Altera PLMG7256

Programming Adapter

$395.00

In Stock

Qty Available: 1

New From Surplus Stock

Open Web Page

https://www.artisantg.com/49382-1

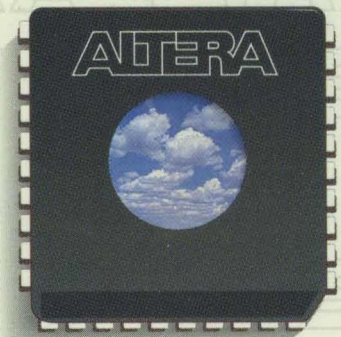
ARTISAN TECHNOLOGY GROUP

Your **definitive** source for quality pre-owned equipment.

Artisan Technology Group

(217) 352-9330 | sales@artisantg.com | artisantg.com

- Critical and expedited services
- In stock / Ready-to-ship
- We buy your excess, underutilized, and idle equipment
- Full-service, independent repair center



The Maximalist Handbook

ALTERA

The Maximalist Handbook

"Maximalist"— A person who favors a radical and immediate approach to the achievement of a set of goals or the completion of a program.

Random House Dictionary

January 1990

Altera Corporation
2610 Orchard Parkway
San Jose, CA 95134-2020
(408) 984-2800
Applications Information:
(408) 984-2805 ext. 102

About This Handbook

This handbook contains a collection of product information bulletins, datasheets, application notes and briefs, and various other pieces of information about Altera's MAX devices and development tools.

Section 1 includes an overview of the MAX architecture and a discussion of the technology issues and alternatives in the application-specific market. This section also answers some questions asked from a manager's perspective.

Section 2 describes the entire MAX EPLD family, MAX+PLUS software, and optional products, and provides recommendations for designing with MAX devices.

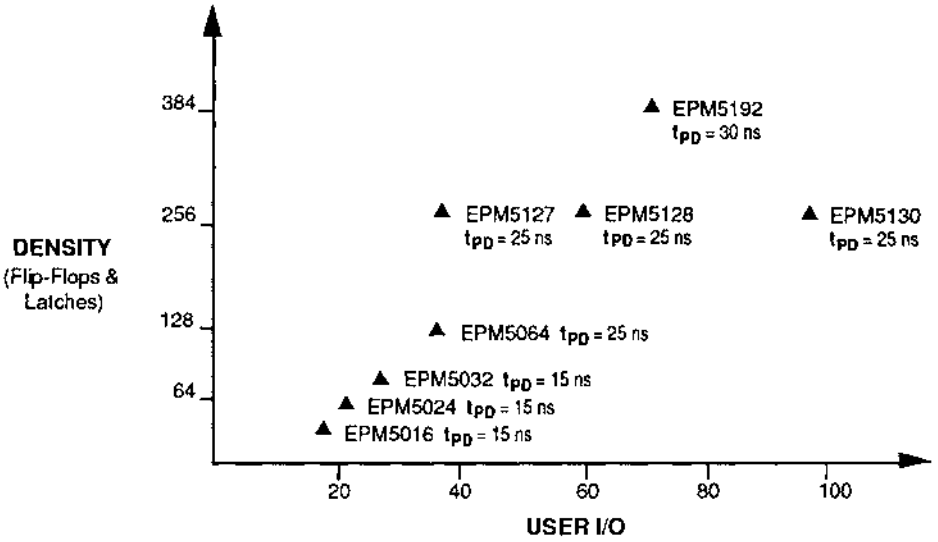
Section 3 addresses specific applications for MAX devices and MAX+PLUS software.

Section 4 provides instructions on how to use Altera's Electronic Bulletin Board and how to order MAX products. It also includes a list of currently available Applications literature.

If this handbook doesn't answer your technical questions, please call Altera Applications at (408) 984-2805 ext. 102 for immediate assistance.

Contents

MAX The Complete Industry-Standard Programmable Logic Family

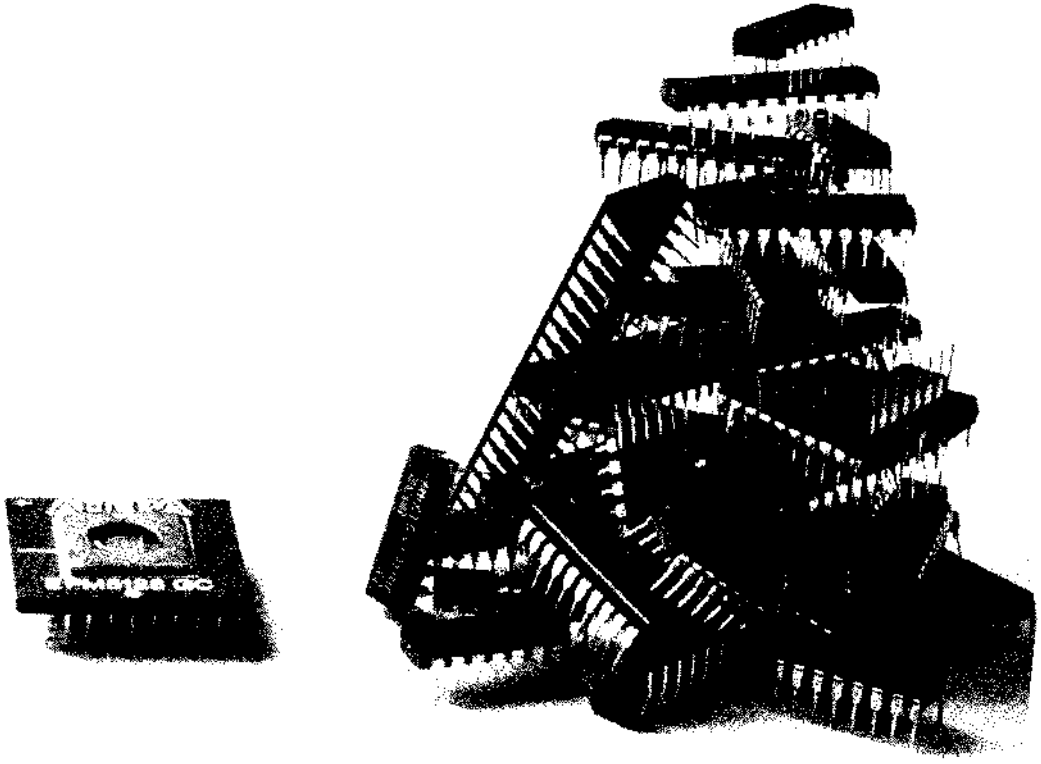


- The Altera EPM-series of Multiple Array MatriX (MAX) EPLDs offers the industry's most comprehensive family of programmable logic building blocks.
- Advanced MAX architecture provides the speed, ease of use, and familiarity of PAL devices with the density of programmable gate arrays.
- Modular family structure solves design tasks from fast 20-pin address decoders to 100-pin LSI custom peripherals.
- Non-volatile, reprogrammable EPROM technology aids prototype development.
- High sequential logic capacity provides up to 384 registers plus latches.
- Up to 66 product terms per output ensure efficient design of complex state-machines.
- Exactly emulates all popular 7400-series functions to convert existing CMOS and TTL designs.
- Easily integrates multiple-package PAL and PLA designs.
- Provides 15-ns combinatorial delays, counters up to 100 MHz, pipelined data rates of 100 MHz, and high-complexity designs with true system clock rates up to 66 MHz.
- A full selection of packages is provided, including DIP, SO, J-lead, PGA, and QFP formats in windowed ceramic and plastic OTP versions.
- Easily converts to custom-masked silicon for very high-volume production.
- MAX+PLUS PC-based design tools—including hierarchical schematic and advanced hardware design language entry methods, efficient logic synthesis-based compiler, and full timing simulation—support MAX EPLDs.
- Logic compilation and automatic place and route of 1000+ gate designs performed in minutes.
- EDIF industry-standard workstation and third-party CAE tool interfaces are available.

Contents

Section 1 MAX Family Overview

1



MAX EPLD
Family Architecture

January 1990 **Product Information Bulletin 4**

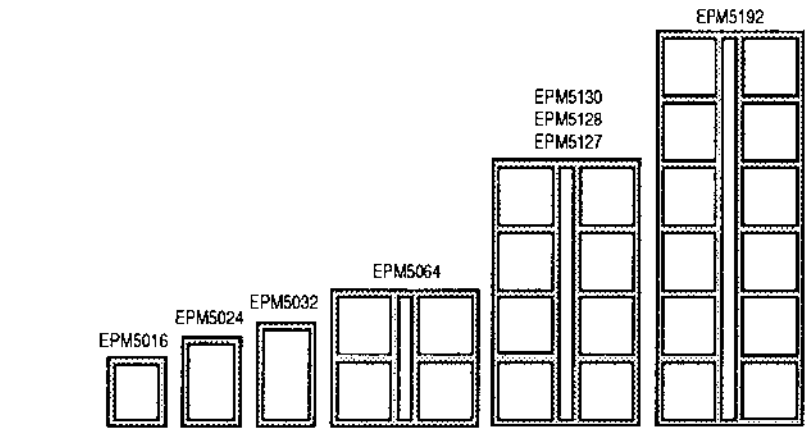
1

Introduction

MAX (Multiple Array MatriX) Erasable Programmable Logic Device (EPLD) architecture was developed to provide the foundation for the first complete high-performance CMOS logic family based on user-configurable technology. Earlier PLDs have found efficient application in selected replacement of standard TTL and CMOS logic products. However, inconsistency of architecture between devices, limitations in logic features, and low device densities have prevented their wide acceptance as a primary logic solution. The MAX EPLD approach breaks through these barriers. See Figure 1.

MAX architecture has evolved from the original EPLD structure, achieving both higher performance and more efficient use of logic resources. The powerful logic array structure and the familiar and easy use of PAL devices and EPLDs are retained as well as the logic density of gate arrays. Additional MAX architecture enhancements permit the integration of high-density combinatorial and register-intensive logic functions. The result is a flexible family of EPLDs that accommodates exact functional equivalents of hundreds of popular 7400-series elements.

Figure 1. MAX Family Modular Architecture



These features are incorporated into the EPM5000 series of EPLDs from Altera. They comprise a consistent series of modular logic building blocks ranging from 20-pin dual-in-line packages to 100-pin pin-grid array and surface-mount packages. The integration density of these devices ranges from 2 or more PAL devices at the low end to 24 to 30 devices at the upper end. The latter range is equivalent to 100 or more standard TTL packages.

The most important objective for the MAX architecture was to overcome the serious limitations of earlier PLD structures, which were unable to achieve higher levels of integration of TTL-based designs. These limitations traditionally have included fixed product-term distribution, lack of register control inputs, and tight coupling of register and I/O pin resources, to name a few. The following pages review the key features of the MAX architecture and describe how MAX EPLDs overcome these deficiencies. Additional technical information is provided in references listed at the end of this Product Information Bulletin. The reader should be familiar with the general characteristics and terminology of PLD technology.

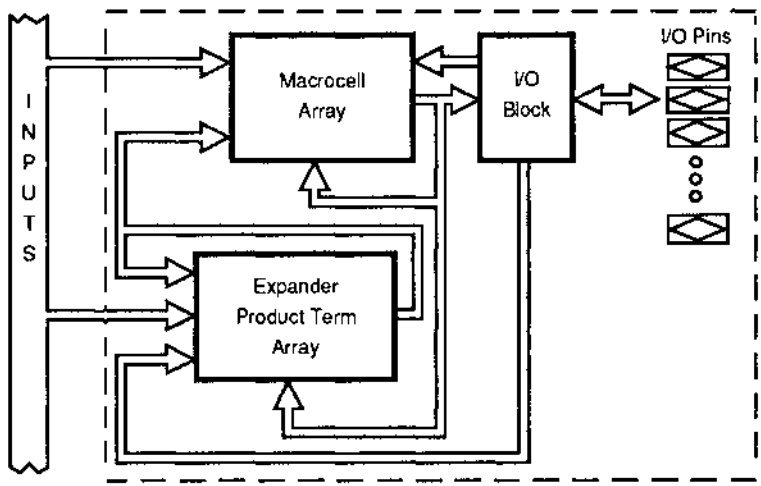
Basic MAX Architecture

Small array performance, large array density.

At the top level, the MAX approach is based on the concept of a small, high-performance, flexible logic array module called a Logic Array Block (LAB).

Smaller members of the MAX family (EPM5016, EPM5024, and EPM5032) have a single LAB. In these devices, all logic signal sources and destinations are fully interconnected directly on the device within the LAB, as shown in Figure 2.

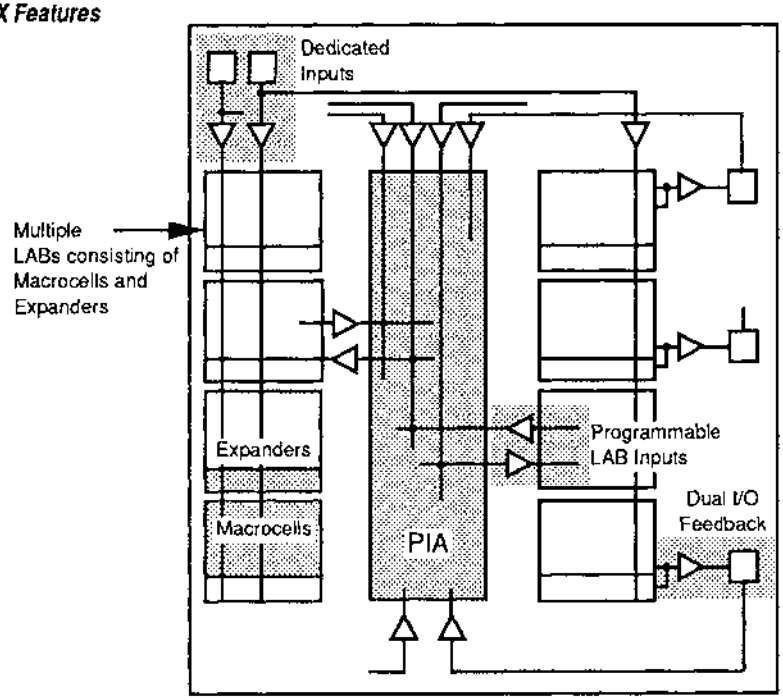
Figure 2. LAB Block Diagram



In larger MAX devices (EPM5064, EPM5128, EPM5130, EPM5192), multiple LABs are linked together via a dedicated, programmable interconnect network called a Programmable Interconnect Array (PIA). The PIA permits any signal source to reach any destination on the chip without routing constraints. This structure allows interconnection of many LABs, achieving large array density with the high performance of small arrays.

The PIA, shown in Figure 3, acts as a programmable highway between all logic functions on the chip. Unlike masked or programmable gate arrays with routing "channels," the PIA provides a crosspoint switch for logic communication, eliminating possible routing bottleneck problems. It also yields a predictable uniform delay. Only a single, fixed-array delay is incurred in the interconnection. Moreover, the delay is substantially smaller than the macrocell delay. The variable and cumulative delays of gate arrays are eliminated.

Figure 3. Key MAX Features



The ability to interconnect all points ensures rapid, automatic design completion on low-cost, accessible, PC-based workstations. Typical MAX designs can be routed in minutes. In contrast, high-density, programmable gate array designs can take hours or days, and in many cases require significant manual intervention. Furthermore, incremental, additive delays between various points can cause debilitating skew and glitch problems, which require additional—often multiple—iterations of the design.

The MAX Macrocell

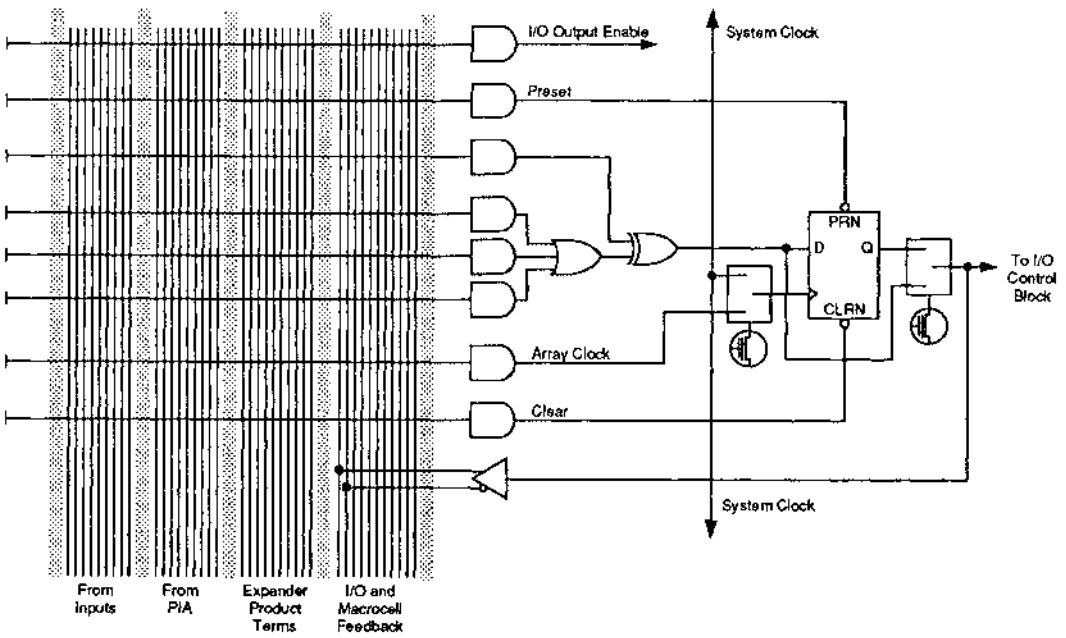
PLA flexibility with PAL economy and performance.

Formerly, PLDs have been classified as either PLA (programmable-AND/programmable-OR) or PAL (Programmable-AND/fixed-OR) devices. The former offers greater logic flexibility, whereas the latter has faster input/output delays and is easier to use. However, in some applications, the PAL structure can be inefficient due to fixed allocation of product terms (p-terms) per output. Typically, 70% of designs require 3 or fewer p-terms. To accommodate as many as possible of the remaining 30%, existing PAL devices and EPLDs use groups of 8 or more p-terms. Thus, at least 5 p-terms are wasted most of the time .

Inside the MAX Logic Array Block are groups of p-terms feeding a sequential logic element. These groups are called macrocells. Each LAB contains between 16 and 32 macrocells. The macrocell in MAX has been designed to combine the flexibility of the PLA device with the speed of the PAL device while eliminating the silicon inefficiency of the latter. Three fixed p-terms are assigned to each MAX macrocell. These are ORed together to drive one input of an XOR (exclusive-OR) gate. The other input of the XOR gate is driven by another product term from the logic array. See Figure 4.

When a logic function requires more than the three fixed p-terms, additional p-terms may be allocated to any macrocell from a pool of uncommitted single p-terms, called Logic Expanders, contained within the LAB.

Figure 4. Macrocell Block Diagram



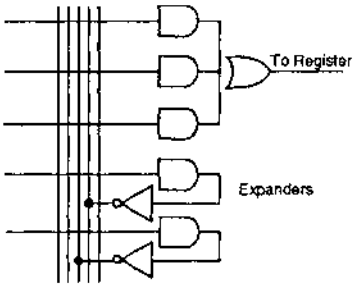
Logic Expanders

Logic where it is needed.

Expanders may be viewed as single product terms with inverted outputs feeding back into the LAB. Expanders provide the PLA-like flexibility of the MAX macrocell. See Figure 5.

Inputs available to the expanders are the same as those for the macrocells in the LAB. Expanders may be distributed to those macrocells that require more than 3 p-terms to implement a desired logic function. The number of fixed p-terms and expanders provided in each LAB has been statistically determined by analyzing hundreds of typical 7400-series TTL and CMOS logic designs as well as PLD-based system designs. Up to 32 expanders are available in an LAB, i.e., all 32 p-terms may be allocated to a single macrocell, or they may be spread among all macrocells. Since expander outputs are available to all macrocells in the LAB, common logic functions may be generated once and used by multiple macrocells. Most importantly, expander logic propagation delays are substantially shorter than macrocell delays for best performance.

Figure 5. Logic Expanders



The expander array can be viewed as a "sea of AND gates" that may be used for sequential and combinatorial functions. Since expanders have inverting outputs, they may be cross-coupled to generate an RS-latched element. Six expanders can be used to create buried, edge-triggered D-type flip-flops, effectively increasing the register count by 33%. When asynchronous latches can be used, this capability can double the total available storage capacity of a MAX device.

This buried flip-flop feature is extremely useful for complex state machine designs. It may be used to hold internal state variables, saving the fixed macrocell flip-flops for decoding of states to the output pins.

Multi-Mode Macrocell Flip-Flop

Latch, D-, JK-, RS-, or T-type.

The basic sequential element in the macrocell may be programmed to operate in either a flow-through latch or edge-triggered (D-type flip-flop) register mode. Flow-through latching provides minimum input-to-output delays for speed-critical applications such as chip select decoding. Edge-triggered operation guarantees glitch-free output generation for applications such as synchronous counters or state machines. A combinatorial output function is realized by bypassing the register.

The output of an XOR gate drives the input to the macrocell's D-type flip-flop. This XOR allows selective inversion of the logic function for polarity control or logic minimization, in addition to direct generation of parity trees, adders, etc. It may be also be used in association with the D-type flip-flop element to emulate JK-, RS-, or T-type functions.

The dedicated sequential elements in the macrocells combined with those obtainable from the expanders give the MAX architecture three to four times more storage capacity than a conventional PAL structure in the same size package.

Flexible Macrocell Control

True emulation of TTL functions.

Each macrocell provides individual asynchronous Preset and Clear, allowing for implementation of asynchronous load counters and true emulation of all popular 7400-series TTL sequential functions. Earlier PLDs and programmable gate arrays do not offer this feature, and designs have to be reworked to fit the devices' more rigid architectures.

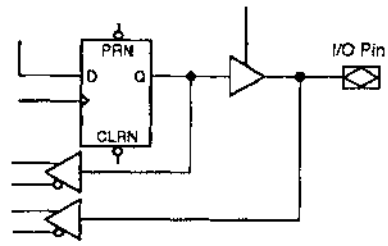
Flip-flops can be configured for synchronous or asynchronous clocking. Synchronous clocking provides a direct connection from a dedicated clock pin minimizing clock-to-output delays. Asynchronous clocking provides the flexibility to clock flip-flops individually from logic. The logic expander array may be used to generate multiple p-term controls for the Clock, Output Enable, Preset, and Clear functions. In contrast, most PLDs have only a single control that consumes an additional macrocell to create relatively simple gated logic functions.

Dual Feedback on All I/Os

All registers can be buried.

In most PLD architectures, when an I/O pin is used as an input, the macrocell associated with that pin cannot be used and is wasted. In the MAX architecture, the I/O pins and macrocell resources (e.g., registers) are decoupled. By providing independent I/O pin and macrocell feedback paths into the array, as shown in Figure 6, all registers can potentially be buried, while I/O pins are used as general inputs to device logic. For example, the EPM5128 may be configured as an 8 input/52 output function or as a 59 input/1 output function without reducing the effective number of register bits available.

Figure 6. Dual Feedback



Sub-Micron Technology

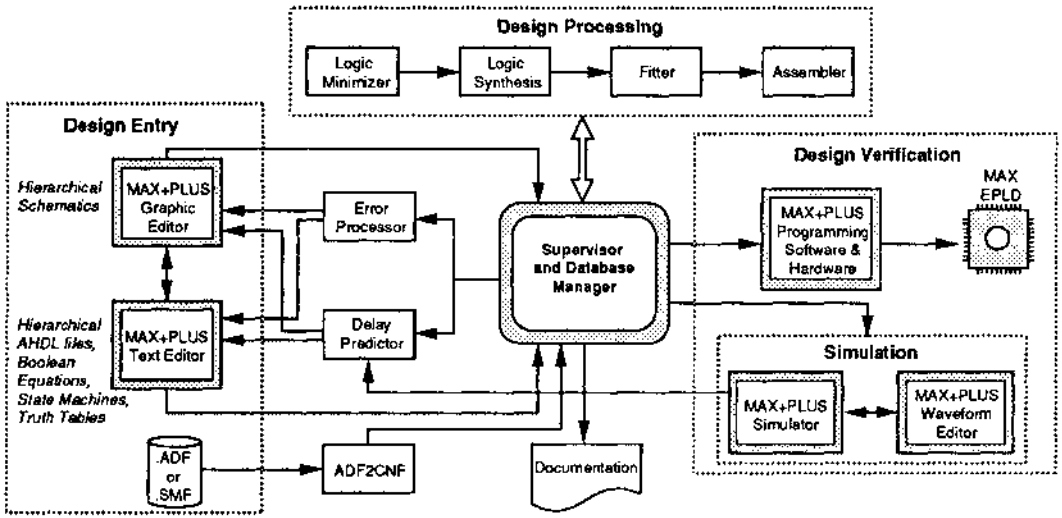
The MAX family is based on a 0.8-micron, double-metal CMOS EPROM technology. The first 8 members of the family, with macrocell counts from 16 to 192, will be available in packages with pin counts ranging from 20 to 100 leads, in DIP, chip carrier, quad flat pack (QFP), and PGA versions where appropriate. They will be available in both UV-erasable/reprogrammable ceramic devices as well as in economical one-time-programmable (OTP) plastic units.

Performance characteristics of this technology are excellent. Internal operating frequencies are 40 to 100 MHz depending on device density. On the EPM5128, 30-ns delays from dedicated input to device output are obtainable, and 15-ns delays on the EPM5016. The EPM5016 also features flip-flop clock-to-output delays of 9 ns. Unlike other high-density PLD technologies (such as programmable gate arrays) that quote extreme f_{MAX} numbers, which must be derated for a real application by two to ten times, MAX performance is realistic and predictable.

MAX+PLUS

Harnessing the power of the MAX architecture is easy with the aid of Altera's sophisticated CAE tools. To support designs for MAX devices, Altera provides MAX+PLUS, a new generation of PC-based PLD design software. MAX+PLUS offers a comprehensive front-to-back design system for the entire MAX family. Hierarchical design entry, automatic logic minimization, device fitting (analogous to automatic place-and-route), timing simulation, and programming support are provided. See Figure 7.

Figure 7. MAX+PLUS



The MAX+PLUS user interface is graphics-based. TTL macrofunctions with SSI and MSI symbols provide a familiar design entry, while the Altera Hardware Description Language (AHDL) with optimized state-machine syntax provides powerful text design entry. Multiple menus offer access to various MAX design processing and simulation sub-systems. Errors discovered during design processing are reported and highlighted in the appropriate schematic or text file for easy debugging.

The MAX+PLUS Compiler uses sophisticated logic synthesis and minimization algorithms to get the most out of the silicon. Extraction of common logic factors to minimize expander consumption, recognition of primitive XOR functions, and multi-level logic compaction are a few of the techniques used. The user need not worry about "pre-packing" logic: the software provides the optimization.

Design fitting in MAX+PLUS is analogous to gate-array place-and-route. However, the PIA eliminates the worry that the software may run out of

routing channels. As a result, there are no functional restrictions on the connections between individual logic blocks. Once the design is fitted, the software generates a device programming file to specify the final design template for the target EPLD.

Timing simulation with graphic waveform output is integrated into MAX+PLUS. Real MAX performance can be predicted with 100-picosecond resolution. To evaluate a design, the MAX user can simulate it rapidly on the PC, or program a device and insert it into the breadboard.

This table summarizes the features of MAX-optimized PLD architecture:

Feature	Benefits
Multiple Array Matrix Architecture	Large array density with fast, small array performance High logic fan-in (>80) and fan-out (>128)
Logic expanders	No wasted logic product terms (100% utilization is possible) Additional sequential logic capacity
Programmable Interconnect Array	Efficient on-chip routing with predictable delays
Macrocell flip-flops	Programmable D,T, JK, SR types
Asynchronous Preset and Clear	True emulation of 7400-series TTL
Multiple product term control of Clock, Output Enable, Preset, Clear	Reduced macrocell consumption and reduced logic delays
Dual I/O feedback	All registers and pins may be used independently

The EPM5000 series of MAX EPLDs is the first complete, high-performance CMOS logic family intended as a general-purpose, high-integration solution to a wide range of applications. Through exploiting the benefits of user-configurable technology, and through multiple source licensing agreements on both development tools and silicon production, Altera aims to establish MAX as the industry-standard logic family of the 1990s.

References

Cole, Bernard C. "Altera Pushes EPLDs." *Electronics* (May 12, 1988).

"Dreifache Komplexität und Doppelte Geschwindigkeit für EPLDs." *Design und Elektronik* (5. Juli 1988).

Faria, Don F. and Robert K. Beachler. "MAX EPLDs Provide Solutions to Both Gate Intensive and Register Intensive Applications." *Professional Program Session Record 31, Electro/88* (May, 1988).

Halleux, Patrick de. "Densité triplée et performance doublée, pour les EPLD d'Altera." *Electronique Industrielle* No. 145 (January 6, 1988).

Lytle, Craig. "MAX+PLUS Eases High-Density EPLD Design Entry." *Professional Program Session Record 14, Electro/88* (May, 1988).

"Multiple Array Matrix High Density EPLDs." *Altera Advanced Information Data Sheet* (1988).

MAX

A Manager's Perspective

January 1990 | Product Information Bulletin 5

1

Introduction

Today, more than ever, managing the development of a digital system is a challenge. From product specification to production transfer, the development strategy must emphasize performance, timeliness, and cost-effectiveness. A major factor in maintaining a product's competitiveness in the marketplace is the selection of state-of-the-art logic design technology—a technology that can satisfy the customer's logic integration needs and time-to-market requirements, as well as quickly accommodate changing system specifications and product features.

"...your development strategy must emphasize performance, timeliness and cost-effectiveness."

Altera's new Multiple Array MatriX (MAX) family offers a combination of logic density, performance, and flexibility that's tough to beat. The EPM5000 MAX series of EPLDs comprises 8 devices with a range of densities to meet logic application needs at system clock rates up to 66 MHz and pipeline data rates of over 100 MHz. A MAX solution is available for every problem—be it a 20-pin device ideal for high-speed address decoding and state machines, or a a 100-lead device that accommodates several thousand gates. And every MAX family member has a fully programmable architecture that efficiently fits any mix of TTL functions without the design and routing headaches commonly associated with fixed and programmable gate arrays.

Design tools are the critical link between designer and chip. The advanced MAX+PLUS system supports multiple design entry methods, including familiar TTL schematic design entry, so a designer won't have to learn a whole new language. The MAX+PLUS Compiler uses logic synthesis that automatically translates a high-level design into a MAX programming file in minutes. So, a programmed EPLD can be in the system just moments after the design is completed.

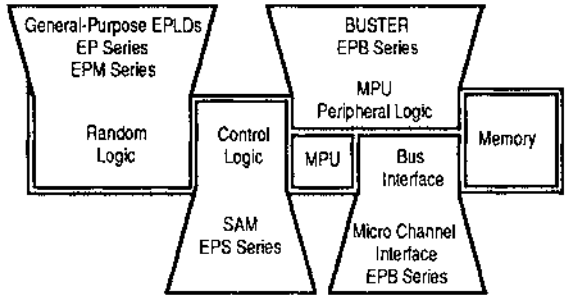
This Product Information Bulletin addresses questions commonly asked by management about the MAX product family and Altera. The MAX architecture's technical features and benefits are summarized in *Product Information Bulletin 4 (MAX Family Architecture)*.

Questions & Answers

◆ Who is Altera?

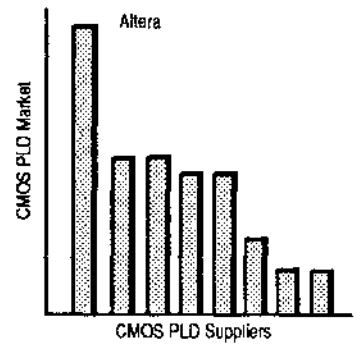
Over the past six years, Altera Corporation has pioneered the concept of large-scale, erasable, programmable logic devices—EPLDs. EPLDs and advanced PC-based design tools are Altera's only business (see Figure 1).

Figure 1. Altera User-Configurable Logic Families



According to Dataquest, a market research group, Altera is the #1 supplier of large and small CMOS programmable logic devices in the world today (see Figure 2). General-purpose EPLDs effectively solve a wide range of logic design problems. Altera has extended the EPLD concept to special-purpose programmable logic devices called function-specific EPLDs. These include the EPS-series of Stand-Alone Microsequencer (SAM) components for high-performance control logic and the EPB2001 Micro Channel Interface device. These devices provide optimized logic functions to solve specific application problems.

Figure 2. CMOS Suppliers



To establish the EPLD concept as an industry standard, Altera Corporation, based in San Jose, California, has entered second-source relationships with Cypress Semiconductor, Intel Corporation, Texas Instruments, and Wafer Scale Integration (WSI).

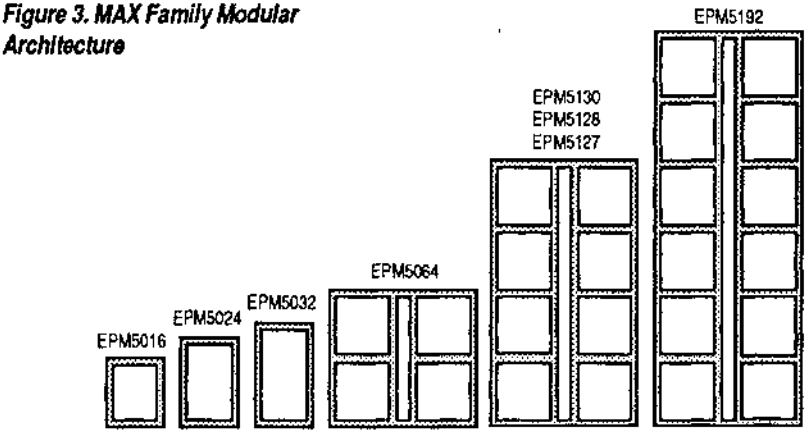
◆ Why will MAX replace TTL and CMOS logic families, as well as PLA and PAL devices and many ASICs?

The MAX family (shown in Figure 3) will become the new standard logic design technology for the 1990s because it offers:

- ❒ **User-Configurability**—Design to silicon in minutes
- ❒ **Performance**—Real performance compatible with today's high-performance microprocessors
- ❒ **Integration**—Packing dozens of components into a single chip in through-hole and surface-mount packages

- ❒ **I/O Options**—Providing pin counts from 20 to more than 100 pins in convenient density increments
- ❒ **Ease of Use**—Familiar, programmable logic-array based architecture gives predictable in-system performance. MAX+PLUS PC-based design tools, using logic synthesis, maximize designer productivity.

Figure 3. MAX Family Modular Architecture

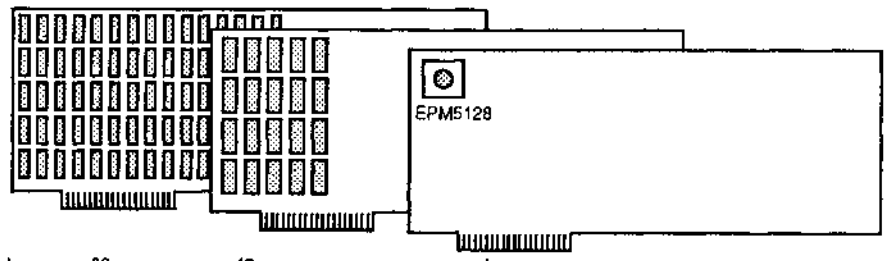


Macrocells	16	24	32	64	128	192
Max. Flip-Flops	21	32	42	84	168	252
Max. Latches	32	48	64	128	256	384
Pins	20	24	28	44	100/68/44	84

◆ How efficient is MAX? How much logic can a designer put into a chip?

The MAX family offers a range of device densities. The 68-pin, J-lead EPM5128 can integrate the equivalent of 20 PAL devices or a typical mix of 75 MSI/SSI TTL chips into a single device; e.g., as many as 128 74151 multiplexers, or 32 four-bit 74161 counters can be designed into a single device. On the other end of the spectrum, the EPM5032 can integrate 4 to 6 PAL devices into a single 28-pin DIP package (equivalent to 10 to 15 TTL components). Other MAX family members offer density and I/O counts above and below these two limits. See also Figure 4.

Figure 4. MAX Board Density

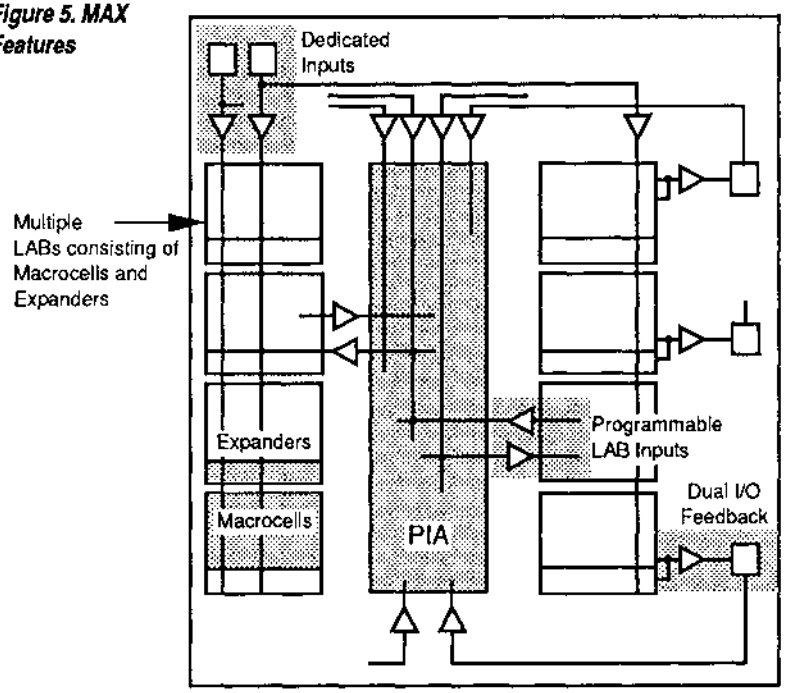


Board Area (sq. in.)	38	12	1
Number of Components	75	20	1
Type of Component	TTL	20-Pin PAL Devices	MAX

"...MAX's Programmable Interconnect Array (PIA) provides 100% routability."

As Figure 5 shows, the MAX architecture is extremely efficient. Unlike programmable gate array technologies, which have severe logic interconnection (routing) limitations that stretch design time and reduce gate utilization dramatically, MAX's Programmable Interconnect Array (PIA) provides 100% routability. Other MAX architectural innovations, such as logic expanders, programmable register control terms, and decoupled register-I/O, guarantee maximum use of the chip's resources in any application. In addition, the MAX macrocell includes options such as asynchronous clocking, and Preset and Clear logic inputs for all registers to directly implement TTL functions. For additional information, refer to *Application Note 17 (Integrating an Intelligent I/O Subsystem)*.

Figure 5. MAX Features

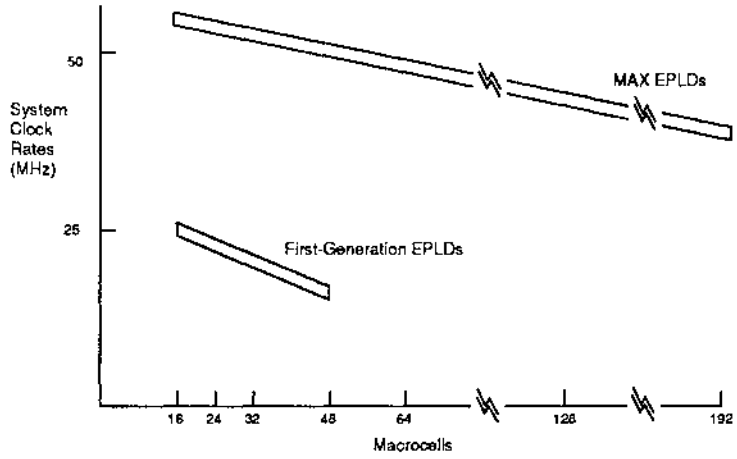


◆ What system level performance can be expected from MAX devices?

Logic clock rates quoted for MAX devices are actual worst-case clock frequencies obtained for basic logic building blocks (such as counters, shift registers, etc.) integrated into the chip. Unlike f_{MAX} flip-flop toggle frequencies, which are commonly quoted for other technologies and have no direct correlation to true system performance, practical logic blocks can be built that operate at these clock rates. Performance of MAX devices,

even in extremely complex configurations, is compatible with today's 25-MHz or faster processors. See Figure 6.

Figure 6. System Performance vs. Density



For example, basic building block functions that do not require logic expanders can run at 76 MHz in an EPM5032. A complex state machine implemented in the EPM5032 and using logic expanders, runs easily at 47 MHz. An EPM5016 delivers 15-ns combinatorial logic for fast decoders and muxes.

