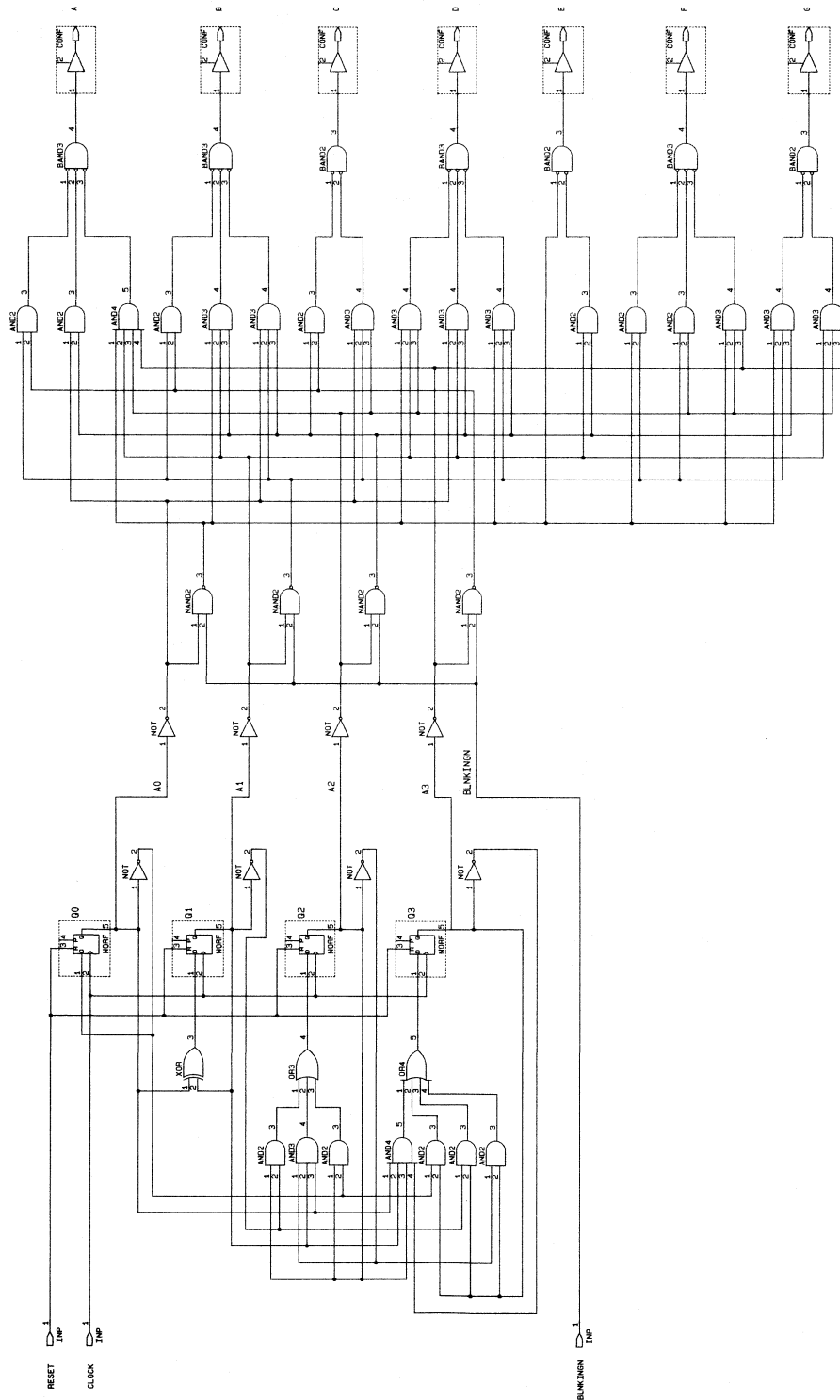




Fig. 5



**Fig. 6**

ALTERA Design Processor Utilization Report

\*\*\*\*\* Design implemented successfully

BOB VENABLE  
 ALTERA CORPORATION  
 MAR. 4, 1985  
 1.00  
 A  
 EP300  
 74393 MODULO-16 COUNTER

EP300			
CLOCK	- 1	20	- Vcc
RESET	- 2	19	- Q3
GND	- 3	18	- Q2
GND	- 4	17	- Q1
GND	- 5	16	- Q0
GND	- 6	15	- GND
GND	- 7	14	- GND
GND	- 8	13	- GND
GND	- 9	12	- GND
GND	- 10	11	- GND

\*\* INPUTS \*\*

Name	Pin	Resource	MCell #	PTerms	Feeds:			
					MCells	OE	Reset	Preset
CLOCK	1	INP	-	-	-	-	-	Reg
RESET	2	INP	-	-	-	-	1	-

\*\* OUTPUTS \*\*

Name	Pin	Resource	MCell #	PTerms	Feeds:			
					MCells	OE	Reset	Preset
Q0	16	RORF	4	1/ 8	1	-	-	-
					2			
					3			
					4			
Q1	17	RORF	3	2/ 8	1	-	-	-
					2			
					3			
Q2	18	RORF	2	3/ 8	1	-	-	-
					2			
Q3	19	RORF	1	4/ 8	1	-	-	-
					2			

\*\* UNUSED RESOURCES \*\*

Name	Pin	Resource	MCell	PTerms
-	3	-	-	-
-	4	-	-	-
-	5	-	-	-
-	6	-	-	-
-	7	-	-	-
-	8	-	-	-
-	9	-	-	-
-	11	-	-	-
-	12	-	8	8
-	13	-	7	8
-	14	-	6	8
-	15	-	5	8

\*\* PART UTILIZATION \*\*

33% Pins  
 50% Macrocells  
 15% Pterms

Moreover, this example demonstrates that these area schematics - even though most are EP300 designs - can be used to create schematic designs for any of Altera's parts. Both the 74393 and 7449 are originally EP300 designs. Yet, in this example, they were combined to produce an EP600 design.

Clearly, the area files available in this library can be powerful, timesaving design tools. They allow you to utilize various TTL functions in Altera EPLD designs without wasting time designing these functions. Moreover, any number of area files can be integrated to create designs for any Altera EPLD.

The Altera Design Processor Utilization Report for the 74393 is shown in Figure 6. The information in the title block of the drawing file is listed at the top of the report. Next, a diagram of the pin assignments is shown. This diagram is followed by the utilization information which lists how the part's resources have

been used by the design. The **\*\*PART UTILIZATION\*\*** subsection then shows the percentage of used pins, macrocells, and product terms. Refer to the Utilization Report section of the A+PLUS Reference Manual for a more detailed description of the utilization report.

An alternative to entering a design with schematic capture is to enter it with Boolean expressions exclusively. One convenient method to enter Boolean expressions is by creating your own Altera Design File (ADF) using a text editor. Figure 7 shows the ADF for the 74393 design. By following the guidelines stated in the Boolean Equation Entry section of the A+PLUS User's Guide, you can create an ADF with a text editor such as WordStar. You then input the ADF into the Altera Design Processor (ADP) where it is further processed into a JEDEC Standard File. If you are accustomed to designing with Boolean expressions, this method of design entry is highly recommended.

**Fig. 7**

```

Boolean design entry by: BOB VENABLE
ALTERA CORPORATION
MAR. 4, 1985
1.00
A
EP300
74393 MODULO-16 COUNTER
PART:EP300
INPUTS:
RESETp, CLOCKp
OUTPUTS:
Q0p, Q1p, Q2p, Q3p
NETWORK:
CLOCK      =      INP(CLOCKp) % Clock Input %
RESET      =      INP(RESETp) %Input Signal %
Q0p,Q0f    =      RORF(Q0      ,CLOCK, RESET, GND , VCC ) % Registered %
Q1p,Q1f    =      RORF(Q1      ,CLOCK, RESET, GND , VCC ) % Registered %
Q2p,Q2f    =      RORF(Q2      ,CLOCK, RESET, GND , VCC ) % Registered %
Q3p,Q3f    =      RORF(Q3      ,CLOCK, RESET, GND , VCC ) % Registered %

EQUATIONS:
RESET = RESET ;

Q3 = Q3f*Q2f'
    + Q3f*Q1f'
    + Q3f*Q0f'
    + Q3f'*Q2f*Q1f*Q0f ;

Q2 = Q2f*Q0f'
    + Q1f'*Q2f
    + Q1f*Q2f'*Q0f ;

Q1 = Q1f'*Q0f
    + Q1f*Q0f' ;

Q0 = Q0f' ;

END$

```



The following designs are examples taken from the Altera TTL library which contains over 100 logic designs. These are intended as practical examples of small circuits implemented in EPLDs. The designs in the library contain many techniques that may be useful when developing more complex circuits.

The complete library contains schematic drawings, and their Boolean algebra equivalents in the Altera Design File format (ADF). These designs and all those published in application notes and application briefs are available on-line via an auto-answer modem link at Altera. Subscribers to the A+PLUS maintenance package have free access to this design database.

Boolean design  
 ALTERA CORPORATION  
 FEB. 14, 1985

1.00

A

EP1200

7442A 1-OF-10 DECODER

PART:EP1200

INPUTS:

A3p, A2p, A1p, A0p

OUTPUTS:

O9Np, O8Np, O7Np, O6Np, O5Np, O4Np, O3Np, O2Np,

O1Np, O0Np

NETWORK:

A3 = INP (A3p) % Input Signal %

A2 = INP (A2p) % Input Signal %

A1 = INP (A1p) % Input Signal %

A0 = INP (A0p) % Input Signal %

O9Np= CONF(09N , VCC)

O8Np= CONF(08N , VCC)

O7Np= CONF(07N , VCC)

O6Np= CONF(06N , VCC)

O5Np= CONF(05N , VCC)

O4Np= CONF(04N , VCC)

O3Np= CONF(03N , VCC)

O2Np= CONF(02N , VCC)

O1Np= CONF(01N , VCC)

O0Np= CONF(00N , VCC)

EQUATIONS:

O0N = /( A3'\*A2'\*A1'\*A0' ) ; % Inverted Equation %

O1N = /( A0\*A3'\*A2'\*A1' ) ; % Inverted Equation %

O2N = /( A1\*A0'\*A2'\*A3' ) ; % Inverted Equation %

O3N = /( A0\*A2'\*A1'\*A3' ) ; % Inverted Equation %

O4N = /( A0'\*A2\*A1'\*A3' ) ; % Inverted Equation %

O5N = /( A2\*A0'\*A1'\*A3' ) ; % Inverted Equation %

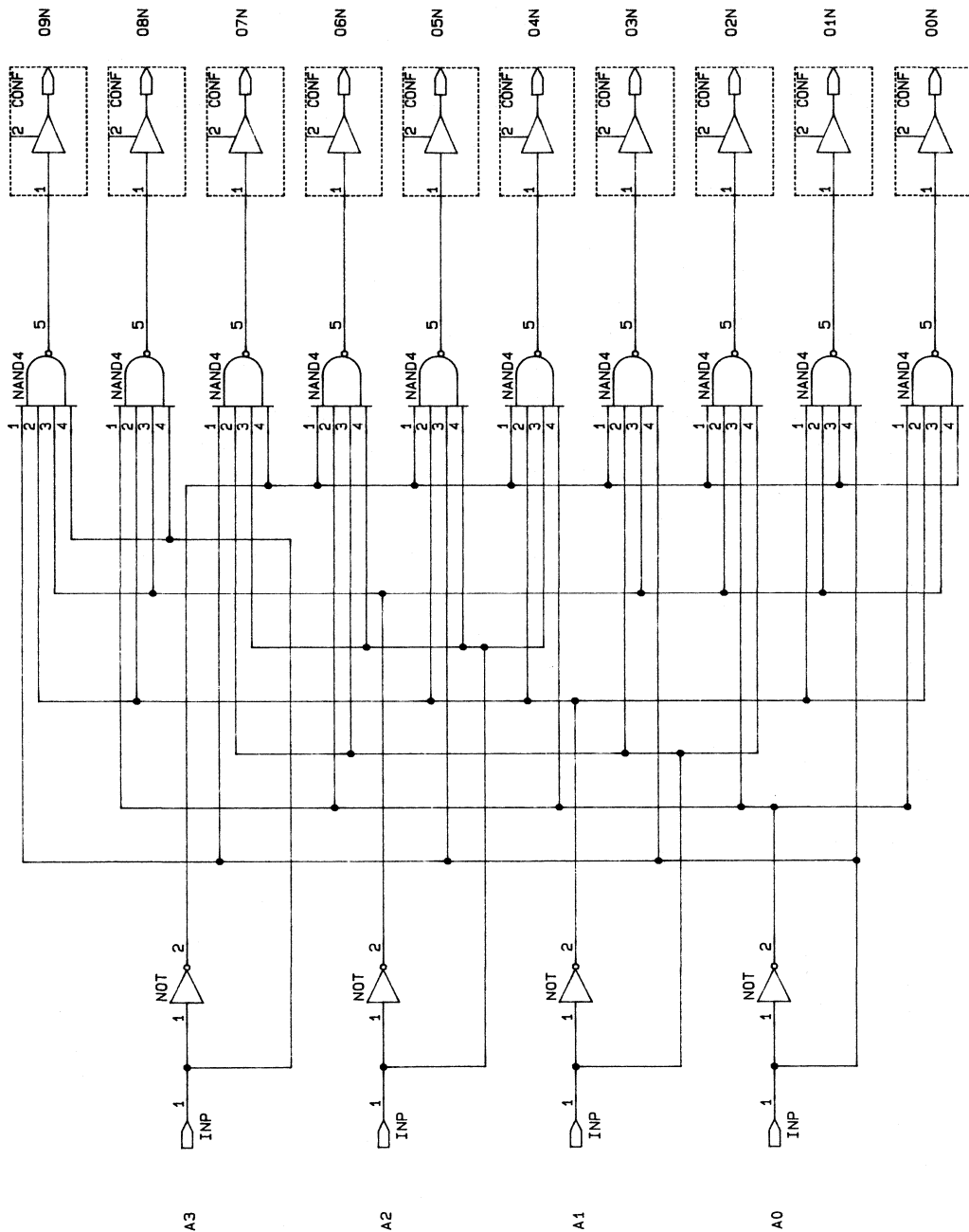
O6N = /( A2\*A1\*A0'\*A3' ) ; % Inverted Equation %

O7N = /( A3'\*A2\*A1\*A0 ) ; % Inverted Equation %

O8N = /( A3\*A2'\*A1'\*A0' ) ; % Inverted Equation %

O9N = /( A3\*A2'\*A1'\*A0 ) ; % Inverted Equation %

END\$



Boolean design  
 ALTERA CORPORATION  
 JAN. 8, 1985

1.00

A

EP300

7449 BCD TO 7-SEG

PART:EP300

INPUTS:

A0p, A1p, A2p, A2p, BLNKINGNp

OUTPUTS:

Ap, Bp, Cp, Dp, Ep, Fp, Gp

NETWORK:

A0 = INP (A0p) % Input Signal %  
 A1 = INP (A1p) % Input Signal %  
 A2 = INP (A2p) % Input Signal %  
 A3 = INP (A3p) % Input Signal %  
 BLNKINGN = INP (BLNKINGNp) % Input Signal %  
 Ap= CONF (A , VCC)  
 Bp= CONF (B , VCC)  
 Cp= CONF (C , VCC)  
 Dp= CONF (D , VCC)  
 Ep= CONF (E , VCC)  
 Fp= CONF (F , VCC)  
 Gp= CONF (G , VCC)

EQUATIONS:

G = A1'\*BLNKINGN\*A3  
 + A1'\*BLNKINGN\*A2  
 + A1'\*BLNKINGN\*A2'  
 + A1'\*BLNKINGN\*A0' ;

F = /( BLNKINGN'  
 + A1\*A2'  
 + A1\*A0  
 + A0\*A2'\*A3' ) ; % Inverted Equation %

E = BLNKINGN\*A0'\*A2'  
 + A1\*BLNKINGN\*A0' ;

D = /( BLNKINGN'  
 + A0'\*A2\*A1'  
 + A0\*A2'\*A1'  
 + A0\*A2\*A1 ) ; % Inverted Equation %

C = BLNKINGN\*A2'\*A1'  
 + BLNKINGN\*A2\*A3'  
 + A0\*BLNKINGN\*A2' ;

B = /( BLNKINGN'  
 + A3\*A1  
 + A1\*A2\*A0'  
 + A1'\*A2\*A0 ) ; % Inverted Equation %

A = /( BLNKINGN'  
 + A2\*A0'  
 + A3\*A1  
 + A3'\*A1'\*A2'\*A0 ) ; % Inverted Equation %

END\$





Boolean design  
 ALTERA CORPORATION  
 JAN. 24, 1985  
 1.00

A  
 EP300  
 7473 JK FLIP-FLOP  
 PART:EP300

INPUTS:

Kp, CLOCKp, Jp, CdNp

OUTPUTS:

Qp, QNp

NETWORK:

K = INP(Kp) % Input Signal %  
 CLOCK = INP(CLOCKp) % Input Signal %  
 J = INP(Jp) % Input Signal %  
 CdN = INP(CdNp) % Input Signal %  
 Qp, Qf = COCF(Q, VCC)  
 QNp, QNf = COCF(QN, VCC)  
 NODE9 = NOCF(NODE10)  
 NODE11 = NOCF(NODE12)

EQUATIONS:

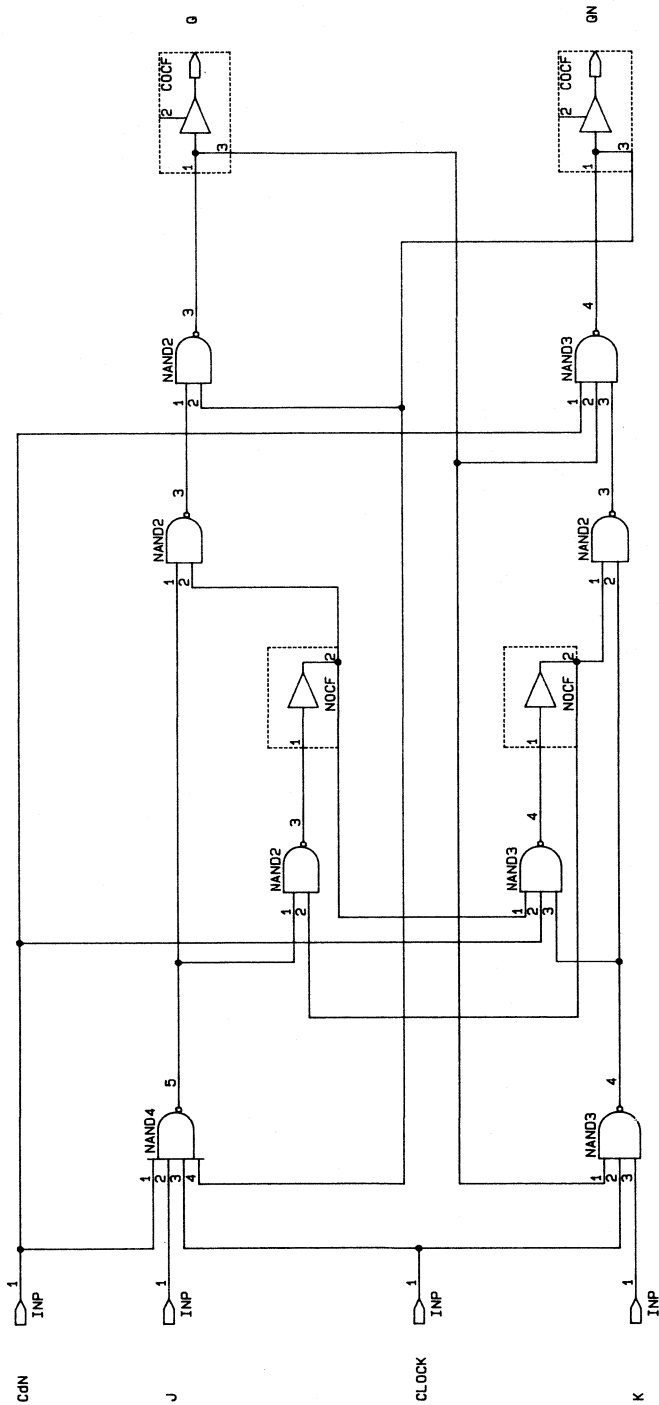
NODE12 = NODE9'  
 + QNf\*CLOCK\*J\*CdN ;

NODE10 = CdN'  
 + NODE11'  
 + K\*CLOCK\*Qf ;

QN = /( NODE9'\*Qf\*CdN  
 + Qf\*CdN\*K\*CLOCK ) ; % Inverted Equation %

Q = /( QNf\*NODE11'  
 + CLOCK\*J\*CdN\*QNf ) ; % Inverted Equation %

END\$



Boolean design  
 ALTERA CORPORATION  
 JAN. 14, 1985  
 1.00

A

EP300

7475 4 BIT BISTABLE LATCH

PART:EP300

INPUTS:

D1p, E12p, D2p, E34p, D3p, D4p

OUTPUTS:

Q1Np, Q1p, Q2Np, Q2p, Q3Np, Q3p, Q4Np, Q4p

NETWORK:

D1 = INP (D1p) % Input Signal %  
 E12 = INP (E12p) % Input Signal %  
 D2 = INP (D2p) % Input Signal %  
 E34 = INP (E34p) % Input Signal %  
 D3 = INP (D3p) % Input Signal %  
 D4 = INP (D4p) % Input Signal %  
 Q1Np, Q1Nf = COCF (Q1N , VCC )  
 Q1p, Q1f = COCF (Q1 , VCC )  
 Q2Np, Q2Nf = COCF (Q2N , VCC )  
 Q2p, Q2f = COCF (Q2 , VCC )  
 Q3Np, Q3Nf = COCF (Q3N , VCC )  
 Q3p, Q3f = COCF (Q3 , VCC )  
 Q4Np, Q4Nf = COCF (Q4N , VCC )  
 Q4p, Q4f = COCF (Q4 , VCC )

EQUATIONS:

$$Q4 = Q4Nf * E34' + D4 * Q4Nf ;$$

$$Q4N = Q4f * E34' + D4' * Q4f ;$$

$$Q3 = Q3Nf * E34' + D3 * Q3Nf ;$$

$$Q3N = Q3f * E34' + D3' * Q3f ;$$

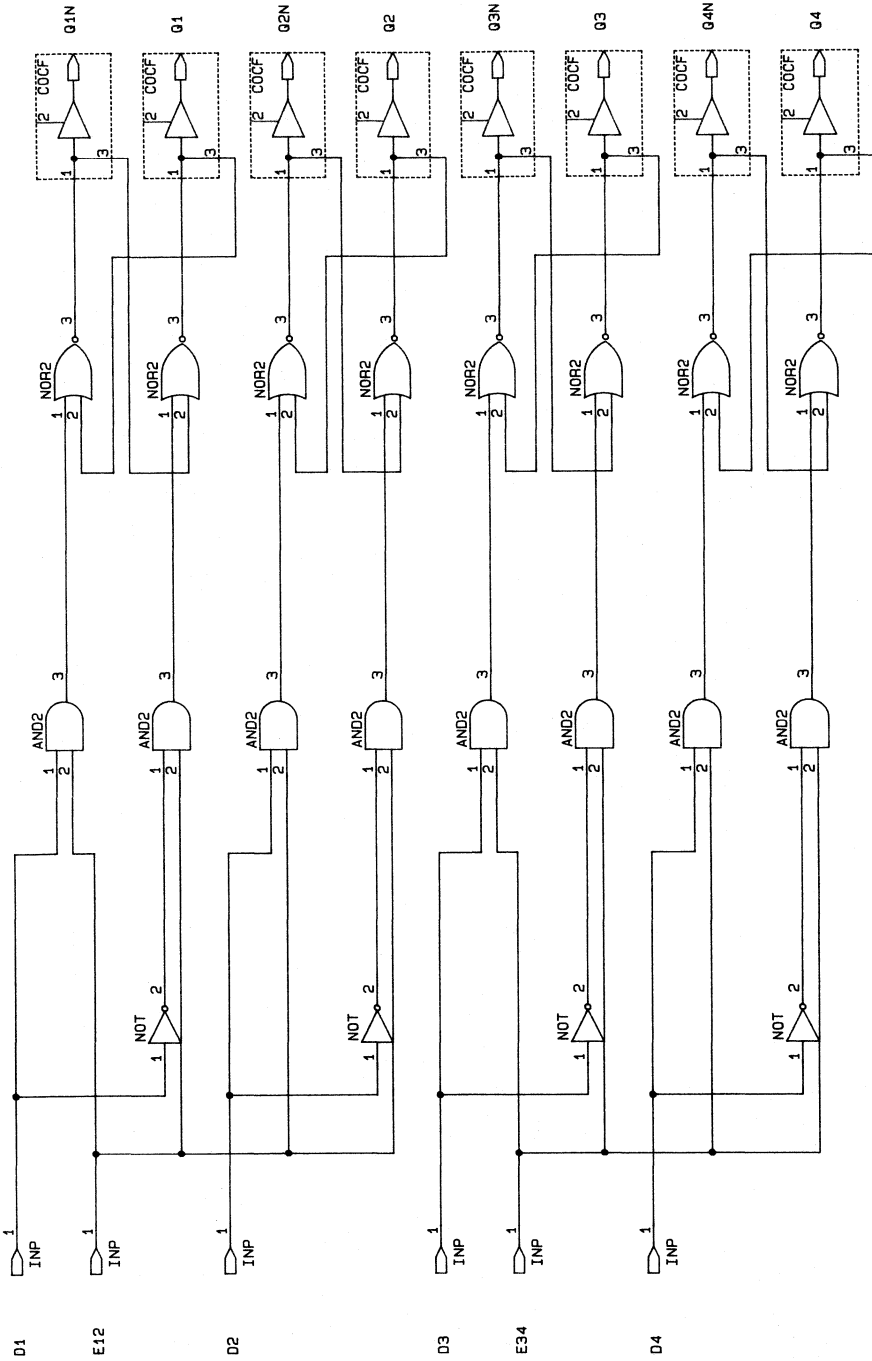
$$Q2 = Q2Nf * E12' + D2 * Q2Nf ;$$

$$Q2N = Q2f * E12' + D2' * Q2f ;$$

$$Q1 = Q1Nf * E12' + D1 * Q1Nf ;$$

$$Q1N = Q1f * E12' + D1' * Q1f ;$$

END\$



Boolean design  
ALTERA CORPORATION  
JAN. 25, 1985

1.00

A

EP300

7483 4-BIT FULL ADDER

PART:EP300

INPUTS:

A2p, B2p, B1p, A1p, C0p, A3p, B3p, A4p, B4p

OUTPUTS:

SUM1p, SUM2p, SUM3p, SUM4p, C4p

NETWORK:

A2 = INP(A2p) % Input Signal %

B2 = INP(B2p) % Input Signal %

B1 = INP(B1p) % Input Signal %

A1 = INP(A1p) % Input Signal %

C0 = INP(C0p) % Input Signal %

A3 = INP(A3p) % Input Signal %

B3 = INP(B3p) % Input Signal %

A4 = INP(A4p) % Input Signal %

B4 = INP(B4p) % Input Signal %

SUM1p= CONF(SUM1 , VCC)

SUM2p= CONF(SUM2 , VCC)

SUM3p= CONF(SUM3 , VCC)

SUM4p= CONF(SUM4 , VCC)

C4p= CONF(C4 , VCC)

NODE15= NOCF(NODE16)

NODE17= NOCF(NODE18)

NODE19= NOCF(NODE20)

EQUATIONS:

NODE20 = /( A3\*B3

+ NODE17\*B3

+ A3\*NODE17 ) ; % Inverted Equation %

NODE18 = /( A2\*B2

+ NODE15\*B2

+ A2\*NODE15 ) ; % Inverted Equation %

NODE 16 = /( A1\*B1

+ C0\*B1

+ A1\*C0 ) ; % Inverted Equation %

C4 = /( A4\*B4

+ NODE19\*B4

+ A4\*NODE19 ) ; % Inverted Equation %

SUM4 = /( A4\*B4\*NODE19

+ A4\*B4\*NODE19

+ A4\*B4\*NODE19

+ A4\*B4\*NODE19 ) ; % Inverted Equation %

SUM3 = NODE19\*B3

+ NODE19\*A3

+ NODE19\*NODE17

+ NODE17\*A3\*B3 ;

SUM2 = /( NODE17\*B2

+ NODE17\*A2

+ NODE17\*NODE15

+ NODE15\*A2\*B2 ) ; % Inverted Equation %

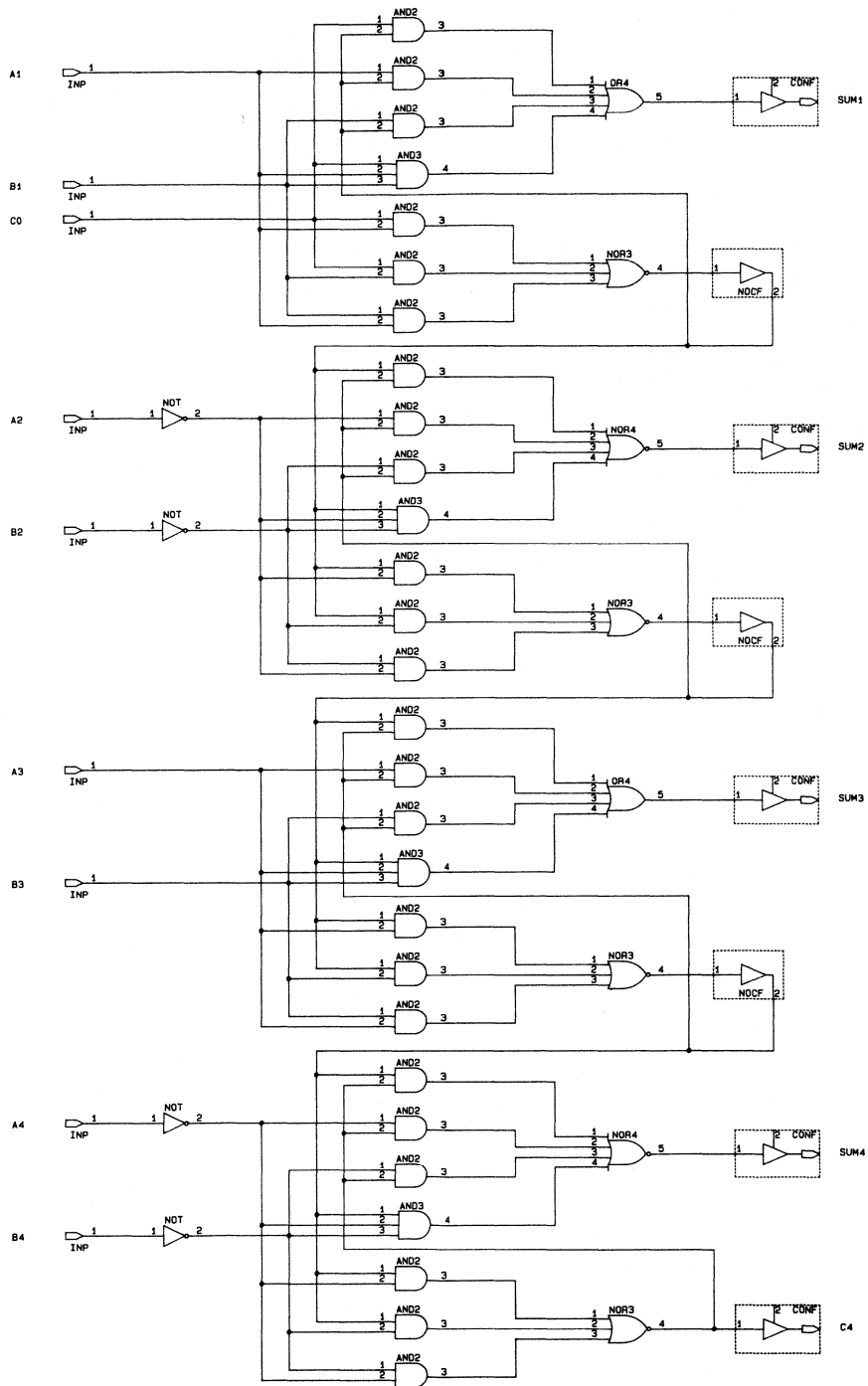
SUM1 = NODE15\*B1

+ NODE15\*A1

+ NODE15\*C0

+ C0\*A1\*B1 ;

END\$



Boolean design  
ALTERA CORPORATION  
MAR. 19, 1985

1.00

A

EP300

7485 4-BIT MAG. COMP.

PART:EP300

INPUTS:

A3pp, B3pp, A2pp, B2pp, IAItBpp, IAeqBpp, IAgtBpp, A1pp,  
B1pp, A0pp, B0pp

OUTPUTS:

OAGtBpp, OAEqBpp, OA1tBpp

NETWORK:

A3= INP(A3pp) % Input Signal %  
B3= INP(B3pp) % Input Signal %  
A2= INP(A2pp) % Input Signal %  
B2= INP(B2pp) % Input Signal %  
IAItB= INP(IAItBpp) % Input Signal %  
IAeqB= INP(IAeqBpp) % Input Signal %  
IAgtB= INP(IAgtBpp) % Input Signal %  
A1= INP(A1pp) % Input Signal %  
B1= INP(B1pp) % Input Signal %  
A0= INP(A0pp) % Input Signal %  
B0= INP(B0pp) % Input Signal %  
OAGtBpp= CONF(OAGtB, VCC )  
OAEqBpp= CONF(OAEqB, VCC )  
OAItBpp= CONF(OAItB, VCC )  
NODE15= NOCF(NODE16)  
NODE17= NOCF(NODE18)  
NODE19= NOCF(NODE20)  
NODE21= NOCF(NODE22)

EQUATIONS:

NODE22 = A0'\*B0'  
+ A0\*B0 ;

NODE20 = A1'\*B1'  
+ A1\*B1 ;

NODE18 = A2'\*B2'  
+ A2\*B2 ;

NODE16 = A3'\*B3'  
+ A3\*B3 ;

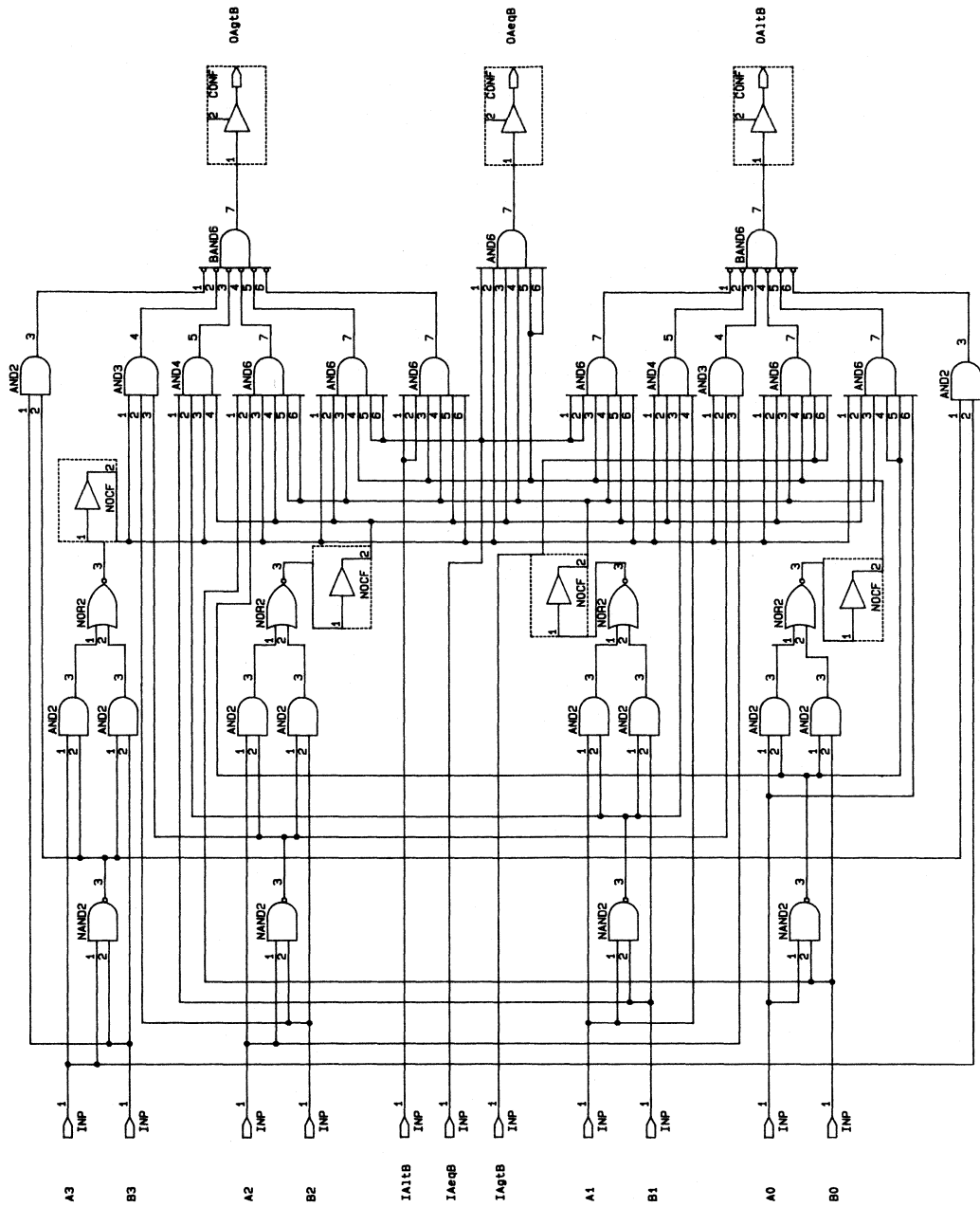
OAItB = /( A3\*B3'  
+ A2\*B2'\*NODE15  
+ NODE15\*A1\*B1'\*NODE17  
+ NODE15\*NODE17\*A0\*B0'\*NODE19  
+ NODE15\*NODE17\*NODE19\*NODE21\*IAeqB  
+ NODE15\*NODE17\*NODE19\*IAgtB\*NODE21 ) ; % Inverted Equation %

OAEqB = IAeqB\*NODE15\*NODE17\*NODE19\*NODE21 ;

OAGtB = /( B3\*A3'  
+ NODE15\*B2\*A2'  
+ NODE15\*B1\*NODE17\*A1'  
+ NODE15\*NODE17\*B0\*NODE19\*A0'  
+ NODE15\*NODE17\*NODE19\*NODE21\*IAeqB  
+ NODE15\*NODE17\*NODE19\*IAItB\*NODE21 ) ; % Inverted Equation %

END\$





Boolean design  
 ALTERA CORPORATION  
 MAR. 12, 1985  
 1.00

A

EP300

7493DIV8 COUNTER

PART:EP300

INPUTS:

RESET1p, RESET2p, CLOCKp

OUTPUTS:

Q2p, Q1p, Q0p

NETWORK:

CLOCK = INP(CLOCKp) % Clock Input %  
 RESET1 = INP(RESET1p) % Input Signal %  
 RESET2 = INP(RESET2p) % Input Signal %  
 Q2p,Q2f = RORF(Q2 ,CLOCK,NODE6, GND , VCC ) % Registered %  
 Q1p,Q1f = RORF(Q1 ,CLOCK,NODE6, GND , VCC ) % Registered %  
 Q0p,Q0f = RORF(Q0 ,CLOCK,NODE6, GND , VCC ) % Registered %

EQUATIONS:

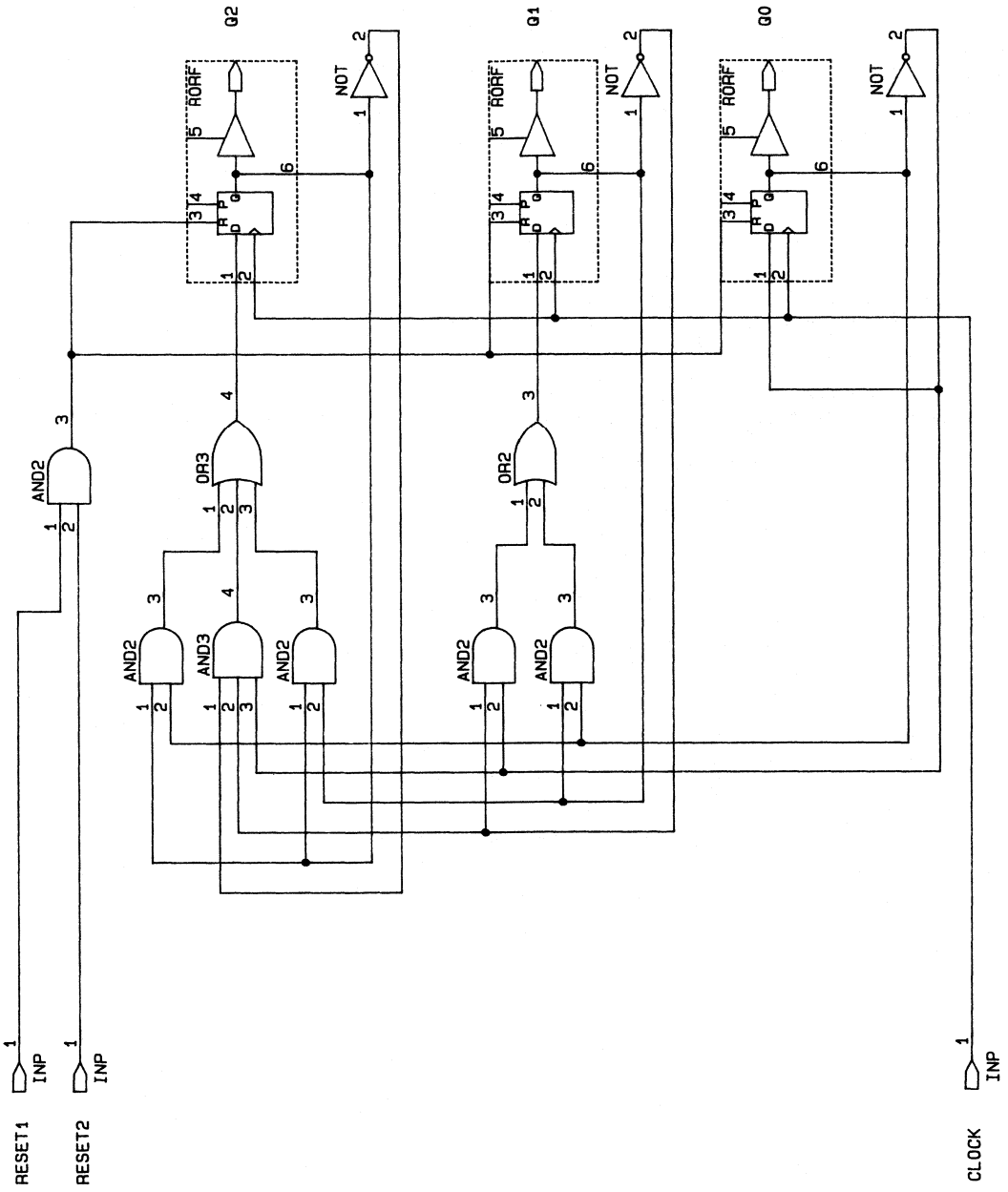
Q0 = Q0f ;

Q1 = Q0f\*Q1f  
 + Q0f\*Q1f ;

NODE6 = RESET2\*RESET1 ;

Q2 = Q2f\*Q1f  
 + Q0f\*Q2f  
 + Q0f\*Q2f\*Q1f ;

END\$



Boolean design  
 ALTERA CORPORATION  
 FEB. 7, 1985  
 1.00

A

EP300

7494 4-BIT SHIFT REGIST.

PART:EP300

INPUTS:

CLEARp,DSp,P1Ap,P2Ap,P1Bp,P2Bp,P1Cp,P2Cp,  
 P1Dp,P2Dp,PL1p,PL2p,CLOCKp

OUTPUTS:

QDp

NETWORK:

CLOCK = INP(CLOCKp) % Clock Input %  
 CLEAR = INP(CLEARp) % Input Signal %  
 DS = INP(DSp) % Input Signal %  
 P1A = INP(P1Ap) % Input Signal %  
 P2A = INP(P2Ap) % Input Signal %  
 P1B = INP(P1Bp) % Input Signal %  
 P2B = INP(P2Bp) % Input Signal %  
 P1C = INP(P1Cp) % Input Signal %  
 P2C = INP(P2Cp) % Input Signal %  
 P1D = INP(P1Dp) % Input Signal %  
 P2D = INP(P2Dp) % Input Signal %  
 PL1 = INP(PL1p) % Input Signal %  
 PL2 = INP(PL2p) % Input Signal %  
 QDp= RONF(QD ,CLOCK,CLEAR, GND , VCC ) % Registered %  
 NODE15= NORF(NODE16,CLOCK,CLEAR, GND ) % Registered %  
 NODE17= NORF(NODE18,CLOCK,CLEAR, GND ) % Registered %  
 NODE19= NORF(NODE20,CLOCK,CLEAR, GND ) % Registered %

EQUATIONS:

CLEAR = CLEAR ;

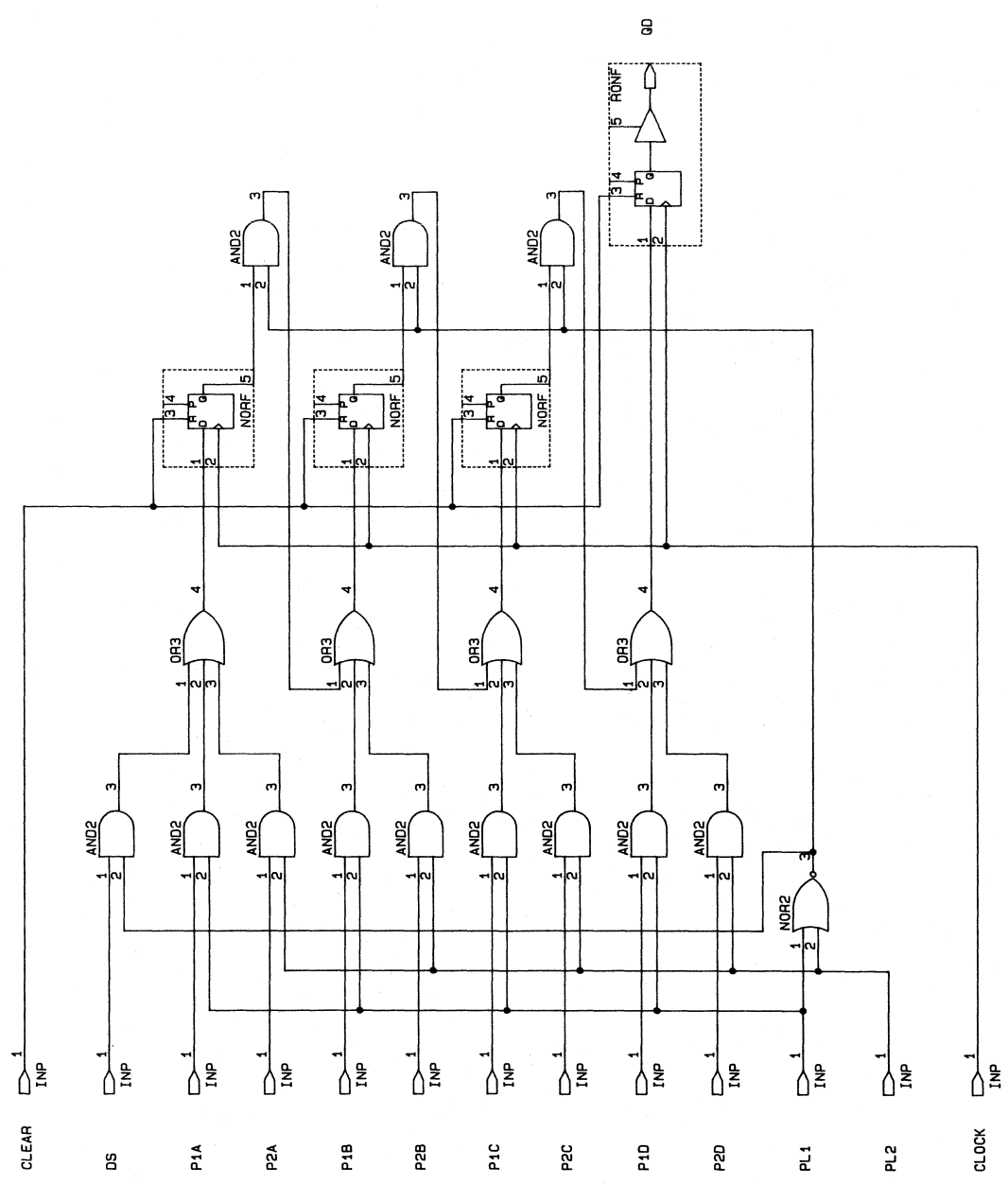
NODE20 = PL2\*P2C  
 + PL1\*P1C  
 + PL2'\*PL1'\*NODE17 ;

NODE18 = PL2\*P2B  
 + PL1\*P1B  
 + PL2'\*PL1'\*NODE15 ;

NODE16 = PL2\*P2A  
 + PL1\*P1A  
 + PL2'\*PL1'\*DS ;

QD = PL2\*P2D  
 + PL1\*P1D  
 + PL2'\*PL1'\*NODE19 ;

END\$



Boolean design  
 ALTERA CORPORATION  
 JAN. 18, 1985  
 1.00

A

EP300

74151 8-INPUT MUX.

PART:EP300

INPUTS:

I6p,I5p,I4p,I3p,I2p,I1p,I0p,S2p,

S1p,S0p,ENp,I7p

OUTPUTS:

ZNp,Zp

NETWORK:

I6 = INP(I6p) % Input Signal %  
 I5 = INP(I5p) % Input Signal %  
 I4 = INP(I4p) % Input Signal %  
 I3 = INP(I3p) % Input Signal %  
 I2 = INP(I2p) % Input Signal %  
 I1 = INP(I1p) % Input Signal %  
 I0 = INP(I0p) % Input Signal %  
 S2 = INP(S2p) % Input Signal %  
 S1 = INP(S1p) % Input Signal %  
 S0 = INP(S0p) % Input Signal %  
 EN = INP(ENp) % Input Signal %  
 I7 = INP(I7p) % Input Signal %  
 ZNp= CONF(ZN , VCC )  
 Zp= CONF(Z , VCC )

EQUATIONS:

Z = S0'S1'S2'EN'I0  
 + S0'S1'S2'EN'I1  
 + S0'S1'S2'EN'I2  
 + S0'S1'S2'EN'I4  
 + S0'S1'S2'EN'I3  
 + S0'S1'S2'EN'I5  
 + S0'S1'S2'EN'I6  
 + S0'S1'S2'EN'I7 ;

ZN = /( S0'S1'S2'EN'I0  
 + S0'S1'S2'EN'I1  
 + S0'S1'S2'EN'I2  
 + S0'S1'S2'EN'I4  
 + S0'S1'S2'EN'I3  
 + S0'S1'S2'EN'I5  
 + S0'S1'S2'EN'I6  
 + S0'S1'S2'EN'I7 ) ; % Inverted Equation %

END\$



Boolean design  
ALTERA CORPORATION  
FEB. 8, 1985

1.00

A

EP300

74160 BCD DECADE COUNTER

PART:EP300

INPUTS:

MRNp,P0p,P1p,P2p,P3p,PENp,CEPp,CETp,

CLOCKp

OUTPUTS:

Q0p,Q1p,Q2p,Q3p,TCp

NETWORK:

CLOCK = INP(CLOCKp) % Clock Input %  
MRN = INP(MRNp) % Input Signal %  
P0 = INP(P0p) % Input Signal %  
P1 = INP(P1p) % Input Signal %  
P2 = INP(P2p) % Input Signal %  
P3 = INP(P3p) % Input Signal %  
PEN = INP(PENp) % Input Signal %  
CEP = INP(CEPp) % Input Signal %  
CET = INP(CETp) % Input Signal %  
Q0p,Q0f = RORF(Q0 ,CLOCK,NODE12, GND , VCC ) % Registered %  
Q1p,Q1f = RORF(Q1 ,CLOCK,NODE12, GND , VCC ) % Registered %  
Q2p,Q2f = RORF(Q2 ,CLOCK,NODE12, GND , VCC ) % Registered %  
Q3p,Q3f = RORF(Q3 ,CLOCK,NODE12, GND , VCC ) % Registered %  
TCp= CONF(TC , VCC)

EQUATIONS:

TC = CET\*Q3f\*Q0f\*Q2f\*Q1f ;

Q3 = PEN\*P3

+ CET'\*PEN\*Q3f  
+ CEP'\*PEN\*Q3f  
+ PEN\*Q0f\*Q3f  
+ PEN\*Q1f\*Q2f\*Q3f  
+ CET\*CEP\*PEN\*Q0f\*Q1f\*Q2f ;

Q2 = PEN\*P2

+ CET'\*PEN\*Q2f  
+ CEP'\*PEN\*Q2f  
+ PEN\*Q2f\*Q0f  
+ PEN\*Q2f\*Q1f  
+ CET\*CEP\*PEN\*Q2f\*Q0f\*Q1f ;

Q1 = PEN\*P1

+ CET'\*PEN\*Q1f  
+ CEP'\*PEN\*Q1f  
+ PEN\*Q1f\*Q0f  
+ CET\*CEP\*PEN\*Q1f\*Q3f\*Q0f ;

NODE12 = MRN' ;

Q0 = PEN\*P0

+ CET'\*PEN\*Q0f  
+ CEP'\*PEN\*Q0f  
+ CET\*CEP\*PEN\*Q0f' ;

END\$





Boolean design  
 ALTERA CORPORATION  
 FEB. 11, 1985  
 1.00

A

EP300

74161 BINARY COUNTER

PART:EP300

INPUTS:

MRNp,P0p,P1p,P2p,P3p,PENp,CEPp,CETp,

CLOCKp

OUTPUTS:

Q0p,Q1p,Q2p,Q3p,TCp

NETWORK:

CLOCK = INP(CLOCKp) % Clock Input %  
 MRN = INP(MRNp) % Input Signal %  
 P0 = INP(P0p) % Input Signal %  
 P1 = INP(P1p) % Input Signal %  
 P2 = INP(P2p) % Input Signal %  
 P3 = INP(P3p) % Input Signal %  
 PEN = INP(PENp) % Input Signal %  
 CEP = INP(CEPp) % Input Signal %  
 CET = INP(CETp) % Input Signal %  
 Q0p,Q0f = RORF(Q0 ,CLOCK,NODE12, GND , VCC ) % Registered %  
 Q1p,Q1f = RORF(Q1 ,CLOCK,NODE12, GND , VCC ) % Registered %  
 Q2p,Q2f = RORF(Q2 ,CLOCK,NODE12, GND , VCC ) % Registered %  
 Q3p,Q3f = RORF(Q3 ,CLOCK,NODE12, GND , VCC ) % Registered %  
 TCp= CONF(TC , VCC)

EQUATIONS:

TC = CET\*Q3f\*Q2f\*Q1f\*Q0f ;

Q3 = PEN'\*P3

+ CET'\*PEN\*Q3f  
 + CEP'\*PEN\*Q3f  
 + PEN\*Q3f\*Q0f  
 + PEN\*Q3f\*Q1f  
 + PEN\*Q3f\*Q2f  
 + CET\*CEP\*PEN\*Q3f\*Q0f\*Q1f\*Q2f ;

Q2 = PEN'\*P2

+ CET'\*PEN\*Q2f  
 + CEP'\*PEN\*Q2f  
 + PEN\*Q2f\*Q0f  
 + PEN\*Q2f\*Q1f  
 + CET\*CEP\*PEN\*Q2f\*Q0f\*Q1f ;

Q1 = PEN'\*P1

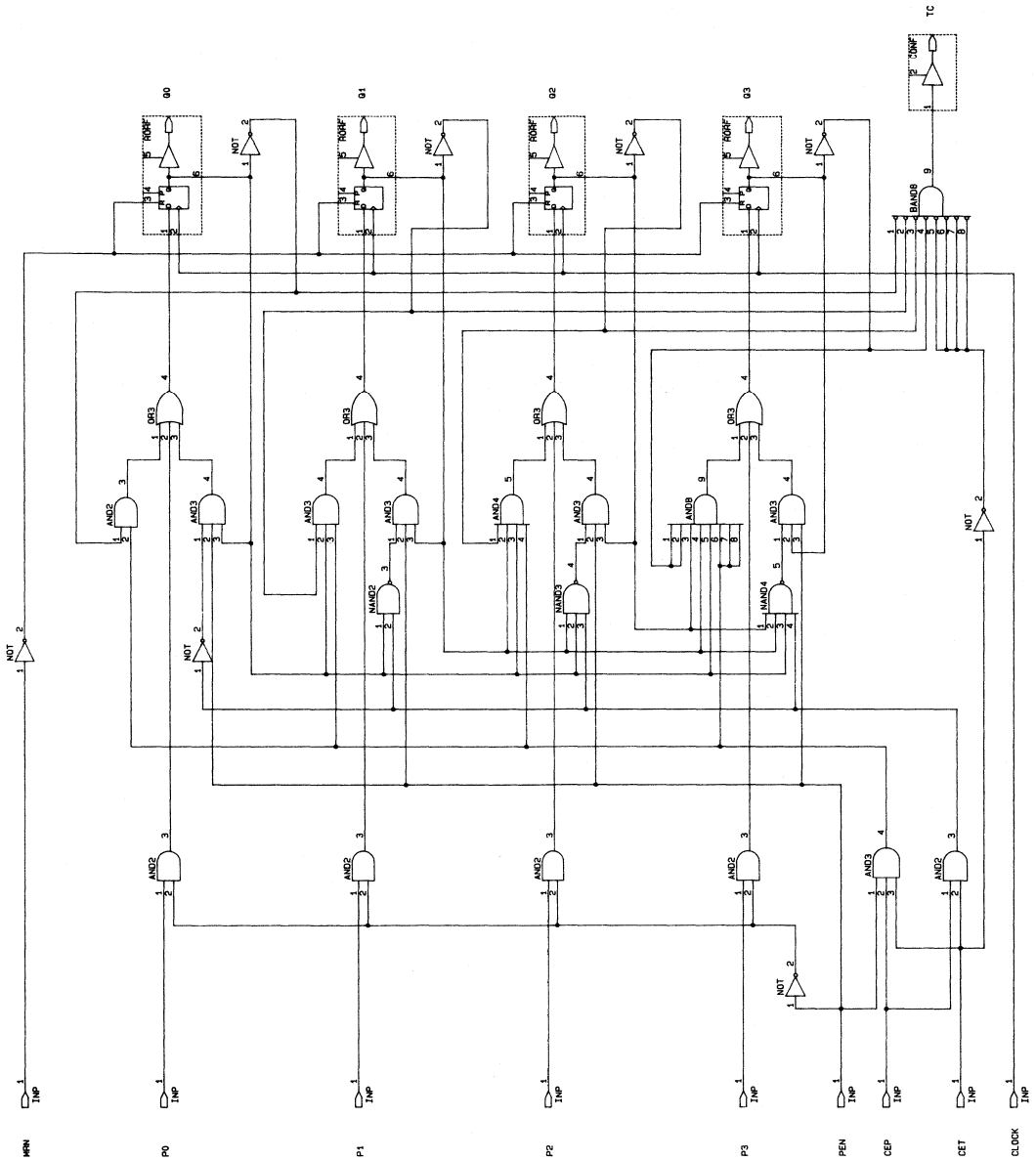
+ CET'\*PEN\*Q1f  
 + CEP'\*PEN\*Q1f  
 + PEN\*Q1f\*Q0f  
 + CET\*CEP\*PEN\*Q1f\*Q0f ;

NODE12 = MRN' ;

Q0 = PEN'\*P0

+ CET'\*PEN\*Q0f  
 + CEP'\*PEN\*Q0f  
 + CET\*CEP\*PEN\*Q0f ;

END\$



Boolean design  
 ALTERA CORPORATION  
 FEB. 11, 1985  
 1.00

A

EP300

74164 SER/PAR SHIFT REG

PART:EP300

INPUTS:

MRNp,Ap,Bp,CLOCKp

OUTPUTS:

Q0p,Q1p,Q2p,Q3p,Q4p,Q5p,Q6p,Q7p

NETWORK:

CLOCK = INP(CLOCKp) % Clock Input %  
 MRN = INP(MRNp) % Input Signal %  
 A = INP(Ap) % Input Signal %  
 B = INP(Bp) % Input Signal %  
 Q0p,Q0f = RORF(Q0 ,CLOCK,NODE7, GND , VCC ) % Registered %  
 Q1p,Q1f = RORF(Q1 ,CLOCK,NODE7, GND , VCC ) % Registered %  
 Q2p,Q2f = RORF(Q2 ,CLOCK,NODE7, GND , VCC ) % Registered %  
 Q3p,Q3f = RORF(Q3 ,CLOCK,NODE7, GND , VCC ) % Registered %  
 Q4p,Q4f = RORF(Q4 ,CLOCK,NODE7, GND , VCC ) % Registered %  
 Q5p,Q5f = RORF(Q5 ,CLOCK,NODE7, GND , VCC ) % Registered %  
 Q6p,Q6f = RORF(Q6 ,CLOCK,NODE7, GND , VCC ) % Registered %  
 Q7p= RONF(Q7 ,CLOCK,NODE7, GND , VCC ) % Registered %

EQUATIONS:

Q6 = Q6f ;

Q5 = Q5f ;

Q4 = Q4f ;

Q3 = Q3f ;

Q2 = Q2f ;

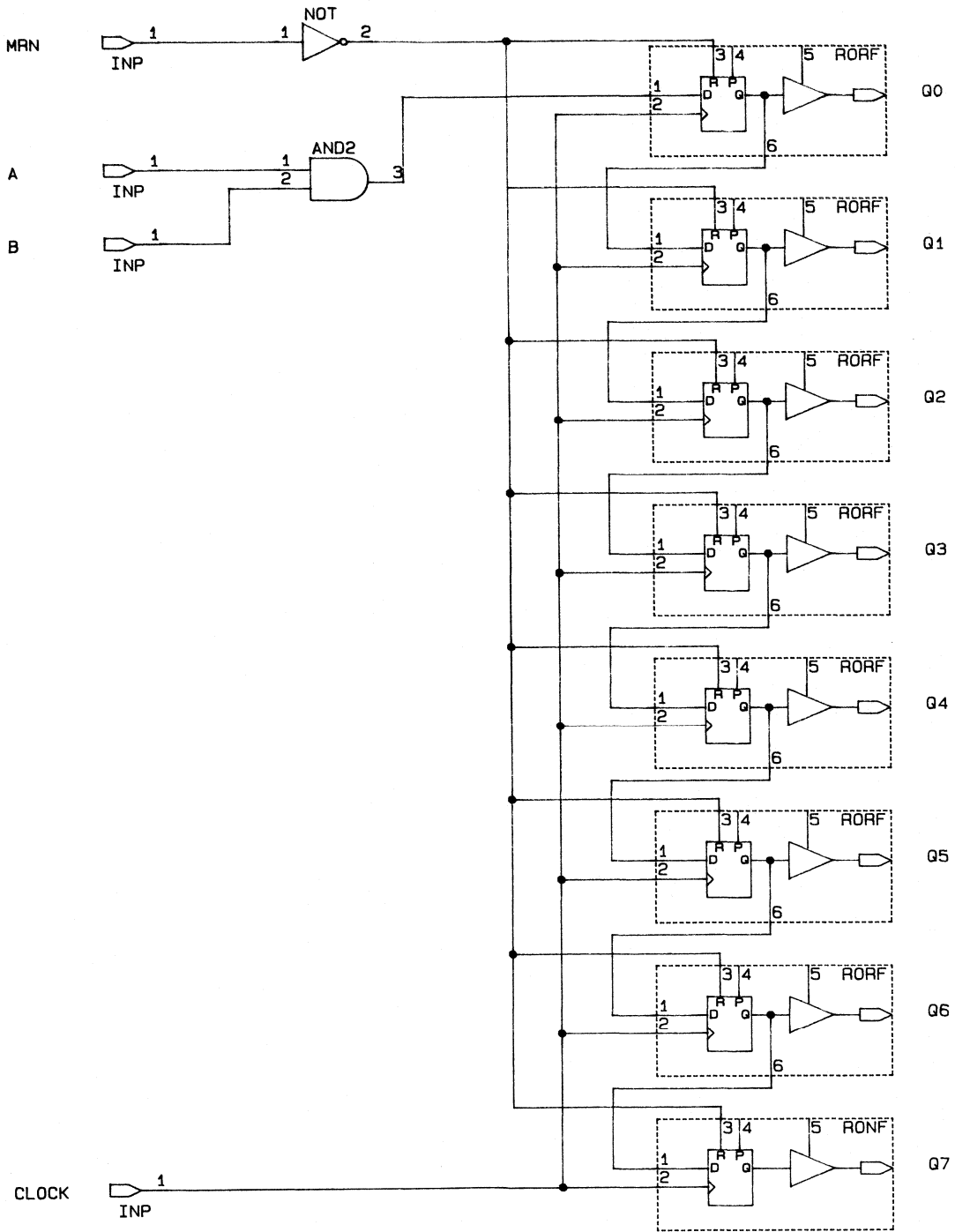
Q1 = Q1f ;

Q0 = Q0f ;

NODE7 = MRN' ;

Q0 = B\*A ;

END\$



Boolean design  
ALTERA CORPORATION  
FEB. 12, 1985

1.00

A

EP300

74168 U/DN BCD DEC COUNT

PART:EP300

INPUTS:

CEPNp,CETNp,UPDOWNp,PENp,P3p,P2p,P1p,P0p,

CLOCKp

OUTPUTS:

TCNp,Q3p,Q2p,Q1p,Q0p

NETWORK:

CLOCK = INP(CLOCKp) % Clock Input %  
 CEPN = INP(CEPNp) % Input Signal %  
 CETN = INP(CETNp) % Input Signal %  
 UPDOWN = INP(UPDOWNp) % Input Signal %  
 PEN = INP(PENp) % Input Signal %  
 P3 = INP(P3p) % Input Signal %  
 P2 = INP(P2p) % Input Signal %  
 P1 = INP(P1p) % Input Signal %  
 P0 = INP(P0p) % Input Signal %  
 TCNp = CONF(TCN , VCC )  
 Q3p,Q3f = RORF(Q3 ,CLOCK, GND, GND , VCC ) % Registered %  
 Q2p,Q2f = RORF(Q2 ,CLOCK, GND, GND , VCC ) % Registered %  
 Q1p,Q1f = RORF(Q1 ,CLOCK, GND, GND , VCC ) % Registered %  
 Q0p,Q0f = RORF(Q0 ,CLOCK, GND, GND , VCC ) % Registered %

EQUATIONS:

Q0 = PEN'\*P0

+ CETN\*Q0f\*PEN  
 + CEPN\*Q0f\*PEN  
 + CETN'\*CEPN'\*Q0f'\*PEN ;

Q1 = PEN'\*P1

+ Q1f\*CEPN\*PEN  
 + Q1f\*CETN\*PEN  
 + Q1f\*UPDOWN'\*PEN\*Q0f  
 + Q1f\*UPDOWN\*PEN\*Q0f'  
 + Q1f\*CETN'\*CEPN'\*Q3f\*UPDOWN'\*PEN\*Q0f'  
 + Q1f\*CETN'\*CEPN'\*UPDOWN'\*PEN\*Q0f'\*Q2f  
 + Q1f\*CETN'\*CEPN'\*Q3f\*UPDOWN\*PEN\*Q0f ;

Q2 = PEN'\*P2

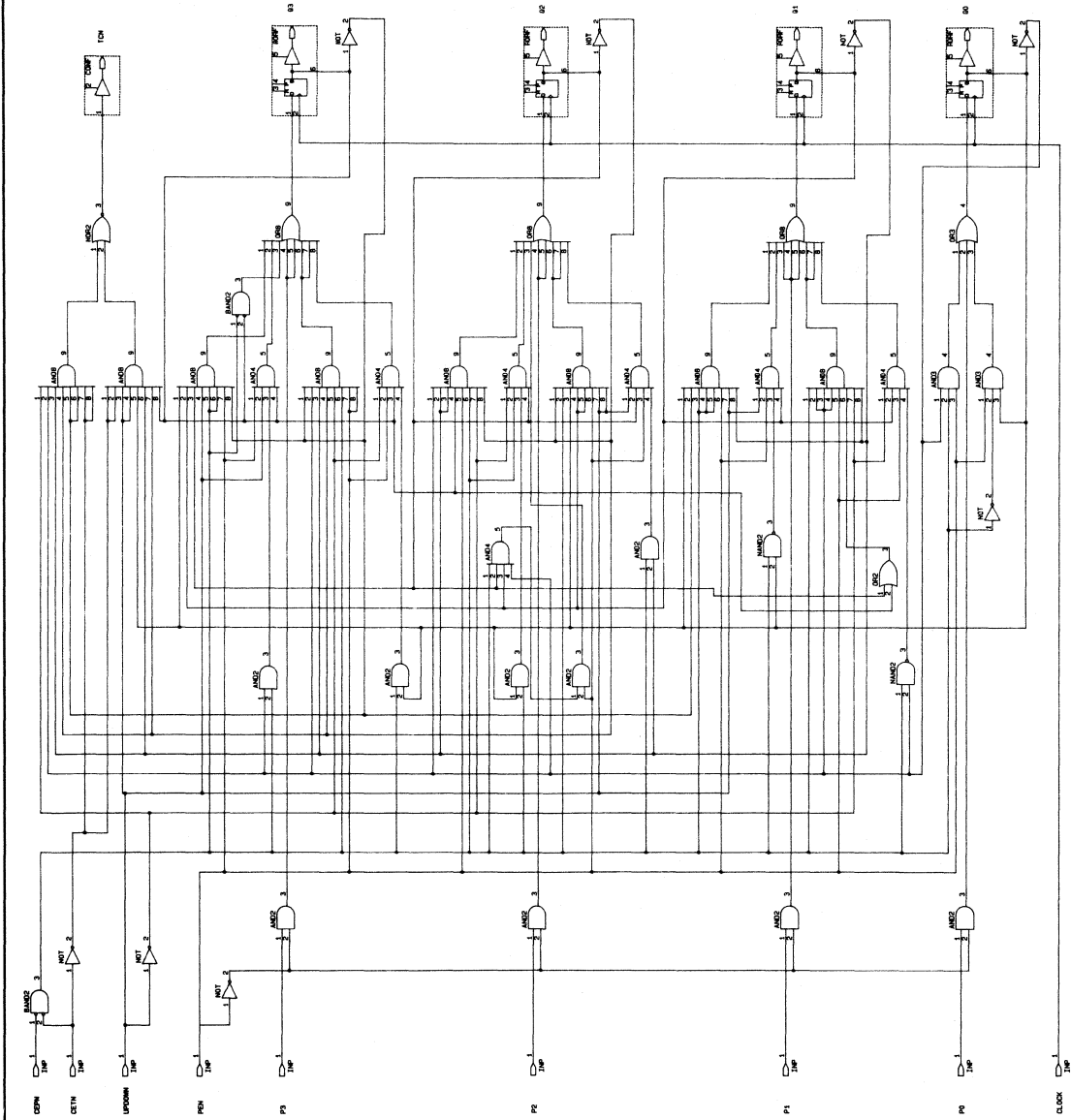
+ Q2f\*UPDOWN'\*CETN'\*CEPN'\*Q0f  
 + Q2f\*UPDOWN'\*CETN'\*CEPN'\*Q1f\*PEN  
 + Q2f\*UPDOWN\*CETN'\*CEPN'\*Q0f\*PEN  
 + Q2f\*UPDOWN\*CETN'\*CEPN'\*Q1f\*PEN  
 + Q2f\*UPDOWN\*CEPN'\*CETN'\*Q1f\*Q0f\*PEN  
 + Q2f\*UPDOWN'\*CETN'\*CEPN'\*Q1f'\*Q0f'\*PEN\*Q3f ;

Q3 = PEN'\*P3

+ Q3f\*CETN  
 + Q3f\*CEPN  
 + Q3f\*PEN\*UPDOWN'\*Q2f\*Q1f\*Q0f'  
 + Q3f\*CETN'\*CEPN'\*PEN\*UPDOWN'\*Q0f  
 + Q3f\*CETN'\*CEPN'\*PEN\*UPDOWN\*Q0f'  
 + Q3f\*PEN\*UPDOWN\*Q2f\*Q1f\*Q0f ;

TCN = /( CETN'\*Q3f'\*Q2f'\*Q1f'\*Q0f'\*UPDOWN'  
 + CETN'\*Q3f'\*Q2f'\*Q1f'\*Q0f'\*UPDOWN ) ; % Inverted Equation %

END\$



Boolean design  
 ALTERA CORPORATION  
 MAR. 15, 1985

1.00

A

EP300

74180 8-BIT PARITY GEN.