◆ Will extended-temperature range MAX devices be available?

Altera offers its EPLDs in a choice of device operating temperature ranges: Commercial (0° C to +70° C), Industrial (-40° C to +85° C) and Military (-55° C to +125° C). MAX devices will be offered in similar grades. In addition, MAX devices will be available fully screened to the requirements of Mil Standard 883B, Level C.

◆ What are our packaging options with EPLDs?

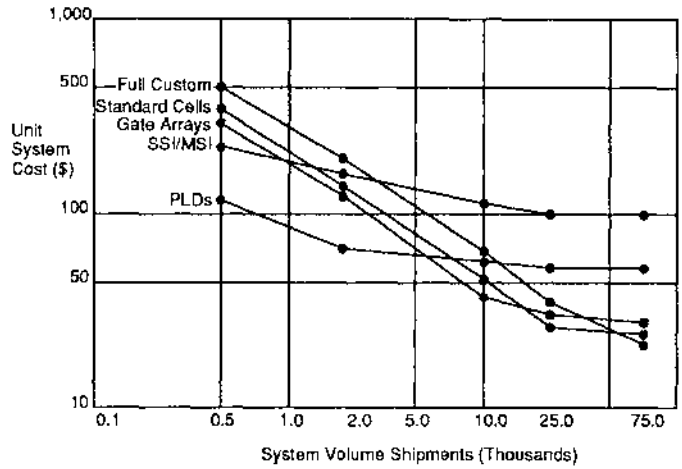
MAX EPLDs are currently available in a variety of I/O counts and package styles from 20 to 100 pins. UV-erasable, reprogrammable ceramic windowed parts are available in dual-in-line (DIP), pin-grid-array, and J-lead chip carrier packages. Small-Outline (S.O.) and quad-flat-pack (QFP) device options are also under development. Plastic one-time-programmable (OTP) devices will be offered in DIP and surface-mount package styles for lowest production costs. As a result, whether a customer's board technology is through-hole or surface-mount, MAX EPLDs fit.

◆ How much does MAX cost relative to other approaches?

"...getting to market six months earlier can triple the profits of a product..."

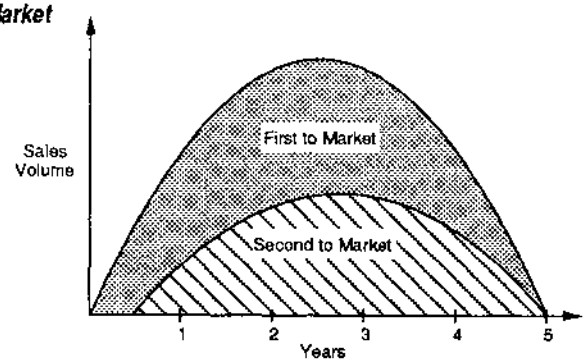
Studies by Integrated Circuit Engineering, an independent semiconductor market research firm, indicate that for system volumes below 10,000 units per code, programmable logic devices are the most cost-effective technology—less costly than TTL, less costly than gate arrays, less costly than any other logic technology. See Figure 7.

Figure 7. PLD Cost-Effectiveness



Also, based on a study by McKinsey and Company, a market research firm, getting to market six months earlier can triple the profits of a product over its life (see Figure 8). So the user-configurable approach of MAX, which allows a designer to get his design into his system the same day, provides the lowest overall cost and highest profit approach for small- to medium-volume systems.

Figure 8. Time to Market



◆ **What are the benefits of MAX EPLDs versus RAM-based programmable gate arrays?**

Some of the benefits of MAX technology, when compared to RAM-based programmable gate arrays, are:

- ❒ **Easy-to-understand logic-array architecture**—Unlike exotic programmable gate array architectures, MAX uses logic-array structures familiar to PLD/TTL system designers.
- ❒ **Stand-alone chips that require no support devices**—RAM-based programmable gate arrays typically require additional external memory chips and waste device pins for initialization.
- ❒ **Non-volatile technology**—MAX's non-volatile CMOS EPROM technology means that MAX EPLDs can be used reliably in critical portions of system designs. They do not require initialization during power-up and do not lose their logic configuration in the event of power supply "brown-out."
- ❒ **Efficient implementation of combinatorial and registered functions**—Unlike programmable gate arrays that implement arithmetic and combinatorial logic poorly, the MAX architecture supports both types of logic efficiently. This ability translates into two to three times the performance in real applications.
- ❒ **A range of device sizes for all design problems**—MAX chips will range from fast 20-pin devices (standard PAL complexity) to 100-lead chips, with up to 384 registers and latches. Programmable gate arrays, on the other hand, provide no support for small to medium logic blocks common in today's system design.
- ❒ **Programmable Interconnect Array (PIA) for guaranteed routability**—Unlike programmable gate arrays with limited signal routing channels (as illustrated in Figure 9), MAX's PIA is programmed by the MAX+PLUS software to route designs without manual intervention. And MAX+PLUS automatically compiles complex logic designs in minutes, unlike programmable gate arrays that can take hours or days.
- ❒ **State-of-the-art design tools**—Programmable gate array software is universally known to be hard to use, requiring a long training process and designer "hand-holding" during compilation, and operating at a very low level. MAX+PLUS software (shown in Figure 10) speaks the designer's language—with Boolean equation, state machine, and TTL schematic design methods—and requires minimal knowledge of the inner workings of MAX. Manual routing of the design is never required.

So MAX is easier to use, gives higher real performance, and more effective logic utilization than programmable gate arrays.

Figure 9. Routing Schemes

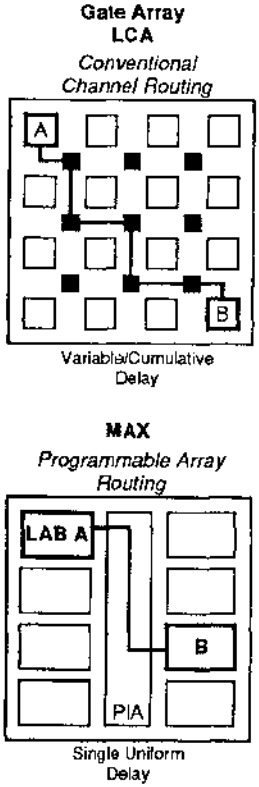
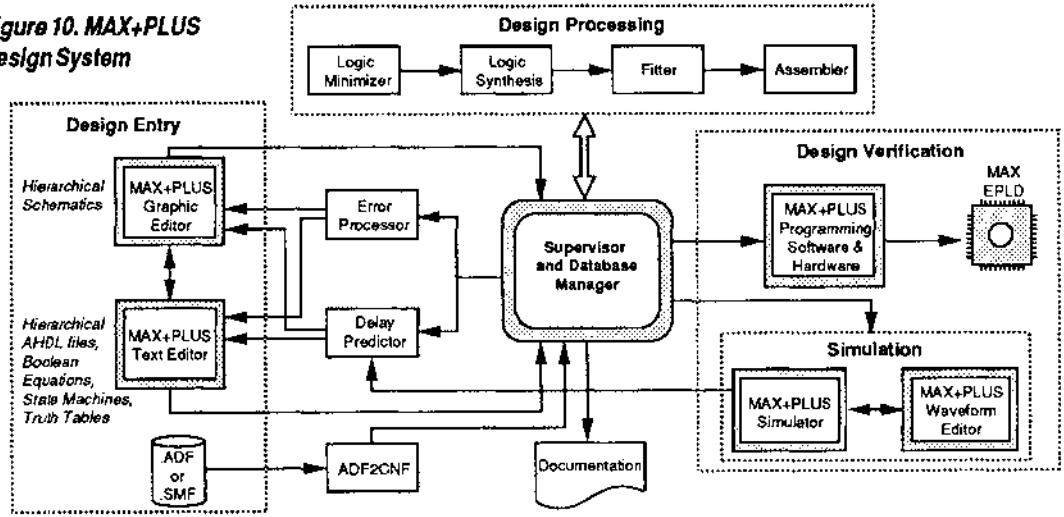


Figure 10. MAX+PLUS Design System



◆ How reliable is MAX EPLD technology?

MAX EPLD technology is based on EPROM memory technology, which has proven highly reliable over the past 15 years. While the application of this technology to programmable logic is relatively new, the underlying principles and reliability are well demonstrated. Millions of EPROM memory components have been shipped and used in systems for years, demonstrating their reliability. Millions of CMOS EPLDs have similarly been shipped over the past five years with the same positive results.

◆ Are there any second sources for MAX?

Cypress Semiconductor, a leading manufacturer of high-performance CMOS ICs, second-sources the MAX product family, guaranteeing continuity and variety of supply to MAX users.

◆ Why does Altera provide its own design tools instead of relying on third-party logic compilers?

"...it is possible to drive a wood screw into a block of wood with a hammer..."

Providing leading-edge IC technology is not enough. Design tools must be able to exploit the capabilities of the silicon to maximize designer productivity. By way of analogy, it is possible to drive a wood screw into a block of wood with a hammer, but a screwdriver does a faster job and results in a stronger bond. The screwdriver and screw are complementary and give the optimum result. Similarly, general-purpose design tools frequently cannot use new device features effectively. Altera's MAX+PLUS design system supplies all the capability customers have come to expect in a state-of-the-art CAE system, and supports all features of the MAX family fully (see Figures 11 and 12). For example, Altera's MAX+PLUS system provides the following advantages:

- **Hierarchical schematic capture**—Supports a structured approach to large design management.
- **Altera Hardware Description Language (AHDL) entry**—Supports hierarchical behavioral logic descriptions using group, conditional, arithmetic, and relational operators.
- **User-defined macro libraries**—Allow the standardization of frequently used custom functions within a project.
- **Timing simulation**—Models actual chip response with 100-picosecond resolution.
- **Advanced logic minimization and synthesis**—Maps TTL designs onto MAX architecture for maximum efficiency/compilation ease.
- **Interactive delay prediction**—Provides quick point-to-point delay prediction in the logic design based on actual device timing parameters.
- **Automatic error location**—By closely linking the MAX logic compiler and language and graphic front ends, design entry errors are reported quickly to minimize debug time.

Figure 11. MAX+PLUS Graphic Editor Screen

The hierarchical Graphic Editor provides a multi-windowed, menu-driven environment.

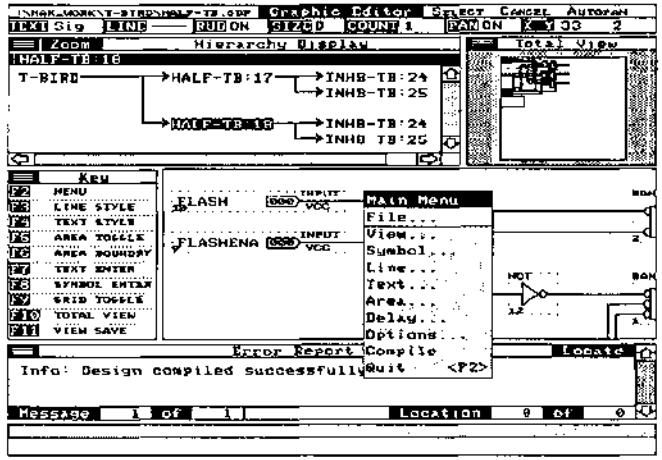
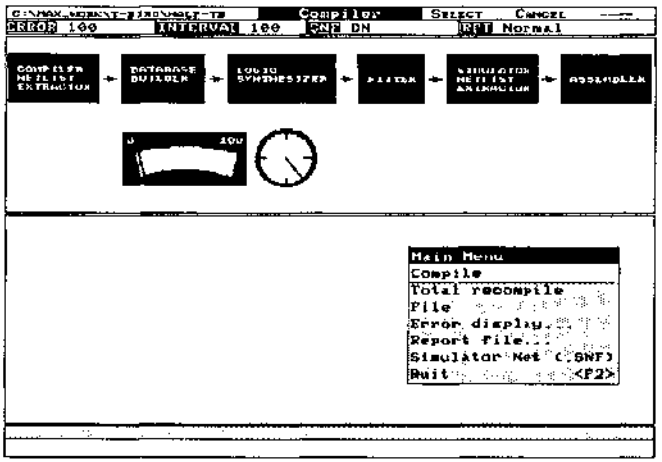


Figure 12. MAX+PLUS Compiler Screen

The Compiler uses minimization, logic synthesis, and heuristic fitting algorithms to place designs into MAX EPLDs.

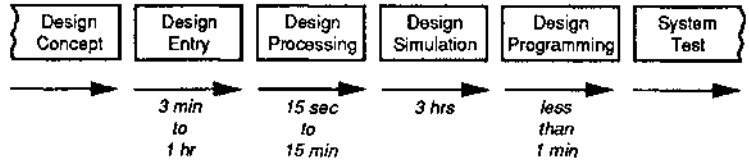


◆ **How long does it take to learn the MAX+PLUS tools? How quickly can a typical design be implemented?**

"...6 hours after receiving MAX+PLUS, the manager had entered and successfully fit his design into an EPM5128."

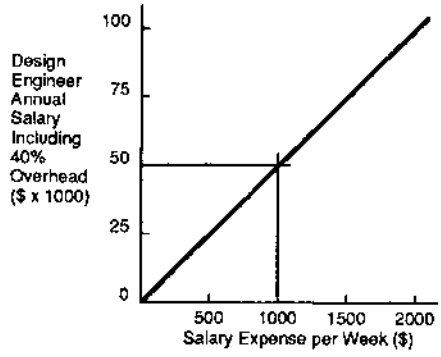
The learning curve for Altera's tools, as with any system, is a function of the designer's previous CAE tool experience. For a user familiar with a PC and A+PLUS, MAX+PLUS designs can be generated by the end of the first day. For the inexperienced PC user, a typical learning curve might stretch to a week. MAX+PLUS software has been structured to minimize surprises and automate all design compilation steps once the design is entered. Figures 13 and 14 illustrate the typical design cycle and cost savings.

Figure 13. EPLD Design Methodology



This ease of use is in stark contrast to the difficulties typically encountered in using more primitive programmable logic design tools. Point-by-point comparisons have shown that the MAX approach is easier to learn and use than alternatives such as programmable gate arrays. In one instance, a design manager assigned a programmable gate array design to a junior engineer. The design was targeted for a device specified as a 9000-gate programmable gate array, but in theory would use only 60% of the device's resources. After three months, the design was still not implemented due to routing problems. The manager took over the project himself, and was unsuccessful in improving matters after a week of constant effort. The programmable gate arrays's inherent architectural weakness had limited its gate utilization to only 30%. At this juncture, the manager turned to the Altera approach: six hours after receiving MAX+PLUS software, the manager had entered and successfully fit his design into an EPM5128. Quite a contrast to three man-months of wasted effort.

Figure 14. CAE Investment Saves Time & Money



As an added support feature, the Altera Applications Hot Line is available to answer any device or tool questions in real-time over the telephone.

Moreover, the Electronic Bulletin Board Service, accessible via modem, keeps Altera users up-to-date on new EPLD applications tips and news.

◆ Why does Altera charge for its design tools?

"...an investment of just under $5000 in MAX+PLUS CAE tools can save weeks of engineering effort..."

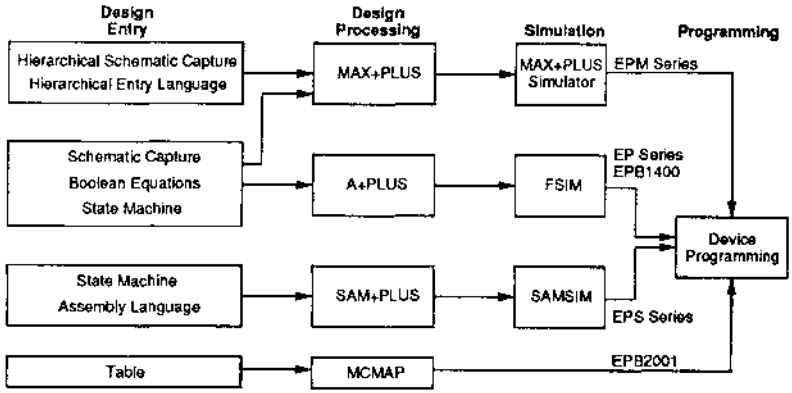
Providing timely support of new device architectures such as MAX takes considerable investment on Altera's part. Fully 50% of Altera's development efforts are in new design tools as opposed to ICs. To develop and maintain these productivity-enhancing tools, Altera must charge a fair price. Second-sourcing of Altera components also means that Altera cannot always recoup its tool development investment from component sales and must directly charge for them. Placed in perspective, the expense of a design engineer's salary and overhead can easily amount to $100,000 or more per year. An investment of just under $5,000 in MAX+PLUS CAE tools can save weeks of engineering effort, and pays for itself on the first project!

◆ Does the MAX+PLUS design system obsolete previous EPLD design tools (e.g., A+PLUS) and require a new learning curve?

No. As shown in Figure 15, the MAX+PLUS design system offers a superset of the capability found in A+PLUS. Most A+PLUS users are up to speed on MAX+PLUS in a day, entering schematics and compiling designs. The pop-up menu/mouse-driven interface is easy to use. New features such as simulation and waveform editing are intuitive and easy to learn.

A+PLUS is still recommended for the EP-series (EP610, EP910, etc.) of zero-power EPLD devices, which draw microamps of supply current at low input frequencies, and are ideal for power-sensitive applications such as hand-held instrument and telecommunications. So the EP- series is an excellent complement to the EPM-series, where power is the issue. For convenience, MAX+PLUS accepts EP-series designs generated with A+PLUS, expanding the designer's options with minimal effort.

Figure 15. EPLD Design Environment



◆ **How do we integrate Altera design tools into our workstation strategy using Mentor, Valid, VIEWlogic, Orcad, or other system-level design tools?**

Altera's design tools support industry-standard EDIF netlist generation, while EDIF netlist input is under development. Leading workstation vendors and third-party tool vendors have committed to support EDIF as well, allowing efficient tool linkage. In addition, third-party vendors such as Logic Automation Inc. have developed interfaces, device models, and other products that allow Altera EPLDs to be integrated into system-level design tools. Altera will encourage similar support, so that the designer can harness the horsepower of MAX+PLUS logic synthesis with existing design entry and verification tools.

◆ **How do we generate test vectors for MAX EPLDs?**

Special-purpose test vectors are not necessary to ensure the functionality or quality of programmed EPLDs. EPLDs are generically tested, which means that every programmable connection is checked at the factory. All possible faults are pre-tested as part of the standard manufacturing and test flow—a major benefit of erasable programmable technology. As a result, Altera guarantees 100% programming yield of its components on approved programming equipment, unlike fuse-based programmable components in which programmable connections cannot be pre-tested.

To summarize testability criteria:

❒ EPLDs are generically testable.

❒ All EPLDs are 100% tested at factory for functionality, programmability, and performance.

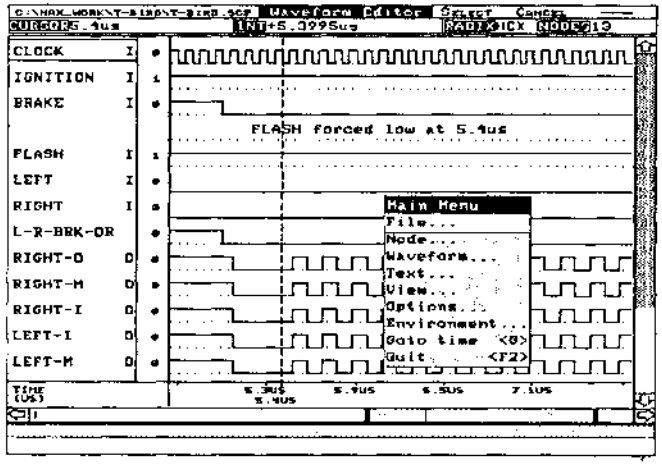
❒ Altera guarantees 100% programmability on certified programmers, thus yielding lower cost and high reliability.

❒ Plastic one-time-programmable (OTP) EPLDs incorporate special test modes to ensure same programming yield as reprogrammable/ windowed parts.

If test vectors are required for system documentation or other purposes, the MAX+PLUS Simulator, a timing simulator, can provide an effective means to predict device response to input vector sequences (see Figure 16). The MAX+PLUS Simulator's output file can be translated to the appropriate tester input format by tester-specific conversion routines.

Figure 16. MAX+PLUS Waveform Editor Screen

The Waveform Editor allows entry and modification of input stimuli and comparison of simulation waveforms.



1

◆ **Can we convert our MAX designs to gate arrays?**

For very high volume designs, or for designs that are stable over a period of years, a gate array might still be an appropriate successor to a MAX design. MAX+PLUS EDIF output can support design transfer at a detailed gate level. In addition, since MAX+PLUS supports TTL schematic entry, higher-level macrofunction translation is very straightforward. No complex format conversions are necessary. MAX schematics can be entered into the target design system of an ASIC vendor, typically in a matter of days. Since the MAX architecture supports features commonly found in TTL functions directly (asynchronous preset and clear, complex clock functions, edge-triggered or flow-through flip-flops, etc.), no functional translation is necessary during the entry process.

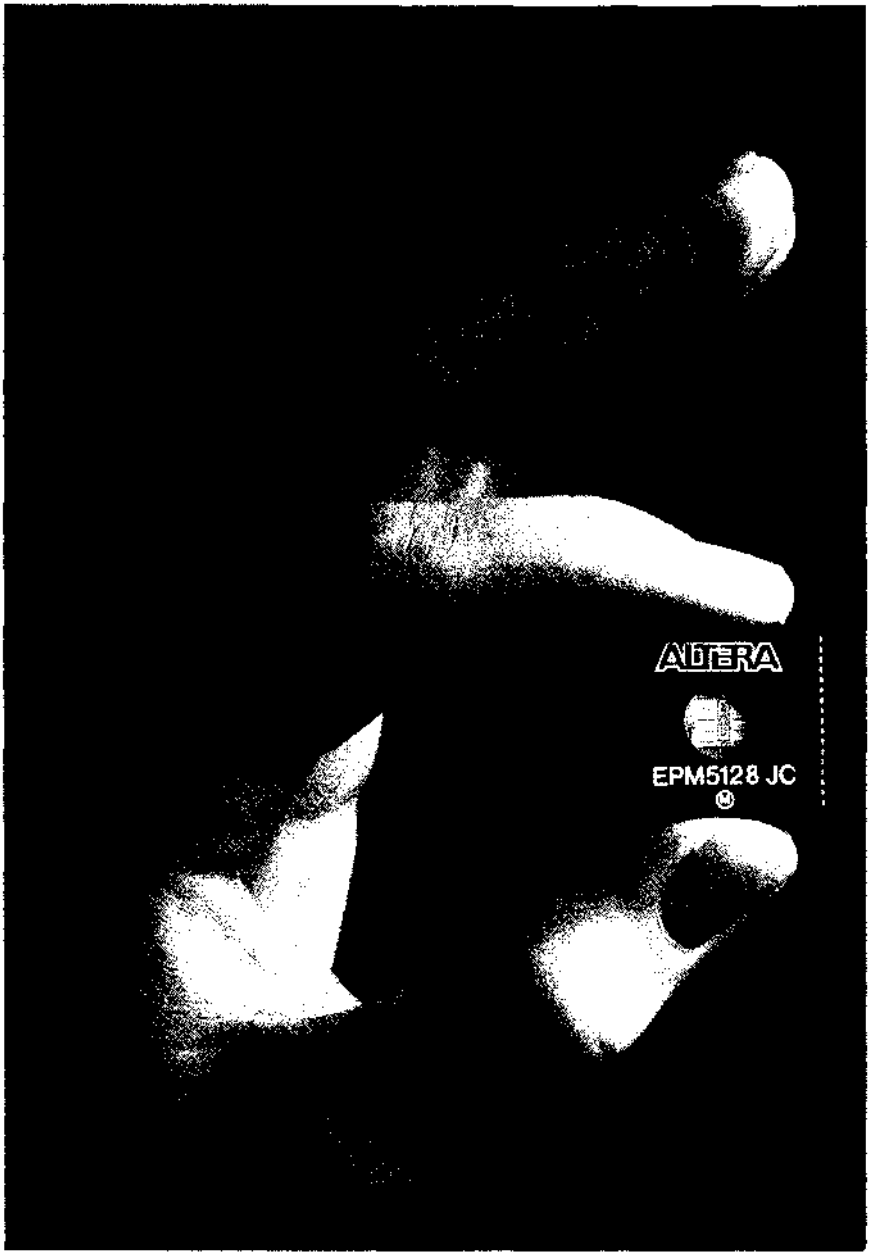
Summary

The MAX family of high-density, high-performance EPLDs has been developed by Altera as a complete set of user-configurable logic building blocks for a wide range of system applications. Through the easy-to-learn, easy-to-use MAX+PLUS set of logic design tools, designers can exploit the maximum capabilities of the programmable MAX chips in minimum time. It is this combination of logic density, speed, and designer productivity that will make MAX the industry-standard logic family of the 1990s.

Contents

Section 2 Data Sheets

2


ALTERA
EPM5128 JC

MAX Family

Device and Software Overview

January 1990 — **Data Sheet**

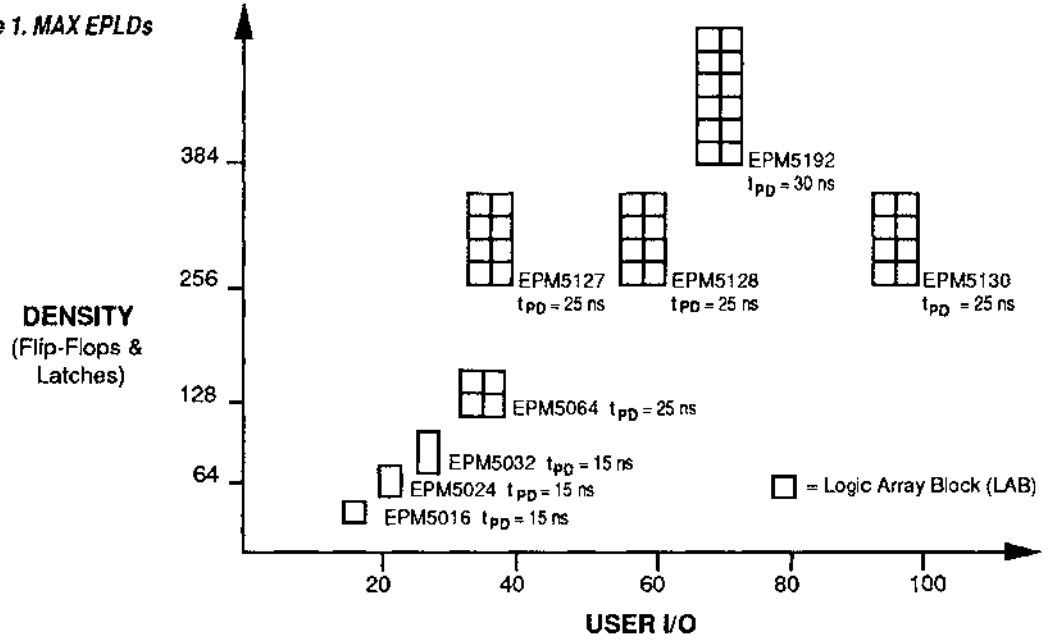
Product Summary

- Complete CMOS EPLD family providing a consistent design solution across a broad range of speed and density requirements
- MAX architecture providing the speed, ease of use, and familiarity of PAL devices with the density of programmable gate arrays
- 15-ns combinatorial delays (t_{PD}), counters as fast as 100 MHz, and pipelined data rates of 100 MHz
- High-density devices integrating up to 384 flip-flops and latches
- Package options from 20-pin DIPs to 100-pin Quad Flat Packs, in windowed, erasable, or one-time-programmable plastic
- Easy conversion to custom-masked devices for very high-volume production
- MAX+PLUS PC-based design tools able to compile large designs in minutes
- EDIF industry-standard workstation and third-party interfaces

Figure 1 shows a graphical representation of device density and the number of I/O pins, as well as the number of Logic Array Blocks (LABs) per device.

2

Figure 1. MAX EPLDs



Family Highlights

- **Multiple Array MatriX (MAX) architecture solves speed, density, and design flexibility problems**
  - Advanced macrocell array provides registered, combinatorial, or flow-through latch operation.
  - Expander product term array automatically provides additional combinatorial or registered logic.
  - Decoupled I/O block with dual feedback on I/O pins allows flexible pin utilization.
  - Programmable Interconnect Array provides automatic 100% routing in devices with multiple LABs.
  - Each macrocell supports synchronous or asynchronous operation by every macrocell, using single or multiple clocks per device.
- **MAX Device Performance**
  - Pipelined data rates to 100 MHz
  - Counters as fast as 100 MHz
  - t_{pd} performance from 15 ns to 30 ns
  - Advanced 0.8 micron CMOS EPROM technology
- **MAX Device Logic Density**
  - 16 to 192 macrocell devices
  - 20- to 100-pin packages
  - 32 to 384 flip-flops and latches
  - More than 32 product terms on a single macrocell
  - Product term expansion on any data or control path
- **MAX+PLUS Design Tools**
  - Design entry via unified, hierarchical schematic capture and Altera Hardware Description Language (AHDL)
  - Fast, automatic design processing with logic synthesis
  - Automatic device fitting, no hand editing needed
  - Hardware and software design verification tools
  - Compiles a 16-bit counter in 34 seconds on a 16-MHz 386 personal computer
- **EDIF reader and writer interfaces to MAX+PLUS provide convenient paths to Dazix, Mentor, Valid, and other workstations.**

Overview

The Altera Multiple Array MatriX (MAX) family of EPLDs provides a user-configurable, high-density solution to general-purpose logic integration requirements. The innovative architecture and state-of-the-art process used by the MAX EPLDs offer LSI density, without sacrificing speed.

MAX architecture can replace large amounts of 7400-series SSI and MSI TTL and CMOS logic. For example, a 74161 counter utilizes only 3% of the 128 macrocells available in the EPM5128. Similarly, a 74151 8-to-1 multiplexer consumes less than 1% of the more than 1,000 product terms available in the EPM5128. Thus, the designer can replace 50 or more TTL packages with just one MAX EPLD. The MAX family is available in a range of densities and packages, shown in Figure 1. By standardizing with a few MAX building blocks, the designer can replace hundreds of different 7400-series and PLD part numbers currently used in most digital systems while gaining the flexibility of a programmable, re-usable technology.

The family is based on an architecture of flexible macrocells grouped into Logic Array Blocks (LABs). Additional product terms, called expander product terms (expanders), are within each LAB. Expanders are used and shared by the macrocells to allow complex functions (more than 32 product terms) to be easily implemented in a single macrocell. A Programmable Interconnect Array (PIA) globally routes all signals in devices with more than one LAB. MAX architecture is fabricated with an advanced 0.8-micron CMOS EPROM process that yields devices with three times the integration density and twice the system clock speed of the other programmable solutions.

All MAX family members use an identical architecture, except for the PIA. Larger devices (EPM5064, EPM5127, EPM5128, EPM5130, and EPM5192) add the PIA to provide high performance and 100% routability for larger single-chip design tasks. This division between PIA and non-PIA MAX devices provides extremely fast parts for small to medium logic design tasks, while continuing to provide high performance when the logic design task grows to hundreds of TTL chip equivalents.

The density and flexibility of the MAX family is complemented by the MAX+PLUS Development System. MAX+PLUS is a PC-based design system that is optimized specifically for MAX architecture. Hierarchical schematic entry is used to capture the design. State machine, truth table and Boolean equation entry mechanisms are also supported with the Altera Hardware Description Language (AHDL). Design methods may be mixed within a design's hierarchy, using the entry method most appropriate for the task. An integrated timing simulator provides full AC simulation and an interactive Waveform Editor which speeds waveform creation and debugging. Optional hardware and software verification tools include the Programmable Logic Synthesis Analyzer (PLSA) Tools for synthesis analysis and transfer to workstation-based board-level simulators, and the Altera Personal Logic Design Verification System (PLDVS) for hardware function test and continuity check programming. See the appropriate software data sheets for additional information.

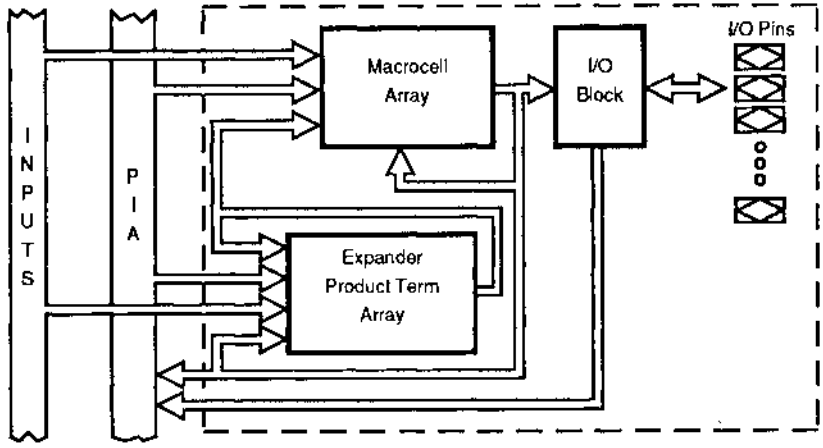
MAX Device Architecture

The architecture for MAX devices consists of five major elements: the Logic Array Block (LAB), the MAX macrocell, expander product terms, the I/O control block, and the Programmable Interconnect Array (PIA). A description of each follows.

Logic Array Block

The Logic Array Block, shown in Figure 2, is the heart of MAX architecture. It consists of a macrocell array, an expander product term array, and an I/O control block. The number of macrocells, expanders, and I/O varies, depending on the device used. Global feedback of all signals is provided in the LAB, giving each functional block complete access to the LAB's

Figure 2. LAB Block Diagram



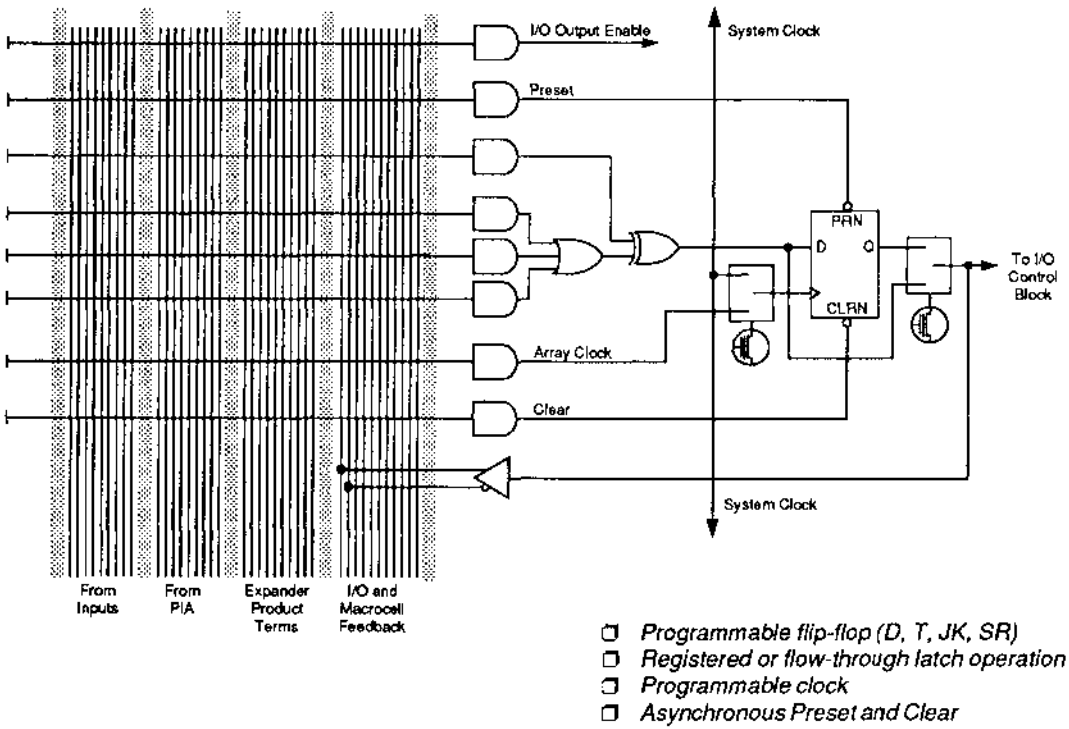
resources. The LAB is fed by a dedicated input bus (PIA), which is used to interconnect devices that contain more than one LAB. The PIA provides sufficient interconnections to fully use all macrocells within a device without routing constraints. Macrocell and I/O pin feedbacks enter the PIA, so that other LABs in the device have access to them. MAX EPLDs that have a single LAB use a global interconnect network, and have no need for a PIA.

MAX Macrocell

Traditionally, PLDs have been divided into PLA (programmable-AND/programmable-OR), and PAL (programmable-AND/fixed-OR) architectures. PAL architecture provides faster input-to-output delays, but can be inefficient due to the fixed product-term distribution. Statistical analysis of PLD logic designs has shown that 70% of all logic functions require at most three product terms. The remaining 30% need anywhere from 4 product terms to as many as 30 in some complex combinatorial or state machine designs.

The macrocell structure of MAX is based on three product terms that are able to add additional product terms when necessary. Each macrocell consists of a product term array and a configurable register (see Figure 3). Combinatorial logic is implemented with three product terms ORed together, feeding an XOR gate. The second input to the XOR gate is also controlled by a product term, providing the ability to control active-high or active-low logic. MAX+PLUS software uses the XOR gate to implement complex mutually exclusive-OR arithmetic logic functions, or to perform DeMorgan's Inversion, reducing the number of product terms required to implement a function.

Figure 3. Macrocell Block Diagram



- *Programmable flip-flop (D, T, JK, SR)*
- *Registered or flow-through latch operation*
- *Programmable clock*
- *Asynchronous Preset and Clear*

2

If more product terms are required to implement a given function, they may be added to the macrocell from the expander product term array. These additional product terms may be added to any macrocell to allow the designer to build gate-intensive logic, such as address decoders, adders, comparators, and complex state machines, without using extra macrocells.

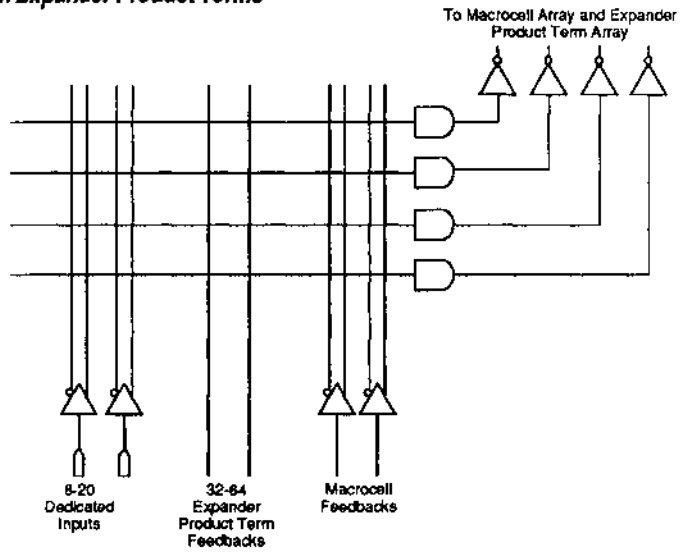
Logic is automatically allocated by the MAX+PLUS Compiler during compilation. Alternatively, the designer can choose specific logic assignments for macrocells and expanders.

The register in a MAX macrocell may be programmed for D, T, JK, or SR operation; it may also be configured as a flow-through latch for minimum input-to-output delays. The register may be by-passed entirely for purely combinatorial functions. In addition, each register supports asynchronous Preset and Clear, allowing asynchronous loading of counters or shift registers, as found in many standard TTL functions. These registers may be clocked with a synchronous system clock or an asynchronous clock from the logic array.

Expander Product Terms

Expander product terms are fed by the dedicated input bus, PIA, macrocell feedback, I/O pin feedback, and other expanders (see Figure 4). Expander outputs can connect to every product term in the macrocell array, allowing expanders to be "shared" by the product terms in the Logic Array Block. One expander may feed all macrocells in the LAB or multiple product terms in the same macrocell. Since these expanders feed the secondary product terms (Preset, Clear, Clock, and Output Enable) of each macrocell, complex logic functions may be implemented without using another macrocell. Likewise, expanders may feed and be shared by other expanders to implement complex multi-level logic and latches. Expanders are normally allocated in parallel to ensure highest performance.