PART:EP300

INPUTS:

I0p, I1p, I2p, I3p, I4p, I5p, I6p, I7p,

ODDINp, EVENINp

OUTPUTS:

SUMEVENp, SUMODDp

NETWORK:

I0 = INP(I0p) % Input Signal %  
 I1 = INP(I1p) % Input Signal %  
 I2 = INP(I2p) % Input Signal %  
 I3 = INP(I3p) % Input Signal %  
 I4 = INP(I4p) % Input Signal %  
 I5 = INP(I5p) % Input Signal %  
 I6 = INP(I6p) % Input Signal %  
 I7 = INP(I7p) % Input Signal %  
 ODDIN = INP(ODDINp) % Input Signal %  
 EVENIN = INP(EVENINp) % Input Signal %  
 SUMEVENp = CONF(SUMEVEN , VCC )  
 SUMODDp = CONF(SUMODD , VCC )  
 NODE13 = NOCF(NODE14)  
 NODE15 = NOCF(NODE16)

EQUATIONS:

NODE16 = I7'I6'I5'I4'  
 + I7'I6'I5'I4'  
 + I7'I6'I5'I4'  
 + I7'I6'I5'I4'  
 + I7'I6'I5'I4'  
 + I7'I6'I5'I4'  
 + I7'I6'I5'I4' ;

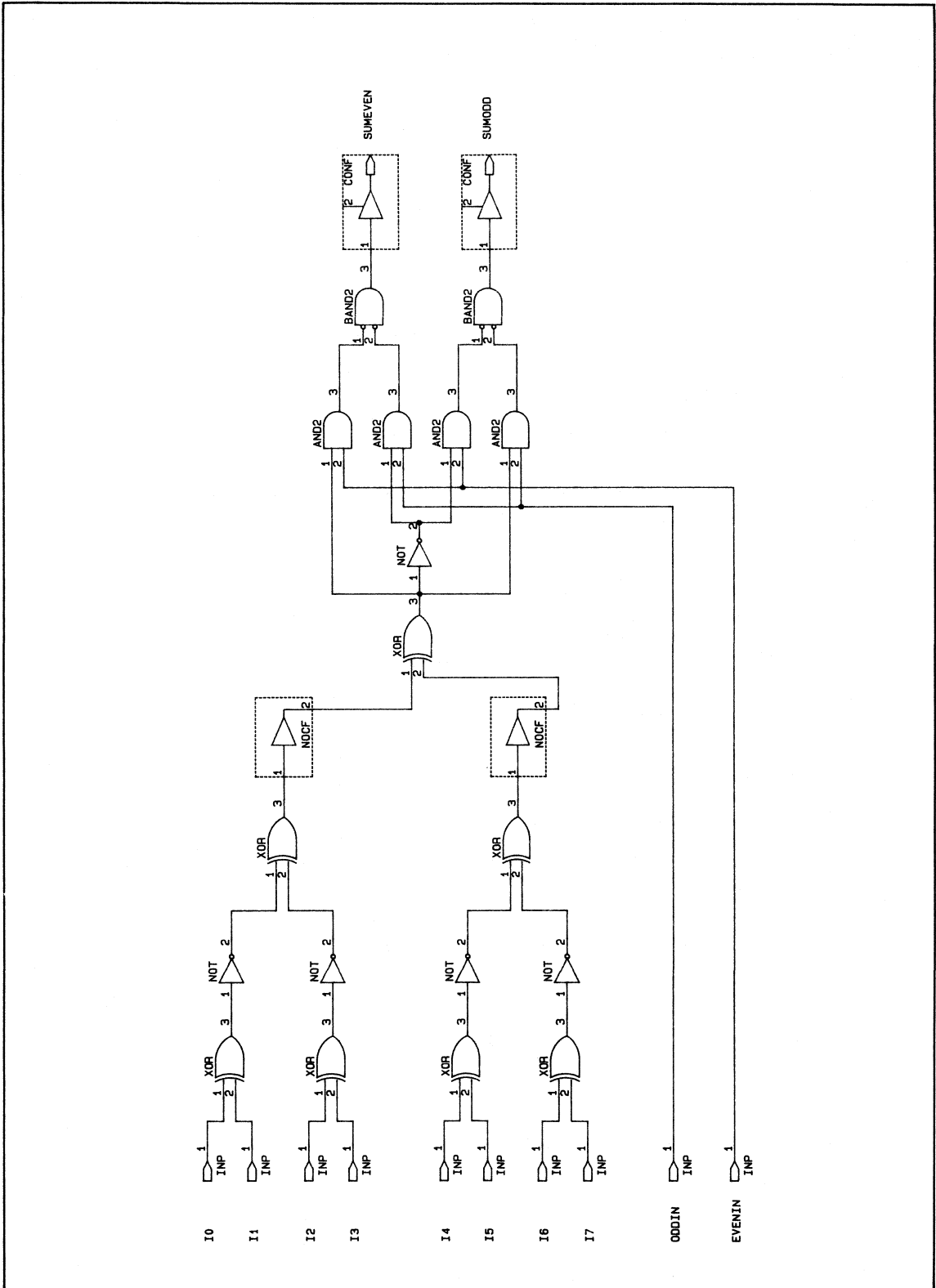
NODE14 = I3'I2'I1'I0'  
 + I3'I2'I1'I0'  
 + I3'I2'I1'I0'  
 + I3'I2'I1'I0'  
 + I3'I2'I1'I0'  
 + I3'I2'I1'I0'  
 + I3'I2'I1'I0' ;

SUMODD = /( EVENIN\*ODDIN  
 + NODE15'\*NODE13'\*EVENIN  
 + NODE15'\*NODE13'\*ODDIN  
 + NODE15'\*NODE13'\*ODDIN  
 + NODE15'\*NODE13'\*EVENIN ) ; % Inverted Equation %

SUMEVEN = /( EVENIN\*ODDIN  
 + NODE15'\*NODE13'\*ODDIN  
 + NODE15'\*NODE13'\*EVENIN  
 + NODE15'\*NODE13'\*EVENIN  
 + NODE15'\*NODE13'\*ODDIN ) ; % Inverted Equation %

END\$





Boolean design  
 ALTERA CORPORATION  
 MAR. 18, 1985  
 1.00

A

EP300

74190 UP/DOWN DEC. COUNT

PART:EP300

INPUTS:

P3p,PARLOADNp,COUNTENNp,P2p,P1p,UPNDOWNNp,P0p,CLOCKp

OUTPUTS:

Q3p,Q2p,Q1p,Q0p,TCp,RCNp

NETWORK:

CLOCK = INP(CLOCKp) % Clock Input %  
 P3 = INP(P3pp) % Input Signal %  
 PARLOADN = INP(PARLOADNp) % Input Signal %  
 COUNTENN = INP(COUNTENNp) % Input Signal %  
 P2 = INP(P2p) % Input Signal %  
 P1 = INP(P1p) % Input Signal %  
 UPNDOWN = INP(UPNDOWNp) % Input Signal %  
 P0 = INP(P0p) % Input Signal %  
 Q3p,Q3f = RORF(Q3 ,CLOCK, GND, GND , VCC ) % Registered %  
 Q2p,Q2f = RORF(Q2 ,CLOCK, GND, GND , VCC ) % Registered %  
 Q1p,Q1f = RORF(Q1 ,CLOCK, GND, GND , VCC ) % Registered %  
 Q0p,Q0f = RORF(Q0 ,CLOCK, GND, GND , VCC ) % Registered %  
 TCp,TCf = CORF(TC ,CLOCK, GND, GND , VCC ) % Registered %  
 RCNp= CONF(RCN , VCC )

EQUATIONS:

RCN = / ( COUNTENN\*TCf ) ; % Inverted Equation %

TC = Q3f\*Q0f+UPNDOWN'  
 + Q3f'\*Q2f'\*Q1f'\*Q0f'\*UPNDOWN ;

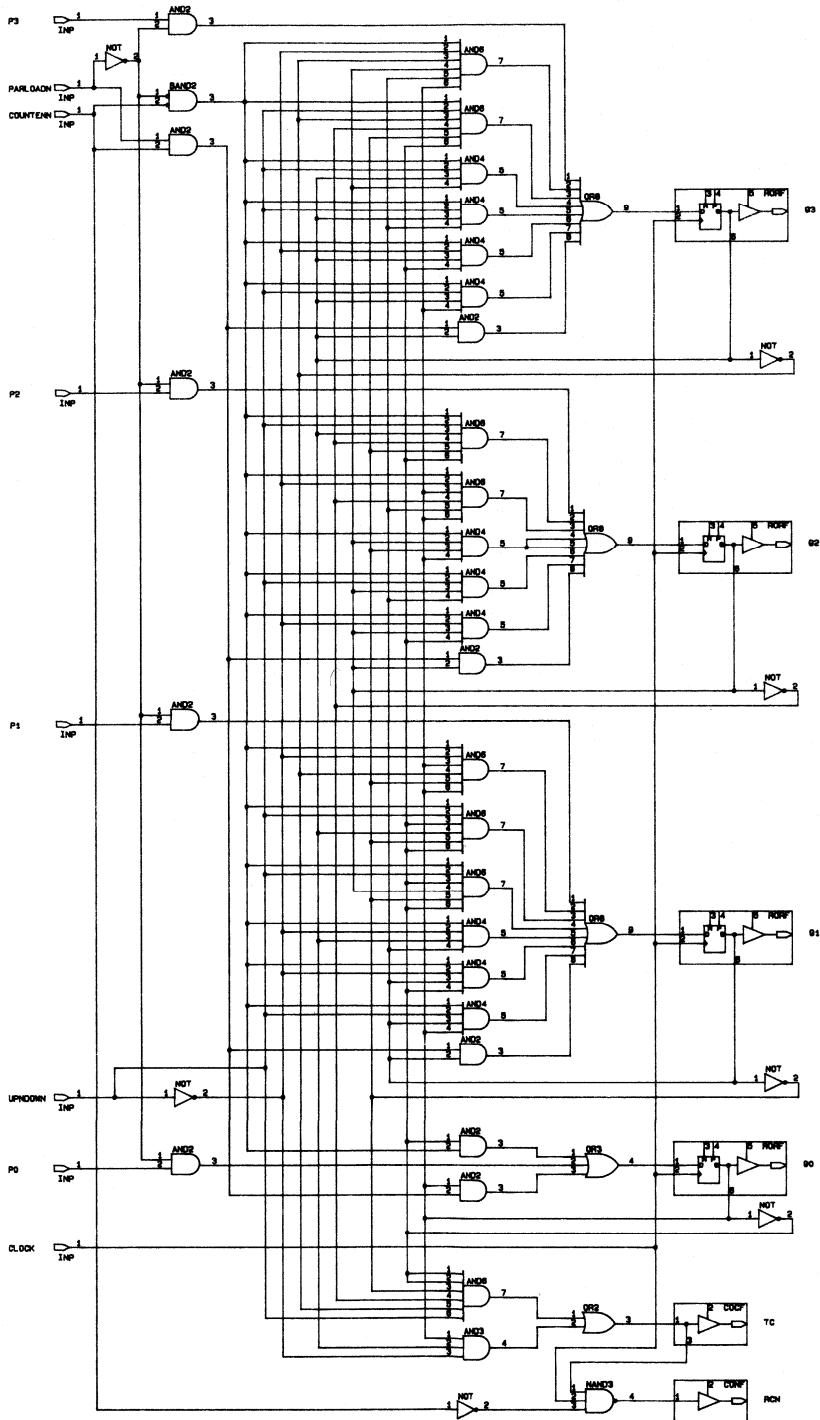
Q0 = PARLOADN'\*P0  
 + PARLOADN\*COUNTENN'\*Q0f'  
 + PARLOADN\*COUNTENN\*Q0f ;

Q1 = PARLOADN'\*P1  
 + PARLOADN\*Q1f\*COUNTENN  
 + PARLOADN\*Q1f\*UPNDOWN'\*Q0f'  
 + PARLOADN\*Q1f\*Q3f\*UPNDOWN'  
 + PARLOADN\*Q1f\*UPNDOWN\*Q0f  
 + PARLOADN\*Q1f\*Q3f\*UPNDOWN'\*COUNTENN'\*Q0f'  
 + PARLOADN\*Q1f\*UPNDOWN\*COUNTENN'\*Q0f'\*Q2f  
 + PARLOADN\*Q1f\*Q3f\*UPNDOWN\*COUNTENN'\*Q0f' ;

Q2 = PARLOADN'\*P2  
 + PARLOADN\*Q2f\*COUNTENN  
 + PARLOADN\*Q0f'\*Q2f\*UPNDOWN'  
 + PARLOADN\*Q0f'\*Q1f\*Q2f  
 + PARLOADN\*Q0f\*Q1f'\*Q2f  
 + PARLOADN\*Q0f\*Q2f\*UPNDOWN  
 + PARLOADN\*Q0f\*Q1f\*Q2f'\*COUNTENN'\*UPNDOWN'  
 + PARLOADN\*Q0f'\*Q1f'\*Q2f'\*COUNTENN'\*Q3f\*UPNDOWN ;

Q3 = PARLOADN'\*P3  
 + PARLOADN\*Q3f\*COUNTENN  
 + PARLOADN\*Q3f\*UPNDOWN'\*Q0f'  
 + PARLOADN\*Q3f\*UPNDOWN\*Q2f  
 + PARLOADN\*Q3f\*UPNDOWN\*Q1f  
 + PARLOADN\*Q3f\*UPNDOWN\*Q0f  
 + PARLOADN\*Q3f\*UPNDOWN\*COUNTENN'\*Q0f'\*Q1f\*Q2f'  
 + PARLOADN\*Q3f\*UPNDOWN'\*COUNTENN'\*Q0f\*Q1f\*Q2f ;

END\$



Boolean design  
 ALTERA CORPORATION  
 FEB. 20, 1985  
 1.00

A

EP300

74194 BIDIR. SHIFT REG.

PART:EP300

INPUTS:

DSLp, P3p, P2p, P1p, P0p, DSRp, RESETNp, SELECT1p,  
 SELECT0p, CLOCKp

OUTPUTS:

Q3p, Q2p, Q1p, Q0p

NETWORK:

CLOCK = INP(CLOCKp) % Clock Input %  
 DSL = INP(DSLp) % Input Signal %  
 P3 = INP(P3pp) % Input Signal %  
 P2 = INP(P2p) % Input Signal %  
 P1 = INP(P1p) % Input Signal %  
 P0 = INP(P0p) % Input Signal %  
 DSR = INP(DSRp) % Input Signal %  
 RESETN = INP(RESETNp) % Input Signal %  
 SELECT1 = INP(SELECT1p) % Input Signal %  
 SELECT0 = INP(SELECT0p) % Input Signal %  
 Q3p,Q3f = RORF(Q3 ,CLOCK, NODE13, GND , VCC ) % Registered %  
 Q2p,Q2f = RORF(Q2 ,CLOCK, NODE13, GND , VCC ) % Registered %  
 Q1p,Q1f = RORF(Q1 ,CLOCK, NODE13, GND , VCC ) % Registered %  
 Q0p,Q0f = RORF(Q0 ,CLOCK, NODE13, GND , VCC ) % Registered %

EQUATIONS:

Q0 = SELECT0'\*SELECT1'\*Q0f  
 + SELECT0\*SELECT1'\*DSR  
 + SELECT0'\*SELECT1\*Q1f  
 + SELECT0\*SELECT1\*P0 ;

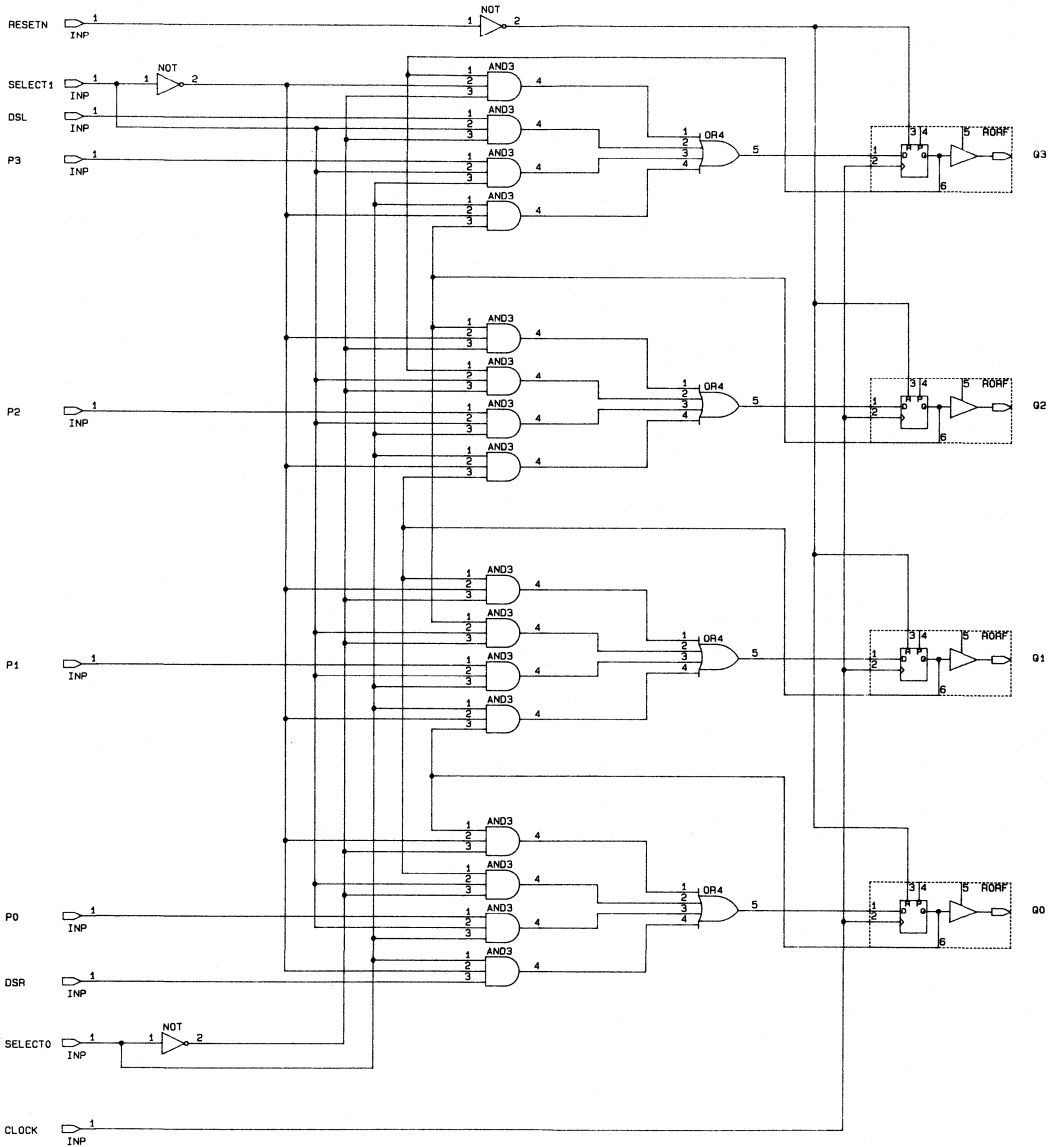
Q1 = SELECT0'\*SELECT1'\*Q1f  
 + SELECT0\*SELECT1'\*Q0f  
 + SELECT0'\*SELECT1\*Q2f  
 + SELECT0\*SELECT1\*P1 ;

Q2 = SELECT0'\*SELECT1'\*Q2f  
 + SELECT0\*SELECT1'\*Q1f  
 + SELECT0'\*SELECT1\*Q3f  
 + SELECT0\*SELECT1\*P2 ;

NODE13 = RESETN' ;

Q3 = SELECT0'\*SELECT1'\*Q3f  
 + SELECT0\*SELECT1'\*Q2f  
 + SELECT0'\*SELECT1\*DSL  
 + SELECT0\*SELECT1\*P3 ;

END\$



- Building block for Local Area Network communications system.

Manchester Encoders and Decoders are used in synchronous communications systems which require common mode isolation in point-to-point or common bus architectures. Examples include Ethernet and IEEE-802.3 standards. The Manchester Decoder would reside between a Local Area Network Controller and the Ethernet transceiver cable at the receiving end, and the Encoder would reside at the transmitting end.

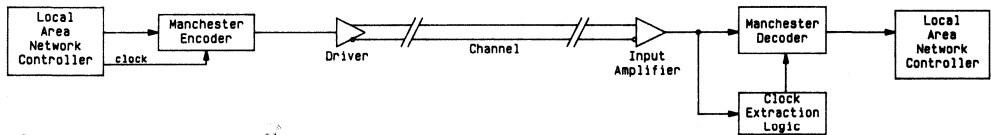
The Manchester Encoder converts separate clock and NRZ serial data signals into a standard Manchester II serial bit stream. The Manchester data format represents a logic 1 bit as a 0 followed by a 1 and a logic 0 bit as a 1 followed by a 0.

The Manchester Decoder restores the NRZ serial data signals from the Manchester data. The Decoder requires a clock signal to be extracted from the

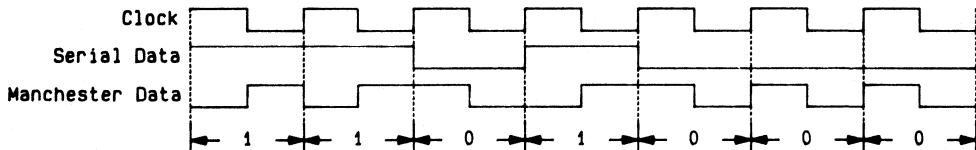
Manchester data stream by using an external phase locked loop circuit, for synchronization of the decoding process.

A State Machine Description of the Encoder and Decoder is used to derive Boolean logic equations for direct entry into the Altera Design File (ADF). Logic minimization and EPLD fitting is automatically performed by the Altera Programmable Logic User System, resulting in a JEDEC file suitable for programming the EPLD. Also generated is a utilization report that describes what resources of the EPLD were required in fitting the design, and what pin numbers were assigned to input and output signals.

From the figure it is apparent that only a small portion of the EPLD is required to implement the Manchester Decoder or Encoder. The remainder of the chip could be used for the actual serial receiver or transmitter, or for the Local Area Network controller and associated logic.

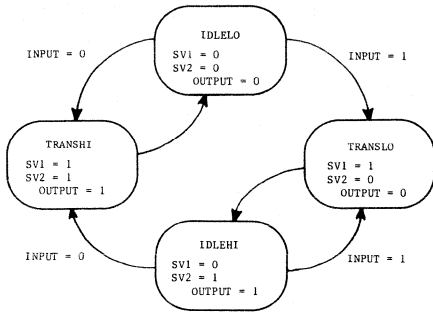


**Figure 1.** Local Area Network Communication systems employ manchester encoded serial data in the communication channel. This permits the data to carry its own synchronization clock, and allows continuity over an AC system.



Manchester data represents a logic 1 as a 0 followed by a 1, and a logic 0 as a 1 followed by a 0.

Encoder state diagram and table

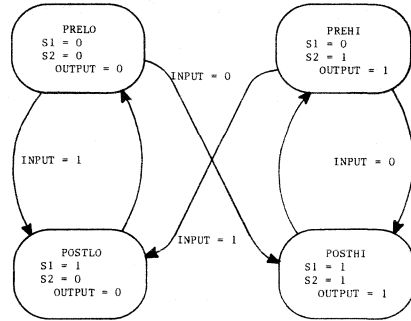


Manchester Encoder State Table

Current State	Serial Input SV1	Serial Input SV2 (SIGIN)	Next State	Manchester Output SV1d	Manchester Output SV2d (SV2)	Manchester Output SV2d (SV2)	
IDLELO	0	0	0	TRANSHI	1	1	0
IDLELO	0	0	1	TRANSLO	1	0	0
TRANSHI	1	1	X	IDLELO	0	0	1
IDLEHI	0	1	0	TRANSHI	1	1	1
IDLEHI	0	1	1	TRANSLO	1	0	1
TRANSLO	1	0	X	IDLEHI	0	1	0

X = don't care

Decoder state diagram and table



Manchester Decoder State Table

Current State	Manchester Input S1	Manchester Input S2 (MANIN)	Next State	Serial Output S1d	Serial Output S2d (S1)	Serial Output S2d (S1)	
PRELO	0	0	0	POSTHI	1	1	0
PRELO	0	0	1	POSTLO	1	0	0
PREHI	0	1	0	POSTHI	1	1	1
PREHI	0	1	1	POSTLO	1	0	1
POSTLO	1	0	X	PRELO	0	0	0
POSTHI	1	1	X	PREHI	0	1	1

X = don't care

MANCH\_SEQ  
06-25-1985 19:47:34  
MANCHESTER ENCODER

PART:EP1200

INPUTS: CLOCKp, SIGINp/2  
OUTPUTS: SV1p, SV2p

NETWORK:

CLK = INP(CLOCKp) % clock input %  
SIGIN = INP(SIGINp)

SV1p, SV1 = RORF(SV1d, CLK, GND, GND, VCC)  
SV2p, SV2 = RORF(SV2d, CLK, GND, GND, VCC)

EQUATIONS:

IDLELO = /SV1 \* /SV2;  
IDLEHI = /SV1 \* SV2;  
TRANSLO = SV1 \* /SV2;  
TRANSHI = SV1 \* SV2;

SV1d = TRANSLO  
+ /SIGIN \* IDLELO  
+ /SIGIN \* IDLEHI;

SV2d = /(TRANSHI + TRANSLO);

ENDS

MANCHD\_SEQ  
07-11-1985 15:48:07  
MANCHESTER DECODER

PART:EP1200

INPUTS: CLOCKp, MANINp/2  
OUTPUTS: S1p, S2p

NETWORK:

CLK = INP(CLOCKp) % clock input %  
MANIN = INP(MANINp)

S1p, S1 = RORF(S1d, CLK, GND, GND, VCC)  
S2p, S2 = RORF(S2d, CLK, GND, GND, VCC)

EQUATIONS:

PRELO = /S1 \* /S2;  
PREHI = /S1 \* S2;  
POSTLO = S1 \* /S2;  
POSTHI = S1 \* S2;

S1d = /(POSTLO  
+ POSTHI);

S2d = POSTHI  
+ /MANIN \* PRELO  
+ /MANIN \* PREHI;

ENDS

Altera Design Files (ADFs) ready to be input to the A+PLUS software where they will be combined with ADFs of other logic for integration in an EPLD.

- Multiplexer for 24 channel 8 bit data meeting telecommunications T1 carrier standards.
- Generates output as 193 bit serial data stream in T-1, D2, D3 or D4 Mode 3 data format.
- Accepts 8 bits of parallel data as inputs.
- Provides channel and frame synchronization timing signals.
- Provides alternate control for alarm reporting and signaling.
- Provides automatic bit insertion for all-zero channel samples.
- Provides current bit, frame, and channel selection information.
- Provides both Binary and Paired Unipolar Outputs.

#### GENERAL DESCRIPTION

This circuit design, when implemented in an ALTERA EP1200 EPLD, becomes a single chip T-1 Serial Transmitter. Data is formatted to be serially transmitted according to T-1 D2 or T-1 D3 specifications, inserting framing and signalling bits along with 24 channels of 8 bit channel data. The circuit provides for alarm reporting via the Bit 2 inhibit method or via the multiframe alignment signal (Fs) modification method.

Figure 1 is a functional block diagram of the T-1 Serial Transmitter. A 1.544 MHz clock runs a modulo 193 counter for generation of bit and channel timing, and a modulo 12 counter for Frame timing. The counters may be externally synchronized by use of the

SYNCIN and FRSYNC inputs, or external circuitry may synchronize to the CHCLKF or SYNOUT outputs.

All inputs are latched when the 8th bit of any channel is being transmitted. The Data Selector outputs the proper sequence of bits, as controlled by the bit count and frame count.

The zero channel monitor function causes Bit 8 to be transmitted as a "one" if the channel data sample is all "zeros." This function could easily be modified to provide zero detection to be transmitted on Bit 7, or to be inhibited by appropriate changes to the schematic prior to programming the EPLD.

#### T-1 TRANSMITTER INPUTS

All inputs except the CLOCK inputs pass through transparent latches that are controlled by the LATCH input. When LATCH is HIGH, the latches are in their transparent mode. Normally the LATCH input would be connected directly to the CHCLKF output resulting in the inputs being sampled as the 8th bit of any channel is being transmitted. Note that for this reason synchronization input signals must be asserted for a minimum of 9 clock cycles to guarantee their recognition by the internal counters.

#### FRSYNC: Frame Synchronization

Frame sync permits external synchronization of the transmitter's internal frame counter. When CHCLKF

**Fig. 1.**  
Functional Block Diagram of T-1 Serial Transmitter.

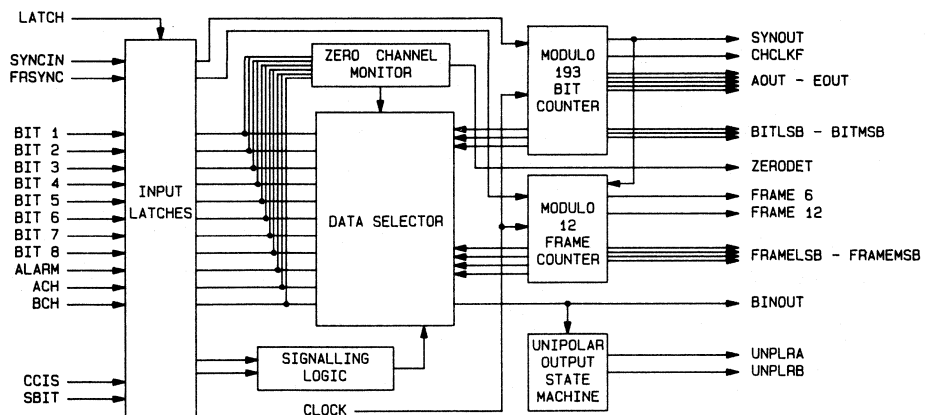






FIG. 3

A+PLUS Generates Logic equations from the Schematic in Figure 2 and outputs the Logic Equation File shown here.

```

design by: BOB DUNCAN
ALTERA
Aug 30, 1985
1.00
B
EP1200
T-1 Transmitter
PART: EP1200

INPUTS: BIT1,BIT2,BIT3,BIT4,BIT5,BIT6,BIT7,BIT8,
        ALARM,BCH,ACH,LATCH,CCIS,SBIT,CLOCK,
        FRSYNC,SYNCIN

OUTPUTS: BINOUT, AOUT, BOUT, COUT, DOUT, EOUT, BITMSB,
         BITMID, BITLSB, CHCLKF, SYNOUT, FRMSB, FRMMID,
         FRMLMID, FRMSBf, ZERODET, UNPLRA, UNPLRf

NETWORK:
LATCH = INP(LATCH)      % Input Signal %
CLOCK = INP(CLOCK)     % Input Signal %
BIT1  = LINP(BIT1, LATCH) % Input Signal %
BIT2  = LINP(BIT2, LATCH) % Input Signal %
BIT3  = LINP(BIT3, LATCH) % Input Signal %
BIT4  = LINP(BIT4, LATCH) % Input Signal %
BIT5  = LINP(BIT5, LATCH) % Input Signal %
BIT6  = LINP(BIT6, LATCH) % Input Signal %
BIT7  = LINP(BIT7, LATCH) % Input Signal %
BIT8  = LINP(BIT8, LATCH) % Input Signal %
ALARM = LINP(ALARM, LATCH) % Input Signal %
BCH   = LINP(BCH, LATCH)  % Input Signal %
ACH   = LINP(ACH, LATCH)  % Input Signal %
CCIS  = LINP(CCIS, LATCH) % Input Signal %
SBIT  = LINP(SBIT, LATCH) % Input Signal %
FRSYNC = LINP(FRSYNC, LATCH) % Input Signal %
SYNCIN = LINP(SYNCIN, LATCH) % Input Signal %
AOUT, AOUTf = RORF(AOUT, CLOCK, GND, GND, VCC)
BOUT, BOUTf = RORF(BOUT, CLOCK, GND, GND, VCC)
COUT, COUTf = RORF(COUT, CLOCK, GND, GND, VCC)
DOUT, DOUTf = RORF(DOUT, CLOCK, GND, GND, VCC)
EOUT, EOUTf = RORF(EOUT, CLOCK, GND, GND, VCC)
BITMSB, BITMSBf = RORF(BITMSB, CLOCK, GND, GND, VCC)
BITMID, BITMIDf = RORF(BITMID, CLOCK, GND, GND, VCC)
BITLSB, BITLSBf = RORF(BITLSB, CLOCK, GND, GND, VCC)
CHCLKF = RORF(NODE34, CLOCK, GND, GND, VCC)
SYNOUT, SYNOUTf = RORF(SYNOUT, CLOCK, GND, GND, VCC)
FRMSB, FRMSBf = RORF(FRMSB, CLOCK, GND, GND, VCC)
FRMMID, FRMMIDf = RORF(FRMMID, CLOCK, GND, GND, VCC)
FRMLMID, FRMLMIDf = RORF(FRMLMID, CLOCK, GND, GND, VCC)
FRMSBf, FRMSBf = RORF(FRMSBf, CLOCK, GND, GND, VCC)
ZERODET, ZERODETf = RORF(ZERODET, CLOCK, GND, GND, VCC)
UNPLRA, UNPLRAf = RORF(UNPLRA, CLOCK, GND, GND, VCC)
UNLRF, UNLRFf = RORF(UNLRF, CLOCK, GND, GND, VCC)
BINOUT, BINOUTf = RORF(BINOUT, CLOCK, GND, GND, VCC)
NODE53 = NORF(NODE54, CLOCK, GND, GND)
NODE55 = NORF(NODE56, CLOCK, GND, GND)
NODE57 = NORF(NODE58, CLOCK, GND, GND)
NODE59 = NORF(NODE60, CLOCK, GND, GND)
NODE61 = NORF(NODE62, CLOCK, GND, GND)
NODE63 = NORF(NODE64, CLOCK, GND, GND)

EQUATIONS:
NODE64 = UNLRFf * UNLRAf
+ NODE63 * UNLRFf'
+ NODE63 * UNLRAf;

NODE62 = NODE59
+ ZERODETf * NODE53 * NODE55';

NODE60 = CCIS * BIT8
+ CCIS' * NODE55 * BCH
+ ACH * NODE53 * CCIS
+ NODE53' * CCIS' * NODE55' * BIT8;

NODE58 = BIT7
+ ZERODETf * NODE55
+ NODE53 * ZERODETf;

NODE56 = FRMSBf * FRMLMIDf * FRMSBf * FRMMIDf;

NODE54 = FRMMIDf * FRMSBf * FRMSBf * FRMLMIDf;

BINOUT = BITLSBf' * BITMIDf * BITMSBf' * BIT3
+ BITLSBf' * BITMIDf' * BITMSBf' * BIT5
+ BITLSBf * BITMIDf * BITMSBf * BIT4
+ BITLSBf * BITMIDf' * BITMSBf * BIT6
+ BITLSBf' * BITMIDf * BITMSBf * NODE57
+ NODE61 * BITLSBf * BITMIDf * BITMSBf
+ BITLSBf' * BITMIDf * BITMSBf' * BIT1 * SYNOUTf
+ BITLSBf * BITMIDf' * BITMSBf' * BIT2 * ALARM'
+ BITLSBf' * BITMIDf' * BITMSBf' * SYNOUTf *
FRMSBf * FRMLMIDf
+ BITLSBf' * BITMIDf' * BITMSBf' * SYNOUTf *
FRMSBf' * FRMMIDf * CCIS'
+ BITLSBf' * BITMIDf' * BITMSBf' * SYNOUTf *
FRMSBf' * CCIS * SBIT
+ BITLSBf' * BITMIDf' * BITMSBf' * SYNOUTf *
FRMSBf' * CCIS' * FRMSBf * FRMLMIDf';

```

```

UNLRF = BINOUTf * UNLRFf' * UNLRAf
+ BINOUTf * NODE63 * UNLRFf'
+ BINOUTf * NODE63 * UNLRAf;

UNLRA = BINOUTf * NODE63' * UNLRAf'
+ BINOUTf * UNLRAf' * UNLRFf
+ BINOUTf * NODE63' * UNLRFf;

ZERODET = NODE59' * BIT1' * BIT3' * BIT4' *
BIT5' * BIT6' * BIT7' * BIT2'
+ NODE59' * BIT1' * BIT3' * BIT4' * BIT5' *
BIT6' * BIT7' * ALARM;

FRMSB = FRMSBf' * FRSYNC'
+ FRMSBf' * SYNOUTf'
+ FRMSBf' * SYNCIN
+ FRMSBf * SYNCIN' * SYNOUTf;

FRMLMID = FRMLMIDf * SYNOUTf'
+ FRMLMIDf * SYNCIN
+ FRMLMIDf * FRMSBf' * FRSYNC
+ FRMLMIDf' * SYNCIN' * SYNOUTf * FRMSBf *
FRSYNC;

FRMMID = FRMMIDf * SYNOUTf'
+ FRMMIDf * SYNCIN
+ FRMMIDf * FRMLMIDf' * FRSYNC
+ FRMMIDf * FRMSBf' * FRSYNC
+ FRMMIDf * FRMSBf * FRSYNC
+ FRMMIDf * FRMSBf * FRMLMIDf * SYNCIN' *
SYNOUTf * FRMSBf * FRSYNC;

FRMSBf = FRMSBf * SYNOUTf'
+ FRMSBf * SYNCIN
+ FRMSBf * FRMLMIDf' * FRSYNC
+ FRMSBf * FRMSBf' * FRSYNC
+ FRMSBf' * FRMMIDf * SYNCIN' * SYNOUTf *
FRMSBf * FRMLMIDf * FRSYNC;

SYNOUT = SYNCIN
+ AOUTf * BOUTf * COUTf * DOUTf * EOUTf *
BITMSBf * BITMIDf * BITLSBf;

NODE34 = SYNCIN
+ BITMIDf * BITLSBf' * BITMSBf * AOUTf'
+ BITMIDf * BITLSBf' * BITMSBf * BOUTf'
+ BITMIDf' * BITLSBf * BITMSBf * BOUTf * AOUTf
+ BITLSBf * BITMSBf * BOUTf * AOUTf * COUTf *
DOUTf * EOUTf;

BITLSB = BITLSBf' * SYNCIN' * AOUTf'
+ BITLSBf' * SYNCIN' * BOUTf'
+ BITLSBf * SYNCIN' * BOUTf * AOUTf;

BITMID = BITMIDf' * BITLSBf * SYNCIN'
+ BITMIDf * BITLSBf' * SYNCIN;

BITMSB = BITMSBf * SYNCIN' * BITLSBf'
+ BITMSBf * SYNCIN' * BITMIDf'
+ BITMSBf' * SYNCIN' * BITMIDf * BITLSBf;

EOUT = EOUTf * SYNCIN' * BITLSBf
+ EOUTf * SYNCIN' * BITMIDf'
+ EOUTf * SYNCIN' * BITMSBf'
+ EOUTf' * SYNCIN' * BITMSBf * BITMIDf *
BITLSBf;

DOUT = DOUTf * SYNCIN' * BITLSBf'
+ DOUTf * SYNCIN' * BITMIDf'
+ DOUTf * SYNCIN' * BITMSBf'
+ DOUTf * SYNCIN' * EOUTf * BITMSBf *
BITMIDf * BITLSBf;

COUT = COUTf * SYNCIN' * BITLSBf'
+ COUTf * SYNCIN' * BITMIDf'
+ COUTf * SYNCIN' * BITMSBf'
+ COUTf * SYNCIN' * EOUTf
+ COUTf * SYNCIN' * DOUTf'
+ COUTf' * SYNCIN' * DOUTf * EOUTf *
BITMSBf * BITMIDf * BITLSBf;

BOUT = SYNCIN
+ BOUTf * AOUTf' * BITLSBf'
+ BOUTf * AOUTf' * BITMIDf'
+ BOUTf * AOUTf' * BITMSBf'
+ BOUTf * AOUTf * EOUTf
+ BOUTf * AOUTf * DOUTf'
+ BOUTf * COUTf' * AOUTf'
+ BOUTf' * COUTf * DOUTf * EOUTf *
BITMSBf * BITMIDf * BITLSBf;

AOUT = SYNCIN
+ AOUTf * BOUTf'
+ AOUTf' * BOUTf * COUTf * DOUTf *
EOUTf * BITMSBf * BITMIDf * BITLSBf;

```

END\$

becomes high with FRSYNC set high, the frame counter is set to frame 1, corresponding to pins FRMMSB, FRMMID, FRMLMID, and FRMSB all at logic 0.

**SYNCIN:**

**Synchronization Input**

SYNCIN permits external synchronization of the modulo 193 bit / channel counter. When CHCLKF becomes high with SYNCIN at a high state, the modulo 193 counter is directly set to the state corresponding to the output of the framing (Ft or Fs) bit. CHCLKF will remain asserted and the modulo 193 counter will not increment until SYNCIN is brought back to a low level.

**BITS 1-8:**

**Parallel Channel Data Inputs**

BIT1, the sign bit, will be serially transmitted first, followed by bits 2 through 8. The data is latched by the LATCH input, normally connected to the CHCLKF output. The data should be stable for the duration of the CHCLKF pulses to avoid errors.

**ACH:**

**"A" Channel Highway Signalling**

ACH allows the user to transmit one bit of signalling per channel as Bit 8 of each channel data sample in Frame 6 only.

**BCH:**

**"B" Channel Highway Signalling**

BCH allows the user to transmit one bit of signalling per channel as Bit 8 of each channel data sample in Frame 12 only.

**SBIT:**

**Multiframe Signalling Bit**

SBIT, in conjunction with CCIS, provides an alternate way to control the multiframe signalling bit (Fs) transmission. The SBIT input is transmitted as the multiframe signalling bit (Fs) if CCIS is held high.

**ALARM:**

**Local Alarm Reporting Input**

Used for reporting alarm conditions. If the ALARM input is held high, Bit 2 (the most significant bit) of every channel data sample of every frame is suppressed (set to logic low). This is referred to as remote alarm signalling.

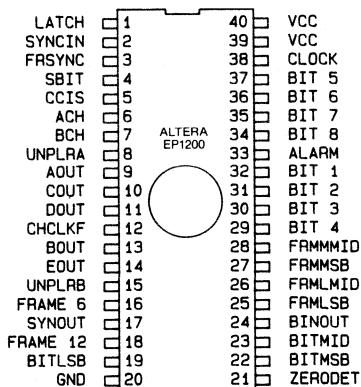
**CCIS:**

**Common Channel Interoffice Signalling Strap**

Provides optional control for substitution of the SBIT data in place of the automatic Fs pattern with a 4-kilobit common channel signalling path. When CCIS is held high, the SBIT input replaces the Fs pattern and the insertion of ACH and BCH is suspended. The CCIS input may also be used to provide the alternate method of alarm reporting.

**FIG. 4**

Pinout of the EP1200 programmed as the T-1 Serial Transmitter.



**LATCH:**

**Input Transparent Latch Enable**

Enables the transparent latches on all data and synchronization inputs. When LATCH is at a logic high, all latches are in there transparent mode. LATCH would normally be connected directly to the CHCLKF output resulting in the inputs being sampled every 8 clock cycles, or 9 when the frame bit is transmitted. If it is desirable to be able to synchronize the internal counters in only one clock cycle, LATCH must be forced high while applying the synchronization data to the FRSYNC or SYNCIN pins.

**CLOCK:**

**T-1 Clock**

The T-1 bit period is bounded by the rising edges of this input. If it is desirable to use the falling edge, a NOT gate could be inserted in the schematic diagram prior to programming the EP1200. Normal T-1 frequency is 1.544 MHz however the EP1200 will operate in excess of 15MHz if desired. Pulse width minimums are in the range of 20 to 30 nanoseconds. (Refer to EP1200 and EP1210 data sheets).

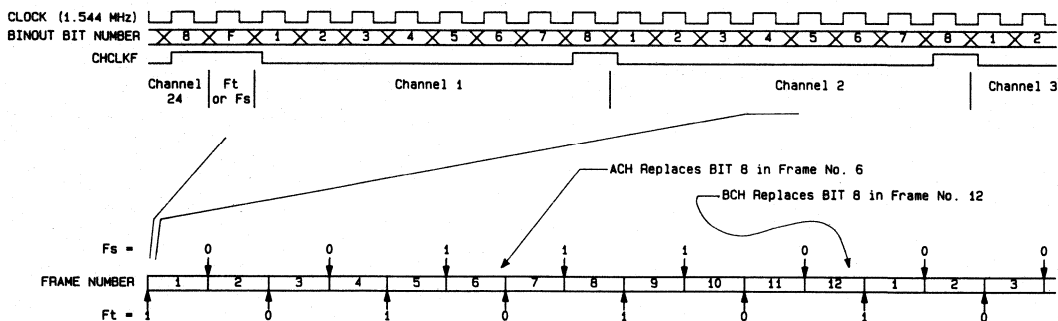
**Table 1.**

Serial Channel Sample Output Data Truth Table.

ALARM	Inputs X = don't care								Current Frame Number	BINOUT Serial Output									
	Input Channel Data Bit									Channel Bit Position									
	1	2	3	4	5	6	7	8		ACH	BCH	1	2	3	4	5	6	7	8
1	X	X	X	X	X	X	X	X	X	X	X	0	X	X	X	X	X	X	
X	X	0	X	X	X	X	X	X	X	X	X	X	0	X	X	X	X	X	
0	P	Q	R	S	T	U	V	X	A	X	6	P	Q	R	S	T	U	V	A
0	P	Q	R	S	T	U	V	X	B	12	P	Q	R	S	T	U	V	B	
0	P	Q	R	S	T	U	V	W	X	Y	P	Q	R	S	T	U	V	W	
0	0	0	0	0	0	0	0	X	0	6	0	0	0	0	0	0	1	0	
0	0	0	0	0	0	0	0	X	0	12	0	0	0	0	0	0	1	0	
0	0	0	0	0	0	0	0	X	X	Y	0	0	0	0	0	0	0	1	

NOTE: Y is any Frame except 6 and 12 with CCIS = 0, or all Frames with CCIS = 1.

**FIG. 5**  
Transmitter Input — Output Signal Relationships.



## T-1 TRANSMITTER OUTPUTS

### AOUT - EOUT:

#### Channel Number Transmitted

Most significant bits of the modulo 193 counter identifies which of the 24 channels is currently being transmitted.

### BITMSD, BITMID, BITLSD:

#### Bit Number Transmitted

Least significant bits of the modulo 193 counter identifies which bit of the channel is next to be transmitted.

### CHCLKF:

#### Channel Clock

Pulse that is at a logic high for one bit time as bit 8 of any channel is transmitted, or when SYNCIN is held high. This output will be high during the transmission of the framing bit.

### FRMMSB, FRMMID, FRMLMID, FRMLSB:

#### Frame Number Transmitted

Frame counter state output. FRMMSB is the most significant bit and FRMLSB is the least significant bit. The frame counter is a modulo 12 counter ranging from 0 to 11.

### FRAME6, FRAME12:

#### Signalling and Synchronization Frames

These outputs are at a logic high during the transmission of frames 6 (12). Frame 6 is when the ACH data replaces BIT 8 of the transmitted data, and Frame 12 is when the BCH data replaces BIT 8 of the transmitted data.

### ZERODET:

#### Zero Channel Detection

A logic high on this output indicates that the data to be transmitted is composed of all zeros, and that automatic bit insertion is in effect. For channels in signalling frames (6 or 12) in which the first six data bits and the signalling highway are all "zero," BIT 7 will be forced to a logic "one." For the other frames if an all "zero" condition exists, BIT 8 of the transmitted data will be forced to a logic "one."

### BINOUT:

#### Serial Data Output, Binary Formatted

BINOUT is the binary formatted serial conversion of the parallel input data and the signalling data. BINOUT data is delayed by one clock cycle from the associated state of the internal counters.

### UNPLRA, UNPLRB:

#### T-1 Serial Data Unipolar Outputs

Two paired unipolar outputs are provided for the purpose of creating a single serial data output transmission in bipolar format. The two outputs are both at a logic low for an equivalent "zero" on the BINOUT output, and are complements of each other for an equivalent "one" on the BINOUT output. Each time a one is transmitted, the sense of the two outputs is reversed. UNPLRA and UNPLRB are delayed by one clock pulse from the data on BINOUT, and by two clock pulses from the internal counter states.

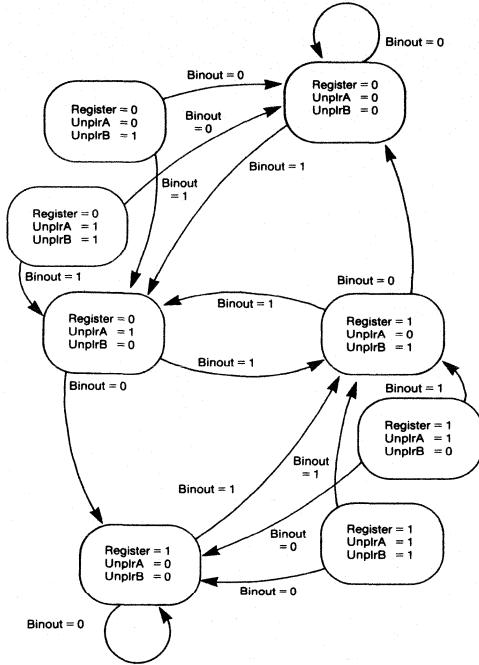
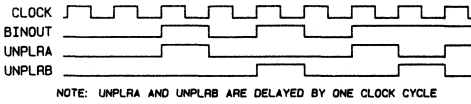
### SYNOUT:

#### Channel Synchronization Output

SYNOUT provides a means to synchronize to the modulo 193 bit / channel counter. SYNOUT is high for one bit time during the transmission of the framing bit.

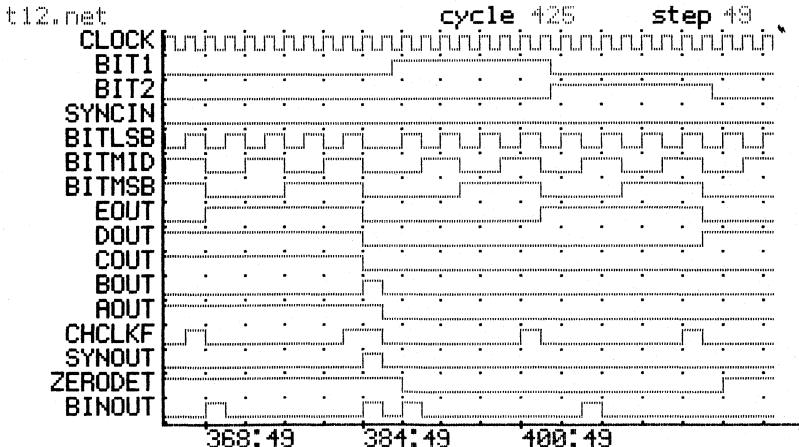
**FIG. 6**

Timing and State Diagram for the logic used to generate the unipolar outputs from the binary output.



**FIG. 7**

Example output from examining this design with the Altera logic simulation package PLSIM1. The pulse on the SYNOUT line occurs during the Frame Bit transmission, followed by Bits 1 to 8 of the Channel Data.



=>sim 64

SPK STB ATS DSP TYP SPL

HOLD

- Single chip adds automatic self diagnostics capability to intelligent systems.
- Provides simple go — no go testing of complex digital signals.
- Custom character set allows viewing upside down.
- Multiple design entry methods allow natural representation of design.
- Low power and small size ease upgrading existing designs to include self diagnostics.

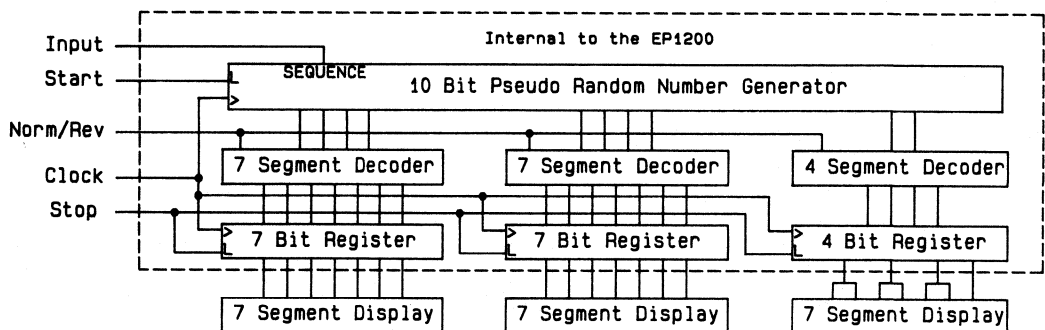
Signature Analyzers are useful tools for troubleshooting complex digital logic circuits. The circuit under test is forced into a continuous test loop by built in logic, such that each accessible node has a unique characteristic switching waveform. The Signature Analyzer is then used to sample this waveform and determine a digital code unique to that waveform.

This code, or 'Signature,' is then compared with that of a known good value to aid in locating any problems. The comparison could be done either by a Technician with a properly marked schematic, or automatically by a computer with an appropriate data base of good signatures.

The design presented here uses a shift register that is configured to produce a pseudo random sequence of binary numbers. The sequence is in-

fluenced by the incoming test data resulting in a different sequence for any given test data waveform. The Unit Under Test (UUT) generates a synchronizing clock and START and STOP pulses separated by a constant duration. The shift register is set to an initial state (000000001) upon reception of the START pulse, and its data is transferred (via a decoder) to the output registers upon reception of the STOP pulse. Because of the pseudo random nature of the shift register, a given test signal input in a given constant period will always end up with the same value, and any change in the test signal, be it pulse duration, pulse position, missing or extra pulses, will result in a different value. This would be true if the shift register were able to produce an infinitely long random sequence without repeating. The shift register used in this design is 10 bits in length and has feedback in such a manner as to give a maximum length sequence of  $2^{10}$  or 1024. Thus the odds of an incorrect signal resulting in a correct signature are 1 in 1024. These odds could be reduced by use of a longer shift register, or by providing selectable feedback taps which would give selectable sequences. Using a selectable sequence, the schematic would be annotated with both patterns. If both sequences were independent and random, the resulting odds of getting a false 'good' pair of patterns would be 1 in 1,048,576.

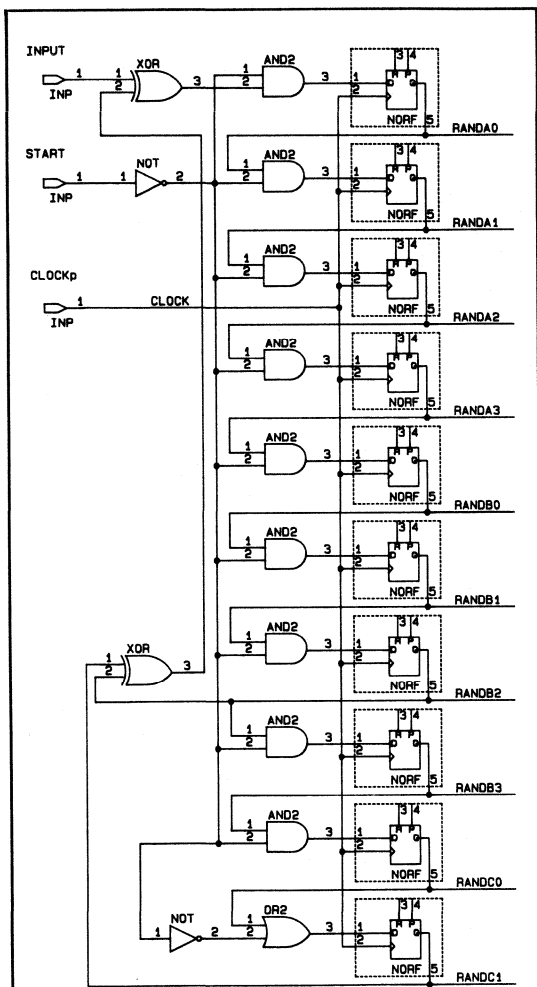
The programmability of the EPLD is very useful for



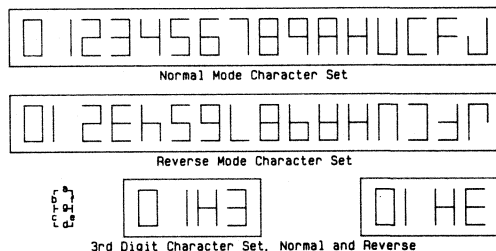
**Fig. 1.**

10 bit pseudo random sequencer periodically feeds data to output registers which in turn drive the LED display. The decoding is done before the output registers because of the internal architecture of the EPLD.

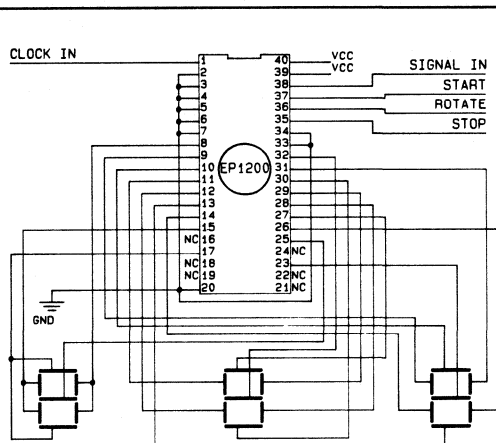
incorporating the signature analyzer into a larger system. As shown in Figure 6 the output decoders and registers could be replaced with a serial transmitter for connection to a computer. Another possibility is to replace the decoders and registers with appropriate interface logic such that the EPLD would be connected directly to a computer's data buss. The EPLD could even be connected directly to a ROM programmed with expected signatures, and a built in state machine would control the comparisons and flag errors.



**Fig. 2.** A maximum sequence pseudo random generator is created by summing selected points in a shift register and feeding the result back to the shift registers input. The signal under test is included in the (MOD 2) sum and will thus influence the resulting sequence. The START pulse forces the shift register to an initial value for repeatability. This portion of the design was input to A+PLUS by use of the Schematic Capture System.



**Fig. 3.** In addition to the normal 0-9 digits, the decoder logic includes 6 letters. These letters were picked so that they would be distinguishable even if viewed upside down. Asserting the ROTATE input causes the display to read upside down as would be useful in portable applications. This is one of those 'free' features that make programmable logic so versatile. Note the third digit has a restricted character set due to the limitation in number of outputs available. Tying appropriate segments of the display together gives readable characters with only four lines.



**Fig. 4.** Three LED displays and one EP1200 make up a complete test fixture suitable to be mounted on an extender card or permanently designed into a system. If LCD displays were used, the power consumption of the total system would be negligible and the whole unit could fit on a few square inches of board space.

```

Altera Design File
7 meg decoder for Signature Analyzer
sequence:
A and B digits 0 1 2 3 4 5 6 7
sequence:      8 9 A H U C F J
Digit         0 H 3
PART:         EP1200
INPUTS:      ROTATEp,STOPp
OUTPUTS:
a0p,b0p,c0p,d0p,e0p,f0p,g0p,
a1p,b1p,c1p,d1p,e1p,f1p,g1p,
a2p,b2p,e2p,g2p

NETWORK:
ROTATE = !NF(ROTATEp)
STOP = !NF(STOPp)

a0d = ORF(a0d,CLOCK,,)
b0d,b0f = ORF(b0d,CLOCK,,)
c0p,c0f = ORF(c0d,CLOCK,,)
d0p,d0f = ORF(d0d,CLOCK,,)
e0p,e0f = ORF(e0d,CLOCK,,)
f0p,f0f = ORF(f0d,CLOCK,,)
g0p,g0f = ORF(g0d,CLOCK,,)
e1p,e1f = ORF(e1d,CLOCK,,)
b1p,b1f = ORF(b1d,CLOCK,,)
c1p,c1f = ORF(c1d,CLOCK,,)
d1p,d1f = ORF(d1d,CLOCK,,)
e1p,e1f = ORF(e1d,CLOCK,,)
f1p,f1f = ORF(f1d,CLOCK,,)
g1p,g1f = ORF(g1d,CLOCK,,)
a2p,a2f = ORF(a2d,CLOCK,,)
b2p,b2f = ORF(b2d,CLOCK,,)
e2p,e2f = ORF(e2d,CLOCK,,)
g2p,g2f = ORF(g2d,CLOCK,,)

EQUATIONS:
a0d = a0e * STOP + a0f * /STOP;
b0d = b0e * STOP + b0f * /STOP;
c0d = c0e * STOP + c0f * /STOP;
d0d = d0e * STOP + d0f * /STOP;
e0d = e0e * STOP + e0f * /STOP;
f0d = f0e * STOP + f0f * /STOP;
g0d = g0e * STOP + g0f * /STOP;
a1d = a1e * STOP + a1f * /STOP;
b1d = b1e * STOP + b1f * /STOP;
c1d = c1e * STOP + c1f * /STOP;
d1d = d1e * STOP + d1f * /STOP;
e1d = e1e * STOP + e1f * /STOP;
f1d = f1e * STOP + f1f * /STOP;
g1d = g1e * STOP + g1f * /STOP;
a2d = a2e * STOP + a2f * /STOP;
b2d = b2e * STOP + b2f * /STOP;
e2d = e2e * STOP + e2f * /STOP;
g2d = g2e * STOP + g2f * /STOP;

g0e = ( AO * A1 * A2 * A3 +
        AO * A1 * A2 * A3 +
        AO * A1 * A2 * A3 +
        AO * A1 * A2 * A3 +
        /AO * /A1 * /A2 * /A3 );
    
```

Fig. 5.

A text editor was used to enter logic equations into the Altera Design File (ADF) for the decoder and output section of the design, later to be input to A+PLUS where it will be combined with the random sequencer section. These logic equations

```

f0e = ( AO * A1 * A2 * A3 +
        AO * A1 * A2 * A3 +
        /AO * /A1 * /A2 * /A3 );
    * /ROTATE +
    ( AO * /A1 * A2 * A3 +
      /AO * A1 * A2 * A3 +
      /AO * A1 * A2 * A3 +
      /AO * A1 * A2 * A3 +
      /AO * /A1 * A2 * A3 +
      /AO * /A1 * A2 * A3 );
    * ROTATE ;

e0e = ( AO * A1 * A2 * A3 +
        AO * A1 * A2 * A3 +
        /AO * /A1 * A2 * A3 );
    * /ROTATE +
    ( AO * A1 * A2 * A3 +
      /AO * A1 * A2 * A3 +
      /AO * A1 * A2 * A3 +
      /AO * A1 * A2 * A3 +
      /AO * /A1 * A2 * A3 +
      /AO * /A1 * A2 * A3 );
    * ROTATE ;

d0e = ( AO * A1 * A2 * A3 +
        /AO * /A1 * A2 * A3 +
        AO * /A1 * A2 * A3 +
        AO * /A1 * A2 * A3 +
        /AO * A1 * A2 * A3 +
        /AO * A1 * A2 * A3 +
        /AO * /A1 * A2 * A3 +
        /AO * /A1 * A2 * A3 );
    * /ROTATE +
    ( AO * A1 * A2 * A3 +
      AO * A1 * A2 * A3 +
      /AO * A1 * A2 * A3 +
      /AO * A1 * A2 * A3 );
    * ROTATE ;

c0e = ( AO * /A1 * A2 * A3 +
        /AO * A1 * A2 * A3 +
        /AO * A1 * A2 * A3 +
        /AO * A1 * A2 * A3 +
        /AO * /A1 * A2 * A3 +
        /AO * /A1 * A2 * A3 );
    * /ROTATE +
    ( AO * A1 * A2 * A3 +
      /AO * A1 * A2 * A3 +
      /AO * A1 * A2 * A3 +
      /AO * A1 * A2 * A3 +
      /AO * /A1 * A2 * A3 +
      /AO * /A1 * A2 * A3 );
    * ROTATE ;

b0e = ( AO * A1 * A2 * A3 +
        /AO * A1 * A2 * A3 +
        /AO * A1 * A2 * A3 +
        /AO * A1 * A2 * A3 +
        /AO * /A1 * A2 * A3 +
        /AO * /A1 * A2 * A3 );
    * /ROTATE +
    ( AO * A1 * A2 * A3 +
      AO * A1 * A2 * A3 +
      AO * A1 * A2 * A3 +
      /AO * A1 * A2 * A3 +
      /AO * A1 * A2 * A3 +
      /AO * A1 * A2 * A3 );
    * ROTATE ;

a0e = ( AO * A1 * A2 * A3 +
        AO * A1 * A2 * A3 +
        AO * /A1 * A2 * A3 +
        AO * /A1 * A2 * A3 +
        /AO * A1 * A2 * A3 +
        /AO * A1 * A2 * A3 +
        /AO * A1 * A2 * A3 +
        /AO * A1 * A2 * A3 );
    * /ROTATE +
    ( AO * A1 * A2 * A3 +
      AO * A1 * A2 * A3 +
      /AO * A1 * A2 * A3 +
      /AO * A1 * A2 * A3 +
      /AO * A1 * A2 * A3 +
      /AO * A1 * A2 * A3 +
      /AO * A1 * A2 * A3 +
      /AO * A1 * A2 * A3 );
    * ROTATE ;

c1e = ( BO * B1 * B2 * B3 +
        /BO * B1 * B2 * B3 +
        /BO * B1 * B2 * B3 +
        /BO * B1 * B2 * B3 +
        /BO * B1 * B2 * B3 +
        /BO * B1 * B2 * B3 +
        /BO * B1 * B2 * B3 +
        /BO * B1 * B2 * B3 );
    * /ROTATE +
    ( BO * B1 * B2 * B3 +
      /BO * B1 * B2 * B3 +
      /BO * B1 * B2 * B3 +
      /BO * B1 * B2 * B3 +
      /BO * B1 * B2 * B3 +
      /BO * B1 * B2 * B3 +
      /BO * B1 * B2 * B3 +
      /BO * B1 * B2 * B3 );
    * ROTATE ;

b1e = ( BO * B1 * B2 * B3 +
        /BO * B1 * B2 * B3 +
        /BO * B1 * B2 * B3 +
        /BO * B1 * B2 * B3 +
        /BO * B1 * B2 * B3 +
        /BO * B1 * B2 * B3 +
        /BO * B1 * B2 * B3 +
        /BO * B1 * B2 * B3 );
    * /ROTATE +
    ( BO * B1 * B2 * B3 +
      /BO * B1 * B2 * B3 +
      /BO * B1 * B2 * B3 +
      /BO * B1 * B2 * B3 +
      /BO * B1 * B2 * B3 +
      /BO * B1 * B2 * B3 +
      /BO * B1 * B2 * B3 +
      /BO * B1 * B2 * B3 );
    * ROTATE ;

a1e = ( BO * B1 * B2 * B3 +
        BO * B1 * B2 * B3 +
        BO * B1 * B2 * B3 +
        BO * B1 * B2 * B3 +
        /BO * B1 * B2 * B3 +
        /BO * B1 * B2 * B3 +
        /BO * B1 * B2 * B3 +
        /BO * B1 * B2 * B3 );
    * /ROTATE +
    ( BO * B1 * B2 * B3 +
      BO * B1 * B2 * B3 +
      BO * B1 * B2 * B3 +
      BO * B1 * B2 * B3 +
      /BO * B1 * B2 * B3 +
      /BO * B1 * B2 * B3 +
      /BO * B1 * B2 * B3 +
      /BO * B1 * B2 * B3 );
    * ROTATE ;

g2e = /CO * C1 + CO * C1 ;
a2e = /CO * /C1 + CO * C1 ;
b2e = /CO * /C1 + /CO * C1 +
        /ROTATE ;
e2e = /CO * /C1 + /CO * C1 +
        ROTATE ;

d1e = ( BO * B1 * B2 * B3 +
        BO * B1 * B2 * B3 +
        BO * B1 * B2 * B3 +
        BO * B1 * B2 * B3 +
        /BO * B1 * B2 * B3 +
        /BO * B1 * B2 * B3 +
        /BO * B1 * B2 * B3 +
        /BO * B1 * B2 * B3 );
    * /ROTATE +
    ( BO * B1 * B2 * B3 +
      BO * B1 * B2 * B3 +
      BO * B1 * B2 * B3 +
      BO * B1 * B2 * B3 +
      /BO * B1 * B2 * B3 +
      /BO * B1 * B2 * B3 +
      /BO * B1 * B2 * B3 +
      /BO * B1 * B2 * B3 );
    * ROTATE ;
    
```

were derived directly from truth tables describing the relations of outputs to inputs. To receive a copy of these equations send a blank 5¼ inch floppy disk to Altera's applications department.

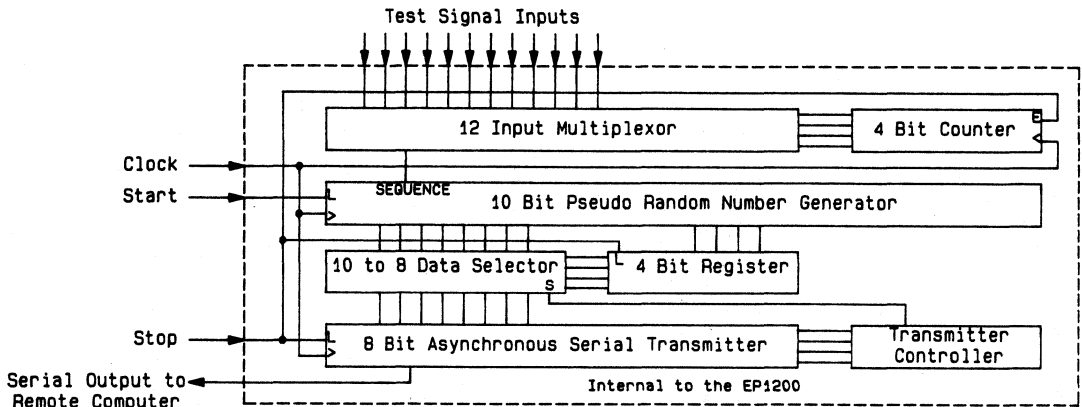


Fig. 6.

Replace the display decoders and registers with shift register and control logic within the EP1200, and you have a remote signal analysis tool that

could be connected over a serial line to a computer for automatic fault detection.



## AB 6 EP1200 as a 6502/6800/68000 Peripheral

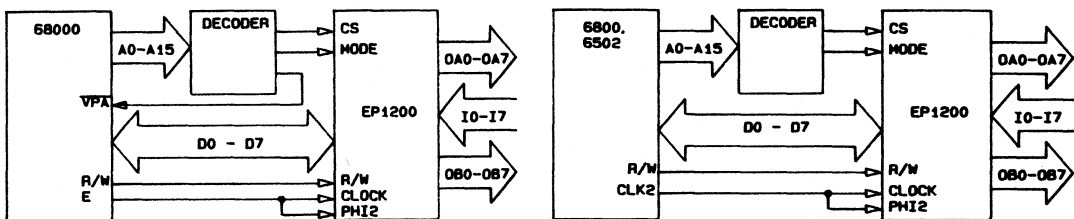
- 6502, 6800, 68000 Peripheral
- FAST - 15 MHz
- Adaptable to suit user need

The EP1200/EP1210 can be programmed to make a highly adaptable peripheral for 6502, 6800, and 68000 microprocessor systems. The block diagram shown below illustrates the overall logic function of a simple peripheral created from an EP1200. In this case the device is used as a single 8-bit input port (I0-I7) with two latched output ports, OA0-OA7 and OB0-OB7. A single control signal (MODE) selects port A or B for data output while (R/W), (PHI2), and chip select (CS) signals are used to read the input port and address the chip.

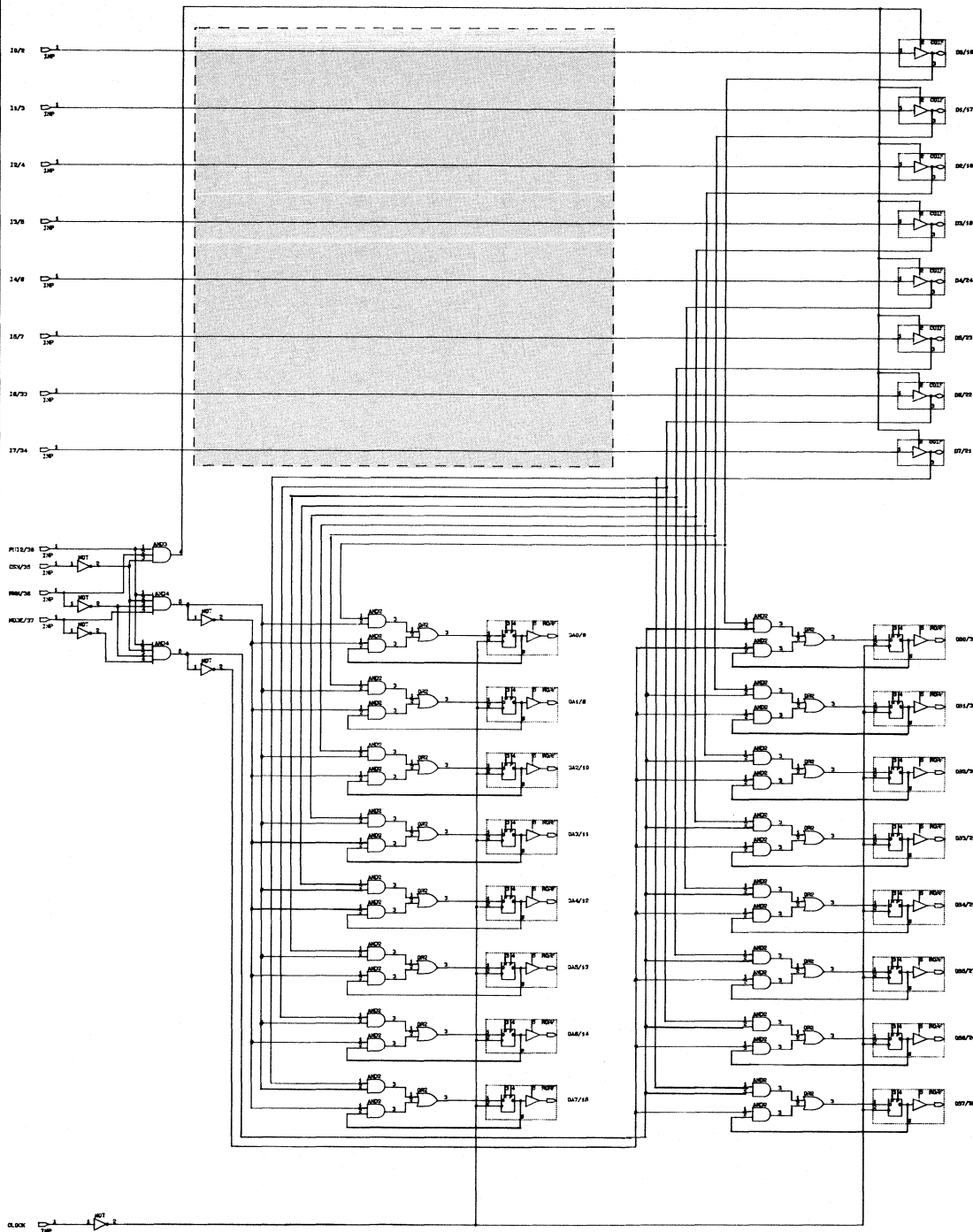
The application shown here has not been customized for any specific task, but with simple modifications the detailed design (shown on the following page) can be modified in a number of ways to suit your needs. The areas that can be readily modified include:

- Custom input logic — the dashed area could include gating logic of the input signals to meet the needs of the application.
- Custom output logic — the simple bus interface can be enhanced with a custom logic function controlled by input signals and data loaded by the microprocessor.
- Interrupt control outputs could be added including 'open-collector' drive for wired-or active low interrupt request signals.

If you have the need for a microprocessor peripheral that involves additional logic, then review the EP1200/EP1210. It can replace your 6820 and integrate the control logic and peripheral tasks all into a single chip, saving valuable board real-estate and system cost.



PHI2	CS	R/W	MODE	OA0 - OA7	OB0 - OB7	INPUT	OUTPUT
0	X	X	X	HOLD DATA	HOLD DATA	D0 - D7 (3-STATED)	-
X	1	X	X	HOLD DATA	HOLD DATA	D0 - D7 (3-STATED)	-
1	0	1	X	HOLD DATA	HOLD DATA	I0 - I7	D0 - D7
1	0	0	0	HOLD DATA	D0 - D7	D0 - D7	OB0 - OB7
1	0	0	1	D0 - D7	HOLD DATA	D0 - D7	OA0 - OA7



- 8085, 8086 Peripheral
- FAST - 15 MHz
- Internal decoding of DA0-DA7
- Adaptable to suit user need

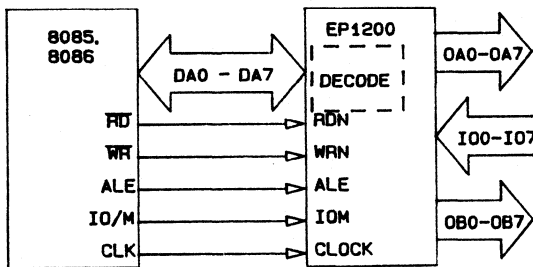
The EP1200/EP1210 can be programmed to make a highly adaptable peripheral for 8085 and 8086 micro-processor systems. The block diagram shown below illustrates the overall logic function of a simple peripheral created from an EP1200/EP1210. In this case the device is used as a single 8-bit input port (IO-17) with two latched output ports, OA0-OA7 and OB0-OB7. Decoding within the EP1200/EP1210 of the DA0-DA7, ALE, RD, WR, and IO/M signals addresses the chip and either selects port A or port B for data output or reads data from the input port (IO-17).

The application shown here has not been customized for any specific task, but with simple modifications the detailed design (shown on the following page) can be modified in a number of ways to suit your needs. The areas that can be readily modified include:

- Custom input logic — the large dashed area could include gating logic of the input signals to meet the needs of the application.

- Custom output logic — the simple bus interface can be enhanced with a custom logic function controlled by input signals and data loaded by the micro-processor.
- Custom address decoding — the three small dashed areas can be modified to decode the DA0-DA7 lines as required by the application.
- Interrupt control outputs could be added including 'open-collector' drive for wired-or active low interrupt request signals.