Figure 4. Expander Product Terms

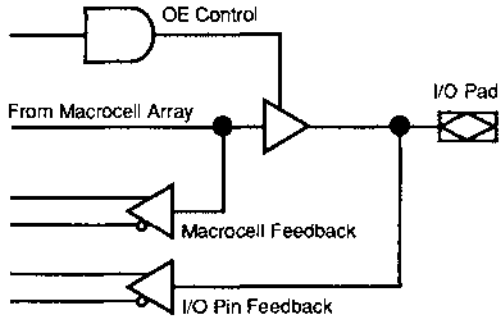


I/O Control Block

In the LAB, the I/O control block (Figure 5) is decoupled from the macrocell array. The tri-state buffer, controlled by a product term, drives the I/O pad. The input of the tri-state buffer comes from a macrocell in the associated LAB. The feedback path from the I/O pin may feed other blocks within the LAB, as well as the PIA.

Decoupling the I/O pins from the flip-flops allows buried registers in the LAB. At the same time, the I/O pin can be preserved as a dedicated input. The I/O pins may be used as dedicated outputs, bidirectional pins, or

Figure 5. I/O Control Block



additional dedicated inputs. Thus, applications requiring many buried flip-flops (such as counters, shift registers, and state machines) no longer consume both the macrocell register and the associated I/O pin, as they did in earlier devices.

Programmable Interconnect Array

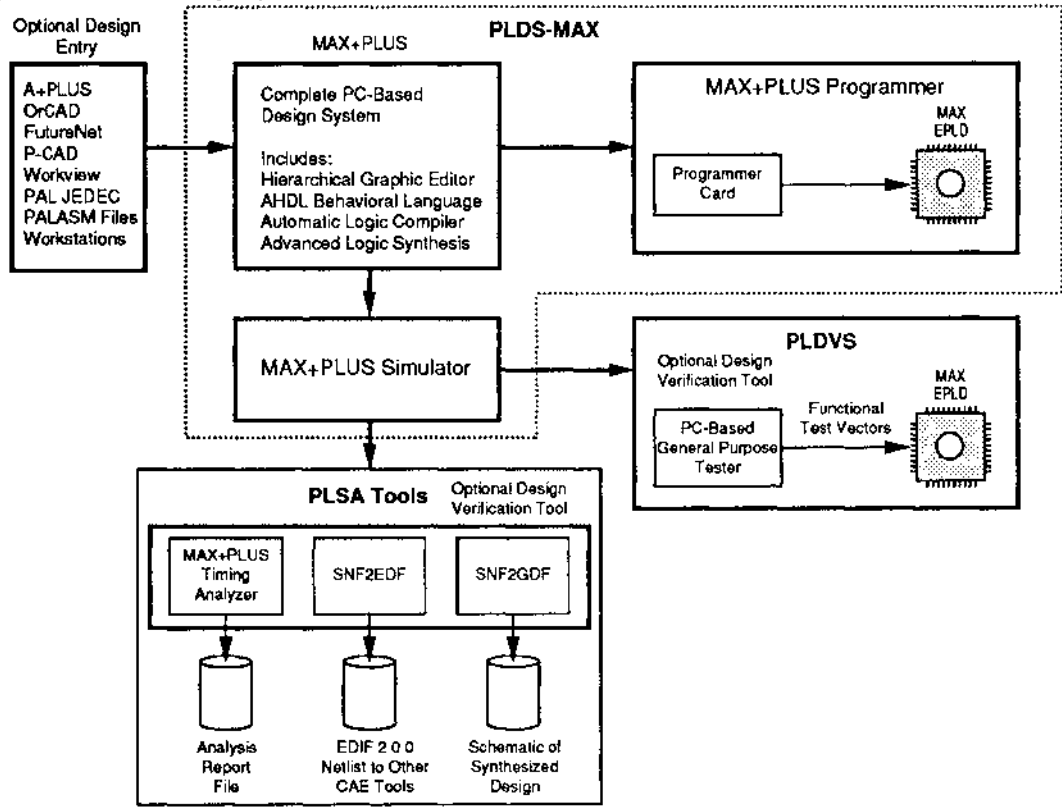
A major problem that has limited PLD density and speed has been signal routing, i.e., getting signals from one macrocell to another. A single array is used for small devices, and all signals are available to all macrocells. As devices increase in density, however, the number of signals being routed becomes very large, increasing the amount of silicon used for interconnections. The added loading on the internal connection path reduces the overall speed performance of the device. Larger MAX devices solve this problem since they are based on small, flexible LABs, connected by a PIA.

The PIA solves interconnect limitations by routing only the signals needed by each LAB. Every signal on the chip is in the PIA, which is programmed to give each LAB access only to the signals that it requires. This process solves any routing problems that may arise in a design, without degrading the performance of the device. Unlike masked or programmable gate arrays, which induce variable delays dependent on routing, the PIA has a fixed delay between any two LABs. The fixed delay eliminates undesired skews among logic signals that may cause glitches in internal or external logic.

MAX+PLUS Development System

The MAX+PLUS Development System (Figure 6) is a complete, integrated CAE system optimized for MAX devices. MAX+PLUS is an open system interfacing a variety of tools, including many third-party tools. In addition, it supports all three phases of a design cycle: design entry, design compilation, and design verification.

Figure 6. PC-Based Design System

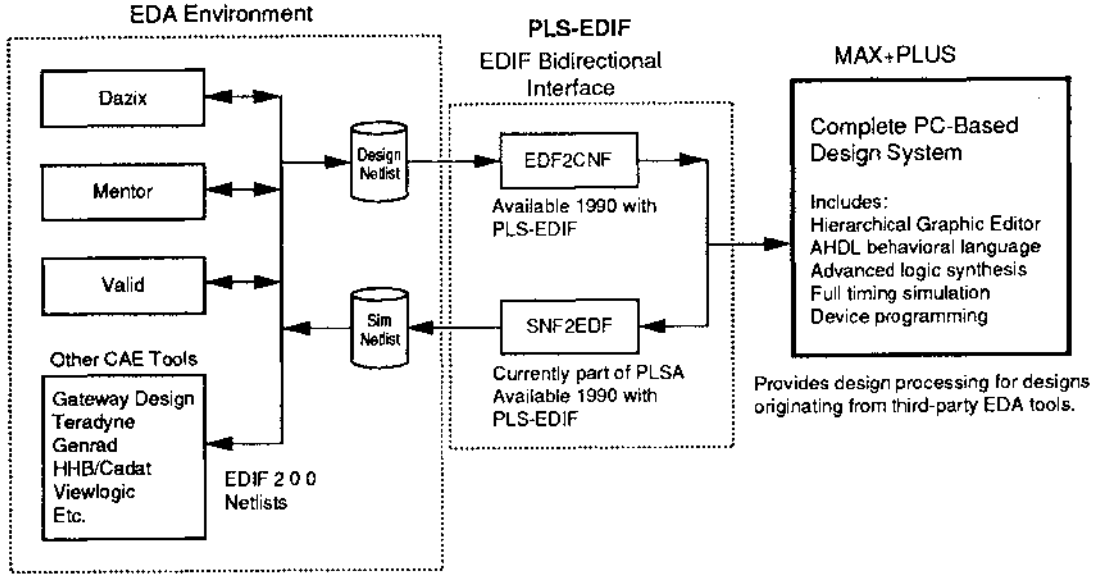


Design Entry

MAX+PLUS supports hierarchical graphic (schematic) and text design entry methods. In addition, designs created with other CAE tools may be placed in MAX EPLDs. MAX+PLUS accepts Altera Design File (ADF) netlists developed for EP-series devices, converts them into a format that can be read by the MAX+PLUS Compiler, and creates a symbol representing the design. Multiple ADFs may be connected in MAX+PLUS. The following companies and software packages produce ADFs: Altera (A+PLUS), Texas Instruments (A+PLUS), Intel (iPLDSI and iPLDSII), ViewLogic (Workview), and OrCAD (SDT III). Altera provides free utilities to enter EPLD designs with FutureNet and P-CAD, and convert their proprietary netlists into Altera ADF files. Free utilities to convert most PAL JEDEC files are also available from Altera Applications.

Altera offers a bidirectional EDIF interface (called PLS-EDIF) for companies that use workstations for design entry and board-level simulation. Users may transfer design information to the MAX+PLUS system with industry-standard EDIF 2 0 0 netlists (see Figure 7). Dazix, Mentor Graphics, and Valid Logic Systems provide EDIF netlists of schematic designs. These EDIF netlist descriptions may be translated into a format that can be

Figure 7. EDA Support



2

synthesized by the MAX+PLUS Compiler. PLS-EDIF is user-extendable and supports many libraries within the CAE vendors' schematic capture packages.

Design Compilation

Once the design is entered, the design is compiled. The MAX+PLUS Compiler provides automatic processing of the design. The design is reduced to primitive, gate-level functions and any unconnected logic is automatically eliminated. Next, the design is minimized to reduce the logic equations to their most compact form. Logic synthesis then analyzes the minimized equations and applies sophisticated heuristic algorithms to the design to ensure maximum device utilization. Once the design has been synthesized, the Fitter does the actual logic placement within the chosen device. This fitting is the equivalent of gate array place and route, except it is performed automatically in MAX+PLUS. All design processing in MAX+PLUS is automatic, allowing the designer to concentrate on the logic design, not the mechanics of mapping it into the logic device.

Design Verification

MAX+PLUS provides a full timing simulator for MAX EPLD designs. For users requiring board-level simulation capabilities, Altera offers interfaces to today's most popular simulators. With the PLSA's SNF2EDF tool or the bidirectional PLS-EDIF product, the Altera representation of a completed design (contained in the MAX+PLUS Simulator Netlist File) may be translated to an EDIF 200 netlist. This EDIF netlist contains all logical and timing information for the completed design and may be read by many third-party board-level simulators, such as Dazix DLS, Valid VALIDsim, Teradyne LASAR, HHB CADAT, and Gateway Verilog-XL.

Altera offers tools to analyze and test MAX EPLD designs. The PLSA Tools' MAX+PLUS Timing Analyzer (MTA) provides user-configurable reports to help analyze critical delay paths, setup and hold timing, and overall system performance of any synthesized MAX EPLD design. The SNF2GDF tool creates a schematic representation of the synthesized design, including delay information. The representation shows exactly how the MAX+PLUS Compiler synthesized the design, whether the design was entered with a schematic, behavioral description, or third-party netlist.

The Personal Logic Design Verification System (PLDVS) product is a low-cost, general-purpose tester for functional verification of TTL or CMOS devices. It can be used for both production and engineering testing of gate arrays, and standard cell and programmable logic devices. Because PLDVS has a seamless interface with the MAX+PLUS Simulator, it is ideally suited for MAX EPLDs. The MAX+PLUS Waveform Editor may be used to define test vectors for Device Under Test (DUT) pins in PLDVS. Additionally, any simulation performed by the MAX+PLUS Simulator can be compared to actual device functionality by submitting the simulation to PLDVS. The Waveform Editor shows both the software simulation and the hardware functional test simultaneously, highlighting any discrepancy between them.

For more software information, refer to *PLDS-MAX / PLS-MAX: MAX+PLUS Programmable Logic Development System, PLS-EDIF,* and *MAX+PLUS PLSA Tools.* For more device information, refer to *EPM5016 to EPM5032: MAX EPLDs with a Single LAB, EPM5064 to EPM5192: MAX EPLDs with Multiple LABs;* and *PLDVS: General Purpose PC-Based Tester.*

EPM5016 to EPM5032

MAX EPLDs with a Single LAB

January 1990 Data Sheet

Product Summary

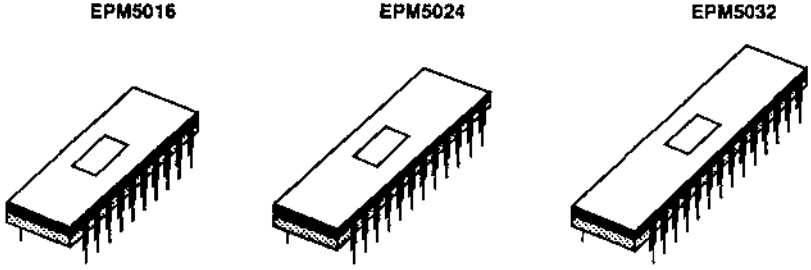
- Single-LAB CMOS EPLDs offering a consistent design solution across a broad range of speed and density requirements
- 15-ns combinatorial delays (t_{PD}), counters up to 100 MHz, and pipelined data rates of 100 MHz
- Fast, low pin-count devices for integration of up to 64 flip-flops and latches
- Package options from 20-pin dual-in-line packages (DIPs) to 28-pin J-lead packages, in windowed, erasable, or one-time-programmable plastic
- Dual feedback on all I/O pins
- Synchronous or asynchronous operation
- Single or multiple clocks
- MAX+PLUS PC-based design tools for compiling large designs in minutes
- EDIF industry-standard workstation and third-party interfaces

All EPLDs discussed in this Data Sheet are shown in Figure 1.

2

Figure 1. MAX Family Single LAB EPLDs

Surface mount packages are also available.



	EPM5016	EPM5024	EPM5032
Macrocells	16	24	32
Maximum number of flip-flops and latches	32	48	64
User I/O Pins	16	20	24
t_{PD}	15	15	15
Package pins	20	24	28

Single LAB Device Highlights

- **Multiple Array MatriX (MAX) architecture solves speed, density, and design flexibility problems:**
  - Advanced macrocell array provides registered, combinatorial, or flow-through latch operation on a macrocell-by-macrocell basis.
  - Expander product-term array automatically provides additional combinatorial or registered logic.
  - Decoupled I/O block with dual feedback on all I/O pins allows flexible pin utilization.
  - Each macrocell supports synchronous or asynchronous operation using single or multiple clocks per device, allowing individual clocking of each macrocell.
- **MAX Device Performance:**
  - Pipelined data rates to 100 MHz
  - Counters as fast as 100 MHz
  - t_{PD} performance from 15 ns
  - Advanced 0.8-micron CMOS EPROM technology
- **MAX Single LAB Device Logic Density:**
  - 16 to 32 macrocell devices
  - 20- to 28-pin packages
  - 32 to 64 flip-flops and latches
  - Up to 66 product terms on a single macrocell
  - Product-term expansion on any data or control path
- **MAX+PLUS Design Tools:**
  - TTL MacroFunction Library for speedy design entry
  - Design entry via unified, hierarchical schematic capture and text design language
  - Fast, automatic design processing with logic synthesis
  - Automatic device fitting, no hand-editing needed
  - Hardware and software design verification tools
  - Compiles a 16-bit complex counter in less than one minute
- **EDIF reader and writer interfaces to MAX+PLUS provide convenient paths to Dazix, Mentor, and Valid workstations.**
- **PAL conversion applications utilities allow rapid conversion to MAX devices.**

General Description

MAX Erasable Programmable Logic Devices (EPLDs) represent a revolutionary step in programmable logic: they combine innovative architecture and state-of-the-art process to offer optimum logic density, performance, and flexibility, and they provide the highest speeds and densities available in general-purpose reprogrammable logic. The single Logic Array Block (LAB) MAX devices described in this data sheet are high-speed, high-density replacements for SSI and MSI TTL and CMOS packages and conventional PLDs. For example, the EPM5032 easily replaces multiple PLA and PAL devices plus several 7400-series packages.

Single-LAB MAX devices discussed in this data sheet are optimized for speed and can be used to build 15-ns decoders or counters running at an 100-MHz count frequency. When expanders are used to build complex

combinatorial logic, the EPM5016 accommodates high product-term state machines that operate at 66 MHz.

Logic Array Blocks. Higher-speed MAX EPLDs have a single LAB. Each LAB contains a macrocell array, an expander product term array, and a decoupled I/O Block. Expander product terms (expanders) are unallocated, inverting product terms that may be used and shared by all macrocells in the LAB to increase combinatorial or registered logic. Expander sharing allows complex state machines and other complex combinatorial logic to be implemented efficiently. For example, the 64 expanders in the EPM5032 may be used to extend the combinatorial logic capacity of a given macrocell from 3 to 66 product terms. Alternatively, some or all of the expander product terms may be used to increase the register capacity of the device. Two expanders may be cross-coupled to form a latch; a D-type flip-flop with asynchronous Preset and Clear consumes 6 expanders. When added to the 32 basic macrocells (which can be configured as a latch or flip-flop) of the EPM5032, up to 64 latches or 42 flip-flops may be created.

Modular Architecture. The modular architecture of MAX EPLDs provides integration solutions over a wide range of densities. Movement within the family is very easy. Single-LAB MAX family devices range from 20 to 28 pins in DIP and J-lead packages. Higher-density MAX EPLDs with up to 192 macrocells and 100 pins are described in the companion MAX EPLD data sheet for multiple-LAB devices. Over the entire family, a wide range of packaging options for both through-hole and surface-mount applications are available. Plastic one-time-programmable (OTP) packages are available for economic volume production.

Logic Design Entry. Logic designs are created and programmed into MAX EPLDs with the MAX+PLUS Development System. MAX+PLUS is a complete CAE system offering hierarchical design entry tools, automatic design compilation and fitting, timing simulation, and device programming. The MAX+PLUS Compiler features advanced logic synthesis algorithms, allowing designs to be entered in a variety of high-level formats while ensuring the most efficient use of EPLD resources. The combination of a flexible architecture and advanced CAE tools ensures rapid design cycles so that a design may go from conception to completion in a single day. Interfaces to third-party tools are available to allow design entry and logic simulation on a variety of workstation platforms.

Functional Description

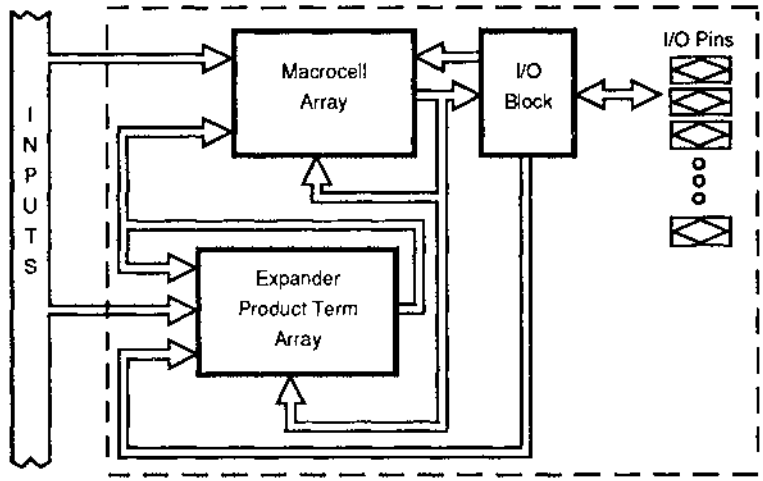
MAX EPLDs use CMOS EPROM cells to configure logic functions within the devices. MAX architecture is user-configurable to accommodate a variety of independent logic functions, and the devices may be erased for quick and efficient iterations during design development and debug cycles.

Logic Array Block

Higher-speed MAX EPLDs have a single Logic Array Block (LAB). The LAB, shown in Figure 2, consists of a macrocell array (the number of macrocells in the macrocell array varies with each device), an expander product term array, and an I/O control block. The macrocells are the primary resource for logic implementation, but if needed, expander product terms (expanders) can be used to supplement the the capabilities of any macrocell. The expander product term array consists of a group of unallocated, inverted product terms. Macrocells and allocatable expanders together facilitate variable product-term designs without the waste associated with fixed product-term architectures. Thus, PAL or PLA devices are easily integrated into MAX EPLDs. The outputs of the macrocells feed the decoupled I/O block that consists of a group of programmable tri-state buffers and I/O pins with optional feedbacks.

Figure 2. Logic Array Block

The LAB consists of a macrocell array, an expander product term array, and a decoupled I/O block. The flexibility of the LAB ensures high speeds and efficient device utilization.

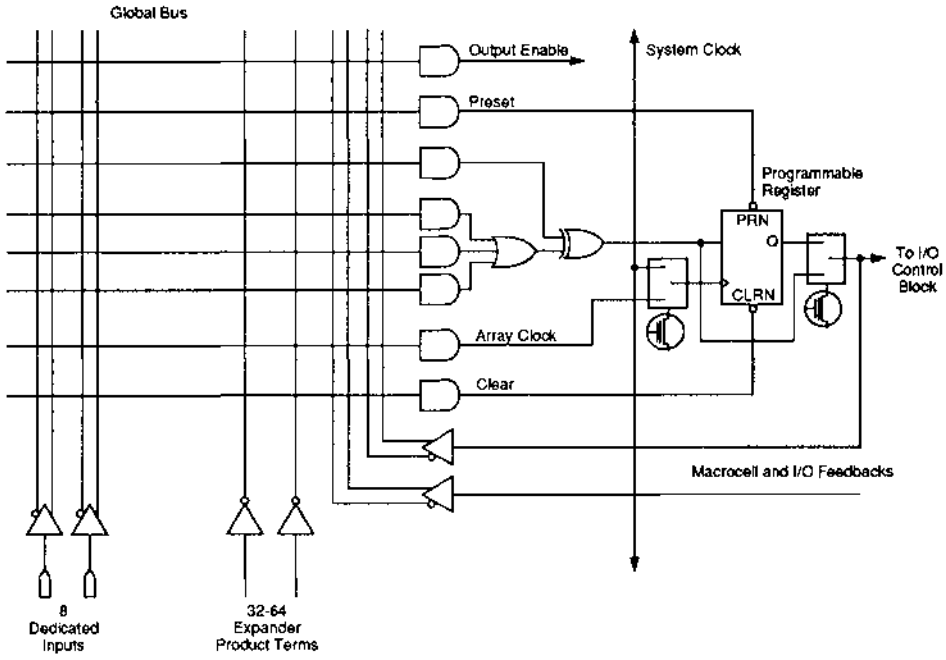


Macrocell

The MAX macrocell, shown in Figure 3, consists of a logic array and a configurable register that may be programmed for D, T, JK, SR operation, or a flow-through latch that may be bypassed for combinatorial operation. Combinatorial logic is implemented in the logic array, which consists of three product terms ORed together feeding one input of an XOR gate. The second input to the XOR gate is also controlled by a product term, providing the ability to implement active-high or active-low logic. The XOR gate is also used for complex, XOR arithmetic logic functions and for DeMorgan's Inversion to reduce the product terms required to implement a function. The output of the XOR gate feeds the programmable register, or bypasses it for purely combinatorial operation.

All macrocells are fully interconnected to a global interconnection network consisting of all inputs, I/O feedbacks, macrocell feedbacks, and expander feedbacks.

Figure 3. MAX Macrocell *The macrocell integrates 7400-series functions as well as low-density PLDs.*



2

Additional product terms, called secondary product terms, are used for Output Enable, Preset, Clear, and Clock logic. Preset and Clear product terms drive the active-low asynchronous Preset and asynchronous Clear input of the configurable flip-flop. The programmable array clock product term clocks the register on a low-to-high transition. Macrocells that drive an output pin may use the I/O Output Enable product term to control the active-high tri-state buffer, shown in Figure 5. Secondary product terms allow for exact emulation of 7400-series TTL functions.

Clock Options

Each LAB has two clocking modes: asynchronous and synchronous. During asynchronous clocking, each macrocell can be clocked independently, and any input or internal logic may be used as a clock. Asynchronous clocking also allows the flip-flops to be configured for positive or negative edge-triggered operation, giving the macrocell a high degree of flexibility.

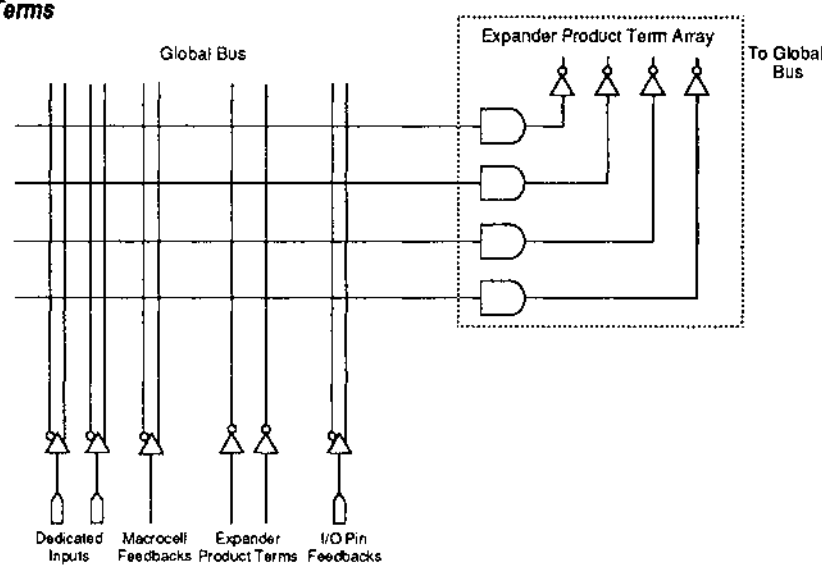
Synchronous clocking is provided by a dedicated system clock (CLK). With this direct connection, the system clock gives enhanced clock-to-output delay times. Since a LAB has one synchronous clock, all flip-flop clocks within it are positive edge-triggered off the CLK pin. If the CLK pin is not used as a system clock, it may be used as a dedicated input.

Expander Product Terms

The expander product term array, shown in Figure 4, contains unallocated, inverted product terms. Expanders may be used and shared by all product terms within the LAB. Wherever extra logic is needed (including the register control functions), expanders may be used to implement the logic. Thus, expanders provide the flexibility to implement register intensive and product-term-intensive designs for MAX EPLDs.

Figure 4. Expander Product Terms

Expander product terms are unallocated logic that can be used and shared by all macrocells in a LAB. Sharing allows efficient integration of complex control functions.



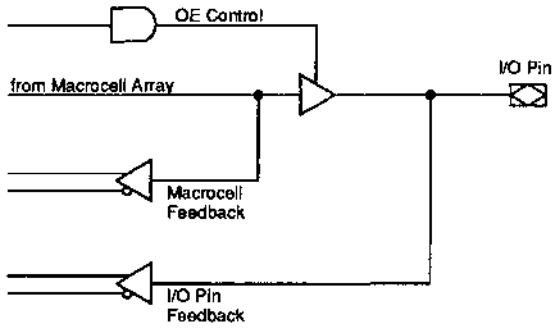
Expanders are fed by all signals in the LAB, including all expanders. Expander outputs are then inverted and fed to the global bus. One expander may feed all macrocells in the LAB, or multiple product terms in the same macrocell. Since expanders feed the secondary product terms (Preset, Clear, Clock, and Output Enable) of each macrocell, complex logic functions may be implemented without using another macrocell. Expanders may be cross-coupled to build additional flip-flops or latches. A MAX EPLD contains twice as many expanders as macrocells.

Each LAB has an I/O control block (Figure 5) consisting of a user-configurable I/O control function for each I/O pin. The I/O control block is fed only by the macrocell array. The tri-state buffer is controlled by a dedicated macrocell product term, and drives the I/O pad.

A MAX EPLD has dual feedback for every I/O pin; a feedback path is located before and after the tri-state buffer. The tri-state buffer is used to decouple the I/O pins from the macrocells so that all registers within the LAB may be "buried." Thus, I/O pins can be configured as dedicated outputs, bi-directional outputs, or dedicated inputs. Configuring an I/O pin as an input reduces the number of available expanders by two.

Figure 5. I/O Control Block

The decoupled I/O control block features dual feedback to maximize use of device pins.

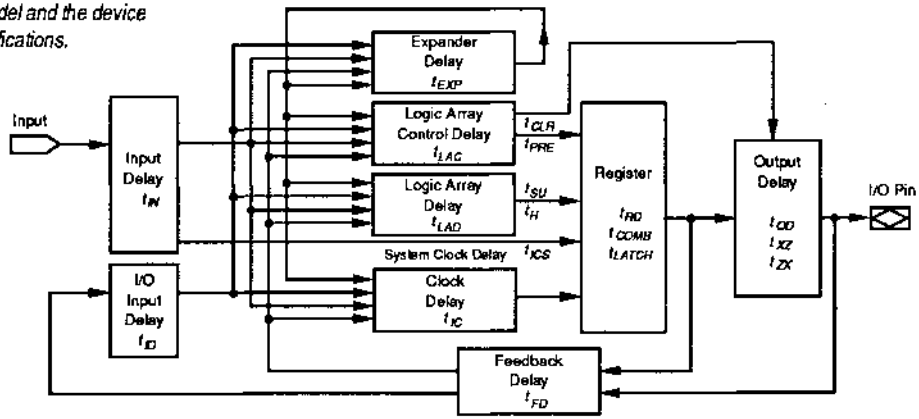


Timing Model

Timing within MAX EPLDs is easily determined with MAX+PLUS software or with the model shown in Figure 6. MAX devices have fixed internal delays, allowing the user to determine the worst-case timing delays for any design. For complete timing information, the MAX+PLUS software provides a timing simulator and a delay predictor.

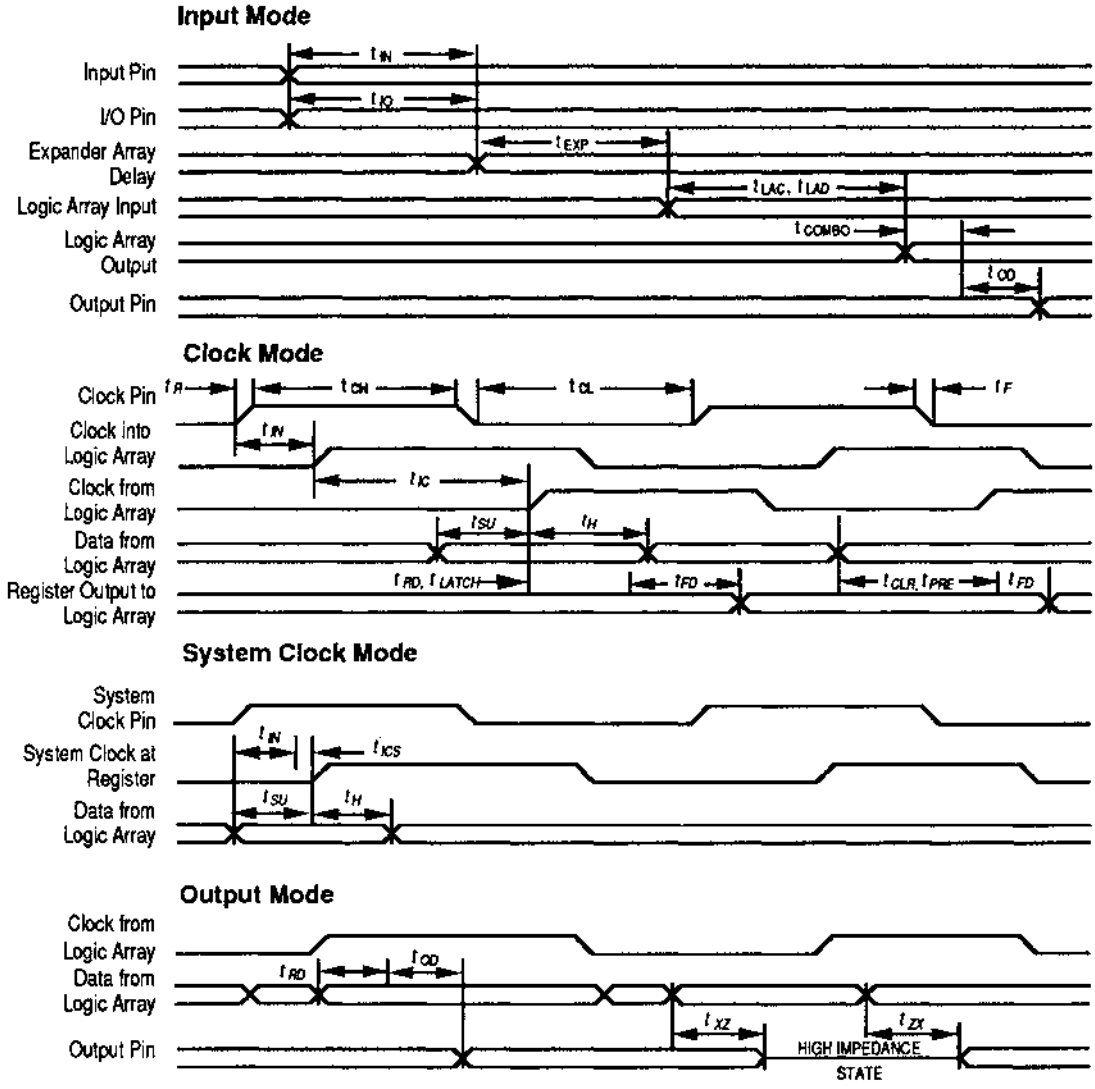
Figure 6. Timing Model

Device performance can be predicted with this timing model and the device performance specifications.



Timing information may be derived from Figure 6, when used together with the internal timing parameters for a particular device. External timing parameters for each device are derived from a sum of internal parameters and represent pin-to-pin timing delays. Figure 7 shows the internal timing waveforms for these devices. Refer to *Application Brief 75 (MAX EPLD Timing)* for further information.

Figure 7. Typical MAX EPLD Switching Waveforms



Design Guidelines

MAX EPLDs will be permanently damaged if they are operated under conditions that surpass those listed under "Absolute Maximum Ratings." 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 Ratings conditions for extended periods of time may affect device reliability. MAX EPLDs contain circuitry to protect device pins from high static voltages or electric fields; however, normal precautions should be taken to avoid application of any voltage higher than maximum rated voltages.

For proper operation, input and output pins must be constrained to the range GND < (V_{IN} or V_{OUT}) < V_{CC}. Unused dedicated inputs must always be tied to an appropriate logic level (either V_{CC} or GND). Unused I/O must be left unconnected. Each set of V_{CC} and GND pins must be connected together directly at the device. Power supply decoupling capacitors of at least 0.2 microfarads must be connected between V_{CC} and GND. For the most effective decoupling, each V_{CC} pin should be separately decoupled to GND, directly at the device. Decoupling capacitors should have good frequency response, such as monolithic ceramic types.

As with any CMOS device, power is a function of frequency and internal node switching. To obtain the most accurate power information, it is recommended that current consumption be measured after the design is completed and the device is placed in the system.

2

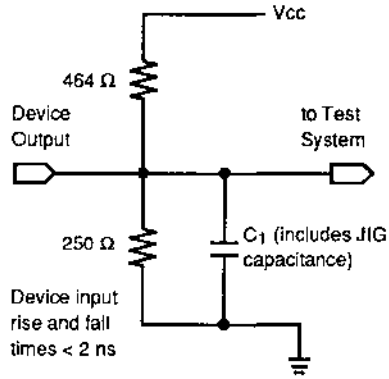
Design Security

MAX EPLDs contain a programmable design security feature that controls access to data programmed into the device. With this feature, a proprietary design implemented in the device cannot be copied or retrieved. This feature enables a high level of design security, since programmed data within EPROM cells is invisible. The Security bit that controls this function, along with all other program data, may be reset by erasing the device.

MAX EPLDs are fully functionally tested and guaranteed. Complete testing of each programmable EPROM bit and all internal logic elements thus ensures 100% programming yield. AC test measurements are performed under the conditions shown in Figure 8.

Figure 8. AC Test Conditions

Power supply transients can affect AC measurements. Simultaneous transitions of multiple outputs should be avoided for accurate measurement. Do not attempt to perform threshold tests under AC conditions. Large-amplitude fast-ground current transients normally occur as the device outputs discharge the load capacitances. These transients flowing through the parasitic inductance between the device ground pin and the test system ground can create significant reductions in observable input noise immunity.



Test programs can be used and then erased during early stages of the production flow. This facility to use application-independent, general-purpose tests is called generic testing and is unique to eraseable user-configurable logic devices. The devices also contain on-board logic test circuitry to allow verification of function and AC specification once encapsulated in non-windowed packages.

MAX+PLUS Development System

The Altera MAX+PLUS Development System is a unified CAE system for integrating designs into MAX EPLDs. Designs can be entered with logic schematics via the Graphic Editor or with state machines, truth tables, and Boolean equations via the Altera Hardware Description Language (AHDL). Logic synthesis and minimization optimize the logic of a submitted design. Design verification and timing analysis is performed with the Simulator or the Delay Predictor. Errors in a design are automatically located and highlighted in the schematic or text design file. Hosted on an IBM PS/2, PC-AT or compatible machine, MAX+PLUS gives the designer tools to quickly and efficiently create complex logic designs. Further details are available in the *PLDS-MAX / PLS-MAX Data Sheet*.

Device Programming

MAX EPLDs may be programmed on an IBM PS/2, PC-AT, or compatible computer using standard Altera hardware; the LP4, LP5, or LP6 programming card, the PLE3-12 or PLE3-12A Master Programming unit, and the appropriate device adapter. These items are included in a complete PLDS-MAX Development System, or may be purchased separately. MAX+PLUS software is available as part of the PLDS-MAX system, or as PLS-MAX stand-alone software package. MAX EPLDs may also be programmed using third-party platforms. Contact Altera's Application Department for the current status of third-party programming support.

Device Erasure

MAX EPLDs begin to erase when exposed to lights with wavelengths shorter than 4000 Å. Since fluorescent lighting and sunlight fall into this range, opaque labels should be placed over the EPLD window to ensure long-term reliability. The recommended erasure procedure for EPLDs is exposure to UV light with a wavelength of 2537 Å. Erasure time depends on the power of the UV lamp. Typically, 60 minutes is adequate to erase a MAX EPLD using a lamp with 12000 w/cm2 power rating. (Some low-power erasers may take longer.) Exposure to high-intensity UV light for extended periods may damage a MAX EPLD. MAX EPLDs may be erased and reprogrammed as often as necessary when the recommended erasure exposure levels are used.

Reference

Altera's Applications Department offers both written and telephone technical support. Supplemental material for MAX EPLDs is available in the form of Application Briefs and Notes. Consult Altera's Marketing Department (408-984-2800) for a current list of available material and the most recent specifications, and the Applications Department (408-984-2805 x102) for any technical questions concerning Altera products.

Features

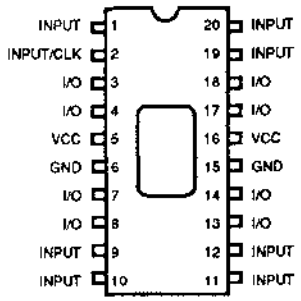
- Fast, 20-pin MAX single-LAB EPLD
  - Combinatorial speeds with t_{PD} equal to 15 ns
  - Counter frequencies of 100 MHz
  - Pipelined data rates of 100 MHz
- 16 individually configurable macrocells.
- 32 expander product terms (expanders) allowing 34 product terms on a single macrocell
- Up to 21 flip-flops or 32 latches
- Up to 10 input latches using cross-coupled expanders
- 24 mA output drivers to allow direct interfacing to system buses
- Programmable I/O architecture allowing for up to 16 inputs and 8 outputs
- Available in a ceramic-windowed or plastic 20-pin DIP package

General Description

The Altera EPM5016 (Figure 9) is a Multiple Array MatriX (MAX) CMOS EPLD optimized for speed. It can integrate multiple SSI and MSI TTL and 74HC devices. In addition, it can replace any 20-pin PAL or PLA device with logic left over for further integration.

Figure 9. Package Pin Out Diagram

The EPM5016 is available in a 20-pin DIP package.



EPM5016

The EPM5016, shown in Figure 10, contains 16 MAX macrocells. The expander product term array for the EPM5016 contains 32 expanders. The I/O control block contains 8 bidirectional I/O pins that can be configured for dedicated input, dedicated output, or bidirectional operation. All I/O pins feature dual-feedback for maximum pin flexibility. Figure 11 shows the output drive characteristics for EPM5016 I/O pins and the typical power consumption versus frequency for the EPM5016.

Figure 10. EPM5016 Block Diagram

The EPM5016 has 16 macrocells and 32 expanders.

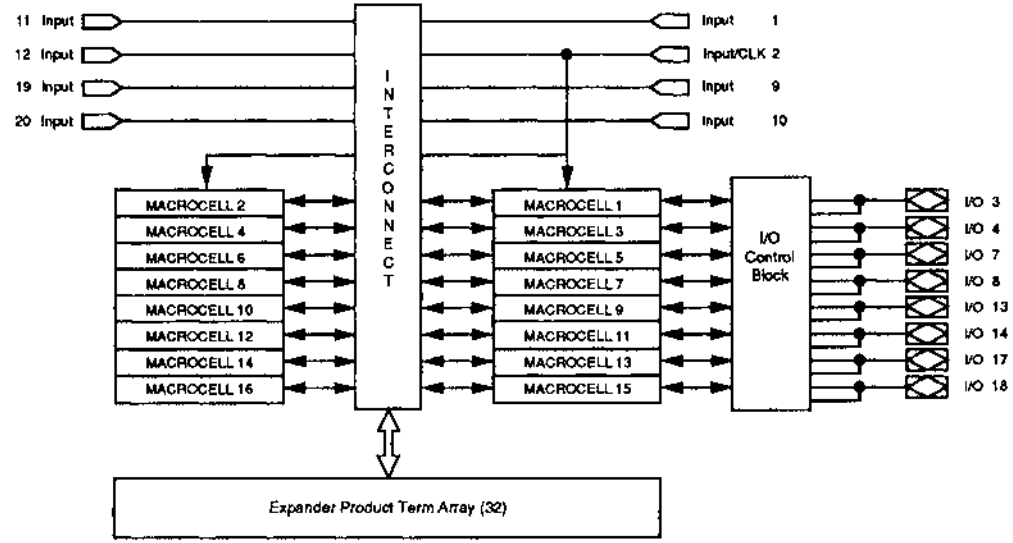
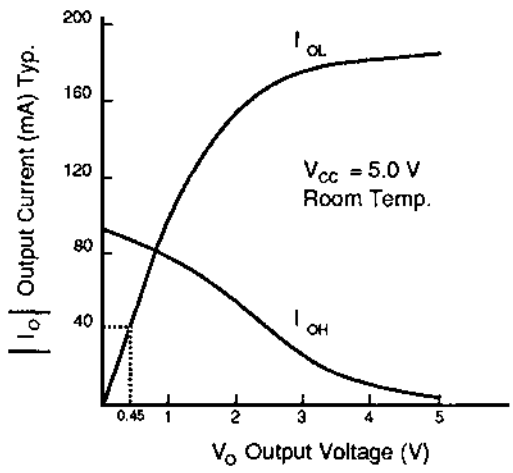
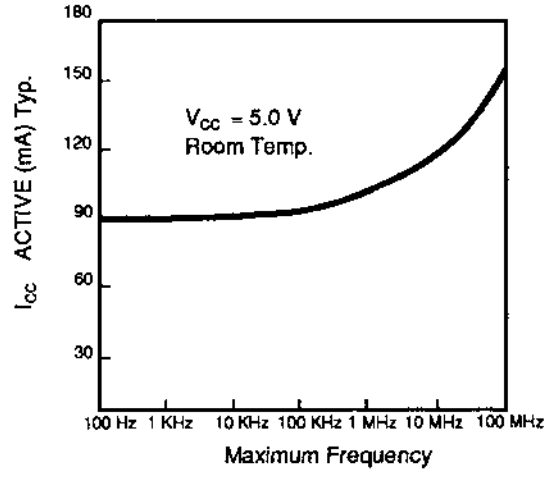


Figure 11. EPM5016 Output Drive Characteristics and I_{CC} vs. Frequency





Absolute Maximum Ratings

Note: See the *Design Recommendations* Data Sheet.

Symbol	Parameter	Conditions	Min	Max	Unit
V_{CC}	Supply voltage	With respect to GND	-2.0	7.0	V
V_{PP}	Programming supply voltage	*See Note (1)*	-2.0	13.5	V
V_I	DC input voltage		-2.0	7.0	V
I_{MAX}	DC V_{CC} or GND current			200	mA
I_{OUT}	DC output current, per pin		-50	50	mA
P_D	Power dissipation			1000	mW
T_{STG}	Storage temperature	No bias	-65	+150	° C
T_{AMB}	Ambient temperature	Under bias	-65	+135	° C
T_J	Junction temperature	Under bias		+150	° C

Recommended Operating Conditions

Symbol	Parameter	Conditions	Min	Max	Unit
V_{CC}	Supply voltage		4.75	5.25	V
V_I	Input voltage		0	V_{CC}	V
V_O	Output voltage		0	V_{CC}	V
T_A	Operating temperature	For Commercial	0	+70	° C
T_A	Operating temperature	For Industrial	-40	+85	° C
T_C	Case temperature	For Military	-55	+125	° C
t_R	Input rise time			100	ns
t_F	Input fall time			100	ns

2

DC Operating Conditions

V_{CC} = 5 V ± 5%, T_A = 0° C to 70° C for commercial, *See Note (2)*

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
V_{IH}	High level input voltage		2.0		V_{CC} + 0.3	V
V_{IL}	Low level input voltage		-0.3		0.8	V
V_{OH}	High level TTL output voltage	I_{OH} = -12 mA DC	2.4			V
V_{OL}	Low level output voltage	I_{OL} = 24 mA DC			0.5	V
I_I	Input leakage current	$V_I = V_{CC}$ or GND	-10		+10	µA
I_{OZ}	3-state output off-state current	$V_O = V_{CC}$ or GND	-40		+40	µA
I_{CC1}	V_{CC} supply current (standby)	$V_I = V_{CC}$ or GND		80	110	mA
I_{CC3}	V_{CC} supply current	$V_I = V_{CC}$ or GND No load, f = 1.0 MHz *See Note (3)*		85	115	mA

Capacitance

Symbol	Parameter	Conditions	Min	Max	Unit
C_{IN}	Input capacitance	V_{IN} = 0 V, f = 1.0 MHz		10	pF
C_{OUT}	Output capacitance	V_{OUT} = 0 V, f = 1.0 MHz		12	pF

AC Operating Conditions V_{CC} = 5 V ± 5%, T_A = 0° C to 70° C for commercial

External Timing Parameters			EPM5016-1		EPM5016-2		EPM5016		
Symbol	Parameter	Conditions	Min	Max	Min	Max	Min	Max	Unit
t_{PD1}	Input to non-registered output	C1 = 35 pF		15		17		20	ns
t_{PD2}	I/O input to non-reg. output	C1 = 35 pF		15		17		20	ns
t_{SU}	Setup time		6		8		11		ns
t_{H}	Hold time		0		0		0		ns
t_{CO1}	Clock to output delay	C1 = 35 pF		9		11		13	ns
t_{ASU}	Asynchronous setup time		5		7		9		ns
t_{AH}	Asynchronous hold time		5		7		8		ns
t_{CH}	Clock high time	*See Note (4)*	5		6		8		ns
t_{CL}	Clock low time	*See Note (4)*	5		6		8		ns
t_{ACO1}	Asynch clock to output delay	C1 = 35 pF		15		17		20	ns
t_{CNT}	Minimum clock period			10		12		16	ns
f_{CNT}	Internal maximum frequency		100		83.3		62.5		MHz
f_{MAX}	Max frequency; pipelined data		100		83.3		62.5		MHz

For complete information on internal timing parameters, refer to *Application Brief 75 (MAX EPLD Timing)*.

Internal Timing Parameters			EPM5016-1		EPM5016-2		EPM5016		
Symbol	Parameter	Conditions	Min	Max	Min	Max	Min	Max	Unit
t_{IN}	Input pad and buffer delay			4		5		5	ns
t_{IO}	I/O input pad and buffer delay			4		5		5	ns
t_{EXP}	Expander array delay			5		8		10	ns
t_{LAD}	Logic array delay			6		7		9	ns
t_{LAC}	Logic control array delay			4		5		7	ns
t_{OD}	Output buffer and pad delay	C1 = 35 pF		4		4		5	ns
t_{ZX}	Output buffer enable delay			7		7		8	ns
t_{XZ}	Output buffer disable delay	C1 = 5 pF		7		7		8	ns
t_{SU}	Register setup time		2		5		8		ns
t_{LATCH}	Flow-through latch delay			1		1		1	ns
t_{RD}	Register delay			1		1		1	ns
t_{COMB}	Combinatorial delay			1		1		1	ns
t_{H}	Register hold time		6		8		9		ns
t_{IC}	Clock delay			6		6		8	ns
t_{ICS}	System clock delay			0		1		2	ns
t_{FD}	Feedback delay			1		1		1	ns
t_{PRE}	Register preset time			3		6		6	ns
t_{CLR}	Register clear time			3		6		6	ns

Notes:
(1) Minimum DC input is -0.3 V. During transitions, inputs may undershoot to -2.0 V for periods shorter than 20 ns.
(2) Typical values are for T_A = 25° C and V_{CC} = 5 V.
(3) Measured with device programmed as a 16-bit counter.
(4) If the Clock is asynchronous (is from a product term), the sum (t_{CH} + t_{CL}) must be greater than or equal to t_{CNT}.

Features

Advanced Information

- Fast, 24-pin DIP or 28-pin J-lead MAX single-LAB EPLD
  - Combinatorial speeds with t_{PD} equal to 15 ns
  - Counter frequencies of over 76 MHz
  - Pipelined data rates of 83 MHz
- 24 individually configurable macrocells
- 48 expander product terms (expanders) allowing 50 product terms on a single macrocell
- Up to 32 flip-flops or 48 latches
- Up to 16 input latches using cross-coupled expanders
- Programmable I/O architecture providing up to 20 inputs and 12 outputs
- Available in a ceramic-windowed or plastic one-time-programmable package
- 24-pin DIP package

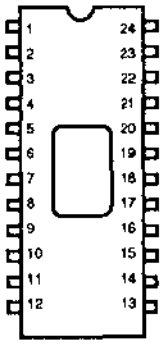
General Description

The Altera EPM5024 (Figure 12) is a Multiple Array MatriX (MAX) CMOS EPLD optimized for speed. It can integrate multiple SSI and MSI TTL and 74HC devices. In addition, it can replace any 20-pin PAL or PLA device with logic left over for further integration.

2

Figure 12. Package Pin Out Diagram

The EPM5024 is available in a 24-pin DIP package.



EPM5024

The EPM5024, shown in Figure 13, contains 24 MAX macrocells. The expander product term array for the EPM5024 contains 48 expanders. The I/O control block contains 12 bidirectional I/O pins that can be configured for dedicated input, dedicated output, or bidirectional operation. All I/O pins feature dual-feedback for maximum pin flexibility. Figure 14 shows output drive characteristics for the EPM5024 I/O pins and the typical power consumption versus frequency for the EPM5024.

Figure 13. EPM5024 Block Diagram

The EPM5024 has 24 macrocells and 48 expanders.

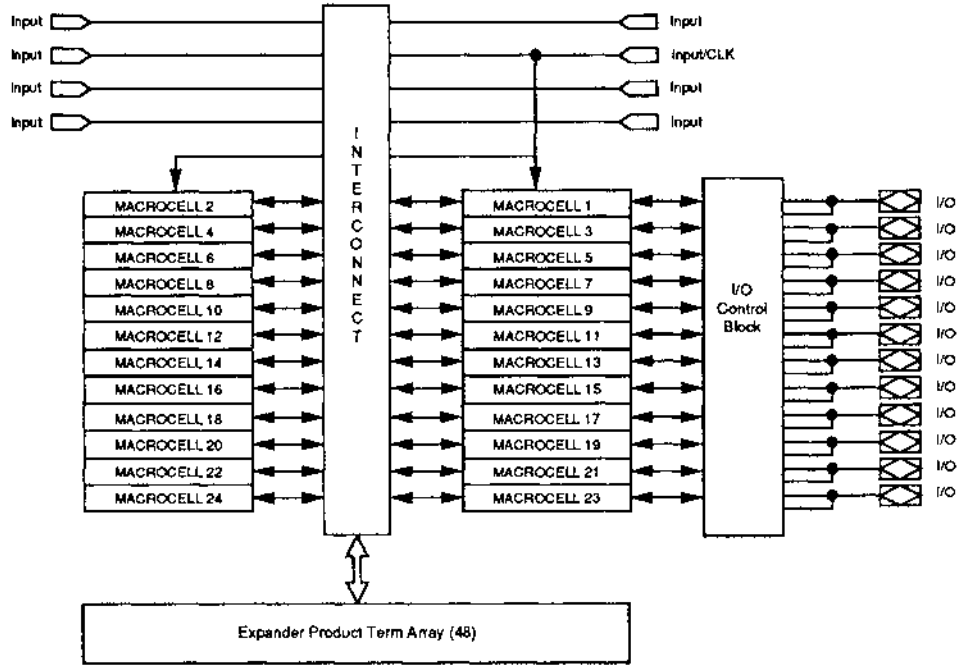
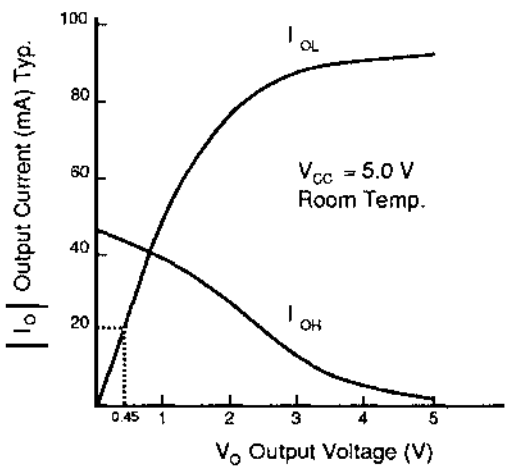
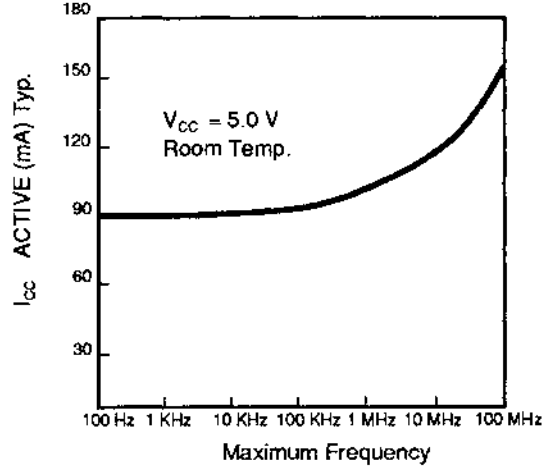


Figure 14. EPM5024 Output Drive Characteristics and I_{CC} vs. Frequency





Features

- Fast, 28-pin DIP or J-lead MAX single-LAB EPLD
  - Combinatorial speeds with t_{PD} equal to 15 ns
  - Counter frequencies of 76 MHz
  - Pipelined data rates of 83 MHz
- 32 individually configurable macrocells
- 64 expander product terms (expanders) allow 66 product terms on a single macrocell
- Up to 42 flip-flops or 64 latches
- Up to 21 input latches using cross-coupled expanders
- Programmable I/O architecture allowing for up to 24 inputs and 16 outputs
- Available in a ceramic-windowed or plastic one-time-programmable package
- 28-pin DIP or J-lead package

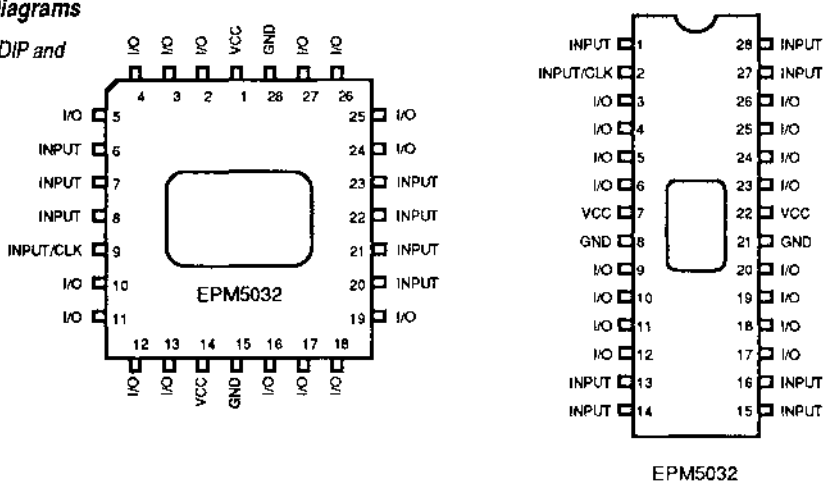
General Description

The Altera EPM5032 (Figure 15) is a Multiple Array MatriX (MAX) CMOS EPLD optimized for speed. It can integrate multiple SSI and MSI TTL and 74HC devices. In addition, it can replace multiple 20-pin PAL or PLA device with logic left over for further integration.

2

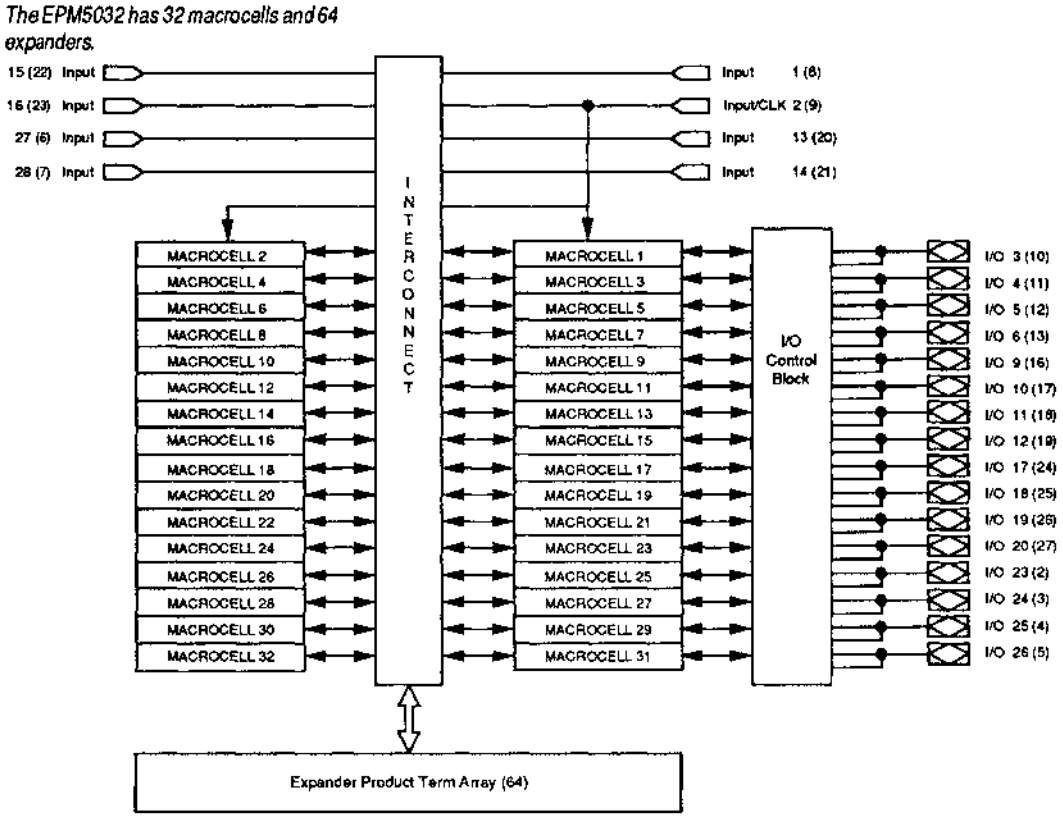
Figure 15. Package Pin Out Diagrams

The EPM5032 is available in DIP and J-lead packages.



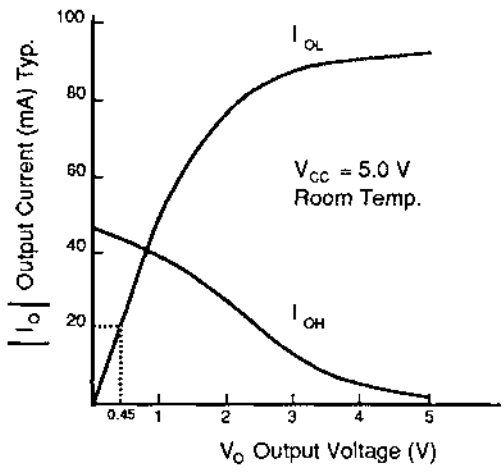
The EPM5032, shown in Figure 16, contains 32 MAX macrocells. The EPM5032 expander product term array contains 64 expanders. The I/O control block contains 16 bidirectional I/O pins that can be configured for dedicated input, dedicated output, or bidirectional operation. All I/O pins feature dual-feedback for maximum pin flexibility. Figure 17 shows output drive characteristics for EPM5032 I/O pins and typical power consumption versus frequency for the EPM5032.

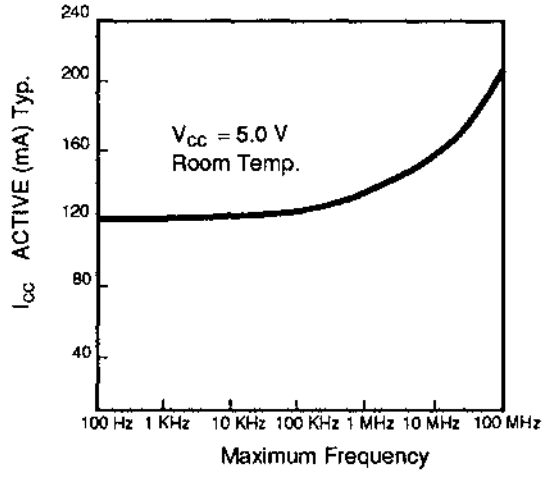
Figure 16. EPM5032 Block Diagram



Note: Pin numbers in () pertain to J-leaded packages.

Figure 17. EPM5032 Output Drive Characteristics and I_{CC} vs. Frequency





Absolute Maximum Ratings

Note: See the *Design Recommendations* Data Sheet.

Symbol	Parameter	Conditions	Min	Max	Unit
V_{CC}	Supply voltage	With respect to GND	-2.0	7.0	V
V_{PP}	Programming supply voltage	*See Note (1)*	-2.0	13.5	V
V_I	DC input voltage		-2.0	7.0	V
I_{MAX}	DC V_{CC} or GND current			300	mA
I_{OUT}	DC output current, per pin		-25	25	mA
P_D	Power dissipation			1500	mW
T_{STG}	Storage temperature	No bias	-65	+150	° C
T_{AMB}	Ambient temperature	Under bias	-65	+135	° C
T_J	Junction temperature	Under bias		+150	° C

Recommended Operating Conditions

Symbol	Parameter	Conditions	Min	Max	Unit
V_{CC}	Supply voltage		4.75	5.25	V
V_I	Input voltage		0	V_{CC}	V
V_O	Output voltage		0	V_{CC}	V
T_A	Operating temperature	For Commercial	0	+70	° C
T_A	Operating temperature	For Industrial	-40	+85	° C
T_C	Case temperature	For Military	-55	+125	° C
t_R	Input rise time			100	ns
t_F	Input fall time			100	ns

DC Operating Conditions

V_{CC} = 5 V ± 5%, T_A = 0° C to 70° C for commercial, *See Note (2)*

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
V_{IH}	High level input voltage		2.0		V_{CC} + 0.3	V
V_{IL}	Low level input voltage		-0.3		0.8	V
V_{OH}	High level TTL output voltage	I_{OH} = -4 mA DC	2.4			V
V_{OL}	Low level output voltage	I_{OL} = 8 mA DC			0.45	V
I_I	Input leakage current	$V_I = V_{CC}$ or GND	-10		+10	µA
I_{OZ}	3-state output off-state current	$V_O = V_{CC}$ or GND	-40		+40	µA
I_{CC1}	V_{CC} supply current (standby)	$V_I = V_{CC}$ or GND		120	150	mA
I_{CC3}	V_{CC} supply current	$V_I = V_{CC}$ or GND No load, f = 1.0 MHz *See Note (3)*		125	155	mA

Capacitance

Symbol	Parameter	Conditions	Min	Max	Unit
C_{IN}	Input capacitance	V_{IN} = 0 V, f = 1.0 MHz		10	pF
C_{OUT}	Output capacitance	V_{OUT} = 0 V, f = 1.0 MHz		12	pF

2

AC Operating Conditions V_{CC} = 5 V ± 5%, T_A = 0° C to 70° C for commercial

External Timing Parameters			EPM5032-1 (4)		EPM5032-2		EPM5032		
Symbol	Parameter	Conditions	Min	Max	Min	Max	Min	Max	Unit
t_{PD1}	Input to non-registered output	C1 = 35 pF		15		20		25	ns
t_{PD2}	I/O input to non-reg. output	C1 = 35 pF		15		20		25	ns
t_{SU}	Setup time		9		12		15		ns
t_{H}	Hold time		0		0		0		ns
t_{CO1}	Clock to output delay	C1 = 35 pF		10		12		15	ns
t_{ASU}	Asynchronous setup time		7		9		12		ns
t_{AH}	Asynchronous hold time		7		9		12		ns
t_{CH}	Clock high time	*See Note (5)*	6		7		8		ns
t_{CL}	Clock low time	*See Note (5)*	6		7		8		ns
t_{ACO1}	Asynch clock to output delay	C1 = 35 pF		15		20		25	ns
t_{CNT}	Minimum clock period			13		16		20	ns
f_{CNT}	Internal maximum frequency		76.9		62.5		50		MHz
f_{MAX}	Max frequency; pipelined data		83.3		71.4		62.5		MHz

For complete information on internal timing parameters, refer to *Application Brief 75 (MAX EPLD Timing)*.

Internal Timing Parameters			EPM5032-1		EPM5032-2		EPM5032		
Symbol	Parameter	Conditions	Min	Max	Min	Max	Min	Max	Unit
t_{IN}	Input pad and buffer delay			4		5		7	ns
t_{IO}	I/O input pad and buffer delay			4		5		7	ns
t_{EXP}	Expander array delay			8		10		15	ns
t_{LAD}	Logic array delay			6		9		10	ns
t_{LAC}	Logic control array delay			4		7		7	ns
t_{OD}	Output buffer and pad delay	C1 = 35 pF		4		5		5	ns
t_{ZX}	Output buffer enable delay			7		8		11	ns
t_{XZ}	Output buffer disable delay	C1 = 5 pF		7		8		11	ns
t_{SU}	Register setup time		5		5		8		ns
t_{LATCH}	Flow-through latch delay			1		1		3	ns
t_{RD}	Register delay			1		1		1	ns
t_{COMB}	Combinatorial delay			1		1		3	ns
t_{H}	Register hold time		6		9		12		ns
t_{IC}	Clock delay			6		8		10	ns
t_{ICS}	System clock delay			1		2		3	ns
t_{FD}	Feedback delay			1		1		1	ns
t_{PRE}	Register preset time			5		6		9	ns
t_{CLR}	Register clear time			5		6		9	ns

Notes:
(1) Minimum DC input is -0.3 V. During transitions, inputs may undershoot to -2.0 V for periods shorter than 20 ns.
(2) Typical values are for T_A = 25° C and V_{CC} = 5 V.
(3) Measured with device programmed as a 32-bit counter.
(4) This version is in development. Consult factory.
(5) If the Clock is asynchronous (is from a product term), the sum (t_{CH} + t_{CL}) must be greater than or equal to t_{CNT}.

EPM5064 to EPM5192
MAX EPLDs with Multiple LABs

January 1990 **Data Sheet**

Features

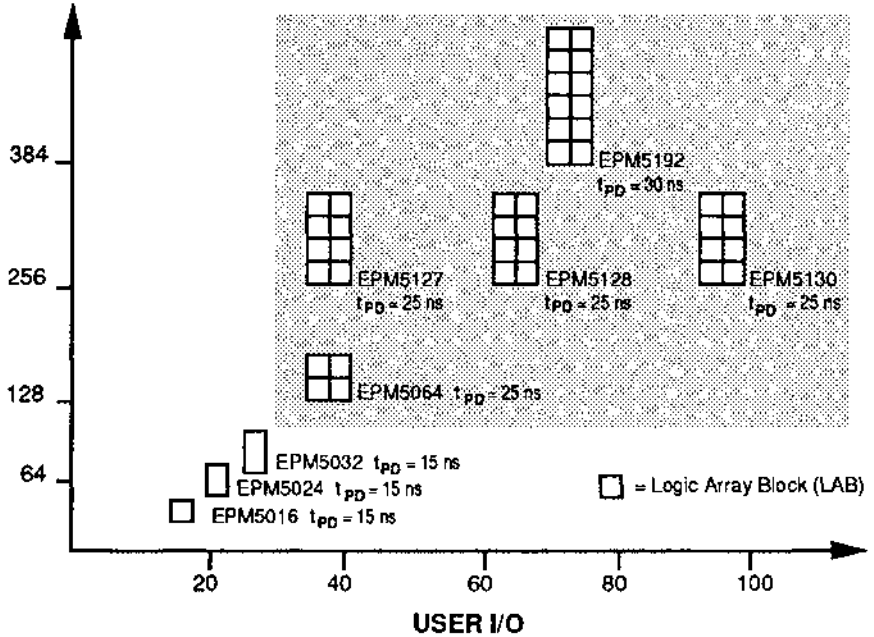
- Complete CMOS EPLD family offering a consistent design solution across a broad range of speed and density requirements
- 25-ns combinatorial delays (t_{PD}), counters up to 50 MHz, and pipelined data rates of 62 MHz
- High-density devices for integration of up to 384 flip-flops and latches
- Package options from 44-lead JLCC to 100-pin quad flat packs (QFPs), in windowed, erasable, or one-time-programmable plastic
- Easy conversion to custom-masked devices for very high volume production
- MAX+PLUS PC-based design tools for compiling large designs in minutes
- EDIF industry-standard workstation and third-party interfaces

Figure 1 shows the MAX family of EPLDs. Highlighted EPLDs are discussed in this Data Sheet.

2

Figure 1. MAX EPLDs

Devices with multiple LABs range in density from 64 to 192 macrocells.



Multiple-LAB Device Highlights

- **Multiple Array MatriX (MAX) architecture solves speed, density, and design flexibility problems:**
  - Advanced macrocell array provides registered, combinatorial, or flow-through latch operation on a macrocell-by-macrocell basis
  - Expander product term array automatically provides additional combinatorial or registered logic
  - Decoupled I/O block with dual feedback on all I/O pins allows flexible pin utilization
  - Programmable Interconnect Array (PIA) provides automatic 100% routing in devices with multiple logic arrays
  - Each macrocell supports synchronous or asynchronous operation using single or multiple clocks per device, allowing individual clocking of each macrocell
- **MAX Device Performance:**
  - Pipelined data rates to 62 MHz
  - Counters as fast as 50 MHz
  - t_{PD} performance from 25 ns
  - Advanced 0.8-micron CMOS EPROM technology
- **MAX Multiple LAB Device Logic Density:**
  - 64 to 192 macrocell devices
  - 44- to 100-pin packages
  - 128 to 384 flip-flops and latches
  - More than 32 product terms on a single macrocell
  - Product term expansion on any data or control path
- **MAX+PLUS Design Tools:**
  - TTL MacroFunction Library for simplified design entry
  - Design entry via unified hierarchical schematic capture and text design language
  - Fast, automatic design processing with logic synthesis
  - Automatic device fitting, no hand editing needed
  - Hardware and software design verification tools
  - Compiles fully-utilized EPM5128 design in 5 to 10 minutes on a 20-MHz 80386-based PC
- **EDIF reader and writer interfaces to MAX+PLUS provide convenient paths to Dazix, Mentor, Valid, and other workstations.**
- **PAL conversion applications utilities allow rapid conversion to high-density MAX devices.**

General Description

MAX Erasable Programmable Logic Devices (EPLDs) represent a revolutionary step in programmable logic: they combine innovative architecture and state-of-the-art process to offer VLSI density without sacrificing speed; they provide the highest speeds and densities available in general-purpose reprogrammable logic; and they are high-speed, high-density replacements for TTL SSI and MSI logic and conventional PLDs. For example, an EPM5192 replaces over 100 7400-series SSI and MSI TTL and CMOS packages, integrating complete subsystems into a single package, saving board area, and reducing power consumption.

MAX EPLDs discussed in this data sheet range in density from 64 to 192 macrocells. They easily achieve system clock frequencies of 31 MHz, and are

capable of counter frequencies of 50 MHz. (Higher speed MAX EPLDs, which are ideal for complex state machines and address decoding, are described in the companion MAX EPLD data sheet for single-LAB devices.) Figure 1 shows the entire MAX EPLD family.

Logic Array Blocks. High-density MAX EPLDs consist of groups of flexible logic arrays called Logic Array Blocks (LABs). Each LAB contains a macrocell array, an expander product term array, and a decoupled I/O block. Expander product terms (expanders) are unallocated, inverting product terms that may be used and shared by all macrocells in the LAB to create combinatorial and registered logic. Thus, expressions requiring up to 32 product terms can be implemented in a single macrocell. Signals are routed between LABs by a Programmable Interconnect Array (PIA) that ensures 100% routability. This multiple array architecture enables MAX EPLDs to offer the speed of smaller arrays with the integration density of larger arrays.

Modular Architecture. The modular architecture of MAX EPLDs provides integration solutions over a wide range of logic densities. Movement within the family is very easy. For example, the EPM5064 has the same pin-out as the EPM5127, so that twice the logic density is available in the same package. Also, the EPM5127, EPM5128, and EPM5130 all have the same logic capacity, but have packages optimized to handle different I/O requirements. Over the entire family, a wide range of packaging options for both through-hole and surface-mount applications are available. Plastic one-time-programmable (OTP) packages are available for economical volume production.

Logic Design Entry. Logic designs are created and programmed into MAX EPLDs with the MAX+PLUS Development System. MAX+PLUS is a complete CAE system offering hierarchical design entry tools, automatic design compilation and fitting, timing simulation, and device programming. The Compiler features advanced logic synthesis algorithms, allowing designs to be entered in a variety of high-level formats while ensuring the most efficient use of EPLD resources. The combination of a flexible architecture and advanced CAE tools ensures rapid design cycles so that a design may go from conception to completion in single day.

Functional Description

MAX EPLDs use CMOS EPROM cells to configure logic functions within the devices. MAX architecture is user-configurable to accommodate a variety of independent logic functions, and the devices may be erased for quick and efficient iterations during design development and debug cycles.

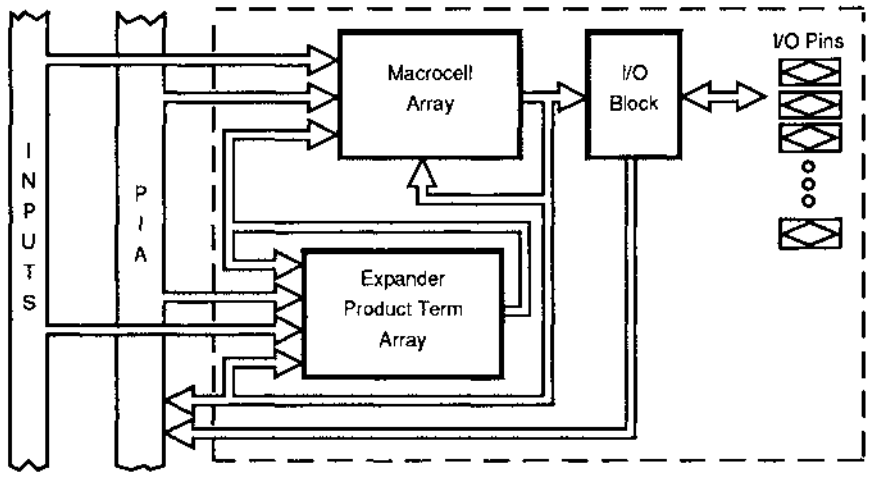
Logic Array Block

Higher-density MAX EPLDs consist of multiple Logic Array Blocks (LABs) connected by a Programmable Interconnect Array (PIA). The LAB, shown in Figure 2, consists of a macrocell array containing 16 macrocells, an expander product-term array containing 32 expanders, and the I/O control block. The macrocells are the primary resource for logic implementation, but if needed, expander product terms can be used to supplement the capabilities of any macrocell. The expander product-term array consists of a group of unallocated

product terms. Flexible macrocells and allocatable expander product terms together facilitate variable product term designs. Thus, multiple low-density PLDs, such as the 22V10, are easily integrated into MAX EPLDs. The outputs of the macrocells feed the decoupled I/O block, which consists of a group of programmable tri-state buffers and I/O pins. The compact LABs allow MAX EPLDs to combine high levels of integration with very high speed.

Figure 2. Logic Array Block

The LAB consists of a macrocell array, an expander product term array, and a decoupled I/O block. The flexibility of the LAB ensures high speeds and efficient device utilization.



Macrocell

The MAX macrocell, shown in Figure 3, has been designed to be as efficient and flexible as possible. Each macrocell consists of a logic array and an independently configurable register that may be programmed for D, T, JK, or SR operation, and may also be configured as a flow-through latch, or bypassed for purely combinatorial operation. Combinatorial logic implemented in the logic array allows use of very high fan-in logic expressions.

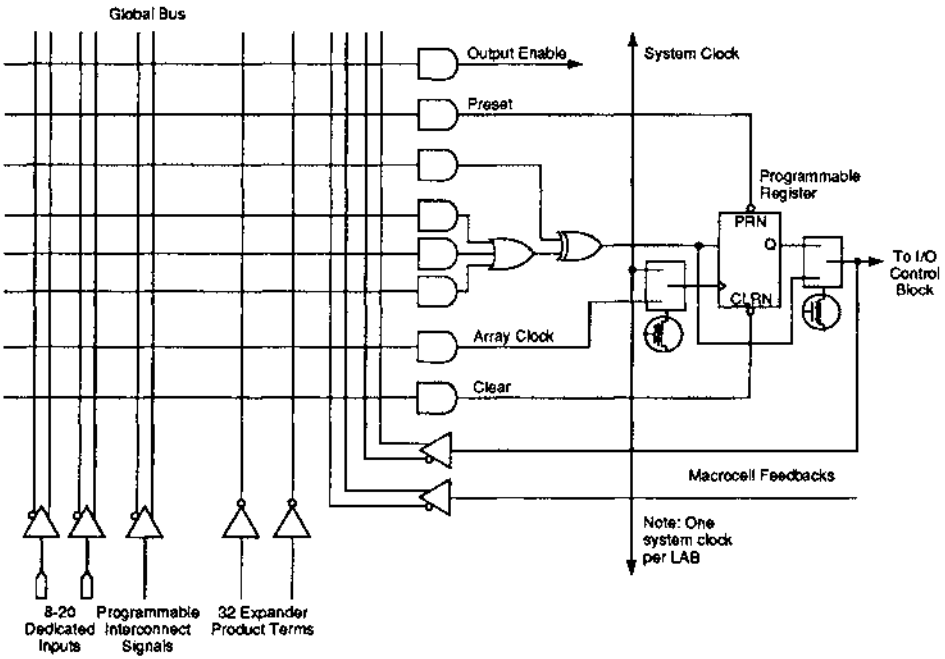
The logic array consists of three product terms ORed together feeding one input of an XOR gate. The second input to the XOR gate—also controlled by a product term—is used by MAX+PLUS software to implement complex XOR arithmetic logic functions such as adders and comparators. The XOR gate is also used to implement active-high or active-low logic and DeMorgan's Inversion to reduce the number of product terms required by a function. The output of the XOR gate feeds the programmable register. The compact logic array ensures high speed while eliminating inefficient, unused product terms. Also, expanders can be allocated to enhance the capability of the logic array.

Additional product terms, called secondary product terms, are used for Output Enable, Preset, Clear, and Clock logic. Preset and Clear product terms drive the active-low asynchronous Preset and asynchronous Clear input of the configurable flip-flop. The combination of asynchronous Clear

and Preset functions enables the MAX devices to easily handle active-low logic. The Clock product term from the programmable array is used to clock the register on a low-to-high transition. Macrocells that drive an output pin may use the I/O Output Enable product term to control the active-high tri-state buffer, shown in Figure 5. Secondary product terms allow for exact emulation of 7400-series TTL functions.

Such configurability allows the MAX macrocell to efficiently integrate complete subsystems into a single device. The macrocell outputs are globally routed within an LAB and also feed the PIA, so that routing of signal-intensive designs is extremely easy.

Figure 3. MAX Macrocell *The macrocell integrates 7400-series functions as well as low-density PLDs.*



Clock Options

Each LAB has two clocking modes: asynchronous and synchronous. During asynchronous clocking, each flip-flop can be independently clocked, and any input or internal logic may be used as a clock. As a result, systems requiring multiple clocks are easily integrated into MAX EPLDs. Asynchronous clocking also allows each flip-flop to be configured for positive or negative edge-triggered operation, giving the macrocell a high degree of flexibility.

Synchronous clocking is provided by a dedicated system clock (CLK). This direct connection provides enhanced clock-to-output delay times. Each LAB

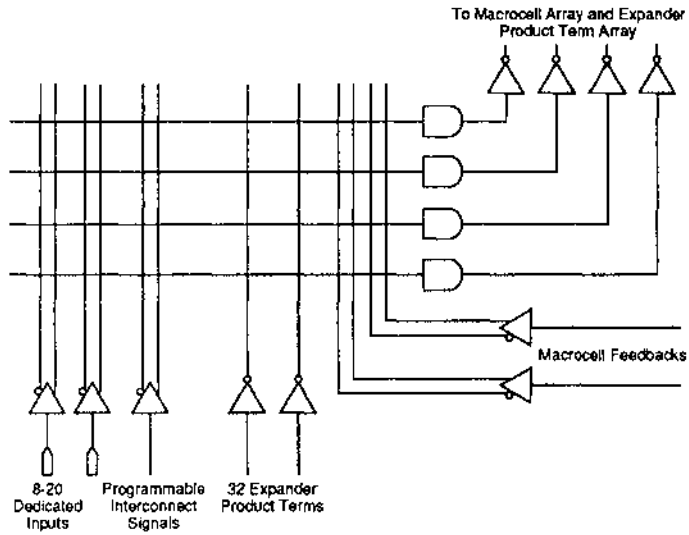
has one synchronous clock, so that all flip-flop clocks within it are positive edge-triggered from the CLK pin. Thus, synchronous and asynchronous clocking may be used in the same device. If the CLK pin is not used as a system clock, it may be used as a dedicated input.