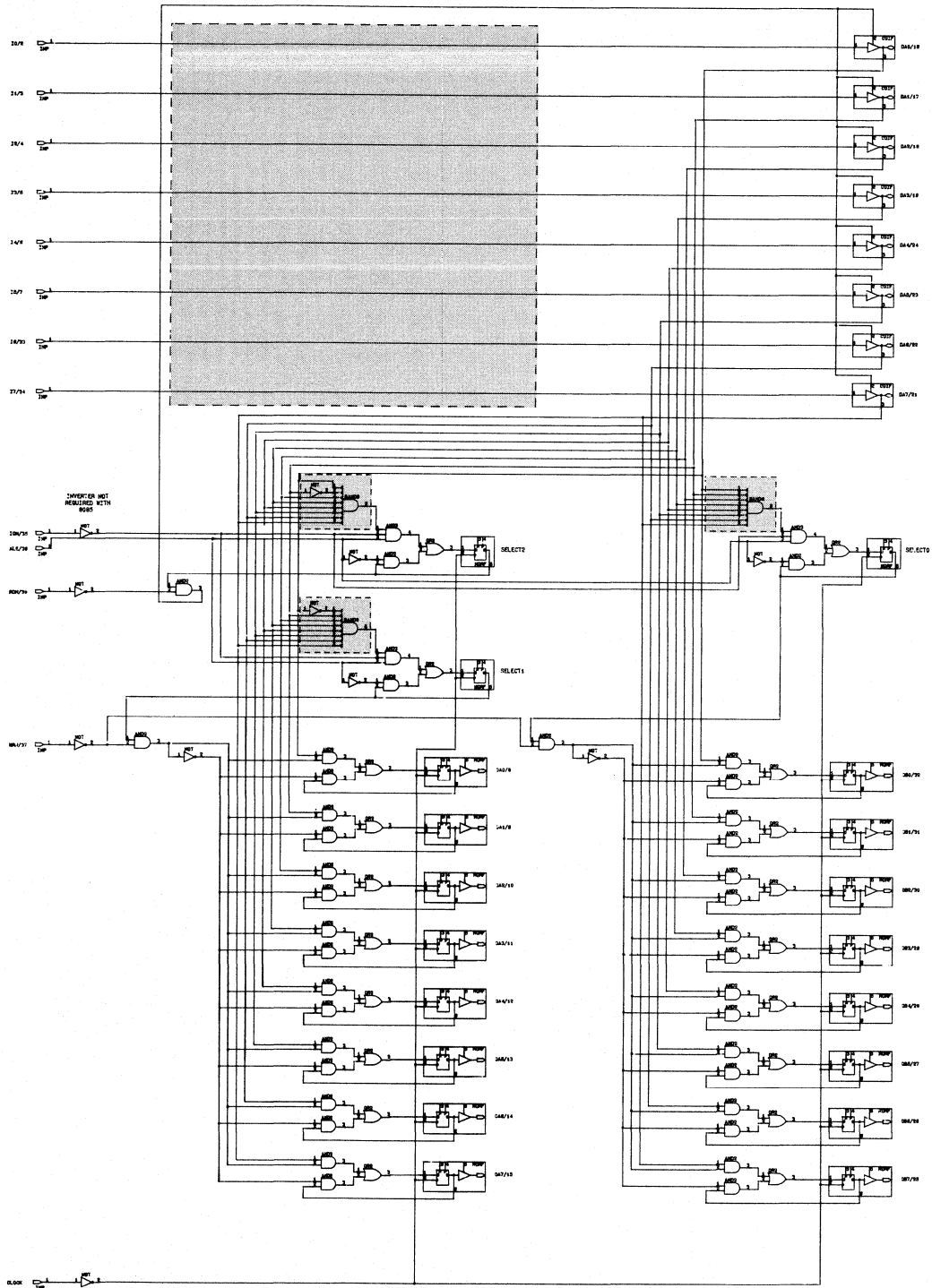
If you have the need for a microprocessor peripheral that involves additional logic, then review the EP1200/EP1210. It can replace your 8255 and integrate decoding, control logic and peripheral tasks all into a single chip, saving valuable board real-estate and system cost.



INTERNAL SIGNALS					
ALE	IOM*	DA0 - DA7	SELECT0	SELECT1	SELECT2
1	1	00000000	1	0	0
1	1	00000001	0	1	0
1	1	00000010	0	0	0
0	X	X	HOLD	HOLD	HOLD
1	0	X	0	0	0

\*IOM SIGNAL INVERTED FOR 8086

SELECT0	SELECT1	SELECT2	RDN	WRN	OA0 - OA7	OB0 - OB7	INPUT	OUTPUT
1	0	0	1	0	HOLD DATA	DO - D7	DO - D7	OB0 - OB7
1	0	0	0	1	HOLD DATA	HOLD DATA	-	-
0	1	0	1	0	DO - D7	HOLD DATA	DO - D7	OA0 - OA7
0	1	0	0	1	HOLD DATA	HOLD DATA	-	-
0	0	1	0	1	HOLD DATA	HOLD DATA	IO - I7	DO - D7
0	0	1	1	0	HOLD DATA	HOLD DATA	-	-
0	0	0	X	X	HOLD DATA	HOLD DATA	-	-



## AB 8 Efficient Counter Design with Toggle Flip-Flops

Toggle flip-flops are the answer to simple and efficient counter designs. To see the advantages offered by T flip-flops, you need only compare Figure 1, which shows an 8-bit binary counter constructed with T flip-flops, to Figure 2, which shows the same counter built with D-type flip-flops. Note that the counter in Figure 2 requires one additional product term for each successive significant bit; thus the most significant bit, Q7, requires nine product terms. In contrast, each bit of the counter created with T flip-flops requires only one product term. Therefore, you can easily construct multi-stage counters without infringing on product term restrictions.

The counter used in this illustration contains an active HIGH count enable input and an active HIGH asynchronous clear input. A HIGH signal applied to the CLEAR input forces all bits of the counter to the LOW state. A LOW signal applied to the ENABLE input inhibits counting, causing each bit of the counter to maintain its present state.

The Altera Design Files (ADFs) for the schematics in Figures 1 and 2 are shown in Figures 3 and 4, respectively.

**FIG. 1.**

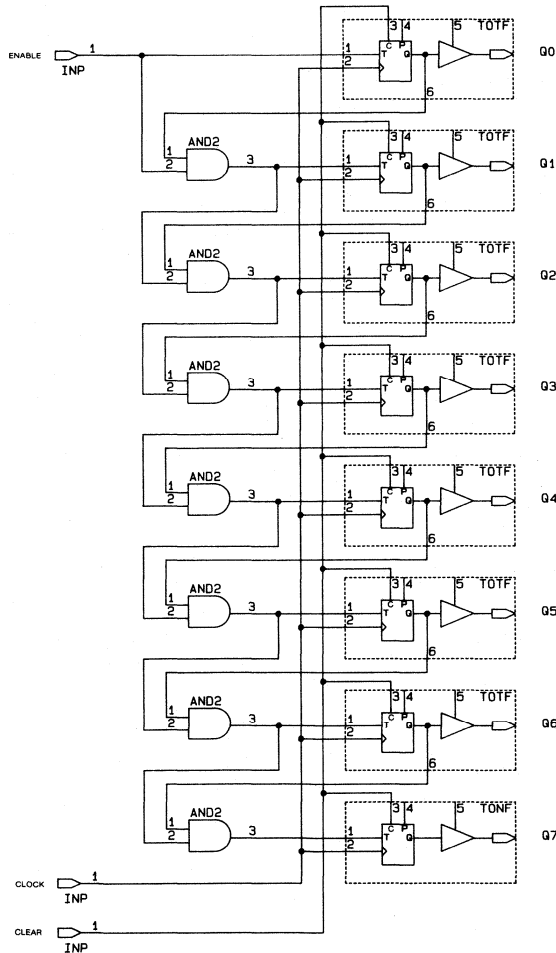


FIG. 2.

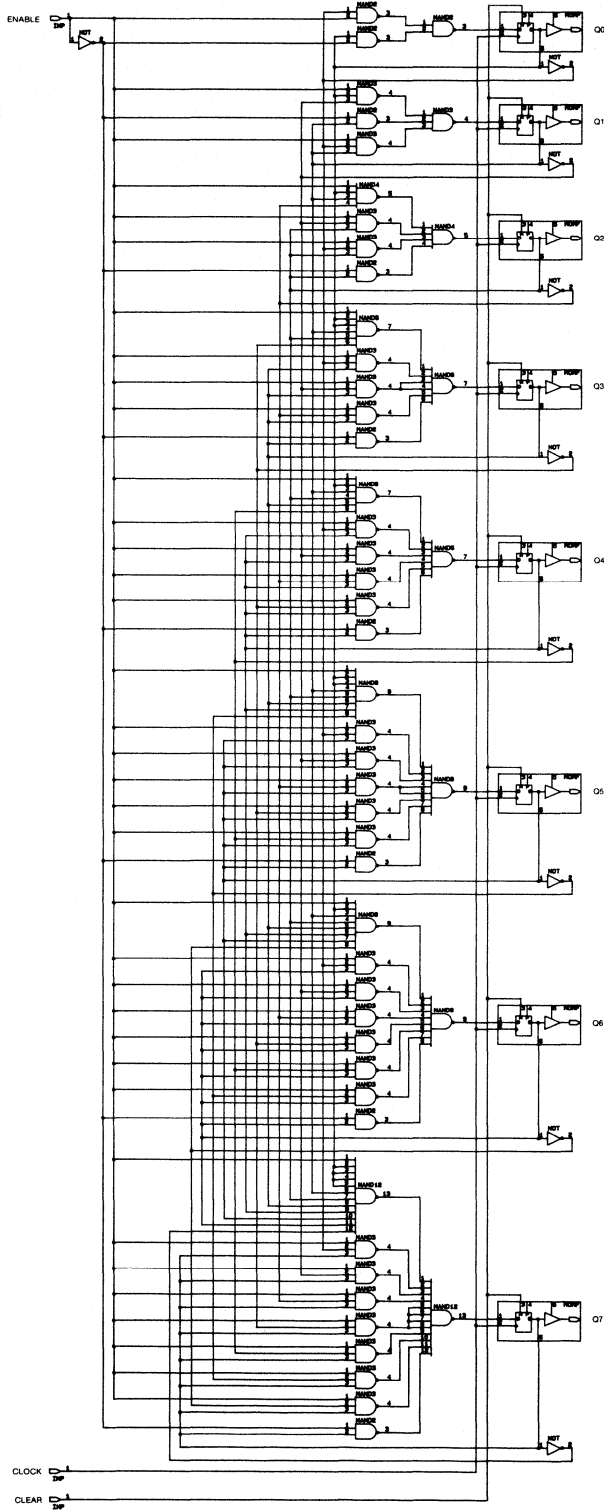


Fig. 3

ALTERA DESIGN FILE

BOB VENABLE
ALTERA CORPORATION
JULY 16, 1985
1.00
A

EP600
8-BIT BINARY COUNTER
PART:

EP600

INPUTS: ENABLEp, CLOCKp, CLEARp

OUTPUTS: Q0p, Q1p, Q2p, Q3p, Q4p, Q5p, Q6p, Q7p

NETWORK:
CLOCK = INP (CLOCKp)
ENABLE = INP (ENABLEp)
CLEAR = INP (CLEARp)
Q0p, Q0f = TOTF ( ENABLE, CLOCK, CLEAR, GND, VCC)
Q1p, Q1f = TOTF ( Q1, CLOCK, CLEAR, GND, VCC)
Q2p, Q2f = TOTF ( Q2, CLOCK, CLEAR, GND, VCC)
Q3p, Q3f = TOTF ( Q3, CLOCK, CLEAR, GND, VCC)
Q4p, Q4f = TOTF ( Q4, CLOCK, CLEAR, GND, VCC)
Q5p, Q5f = TOTF ( Q5, CLOCK, CLEAR, GND, VCC)
Q6p, Q6f = TOTF ( Q6, CLOCK, CLEAR, GND, VCC)
Q7p, Q7f = TOTF ( Q7, CLOCK, CLEAR, GND, VCC)

EQUATIONS:
Q1 = Q0f \* ENABLE;
Q2 = Q1f \* Q0f \* ENABLE;
Q3 = Q2f \* Q1f \* Q0f \* ENABLE;
Q4 = Q3f \* Q2f \* Q1f \* Q0f \* ENABLE;
Q5 = Q4f \* Q3f \* Q2f \* Q1f \* Q0f \* ENABLE;
Q6 = Q5f \* Q4f \* Q3f \* Q2f \* Q1f \* Q0f \* ENABLE;
Q7 = Q6f \* Q5f \* Q4f \* Q3f \* Q2f \* Q1f \* Q0f \* ENABLE;

END\$

Fig. 4

ALTERA DESIGN FILE

BOB VENABLE
ALTERA CORPORATION
JULY 16, 1985
1.00
A

EP1210
8-BIT BINARY COUNTER
PART:

EP1210

INPUTS: ENABLEp, CLOCKp, CLEARp

OUTPUTS: Q0p, Q1p, Q2p, Q3p, Q4p, Q5p, Q6p, Q7p

NETWORK:
CLOCK = INP (CLOCKp)
ENABLE = INP (ENABLEp)
CLEAR = INP (CLEARp)
Q0p, Q0f = RORF ( Q0, CLOCK, CLEAR, GND, VCC)
Q1p, Q1f = RORF ( Q1, CLOCK, CLEAR, GND, VCC)
Q2p, Q2f = RORF ( Q2, CLOCK, CLEAR, GND, VCC)
Q3p, Q3f = RORF ( Q3, CLOCK, CLEAR, GND, VCC)
Q4p, Q4f = RORF ( Q4, CLOCK, CLEAR, GND, VCC)
Q5p, Q5f = RORF ( Q5, CLOCK, CLEAR, GND, VCC)
Q6p, Q6f = RORF ( Q6, CLOCK, CLEAR, GND, VCC)
Q7p, Q7f = RORF ( Q7, CLOCK, CLEAR, GND, VCC)

EQUATIONS:
Q7 = /Q0f \* Q7f
+ /Q1f \* Q7f
+ /Q2f \* Q7f
+ /Q3f \* Q7f
+ /Q4f \* Q7f
+ /Q5f \* Q7f
+ /Q6f \* Q7f
+ /ENABLE \* Q7f
+ ENABLE \* Q0f \* Q1f \* Q2f \* Q3f \* Q4f \* Q5f \* Q6f \* /Q7f;
Q6 = /Q0f \* Q6f
+ /Q1f \* Q6f
+ /Q2f \* Q6f
+ /Q3f \* Q6f
+ /Q4f \* Q6f
+ /Q5f \* Q6f
+ /ENABLE \* Q6f
+ ENABLE \* Q0f \* Q1f \* Q2f \* Q3f \* Q4f \* Q5f \* /Q6f;
Q5 = /Q0f \* Q5f
+ /Q1f \* Q5f
+ /Q2f \* Q5f
+ /Q3f \* Q5f
+ /Q4f \* Q5f
+ /ENABLE \* Q5f
+ ENABLE \* Q0f \* Q1f \* Q2f \* Q3f \* Q4f \* /Q5f;
Q4 = /Q0f \* Q4f
+ /Q1f \* Q4f
+ /Q2f \* Q4f
+ /Q3f \* Q4f
+ /ENABLE \* Q4f
+ ENABLE \* Q0f \* Q1f \* Q2f \* Q3f \* /Q4f;
Q3 = /Q0f \* Q3f
+ /Q1f \* Q3f
+ /Q2f \* Q3f
+ /ENABLE \* Q3f
+ ENABLE \* Q0f \* Q1f \* Q2f \* /Q3f;
Q2 = /Q0f \* Q2f
+ /Q1f \* Q2f
+ /ENABLE \* Q2f
+ ENABLE \* Q0f \* Q1f \* /Q2f;
Q1 = /Q0f \* Q1f
+ /ENABLE \* Q1f
+ ENABLE \* Q0f \* /Q1f;
Q0 = /Q0f \* ENABLE
+ Q0f \* /ENABLE;

END\$

- **TRANSPARENT D-TYPE LATCH**  
— 74LS373 Equivalent
- **R-S LATCH**  
— 74LS279 Equivalent
- **PROGRAMMABLE INPUT/OUTPUT POLARITY**
- **THREE-STATE OUTPUT CAPABILITY**

Asynchronous D-type latches commonly implemented with 74LS373 TTL devices hold data at the latch input until a control signal is applied. Once enabled, the output will follow the data input. The symbolic representation and function table for an asynchronous D-type latch is shown below.

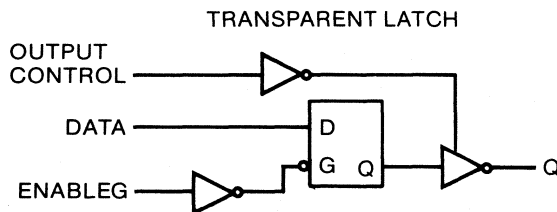
Asynchronous R-S latches are constructed with either cross-coupled NOR or NAND gates. For the NOR-NOR implementation, the output will go to a logical one (HIGH) if the input SET is HIGH. The output will produce a logical zero (LOW) if the RESET input is HIGH. The symbolic representation and function table for an asynchronous R-S latch is also shown below. This circuit is equivalent to a 74LS279 TTL device.

The logic implementation for each latch using

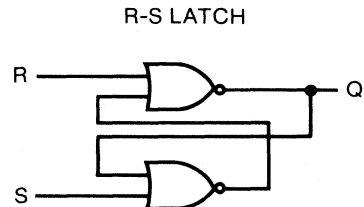
Altera Symbol Primitives is given in Figure 2. To design either latch into an EPLD requires just one Macrocell using combinatorial feedback.

Altera EPLDs offer increased flexibility for specifying the latch operation. For the transparent latch, the output polarity is active high (Q=DATA when ENABLEG=HIGH), when the DATA input pin drives the inverter (NOT) gate as shown in Figure 2. For an active low output (Q=/DATA when ENABLEG=HIGH), remove the inverter gate and connect the DATA input pin directly to the AND gate. In addition, the OUTPUT CONTROL and ENABLEG inputs can also be defined active high or low. For example, connecting an inverter gate after the ENABLEG input pin will cause the circuit to pass DATA when ENABLEG=LOW and latch DATA when ENABLEG=HIGH.

**FIG. 1**



OUTPUT CONTROL	ENABLE G	DATA	OUTPUT Q
L	H	H	H
L	H	L	L
L	L	X	Q
H	X	X	Z



S	R	Q
L	L	Q
L	H	L
H	L	H



The output polarity for the R-S latch can also be designed active high or low. Replacing the NOR gate which drives the COCF I/O primitive with an OR gate gives an active low output.

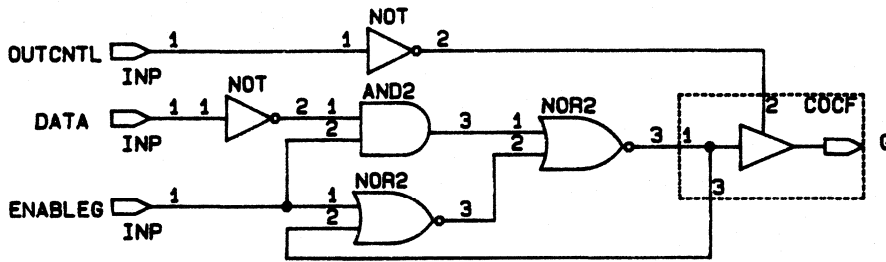
For the Altera EPLDs whose I/O's do not support direct combinatorial feedback (EP600, EP900, and EP1200) Input feedback may be substituted to route the

signal back into the AND array.

For example, the SR-LATCH in Figure 3 uses a COIF I/O primitive in place of the COCF to obtain the feedback path. By using a COIF primitive the feedback delay slightly increases. (See App. Note #4 for details). This SR-LATCH uses cross-coupled NAND gates. The inputs are active low.

FIG. 2

LOGIC DIAGRAM FOR TRANSPARENT LATCH USING ALTERA SYMBOL PRIMITIVES.



LOGIC DIAGRAM FOR R-S LATCH USING ALTERA SYMBOL PRIMITIVES.

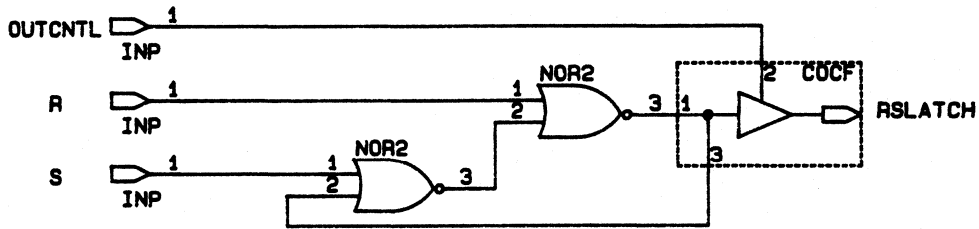
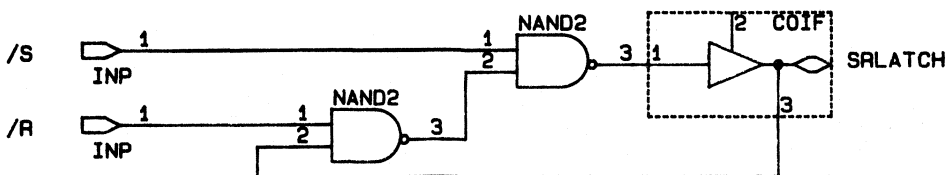


FIG. 3

THE COIF I/O PRIMITIVE MAY BE USED TO OBTAIN COMBINATORIAL FEEDBACK IN AN EPLD.



**ALTERA DESIGN FILE — TRANSPARENT LATCH**

DON FARIA  
 ALTERA  
 07/29/1985  
 1.00  
 EP310  
 TRANSPARENT D-TYPE LATCH

PART: EP310  
 INPUTS: DATA, ENABLEG, OUTCNTL  
 OUTPUTS: Q

NETWORK:

DATA = INP(DATA)  
 ENABLEG = INP(ENABLEG)  
 OUTCNTL = INP(OUTCNTL)  
 Q,Q = COCF(Qc,OE)

EQUATIONS:

$Qc = \text{NOR}(\text{DATA} * \text{ENABLEG}, \text{ENABLEG} + Q)$ ;  
 OE = /OUTCNTL;

END\$

**ALTERA DESIGN FILE — R-S LATCH**

DON FARIA  
 ALTERA  
 07/29/1985  
 1.00  
 EP310  
 RS LATCH

PART: EP310  
 INPUTS: OUTCNTL, R, S  
 OUTPUTS: RSLATCH

NETWORK:

OUTCNTL = INP(OUTCNTL)  
 R = INP(R)  
 S = INP(S)  
 RSLATCH,Q = COCF(RSLATCHc, OE)

EQUATIONS:

$RSLATCHc = \text{NOR}(S + Q, R)$ ;      % NOR IMPLEMENTATION %  
 OE = OUTCNTL;      % OUTPUT ENABLE FOR RSLATCH %

END\$

- 8 BIT UP COUNTER WITH PARALLEL LOAD
- 8 BIT UP/DOWN COUNTER WITH PARALLEL LOAD
- 26 BIT UP COUNTER
- CARRY/BORROW LOOK AHEAD CIRCUITS
- HOLD COUNT CONTROL
- TOGGLE FLIP-FLOPS

Synchronous counters in a variety of sizes and functional capability can be implemented in Altera EPLDs. The following presents three binary counter designs using the EP1200/EP1210: an octal up counter, an octal up/down counter, each with parallel load and count hold functions, and a 26 bit up counter. The function table, schematic representation, and Altera Design File are included.

Each of the counters utilize Toggle flipflops in place of the commonly used D-type flipflop. By using the T-flipflop, the number of product terms required for each counter bit is significantly reduced. This allows large synchronous counters to be implemented in the EP1200/EP1210. A T-flipflop is constructed from a D-flipflop by using an XOR gate as shown in Figure 1.

The Function Table describing the operation of each counter is given below. All three counters are synchronous with transitions occurring on the clock rising edge.

**FUNCTION TABLE**

COUNTER TYPE	ENABLE	LOAD	UPDN	OPERATION
8 BIT UP	X	H	-	LOAD DATA
	L	L	-	HOLD COUNT
	H	L	-	COUNT UP
8 BIT UP/DOWN	X	H	X	LOAD DATA
	L	L	X	HOLD COUNT
	H	L	H	COUNT UP
	H	L	H	COUNT DOWN
26 BIT UP	L	-	-	HOLD COUNT
	H	-	-	COUNT UP

The 8 bit binary up counter features parallel load and count enable. The two inputs ENABLE and LOAD control the circuit operation. If ENABLE=HIGH the counter will count up. If ENABLE=LOW the counter will stop count and hold its present value. If LOAD=HIGH then the data inputs (D1-D7) are loaded into the counter registers.

The 8 bit up/down binary counter adds an additional input. If UPDN=HIGH this circuit will count up. If UPDN=LOW then it will count down. The circuit also

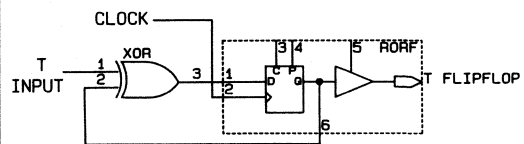
uses a carry/borrow look ahead stage. This stage minimizes the number of product terms required by the upper bits. Without this stage the product term requirements for the two most significant bits would exceed the EP1200/EP1210 specifications. Note, on power up the EP1200/EP1210 registers are cleared to a logical zero. If counting down, the counter must first be loaded with its initial value.

The largest counter the EP1200/EP1210 can support is a 26 bit up counter. Because of the EP1200/EP1210 maximum I/O of 24 outputs, the two least significant bits (Q1 and Q2) are located in the EP1200/EP1210 buried registers. This circuit employs two remaining buried registers. Because the EP1200/EP1210 Macrocells are partitioned into local and global feedback sectors, the 26 bit counter is achieved by constructing two ten bit counters and one six bit counter. The carry look ahead circuits connect the individual counters to form a synchronous 26 bit counter.

Other functions such as asynchronous clear and output enable can also be added to the counters. Each EPLD Macrocell provides additional product terms for these functions. For example, the counter may be

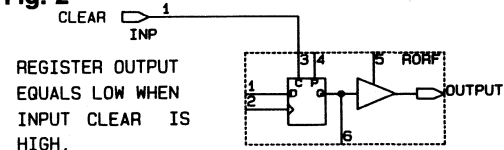
**FIG. 1**

T-FLIPFLOPS CAN BE CONSTRUCTED FROM D-TYPE FLIPFLOPS BY USING AN XOR GATE.

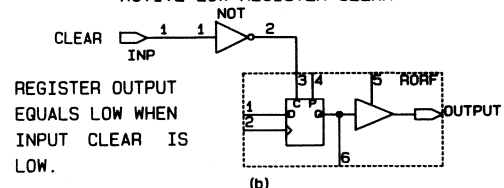


**Fig. 2**

ACTIVE HIGH REGISTER CLEAR



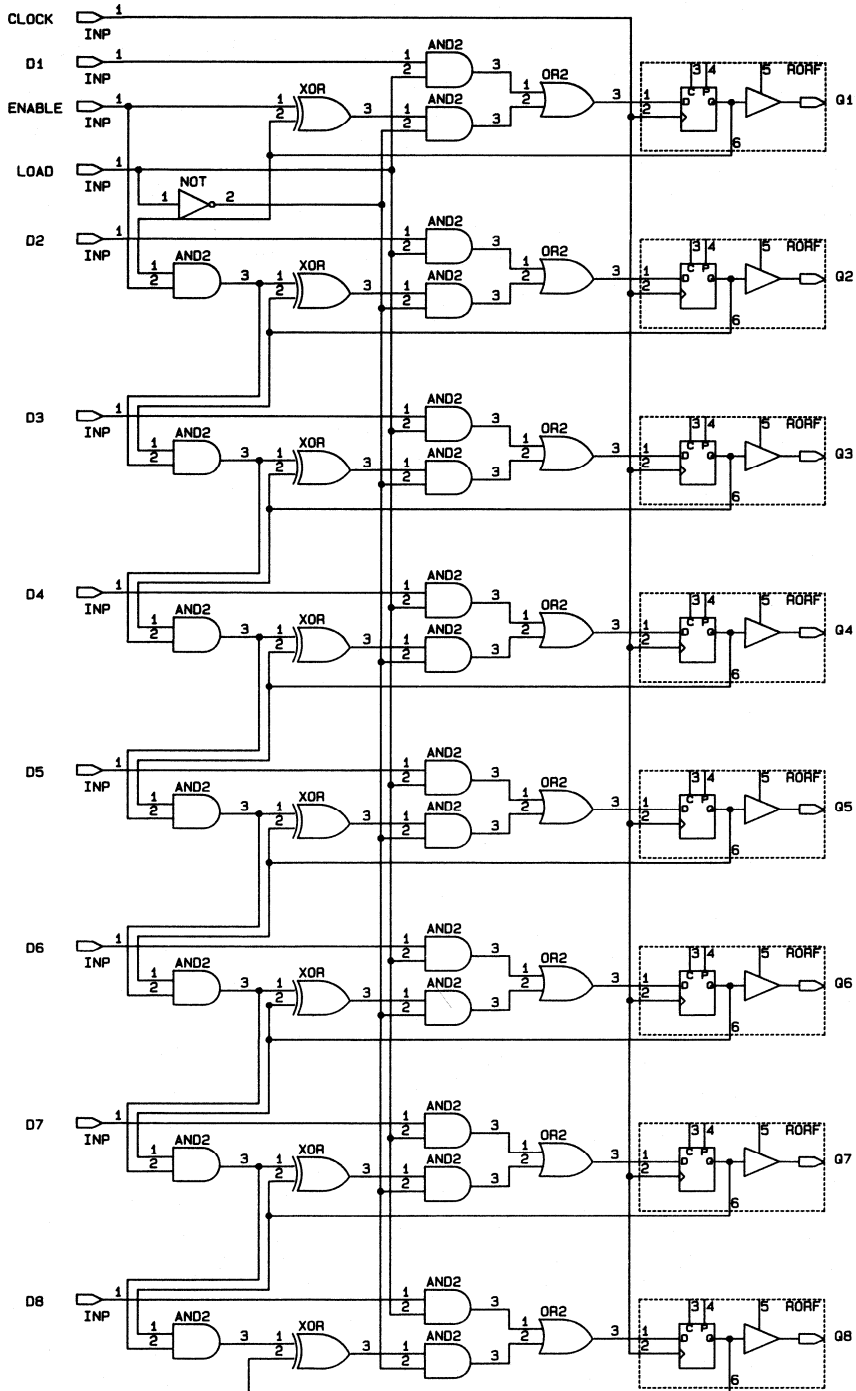
ACTIVE LOW REGISTER CLEAR



asynchronously cleared (all outputs go LOW independent of the clock), when the CLEAR input of the

register is HIGH. Figure 2 shows active high and active low clear functions.

FIG. 3 8 BIT UP COUNTER



DON FARIA  
 ALTERA  
 08/02/1985  
 REV 1.0  
 EP1210  
 8 BIT UP COUNTER

PART:EP1210  
 INPUTS: CLOCK,D1,D2,D3,D4,D5,D6,D7,D8,LOAD,ENABLE  
 OUTPUTS: Q1,Q2,Q3,Q4,Q5,Q6,Q7,Q8

NETWORK:

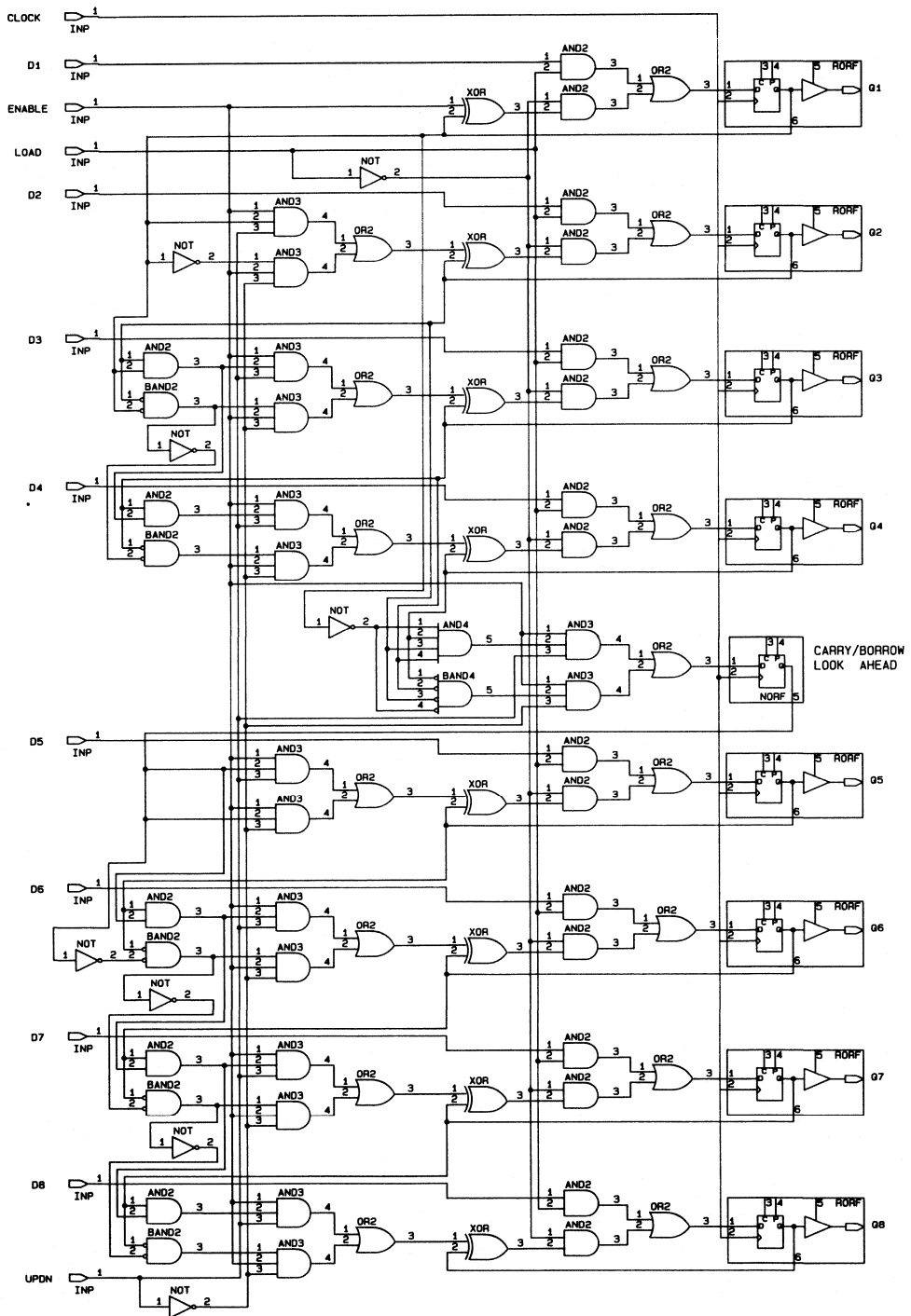
CLOCK = INP(CLOCK)  
 D1 = INP(D1)  
 D2 = INP(D2)  
 D3 = INP(D3)  
 D4 = INP(D4)  
 D5 = INP(D5)  
 D6 = INP(D6)  
 D7 = INP(D7)  
 D8 = INP(D8)  
 LOAD = INP(LOAD)  
 ENABLE = INP(ENABLE)  
 Q1,Q1 = RORF(Q1d,CLOCK,GND,GND,VCC)  
 Q2,Q2 = RORF(Q2d,CLOCK,GND,GND,VCC)  
 Q3,Q3 = RORF(Q3d,CLOCK,GND,GND,VCC)  
 Q4,Q4 = RORF(Q4d,CLOCK,GND,GND,VCC)  
 Q5,Q5 = RORF(Q5d,CLOCK,GND,GND,VCC)  
 Q6,Q6 = RORF(Q6d,CLOCK,GND,GND,VCC)  
 Q7,Q7 = RORF(Q7d,CLOCK,GND,GND,VCC)  
 Q8,Q8 = RORF(Q8d,CLOCK,GND,GND,VCC)

EQUATIONS:

Q1d = LOAD\*D1 + /LOAD\*(ENABLE\*/Q1+Q1\*/ENABLE);      % LSB %  
 Q2d = LOAD\*D2 + /LOAD\*((ENABLE\*Q1)\*/Q2+Q2\*/(ENABLE\*Q1));  
 Q3d = LOAD\*D3 + /LOAD\*((ENABLE\*Q1\*Q2)\*/Q3+Q3\*/(ENABLE\*Q1\*Q2));  
 Q4d = LOAD\*D4 + /LOAD\*((ENABLE\*Q1\*Q2\*Q3)\*/Q4+Q4\*/(ENABLE\*Q1\*Q2\*Q3));  
 Q5d = LOAD\*D5 + /LOAD\*((ENABLE\*Q1\*Q2\*Q3\*Q4)\*/Q5+Q5\*/(ENABLE\*Q1\*Q2\*Q3\*Q4));  
 Q6d = LOAD\*D6 + /LOAD\*((ENABLE\*Q1\*Q2\*Q3\*Q4\*Q5)\*/Q6  
 +Q6\*/(ENABLE\*Q1\*Q2\*Q3\*Q4\*Q5));  
 Q7d = LOAD\*D7 + /LOAD\*((ENABLE\*Q1\*Q2\*Q3\*Q4\*Q5\*Q6)\*/Q7  
 +Q7\*/(ENABLE\*Q1\*Q2\*Q3\*Q4\*Q5\*Q6));  
 Q8d = LOAD\*D8 + /LOAD\*((ENABLE\*Q1\*Q2\*Q3\*Q4\*Q5\*Q6\*Q7)\*/Q8      % MSB %  
 +Q8\*/(ENABLE\*Q1\*Q2\*Q3\*Q4\*Q5\*Q6\*Q7));

END\$

FIG. 4 8 BIT UP/DOWN COUNTER



DON FARIA

ALTERA

08/02/1985

REV 1.0

EP1210

8 BIT UP/DOWN COUNTER

PART:EP1210

INPUTS:CLOCK,D1,D2,D3,D4,D5,D6,D7,D8,ENABLE,UPDN,LOAD

OUTPUTS:Q1,Q2,Q3,Q4,Q5,Q6,Q7,Q8

NETWORK:

CLOCK = INP(CLOCK)  
 D1 = INP(D1)  
 D2 = INP(D2)  
 D3 = INP(D3)  
 D4 = INP(D4)  
 D5 = INP(D5)  
 D6 = INP(D6)  
 D7 = INP(D7)  
 D8 = INP(D8)  
 ENABLE = INP(ENABLE)  
 UPDN = INP(UPDN)  
 LOAD = INP(LOAD)  
 Q1,Q1 = RORF(Q1d,CLOCK,GND,GND,VCC)  
 Q2,Q2 = RORF(Q2d,CLOCK,GND,GND,VCC)  
 Q3,Q3 = RORF(Q3d,CLOCK,GND,GND,VCC)  
 Q4,Q4 = RORF(Q4d,CLOCK,GND,GND,VCC)  
 Q5,Q5 = RORF(Q5d,CLOCK,GND,GND,VCC)  
 Q6,Q6 = RORF(Q6d,CLOCK,GND,GND,VCC)  
 Q7,Q7 = RORF(Q7d,CLOCK,GND,GND,VCC)  
 Q8,Q8 = RORF(Q8d,CLOCK,GND,GND,VCC)  
 CYA = NORF(CYA d,CLOCK,GND,GND)

EQUATIONS:

$$Q1d = LOAD \cdot D1 + /LOAD \cdot (ENABLE \cdot /Q1 + Q1 \cdot /ENABLE); \quad \% \text{ LSB } \%$$

$$Q2d = LOAD \cdot D2 + /LOAD \cdot (Q2 \cdot / (ENABLE \cdot Q1 \cdot UPDN + ENABLE \cdot /Q1 \cdot /UPDN) + (ENABLE \cdot Q1 \cdot UPDN + ENABLE \cdot /Q1 \cdot /UPDN) \cdot /Q2);$$

$$Q3d = LOAD \cdot D3 + /LOAD \cdot (Q3 \cdot / (ENABLE \cdot Q1 \cdot Q2 \cdot UPDN + ENABLE \cdot /Q1 \cdot /Q2 \cdot /UPDN) + (ENABLE \cdot Q1 \cdot Q2 \cdot UPDN + ENABLE \cdot /Q1 \cdot /Q2 \cdot /UPDN) \cdot /Q3);$$

$$Q4d = LOAD \cdot D4 + /LOAD \cdot (Q4 \cdot / (ENABLE \cdot Q1 \cdot Q2 \cdot Q3 \cdot UPDN + ENABLE \cdot /Q1 \cdot /Q2 \cdot /Q3 \cdot /UPDN) + (ENABLE \cdot Q1 \cdot Q2 \cdot Q3 \cdot UPDN + ENABLE \cdot /Q1 \cdot /Q2 \cdot /Q3 \cdot /UPDN) \cdot /Q4);$$

$$Q5d = LOAD \cdot D5 + /LOAD \cdot (Q5 \cdot / (ENABLE \cdot CYA \cdot UPDN + /UPDN \cdot CYA \cdot ENABLE) + (ENABLE \cdot CYA \cdot UPDN + ENABLE \cdot /CYA \cdot /UPDN) \cdot /Q5);$$

$$Q6d = LOAD \cdot D6 + /LOAD \cdot (Q6 \cdot / (ENABLE \cdot CYA \cdot Q5 \cdot UPDN + ENABLE \cdot CYA \cdot /Q5 \cdot /UPDN) + (ENABLE \cdot CYA \cdot Q5 \cdot UPDN + ENABLE \cdot /CYA \cdot /Q5 \cdot /UPDN) \cdot /Q6);$$

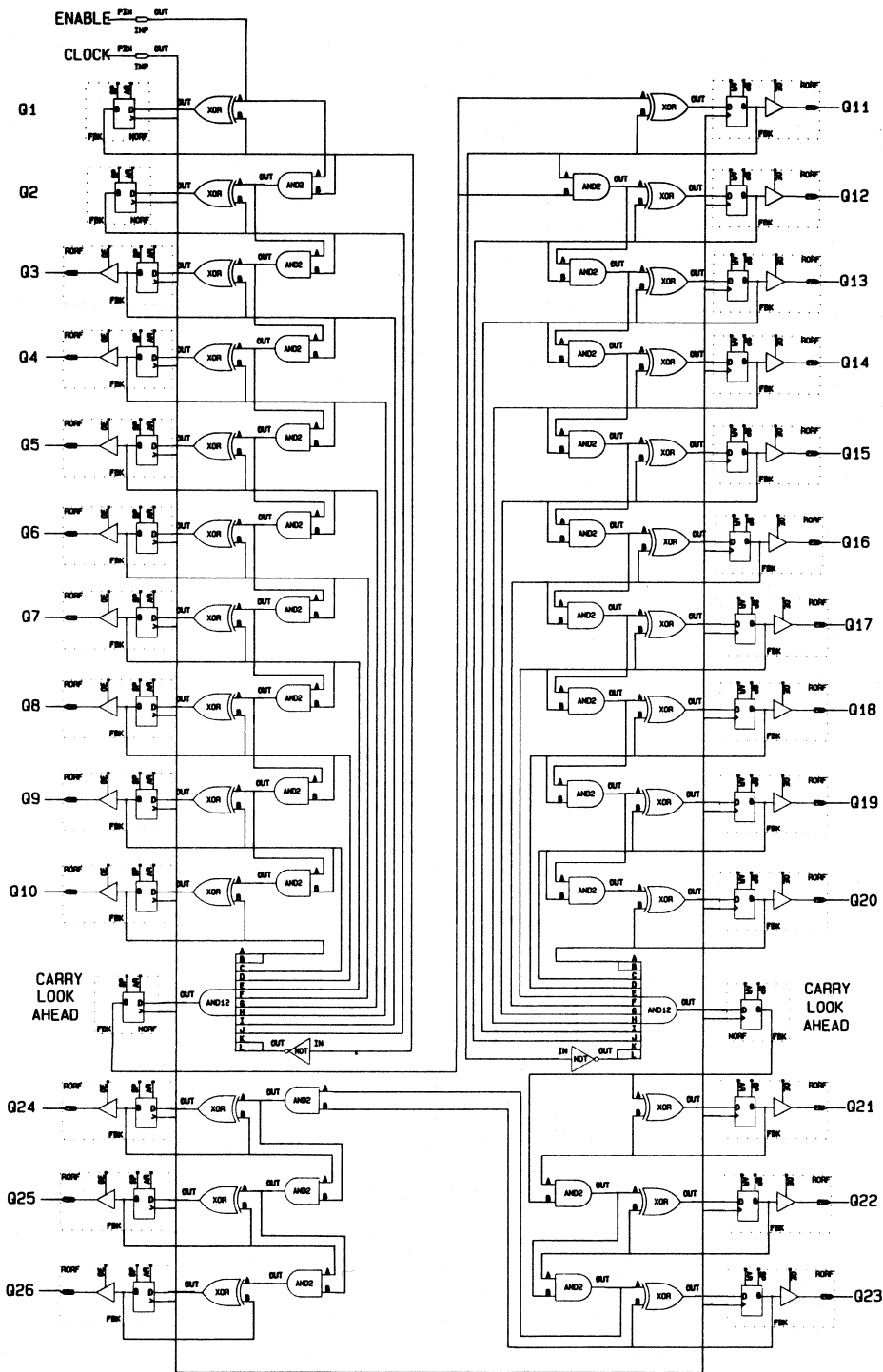
$$Q7d = LOAD \cdot D7 + /LOAD \cdot (Q7 \cdot / (ENABLE \cdot CYA \cdot Q5 \cdot Q6 \cdot UPDN + ENABLE \cdot /CYA \cdot /Q5 \cdot /Q6 \cdot /UPDN) + (ENABLE \cdot CYA \cdot Q5 \cdot Q6 \cdot UPDN + ENABLE \cdot /CYA \cdot /Q5 \cdot /Q6 \cdot /UPDN) \cdot /Q7);$$

$$Q8d = LOAD \cdot D8 + /LOAD \cdot (Q8 \cdot / (ENABLE \cdot CYA \cdot Q5 \cdot Q6 \cdot Q7 \cdot UPDN + ENABLE \cdot /CYA \cdot /Q5 \cdot /Q6 \cdot /Q7 \cdot /UPDN) + (ENABLE \cdot CYA \cdot Q5 \cdot Q6 \cdot Q7 \cdot UPDN + ENABLE \cdot /CYA \cdot /Q5 \cdot /Q6 \cdot /Q7 \cdot /UPDN) \cdot /Q8);$$

$$CYA d = UPDN \cdot ENABLE \cdot /Q1 \cdot Q2 \cdot Q3 \cdot Q4 + /UPDN \cdot ENABLE \cdot Q1 \cdot /Q2 \cdot /Q3 \cdot /Q4;$$

END\$

FIG. 5 26 BIT UP COUNTER





DON FARIA  
 ALTERA CORPORATION  
 26 BIT BINARY COUNTER

THIS 26 BIT BINARY COUNTER IS IMPLEMENTED IN AN EP1210 AS TWO 10 BIT COUNTERS AND ONE 6 BIT COUNTER. TWO BURIED REGISTERS GENERATE LOOK AHEAD CARRIES AND ALLOW COMMUNICATION BETWEEN COUNTERS ON OPPOSITE SIDES OF THE EP1210.

PART: EP1210

INPUTS:ENABLE,CLOCK

OUTPUTS:

% FIRST 10 BIT COUNTER (TWO BITS ARE BURIED) %  
 Q3,Q4,Q5,Q6,Q7,Q8,Q9,Q10,

% SECOND 10 BIT COUNTER %  
 Q11,Q12,Q13,Q14,Q15,Q16,Q17,Q18,Q19,Q20,

% 6 BIT COUNTER %  
 Q21,Q22,Q23,Q24,Q25,Q26

NETWORK:

% INPUT PINS %

CLOCK = INP(CLOCK)

ENABLE = INP(ENABLE)

% FIRST 10 BIT COUNTER %

Q1 = NORF(D1,CLOCK,GND,GND)  
 Q2 = NORF(D2,CLOCK,GND,GND)  
 Q3,Q3 = RORF(D3,CLOCK,GND,GND,VCC)  
 Q4,Q4 = RORF(D4,CLOCK,GND,GND,VCC)  
 Q5,Q5 = RORF(D5,CLOCK,GND,GND,VCC)  
 Q6,Q6 = RORF(D6,CLOCK,GND,GND,VCC)  
 Q7,Q7 = RORF(D7,CLOCK,GND,GND,VCC)  
 Q8,Q8 = RORF(D8,CLOCK,GND,GND,VCC)  
 Q9,Q9 = RORF(D9,CLOCK,GND,GND,VCC)  
 Q10,Q10 = RORF(D10,CLOCK,GND,GND,VCC)  
 CARRY10 = NORF(CY10,CLOCK,GND,GND)

% SECOND 10 BIT COUNTER %

Q11,Q11 = RORF(D11,CLOCK,GND,GND,VCC)  
 Q12,Q12 = RORF(D12,CLOCK,GND,GND,VCC)  
 Q13,Q13 = RORF(D13,CLOCK,GND,GND,VCC)  
 Q14,Q14 = RORF(D14,CLOCK,GND,GND,VCC)  
 Q15,Q15 = RORF(D15,CLOCK,GND,GND,VCC)  
 Q16,Q16 = RORF(D16,CLOCK,GND,GND,VCC)  
 Q17,Q17 = RORF(D17,CLOCK,GND,GND,VCC)  
 Q18,Q18 = RORF(D18,CLOCK,GND,GND,VCC)  
 Q19,Q19 = RORF(D19,CLOCK,GND,GND,VCC)  
 Q20,Q20 = RORF(D20,CLOCK,GND,GND,VCC)  
 CARRY20 = NORF(CY20,CLOCK,GND,GND)

% 6 BIT COUNTER %

Q21,Q21 = RORF(D21,CLOCK,GND,GND,VCC)  
 Q22,Q22 = RORF(D22,CLOCK,GND,GND,VCC)  
 Q23,Q23 = RORF(D23,CLOCK,GND,GND,VCC)  
 Q24,Q24 = RORF(D24,CLOCK,GND,GND,VCC)  
 Q25,Q25 = RORF(D25,CLOCK,GND,GND,VCC)  
 Q26,Q26 = RORF(D26,CLOCK,GND,GND,VCC)

## EQUATIONS:

$$D1 = \text{ENABLE} \cdot /Q1 \\ + / \text{ENABLE} \cdot Q1 ;$$

$$D2 = Q2 \cdot /Q1 \cdot \text{ENABLE} \\ + /Q2 \cdot Q1 \cdot \text{ENABLE} ;$$

$$D3 = Q3 \cdot /Q1 \cdot Q2 \cdot \text{ENABLE} \\ + /Q3 \cdot Q1 \cdot Q2 \cdot \text{ENABLE} ;$$

$$D4 = Q4 \cdot /Q1 \cdot Q2 \cdot Q3 \cdot \text{ENABLE} \\ + /Q4 \cdot Q1 \cdot Q2 \cdot Q3 \cdot \text{ENABLE} ;$$

$$D5 = Q5 \cdot /Q1 \cdot Q2 \cdot Q3 \cdot Q4 \cdot \text{ENABLE} \\ + /Q5 \cdot Q1 \cdot Q2 \cdot Q3 \cdot Q4 \cdot \text{ENABLE} ;$$

$$D6 = Q6 \cdot /Q1 \cdot Q2 \cdot Q3 \cdot Q4 \cdot Q5 \cdot \text{ENABLE} \\ + /Q6 \cdot Q1 \cdot Q2 \cdot Q3 \cdot Q4 \cdot Q5 \cdot \text{ENABLE} ;$$

$$D7 = Q7 \cdot /Q1 \cdot Q2 \cdot Q3 \cdot Q4 \cdot Q5 \cdot Q6 \cdot \text{ENABLE} \\ + /Q7 \cdot Q1 \cdot Q2 \cdot Q3 \cdot Q4 \cdot Q5 \cdot Q6 \cdot \text{ENABLE} ;$$

$$D8 = Q8 \cdot /Q1 \cdot Q2 \cdot Q3 \cdot Q4 \cdot Q5 \cdot Q6 \cdot Q7 \cdot \text{ENABLE} \\ + /Q8 \cdot Q1 \cdot Q2 \cdot Q3 \cdot Q4 \cdot Q5 \cdot Q6 \cdot Q7 \cdot \text{ENABLE} ;$$

$$D9 = Q9 \cdot /Q1 \cdot Q2 \cdot Q3 \cdot Q4 \cdot Q5 \cdot Q6 \cdot Q7 \cdot Q8 \cdot \text{ENABLE} \\ + /Q9 \cdot Q1 \cdot Q2 \cdot Q3 \cdot Q4 \cdot Q5 \cdot Q6 \cdot Q7 \cdot Q8 \cdot \text{ENABLE} ;$$

$$D10 = Q10 \cdot /Q1 \cdot Q2 \cdot Q3 \cdot Q4 \cdot Q5 \cdot Q6 \cdot Q7 \cdot Q8 \cdot Q9 \cdot \text{ENABLE} \\ + /Q10 \cdot Q1 \cdot Q2 \cdot Q3 \cdot Q4 \cdot Q5 \cdot Q6 \cdot Q7 \cdot Q8 \cdot Q9 \cdot \text{ENABLE} ;$$

$$\% \text{ CY10 IS TRUE WHEN BITS Q10 TO Q1} = 1111111110 \% \\ \text{CY10} = \text{ENABLE} \cdot /Q1 \cdot Q2 \cdot Q3 \cdot Q4 \cdot Q5 \cdot Q6 \cdot Q7 \cdot Q8 \cdot Q9 \cdot Q10 ;$$

$$D11 = \text{CARRY10} \cdot /Q11 \\ + / \text{CARRY10} \cdot Q11 ;$$

$$D12 = Q12 \cdot /Q11 \cdot \text{CARRY10} \\ + /Q12 \cdot Q11 \cdot \text{CARRY10} ;$$

$$D13 = Q13 \cdot /Q11 \cdot Q12 \cdot \text{CARRY10} \\ + /Q13 \cdot Q11 \cdot Q12 \cdot \text{CARRY10} ;$$

$$D14 = Q14 \cdot /Q11 \cdot Q12 \cdot Q13 \cdot \text{CARRY10} \\ + /Q14 \cdot Q11 \cdot Q12 \cdot Q13 \cdot \text{CARRY10} ;$$

$$D15 = Q15 \cdot /Q11 \cdot Q12 \cdot Q13 \cdot Q14 \cdot \text{CARRY10} \\ + /Q15 \cdot Q11 \cdot Q12 \cdot Q13 \cdot Q14 \cdot \text{CARRY10} ;$$

$$D16 = Q16 \cdot /Q11 \cdot Q12 \cdot Q13 \cdot Q14 \cdot Q15 \cdot \text{CARRY10} \\ + /Q16 \cdot Q11 \cdot Q12 \cdot Q13 \cdot Q14 \cdot Q15 \cdot \text{CARRY10} ;$$

$$D17 = Q17 \cdot /Q11 \cdot Q12 \cdot Q13 \cdot Q14 \cdot Q15 \cdot Q16 \cdot \text{CARRY10} \\ + /Q17 \cdot Q11 \cdot Q12 \cdot Q13 \cdot Q14 \cdot Q15 \cdot Q16 \cdot \text{CARRY10} ;$$

$$D18 = Q18 \cdot /Q11 \cdot Q12 \cdot Q13 \cdot Q14 \cdot Q15 \cdot Q16 \cdot Q17 \cdot \text{CARRY10} \\ + /Q18 \cdot Q11 \cdot Q12 \cdot Q13 \cdot Q14 \cdot Q15 \cdot Q16 \cdot Q17 \cdot \text{CARRY10} ;$$

$$D19 = Q19 \cdot /Q11 \cdot Q12 \cdot Q13 \cdot Q14 \cdot Q15 \cdot Q16 \cdot Q17 \cdot Q18 \cdot \text{CARRY10} \\ + /Q19 \cdot Q11 \cdot Q12 \cdot Q13 \cdot Q14 \cdot Q15 \cdot Q16 \cdot Q17 \cdot Q18 \cdot \text{CARRY10} ;$$

$$D20 = Q20 \cdot /Q11 \cdot Q12 \cdot Q13 \cdot Q14 \cdot Q15 \cdot Q16 \cdot Q17 \cdot Q18 \cdot Q19 \cdot \text{CARRY10} \\ + /Q20 \cdot Q11 \cdot Q12 \cdot Q13 \cdot Q14 \cdot Q15 \cdot Q16 \cdot Q17 \cdot Q18 \cdot Q19 \cdot \text{CARRY10} ;$$

$$\% \text{ CY20 IS TRUE WHEN BITS Q20 TO Q11} = 1111111110 \% \\ \text{CY20} = \text{CARRY10} \cdot /Q11 \cdot Q12 \cdot Q13 \cdot Q14 \cdot Q15 \cdot Q16 \cdot Q17 \cdot Q18 \cdot Q19 \cdot Q20 ;$$

$$D21 = \text{CARRY20} \cdot /Q21 \\ + / \text{CARRY20} \cdot Q21 ;$$

$$D22 = Q22 \cdot /Q21 \cdot \text{CARRY20} \\ + /Q22 \cdot Q21 \cdot \text{CARRY20} ;$$

$$D23 = Q23 \cdot /Q21 \cdot Q22 \cdot \text{CARRY20} \\ + /Q23 \cdot Q21 \cdot Q22 \cdot \text{CARRY20} ;$$

$$D24 = Q24 \cdot /Q21 \cdot Q22 \cdot Q23 \cdot \text{CARRY20} \\ + /Q24 \cdot Q21 \cdot Q22 \cdot Q23 \cdot \text{CARRY20} ;$$

$$D25 = Q25 \cdot /Q21 \cdot Q22 \cdot Q23 \cdot Q24 \cdot \text{CARRY20} \\ + /Q25 \cdot Q21 \cdot Q22 \cdot Q23 \cdot Q24 \cdot \text{CARRY20} ;$$

$$D26 = Q26 \cdot /Q21 \cdot Q22 \cdot Q23 \cdot Q24 \cdot Q25 \cdot \text{CARRY20} \\ + /Q26 \cdot Q21 \cdot Q22 \cdot Q23 \cdot Q24 \cdot Q25 \cdot \text{CARRY20} ;$$

END\$

## AB 11 16 Bit Up/Down Counter with Left Right Shift Register

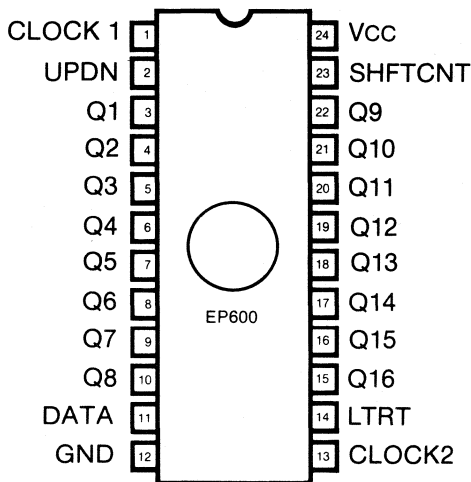
- 16 BIT UP/DOWN LOADABLE COUNTER
- TOGGLE FLIPFLOPS
- 16 BIT LEFT/RIGHT SHIFT REGISTER
- FAST - 30 MHz OPERATION

The 16 bit up/down binary counter with built-in shift register implemented in the EP600 will count up, count down, shift right, or shift left as specified by the three control signals UPDN, SHFTCNT, and LTRT. The two synchronous clock inputs (CLK1 and CLK2) are tied together externally so that all 16 of the EP600's flipflops can be clocked together. The application also takes advantage of the EP600 Toggle flipflops for a more efficient counter design.

The 16 bit shift register acts as a serial load/unload to the counter. In the shift mode, (SHFTCNT=HIGH), input data is shifted into the device on the DATA input. In the case of a left shift, (LTRT=HIGH), data is shifted into the EP600 LSB (Q1) and will shift out of the MSB (Q16). For a right shift, (LTRT=LOW) data will be shifted into the MSB (Q16), and will shift out of the LSB (Q1).

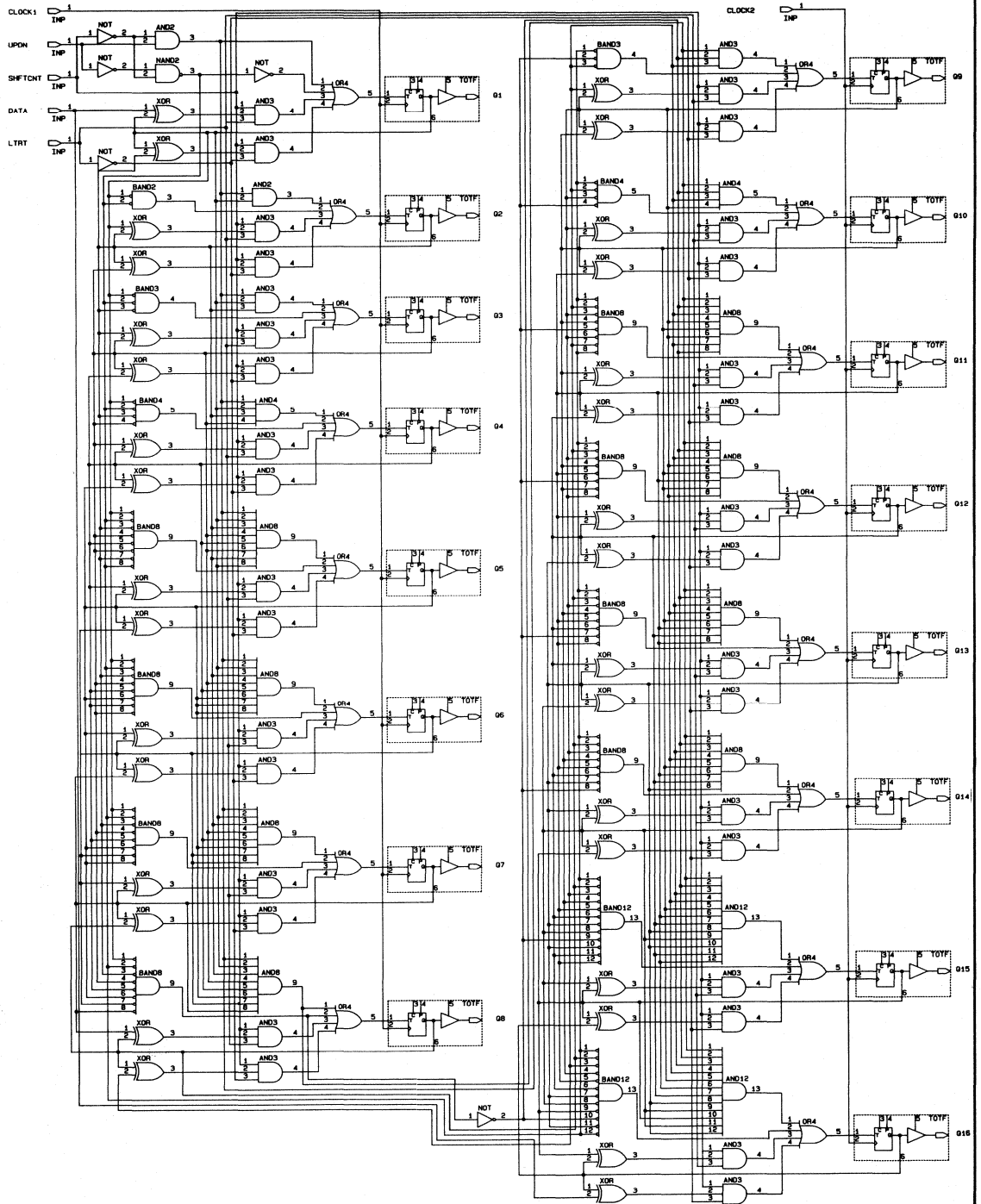
In the count mode, (SHFTCNT=LOW), the direction of the count is controlled by the UPDN input signal. The counter will count up when UPDN=HIGH and count down when UPDN=LOW.

Fig. 1.



FUNCTION TABLE			
SHFTCNT	UPDN	LTRT	OPERATION
HIGH	X	HIGH	SHIFT LEFT
HIGH	X	LOW	SHIFT RIGHT
LOW	HIGH	X	COUNT UP
LOW	LOW	X	COUNT DOWN

FIG. 2. 16 BIT UP/DOWN COUNTER WITH LEFT RIGHT SHIFT REGISTER



DON FARIA  
ALTEA CORPORATION  
07/26/1985

1  
EP600  
16 BIT UP/DOWN COUNTER WITH LEFT/RIGHT SHIFT REGISTER

PART: EP600

INPUTS:CLOCK1,UPDN,DATA,CLOCK2,SHFTCNT,LTRT

OUTPUTS:Q1,Q2,Q3,Q4,Q5,Q6,Q7,Q8,Q9,Q10,Q11,Q12,Q13,Q14,Q15,Q16

NETWORK:

CLOCK1 = INP(CLOCK1)  
 CLOCK2 = INP(CLOCK2)  
 LTRT = INP(LTRT)  
 DATA = INP(DATA)  
 SHFTCNT = INP(SHFTCNT)  
 UPDN = INP(UPDN)  
 Q1,Q1 = TOTF(Q1t,CLOCK1,GND,GND,VCC)  
 Q2,Q2 = TOTF(Q2t,CLOCK1,GND,GND,VCC)  
 Q3,Q3 = TOTF(Q3t,CLOCK1,GND,GND,VCC)  
 Q4,Q4 = TOTF(Q4t,CLOCK1,GND,GND,VCC)  
 Q5,Q5 = TOTF(Q5t,CLOCK1,GND,GND,VCC)  
 Q6,Q6 = TOTF(Q6t,CLOCK1,GND,GND,VCC)  
 Q7,Q7 = TOTF(Q7t,CLOCK1,GND,GND,VCC)  
 Q8,Q8 = TOTF(Q8t,CLOCK1,GND,GND,VCC)  
 Q9,Q9 = TOTF(Q9t,CLOCK2,GND,GND,VCC)  
 Q10,Q10 = TOTF(Q10t,CLOCK2,GND,GND,VCC)  
 Q11,Q11 = TOTF(Q11t,CLOCK2,GND,GND,VCC)  
 Q12,Q12 = TOTF(Q12t,CLOCK2,GND,GND,VCC)  
 Q13,Q13 = TOTF(Q13t,CLOCK2,GND,GND,VCC)  
 Q14,Q14 = TOTF(Q14t,CLOCK2,GND,GND,VCC)  
 Q15,Q15 = TOTF(Q15t,CLOCK2,GND,GND,VCC)  
 Q16,Q16 = TOTF(Q16t,CLOCK2,GND,GND,VCC)

EQUATIONS:

$$Q1t = /SHFTCNT*(UPDN + /UPDN) \quad \% \text{ LEAST SIGNIFICANT BIT } \% \\ + SHFTCNT*(LTRT*(DATA*/Q1 + Q1*/DATA) + /LTRT*(Q1*/Q2 + Q2*/Q1));$$

$$Q2t = /SHFTCNT*(UPDN*Q1 + /UPDN*/Q1) \\ + SHFTCNT*(LTRT*(Q1*/Q2 + Q2*/Q1) + /LTRT*(Q2*/Q3 + Q3*/Q2));$$

$$Q3t = /SHFTCNT*(UPDN*Q1*Q2 + /UPDN*/Q1*/Q2) \\ + SHFTCNT*(LTRT*(Q2*/Q3 + Q3*/Q2) + /LTRT*(Q3*/Q4 + Q4*/Q3));$$

$$Q4t = /SHFTCNT*(UPDN*Q1*Q2*Q3 + /UPDN*/Q1*/Q2*/Q3) \\ + SHFTCNT*(LTRT*(Q3*/Q4 + Q4*/Q3) + /LTRT*(Q4*/Q5 + Q5*/Q4));$$

$$Q5t = /SHFTCNT*(UPDN*Q1*Q2*Q3*Q4 + /UPDN*/Q1*/Q2*/Q3*/Q4) \\ + SHFTCNT*(LTRT*(Q4*/Q5 + Q5*/Q4) + /LTRT*(Q5*/Q6 + Q6*/Q5));$$

$$Q6t = /SHFTCNT*(UPDN*Q1*Q2*Q3*Q4*Q5 + /UPDN*/Q1*/Q2*/Q3*/Q4*/Q5) \\ + SHFTCNT*(LTRT*(Q5*/Q6 + Q6*/Q5) + /LTRT*(Q6*/Q7+Q7*/Q6));$$

$$Q7t = /SHFTCNT*(UPDN*Q1*Q2*Q3*Q4*Q5*Q6 + /UPDN*/Q1*/Q2*/Q3*/Q4*/Q5*/Q6) \\ + SHFTCNT*(LTRT*(Q6*/Q7 + Q7*/Q6) + /LTRT*(Q7*/Q8 + Q8*/Q7));$$

$$Q8t = /SHFTCNT*(UPDN*Q1*Q2*Q3*Q4*Q5*Q6*Q7$$

$$+ /UPDN*/Q1*/Q2*/Q3*/Q4*/Q5*/Q6*/Q7)$$

$$+ SHFTCNT*(LTRT*(Q7*/Q8 + Q8*/Q7) + /LTRT*(Q8*/Q9 + Q9*/Q8));$$

$$Q9t = /SHFTCNT*(UPDN*Q1*Q2*Q3*Q4*Q5*Q6*Q7*Q8$$

$$+ /UPDN*/Q1*/Q2*/Q3*/Q4*/Q5*/Q6*/Q7*/Q8)$$

$$+ SHFTCNT*(LTRT*(Q8*/Q9 + Q9*/Q8) + /LTRT*(Q9*/Q10 + Q10*/Q9));$$

$$Q10t = /SHFTCNT*(UPDN*Q1*Q2*Q3*Q4*Q5*Q6*Q7*Q8*Q9$$

$$+ /UPDN*/Q1*/Q2*/Q3*/Q4*/Q5*/Q6*/Q7*/Q8*/Q9)$$

$$+ SHFTCNT*(LTRT*(Q9*/Q10 + Q10*/Q9) + /LTRT*(Q1*/Q11 + Q11*/Q1));$$

$$Q11t = /SHFTCNT*(UPDN*Q1*Q2*Q3*Q4*Q5*Q6*Q7*Q8*Q9*Q10$$

$$+ /UPDN*/Q1*/Q2*/Q3*/Q4*/Q5*/Q6*/Q7*/Q8*/Q9*/Q10)$$

$$+ SHFTCNT*(LTRT*(Q10*/Q11 + Q11*/Q10) + /LTRT*(Q11*/Q12 + Q12*/Q11));$$

$$Q12t = /SHFTCNT*(UPDN*Q1*Q2*Q3*Q4*Q5*Q6*Q7*Q8*Q9*Q10*Q11$$

$$+ /UPDN*/Q1*/Q2*/Q3*/Q4*/Q5*/Q6*/Q7*/Q8*/Q9*/Q10*/Q11)$$

$$+ SHFTCNT*(LTRT*(Q11*/Q12 + Q12*/Q11) + /LTRT*(Q12*/Q13 + Q13*/Q12));$$

$$Q13t = /SHFTCNT*(UPDN*Q1*Q2*Q3*Q4*Q5*Q6*Q7*Q8*Q9*Q10*Q11*Q12$$

$$+ /UPDN*/Q1*/Q2*/Q3*/Q4*/Q5*/Q6*/Q7*/Q8*/Q9*/Q10*/Q11*/Q12)$$

$$+ SHFTCNT*(LTRT*(Q12*/Q13 + Q13*/Q12) + /LTRT*(Q13*/Q14 + Q14*/Q13));$$

$$Q14t = /SHFTCNT*(UPDN*Q1*Q2*Q3*Q4*Q5*Q6*Q7*Q8*Q9*Q10*Q11*Q12*Q13$$

$$+ /UPDN*/Q1*/Q2*/Q3*/Q4*/Q5*/Q6*/Q7*/Q8*/Q9*/Q10*/Q11*/Q12*Q13)$$

$$+ SHFTCNT*(LTRT*(Q13*/Q14 + Q14*/Q13) + /LTRT*(Q14*/Q15 + Q15*/Q14));$$

$$Q15t = /SHFTCNT*(UPDN*Q1*Q2*Q3*Q4*Q5*Q6*Q7*Q8*Q9*Q10*Q11*Q12*Q13*Q14$$

$$+ /UPDN*/Q1*/Q2*/Q3*/Q4*/Q5*/Q6*/Q7*/Q8*/Q9*/Q10*/Q11*/Q12*Q13*Q14)$$

$$+ SHFTCNT*(LTRT*(Q14*/Q15 + Q15*/Q14) + /LTRT*(Q15*/Q16 + Q16*/Q15));$$

$$Q16t = /SHFTCNT*(UPDN*Q1*Q2*Q3*Q4*Q5*Q6*Q7*Q8*Q9*Q10*Q11*Q12*Q13*Q14*Q15$$

$$+ /UPDN*/Q1*/Q2*/Q3*/Q4*/Q5*/Q6*/Q7*/Q8*/Q9*/Q10*/Q11*/Q12*Q13*Q14*Q15)$$

$$+ SHFTCNT*(LTRT*(Q15*/Q16 + Q16*/Q15) + /LTRT*(Q16*/DATA + DATA*/Q16));$$

END\$

### ESTIMATING GATE COMPLEXITY OF PROGRAMMABLE LOGIC DEVICES

An inflationary phenomenon may be occurring with respect to the gate complexity of programmable logic devices (PLDs). Although the equivalent gate count of early PAL™ and integrated fuse logic (IFL) circuits had been thought to be about 250 gates, newer parts that are only slightly more complex are purported to have as many as 900 gate equivalents. Perhaps this is a case of "gate-flation."

We all like to put our best foot forward, and manufacturers of PLDs are no exception. However, we are also responsible for establishing reasonable standards of comparison for PLDs lest we simply create confusion. The programmable logic market is growing at least as fast as the gate-array market is growing, and many new PLDs will appear in the next few years. We must begin to set the standards of comparison between various PLDs and other application-specific integrated circuits (ASICs). This article presents some general rules for establishing the gate-equivalent value of PLDs.

#### The Calculation Method

First, we must decide what a "gate" means. Let us use the definition that is accepted for gate arrays. A "gate" is a two-input NAND gate. Next, let's establish an "exchange rate" between the various resources in a PLD and the gates required to implement the equivalent function in a silicon-gate CMOS gate array. Last, let's add up the gates to obtain an overall exchange rate.

#### Fixed Resources

The resources in a PLD can be classified as being either *fixed* or *variable*. The fixed resources consist of input buffers and the associated inverters that drive the array, flip-flops, input latches, and output drivers. Exchange rates for fixed resources can be determined according to the gate count values in Figure 1. According to this data, a TTL input buffer and the associated inverters are equivalent to 4 gates; a D-type flip-flop (including SET, RESET and clock buffering) is equivalent to 9 gates; and a three-state output driver (excluding the output drive transistors) is equivalent to 7 gates. CMOS flip-flops built with transmission gates require both phases of the clock signal. In order to avoid the question of how many gates to allocate to clock trees, we have assumed that the clock inverters are built into the flip-flop cell. We have also assumed that it is necessary to connect gates in parallel to increase the

total drive capability for high-fan-out loads such as output transistors.

#### Variable Resources

The variable resource of a PLD consists of a programmable AND-OR array. The OR array can be either fixed or programmable, depending on the type of device. (The PAL structure uses a fixed OR array and the FPLA structure uses a programmable OR array). Each OR output can generate a sum-of-products expression that can involve any of the AND-array input or feedback terms.

So far, so good. The next step is to determine how many gates to associate with each OR output. For example, the Altera EP300 PLD has 74 product terms (programmable AND gates), each of which has 18 "true" and "complement" inputs. An 18-input AND gate would require at least 9 gates in a gate array. One could therefore infer that the EP300 array is the equivalent of 666 (74 × 9) gates. Fantastic! Unfortunately, few (if indeed any) useful applications could begin to use these 666 gates.

#### Example Calculations

To illustrate, we will examine a barrel shifter (Agrawal and Laws 1984). Figure 2 shows a possible gate-array implementation for one section of the barrel-shifter logic (excluding any inputs, outputs, or registers). Eight similarly structured sections are required for a complete 8-bit shifter. Thus, this gate-array implementation can be compared directly to the AND-OR implementation that would be used in a PLD. A total of 136 gates would be required for an eight-bit barrel shifter using the implementation shown in Figure 2.