Expander Product Term

The expander product-term array (Figure 4) contains unallocated product terms that enhance the macrocell array. Expanders may be used and shared by all product terms in the LAB. Wherever extra logic is needed (including register control functions), expanders can be used to implement logic so MAX EPLDs can efficiently integrate registered and combinatorial functions.

Figure 4. Expander Product Terms

Expander product terms are unallocated logic that can be used and shared by all macrocells in an LAB. Sharing allows efficient integration of complex control functions.



Expanders are fed by all signals in the LAB, including all expanders. One expander may feed all macrocells in the LAB or multiple product terms in the same macrocell. Since expanders feed the secondary product terms (Preset, Clear, Clock, and Output Enable) of each macrocell, complex logic functions may be implemented without using another macrocell. Expanders may be cross-coupled to build additional flip-flops or latches. A MAX EPLD contains twice as many expanders as macrocells.

I/O Control Block

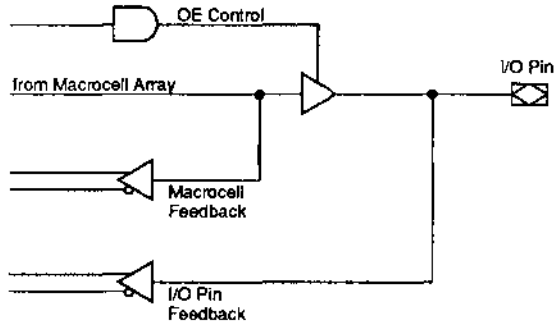
Each LAB has an I/O control block (Figure 5) consisting of a user configurable I/O control function for each I/O pin. The I/O control block is fed only by the macrocell array. The tri-state buffer is controlled by a dedicated macrocell product term, and drives the I/O pad.

A MAX EPLD has dual feedback for every I/O pin, which means that there is a feedback path before and after the tri-state buffer. The tri-state buffer is used to decouple the I/O pins from the macrocells so that all registers within the LAB are "buried." Thus, all I/O pins can be configured as dedicated

outputs, bi-directional outputs, or as dedicated inputs. I/O pins feed the PIA and are routable to all LABs.

Figure 5. I/O Control Block

The decoupled I/O control block features dual feedback to maximize use of device pins.



Programmable Interconnect Array

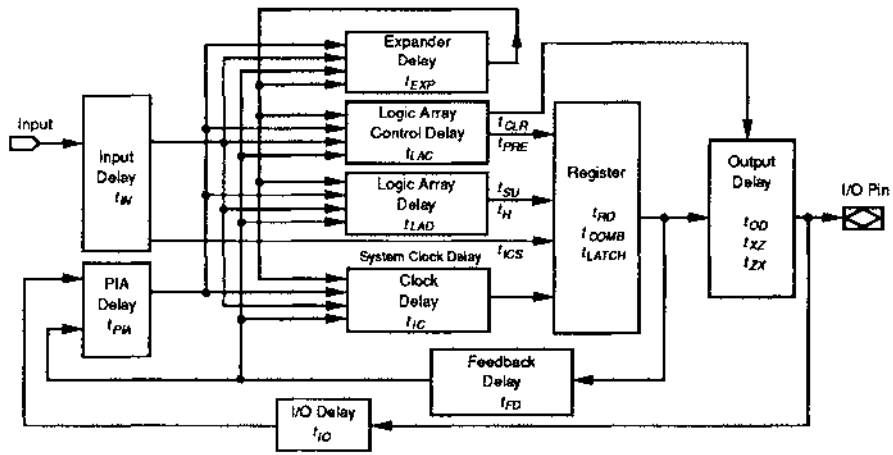
The Programmable Interconnect Array (PIA) routes signals between the various LABs. It only routes the signals required for implementing logic in an LAB, and is fed by all macrocell feedbacks and all I/O pin feedbacks. Unlike channel routing in masked or programmable gate arrays—where routing delays are variable and path-dependent—the PIA has a fixed delay. As a result, the PIA eliminates skew between signals, so that timing performance is easy to predict before a design is started.

Timing Model

Timing within MAX EPLDs is easily determined with MAX+PLUS software or with the model shown in Figure 6. MAX devices have fixed internal delays, allowing the user to determine the worst-case timing delays for any design. For complete timing information, MAX+PLUS software provides a timing simulator and a delay predictor.

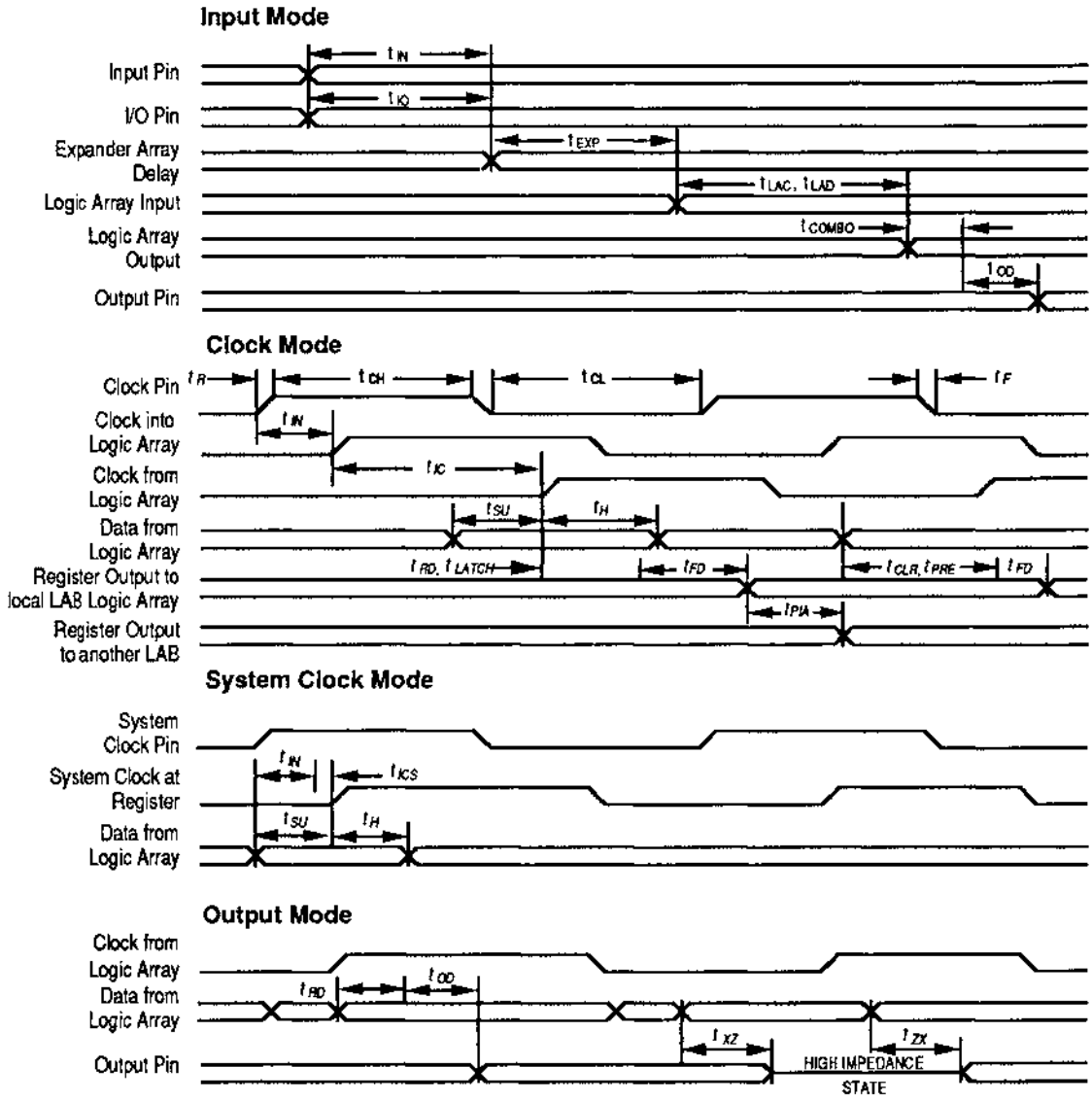
Figure 6. Timing Model

Device performance can be predicted with this timing model and the device performance specifications.



Timing information may be derived from Figure 6, when used together with the internal timing parameters for a particular device. External timing parameters for each device are derived from a sum of internal parameters and represent pin-to-pin timing delays. Figure 7 shows the internal timing waveforms for these devices. Refer to Altera's *Application Brief 75 (MAX EPLD Timing)* for further information.

Figure 7. Typical MAX EPLD Switching Waveforms



Design Guidelines

MAX EPLDs will be permanently damaged if they are operated under conditions that surpass those listed under "Absolute Maximum Ratings." This is a stress rating only, and functional operation of the devices at these or any other conditions above those indicated in the operational sections of this data sheet is not implied. Exposure to Absolute Maximum Ratings conditions for extended periods of time may affect device reliability. MAX EPLDs contain circuitry to protect device pins from 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.

For proper operation, input and output pins must be constrained to the range GND < V_{IN} or V_{OUT} < V_{CC}. Unused dedicated inputs must always be tied to an appropriate logic level (either V_{CC} or GND). Unused I/O pins must be left unconnected. Each set of V_{CC} and GND pins must be connected together directly at the device. Power supply decoupling capacitors of at least 0.2 microfarads must be connected between V_{CC} and GND. For the most effective decoupling, each V_{CC} pin should be separately decoupled to GND, directly at the device. Decoupling capacitors should have good frequency response, such as monolithic ceramic types.

As with any CMOS device, power is a function of frequency and internal node switching. To obtain the most accurate power information, it is recommended that current consumption be measured after the design is completed and the device is placed in the system.

2

Design Security

MAX EPLDs contain a programmable design security feature that controls access to the data programmed into the device. If this feature is used, a proprietary design implemented in the device cannot be copied or retrieved. This feature enables a high level of design security since programmed data within EPROM cells is invisible. The Security bit that controls this function, along with all other program data, may be reset simply by erasing the device.

MAX EPLDs are fully functionally tested and guaranteed. Complete testing of each programmable EPROM bit and all internal logic elements thus ensures 100% programming yield. AC test measurements are performed under the conditions shown in Figure 8.

Figure 8. AC Test Conditions

Power supply transients can affect AC measurements. Simultaneous transitions of multiple outputs should be avoided for accurate measurement. Do not attempt to perform threshold tests under AC conditions. Large-amplitude, fast-ground current transients normally occur as the device outputs discharge the load capacitances. These transients flowing through the parasitic inductance between the device ground pin and the test system ground can create significant reductions in observable input noise immunity.

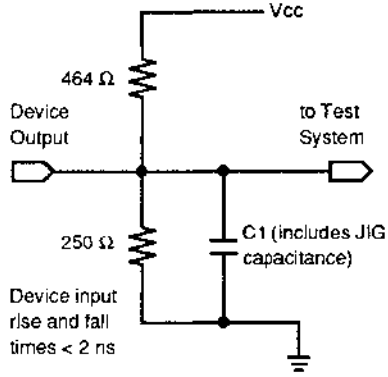
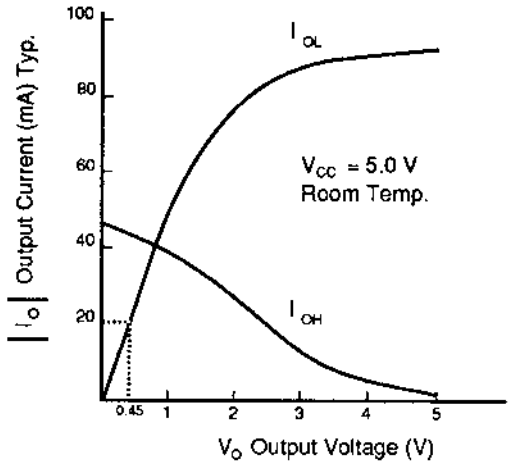


Figure 9. Output Drive Characteristics



Test programs may be used and then erased during early stages of the production flow. This facility to use application-independent, general purpose tests is called generic testing and is unique among user-configurable logic devices. The devices also contain on-board logic test circuitry to allow verification of function and AC specification once encapsulated in non-windowed packages.

Figure 9 shows output drive characteristics for MAX EPLDs with multiple LABs.

MAX+PLUS Development System

The MAX+PLUS Development System is a unified CAE system for integrating designs into MAX EPLDs. Designs can be entered with logic schematics via the Graphic Editor or with state machines, truth tables, and Boolean equations via the Altera Hardware Description Language (AHDL). Logic synthesis and minimization optimize the logic of a submitted design. Design verification and timing analysis is performed with the Simulator or the Delay Predictor. Errors in a design are automatically located and highlighted in the schematic or text design file. Hosted on an IBM PS/2, PC-AT or compatible machine, MAX+PLUS gives the designer the tools to quickly and efficiently create complex logic designs. Further details about the MAX+PLUS Development System are available in the *PLDS-MAX / PLS-MAX Data Sheet*.

Device Programming

MAX EPLDs may be programmed on an IBM PS/2, PC-AT or compatible computer using standard Altera hardware: the LP4, LP5, or LP6 programming card, the PLE3-12 or PLE3-12A Master Programming unit, and the appropriate device adapter. These items are included in a complete PLDS-MAX Development System, or may be purchased separately. MAX EPLDs may also be programmed using third-party platforms. Contact Altera's Application Department for the current status of third-party programming support.

Device Erasure

MAX EPLDs begin to erase when exposed to lights with wavelengths shorter than 4000 Å. Since fluorescent lighting and sunlight fall into this range, opaque labels should be placed over the EPLD window to ensure long-term reliability. The recommended erasure procedure for EPLDs is exposure to UV light with a wavelength of 2537 Å. Erasure time depends on the power of the UV lamp. Typically, 60 minutes is adequate to erase a MAX EPLD using a lamp with 12000 w/cm2 power rating. (Some low-power erasers may take longer.) Exposure to high-intensity UV light for extended periods may damage a MAX EPLD. MAX EPLDs may be erased and reprogrammed as often as necessary when the recommended erasure exposure levels are used.

Features

- High-density, 64-macrocell general purpose MAX EPLD
- 128 shareable expander product terms providing flexible logic expansion
  - over 32 product terms in a single macrocell
  - 64 additional latches provided by cross-coupling expanders
- Multiple-LAB MAX architecture with t_{PD} equal to 25 ns, counter frequencies of up to 50 MHz, and pipelined data rates of 62 MHz
- Programmable I/O architecture allowing up to 36 inputs and 28 outputs
- 44-pin J-lead package which easily integrates 10 standard PALS in 1/2 square inch of board space
  - ceramic windowed or plastic OTP packages for volume production

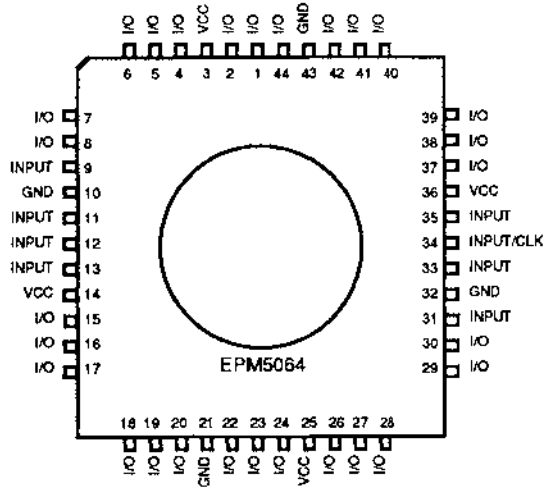
General Description

The Altera EPM5064 is a user-configurable, high-performance MAX EPLD. It is a high-density replacement for 7400-series SSI and MSI TTL and CMOS logic. In addition, it can integrate multiple 20- and 24-pin low density PLDs. For example, the EPM5064 can integrate the logic contained in over 10 standard 20-pin PALs.

Figure 10 shows the package pinout for the EPM5064 J-leaded packages. The J-leaded package occupies only 1/2 square inch of board space. The EPM5064 is pin-to-pin compatible with the EPM5127 for easy doubling of integration density without pinout changes.

2

Figure 10. J-Lead Package for the EPM5064



The EPM5064, shown in Figure 11, consists of 64 macrocells equally divided into 4 Logic Array Blocks, each containing 16 macrocells. Each LAB also contains 32 expander product terms. The flexibility of the LABs allows easy integration of any common PLD.

The EPM5064 has 8 dedicated input pins, one of which may be used as a synchronous system clock providing enhanced clock-to-output delays. The device has 28 I/O pins that can be configured for input, output, or bidirectional data flow. The I/O pins feature dual feedback so that any macrocell can be buried. Two of the LABs have 8 I/O pins, ensuring high speed for 8-bit bus functions, and 2 of the LABs have 6 I/O pins.

Figure 11. EPM5064 Block Diagram

The EPM5064 has 64 macrocells divided into 4 Logic Array Blocks. Note: Parentheses indicate J-leaded package pin numbers.

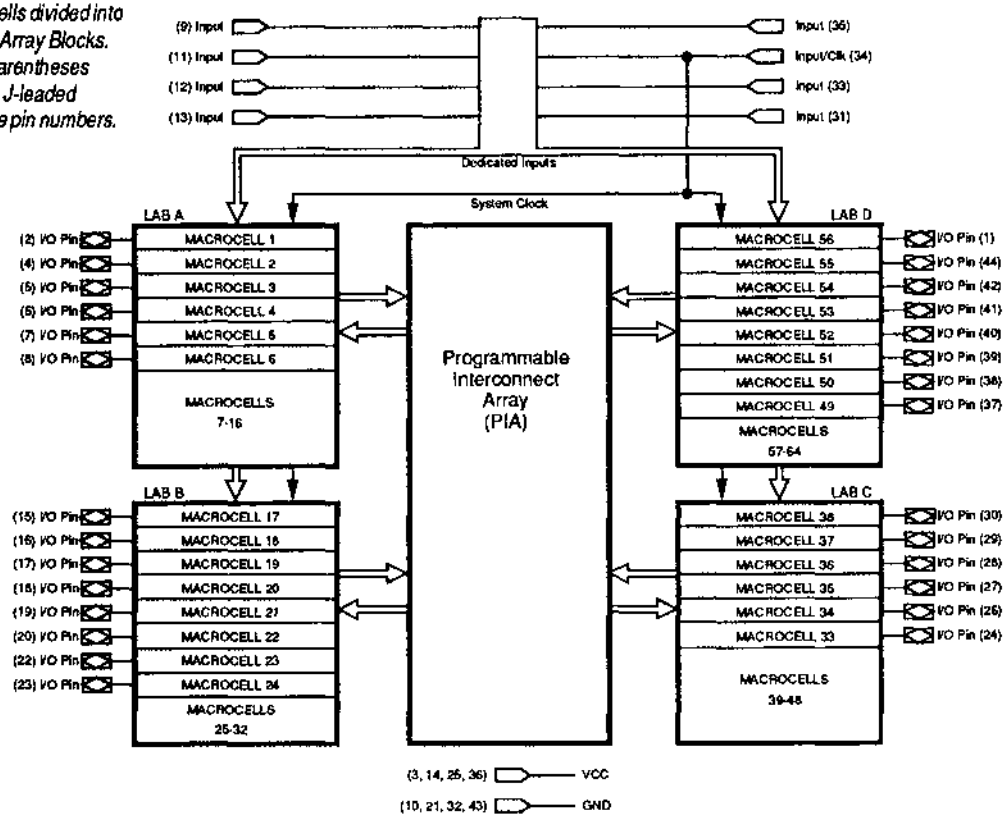
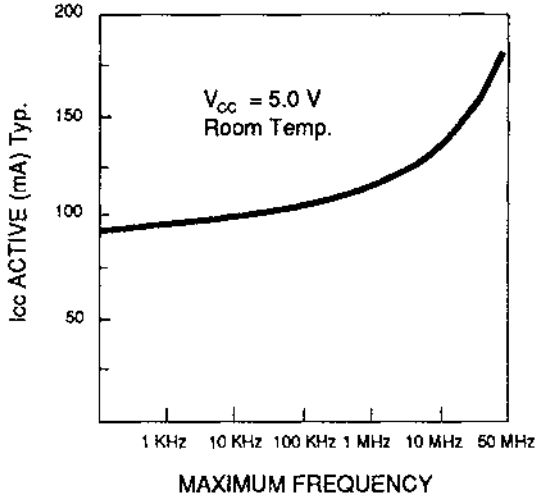


Figure 12 shows typical power consumption versus frequency for the EPM5064. The high integration density of the EPM5064 often results in greatly reduced system power requirements.

Figure 12. I_{CC} vs. Frequency



2

Absolute Maximum Ratings

Note: See the *Design Recommendations* Data Sheet.

Symbol	Parameter	Conditions	Min	Max	Unit
V_{CC}	Supply voltage	With respect to GND	-2.0	7.0	V
V_{PP}	Programming supply voltage	*See Note (1)*	-2.0	13.5	V
V_I	DC input voltage		-2.0	7.0	V
I_{MAX}	DC V_{CC} or GND current			400	mA
I_{OUT}	DC output current, per pin		-25	25	mA
P_D	Power dissipation			2000	mW
T_{STG}	Storage temperature	No bias	-65	+150	° C
T_{AMB}	Ambient temperature	Under bias	-65	+135	° C
T_J	Junction temperature	Under bias		+150	° C

Recommended Operating Conditions

Symbol	Parameter	Conditions	Min	Max	Unit
V_{CC}	Supply voltage		4.75	5.25	V
V_I	Input voltage		0	V_{CC}	V
V_O	Output voltage		0	V_{CC}	V
T_A	Operating temperature	For Commercial	0	+70	° C
T_A	Operating temperature	For Industrial	-40	+85	° C
T_C	Case temperature	For Military	-55	+125	° C
t_R	Input rise time			100	ns
t_F	Input fall time			100	ns

DC Operating Conditions

V_{CC} = 5 V ± 5%, T_A = 0° C to 70° C for commercial, *See Note (2)*

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
V_{IH}	High level input voltage		2.0		V_{CC} + 0.3	V
V_{IL}	Low level input voltage		-0.3		0.8	V
V_{OH}	High level TTL output voltage	I_{OH} = -4 mA DC	2.4			V
V_{OL}	Low level output voltage	I_{OL} = 8 mA DC			0.45	V
I_I	Input leakage current	$V_I = V_{CC}$ or GND	-10		+10	µA
I_{OZ}	3-state output off-state current	$V_I = V_{CC}$ or GND	-40		+40	µA
I_{CC1}	V_{CC} supply current (standby)	$V_I = V_{CC}$ or GND		90	125	mA
I_{CC2}	V_{CC} supply current	$V_I = V_{CC}$ or GND No load, f = 1.0 MHz *See Note (3)*		95	135	mA

Capacitance

Symbol	Parameter	Conditions	Min	Max	Unit
C_{IN}	Input capacitance	V_{IN} = 0 V, f = 1.0 MHz		10	pF
C_{OUT}	Output capacitance	V_{OUT} = 0 V, f = 1.0 MHz		20	pF

AC Characteristics

V_{CC} = 5 V ± 5%, T_A = 0° C to 70° C for commercial

External Timing Parameters			EPM5064-1 (4)		EPM5064-2		EPM5064		
Symbol	Parameter	Conditions	Min	Max	Min	Max	Min	Max	Unit
t_{PD1}	Input to non-registered output	C1 = 35 pF		25		30		35	ns
t_{PD2}	I/O Input to non-reg. output	C1 = 35 pF		40		45		55	ns
t_{SU}	Setup time		15		20		25		ns
t_{H}	Hold time		0		0		0		ns
t_{CO1}	Clock to output delay	C1 = 35 pF		14		16		20	ns
t_{ASU}	Asynchronous setup time		5		6		8		ns
t_{AH}	Asynchronous hold time		6		8		10		ns
t_{CH}	Clock high time		8		10		12.5		ns
t_{CL}	Clock low time		8		10		12.5		ns
t_{ACO1}	Asynch clock to output delay	C1 = 35 pF		25		30		35	ns
t_{CNT}	Minimum clock period			20		25		30	ns
f_{CNT}	Internal maximum frequency		50		40		33.3		MHz
f_{MAX}	Max frequency; pipelined data		62.5		50		40		MHz

For complete information on internal timing parameters, refer to *Application Brief 75 (MAX EPLD Timing)*.

Internal Timing Parameters			EPM5064-1		EPM5064-2		EPM5064		
Symbol	Parameter	Conditions	Min	Max	Min	Max	Min	Max	Unit
t_{IN}	Input pad and buffer delay			6		7		9	ns
t_{IO}	I/O Input pad and buffer delay			6		6		9	ns
t_{EXP}	Expander array delay			12		14		20	ns
t_{LAD}	Logic array delay			12		14		16	ns
t_{LAC}	Logic control array delay			10		12		13	ns
t_{OD}	Output buffer and pad delay	C1 = 35 pF		4		5		6	ns
t_{ZX}	Output buffer enable delay			10		11		13	ns
t_{XZ}	Output buffer disable delay	C1 = 5 pF		10		11		13	ns
t_{SU}	Register setup time		6		8		8		ns
t_{LATCH}	Flow-through latch delay			3		4		4	ns
t_{RD}	Register delay			1		2		2	ns
t_{COMB}	Combinatorial delay			3		4		4	ns
t_{H}	Register hold time		6		8		12		ns
t_{IC}	Clock delay			14		16		18	ns
t_{ICS}	System clock delay			2		2		3	ns
t_{FD}	Feedback delay			1		1		2	ns
t_{PRE}	Register preset time			5		6		7	ns
t_{CLR}	Register clear time			5		6		7	ns
t_{PIA}	Progr. Interconn. Array delay			13		16		20	ns

2

Notes to tables:

(1) Minimum DC input is -0.3 V. During transitions, the inputs may undershoot to -2.0 V for periods less than 20 ns.

(2) Typical values are for $T_A = 25°$ C and $V_{CC} = 5$ V.

(3) Measured with device programmed as a 16-bit counter in each LAB.

(4) This version is in development. Contact factory.

Features

Advanced Information

- 128-macrocell general purpose MAX EPLD optimized for designs requiring large amounts of buried logic
- 256 shareable expander product terms providing flexible logic expansion
  - over 32 product terms in a single macrocell
  - 128 additional latches provided by cross-coupling expanders
- Highest density 44 J-lead programmable logic device available; easily integrates over 60 TTL MSI and SSI components while consuming only 1/2 square inch of valuable board space
- Multiple-LAB MAX architecture with combinatorial decodes as fast as 25 ns, counter frequencies of 50 MHz, and pipelined data rates of 62 MHz
- Programmable I/O architecture allowing up to 36 inputs and 28 outputs
- 44-pin J-leaded, ceramic windowed or one-time-programmable plastic packages

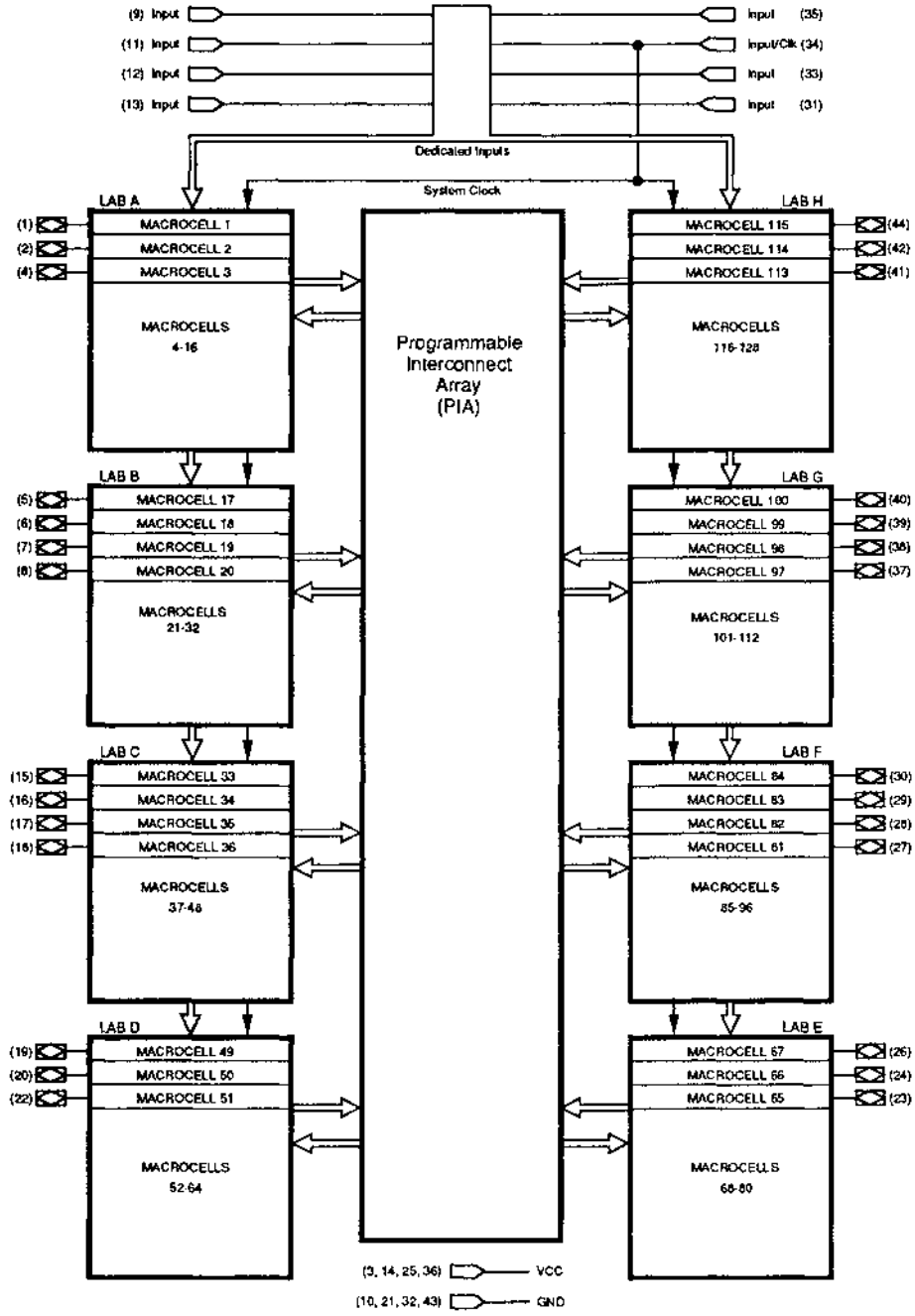
General Description

The Altera EPM5127 is a user-configurable, high-performance MAX EPLD optimized for designs with large amounts of buried logic. For example, high-density serial communication subsystems and 8-bit data path functions can be quickly integrated into an EPM5127. The EPM5127 is a high-density replacement for 7400-series SSI and MSI TTL and CMOS logic and 74HC logic. For example, a 74151 8-to-1 multiplexer consumes less than 1% of an EPM5127. In addition, it can integrate multiple 20- and 24-pin low-density PLDs.

The EPM5127, shown in Figure 13, consists of 128 macrocells equally divided into 8 Logic Array Blocks, each containing 16 macrocells. Each LAB also contains 32 expander product terms. The compact size of the LABs ensures high speeds, thus allowing better system performance.

The EPM5127 has 8 dedicated input pins, one of which may be used as a synchronous system clock. The device has 28 I/O pins that can be configured for input, output, or bidirectional data flow. These I/O pins feature dual feedback so that any macrocell can be buried while the I/O pin is being used as an input. Four of the LABs have 4 I/O pins, and 4 of the LABs have 3 I/O pins.

Figure 13. EPM5127 Block Diagram *The EPM5127 has 128 macrocells divided into 8 Logic Array Blocks. Note: Parentheses indicate J-leaded package pin numbers.*



Features

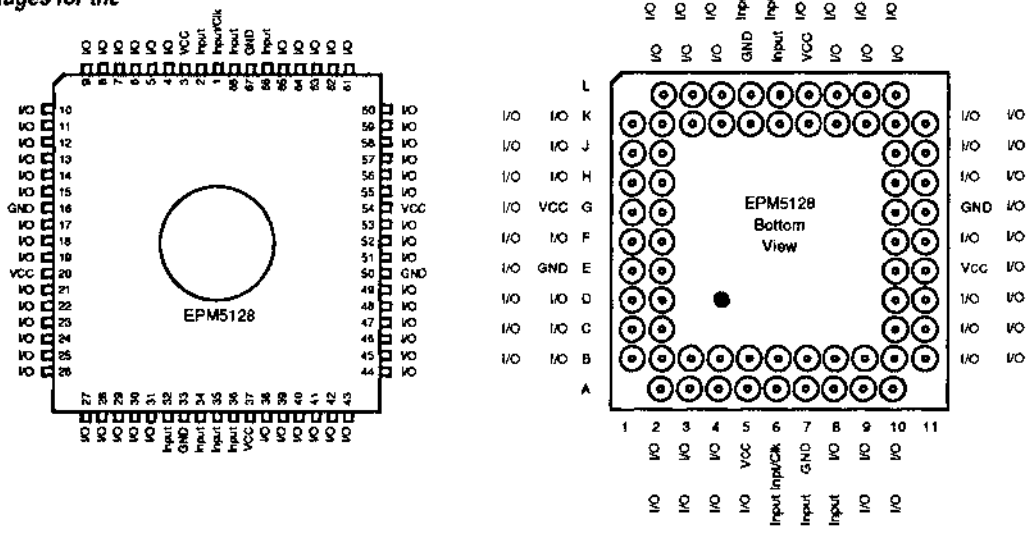
- 128 MAX architecture macrocells
- 256 shareable expander product terms allowing over 32 product terms in a single macrocell
- High-speed multiple LAB architecture
  - t_{PD} as fast as 25 ns
  - Counter frequencies of 50 MHz
  - Pipelined data rates of 62 MHz
- Programmable I/O architecture allowing as many as 60 inputs or 52 outputs
- Available in 68-pin ceramic windowed or plastic JLCC one-time-programmable package - Also available in a ceramic 68-pin PGA package

General Description

The Altera EPM5128 is a user-configurable, high-performance MAX EPLD that serves as a high-density replacement for 7400-series SSI and MSI TTL and CMOS logic. For example, a 74161 counter uses only 3% of the EPM5128. The EPM5128 can replace over 60 TTL MSI and SSI components and integrate multiple 20- and 24-pin low-density PLDs. Figure 14 shows the J-leaded and PGA package diagrams for the EPM5128.

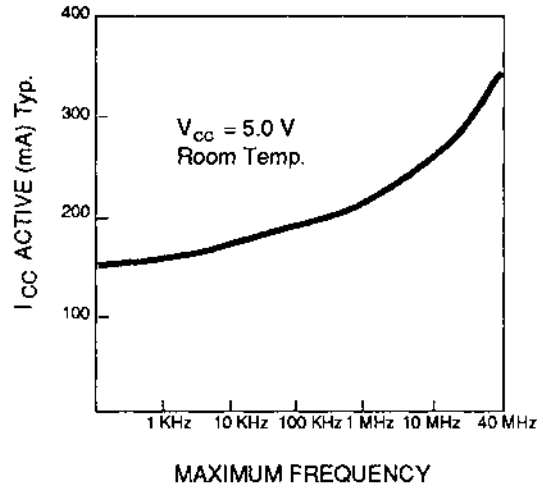
2

Figure 14. J-Lead and PGA Packages for the EPM5128



Typical power consumption versus frequency for the EPM5128 is shown in Figure 15.

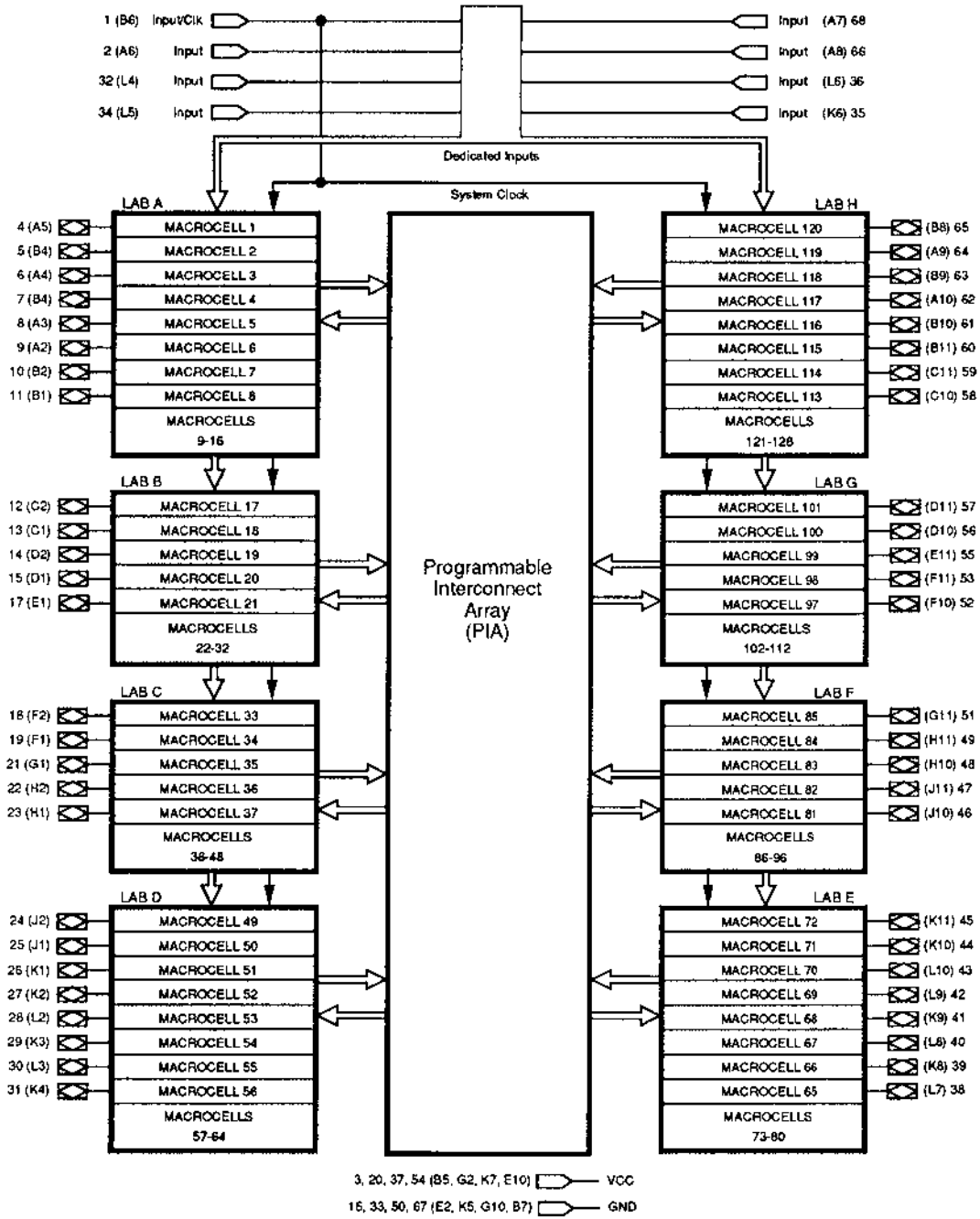
Figure 15. I_{CC} vs. Frequency



The EPM5128 consists of 128 macrocells equally divided into 8 Logic Array Blocks, each containing 16 macrocells (see Figure 16). Each LAB also contains 32 expander product terms. The EPM5128 has 8 dedicated input pins, one of which may be used as a synchronous system clock. The EPM5128 contains 52 I/O pins that may be configured for input, output, or bidirectional data flow. Four of the LABs have 8 I/O pins, and 4 of the LABs have 5 I/O pins.

Figure 16. EPM5128 Block Diagram

Note: Parentheses indicate 68-pin PGA package pin numbers.



Absolute Maximum Ratings

Note: See the *Design Recommendations* Data Sheet.

Symbol	Parameter	Conditions	Min	Max	Unit
V_{CC}	Supply voltage	With respect to GND	-2.0	7.0	V
V_{PP}	Programming supply voltage	*See Note (1)*	-2.0	13.5	V
V_I	DC input voltage		-2.0	7.0	V
I_{MAX}	DC V_{CC} or GND current			500	mA
I_{OUT}	DC output current, per pin		-25	25	mA
P_D	Power dissipation			2500	mW
T_{STG}	Storage temperature	No bias	-65	+150	° C
T_{AMB}	Ambient temperature	Under bias	-65	+135	° C
T_J	Junction temperature	Under bias		+150	° C

Recommended Operating Conditions

Symbol	Parameter	Conditions	Min	Max	Unit
V_{CC}	Supply voltage		4.75	5.25	V
V_I	Input voltage		0	V_{CC}	V
V_O	Output voltage		0	V_{CC}	V
T_A	Operating temperature	For Commercial	0	+70	° C
T_A	Operating temperature	For Industrial	-40	+85	° C
T_C	Case temperature	For Military	-55	+125	° C
t_R	Input rise time			100	ns
t_F	Input fall time			100	ns

DC Operating Conditions

V_{CC} = 5 V ± 5 %, T_A = 0° C to 70° C for commercial, *See Note (2)*

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
V_{IH}	High level input voltage		2.0		V_{CC} + 0.3	V
V_{IL}	Low level input voltage		-0.3		0.8	V
V_{OH}	High level TTL output voltage	I_{OH} = -4 mA DC	2.4			V
V_{OL}	Low level output voltage	I_{OL} = 8 mA DC			0.45	V
I_I	Input leakage current	$V_I = V_{CC}$ or GND	-10		+10	µA
I_{OZ}	3-state output off-state current	$V_I = V_{CC}$ or GND	-40		+40	µA
I_{CC1}	V_{CC} supply current (standby)	$V_I = V_{CC}$ or GND		150	225	mA
I_{CC3}	V_{CC} supply current	$V_I = V_{CC}$ or GND No load, f = 1.0 MHz *See Note (3)*		155	250	mA

Capacitance

Symbol	Parameter	Conditions	Min	Max	Unit
C_{IN}	Input capacitance	V_{IN} = 0 V, f = 1.0 MHz		10	pF
C_{OUT}	Output capacitance	V_{OUT} = 0 V, f = 1.0 MHz		20	pF

AC Operating Conditions

V_{CC} = 5 V ± 5 %, T_A = 0° C to 70° C for commercial

External Timing Parameters			EPM5128-1 (4)		EPM5128-2		EPM5128		
Symbol	Parameter	Conditions	Min	Max	Min	Max	Min	Max	Unit
t_{PD1}	Input to non-registered output	C1 = 35 pF		25		30		35	ns
t_{PD2}	I/O input to non-reg. output	C1 = 35 pF		40		45		55	ns
t_{SU}	Setup time		15		20		25		ns
t_{H}	Hold time		0		0		0		ns
t_{CO1}	Clock to output delay	C1 = 35 pF		14		16		20	ns
t_{ASU}	Asynchronous setup time		5		6		8		ns
t_{AH}	Asynchronous hold time		5		8		10		ns
t_{CH}	Clock high time		8		10		12.5		ns
t_{CL}	Clock low time		8		10		12.5		ns
t_{ACO1}	Asynch clock to output delay	C1 = 35 pF		25		30		35	ns
t_{CNT}	Minimum clock period			20		25		30	ns
f_{CNT}	Internal maximum frequency		50		40		33.3		MHz
f_{MAX}	Max frequency; pipelined data		62.5		50		40		MHz

For complete information on internal timing parameters, refer to *Application Brief 75 (MAX EPLD Timing)*.

Internal Timing Parameters			EPM5128-1		EPM5128-2		EPM5128		
Symbol	Parameter	Conditions	Min	Max	Min	Max	Min	Max	Unit
t_{IN}	Input pad and buffer delay			5		7		9	ns
t_{IO}	I/O input pad and buffer delay			6		6		9	ns
t_{EXP}	Expander array delay			12		14		20	ns
t_{LAD}	Logic array delay			12		14		16	ns
t_{LAC}	Logic control array delay			10		12		13	ns
t_{OD}	Output buffer and pad delay	C1 = 35 pF		5		5		6	ns
t_{ZX}	Output buffer enable delay			10		11		13	ns
t_{XZ}	Output buffer disable delay	C1 = 5 pF		10		11		13	ns
t_{SU}	Register setup time		6		8		10		ns
t_{LATCH}	Flow-through latch delay			3		4		4	ns
t_{RD}	Register delay			1		2		2	ns
t_{COMB}	Combinatorial delay			3		4		4	ns
t_{H}	Register hold time		6		8		10		ns
t_{IC}	Clock delay			14		16		18	ns
t_{ICS}	System clock delay			2		2		3	ns
t_{FD}	Feedback delay			1		1		2	ns
t_{PRE}	Register preset time			5		6		7	ns
t_{CLR}	Register clear time			5		6		7	ns
t_{PIA}	Progr. Interconn. Array delay			14		16		20	ns

2

Notes to tables:

(1) Minimum DC input is -0.3 V. During transitions, the inputs may undershoot to -2.0 V for periods less than 20 ns.

(2) Typical values are for T_A = 25° C and V_{CC} = 5 V.

(3) Measured with device programmed as a 16-bit counter in each LAB.

(4) This version is in development. Consult factory.

Features

Advanced Information

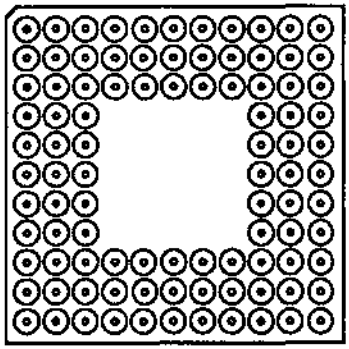
- 128 macrocells optimized for pin-intensive applications easily integrating over 60 TTL MSI and SSI components
- High pin count for 16- or 32-bit data paths
- 256 shareable expander product terms incorporated in each EPM5130
  - More than 32 product terms contained in a single macrocell
  - 128 additional latches provided by cross-coupling expanders
  - All inputs can be latched without using macrocells
- 20 high-speed dedicated inputs for fast latching of 16-bit functions
- Multiple LAB architecture ensuring high speeds
  - t_{PD} as fast as 25 ns
  - Counter frequencies of 50 MHz
  - Pipelined data rates of 62 MHz
- Synchronous clocking for fast CLOCK-to-Q delays for bus-oriented functions
- Programmable I/O architecture allowing up to 84 inputs or 64 outputs in a PGA or 100-pin Plastic Quad Flat Pack (PQFP) package

General Description

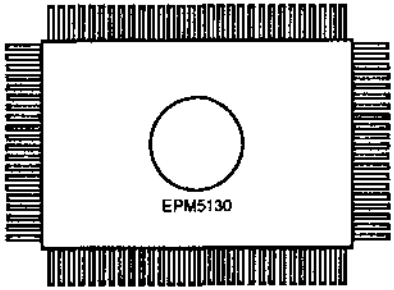
The Altera EPM5130 is a user-configurable, high-performance MAX EPLD that is optimized for pin-intensive designs and serves as a high-density replacement 7400-series SSI and MSI TTL and CMOS logic. A single EPM5130 can quickly integrate multiple 20- and 24-pin low-density PLDs and high pin count subsystems, such as custom DMA controllers. In addition, it can handle a 32-bit data path with enough I/O for control use. Package diagrams for the EPM5130 are shown in Figure 17.

2

Figure 17. QFP and PGA Packages for the EPM5130



EPM5130
PGA Package Bottom View

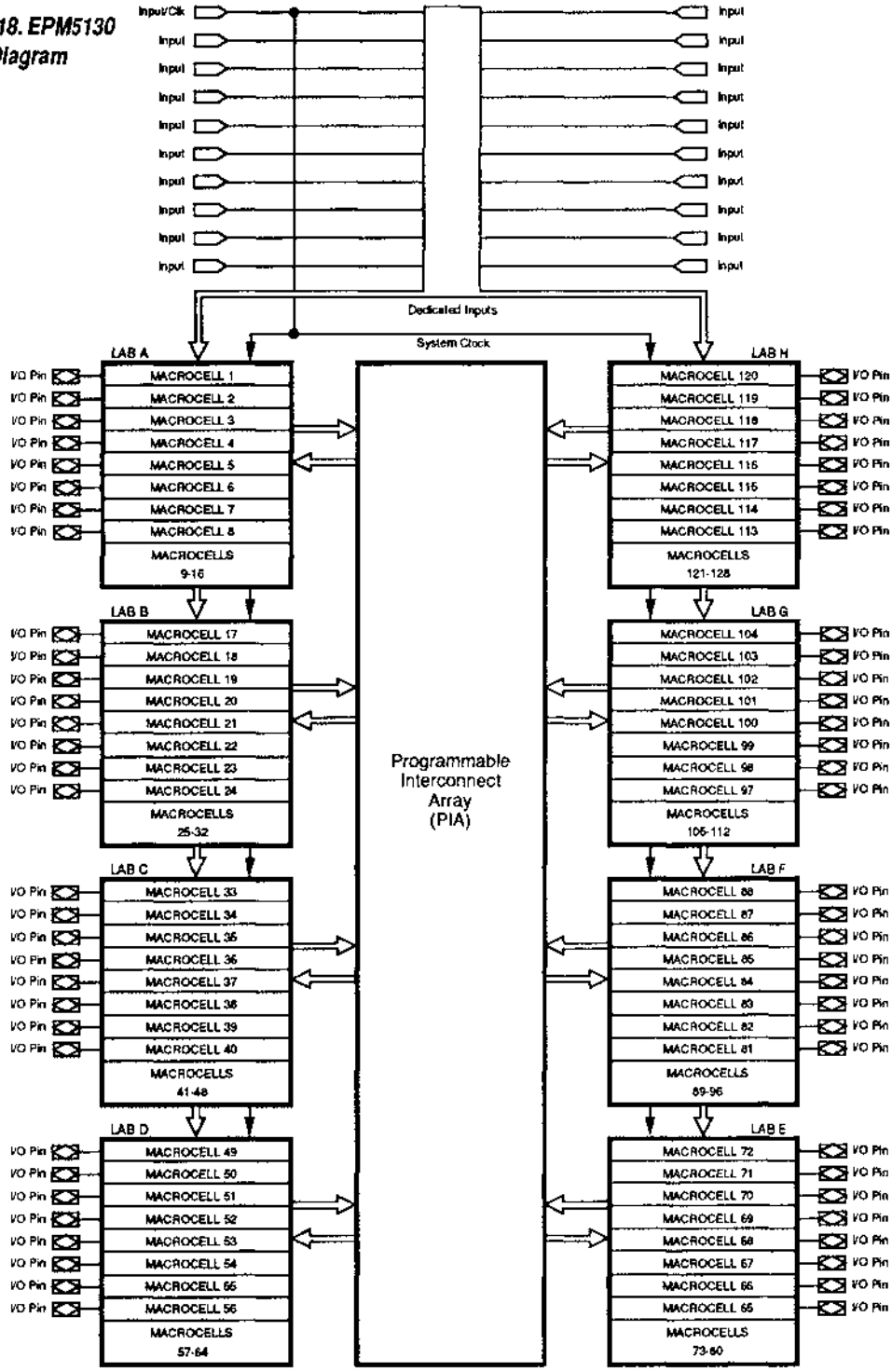


EPM5130
QFP Package

The EPM5130 consists of 128 macrocells equally divided into 8 Logic Array Blocks (LABs) that each contain 16 macrocells and 32 expander product terms. See Figure 18. Expander product terms can be used and shared by all macrocells in the device to ensure efficient use of device resources. Because the LAB is very compact, the high speeds required by most I/O subsystems are maintained.

The EPM5130 has 20 dedicated input pins that allow high-speed input latching of 16-bit functions. One of these inputs may be configured as a synchronous system clock to provide enhanced clock-to-output delays for bus-oriented functions. The EPM5130 also has 64 I/O pins, 8 in each LAB, that may be configured for input, output, or bidirectional data flow. Dual feedback on the I/O pins provides the most efficient use of device pin resources.

Figure 18. EPM5130 Block Diagram



2

Features

Advanced Information

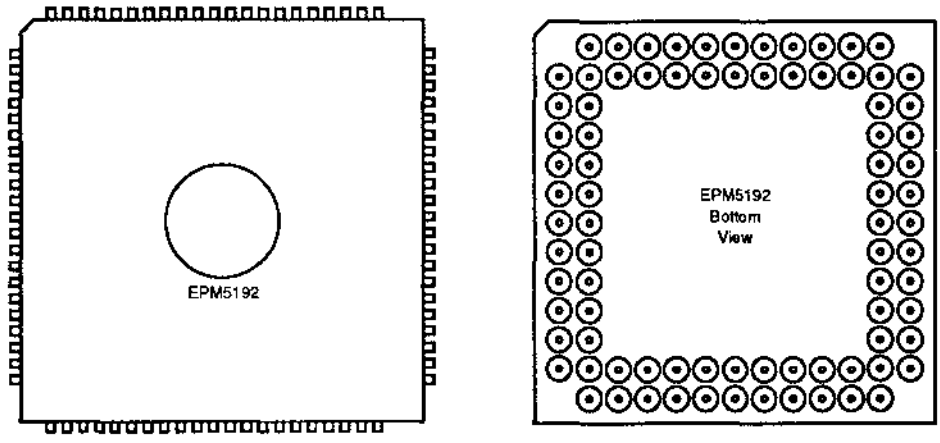
- 192 macrocells to easily replace over 100 TTL MSI devices and to integrate complete boards of logic into a single package
- 384 shareable expander product terms offering flexibility for register and combinatorial logic expansion
- Multiple LAB architecture ensuring high speeds
  - t_{PD} as fast as 30 ns
  - Counter frequencies of 40 MHz
  - Pipelined data rates of 50 MHz
- Programmable I/O architecture allowing as many as 72 inputs or 64 outputs - I/O tri-state buffers facilitate connections to system buses
- Available in 84-pin ceramic-windowed or plastic one-time-programmable JLCC package - Also available in a ceramic-windowed, 84-pin PGA package

General Description

The Altera EPM5192 is a user-configurable, high-performance MAX EPLD that serves as a high-density replacement for 7400-series SSI and MSI TTL and CMOS logic. The EPM5192 can replace over 100 TTL SSI and MSI components and integrate the logic contained in over 20 22V10-type devices. In addition, it accommodates other low-density PLDs of all sizes. These features enable the EPM5192 to easily integrate complete systems into a single device.

The EPM5192 is available in J-lead (JLCC) ceramic-windowed and plastic packages and in PGA ceramic-windowed packages. See Figure 19. Because ceramic-windowed packages are erasable, they may be used for quick and

Figure 19. J-Lead and PGA Packages for the EPM5192

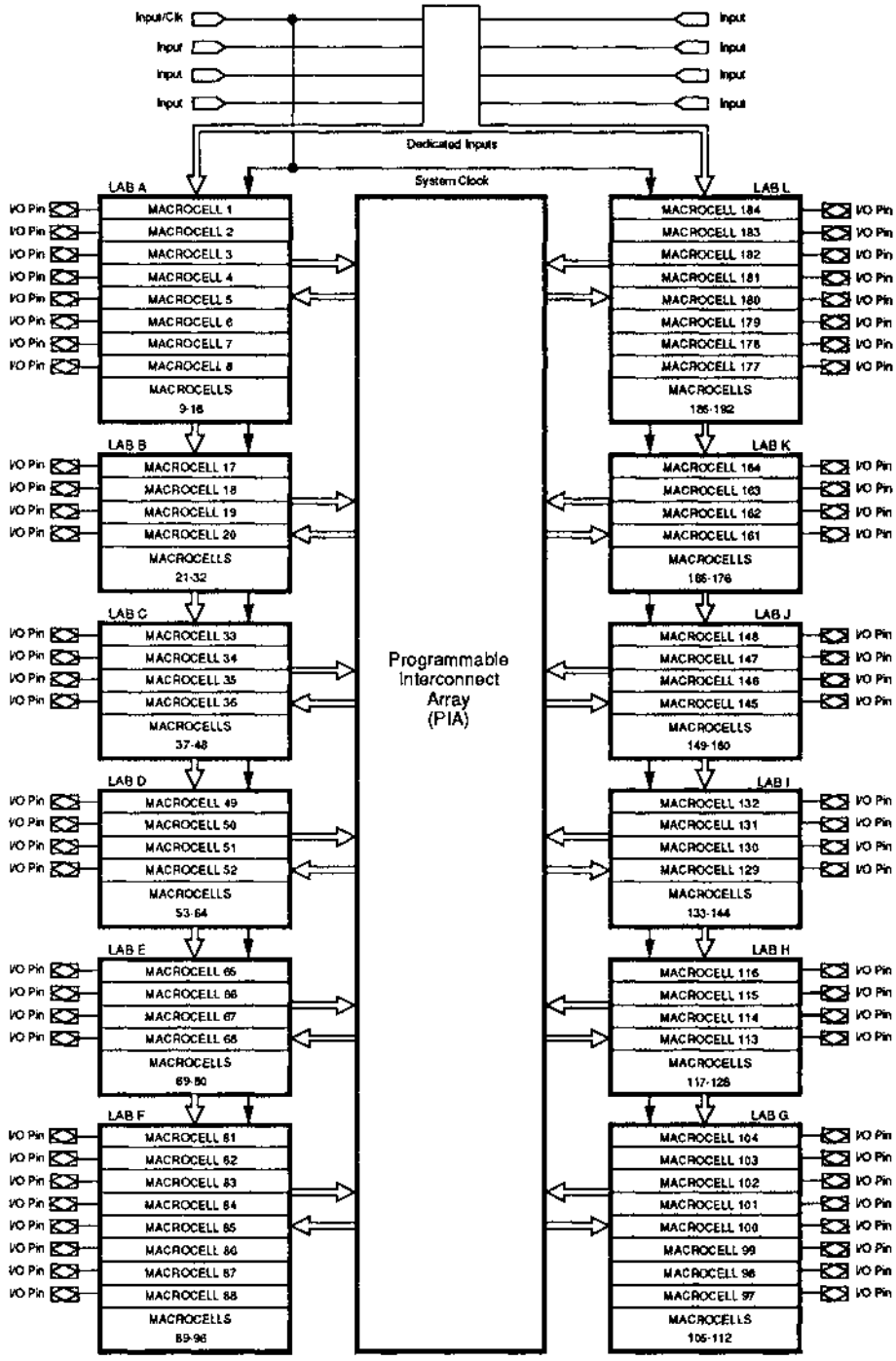


efficient system prototyping. On the other hand, plastic one-time-programmable (OTP) packages provide a low-cost solution for volume production.

The EPM5192 consists of 192 macrocells equally divided into 12 Logic Array Blocks (LABs), each containing 16 macrocells and 32 expander product terms (see Figure 20). Because each LAB is very compact, high performance is maintained and device resources are efficiently utilized.

The EPM5192 has 8 dedicated input pins, one of which may be used as a system clock. It can mix synchronous and asynchronous clocking in one device, facilitating easy integration of multiple sub-systems. The EPM5192 also has 64 I/O pins that may be configured for input, output or bidirectional data flow. To ensure high speeds for bus-oriented applications, 4 of the LABs contain 8 I/O pins. The remaining 8 LABs each contain 4 I/O pins.

Figure 20. EPM5192 Block Diagram



Design Recommendations for MAX EPLDs

January 1990 **Data Sheet**

MAX family devices have a unique architecture plus an advanced 0.8-micron CMOS EPROM process for exceptional performance. Like any high-performance CMOS process, systems must be designed with care to obtain maximum performance with minimum problems.

Operating Conditions

Operation of MAX devices at conditions above those listed under "Absolute Maximum Ratings" in the MAX device data sheets may cause permanent damage to the devices. This rating is a stress rating only. Functional operation of the device at these conditions or at any other conditions above those indicated in the operational sections of these data sheets is not implied. Exposure to absolute maximum ratings conditions for extended periods of time may affect device reliability. MAX EPLDs contain circuitry to protect device pins from high-static voltages or electric fields; however, precautions should be taken to avoid voltages higher than maximum-rated voltages.

2

For proper operation, input and output pins must be in the range GND < (V_{IN} or V_{OUT}) < V_{CC}. Unused inputs must be tied to V_{CC} or GND. Each set of V_{CC} and GND pins must be connected directly at the device, with power supply decoupling capacitors of at least 0.2 microfarads connected between them. For effective decoupling, each V_{CC} pin should be separately decoupled to GND, directly at the device. Decoupling capacitors should have good frequency response, such as the response in monolithic-ceramic types.

Noise Precautions

If more than 16 EPLD output pins are switching simultaneously, take precautions to minimize system noise. Certain board layouts can induce switching noise into the system from high-speed devices due to transmission line effects and radiated coupling. These effects can be minimized by using printed circuit boards with embedded V_{CC} and GND planes. They can also be lessened by restricting trace length in a board to under eight inches. In cases where long board traces or highly capacitive loads are impossible to avoid, a small (10-30 Ω) series resistance usually lessens undershoot and overshoot voltages if they cause a problem with a certain printed circuit board layout.

Device Erasure

MAX EPLDs begin to erase when exposed to lights with wavelengths shorter than 4000 Å. Since fluorescent lighting and sunlight fall into this range, opaque labels should be placed over the EPLD window to ensure long-term reliability. The recommended erasure procedure for EPLDs is exposure to UV light with a wavelength of 2537 Å. Erasure time depends on

the power of the UV lamp. Typically, 60 minutes is adequate to erase a MAX EPLD using a lamp with 12000 $\mu W/cm^2$ power rating. (Some low-power erasers may take longer.) MAX EPLDs may be damaged by exposure to high-intensity UV light for extended periods. MAX EPLDs may be erased and reprogrammed as often as necessary when the recommended erasure exposure levels are used.

ESD and Latch-Up Protection

EPLD input, I/O pins, and clock pins have been designed to resist the electrostatic discharge (ESD) and latch-up inherent in CMOS structures. Unless otherwise noted, each of the EPLD pins will withstand voltage energy levels exceeding 1500 V, per method specified by MIL STD 883C. The pins will not latch-up for input voltages between -1 V to V_{CC} + 1 V with currents up to 100 mA. During transitions, the inputs may undershoot to -2.0 V for periods less than 20 ns. Additionally, the programming pin is designed to resist latch-up to the 13.5 V maximum device limit.

Power Calculations

As with any CMOS device, power is a function of frequency and internal node switching. To obtain the most accurate power information, current consumption should be measured after the design is completed and the device is placed in the system.

If these precautions are taken during system and board design, MAX devices should provide superior system performance and design flexibility, regardless of design size or production volume.

PLDS-MAX / PLS-MAX

MAX+PLUS Programmable Logic Development System

January 1990 **Data Sheet**

Features

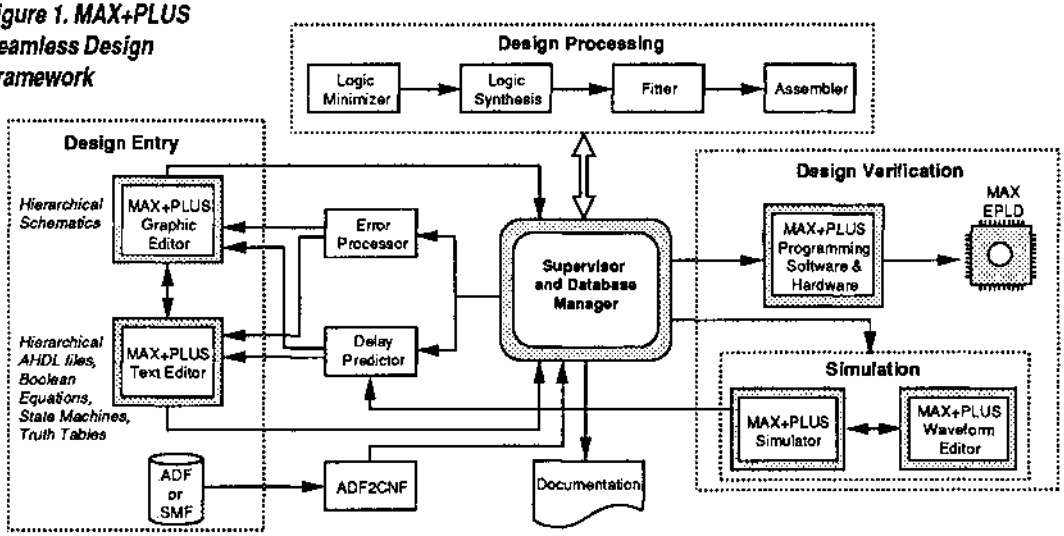
- ❒ Unified development system for MAX (Multiple Array MatriX) EPLDs
- ❒ Hierarchical design entry methods for both graphical and textual designs
  - Multi-level schematics and hardware language descriptions
  - 7400-series TTL and bus macrofunctions optimized for MAX architecture
  - Altera Hardware Description Language (AHDL) supporting state machines, Boolean equations, truth tables, arithmetic, and relational operations
  - Delay prediction for graphic and text designs
- ❒ Logic synthesis and minimization for quick and efficient processing
- ❒ Compiler that compiles a 100% utilized EPM5128 in only 10 minutes
- ❒ Automatic error location for AHDL text files and schematics
- ❒ Interactive Simulator with probe assignments for internal nodes
- ❒ Waveform Editor for entering and editing waveforms and viewing simulation results
- ❒ Used with IBM PS/2, PC-AT, or compatible machines
- ❒ EDIF industry-standard workstation and third-party interfaces

2

General Description

The Altera PLDS-MAX (Programmable Logic Development System) is a unified CAE system for designing logic with Altera's MAX family of EPLDs (Figure 1). PLDS-MAX includes design entry, design processing, timing

Figure 1. MAX+PLUS Seamless Design Framework



simulation, and device programming support. PLDS-MAX runs on IBM PS/2, PC-AT, or compatible machines, and provides tools to quickly and efficiently create and verify complex logic designs.

Compiles complex designs in minutes.

The MAX+PLUS software compiles designs for MAX EPLDs in minutes. Designs may be entered with a variety of design entry mechanisms. MAX+PLUS supports hierarchical entry of both Graphic Design Files (GDFs) with the MAX+PLUS Graphic Editor, and Text Design Files (TDFs) with the Altera Hardware Description Language (AHDL). The Graphic Editor offers advanced features such as multiple hierarchy levels, symbol editing, and a library of 7400-series devices as well as basic SSI gates. AHDL designs may be mixed into any level of the hierarchy or used on a stand-alone basis. AHDL is tailored especially for EPLD designs and includes support for complex Boolean and arithmetic functions, relational comparisons, multiple hierarchy levels, state machines with automatic state variable assignment, truth tables, and function calls.

In addition to multiple design entry mechanisms, MAX+PLUS includes the sophisticated MAX+PLUS Compiler that synthesizes and optimizes designs for MAX EPLDs in minutes. The Compiler uses advanced logic synthesis and minimization techniques in conjunction with heuristic fitting rules to efficiently place designs within MAX EPLDs. A programming file created by the Compiler is then used by MAX+PLUS to program MAX devices with standard Altera programming hardware.

Simulations may be performed with a powerful, event-driven timing simulator. The MAX+PLUS Simulator interactively displays timing results in the MAX+PLUS Waveform Editor. Hardcopy table and waveform output is also available. With the Waveform Editor, input vector waveforms may be entered, modified, grouped, and ungrouped. In addition, the Waveform Editor compares simulation runs and highlights the differences.

Automatic error location and delay prediction in schematics and text files.

The integrated structure of MAX+PLUS provides features such as automatic error location and delay prediction. If a design contains an error in either a schematic or a text file, MAX+PLUS flags the error *and* takes the user to the actual location of the error in the original schematic or text file. In addition, propagation delays of critical paths may be determined in both the Graphic and Text Editors with the delay predictor. After the source and destination nodes are tagged, the shortest and longest timing delays are calculated.

MAX+PLUS provides a seamless design framework using a consistent graphical user interface throughout. This framework simplifies all stages of the design cycle: design entry, processing, verification, and programming. In addition, MAX+PLUS offers on-line help to aid the user.

Design Entry

MAX+PLUS offers both graphic and text design entry methods. GDFs are entered with the MAX+PLUS Graphic Editor; Boolean equations, state machines, and truth tables may be entered with the MAX+PLUS Text Editor using AHDL. The ability to freely mix graphics and text files at all levels of the design hierarchy and to use either a top-down or bottom-up design method makes design entry simple and versatile.

MAX+PLUS also accepts third-party netlists, such as OrCAD, VIEWlogic, and FutureNet, as well as existing EPLD designs implemented with Altera's or Texas Instruments' A+PLUS, or Intel's iPLDS or iPLDS II systems.

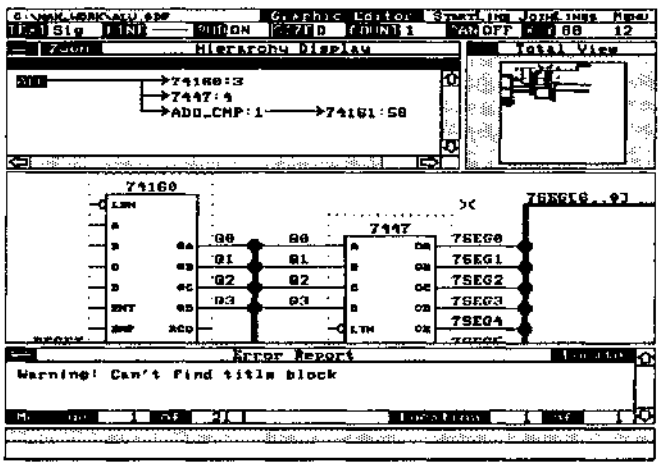
Graphic Editor

The Graphic Editor (Figure 2) provides a mouse-driven, multi-windowed environment in which commands are entered with pop-up menus or simple key-strokes. The Hierarchy Display window, shown at the top, lists all schematics used in a design. The designer navigates the hierarchy by placing the cursor on the name of the design to be edited and clicking the left mouse button. The Total View window (next to the Hierarchy window) shows the entire design. By clicking on an area in this window, the user is moved to that area of the schematic. The Error Report window lists all warnings and errors in the compiled design; selecting an error with the cursor highlights the problem node and symbol. A design is edited in the main area, which may be enlarged by closing the auxiliary windows.

2

Figure 2. MAX+PLUS Graphic Editor

The Graphic Editor provides a multi-windowed, menu-driven environment. The auxiliary windows may be closed to increase work space.



When entering a design, the user may choose from a library of over 200 7400-series and special-purpose macrofunctions that are all optimized for MAX architecture. In addition, the designer may create custom functions that can be used in any MAX+PLUS design.

To take advantage of the hierarchy features, the user first saves the entered design so the Graphic Editor can automatically create a symbol representing

the design. This symbol may be used in a higher-level schematic or in another design. It may also be modified with the Symbol Editor.

Tag-and-drag editing is used to move individual symbols or entire areas. Lines stay connected with orthogonal rubberbanding. A design may be printed on an Epson FX-compatible printer, or plotted on an HP- or Houston Instruments-compatible plotter.

Symbol Editor

The MAX+PLUS Symbol Editor enables the designer to create or modify a custom symbol representing a GDF or TDF. It is also possible to modify input and output pin placement of an automatically generated symbol.

The created symbol represents a lower-level design, described by a GDF or TDF. The lower-level design represented by the symbol may be displayed with a single command that invokes either the Graphic Editor for schematics or the Text Editor for AHDL designs.

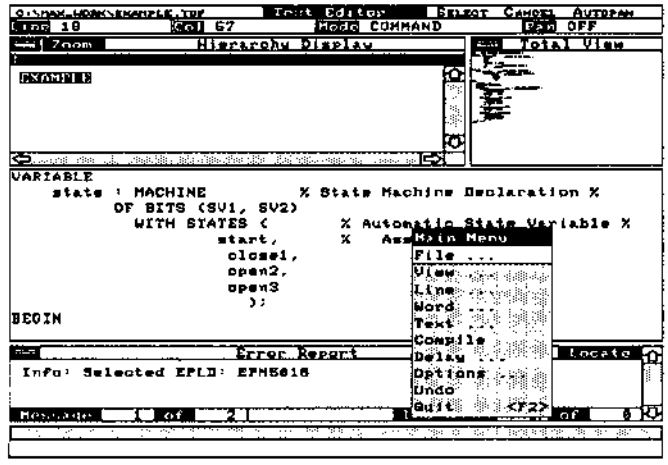
AHDL

The Altera Hardware Description Language (AHDL) is a high-level, modular language used to create logic designs for MAX EPLDs. It is completely integrated into MAX+PLUS, so AHDL files may be created, edited, compiled, simulated, and programmed from within MAX+PLUS.

AHDL provides support for state machines (see Figure 3), truth tables, and Boolean equations, as well as arithmetic and relational operations. AHDL is hierarchical, which allows frequently-used functions such as TTL and bus macrofunctions to be incorporated in a design. AHDL supports complex arithmetic and relational operations, such as addition, subtraction, equality, and magnitude comparisons, with the logic functions automatically generated. Standard Boolean functions, including AND, OR, NAND, NOR, XOR, and XNOR are also included. Groups are fully supported so operations may be performed on groups as well as on single variables. AHDL also

Figure 3. MAX+PLUS Text Editor

The Text Editor and AHDL offer such features as State Machine synthesis and automatic error location.



allows the designer to specify the location of nodes within MAX EPLDs. Together, these features enable complex designs to be implemented in a concise, high-level description (see Figure 4).

Figure 4. AHDL

AHDL allows complex arithmetic and relational operators to be described in a few lines.

```
TITLE "Timed Add and Compare function.";

DESIGN IS "add_cmp" DEVICE IS "EPM5128-2";

FUNCTION 74161 (LDN,A,B,C,D,ENT,ENP,CLRN,CLK) RETURNS (QA,QB,QC,QD,RCO);

SUBDESIGN add_cmp (
                a[7..0],            % inputs for adder/comparator %
                b[7..0],
                cmpen,
                clock,reset         :INPUT;

                result[7..0],
                elapse[3..0],
                equal,
                less_than,
                grtr_than,
                done                :OUTPUT;
)
VARIABLE
        timer           : 74161;    % timer is 74161 counter %
        register[7..0]  : dff;      % register is an octal FF %
        flag            : NODE;

BEGIN
        result[] = register[];      % Set up accumulate register %
        register[].clrn = reset;
        register[].clk = clock;

        register[] = a[] + b[];     % this is the actual addition %

        flag = (register[] != 0); %set flag high if register is not empty%

        done = flag;

        timer.enp = cmpen & flag; % connect inputs for timer (74161) %
        timer.clk = clock;
        timer.clrn = reset;

% elapse is the number of clock cycles it takes to do add %

        elapse[3..0] = (timer.QA,timer.QB,timer.QC,timer.QD);

        equal = ( a[] == b[]);      % The comparator section %
        less_than = (a[] < b[]);
        grtr_than = (a[] > b[]);

END;
```

Text Editor

The MAX+PLUS Text Editor enables the user to view and edit text files within the MAX+PLUS environment. Any ASCII text file, including Vector Files, Table Files, Report Files, and AHDL Text Design Files (TDFs) may be viewed and edited without having to exit to DOS.

The Text Editor parallels the Graphic Editor's menu structure. It has a Hierarchy Display and a Total View window for moving through the hierarchy levels and around the design. It includes automatic error location and hierarchy traversal. If an error is found in a TDF during compilation, the Text Editor is automatically invoked and the line of AHDL code where the error occurred is highlighted. In addition, a design may use both text and graphic files. As the designer traverses the hierarchy, the Text Editor is invoked for text files, and the Graphic Editor is invoked for schematics.

Symbol Libraries

The library provided with MAX+PLUS contains the most commonly used 7400-series devices such as counters, decoders, encoders, shift registers, flip-flops, latches, and multipliers, as well as special bus macrofunctions, all of which increase design productivity. Because of the flexible architecture of MAX EPLDs (that includes asynchronous preset and clear), true TTL device emulation is achieved. Altera has also created special purpose bus macrofunctions for designs that use buses. All macrofunctions have been optimized to maximize speed and utilization. Table 1 lists the main macrofunctions currently available. Refer to the ***MAX+PLUS TTL MacroFunctions*** manual for more information on TTL macrofunctions. Contact Altera Applications at (408) 984-2805 ext. 102 for updates to the MacroFunction Library, since new devices are regularly added.

Table 1. TTL MacroFunction Library

TTL Macrofunctions:

Adders:	8FADD, 7480, 7482, 7483, 74183
ALU:	74181
AND-OR Gates:	7452
Comparators:	8MCOMP, 7485, 74518
Code Converters:	74184, 74185
Counters:	4COUNT, 8COUNT, 16CUDSLR, GRAY4, UNICNT, 7493, 74160, 74161, 74162, 74163, 74190, 74191, 74192, 74193, 74393
Decoders:	7442, 7443, 7444, 7445, 7446, 7447, 7448, 7449, 74138, 74139, 74154, 74155, 74156
Encoders:	74147, 74148
Frequency Divider:	FREQDIV
Latches:	INPLTCH, NANDLTCH, NORLTCH, 7475, 7477, 74116, 74259, 74279, 74373
Multipliers:	MULT2, MULT4, MULT24, 74261
Multiplexers:	21MUX, 74151, 74153, 74157, 74158, 74298
Parity Generators:	74180, 74280
Registers:	7470, 7471, 7472, 7473, 7474, 7476, 7478, 74173, 74174, 74175, 74178, 74273, 74374
Shift Registers:	BARRELST, 7491, 7494 ,7496, 7499, 74164, 74165, 74166, 74179, 74194, 74198
SSI Gates:	CBUF, INHB, 7400, 7402, 7404, 7408, 7410, 7411, 7420, 7421, 7427, 7430, 7432, 7486
Storage Elements:	7498, 74278
True/Comp Elements:	7487

Bus Macrofunctions:

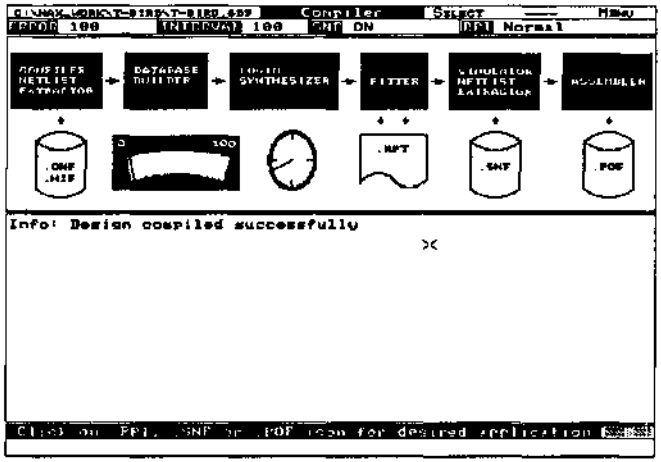
Adder:	8FADDB
Buffers:	74240B, 74241B, 74244B
Comparators:	8MCOMPB, 74518B
Counter:	16CUDSRB
Latches:	74373B, 74841B, 74842B
Multiplexers:	74151B
Multipliers:	MULT4B
Parity Generators:	74180B, 74280
Registers:	74174B, 74273B, 74374B, 74821B, 74822B, 74823B, 74824B, 74825B, 74826B
Shift Registers:	BARRLSTB, 74164B, 74165B

Design Processing

The MAX+PLUS Compiler processes MAX designs (see Figure 5). The Compiler offers options that speed the processing and analysis of a design. The user can set the degree of detail of the Report File and the maximum number of errors generated. In addition, the user may select whether or not to extract a netlist file for simulation.

Figure 5. MAX+PLUS Compiler

The Compiler uses minimization, logic synthesis, and heuristic fitting algorithms to place designs into MAX EPLDs.



The Compiler compiles a design in increments. If a design has been previously processed, only the portion of the design that has been changed is re-extracted, which decreases the compilation time. This "Make" facility is an automatic feature of the Compile command.

The first module of the Compiler, the Compiler Netlist Extractor, extracts the netlist that is used to define the design from each file. At this time, design rules are checked for any errors. If errors are found, the Graphic Editor is invoked when the error appears in a GDF, and the Text Editor is invoked when the error appears in a TDF. The Error Report window in both editors highlights the location of the error. A successfully-extracted design is built into a database to be used by the Logic Synthesizer.

The Logic Synthesizer module translates and optimizes the user-defined logic for the MAX architecture. The design is first minimized with SALSA (Speedy Altera Logic Simplification Algorithm). Any unused logic within the design is automatically removed. The Logic Synthesizer uses expert system synthesis rules to factor and map logic within the multi-level MAX architecture. It then chooses the approach that ensures the most efficient use of silicon resources.