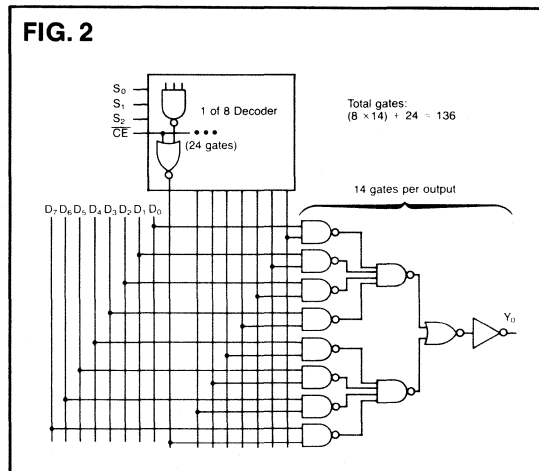
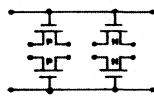







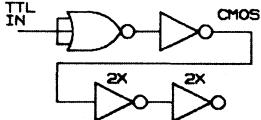
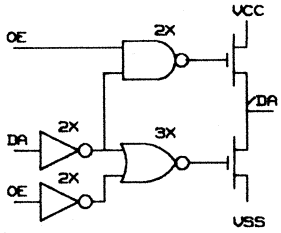
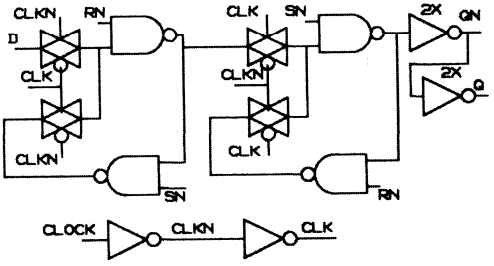


FIG. 1

Cell Type	Number of Core Cells	Schematic or Logic Diagram	Comment
Basic Cell	1		Typical 2n,2p silicon-gate CMOS core cell
Inverter	1/2		
High-Drive Inverter	1		Same as inverter with 2X drive
2-input NAND	1		
3-input NAND	1-1/2		Remaining 1/2 might be used as an inverter
2-input NOR	1		
3-input NOR	1-1/2		Remaining 1/2 might be used as an inverter
High-Drive NOR	3		Example of high drive capability for increased speed or fan-out
TTL-to-CMOS Translator Followed by High-Drive Inverter Pair	2		Typical for high fan-out signals from a TTL input
Three-State Output (Inverting)	7		Note: Output transistors are not made of core cells.
D-type Flip-Flop with SET and RESET, Buffered Outputs, and Clock	9		



Resource	Manufacturer and Part Type							
	Altera EP300		AMD 22V10		MMI PAL16R8		Signetics 82S159	
	Number	Gate Equivalents	Number	Gate Equivalents	Number	Gate Equivalents	Number	Gate Equivalents
Inputs	18	72	22	88	10	40	18	72
Flip-Flops <sup>(1)</sup>	8	72	10	90	8	72	8	72
Outputs (3-state)	8	56	10	70	8	56	12	84
OR Array Outputs (15 gates per OR) (30 gates per OR)	8	120 240	10	150 300	8	120 240	12	180 360
Total Gates								
Typical		320		398		288		408
Maximum		440		548		408		588

(1) Assumes D-type flip-flop option.

In addition to the number of gates actually required to implement the function, we need to account for the gates that are not used but nonetheless are obstructed on the array as a result of routing the interconnections for the desired function. Although gate array utilizations approaching 100% are possible, they are not typical. Because the barrel shifter requires a great deal of routing, we would expect less than 100% of the gate array to be utilized. If we assume approximately 70% utilization, then the actual usage for an 8-bit barrel shifter is 25 gates per bit. Thus, each OR output from the AND-OR array is equivalent to 25 gates.

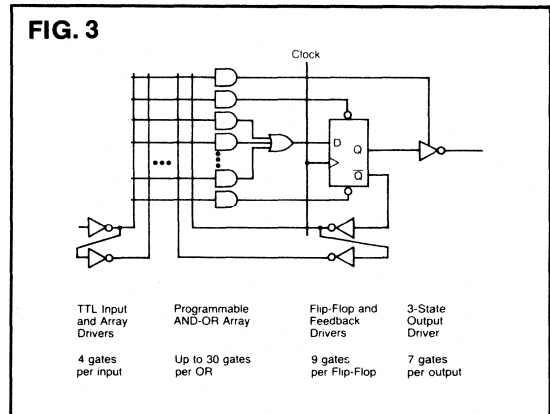
The barrel-shifter function is both routing intensive and product-term intensive. Therefore, it seems reasonable to assume that this function represents "near-maximum" utilization of the AND-OR array (or near-minimum utilization of the gate array). In addition to the barrel-shifter example, we have examined 40 gate-array logic diagrams and also many PLDs. Based on this sample, we believe that the exchange rate for the AND-OR array is from 15 to 30 gates per OR output. Given this range for the variable resource, we can formulate the exchange rate for any PLD, as indicated below:

$$\text{Exchange Rate} = (4 \times \text{number of inputs}) + (9 \times \text{number of flip-flops}) + (7 \times \text{number of three-state outputs}) + ((15 \text{ to } 30) \times \text{number of OR outputs from the AND-OR array}).$$

Figure 3 shows a typical PLD macrocell in a registered feedback configuration. Applying the above formula to this configuration yields an answer of 35 (typical) to 50 (maximum) gates per macrocell. Table 1 is a summary of the typical and maximum gate replacement values for several PLD types.

## ADDITIONAL FEATURES

The evolution of Programmable Logic Device architectures and functionality has been rapid. Since the publication of "Estimating Device Complexity of Programmable Logic Devices" in VLSI Design magazine in May, 1984, more advanced devices such as Altera's EP600, EP900 and EP1800 have been introduced. These sophisticated devices require extensions to the gate



count estimation method previously published in order to correctly assess their capabilities. The extensions to the gate count methodology are:

- **Input Latch** - Input latches, such as are present on the EP1200/EP1210, are given an equivalent weight of 2.5 gates per input.
- **Programmable Flip-Flops** - The versatile programmable flip-flops found on the EP600, EP900 and EP1800 are valued at 12 gates versus 9 gates for the D flip-flops found on other devices. These flip-flops may be configured as JK, SR, T or D elements.
- **Invert Control** - The ability to invert the output of the AND-OR array in these devices is valued at 1.5 gates per Macrocell.
- **Power-On Reset (POR)** - The Power-On Reset of registers is valued at 4 gates per device.
- **Preload** - Preloading of the internal registers for ease-of-testing is an enormous advantage. This function has been valued at 2 gates per flip-flop.

- Asynchronous Clocking - The ability to asynchronous clock (clock by Product Term) internal flip-flops aids the integration of common TTL glue logic. This option has been valued at 2 gates per flip-flop.

As mentioned in the article referenced above, the basic resources of PLD devices are weighted:

- Inputs - 4 gates per input line.
- Outputs - 7 gates per output.
- Macrocell - 15 gates per Macrocell associated logic array.
- D Flip-Flop - 9 gates per D Flip-Flop.

Together, between the basic gate count methodology and newer extensions, the user can accurately assess the capability of various PLD devices and find the one which best suits his particular application.

Many newer PLDs have programmable "features" other than the AND-OR array. These features are valuable when you are configuring a part to fit a particular application. For example, each output pin in the EP300 can be programmed individually to be registered or combinatorial, and active HIGH or active LOW. Also, the feedback can be programmed independently to be combinatorial or registered, or to come from the I/O pin. Although these features substantially increase the utility (one part does all) and ease of use of the part, it does not add greatly to the exchange rate of the part; therefore, in this analysis, we have ignored it.

In the next few years, PLDs containing several thousand "gates" will become available. Because of the overhead due to on-chip programming circuits, PLDs will probably always be less dense than gate arrays. Clearly, evaluating trade-offs between gate arrays and PLDs requires a common specification of device complexity. As discussed above, a few simple calculations will provide a good measure of the gate-equivalence value of PLDs.

The table below shows the available resources for each of the present members of the Altera product line.

Resource	Altera Product Resources				
	Device				
	EP300/ 310	EP600	EP900	EP1200/ 1210	EP1800
Inputs	18	20	38	36	64
Input Latch	-	-	-	36	-
Outputs	8	16	24	24	48
Macrocells	8	16	24	28	48
D Flip-Flops	8	-	-	28	-
Prog. Flip-Flops	-	16	24	-	48
Inv. Control	8	16	24	28	48
POR	1	1	1	1	1
Preload	8	16	24	28	48
Asynchronous Clock	-	16	24	-	48

Given these available resources and the above assumption set, the equivalent 2-input NAND gate complexity of the Altera product family is as follows:

	Altera Product Gate Equivalents				
	Device				
	EP300/ 310	EP600	EP900	EP1200/ 1210	EP1800
Total Gate Equivalent	352	716	1104	1176	2156

### Conclusion

Programmable logic devices have some advantages over other application-specific integrated circuits. Among these are immediate availability, virtually no non-recurring engineering charges, and multiple sources. Newer device types (such as the Altera EP300) are built with CMOS EPROM technology, which provides reprogrammability and low power consumption.

### Reference

Agrawal, O., and D. Laws. March 1984. "The Role of Programmable Logic in System Design." *VLSI DESIGN*.

### Acknowledgement

Portions of this Applications Brief appeared in the May 1984 issue of VLSI DESIGN magazine.



---

**APPENDICES****PAGE NO.**

---

Glossary of Terms .....	229
Reference Information .....	233
P.C. Compatibles Suitable for A+PLUS .....	233
Third Party Programming Support .....	234
Ordering Information .....	235
Package Information .....	236
Quality Information .....	238
Altera Distributors .....	241
Altera Sales Representatives .....	244



### SECTION I — GENERAL TERMS

#### **Altera Design File (ADF)**

The user-readable common entry file for the Altera Design Processor. It uses Boolean equation as well as a netlist format to describe the user's design. The ADF is the output of the NetMap and the Converter modules. It is converted by the Translator module into a Logic Equation File (LEF).

#### **Altera Design Processor (ADP)**

That portion of A+PLUS that processes an Altera Design File. It is the user interface to the A+PLUS software. It controls the individual modules and converts the ADF into a JEDEC Standard File used to program an Altera part.

#### **Altera Schematic Primitive**

One of the basic functional blocks needed to design circuits for Altera programmable logic products. Schematic primitives are used in both the Schematic Capture and the NetMap interfaces. Together, primitives make up the Altera symbol library which must be used exclusively when generating a design.

#### **A+PLUS**

Altera Programmable Logic User System. A+PLUS is a set of computer programs that facilitate design and implementation of custom logic circuits with Altera programmable logic products.

#### **Analyzer**

The module that converts a binary Logic Equation File (LEF) output by the Minimizer into a human-readable file.

#### **Assembler**

The ADP module that takes the output of the Fitter module and produces a JEDEC Standard File to be used by the LogicMap II program for programming an Altera part.

#### **Asynchronous Clear**

An input which causes the register to go to reset condition without waiting for a clock signal.

#### **Boolean Logic**

Describes logic that obeys the theorems of Boolean algebra. The Boolean portion of a design is that portion which can be implemented in the AND-OR matrix of an Altera part.

#### **Buried Register**

A register in an Altera part that is not associated with any pin and can be used to implement internal logic such as state machines.

#### **Combinatorial Feedback**

Feedback which is the direct function of the inputs, without regard to the clock; i.e. it does not retain values resulting from earlier input values.

#### **Combinatorial Output**

Output which is a direct function of the inputs, without regard to the clock; i.e., it does not retain values resulting from earlier input values.

#### **Converter**

The ADP module that translates the pinlist file output by the Schematic Capture package into a standard Altera Design File (ADF).

#### **Demander**

The ADP module that takes the equations output by the Expander or the Minimizer and converts them into the format needed by the Fitter module.

#### **DeMorgan's Inversion Theorem**

A theorem used in Boolean algebra stating that the complement of the product of the factors equals the sum of the complements of the addends; or that the complement of the sum of the addends equals the product of the complement of each factor. Example:  $(A*B)' = A' + B'$

#### **EPLD**

Erasable Programmable Logic Device. Any member of the Altera family of logic components.

#### **Expander**

The ADP module that expands the Boolean equations in the Logic Equation File (LEF) into sum-of-products form, checks them for evidence of combinatorial feedback, simplifies the results of the expansion, and produces another LEF.

#### **Fitter**

The ADP module that tries to match resource requests with the resources of the Altera part. If it is successful, it passes the output on to the Assembler; if not, it notifies the user. In either case, it produces a Utilization Report.

#### **Global Feedback**

A macrocell's capability to pass its output back to all other macrocells of the part.

#### **I/O Feedback**

Feedback from the output pin on an Altera part. It allows an output pin to also be used as an input pin.

#### **JEDEC Standard File**

An industry-wide standard for the transfer of information between a data preparation system and a logic device programmer.

### **Local Feedback**

A macrocell's capability to pass back its output to only a restricted number of macrocells rather than to all of the macrocells in the part. See "Global Feedback."

### **Logic Array Input**

A signal which is fed to the Boolean portion of a design.

### **Logic Equation File (LEF)**

A data structure used by several ADP modules. In the LEF the Boolean portion of the design is separated from the non-Boolean portion to allow down-stream programs to process the design according to their own specifications.

### **LogicMap II Program**

A program in A+PLUS which allows the user to program an Altera part at the bit level. The program includes an interactive data editor for manual entering of data. It performs timing and transfer functions for part programming. Data is stored in standard JEDEC format.

### **Macrocell**

A basic building block of Altera's programmable logic devices. A macrocell consists of two sections: combinatorial logic and output logic. The combinatorial logic allows a wide variety of logic functions. The output logic has two data paths: one leads to the other macrocells or feeds back to the macrocell itself; the other is configured as a pin connection acting as input, output, or bi-directional I/O port on the chip.

### **Minimizer**

The ADP module that takes the Boolean equations output by the Expander and reduces them using various theorems of Boolean logic.

### **Node**

The name given to a wire connecting two or more primitives in a schematic.

### **Pin**

The actual pin of an EPLD, i.e., a node that is connected to an Input or I/O Primitive on one end and a pin of the chip on the other end.

### **Product Term (P-Term)**

Two or more factors in a Boolean expression combined with the AND-operator constitute a product term, "product" meaning "logic product."

### **Race Condition**

An undesirable logic state resulting from Boolean combinations of signals which are the outputs of logic elements having finite propagation delays.

### **Registered Feedback**

Feedback which is the output of a clocked storage device.

### **Registered Output**

The output of a clocked storage device.

### **State Diagram**

A pictorial representation of a sequential design. Each "circle" represents a state of the machine, while the line segments between the circles are graphical indications

of state changes. It allows the user to scan each state and gather information needed to determine the previous state, the conditions for the next state, and the output for each state.

### **Sum-of-Products**

A Boolean expression is said to be in sum-of-products form if it consists of product terms combined with the OR-operator.

### **Synchronous Preset**

An input which has no impact on the register until an appropriate transition of the clock is effected.

### **Three-State Buffer**

A buffer with an input, output, and controlling signal. If the controlling signal is HIGH, the output is a defined function of the input. If the controlling signal is LOW, the output is not a defined function of the input.

### **Translator**

The ADP module that converts an Altera Design File (ADF) into a Logic Equation File. It acts as initial screen for input errors by checking the validity of the requests in the ADF.

### **Turbo-Bit**

A control bit that allows the user to choose the speed and power characteristics of a device.

### **Verify Protect**

A feature that prevents a device from being interrogated or inadvertently reprogrammed.

## **SECTION II — OPERATING CONDITIONS**

## **AND CHARACTERISTICS (INCLUDING**

## **LETTER SYMBOLS)**

### **INTRODUCTION**

These symbols and definitions are in accordance with those currently agreed upon by the JEDEC Council of the Electronic Industries Association (EIA) for use in the USA and by the International Electrotechnical Commission (IEC) for international use.

### **Clock Frequency**

#### **Maximum clock frequency, $f_{max}$**

The highest rate at which the clock input of a bistable circuit can be driven through its required sequence while maintaining stable transitions of logic level at the output with input conditions established that should cause changes of output logic level in accordance with the specification.

### **Current**

#### **High-level input current, $I_{IH}$**

The current into an input when a high-level voltage is applied to that input.

#### **High-level output current, $I_{OH}$**

The current into an output with input conditions applied that according to the product specification will establish a high level at the output.

**Low level input current,  $I_{IL}$** 

The current into an input when a low-level voltage is applied to that input.

**Low level input current,  $I_{OL}$** 

The current into an output with input conditions applied that according to the product specification will establish a low level at the output.

**Off-state output current,  $I_{OL(off)}$** 

The current flowing into an output with input conditions applied that according to the product specification will cause the output switching element to be in the off state.

**Note:** This parameter is usually specified for open-collector outputs intended to drive devices other than logic circuits.

**Off-state (high-impedance-state) output current (of a three-state output),  $I_{OZ}$** 

The current into an output having three-state capability with input conditions applied that according to the product specification will establish the high-impedance state at the output.

**Short-circuit output current,  $I_{OS}$** 

The current into an output when the output is short-circuited to ground (or other specified potential) with input conditions applied to establish the output logic level farthest from ground potential (or other specified potential).

**Supply current  $I_{CC}$** 

The current into the  $V_{CC}$  supply terminal of an integrated circuit.

**Hold Time****Hold time,  $t_h$** 

The interval during which a signal is retained at a specified input terminal after an active transition occurs at another specified input terminal.

**NOTES:**

1. The hold time is the actual time between two events and may be insufficient to accomplish the intended result. A minimum value is specified that is the shortest interval for which correct operation of the logic element is guaranteed.
2. The hold time may have a negative value in which case the minimum input defines the longest interval (between the release of data and the active transition) for which correct operation of the logic element is guaranteed.

**Output Enable and Disable Time****Output enable time (of a three-state output) to high level  $t_{PZH}$  (or low level,  $t_{PZL}$ )**

The propagation delay time between the specified reference points on the input and output voltage waveforms with the three-state output changing from a high-impedance (off) state to the defined high (or low) level.

**Output enable time (of a three-state output) to high or low level,  $t_{PZX}$** 

The propagation delay time between the specified reference points on the input and output voltage waveforms with the three-state output changing from a high-impedance (off) state to either of the defined active levels (high or low).

**Output disable time (of a three-state output) from high level,  $t_{PHZ}$  (or low level,  $t_{PLZ}$ )**

The propagation delay time between the specified reference points on the input and output voltage waveforms with the three-state output changing from the defined high (or low) level to high-impedance (off) state.

**Output disable time (of a three-state output) from high or low level,  $t_{PZX}$** 

The propagation delay time between the specified reference points on the input and output voltage waveforms with the three-state output changing from either of the defined active levels (high or low) to a high-impedance (off) state.

**Propagation Time****Propagation delay time,  $t_{PD}$** 

The time between the specified reference points on the input and output voltage waveforms with the output changing from one defined level (high or low) to the other defined level.

**Propagation delay time, low-to-high-level output,  $t_{PLH}$** 

The time between the specified reference points on the input and output voltage waveforms with the output changing from the defined low level to the defined high level.

**Propagation delay time, high-to-low-level output,  $t_{PHL}$** 

The time between the specified reference points on the input and output voltage waveforms with the output changing from the defined high level to the defined low level.

**Pulse Width****Pulse width,  $t_w$** 

The time interval between specified reference points on the leading and trailing edges of the pulse waveform.

**Recovery Time****Sense recovery time,  $t_{SR}$** 

The time interval needed to switch a memory from a write mode to a read mode and to obtain valid data signals at the output.

**Release Time****Release time,  $t_{release}$** 

The time interval between the release from a specified input terminal of data intended to be recognized and the occurrence of an active transition at another specified input terminal.

**Note:** When specified, the interval designated “release time” falls within the setup interval and constitutes, in effect, a negative hold time.

## Setup Time

### Setup time, $t_{su}$

The time interval between the application of a signal that is maintained at a specified input terminal and a consecutive active transition at another specified input terminal.

#### NOTES:

1. The setup time is the actual time between two events and may be insufficient to accomplish the setup. A minimum value is specified that is the shortest interval for which correct operation of the logic element is guaranteed.
2. The setup time may have a negative value in which case the minimum limit defines the longest interval (between the active transition and the application of the other signal) for which correct operation of the logic element is guaranteed.

## Transition Time

### Transition time, low-to-high-level, $t_{TLH}$

The time between a specified low-level voltage and a specified high-level voltage on a waveform that is changing from the defined low level to the defined high level.

### Transition time, high-to-low-level, $t_{THL}$

The time between a specified high-level voltage and a specified low-level voltage on a waveform that is changing from the defined high level to the defined low level.

## Voltage

### High-level input voltage, $V_{IH}$

An input voltage within the more positive (less negative) of the two ranges of values used to represent the binary variables.

**NOTE:** A minimum is specified that is the least positive value of high-level input voltage for which operation of the logic element within specification limits is guaranteed.

### High-level output voltage, $V_{OH}$

The voltage at an output terminal with input conditions applied that according to the product specification will establish a high level at the output.

### Input clamp voltage, $V_{IK}$

An input voltage in a region of relatively low differential resistance that serves to limit the input voltage swing.

### Low-level input voltage, $V_{IL}$

An input voltage level within the less positive (more negative) of the two ranges of values used to represent the binary variables.

**NOTE:** A maximum is specified that is the most positive value of low-level input voltage for which operation of the logic element within specification limits is guaranteed.

### Low-level output voltage, $V_{OL}$

The voltage at an output terminal with input conditions applied that according to the product specifications will establish a low level at the output.

### Negative-going threshold voltage, $V_{T-}$

The voltage level at a transition-operated input that causes operation of the logic element according to specification as the input voltage falls from a level above the positive-going threshold voltage,  $V_{T+}$ .

### Off-state output voltage, $V_{O(off)}$

The voltage at an output terminal with input conditions applied that according to the product specification will cause the output switching element to be in the off state.

**Note:** This characteristic is usually specified only for outputs not having internal pull-up elements.

### On-state output voltage, $V_{O(on)}$

The voltage at an output terminal with input conditions applied that according to the product specification will cause the output switching element to be in the on state.

**Note:** This characteristic is usually specified only for outputs not having internal pull-up elements.

### Positive-going threshold voltage, $V_{T+}$

The voltage level at a transition-operated input that causes operation of the logic element according to specification as the input voltage rises from a level below the negative-going threshold voltage,  $V_{T-}$ .



#### MAGAZINE ARTICLES

#### AND TECHNICAL PAPERS

1. Hartmann, R. May 1984. "Estimating Gate Complexity of Programmable Logic Devices," VLSI DESIGN.
2. McCarthy, C. June 14, 1984. "Neither Masks nor Fuses Mar Design Flexibility of EPROM-based Logic," ELECTRONIC DESIGN.
3. Rappaport, A. May 1984. "First CMOS Reprogrammable Logic Array Specs Low Power and UV Erasability," EDN.
4. Saxby, D., Duncan, R., Glaviano, M., and Neroth, C. February 1985. "Erasable-PLD Program Translates Logic Directly into Chips," EDN.
5. Hartmann, R. November 1984. "Instant Turnaround Custom ICs — New Concepts in VLSI Design," Wescon Proceedings.
6. Chan, Y. November 1984. "Programmable Logic Replaces Gate Arrays," Wescon Proceedings.
7. Hartmann, R. March 1985. "CMOS Erasable Programmable Logic Devices — TTL Replacement Made Easy," Southcon Conference Proceedings.
8. Neroth, C. and Saxby D. July 1985. "Design Software for Erasable Programmable Logic Devices," VLSI DESIGN.
9. Hartmann, R. and Wong, S. July 11, 1985. "EPROM-based Logic Chip Opens Its Gates to All Flip-Flop Types, Clocks," ELECTRONIC DESIGN.
10. Kopec, S. August 26, 1985. "Advanced PLD Architectures Require Sophisticated Support," ELECTRONIC ENGINEERING TIMES.

#### COMPATIBLE COMPUTERS

The following is a partial list of computers that have been found to be compatible with the A+PLUS software.

AT&T  
 Compaq Plus  
 Compaq Deskpro  
 Columbia  
 Columbia portable  
 Corona  
 Eagle  
 IBM PC  
 IBM XT  
 IBM AT  
 IBM portable  
 ITT xtra  
 Leading Edge  
 MAD 86186  
 NCR adds  
 Televideo  
 Sperry  
 Intersil system pc  
 ISI model 5160  
 Zenith 100 & 151

#### Minimum computer configuration:

- Monochrome display
- 512k bytes of main memory
- Dual floppy-disk drives
- MS-DOS or PC-DOS versions 2.0 or later releases
- Full-card slot for programming card

#### Recommended computer configuration:

- Color graphics or Enhanced graphics display
- 640k bytes of main memory
- 10M byte hard disk drive and floppy-disk drive
- MS-DOS or PC-DOS versions 2.0 or later releases
- Full-card slot for programming card

#### TEXTBOOKS

1. LOGIC DESIGN OF DIGITAL SYSTEMS, Donald L. Dietmeyer, Allyn and Bacon, Inc. 1978, Boston, MA
2. LOGIC MINIMIZATION ALGORITHMS FOR VLSI SYNTHESIS, Robert K. Brayton, Gary D. Hachtel, Curtis T. McMullen, Alberto L. Sangiovanni, Vincenzelli, Kluwer Academic Publishers, 1984, Boston, MA
3. DIGITAL CIRCUITS AND MICROPROCESSORS, Herbert Taub, McGraw-Hill Book Co., 1982, New York, NY
4. ANALYSIS AND DESIGN OF DIGITAL INTEGRATED CIRCUITS, David A. Hodges, Horace G. Jackson, McGraw-Hill Book Co., 1983, New York, NY
5. AN ENGINEERING APPROACH TO DIGITAL DESIGN, William I. Fletcher, Prentice-Hall, Inc., 1980, Englewood Cliffs, NJ
6. FUNDAMENTALS OF LOGIC DESIGN, Charles H. Roth, Jr., West Publishing Co., 1979, St. Paul, MN
7. DIGITAL NETWORKS AND COMPUTER SYSTEMS, Taylor L. Booth, John Wiley & Sons, 1978, New York, NY

All EPLDs manufactured by Altera are supported by the A+PLUS development system and by third party programmer manufacturers.

addresses. They will be able to tell you the specific programmer model that supports Altera EPLDs.

Third Party programmer manufacturers that support Altera EPLDs may be contacted at the following

#### Data I/O

10525 Willows Road N.E.  
P.O. Box 97046  
Redmond, Washington  
98073-9746  
United States  
phone (206) 881-6444

World Trade Centre  
Strawinskylaan 633  
1077 XX Amsterdam  
The Netherlands  
phone (20) 622866

#### Stag Electronic Designs

Tewin Court  
Welwyn Garden City  
Hertfordshire AL7 1AU  
United Kingdom  
phone (07073) 32148  
528-5 Weddell Drive  
Sunnyvale, California 94089  
United States  
phone (408) 745-1991

#### INTEL Corporation

5200 N.E. Elam Young Parkway  
Hillsboro, Oregon 97124-6497

#### Valley Data Sciences

Charleston Business Park  
2426 Charleston Road  
Mountain View, California 94043  
phone (415) 968-2900

#### Varix

122 Spanish Village suite 608  
Dallas, Texas 75248  
phone (214) 437-0777

#### OAE

676 West Wilson Avenue  
Glendale, California 91203  
phone (818) 240-0080

#### Logical Devices

1321 North West 65th Place  
Fort Lauderdale, Florida 33309  
phone (305) 974-0975

#### Digelec Inc.

1602 Lawrence Avenue suite 113  
Ocean, New Jersey 07712  
phone (201) 493-2420

#### Elan Digital Systems Ltd.

16-20 Kelvin Way  
Crawley, West Sussex  
England  
RH10 2TS

#### Japan Macnics Corp.

516 Imaiminami-Cho, Nakahara-Ku  
Kawasaki-City, 211  
Japan  
phone 044-711-0022

#### Minato Electronics Inc.

4105 Minami Yamada-Cho  
Kohoku-Ku, Yokohama 223  
Japan

#### Digitronics Israel Ltd.

25 Galgaley Haplada Street  
Herzliya B'  
Israel  
46722

#### Kontron Messtechnik GMBH

Breslauer Strasse 2,8057  
Eching B. Munchen  
West Germany  
phone 89-3-19-01-374

#### AVAL Corporation

11 Deansgrange Ind. Estate  
Deansgrange County Dublin  
Ireland  
850533

#### Sunrise Electronics

524 South Vermont Avenue  
Glendora, California 91740  
phone (818) 914-1926

#### Digital Media

3178 Gibraltar Avenue  
Costa Mesa, California 92626

#### Electronic System Products Inc.

1135-C San Antonio Road  
Palo Alto, California 94303  
phone (415) 964-5338

#### Micropross

5 Rue Denis-Papin  
59650 Villeneuve-D'Ascq  
France  
phone 20-47-90-40

#### Stack

43 Lower Close  
Aylesbury Buckinghamshire  
England  
HP21 8SB  
phone 44-296-33470

Note: Altera assumes no responsibility for the suitability or accuracy of third party programming equipment.

Altera EPLDs should be programmed using data formatted within the JEDEC recommended standard for PLD object code. To obtain electrical programming information for programming Altera EPLDs from the JEDEC format write to:

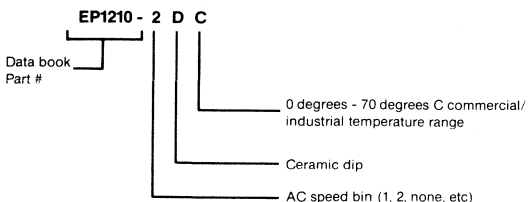
Applications Department  
(EPLD programming data)  
**Altera Corporation**  
3525 Monroe Street  
Santa Clara, California 95051  
United States

Copies of the appropriate JEDEC standard:  
JEDEC Standard No.3 (JC-42.1)  
may be obtained from the following address:

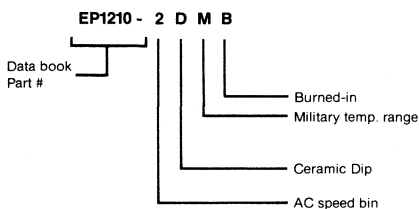
JEDEC Executive Secretary  
**Electronics Industries Association**  
2001 Eye Street N.W.  
Washington, D.C. 20006  
United States

Some examples of ordering various package and electrical as well as temperature grades are given below.

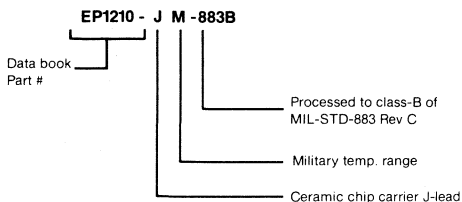
**a) Level-1 Product:**



**b) Level-2 Product:**



**c) Level-B, MIL-STD-883C:**



SYSTEM DESIGNATION	ORDER BY
PLDS 2	Same designation
PLCAD 1	Same designation
PLE 1	Same designation

### PRODUCT CODE SUMMARY

### AND ORDERING INFORMATION.

#### PACKAGE CODES:

PACKAGE TYPE	MARKING/ORDERING LETTER DESIGNATOR
CERAMIC DIP	D
PLASTIC MOLDED DIP	P
CERAMIC J-LEAD CHIP CARRIER	J
PLASTIC MOLDED J-LEAD CHIP CARRIER	L
CERAMIC PIN GRID ARRAY	G

#### PRODUCT GRADES:

APPLICATION	TEMPERATURE RANGE	MARKING DESIGNATOR
COMMERCIAL	0° C TO + 70° C	C
AUTOMOTIVE/ INDUSTRIAL	-40° C TO + 85° C	I
MILITARY	-55° C TO +125° C	M
MIL-STD-883 REV. C CLASS-B	-55° C TO +125° C	883 B

**Notes:**

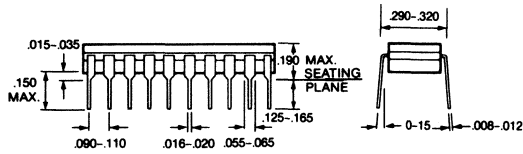
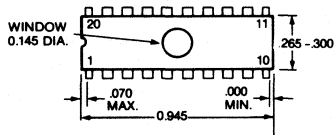
For specific package/grade/speed combinations that are available, please refer to product listing on page 18 or call Altera marketing department.

### DEVELOPMENT SYSTEMS/SOFTWARE

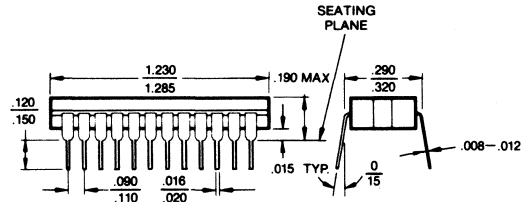
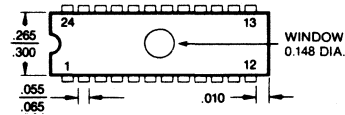
### ORDERING INFORMATION

All development systems and software products should be ordered by their data sheet nomenclature. No other codes are assigned. Some examples are listed.

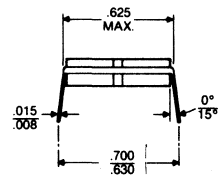
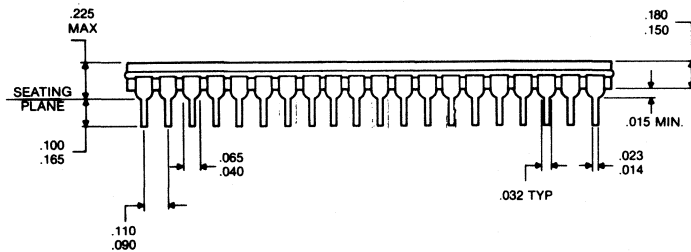
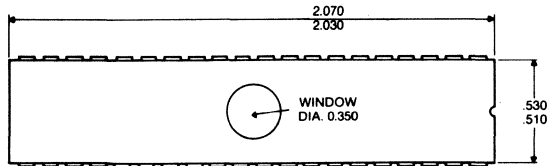
### 20 LEAD HERMETIC DIP



### 24 LEAD HERMETIC DIP

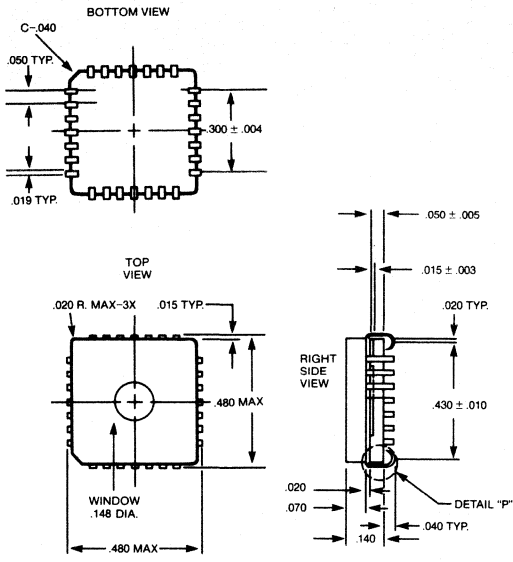


### 40 LEAD HERMETIC DIP

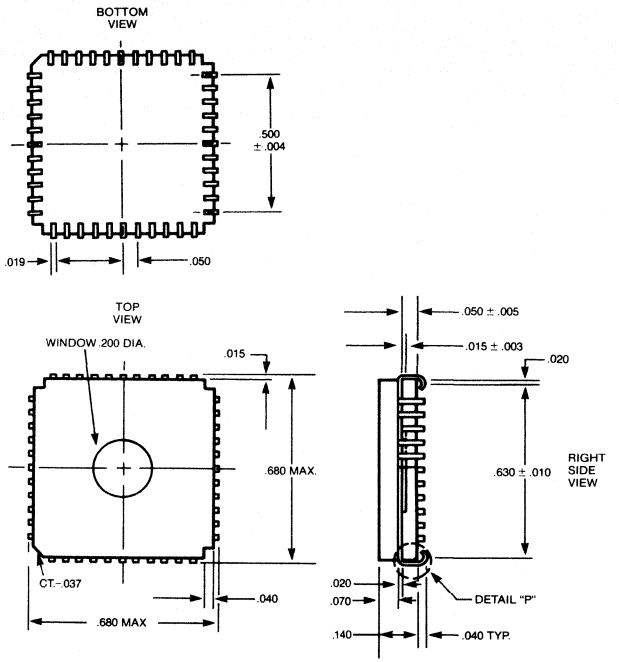


ALL DIMENSIONS MIN.-MAX. IN INCHES.

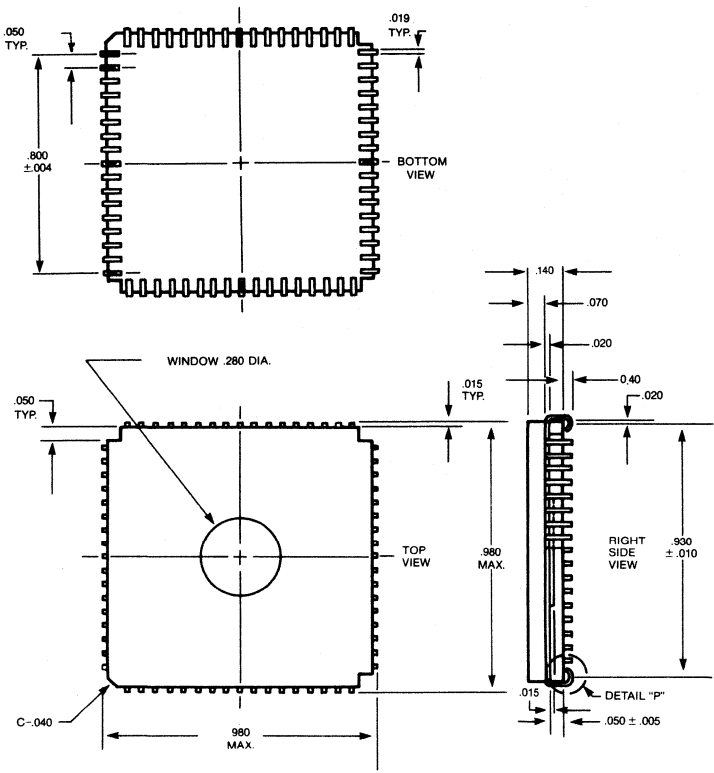
**28 LEAD QUAD CERPAC TYPE "J"**



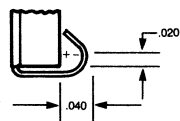
**44 LEAD QUAD CERPAC TYPE "J"**



**68 LEAD QUAD CERPAC TYPE "J"**



**DETAIL "P"**



**CORPORATE QUALITY PHILOSOPHY:** "Defect prevention rather than defect detection" is the underlying theme of Altera's quality philosophy. "Zero defects" quality level for Altera products is achieved by placing major responsibility for this task with the engineering and manufacturing personnel rather than the quality personnel. In other words, "build quality in so that you do not have to inspect it out." Processes, material and training of personnel are controlled at every step so that products that fully comply to specifications and long term reliability expectations are delivered to the customers. Some of the actions taken to achieve very high quality levels are:

- Quality and reliability requirements form a very important part of every product design from its inception.
- Constant control of processes and materials at all manufacturing steps ensures that non-compliant material or process does not progress before corrective actions are taken.
- Ongoing training of personnel ensures their awareness of material and process compliance requirements.
- Vendors are selected so that the highest quality material is received by Altera.