The next module, the Fitter, uses heuristic rules to optimally place the synthesized design into the chosen MAX EPLD. For MAX devices that

have a Programmable Interconnect Array (PIA), the Fitter also routes the signals across this interconnect structure, so the designer doesn't have to worry about placement and routing issues. A Report File (**.RPT**) is issued by the Fitter, which shows design implementation as well as any unused resources in the EPLD. The designer can then determine how much additional logic may be placed in the EPLD.

A Simulator Netlist File (**.SNF**) may be extracted from the compiled design by the Simulator Netlist Extractor if simulation is desired. Finally, the Assembler creates a Programmer Object File (**.POF**) from the compiled design. This file is used with Altera hardware to program the desired part.

As a result of the advanced synthesis and minimization techniques employed by the Compiler, designs may be placed within the architecture in a matter of minutes. For example, a 16-bit counter/shift register compiles in less than 1 minute on a 16-MHz 386-based PC. The Compiler is equally efficient when compiling complex designs. For example, a series of 5 serially-linked multiplier/adder circuits that uses 100% of the macrocells and 95% of all expanders in an EPM5128 takes only 10 minutes to compile on a 20-MHz 386-based PC.

Delay Prediction and Probes

MAX+PLUS includes powerful analysis tools to verify and analyze the completed design. Delay analysis with the delay predictor may be performed interactively in the Graphic Editor, or in the Simulator. The Simulator is interactive and event-driven, yielding true timing and functional characteristics of the compiled design.

The delay predictor provides instant feedback about the timing of the processed design. After selecting the start point and end point of a path, the designer may determine the shortest and longest propagation delays of speed-critical paths.

In addition, a designer may use probes to mark internal nodes in a design. The designer may enter a probe by placing the cursor on any node in a graphic design, selecting the **SPE** (**Symbol : Probe : Enter**) command, and then entering a unique name to define the probe. This name may then be used in the Graphic Editor, Simulator, and Waveform Editor to reference that node, so that lengthy hierarchical path names are avoided.

Simulator

Input stimuli can be defined with a straightforward vector input language, or waveforms can be directly drawn using the Waveform Editor. Outputs may also be viewed in the Waveform Editor, or hardcopy table and waveform files may be printed.

The Simulator uses the Simulator Netlist File (SNF) extracted from the compiled design to perform timing simulation with 1/10-nanosecond resolution. A Command File may be used for batch operation, or commands may entered interactively. Simulator commands allow the user to halt the simulation dependent on user-defined conditions, to force and group nodes, and perform AC detection.

If flip-flop setup or hold times have been violated, the Simulator warns the user. In addition, the minimum pulse width and period of oscillation may be defined. If a pulse is shorter than the minimum pulse width specified, or if a node oscillates for longer than the specified time, the Simulator issues a warning.

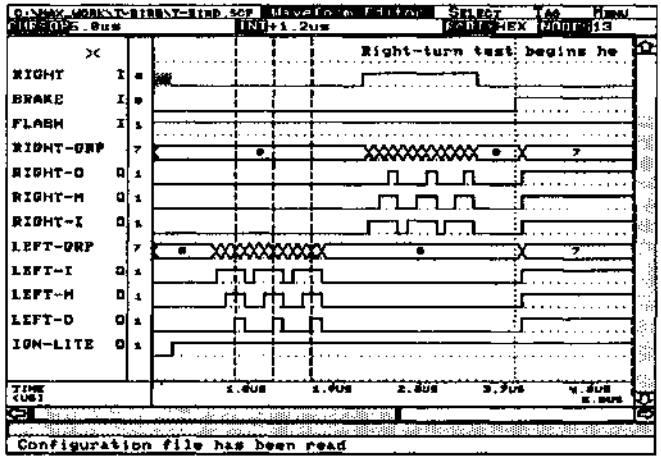
Waveform Editor

The MAX+PLUS Waveform Editor, shown in Figure 6, provides a mouse-driven environment in which timing waveforms may be viewed and edited. It functions as a logic analyzer, enabling the user to observe simulation results. Simulated waveforms may be viewed and manipulated at multiple zoom levels. Nodes may be added, deleted, and combined into buses, which may contain up to 32 signals represented in binary, octal, decimal, or hexadecimal format. Logical operators may also be performed on pairs of waveforms, so that waveforms may be inverted, ORed, ANDed, or XORed together.

2

Figure 6. MAX+PLUS Waveform Editor

With the Waveform Editor, input stimuli can be entered and modified, and Simulator waveforms can be viewed and compared.



The Waveform Editor includes sophisticated editing features to define and modify input vectors. Input waveforms are created with the mouse and familiar text editing commands. Waveforms may be copied, patterns may be repeated, and blocks may be moved and copied. For example, all or part of a waveform may be contracted to simulate the increase in clock frequency.

The Waveform Editor also compares and highlights the difference between two different simulations. A user may simulate a design, observe and edit the results, and then resimulate the design, and the Waveform Editor will show the results superimposed upon each other to highlight the differences.

Device Programming

PLDS-MAX contains the basic hardware and software for programming the MAX EPLD family. Adapters are included for programming the EPM5032 and EPM5128 devices. Additional adapters supporting other MAX devices may be purchased separately. MAX+PLUS programming software drives the PC-AT or PS/2 add-in card and uses standard Altera programming hardware. The designer can use MAX+PLUS to program and verify MAX EPLDs. If the security bit of the device is not set to **ON**, the designer may also read the contents of a MAX device and use this information to program additional devices.

System Requirements

Minimum System Configuration:

- ❒ PC-AT or compatible computers; IBM PS/2 model 50 or higher
- ❒ PC-DOS version 3.1 or higher
- ❒ 640 Kbytes of DOS RAM, recommended 1 Mbyte of Expanded Memory[1]
- ❒ EGA, VGA, or Hercules Monochrome display
- ❒ 20 Mbyte hard disk drive
- ❒ 1.2 Mbyte 5-1/4 in. or 1.44 Mbyte 3-1/2 in. floppy disk drive
- ❒ 3-button serial port mouse or MS-MOUSE-compatible bus and serial mouse
- ❒ Full 8-bit card slot for programming hardware

Recommended System Configuration:

- ❒ IBM PS/2 Model 70 or higher, or AT compatible 386 20-MHz machine or higher
- ❒ PC-DOS version 3.3
- ❒ 640 Kbytes of DOS RAM
- ❒ 2 Mbytes of Expanded Memory with a LIM 3.2 (or higher) compatible EMS driver
- ❒ 40 Mbyte hard disk or larger
- ❒ 1.2 Mbyte 5-1/4 in. or 1.44 Mbyte 3-1/2 in. floppy disk drive
- ❒ VGA Graphics display
- ❒ 3-button serial port mouse
- ❒ Full card slot for programming

[1]Some larger designs may not compile or simulate without Expanded Memory.

2

Sample System Configurations

- ❒ Compaq 386-20 with 3 Mbytes of RAM using the CEMM Expanded Memory Manager, Compaq VGA display, Mouse Systems 3-button serial mouse
- ❒ Wyse 386-16 with Intel Above Board 286 and VEGA VGA card with NEC Multi-Sync II monitor, Logitech C-7 serial mouse
- ❒ IBM PS/2 Model 80 with 4 Mbytes of RAM using DOS 4.01 Expanded Memory Manager, Logitech Series 9 Mouse using pointing device port, EGA display
- ❒ Everex 386-25 using Extended Memory configured as expanded memory, Paradise VGA card, VGA monitor, Microsoft Bus mouse

Package Contents

PLS-MAX Contents

- ❒ Floppy disks containing all programs and files for MAX+PLUS software for both PC-AT and PS/2 platforms
- ❒ Documentation

PLDS-MAX Contents

- ❒ All software and documentation included in PLS-MAX
- ❒ Programing card: LP4 for PC-AT or LP5 for PS/2 PCs
- ❒ Programming module (PLE3-12A)
- ❒ PLED5032A programming adapter for EPM5032
- ❒ PLEJ5128A programming adapter for EPM5128
- ❒ EPM5032D, EPM5128J device samples
- ❒ PLAESW-PC 12-month software warranty and update service

PLAESW-PC

PLAESW-PC is a 12-month renewable warranty for all PC-based Altera software. This contract covers all software contained in PLDS-MAX. PLAESW-PC includes automatic upgrades to each new revision of Altera software and guarantees software support for new MAX family EPLDs introduced by Altera. It also includes a toll-free hotline and 24-hour modem interface to Altera's Electronic Bulletin Board service.

Ordering Information

- ❒ PLDS-MAX (PC-AT)
- ❒ PLDS-MAX/PS (PS/2 Model 50, 60, 70, 80)
- ❒ PLS-MAX (PC-AT or PS/2)

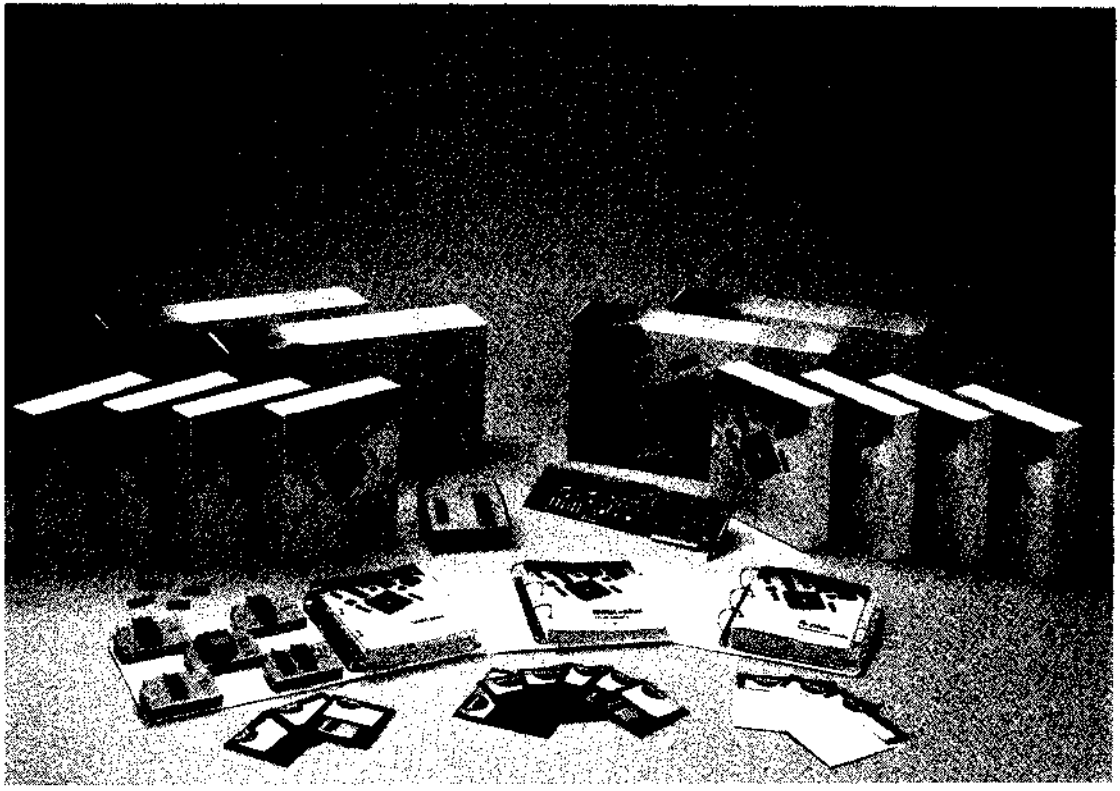
PLDS-ENCORE
Programmable Logic Development System

January 1990 **Data Sheet**

PLDS-ENCORE Contents

- ❒ Complete PLCAD-SUPREME system
- ❒ PLS-MAX Development System software
- ❒ PLS-SAM Development System software
- ❒ PLED5032A DIP adapter
- ❒ PLEJ5128A J-Lead adapter
- ❒ PLED448 DIP adapter
- ❒ PLAESW-PC, 12-month software warranty and update service
- ❒ Device samples

PLDS-ENCORE contents may be purchased separately.



2

Overview

PLDS-ENCORE is the most comprehensive EPLD development software package available. It supports entry, optimization, and verification of general purpose (EP-series and EPM (MAX)-series) and SAM EPLDs. EP-series designs are implemented with the PLCAD-SUPREME system, which includes the A+PLUS design software, LogiCaps schematic capture, state machine entry, TTL macrofunctions, and functional simulation software. MAX EPLD designs are implemented with PLS-MAX software. EPS448 (SAM) designs are implemented with PLS-SAM software. The PLDS-ENCORE Development System also includes the programming card and basic programming adapters necessary to program devices at a desktop.

PLDS-ENCORE provides comprehensive support at a discounted price.

Ordering Information

PLDS-ENCORE	(for PC-AT and compatible computers)
PLDS-ENCORE/PS	(for PS/2 Model 50, 60, 70, 80)

MAX+PLUS PLSA Tools
for Logic Synthesis Analysis

January 1990 **Data Sheet**

Features

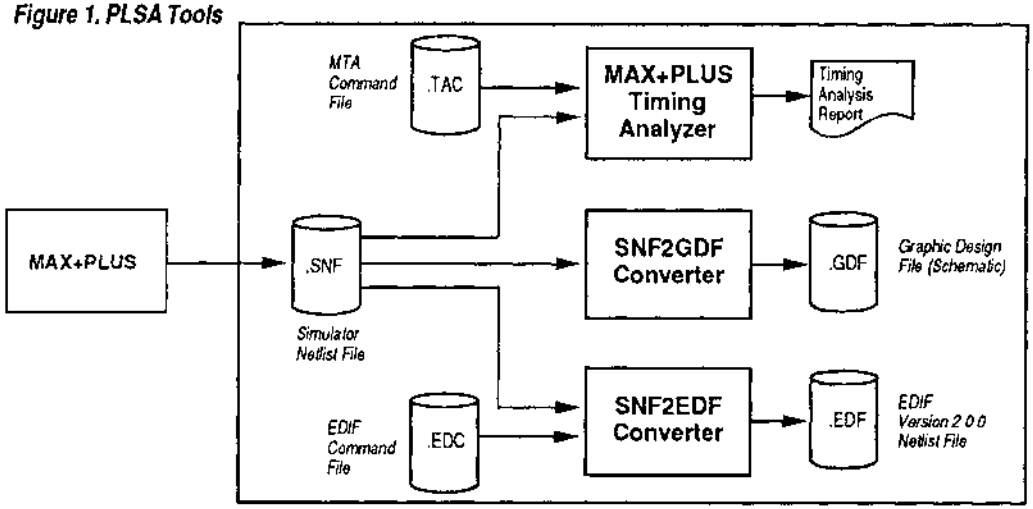
- Performs sophisticated analysis of AC timing in MAX EPLD designs.
- Facilitates critical delay path analysis within a design.
- Generates all setup and hold timing data for design verification.
- Automatically generates schematics from synthesized designs.
- Simplifies speed and density design optimization.
- Converts a design's netlist to the industry standard EDIF version 2 0 0 to facilitate translation to workstation-based board-level simulators.

General Description

Altera's PLSA (Programmable Logic Synthesis Analyzer) software consists of three applications programs (**MAX+PLUS Timing Analyzer**, **SNF2GDF Converter**, and **SNF2EDF Converter**). The first two enable the designer to analyze and fine-tune a compiled design. SNF2EDF then translates the design into the industry-standard EDIF format, allowing MAX+PLUS designs to be integrated into workstation environments for board-level simulation or further analysis. See Figure 1.

2

Figure 1. PLSA Tools



PLSA tools are especially valuable for optimizing design performance, since together they provide graphic and netlist representations of the synthesized logic as well as precise timing analysis.

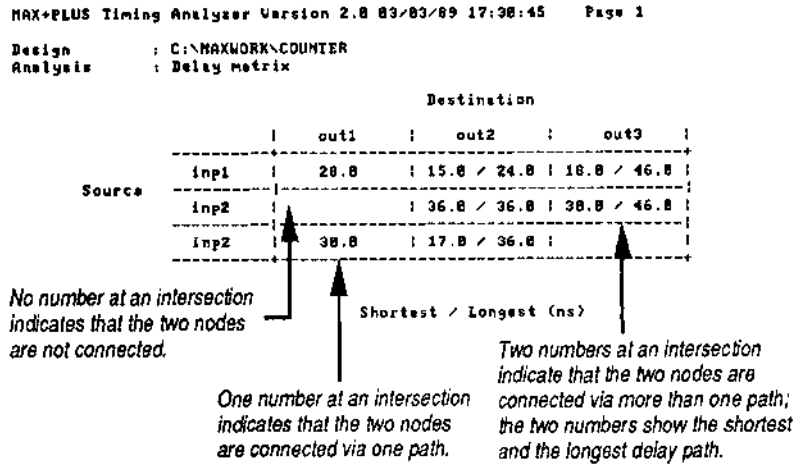
MAX+PLUS produces a general-purpose output file called Simulator Netlist File (SNF). The SNF of a design, compiled and synthesized byMAX+PLUS, contains the information of a fitted design, including logic functionality and discrete delays, as well as setup and hold requirements of any flip-flops in the design. This file may be used as a starting point for determining critical performance characteristics of a MAX EPLD design. Each PLSA tool uses this SNF as an input file (see Figure 1).

MAX+PLUS Timing Analyzer (MTA)

The MAX+PLUS Timing Analyzer (MTA) provides user-configurable reports that assist the designer in analyzing critical delay paths, setup and hold timing, and overall system performance of any MAX EPLD design. Critical paths identified by these reports may be displayed and highlighted .

Timing delays between multiple source and destination nodes may be calculated, thus creating a connection matrix giving the shortest and longest delay paths between all source and destination nodes specified (Figure 2). Or, the designer may specify that the detailed paths and delays between specific sources and destinations be shown.

Figure 2. Timing Analyzer Connection Matrix



The setup/hold option provides setup and hold requirements at the device pins for all pins that feed the **D**, **CLK**, or **ENABLE** inputs of flip-flops and latches. Critical source nodes may be specified individually, or setup and hold at all pins may be calculated. This information is then displayed in a table, one set of setup and hold times per flip-flop/latch.

The MTA also allows the user to print a complete list of all accessible nodes in a design, i.e., all nodes that may be displayed during simulation or delay prediction.

All MTA options may be listed in an MTA command file. With this file, the user may specify all information needed to configure the output.

SNF2GDF Converter

SNF2GDF converts the SNF into logic schematics represented with basic gates and flip-flop elements. It uses the SNF's delay and connection information and creates a series of schematics fully annotated with propagation delay and setup and hold information at each logic gate. Certain speed paths of a design may be specified for conversion, so the user may graphically analyze only thoses paths considered critical.

If State Machine or Boolean Equation design entry is used, SNF2GDF shows how the high-level description has been synthesized and placed into the MAX architecture. See Figures 3 and 4.

Figure 3. Schematic Submitted to SNF2GDF

This screen capture shows a schematic and its Boolean equation equivalent.

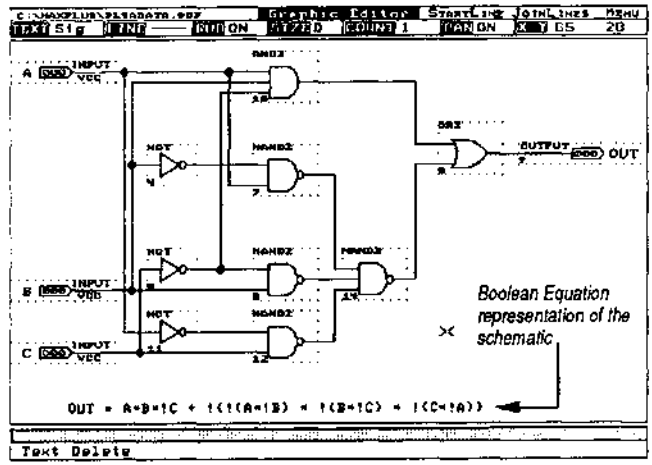
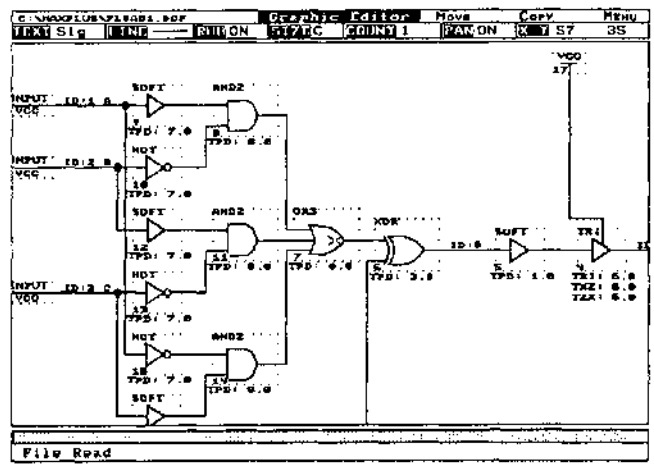


Figure 4. Schematic Converted & Annotated with SNF2GDF

This screen capture shows the converted schematic, displaying the results of logic synthesis as well as delay information.



SNF2EDF Converter

SNF2EDF provides a bridge to other CAE systems. It translates the SNF into the industry standard Electronic Design Interchange Format (EDIF version 2 0 0), supporting EDIFLEVEL 0 and 1 constructs. This human-readable file contains all information needed to create functional and timing models for board-level simulators. An optional command file may be used with SNF2EDF to customize the EDIF output for a specific workstation environment. Many workstation software vendors, such as Daisy/Cadnetix, Mentor Graphics, and Valid Systems, support or plan to support EDIF netlist formats for information transfer.

System Requirements

- ❒ PC-AT or compatible computers; IBM PS/2 models 50, 60, 70, 80
- ❒ MS-DOS version 3.1 or higher
- ❒ 640 Kbytes of RAM, 1 Mbyte of Expanded Memory with LIM (Lotus, Intel, Microsoft) version 3.2 (or higher) compatible EMS driver
- ❒ EGA,VGA, or Hercules Monochrome display
- ❒ 20 Mbyte hard disk drive
- ❒ 1.2 Mbyte 5-1/4 in. or 1.44 Mbyte 3-1/2 in. floppy disk drive
- ❒ 3-button serial port mouse

Product Contents

- ❒ Floppy diskettes containing all programs and files for PLSA software for both PC-AT and PS/2 platforms
- ❒ Documentation

Ordering Information

PLSA (contains diskettes for PC-AT and PS/2)

PLDVS

General Purpose PC-Based Tester

January 1990 **Data Sheet**

Features

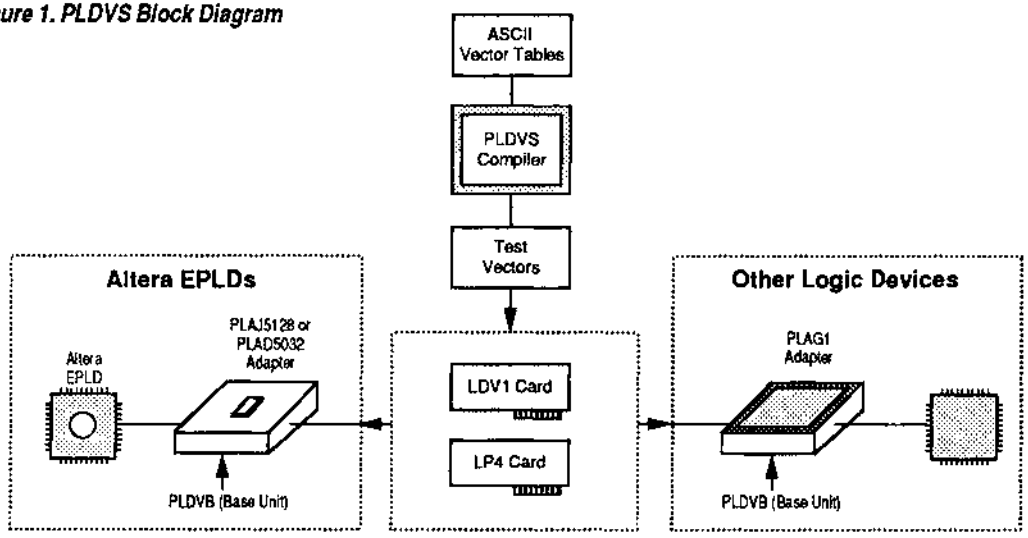
- Powerful PC-based generic functional tester
- 128 channels for data acquisition and generation
- Vector rate of 200,000 vectors per second
- Continuity test on all channels
- Pascal-like vector language
- Stored or algorithmic test pattern generation
- 8 high-level drivers, hardware assignable to any Device Under Test (DUT) pins
- 5 programmable voltage supplies
- Programs EPROM-based devices
- Fully integrated with MAX+PLUS software
- For gate arrays, standard cells, and PLDs

General Description

The Altera Personal Logic Design Verification System (PLDVS) is a low-cost, general-purpose tester for functional verification of TTL- or CMOS-compatible device prototypes such as gate array, standard cell, and programmable logic devices. PLDVS (Figure 1) can be used for both production and engineering applications.

2

Figure 1. PLDVS Block Diagram



The PLDVS package consists of two PC add-in boards, cables, DUT interface hardware, and the PLDVS Compiler. The LP4 and LDV1 add-in boards provide 5 individually programmable power supplies, I_{CC} measurement circuitry, 8 three-level drivers, and 128 independently-configurable, bidirectional test channels. Adapters for the EPM5032 AND EPM5128 MAX family parts and a generic adapter that can be configured for generic devices are also supplied.

PLDVS can be used for simple to very complex applications. First-time users can create simple test programs in minutes, and more advanced users can develop highly complex test routines. The PLDVS Compiler combines high-level Pascal-like programming features with a powerful vector language. Fast compilation times of 200 instructions per second and automatic error detection reduce the debug time required for test programs and circuits. Test results can be stored, manipulated, and displayed from high-level software.

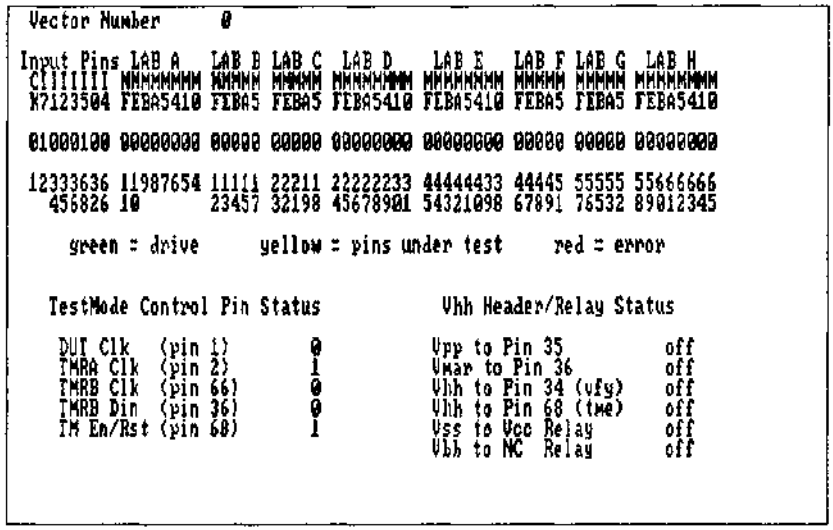
Compiler Interface

The PLDVS Compiler handles all hardware interactions, allowing the user to specify test requirements for the device rather than for test hardware pin mapping. It also performs complex algorithms for using stored vectors, algorithmically-generated vectors, and for passing procedure-based vector parameters.

The vector language uses high-level Pascal constructs and user-declared variables. DUT vector data can be stored in variables and processed in mathematical or logical expressions. Test set-up, test flow, data logging control, and display data acquisition run at 200,000 vectors per second.

Simple test programs created with the PLDVS Compiler can be run independently; larger and more complex test routines can be run by linking smaller, independent modules from within an executive shell routine written with Turbo Pascal. Figure 2 shows an example of data displays created from within an executive shell program.

Figure 2. Real-Time Data Display in PLDVS



Programming Language

The programming language includes specific commands for applying test vectors to the DUT, comparing results, and masking data. Basic language elements allow simple vector sets to be written, compiled, and run in minutes. Unlike traditional test-program languages, vector data can be integrated with Pascal constructs to create sophisticated, flexible algorithms. Figure 3 shows a program created to generate 16,000 test vectors.

The PLDVS Compiler provides additional instructions, including checking the continuity, setting programmable voltages, measuring I_{CC} and V_{CC} values, and verifying that the device being tested is correctly seated in its socket.

Figure 3. Compiler Routine

This routine algorithmically creates 16,000 test vectors.

```
MaskX $F ToCG 2;                { Only test the 4 'F' output bits

For M := 0 To 1 Do              { For both logic and arithmetic modes
 For A := #0000 To #1111 Do     { test all 16 possible 'A' input combinations
  For B := #0000 To #1111 Do    { with all 16 possible 'B' input combinations
   For S := #0000 To #1111 Do   { for all 16 possible functions
    For Ci := 0 To 1 Do         { in both carry and non-carry mode
     Begin

      IFDEF Display CheckKey; ENDIF

      Cy := 1-Ci;               { Invert logic of Ci    }

      Inputs  := (M<<13) + (Ci<<12) + (S<<8) + (B<<4) + A;

{
   | SELECTED | MODE |   LOGIC   |          |
   | FUNCTION |SELECT| FUNCTIONS |  ARITHMETIC FUNCTIONS  |
}
      Case S of
        #0000 : If M Then  F := ~A          Else  F := A + Cy;
        #0001 : If M Then  F := ~(A | B)    Else  F := (A | B) + Cy;
        #0010 : If M Then  F := ~A & B      Else  F := (A | ~B) + Cy;
```

Hardware Description

PLDVS hardware provides 5 programmable power supplies and 128 configurable test channels. All data channels are double-buffered, which allows data to be presented to the DUT after all channel states have been set. With this feature, any number of DUT pins may be changed simultaneously. The computer reads DUT output data by tri-stating the appropriate tester channels and reading the data through PLDVS driver channels.

All tester channels drive CMOS and TTL levels and can read back any TTL-compatible logic levels. Data is driven to or read from the DUT at a rate that depends on the computer and data transfers to the interface board. A 16-MHz 386 system generates 200,000 vectors per second.

PLDVS has 1 negative and 4 positive programmable power supplies. It also has 8 three-level drivers to drive DUT pins, which require a high-level voltage (5 to 12 V) in addition to normal logic level voltages. These drivers are used to enter test modes or to program EPLDs and EPROMs. I_{CC} measurement circuitry, which can measure supply currents from 50 µA to 500 mA, is also provided.

MAX+PLUS Interface

PLDVS can be integrated into Altera's MAX+PLUS development environment. MAX+PLUS software provides a user-friendly, menu-driven environment to generate and simulate designs for the MAX family of EPLDs. When used with MAX+PLUS software, PLDVS can provide fast hardware simulations and program MAX EPLDs. In addition, test vectors can be applied to the programmed parts to verify that they are functional.

MAX+PLUS software includes the MAX+PLUS Waveform Editor, which is a mouse-driven application for editing and displaying waveforms. Hardware test results can be viewed as functional waveforms, and nodes may be added, deleted, and grouped into buses. Actual output waveforms can also be graphically compared with simulated outputs.

Input vectors are defined either in a simple text file format known as a Vector File, or via waveform files created and edited in the Waveform Editor. Once the input vectors have been defined, they can be applied to the device continuously or with break conditions. PLDVS may be used entirely within the MAX+PLUS environment, without the PLDVS Compiler, to simulate and verify MAX EPLDs only.

MAX+PLUS software can also be used to generate input test waveforms and to display functional results for general purpose devices. The designer simply defines the channel map and input stimuli for the device to be tested. MAX+PLUS will apply the test vectors and display the results in the Waveform Editor or in an ASCII table format. For more information regarding MAX+PLUS software, refer to the *PLDS-MAX / PLS-MAX* Data Sheet.

2

Device Programming

PLDVS can read and program EPLDs and apply test vectors. Adapters are included for programming the EPM5128 and the EPM5032. In addition, the system provides full continuity-checked programming capability.

Adapters

PLDVS includes PLAD5032, a 28-pin DIP adapter socket for the EPM5032, and the PLAJ5128, a 68-pin J-lead adapter socket for the EPM5128. A generic socket card (PLAG1) that can be configured to support the specific device pinout is also included.

Specifications

Data Channels:	128
Tri-State on Cycle Basis:	Individually controlled
Vector Rate:	200 K 8-bit vectors/second
Programmable Supplies:	8
Pattern Size:	Unlimited (limited to hard disk capacity)
High-Level Programming:	Conditional jumping, conditional interrupt, graphical displays
Temperature:	10° to 43° C
Humidity:	8 to 80%

Minimum System Configuration

- ❒ IBM PC-AT or compatible computer with EGA or VGA graphics display
- ❒ PC-DOS version 3.1 or higher
- ❒ Borland Turbo Pascal Compiler version 5.0
- ❒ 20 Mbyte hard disk drive
- ❒ 1.2 Mbyte 5-1/4" disk drive
- ❒ 3 full card slots for PLDV1, LP4, and cables (one 16-bit slot required)

PLDVS Contents

- ❒ PLDV1 tester card, PLDVB base unit
- ❒ PLAD5032 28-pin DIP adapter socket
- ❒ PLAJ5128 68-pin J-lead adapter socket
- ❒ PLAG1 generic adapter board
- ❒ LP4 programming card
- ❒ PLDV Compiler manual
- ❒ PLDVC diskettes

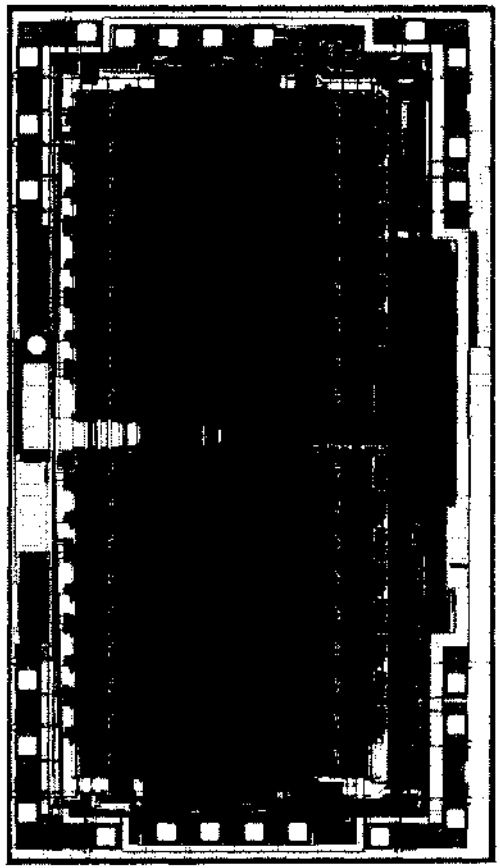
Ordering Information

PLDVS

Contents

Section 3 Application Notes and Briefs

3



Integrating PAL and PLA Devices with the EPM5032

January 1990 **Application Note 16**

Introduction

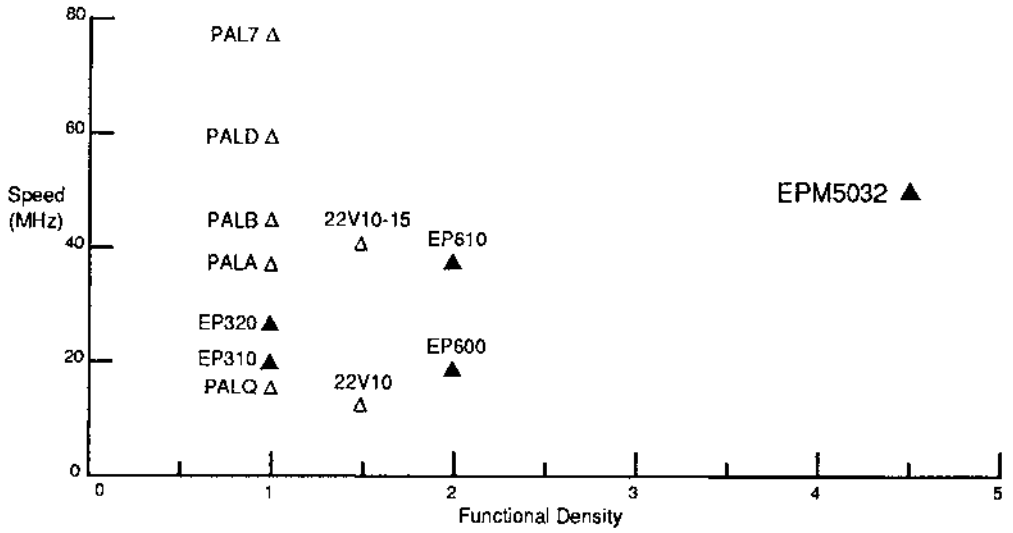
The EPM5032 is a second-generation Erasable Programmable Logic Device (EPLD) based on the Logic Array Block function developed for the Altera Multiple Array MatriX (MAX) device family. Its highly flexible architecture facilitates general-purpose logic design, and integrates existing functions based on multiple TTL MSI, PLA, and PAL elements. Other benefits include flexible logic utilization, advanced registers, better control of I/O pins, and better system performance. Refer also to *EPM5016 to EPM5032: MAX EPLDs with a Single LAB* in Section 2 of this handbook.

EPM5032 vs. PAL and PLA Architectures

The EPM5032 is a 28-lead, 32-macrocell member of the Altera MAX CMOS EPLD family. It may be programmed to accommodate 64 latches or 42 flip-flops, in addition to powerful combinatorial and control logic functions. The EPM5032 may also achieve flip-flop toggle rates of over 80 MHz.

The architecture of the EPM5032 allows it to emulate PAL or PLA structures and integrate functions that are not possible with these devices. The EPM5032 provides four times the integration density of traditional PAL and PLA functions, while supporting true system clock rates of 47 MHz (see Figure 1).

Figure 1. EPM5032 vs. PALs *EPM5032 provides more than four times the functional density of traditional PALs.*



3

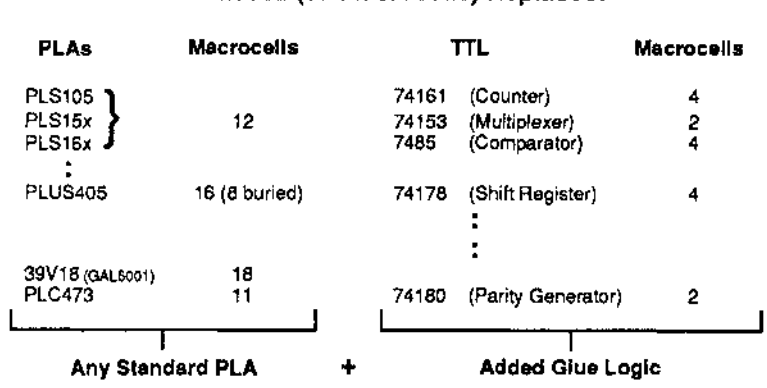
The basic EPM5032 architecture is an evolution of the AND-OR array structure that uses available logic far more efficiently than PAL or PLA approaches. Statistical analysis of hundreds of designs indicates that 3 product terms are adequate for 70% of applications. The fixed-product-term allocation of PALs usually wastes 5 of the 8 product terms. In contrast, the MAX approach uses a streamlined, 3-product-term macrocell that may be supplemented with as many as 64 additional expander product terms when required by a design. While PLA structures provide more flexible distribution of logic capability, the cost is a considerable loss in performance.

The EPM5032's method of using expander product terms increases the effective device density and permits higher integration than standard devices. This feature allows very large equations to be implemented in a single macrocell, leaving other macrocells free to perform additional functions. Macrocells can also share expanders to efficiently implement functions with common product terms (e.g., state machines).

Expander product terms can also be used to create additional latches and flip-flops. As many as 32 latches can be created without consuming macrocell registers. Two expanders can be cross-coupled to form an RS latch, and D registers and latches can be implemented with other expander configurations. Although these functions can be created in PAL devices, additional macrocells would be required.

Figure 2 shows that a single EPM5032 can be used to replace multiple PLAs plus a considerable amount of TTL glue logic in a variety of applications.

Figure 2. Using EPM5032 to Replace PLAs and TTL Glue Logic



Replacing 7400 TTL Devices

The EPM5032's register and I/O features enable it to emulate 7400-series TTL functions and replace not only programmable logic but also associated discrete TTL glue logic. Table 1 shows common TTL functions and their typical EPM5032 device utilization.

Table 1. EPM5032 TTL Function Integration

Function	Part	% Used
4-Bit Latch	7475	8%
4-Bit Mag.Comparator	7485	13%
8-Bit Shift Register	7491	19%
Dual 4:1 Mux	74153	4%
Quad 4:1 Mux	74157	5%
4-Bit Binary Counter	74161	10%
4-Bit Decade Counter	74162	10%
4-Bit Shift Register	74179	10%
9-Bit Parity Generator	74180	16%
Octal Latch	74373	16%

The MAX+PLUS TTL MacroFunction library simplifies entry of 7400-series functions by allowing familiar TTL symbols to be used when entering a design. Because MAX+PLUS automatically performs all design translation and optimization, a TTL function may be implemented simply by selecting the symbol. The MAX macrocell architecture provides true TTL emulation. MAX+PLUS can merge schematic designs with Boolean or state machine language descriptions when it compiles a design, enabling the designer to complete each section of a design in its most natural form.