Altera is committed to the concept of a strong self-audit and vendor-audit program which identifies key review areas and the frequency of their audit in order to keep those areas in strict compliance with specifications.

**INTEGRATED CIRCUITS:** Altera's CMOS-EPROM-processed ICs are offered for performance in three different temperature ranges and three different processing levels as outlined below.

**Ambient Temperature Range:**

- Commercial-Industrial: 0°C to 70°C
- Industrial/automotive -40°C to +85°C
- Aerospace & Defense: -55°C to +125°C

**Processing Levels:**

Level 1: Where the device has achieved processing maturity and reliability to the degree that 100% burn-in is not performed (Table A).

Level 2: For commercial/industrial as well as military temperature ranges with 100% burn-in and other tests as outlined in Table A.

Level B,883C: Aerospace and defense applications product processed in full compliance to MIL-STD-883, Rev. C as outlined in Table B.

**Group C Tests:** Life tests per Method 5005.8 on one inspection lot from each technology group are performed every 13 weeks for appropriate date code material.

**Group D Tests:** Package related tests per Method 5005.8 are performed every six months for each package type and lead finish.

Customers needing information on process flows and specifications beyond what is covered in this databook are requested to contact Altera Marketing Department.

**IC PRODUCT QUALIFICATIONS:** All IC products are subjected to the following testing and characterization before full release to production processing and shipment.

1. **Electrical Characterization:** Functional, parametric and dynamic testing over the guaranteed temperature ranges and anticipated wafer fabrication parametric distribution. Testing, followed by a thorough data analysis provides proper limits for data sheet conditions.
2. **Operating Life Test:** A sample of at least 129 parts is subjected to 2000 hours of operating life test at 125 degrees C for chip qualification. A maximum failure rate of 200 fits at 55 degrees C and activation energy of 1.0 ev is the criterion for product qualification.
3. **High Temperature Storage:** Products are subjected to a 250 degrees C bake for up to 500 hours with electrical testing performed at time points 0, 48 hrs, 168 hrs and 500 hrs. A failure rate of 200 fits at 55 degrees C and activation energy of 1.0 ev is the criterion for product qualification.
4. **Electrostatic Discharge Test (ESD):** ESD testing is performed per method 3015.2 of MIL-STD-883-C. A voltage level  $\geq 1000$  volts without any ESD damage is the criterion for product qualification.
5. **IC Package Qualification Tests:** Each package type is subjected to a series of reliability tests depending on whether it is hermetic or plastic. Table C below provides a list of tests that are performed on hermetic packages before initial production release.

**Development Systems Hardware:**

Development system hardware quality is emphasized for defect prevention as much as the IC quality. Hardware is processed as indicated below.

- Board layout and check using CAD TOOLS.
- Board fabrication using UL94V-0 flammability standard material.
- Incoming board inspection.
- Procure burned-in active components for systems.

- Board stuffing and wave soldering with controls for preheat temperature and solder temperature.
- 100% inspection and touch-up as needed.
- Board clean and cleanliness monitor for ionic contamination.
- Electrical test.
- Board burn-in, 65 degrees C, 72 hours.
- Final electrical test in conjunction with system software.
- Final inspection.
- Other hardware assembly as required.
- System pack and inspect before shipment.

Several improvements were made in the next generation of HMOS-E EPROMs which significantly reduced the incidence of first-gate defects. Improved testing techniques allowed first-gate oxide defects to be caught in standard retention bake testing performed on every EPROM shipped. Second-gate oxide defects now account for the majority of charge-loss defects in HMOS EPROMs. Failure rates per bit have decreased with each new generation of EPROMs introduced, owing to improvements in processing and screening from knowledge gained on previous generations. Since the introduction of N-channel EPROMs in 1975, the failure rate has dropped from 800 FIT to 70 FIT, (100 FIT = 0.1 percent fail/1000 hrs test) and the density has increased from 8K bits to 256K bits for over a 300-times improvement in failure rate per kilobit.

### EPROM CHARGE RETENTION RELIABILITY

Altera devices are tested with all the reliability screens used in standard EPROM production. The unique reliability characteristic of the EPROM is the ability to store charge on the floating gate for an unlimited time period. By surrounding the floating gate with high quality, thermally grown silicon dioxide, a 3.1 eV energy barrier traps electrons on the floating gate.

Typical EPROM cells have been shown to retain their charge essentially forever. The failure rate of EPROMs is limited by oxide defects causing charge loss or charge gain. Defects in either gate oxide will cause charge loss. In early NMOS EPROMs, the first-gate oxide defect was found to be the predominant defect.

EPROM reliability shows very little dependence on the oxide breakdown failure mechanism, which is the dominant failure mechanism in most MOS integrated circuits. This reliability is due to the effect of programming as a latent oxide defect screen. Due to this screen, oxide breakdown makes a very small contribution to the device failure rate, providing one of the reasons for the excellent reliability exhibited by EPROMs.

Acknowledgement: Contents of these three paragraphs based on "Components Quality/Reliability Handbook" 1985, (Pages 3-25 to 3-27), published by Intel Corp., Santa Clara, Ca.

**TABLE A Hermetic Packages**

Major Manufacturing Steps Following Electrically Tested Wafers	Method MIL-STD-883 or Other	Process Level-1	Process Level-2
Die visual prior to attach	Altera spec.	100%	100%
Die attach shear test	Method 2019.3	Monitor	Monitor
Wire bond pull test	Method 2011, Cond. D	Monitor	Monitor
Internal visual Hermetic seal	Altera spec.	100%	100%
Post seal high temperature storage 250 degrees C	Altera spec.	4 hrs.	4 hrs.
Temp. cycle, 10 cycles	Method 1010.5, cond. C	100%	100%
Fine/Gross leak check	Method 1014.6	LTPD 5/0	LTPD 5/0
External visual acceptance	Altera spec.	100%	100%
Pattern program EPROM	Device spec.	100%	100%
Retention bake 72 hrs., 140° C	Altera spec.	100%	100%
Post bake elect. test	Device spec.	100%	100%
Burn-in, 160 hrs., 125° C	Altera schematic	Sample	100%
Post burn-in & final electrical test	Device spec.	100% +25° C 100% +75° C	100% +25° C *100% -55° C *100% +125° C
Electrical sample acceptance	Device spec.	LTPD 2/0	LTPD 2/0
External visual acceptance	Altera spec.	LTPD 3/1	LTPD 3/1

\* For military temperature range only. +75 degrees C commercial/industrial grades.

**TABLE B**  
**Hermetic Packages**  
**Major Process Steps for MIL-STD-883-C, Level-B Product**

<b>Process</b>	<b>MIL-STD-883-C Method &amp; Condition</b>	<b>100% or LTPD as Specified</b>
Die visual	2010.7, Cond. B	100%
Die attach shear monitor	2019.3	3 samples/lot
Wire bond pull monitor	2011.4, Cond. D	10/0, 4 units
Internal visual	2010.7, Cond. B	100%
Hermetic seal		100%
Stabilization bake	1008.2, Cond. C	100%
Temperature cycle	1010.5, Cond. C	100%
Constant acceleration	2001.2, Cond. E Y1 axis only	100%
Fine-gross leak	1014.6, Cond. B fine leak Cond. C2 gross leak	100%
Pattern program EPROM, ≥99% bits	Device spec.	100%
Retention bake 72 hrs., 140 degrees C		100%
Post bake elect. test	Device spec.	100%
Pre burn-in electrical test	Altera spec.	100%
Burn-in	1015.5, Cond. D	100%
Post burn-in elect. test	25 degrees C — Altera data sheet specs.  125 degrees C — Altera  -55 degrees C — Altera	100%  100%  100%
Group A acceptance	5005.8, Table I	LTPD as applicable for each test temperature
Mark & bake	Marking per Altera data sheet	
Group-B acceptance	5005.8, Table IIB	LTPD as applicable
Documentation acceptance & transfer to inventory		

**TABLE C**

Hermetic packages must successfully complete the following test sequence before production release.

THERMAL SHOCK -55 DEG C TO +125 DEG C 15 CYCLES	100 TEMP. CYCLES -65 DEG C +150 DEG C	MECHANICAL SHOCK METHOD 2002, COND. B	LEAD INTEGRITY METHOD 2004, COND. B2	RESISTANCE TO SOLVENTS METHOD 2015 5 SAMPLES 0 REJECTS	PHYSICAL DIMENSIONS METHOD 2016 5 SAMPLES 0 REJECTS	INTERNAL WAFER VAPOR METHOD 1018 3 SAMPLES 0 REJECTS
FINE LEAK	FINE LEAK	CONSTANT ACCELERATION METHOD 2001, COND. E Y1 AXIS	FINE LEAK	LID TORQUE METHOD 2024 5 SAMPLES 0 REJECTS	SOLDERABILITY METHOD 2003.3 OR METHOD 2022 5 SAMPLES 0 REJECTS	
GROSS LEAK ELECTRICALS	GROSS LEAK ELECTRICALS	FINE LEAK GROSS LEAK	GROSS LEAK BOND PULL METHOD 2011, COND. D			
LTPD 5/1	LTPD 5/1	ELECTRICALS LTPD 15/0	LTPD 15/0 DIE SHEAR METHOD 2019.2 5 SAMPLES 0 REJECTS			

**NOTE:**

1. All methods referred to are per MIL-STD-883-C.
2. All fine and gross leak tests are per Method 1014.





## ALTERA DISTRIBUTORS USA and CANADA

### ALABAMA

Pioneer Standard  
4825 University  
Huntsville, AL 35805  
(205) 837-9300

Schweber Electronics  
2227 Drake Avenue  
Suite 14  
Huntsville, AL 35805  
(205) 882-2200

### ARIZONA

Schweber Electronics  
11049 N 23rd Drive  
Suite 100  
Phoenix, AZ 85029  
(602) 997-4874

Wyle Laboratories  
17855 North Black Canyon Hwy  
Phoenix, AZ 85023  
(602) 866-2888

### CALIFORNIA

Schweber Electronics  
21139 Victory Blvd.  
Canoga Park, CA 91303  
(818) 999-4702

Schweber Electronics  
90 East Tasman Drive  
San Jose, CA 95134  
(408) 946-7171

Schweber Electronics  
17822 Gillette Avenue  
Irvine, CA 92714  
(714) 863-0200

Schweber Electronics  
6750 Nancy Ridge Drive  
Suite D & E, Building 7  
Carroll Ridge Business Park  
San Diego, CA 92121  
(619) 450-0454

Schweber Electronics  
1771 Tribute Road  
Suite B  
Sacramento, CA 95815  
(916) 929-9732

Wyle Laboratories  
11151 Sun Center Drive  
Rancho Cordova, CA 95670  
(916) 638-5282

Wyle Laboratories  
3000 Bowers Avenue  
Santa Clara, CA 95051  
(408) 727-2500

Wyle Laboratories  
17872 Cowan Avenue  
Irvine, CA 92714  
(714) 863-9953

Wyle Laboratories  
124 Maryland Street  
El Segundo, CA 90245  
(213) 322-8100

Wyle Laboratories  
9525 Chesapeake Drive  
San Diego, CA 92123  
(619) 565-9171

Wyle Laboratories  
26560 Agoura Road  
Suite 203  
Calabasas, CA 91302  
(818) 880-9001

### COLORADO

Schweber Electronics  
Highland Tech Business Park  
Suite 200  
8955 East Nichols Avenue  
Englewood, CO 80112  
(303) 799-0258

Wyle Laboratories  
451 E 124th Street  
Thornton, CO 80241  
(303) 457-9953

### CONNECTICUT

Pioneer Standard  
112 Main Street  
Norwalk, CT 06851  
(203) 853-1515

Schweber Electronics  
Commerce Park  
Finance Drive  
Danbury, CT 06810  
(203) 748-7080

### FLORIDA

Pioneer Standard  
221 North Lake Blvd.  
Altamonte Springs, FL 32701  
(305) 834-9090

Pioneer Standard  
674 South Military Trail  
Deerfield Beach, FL 33441  
(305) 428-8877

Schweber Electronics  
2830 N 28th Terrace  
Hollywood, FL 33020  
(305) 927-0511

Schweber Electronics  
215 N Lake Blvd.  
Altamonte Springs, FL 32701  
(305) 331-7555

### GEORGIA

Pioneer Standard  
5835 B Peachtree Corners East  
Norcross, GA 30092  
(404) 448-1711

Schweber Electronics  
303 Research Drive  
Suite 210  
Norcross, GA 30092  
(404) 449-9170

### IOWA

Schweber Electronics  
5270 N Park Place N E  
Cedar Rapids, IA 52402  
(319) 373-1417

### ILLINOIS

Pioneer Standard  
1551 Carmen Drive  
Elk Grove Village, IL 60007  
(312) 437-9680

Schweber Electronics  
904 Cambridge Drive  
Elk Grove Village, IL 60007  
(312) 364-3750

### INDIANA

Pioneer Standard  
6408 Castleplace Drive  
Indianapolis, IN 46250  
(317) 849-7300

### KANSAS

Schweber Electronics  
10300 W 103rd Street  
Suite 200  
Overland Park, KS 66214  
(913) 492-2922

### MASSACHUSETTS

Pioneer Standard  
44 Hartwell Avenue  
Lexington, MA 02173  
(617) 861-9200

Schweber Electronics  
25 Wiggins Avenue  
Bedford, MA 01730  
(617) 275-5100

Schweber Electronics  
265 Ballardvale Street  
Wilmington, MA 01887  
(617) 657-8760

### MARYLAND

Pioneer Standard  
9100 Gaither Road  
Gaithersburg, MD 20877  
(301) 921-0660

Schweber Electronics  
9330 Gaither Road  
Gaithersburg, MD 20877  
(301) 840-5900

### MICHIGAN

Pioneer Standard  
13485 Stamford  
Livonia, MI 48150  
(313) 525-1800

Schweber Electronics  
12060 Hubbard Drive  
Livonia, MI 48150  
(313) 525-8100

### MINNESOTA

Pioneer Standard  
10203 Bren Road East  
Minnetonka, MN 55343  
(612) 935-5444

Schweber Electronics  
7424 W 78th Street  
Edina, MI 55435  
(612) 941-5280

### MISSOURI

Schweber Electronics  
502 Earth City Expressway  
Earth City, MO 63045  
(314) 739-0526

### NORTH CAROLINA

Pioneer Standard  
9801 A Southern Pine Blvd.  
Charlotte, NC 28210  
(704) 527-8188

Schweber Electronics  
1 North Commerce Center  
5285 North Blvd.  
Raleigh, NC 27604  
(919) 876-0000

### NEW HAMPSHIRE

Schweber Electronics  
Belford Farms, Bldg. 2  
First Floor  
Manchester, NH 03102  
(603) 625-2250

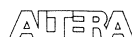
### NEW JERSEY

Pioneer Standard  
45 Route 46  
Pine Brook, NJ 07058  
(201) 575-3510

Schweber Electronics  
18 Madison Road  
Fairfield, NJ 07006  
(201) 227-7880

### NEW MEXICO

Alliance Electronics  
11030 Cochiti SE  
Albuquerque, NM 87123  
(505) 292-3360



**NEW YORK**

Pioneer Standard  
60 Crossways Park West  
Woodbury, NY 11797  
(516) 921-8700

Pioneer Standard  
1806 Vestal Parkway East  
Vestal, NY 13850  
(607) 748-8211

Pioneer Standard  
840 Fairport Park  
Fairport, NY 14450  
(716) 381-7070

Schweber Electronics  
3 Town Line Circle  
Rochester, NY 14623  
(716) 424-2222

Schweber Electronics  
Jericho Turnpike  
Westbury, NY 11590  
(516) 334-7474

**OHIO**

Pioneer Standard  
4800 East 131st Street  
Cleveland, OH 44105  
(216) 587-3600

Pioneer Standard  
4433 Interpoint Blvd.  
Dayton, OH 45424  
(513) 236-9900

Schweber Electronics  
7865 Paragon Road  
Suite 210  
Dayton, OH 45459  
(513) 439-1800

Schweber Electronics  
23880 Commerce Park Road  
Beachwood, OH 44122  
(216) 464-2970

**OKLAHOMA**

Quality Components  
9934 East 21st Street South  
Tulsa, OK 74129  
(918) 664-8812

Schweber Electronics  
4815 South Sheridan  
Fountain Plaza  
Suite 109  
Tulsa, OK 74145  
(918) 622-8000

**OREGON**

Wyle Laboratories  
5289 NE Elam Young Parkway  
Bldg. E100  
Hillsboro, OR 97123  
(503) 640-6000

**PENNSYLVANIA**

Pioneer Standard  
261 Gibraltar Road  
Horsham, PA 19044  
(215) 674-4000

Pioneer Standard  
259 Kappa Drive  
Pittsburg, PA 15238  
(412) 782-2300

Schweber Electronics  
1000 R I D C Plaza  
Suite 203  
Pittsburg, PA 15238  
(412) 782-1600

Schweber Electronics  
231 Gibraltar Road  
Horsham, PA 19044  
(215) 441-0600

**TEXAS**

Pioneer Standard  
5853 Point West Drive  
Houston, TX 77036  
(713) 988-5555

Pioneer Standard  
9901 Burnet Road  
Austin, TX 78758  
(512) 835-4000

Pioneer Standard  
13710 Omega Road  
Dallas, TX 75244  
(214) 386-7300

Quality Components  
2120 West Braker Lane  
Suite M  
Austin, TX 78758  
(512) 835-0220

Quality Components  
1005 Industrial Blvd.  
Sugar Land, TX 77487  
(713) 240-2255

Quality Components  
4257 Kellway Circle  
Addison, TX 75001  
(214) 733-4300

Schweber Electronics  
4202 Beltway Drive  
Dallas, TX 75234  
(214) 661-5010

Schweber Electronics  
10625 Richmond Avenue  
Suite 100  
Houston, TX 77042  
(713) 784-3600

Schweber Electronics  
6300 La Calma Drive  
Suite 240  
Austin, TX 78752  
(512) 458-8253

Wyle Laboratories  
2120 West Braker Lane  
Suite F  
Austin, TX 78758  
(512) 834-9957

Wyle Laboratories  
1810 Greenville Avenue  
Richardson, TX 75081  
(214) 235-9953

**UTAH**

Wyle Laboratories  
1959 South 4130 West  
Salt Lake City, UT 84104  
(801) 974-9953

**WASHINGTON**

Wyle Laboratories  
1750 132nd Avenue NE  
Bellevue, WA 98005  
(206) 453-8300

**WISCONSIN**

Schweber Electronics  
150 South Sunnyslope Road  
Brookfield, WI 53005  
(414) 784-9020

**CANADA****QUEBEC**

Future Electronics  
237 Hymus Blvd.  
Pointe-Claire, Quebec  
H9R 5C7  
(514) 694-7710

**ONTARIO**

Future Electronics  
Baxter Center  
1050 Baster Rd.  
Ottawa, Ontario  
K2C 3P2  
(613) 820-8313

Future Electronics  
82 St. Regis Crescent North  
Downsview, Ontario  
M35 1Z3  
(416) 638-4771

**ALBERTA**

Future Electronics  
3220 5th Ave.  
North East Calgary, Alberta  
T2A 5N1  
(403) 235-5325

Future Electronics  
5312 Calgary Trail  
Edmonton, Alberta  
T6H 458  
(403) 438-2858

**BRITISH****COLUMBIA**

Future Electronics  
3070 Kings Way  
Vancouver, B.C.  
V5R 5S7  
(604) 438-5545

### DENMARK

E.V. Johansen Elektronik A/S  
Titangade 15  
DK-2200 København N  
Denmark  
Telephone: (01) 83 90 22  
Telex: 16522 evicas dk

### ENGLAND

Cascom Microelectronics Ltd.  
Thame Park Road  
Thame  
Oxon OX93XD  
England  
Telephone: (44) 84421-7222  
Telex: 837508  
Telefax: 084421 7185

### FRANCE

Tekelec-Airtronic  
Paris Sud  
Tour Evry 2  
523 Place des Terrasses  
91034 Evry Cedex  
Telephone: 16 (6) 077 82 66  
Telex: 691 158 F

Tekelec-Airtronic  
Paris 92  
BP NR 2  
1 Rue Carle Vernet  
92310 Sevres  
Telephone: 16 (1) 534 75 92  
Telex: 204 552 F

Tekelec-Airtronic  
Paris 78  
5 Allée du Bourbonnais  
78310 Maurepas  
Telephone: 16 (3) 062 00 58  
Telex: 698 121 F

Tekelec-Airtronic  
Paris Est  
424, La Closerie Bât A  
Clos Mont d'Est  
93160 Noisy Le Grand  
Telephone: 16 (1) 304 62 00  
Telex: 220 368 F

Tekelec-Airtronic  
Paris Nord  
8 Avenue Salvador Allende  
93804 Epinay Cedex  
Telephone: 16 (1) 821 60 44  
Telex: 630 260 F

Tekelec-Airtronic  
Lyon  
26, Rue de la Baisse  
69100 Villeurbanne  
Telephone: 16 (6) 884 06 08  
Telex: 370 481 F

Tekelec-Airtronic  
Grenoble  
Locazirst 4  
Chemin des Prés  
38240 Meylan  
Telephone: 16 (76) 41 11 36  
Telex: 980 207 F

Tekelec-Airtronic  
Aix en Provence  
Bâtiment "Le Mercure"  
Avenue Ampère BP 77  
13762 Les Milles Cedex  
Telephone: 16 (42) 24 40 45  
Telex: 440 928 F

Tekelec-Airtronic  
Toulouse  
22/24 Boulevard Thibaud  
31084 Toulouse Cedex  
Telephone: 16 (61) 40 83 94  
Telex: 520 374 F

Tekelec-Airtronic  
Bordeaux  
Immeuble "Le Montesquieu"  
Avenue Président Kennedy  
33700 Merignac  
Telephone: 16 (56) 34 84 11  
Telex: 550 589 F

Tekelec-Airtronic  
Lille  
Immeuble Moulin 2  
5 Rue du Colibri  
59650 Villeneuve D'ASCQ  
Telephone: 16 (20) 05 17 00  
Telex: 160 011 F

Tekelec-Airtronic  
Strasbourg  
1 Rue Gustave Adolphe Hirn  
67000 Strasbourg  
Telephone: 16 (88) 22 31 51  
Telex: 880 765 F

Tekelec-Airtronic  
Rennes  
20 Avenue de Crimée  
B.P. 2246  
35022 Rennes Cedex  
Telephone: 16 (99) 50 62 35  
Telex: 740 414 F

Tekelec-Airtronic  
Strasbourg  
1 Rue Gustave Adolphe Hirn  
67000 Strasbourg  
Telephone: 16 (88) 22 31 51  
Telex: 880 765 F

Tekelec-Airtronic  
Rennes  
20 Avenue de Crimée  
B.P. 2246  
35022 Rennes Cedex  
Telephone: 16 (99) 50 62 35  
Telex: 740 414 F

### GERMANY

Electronic 2000  
München  
Stahlgruberring 12  
8000 München 82  
Telephone: 0 89/42 00 1-0  
Telex: 522 561

Electronic 2000  
Ditzingen  
Siemensstr. 1  
7257 Ditzingen 1  
Telephone: 0 71 56/70 83  
Telex: 7-245 265

Electronic 2000  
Nürnberg  
AuBere Sulzbacher Str. 37  
8500 Nürnberg 20  
Telephone: 09 11/59 50 58  
Telex: 6-26 495

Electronic 2000  
Frankfurt  
Langer Weg 18  
6000 Frankfurt/M 90  
Telephone: 069/78 00 87  
Telex: 4-189 486

Electronic 2000  
Düsseldorf  
Werstener Dorfstr. 27  
4000 Düsseldorf 13  
Telephone: 02 11/76 71 41  
Telex: 8-586 810

Electronic 2000  
Hamburg  
Überseering 25  
2000 Hamburg 60  
Telephone: 040/6 30 40 81  
Telex: 2-164 921

Electronic 2000  
Berlin  
Otto-Suhr-Allee 9  
1000 Berlin 10  
Telephone: 030/341 70 81-82  
Telex: 185 323

### BELGIUM

Diode  
Rue de l'Aeronef 2  
1140 Bruxelles, Belgique  
Telephone: (02) 216 21 00  
Telex: 25903

### HOLLAND

Diode  
Hollantlaan 22  
3526 AM Utrecht, Holland  
Telephone: (030) 88 42 14  
Telex: 47388

### ISRAEL

Vectronics Ltd.  
60 Medinat Hayehudim Street  
P.O. Box 2024  
Herzlia B 46120  
Israel  
Telephone: (972) 52-556-070  
Telex: 342579

### ITALY

Inter-Rep  
10148 Torino  
Via Orbetello, 98  
Telephone: 011/21.65.901 (15 linee)  
Telex: 221422

Inter-Rep  
20151 Milano  
Via Gadames, 128  
Telephone: 02/30.11.620 (rra.)  
Telex: 221422

Inter-Rep  
36016 Thiene  
Via Valbella, 10 (cond. Alfa)  
Telephone: 0445/36.49.61-36.38.90  
Telex: 221422

Inter-Rep  
40138 Bologna  
Via E. Mattei, 40  
Telephone: 051/53.11.99 (rra.)  
Telex: 221422

Inter-Rep  
50127 Firenze  
Via Panciatichi, 40  
Telephone: 055/43.60.422-43.60.392  
Telex: 221422

Inter-Rep  
00159 Roma  
Via Tiburtina, 436  
Telephone: 06/43.90.490  
Telex: 221422

### JAPAN

Japan Macnics Corporation  
516 Imaiminami-Cho  
Nakahara-Ku  
Kawasaki-City  
211 Japan  
Telephone: (81) 44-711-0022  
Telex: 28988

Paltek Corporation  
15-1-1104 Nanpeidai  
Shibuya-Ku  
Tokyo 150  
Japan  
Telephone: (81) 3-464-1554  
Telex: 02425205

### SPAIN

Selco  
Paseo de la Habana, 190  
28036-Madrid  
Spain  
Telephone: (34) 405-4213  
Telex: 45458

### SWEDEN

Betoma  
Box 3005, S-171 03  
Solna, Sweden  
Visitadress: Dalvägen 12  
Telephone: (46) 8 82 02 80  
Telex: 19389 betoma s.  
Telefax: 46-8 82 80 90

Fertronic  
Box 56, S-161  
26 Bromma, Sweden  
Visitadress: Snörmarkarvägen 35  
Telephone: (46) 8 80 28 00  
Telex: 11181 fertron s.

### SWITZERLAND

Stolz Ag  
Täferstrasse 15  
CH-5405 Baden-Dättwil  
Telephone: (41) 56 84 01 51  
Telex: 825088

Stolz Ag  
Av. Louis-Casati 81  
CH-1216 Genève  
Telephone: (41) 22 98 78 77

### ALABAMA

Montgomery Marketing, Inc.  
4922 Cotton Row  
Huntsville, AL 35805  
(205) 830-0498

### ARIZONA

Tusar  
6016 E. Larkspur  
Scottsdale, AZ 85254  
(602) 998-3688

### ARKANSAS

Technical Marketing, Inc.  
3320 Wiley Post Road  
Carrollton, TX 75006  
(214) 387-3601

### CALIFORNIA

Addem  
1015 Chestnut Avenue  
Suite 330  
Carlsbad, CA 92008  
(619) 729-9216

Magna Sales  
3333 Bowers Avenue  
Suite 251  
Santa Clara, CA 95051  
(408) 727-8753

Hi-Tech Rep Company  
1111 Laguna Road  
Suite 108  
Tustin, CA 92680  
(714) 730-9561

Hi-Tech Rep Company  
31332 Via Colinas  
Suite 109  
Westlake Village, CA 91362  
(818) 706-2916

### COLORADO

Simpson Associates, Inc.  
2552 Ridge Road  
Littleton, CO 80120  
(303) 794-8381

### DISTRICT OF COLUMBIA

Robert Electronic Sales  
5525 Twin Knolls Road  
Suite 331  
Columbia, MD 21045  
(301) 982-1177

### CONNECTICUT

Technology Sales Inc.  
60 Church Street  
Suite 18  
Yalesville, CT 06492  
(203) 269-8853

### DELAWARE

BGR Associates  
2002-D Greentree  
Executive Campus  
Marlton, NJ 08053  
(609) 983-1020

### FLORIDA

EIR Inc.  
1057 Maitland Ctr Commons  
Maitland, FL 32751  
(305) 660-9600

### GEORGIA

Montgomery Marketing, Inc.  
3000 Northwoods Parkway  
Suite 245  
Norcross, GA 30071  
(404) 447-6124

### IDAHO

Simpson Associates, Inc.  
7324 South 1300 East  
Suite 350  
Midvale, UT 84047  
(801) 566-3691

Westerberg & Associates, Inc.  
12505 NE Bel-Red Road  
Suite 112  
Bellevue, WA 98005  
(206) 453-8881

### ILLINOIS

Oasis Sales Corporation  
1101 Tonne Road  
Elk Grove Village, IL 60007  
(312) 640-1850

Midwest Technical Sales Inc.  
136 Cedar Crest Ct.  
St. Charles, MO 63301  
(314) 441-1012

### INDIANA

Electro Reps Inc.  
6535 East 82nd Street  
Suite 214  
Indianapolis, IN 46250  
(317) 842-7202

### IOWA

HMR Incorporated  
9065 Lyndale Avenue South  
Minneapolis, MN 55420  
(612) 888-2122

### KANSAS

Midwest Technical Sales Inc.  
8015 W. 63rd Suite 1  
Merriam, KS 66202  
(913) 236-8555

Midwest Technical Sales Inc.  
837 Perry  
Wichita, KS 67203  
(316) 262-7240

### KENTUCKY

Electro Reps Inc.  
6535 East 82nd Street  
Suite 214  
Indianapolis, IN 46250  
(317) 842-7202

### LOUISIANA

Technical Marketing, Inc.  
2901 Wilcrest Drive  
Suite 139  
Houston, TX 77042  
(713) 783-4497

### MAINE

Technology Sales Inc.  
60 Church Street  
Suite 18  
Yalesville, CT 06492  
(203) 269-8853

### MARYLAND

Robert Electronic Sales  
5525 Twin Knolls Road  
Suite 331  
Columbia, MD 21045  
(301) 995-1900

### MASSACHUSETTS

Technology Sales Inc.  
21 Green Street  
Waltham, MA 02154  
(617) 647-5700

### MICHIGAN

Rathsburg Associates Inc.  
16621 East Warren Avenue  
Detroit, MI 48224  
(313) 882-1717

### MINNESOTA

HMR Incorporated  
9065 Lyndale Avenue South  
Minneapolis, MN 55420  
(612) 888-2122

### MISSISSIPPI

Montgomery Marketing, Inc.  
3000 Northwoods Parkway  
Suite 245  
Norcross, GA 30071  
(404) 447-6124

### MISSOURI

Midwest Technical Sales Inc.  
136 Cedar Crest Ct.  
St. Charles, MO 63301  
(314) 441-1012

### MONTANA

Simpson Associates, Inc.  
2552 Ridge Road  
Littleton, CO 80120  
(303) 794-8381

### NEBRASKA

Midwest Technical Sales Inc.  
8015 W. 63rd Suite 1  
Merriam, KS 66202  
(913) 236-8555

Midwest Technical Sales Inc.

837 Perry  
Wichita, KS 67203  
(316) 262-7240

### NEVADA

Magna Sales  
3333 Bowers Avenue  
Suite 251  
Santa Clara, CA 95051  
(408) 727-8753

Tusar  
6016 E. Larkspur  
Scottsdale, AZ 85254  
(602) 998-3688

### NEW HAMPSHIRE

Technology Sales Inc.  
21 Green Street  
Waltham, MA 02154  
(617) 647-5700

### NEW JERSEY

BGR Associates  
2002-D Greentree  
Executive Campus  
Marlton, NJ 08053  
(609) 983-1020

ERA Inc.  
354 Veterans Memorial Highway  
Commack, NY 11725  
(516) 543-0510

### NEW MEXICO

Nelco Electronix  
4801 General Bradley, N.E.  
Albuquerque, NM 87111  
(505) 292-3657

**NEW YORK (Metro)**

ERA Inc.  
354 Veterans Memorial Highway  
Commack, NY 11725  
(516) 543-0510

**NEW YORK STATE**

T-Squared Electronics Co., Inc.  
7343 Victor-Pittsford Road  
Victor, NY 14564  
(716) 924-9101

T-Squared Electronics Co., Inc.  
6443 Ridings Road  
Suite 126  
Syracuse, NY 13206  
(315) 463-8592

**NORTH CAROLINA**

Montgomery Marketing, Inc.  
Box 520  
Cary, NC 27511  
(919) 467-6319

**NORTH DAKOTA**

HMR Incorporated  
9065 Lyndale Avenue South  
Minneapolis, MN 55420  
(612) 888-2122

**OHIO**

The Lyons Corporation  
4812 Frederick Road  
Suite 101  
Dayton, OH 45414  
(513) 278-0714

The Lyons Corporation  
4615 W Streetsboro Road  
Richfield, OH 44286  
(216) 659-9224

**OKLAHOMA**

Technical Marketing, Inc.  
3320 Wiley Post Road  
Carrollton, TX 75006  
(214) 387-3601

**OREGON**

Westerburg & Associates Inc.  
7165 SW Fir Loop  
Portland, OR 97223  
(503) 620-1931

**PENNSYLVANIA**

The Lyons Corporation  
4812 Frederick Road  
Suite 101  
Dayton, OH 45414  
(513) 278-0714

BGR Associates  
2002-D Greentree  
Executive Campus  
Marlton, NJ 08053  
(609) 983-1020

**PUERTO RICO**

Technology Sales Inc.  
Box 121  
San German, PR 00753-1021  
(809) 892-4745

**RHODE ISLAND**

Technology Sales Inc.  
60 Church Street  
Suite 18  
Yalesville, CT 06492  
(203) 269-8853

**SOUTH CAROLINA**

Montgomery Marketing, Inc.  
Box 520  
Cary, NC 27511  
(919) 467-6319

**SOUTH DAKOTA**

HMR Incorporated  
9065 Lyndale Avenue South  
Minneapolis, MN 55420  
(612) 888-2122

**TENNESSEE**

Montgomery Marketing, Inc.  
4922 Cotton Row  
Huntsville, AL 35805  
(205) 830-0498

**TEXAS**

Technical Marketing, Inc.  
3320 Wiley Post Road  
Carrollton, TX 75006  
(214) 387-3601

Technical Marketing, Inc.  
2901 Wilcrest Drive  
Suite 139  
Houston, TX 77042  
(713) 783-4497

Technical Marketing, Inc.  
9027 Northgate Boulevard  
Suite 140  
Austin, TX 78758  
(512) 835-0064

**UTAH**

Simpson Associates, Inc.  
7324 South 1300 East  
Suite 350  
Midvale, UT 84047  
(801) 566-3691

**VERMONT**

Technology Sales Inc.  
60 Church Street  
Suite 18  
Yalesville, CT 06492  
(203) 269-8853

**VIRGINIA**

Robert Electronic Sales  
7637 Hull St. Road  
Suite 103  
Richmond, VA 23235  
(804) 276-3979

**WASHINGTON**

Westerberg & Associates, Inc.  
12505 NE Bel-Red Road  
Suite 112  
Bellevue, WA 98005  
(206) 453-8881

**WEST VIRGINIA**

The Lyons Corporation  
4812 Frederick Road  
Suite 101  
Dayton, OH 45414  
(513) 278-0714

**WISCONSIN**

Oasis Sales Corporation  
1305 N Barker Road  
Brookfield, WI 53005  
(414) 782-6660

HMR Incorporated  
9065 Lyndale Avenue South  
Minneapolis, MN 55420  
(612) 888-2122

**WYOMING**

Simpson Associates, Inc.  
2552 Ridge Road  
Littleton, CO 80120  
(303) 794-8381

**CANADA**

Kaytronics  
106-10334-152 A Street  
Surrey  
BC, Canada V3R 7P8  
(604) 581-5005

Kaytronics  
4019 Carling Avenue  
Suite #204  
Kanata  
Ontario, Canada K2K 2A3  
(613) 592-6606

Kaytronics  
Unit No. 1, 331 Bowes Road  
Concord  
Ontario, Canada L4K 1B1  
(416) 669-2262

Kaytronics  
200 Richmond  
Ville St. Pierre  
Quebec, Canada H8R LY8  
(514) 367-0101

**ALTERA SALES****OFFICES****CALIFORNIA**

Altera Corporation  
3525 Monroe Street  
Santa Clara, CA 95051  
(408) 984-2800  
Telex: 888496

**MASSACHUSETTS**

Altera Corporation  
945 Concord Street  
Framingham, MA 01701  
(617) 626-0181  
Telex: 948477