Replacing PAL Devices

PAL circuits have a programmable-AND/fixed-OR structure. Each macrocell generally includes 7 (e.g., 16L8) or 8 (e.g., 16R8) AND gates, which are also called product terms. Advanced PALs, such as the 22V10, support a variable allocation of product terms for macrocells, with as many as 16 product terms per macrocell. However, sharing product terms between macrocells is prohibited, and product terms cannot be reallocated.

Figure 3 shows how the EPM5032 emulates a high product-term macrocell. The AND-OR logic expression of a PAL can be directly implemented with the EPM5032 macrocell. The expander product terms provide product-term expansion if an individual application needs more product terms. An EPM5032 macrocell can have as many as 64 product terms, which is far more than any existing PAL device. This architecture is ideal for designs with multiple high product-term outputs and common input signals. For example, state machines, arithmetic logic circuits, and complex combinatorial functions fit easily on MAX architecture EPLDs such as the EPM5032, but they are often impossible to efficiently design into PALs.

Figure 3. Emulating High Product-Term Count Macrocells

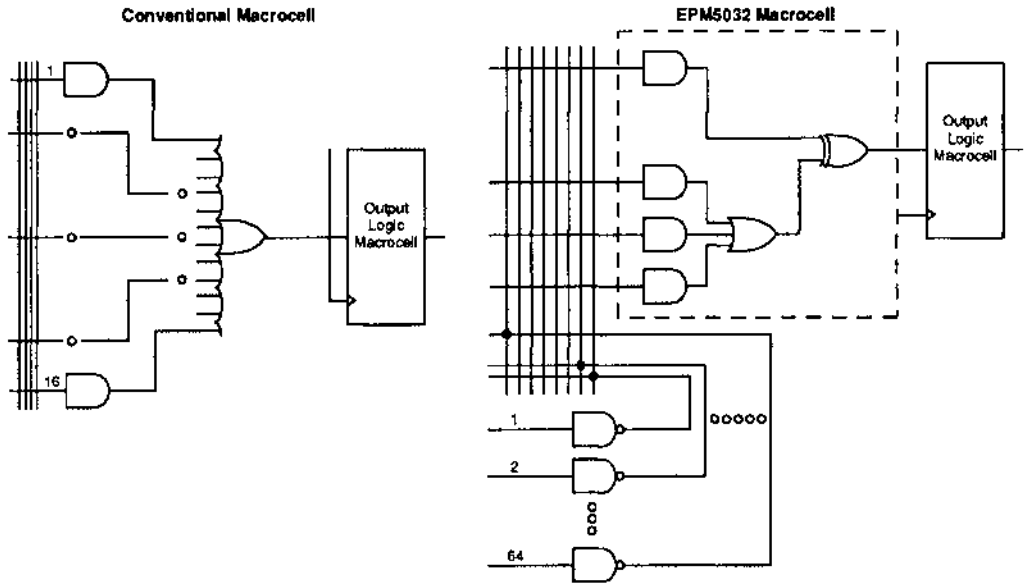
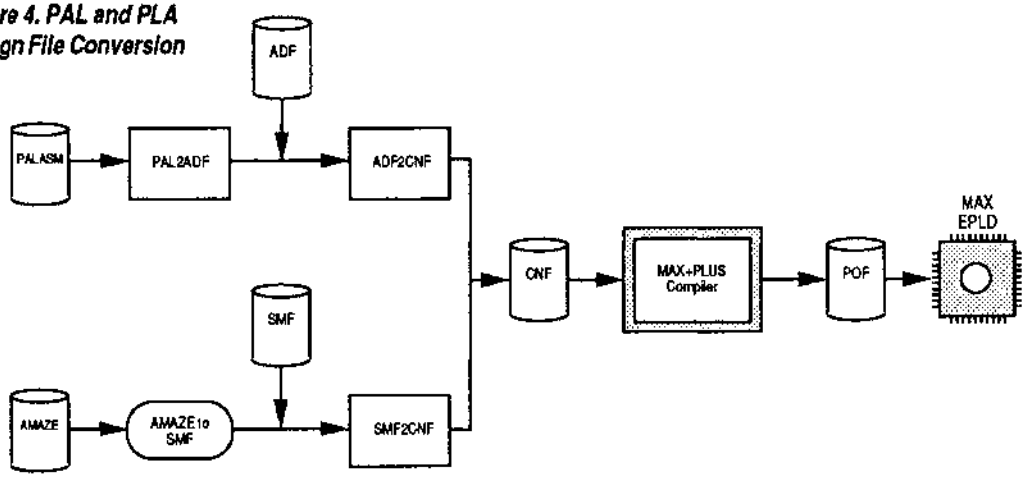


Figure 4 shows how existing PAL designs can be automatically incorporated into EPM5032 designs. Altera's PAL2ADF utility, available free from the Altera Electronic Bulletin Board (see *Electronic Bulletin Board Services* in Section 4 of this handbook), converts PALASM files into Altera Design Files (ADFs). The ADF2CNF conversion utility, included with MAX+PLUS, transforms ADFs into MAX+PLUS-compatible Compiler Netlist Files (CNFs). The Compiler can then process the CNFs into EPM5032 designs.

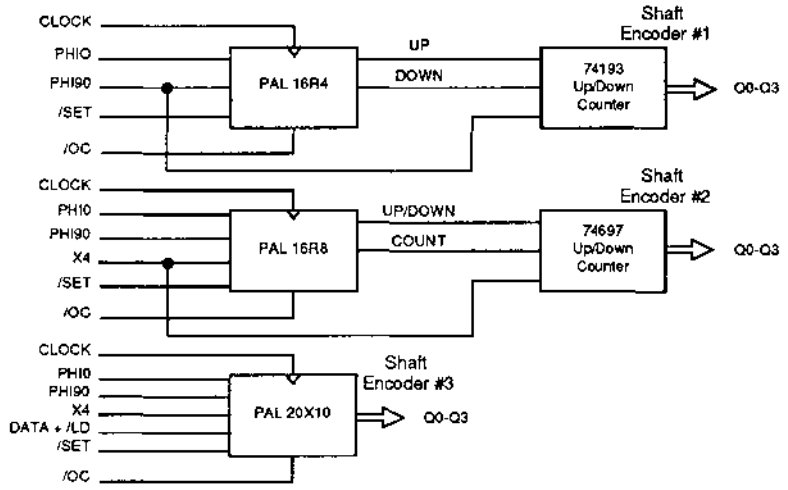
Figure 4. PAL and PLA Design File Conversion



PAL Design Example

The *Shaft Encoders* section of AMD's *PAL Device Handbook* shows a multi-PAL/TTL implementation of a digital shaft encoder. The circuit measures shaft rotation by counting and comparing pulses produced by a pair of LEDs, a pair of photo sensors, and a rotating disk. The circuits are divided into three major blocks. The first is a set of registers that accepts a pair of 90° out-of-phase digital pulse trains as inputs, one from each photosensor. The registers discriminate the phase difference between the two signals and feed them into the decoder, the next block of the circuit. The decoder converts the register outputs into an up-and-down signal to control the final block, the counter. Outputs from the counter determine the position and direction of shaft rotation. Figure 5 shows the circuitry required: three PALs (16R4, 16R8, and 20X10), and two discrete TTL devices (74193 and 74697). All five parts fit in a portion of a single EPM5032.

Figure 5. Shaft Encoder



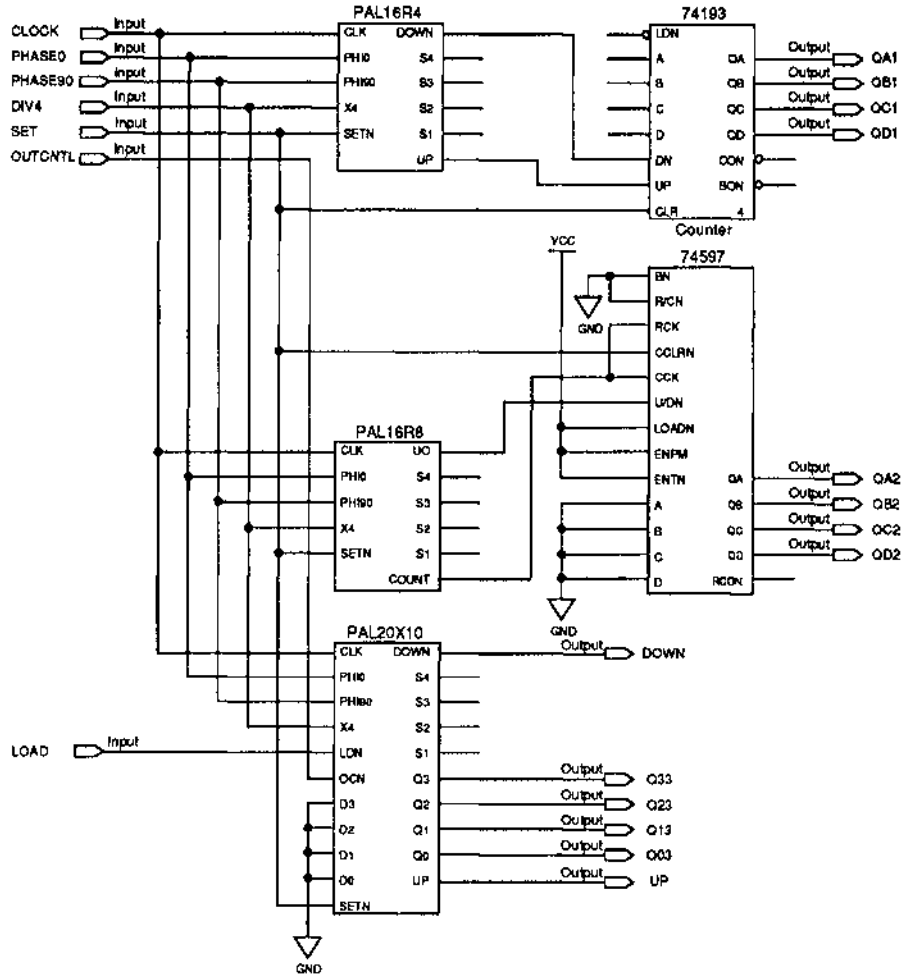
The 16R4 (see Figure 5) implements a set of four registers for phase discrimination, as well as decoding circuitry that produces UP and DOWN control signals for a 74193 counter. Converting these functions from the original PALASM source file to an equivalent Altera Design File (see Design File 1) was straightforward. Note that although the PAL design included a control signal (/OC) for disabling the tri-state buffer outputs, it was left out of the ADF because this signal is not needed in a completely integrated design. The entire translation process took just a few minutes.

The second PAL in the design (16R8) also implements four discrimination registers and decoding circuitry that outputs to a counter. However, the decoded outputs are configured into UP/DOWN and COUNT control signals, and the counter is a 74697 instead of a 74193. The PALASM source file for this design was translated into an ADF (see Design File 2), and the control signal (/OC) was left out. Translation took only a few minutes.

The third PAL (20X10) is more sophisticated than the other PALs shown. It implements register and decoding functions in addition to an UP/DOWN counter. Because this PAL includes its own counter and some signals that will actually appear on output pins in the integrated design, the Output Enable control was left intact in the ADF. The PALASM-to-ADF conversion in this case took more time than the previous two (about twenty minutes) because the original design file was longer. See Design File 3.

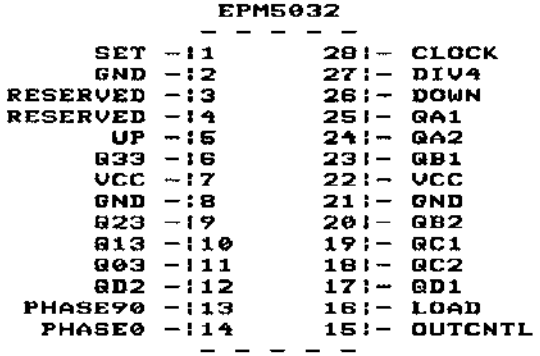
After all 3 PAL designs had been reproduced in ADF format, they were converted to CNF files by the ADF2CNF utility for use in MAX+PLUS. In addition, the 74697 counter was entered as a schematic with the MAX+PLUS Graphic Editor and compiled to produce a CNF. Since the 74193 is in the MAX+PLUS TTL MacroFunction library, its CNF file was already available. As shown in Figure 6, automatically-created symbols for each of the 5

Figure 6. MAX+PLUS Shaft Encoder Schematic



devices were entered with the MAX+PLUS Graphic Editor and then connected to complete the design. This schematic was compiled and easily fit into a single EPM5032. Integration of these 3 PALs plus the 2 discrete counters required 29 macrocells and 28 expanders. Less than 90% of the EPM5032 was utilized, so there would have been room to implement even more logic. Figure 7 shows the pinout MAX+PLUS produced as part of the Report File.

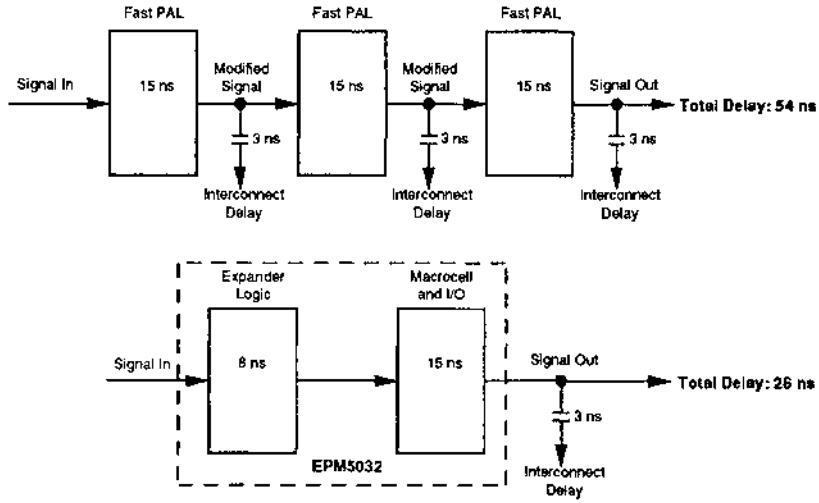
Figure 7. Integrated Shaft Encoder



As a result of its ability to replace multiple devices, the EPM5032 provides increased performance. The "single-chip solution" saves board space over standard devices and eliminates chip-to-chip delays incurred by designs that require multiple PALs or PLAs. (See Figure 8.) For example, a PAL design with 3 logic levels has an overall delay of 54 ns (three 15-ns propagation delays and three 3-ns interconnect delays). The same design can be implemented in the EPM5032 with one level of expander logic and one level of macrocell logic for a total delay of 26 ns (including the 3-ns interconnect delay to the next stage). This single device implementation provides faster speeds and better overall system performance.

3

Figure 8. Replacing High-Speed PALs with the EPM5032-2

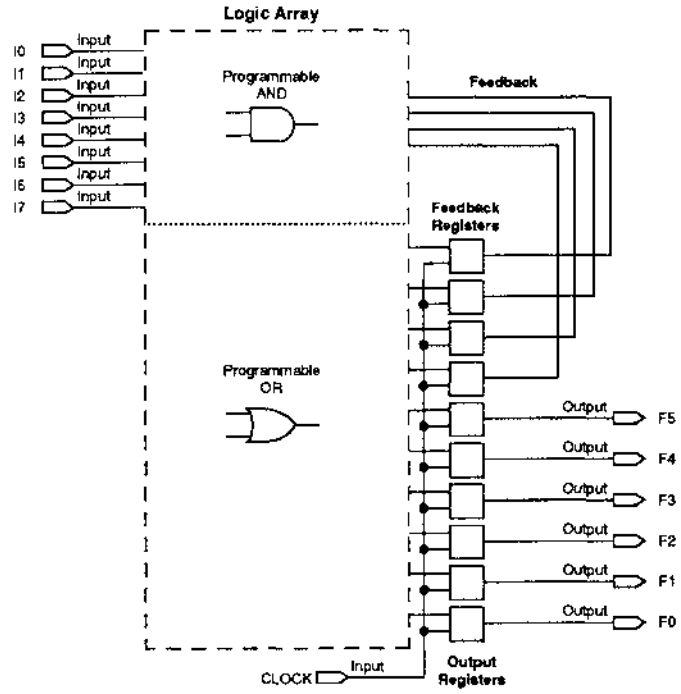


Replacing PLAs

In addition to PAL and random TTL replacement, the EPM5032 can also implement PLA architectures. Each PLA provides an additional programmable array, which allows variable product-term distribution.

Figure 9 shows a typical PLA structure that consists of a programmable AND array (containing 32 to 48 AND gates) that feeds a programmable OR

Figure 9. Traditional PLA Architecture



array. Each of the AND gates is fed by 8 to 12 input pins and feedback signals, plus their complements. The OR array allows the logical ORing of any of the AND terms in the array. Typically, 8 to 12 programmed OR terms feed either feedback combinatorial buffers, feedback registers, or output pins.

The PLA's programmable-AND/programmable-OR structure is emulated with the expander product-term array, as shown in Figure 10. As in high product-term PAL applications, the unallocated product terms feed an inverted AND gate in the macrocell to create an AND/OR-equivalent NAND/NAND structure.

Figure 10. Emulating the PLA's AND/OR Structure

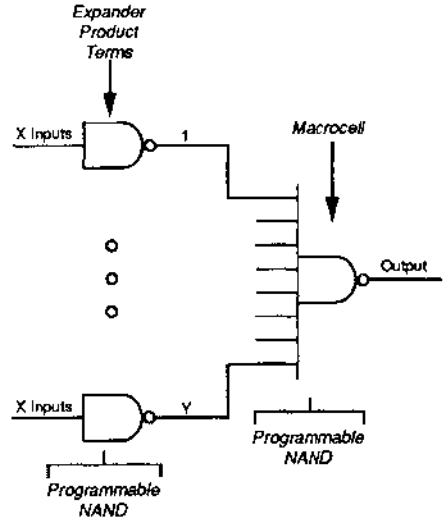


Figure 4 shows how existing PLAs can be incorporated into EPM5032 designs. First, the PLA source files from Signetics, which are usually written in the AMAZE language, must be converted to the Altera State Machine File (SMF) format. AMAZE and SMF formats are similar, so the source files may be quickly converted with a standard text editor. Figure 11 contains AMAZE and SMF representations of a state diagram depicting a simple beverage dispenser. Notice the similarities in state and transition definition. After the AMAZE source file is transformed into an SMF, the SMF2CNF conversion utility (contained in MAX+PLUS) will translate the SMF into a MAX+PLUS-compatible CNF. The MAX+PLUS Compiler may then process the CNF into EPM5032 designs.

3

Figure 11. SMF and AMAZE

(A) ALTERA SMF

```
INPUTS: COINDROP, CUPFULL
OUTPUTS: DROPCUP, POURDRNK
STATES: [DROPCUP POURDRNK]
S1          [0          0]
S2          [1          0]
S3          [0          1]
S1: IF COINDROP THEN S2
S2: S3
S3: IF CUPFULL THEN S1
```

(B) SIGNETICS AMAZE

```
STATE VECTORS
S1 = 00b;
S2 = 10b;
S3 = 01b;
INPUT VECTORS
[COINDROP, CUPFULL]
OUTPUT VECTORS
[DROPCUP POURDRNK]
OUT1 = 00b;
OUT2 = 10b;
OUT3 = 01b;
TRANSITIONS
WHILE [S1]
            IF [/COINDROP] THEN [S1] WITH [OUT1]
            IF [COINDROP] THEN [S2] WITH [OUT2]
WHILE [S2] THEN [S3] WITH [OUT3]
```

PLA Design Example

A serial communications interface with a custom protocol is described in *Custom Communication Protocol—PLS105A, 157, 167, 168A* of the Signetics ***Sequencer Solutions*** manual. The design shown in Figure 12 consists of three blocks, each of which is implemented in a Signetics PLA. The first block, the Preamble Sequence Detector, is a state machine that recognizes a bit pattern signalling the beginning of a valid data block. The second block, the Cyclic Redundancy Check (CRC) Controller state machine, coordinates the loading of parallel-to-serial and serial-to-parallel shift registers and provides status of data and CRC transmissions. The final block is a 5-bit CRC Generator that detects errors. These three blocks were integrated with Altera's MAX+PLUS Development System, and the resulting design was fit into a single EPM5032.

Preamble Sequence Detector

The first module was implemented in the Signetics PLS167A. The Signetics AMAZE source file was transcribed into an SMF, and the design fit into less than one third of an EPM5032.

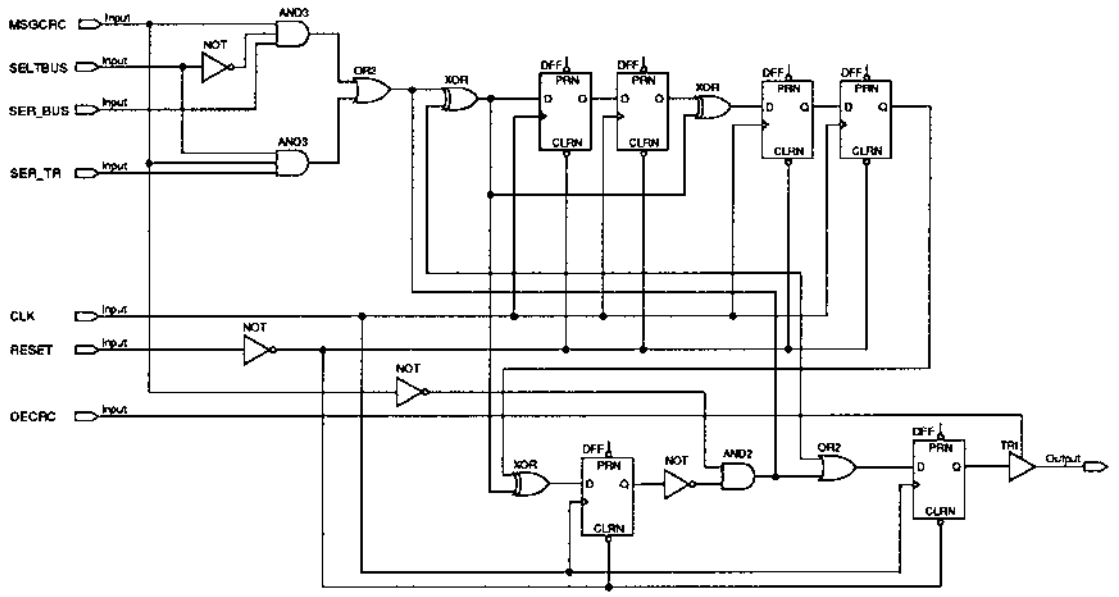
CRC Controller

A Signetics source file was again converted into an SMF to implement the CRC Controller. The Signetics implementation required 100% register utilization of a PLS105A, while Altera's CRC Controller implementation left plenty of the EPM5032 unused.

CRC Generator

The schematic shown in Figure 12 is the CRC Generator. It was submitted directly to MAX+PLUS for processing. This module required 100% register utilization of a PLS157, but consumed less than 20% of the EPM5032.

Figure 12. CRC Generator



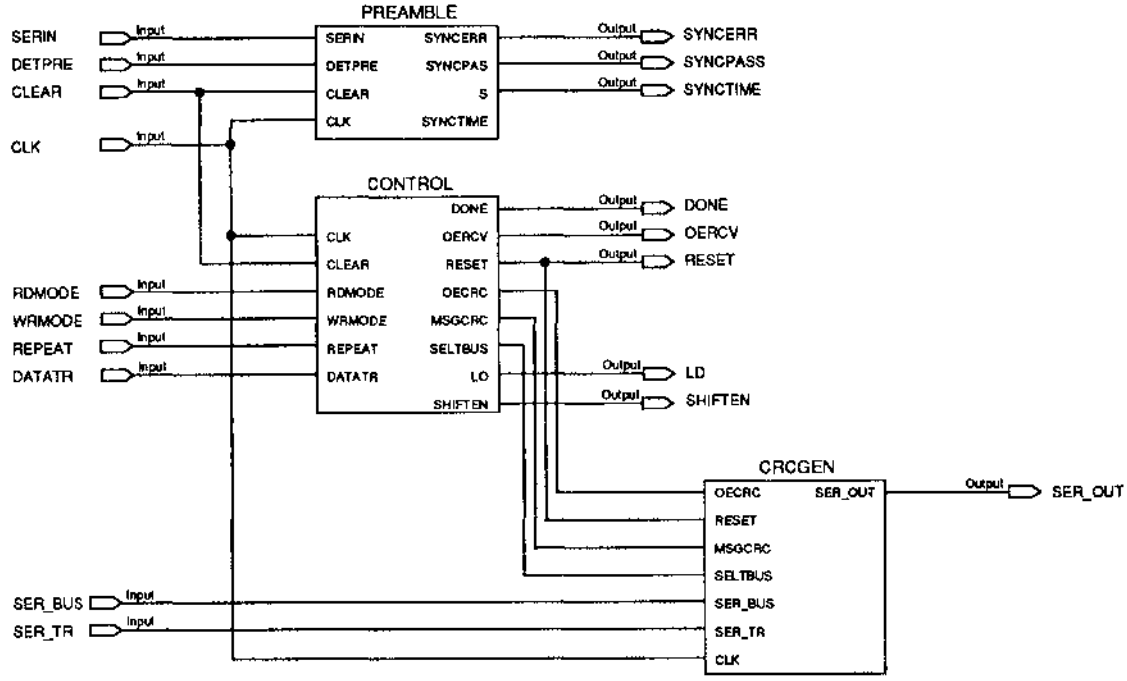
All three modules were integrated to fit into a single EPM5032. Integration was simplified by MAX+PLUS. Each of the three source files was compiled separately with the MAX+PLUS Compiler, and the automatically-created symbols representing each module were entered with the MAX+PLUS Graphic Editor. Appropriate interconnections were made between the symbols to complete the design, and the final schematic was then submitted to the design processor (see Figure 13). The result was integration of a PLS105A, PLS157, and PLS167A into one EPM5032 (see Figure 14).

Conclusion

The EPM5032 offers better integration than PALs or PLAs in addition to a flexible architecture supporting a broad range of applications. A single EPM5032 typically replaces three to four conventional PLDs, and can provide extensive TTL integration as well. As shown in the design examples, one EPM5032 may be used in place of three PLAs or three PALs and two TTL devices. Designs may be easily converted from PALs or PLAs to the EPM5032. The EPM5032 also has a flexible architecture that supports a broad range of applications, and offers flexible logic utilization, advanced registers, better control of I/O pins, and system performance benefits.

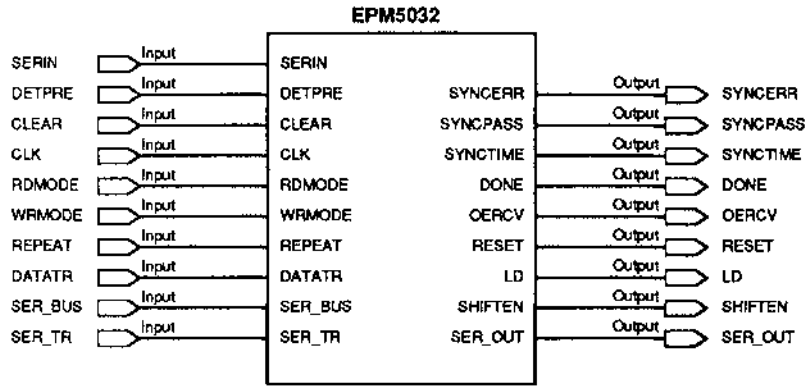
3

Figure 13. Serial Communications Interface



The design files used to generate the PAL Shaft Encoder and the PLA Serial Communications Interface are available and can be downloaded with the Altera Electronic Bulletin Board Service. Details on access and use of the bulletin board are available in Section 4 of this handbook. Altera

Figure 14. Integrated Serial Communications Interface



Applications Engineers may also be reached at (408) 984-2805 ext. 102. The following files are available:

PAL Design Files: **SHFTENC1.ADF, SHFTENC2.ADF, SHFTENC3.ADF**

PLA Design Files: **PREAMBLE.SMF, CONTROL.SMF, CRCGEN.GDF**

Design File 1

```
Altera Corporation
10-4-1989 1.00 A EPM5032
Shaft Encoder 1

OPTIONS: SECURITY = OFF
PART: AUTO
INPUTS: CLK, PHI0, PHI90, X4, SETN
OUTPUTS: DOWN, S4, S3, S2, S1, UP

NETWORK:
CLK = INP(CLK)
PHI0 = INP(PHI0)
PHI90 = INP(PHI90)
X4 = INP(X4)
SETN = INP(SETN)
DOWN = CONF(DOWNc,VCC)
S4,S4 = RORF(S4d, CLK, , ,VCC)
S3,S3 = RORF(S3d, CLK, , ,VCC)
S2,S2 = RORF(S2d, CLK, , ,VCC)
S1,S1 = RORF(S1d, CLK, , ,VCC)
UP = CONF(UPc,VCC)

EQUATIONS:
SET = /SETN;
/S1d = /PHI0 + SET;
/S2d = /S1 + SET;
/S3d = /PHI90 + SET;
/S4d = /S3 + SET;
/DOWNc = S1 * S2 * S3 * /S4 * PHI0 * PHI90
         + /S1 * /S2 * /S3 * S4 * /PHI0 * /PHI90
         +S1 * /S2 * /S3 * /S4 * PHI0 * /PHI90
         + /S1 * S2 * S3 * S4 * /PHI0 * PHI90;
/UPc = /S1 * /S2 * S3 * /S4 * /PHI0 * PHI90
         + S1 * S2 * /S3 * S4 * PHI0 * /PHI90
         +S1 * /S2 * S3 * S4 * PHI0 * PHI90
         + /S1 * S2 * /S3 * /S4 * /PHI0 * /PHI90;
END$
```

Design File 2

```
Altera Corporation
10-4-1989 1.00 A EPM5032
Shaft Encoder 2

OPTIONS: SECURITY = OFF
PART: AUTO
INPUTS: CLK, PHI0, PHI90, X4, SETN, OCN
OUTPUTS: UD, S4, S3, S2, S1, COUNT

NETWORK:
CLK = INP(CLK)
PHI0 = INP(PHI0)
PHI90 = INP(PHI90)
X4 = INP(X4)
SETN = INP(SETN)
OCN = INP(OCN)
UD, UD = RORF(UDd, CLK, , ,OC)
S4, S4 = RORF(S4d, CLK, , ,OC)
S3, S3 = RORF(S3d, CLK, , ,OC)
S2, S2 = RORF(S2d, CLK, , ,OC)
S1, S1 = RORF(S1d, CLK, , ,OC)
COUNT, COUNT = RORF(COUNTd, CLK, , ,OC)

EQUATIONS:
OC = /OCN;
SET = /SETN;
/S1d = /PHI0 + SET;
/S3d = /PHI90 + SET;
/S2d = S1 + SET;
/S4d = S3 + SET;
/COUNTd = S1* S2*/S3 * S4
          + /S1*/S2*S3*/S4 + /S1*S2*/S3* /S4* X4
          + S1*/S2* S3* S4*X4 + S1*/S2*S3*/S4
          + /S1* /S2*/S3*S4 + /S1* S2*S3*S4*X4
          + S1* /S2*/S3*/S4*X4;
/UDd = /S1*S2* /S3*S4
          + /S1* S2*S3*S4+ /S1*S2*S3*/S4
          + S1* S2*S3* /S4+ S1* /S2*S3*/S4
          + S1* /S2* /S3 * /S4+ S1*/S2*/S3*S4
          + /S1* /S2*/S3*S4;

END$
```

Design File 3

```
Altera Corporation
10-4-1989 1.00 A
Shaft Encoder 3

OPTIONS: SECURITY = OFF
PART: AUTO
INPUTS: CLK, PHI0, PHI90, X4, LDN, OCN, D3, D2, D1, D0, SETN
OUTPUTS: DOWN, S4, S3, S2, S1, Q3, Q2, Q1, Q0, UP

NETWORK:
CLK = INP(CLK)                          PHI0 = INP(PHI0)
PHI90 = INP(PHI90)                      X4 = INP(X4)
LDN = INP(LDN)                          D3 = INP(D3)
D2 = INP(D2)                            D1 = INP(D1)
D0 = INP(D0)                            SETN = INP(SETN)
OCN = INP(OCN)                          DOWN, DOWN = RORF(DOWNd, CLK, , ,OC)
S4, S4 = RORF(S4d, CLK, , ,OC)          S3, S3 = RORF(S3d, CLK, , ,OC)
S2, S2 = RORF(S2d, CLK, , ,OC)          S1, S1 = RORF(S1d, CLK, , ,OC)
Q3, Q3 = RORF(Q3d, CLK, , ,OC)          Q2, Q2 = RORF(Q2d, CLK, , ,OC)
Q1, Q1 = RORF(Q1d, CLK, , ,OC)          Q0, Q0 = RORF(Q0d, CLK, , ,OC)
UP, UP = RORF(UPd, CLK, , ,OC)

EQUATIONS:
OC = /OCN;
SET = /SETN;
LD = /LDN;
/S1d = /PHI0    + SET;
/S3d = /PHI90   + SET;
/S2d = /S1   + SET;
/S4d = /S3   + SET;
A = S1 * S2 * S3 * /S4 *PHI0 * PHI90 * X4  + /S1 * /S2 * /S3 * S4 * /PHI0 * /PHI90 * X4;
B = S1 * /S2 * /S3 * /S4 * PHI0 * /PHI90  + /S1 * S2 * S3 * S4 * /PHI0 * PHI90;
C = /S1 * /S2 * S3 * /S4 * /PHI0 * PHI90  + S1 * S2 * /S3 * S4 * PHI0 * /PHI90;
D = S1 * /S2 * S3 * S4 * PHI0 * PHI90 * X4  + /S1 * S2 *
/S3 * /S4 * /PHI0 * /PHI90 * X4;
E = /SET * LD * /D0  + /SET * /LD * /Q0;
F = /SET * /LD * UP * /DOWN  + /SET * /LD * /UP * DOWN;
G = /SET * LD * /D1  + /SET * /LD * /Q1;
H = /SET * /LD * UP * /DOWN * /Q0  + /SET * /LD * /UP * DOWN * Q0;
I = /SET * LD * /D2  + /SET * /LD * /Q2;
J = /SET * /LD * UP * /DOWN * /Q0 * /Q1  + /SET * /LD * /UP * DOWN * Q0 * Q1;
K = /SET * LD * /D3  + /SET * /LD * /Q3;
L = /SET * /LD * UP * /DOWN * /Q0 * /Q1 * /Q2  + /SET * /LD * /UP * DOWN * Q0 * Q1 * Q2;
/DOWNd = (A+B) * /(A*B);                /UPd = (C+D) * /(C*D);
/Q0d = (E+F) * /(E*F);                  /Q1d = (G+H) * /(G*H);
/Q2d = (I+J) * /(I*J);                  /Q3d = (K+L) * /(K*L);

END0
```

References

Advanced Micro Devices, Inc. *PAL Device Handbook* (1988).

Signetics Corporation. *Sequencer Solutions* (1988).

Integrating an Intelligent I/O Subsystem with a Single EPM5128

January 1990 **Application Note 17**

Introduction

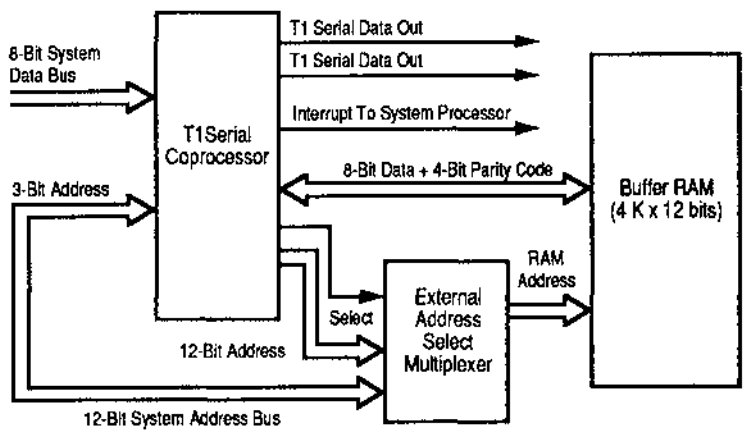
When higher system performance is necessary, many designers first consider a faster microprocessor or a new microprocessor architecture. In many cases, however, sufficient speed may be achieved through an intelligent I/O subsystem that has been optimized for high-performance of a particular task. Transferring I/O processing to intelligent subsystems also allows the system processor to dedicate more processing power to primary system functions.

This Application Note describes how the custom logic requirements of such a subsystem may be integrated into a single Altera user-configurable EPM5128 MAX EPLD. The design of an intelligent serial transmitter, the T1 serial coprocessor, is described under the heading "T1 Serial Coprocessor Example" later in this Application Note. This design replaces over 75 standard TTL packages.

Figure 1 shows a subsystem using the T1 serial coprocessor implemented in an EPM5128. The serial coprocessor consists of two T1 serial transmitters, control and address generation logic for a buffer RAM, as well as the I/O subsystem control logic. The system processor writes data for serial transmission through the T1 serial coprocessor into the buffer RAM. When the system processor finishes the transfer of the data into buffer RAM, it

Figure 1. Block Diagram of a Subsystem Using the T1 Serial Coprocessor

Altera's EPM5128 can integrate the logic requirements of high performance I/O subsystems such as the intelligent T1 serial coprocessor.



3

signals the serial coprocessor that data is ready for transmission. The serial coprocessor then transfers the data to the transmitters for serialization. The buffer RAM control logic also features error detection and correction. Data may be sent over either T1 serial channel with individual variations in protocol. Once all of the data in buffer RAM has been transferred, the processor is interrupted, and the cycle may repeat.

The T1 serial coprocessor is a useful example of an intelligent I/O subsystem design because it contains the types of logic, such as decode and control state machines, that are common to most digital designs. Its error detection and correction, RAM control and address generation, and parallel-to-serial conversion features also have broad use in system design. Furthermore, digital data communication of both voice and data plays an increasingly important role in modern systems.

Although this design example shows a T1 serial transmission application operating at 1.544 MHz, the same concept may be applied to create high-performance subsystems for applications such as local area networks or disk controllers. The EPM5128 supports serial data rates of up to 40 megabits per second. The EPM5128 is user-configurable to allow logic customization to fit the application, rather than compromising the application by fitting it into available standard components.

This Application Note includes complete details on design methodology, CAE support, and a detailed discussion of the design implementation. The design is developed with MAX+PLUS, an advanced PC-based CAE system that allows designs to be entered, compiled, simulated, and programmed at the designer's desk. This Application Note assumes familiarity with the basic concepts of programmable logic.

EPM5128 Overview

The EPM5128 is a member of Altera's MAX family of high-density EPLDs. The chip consists of 128 flexible macrocells that are grouped into 8 Logic Array Blocks (LABs). Each LAB contains 16 macrocells and 32 expander product terms (expanders). Expanders are freely allocatable product terms that may be used and shared by any macrocell function within an LAB. These functions may be macrocell register control functions such as asynchronous Preset, asynchronous Clear, and the register Clock signal. Very complex equations may thus be implemented for both registered and combinatorial applications.

A Programmable Interconnect Array (PIA) routes signals between the LABs. All 52 decoupled I/O pins and all 128 macrocell outputs enter the PIA. Each LAB may then receive any of the signals it requires from the PIA. The PIA has sufficient routing resources to accommodate the most complex designs. A fixed interconnect delay across the PIA also eliminates skew, allowing accurate timing performance prediction before the design is completed. Refer to *EPM5064 to EPM5192: MAX EPLDs with Multiple LABs* in Section 2 of this handbook for more information about EPM5128 architecture and performance.

Designing with the EPM5128

Designs for the EPM5128 and for the entire MAX family of EPLDs are entered, processed, simulated, and programmed with MAX+PLUS software. MAX+PLUS provides a complete CAE design environment featuring hierarchical design entry, logic synthesis, automatic error location, and timing simulation. For more information about MAX+PLUS refer to the *PLDS-MAX / PLS-MAX: MAX+PLUS Programmable Logic Development System* in Section 2 of this handbook and the MAX+PLUS manuals.

The following design steps are discussed:

1. Design Entry
2. Design Processing
3. Timing Prediction and Simulation

TI serial coprocessor components are used to illustrate these design processes. The design of the serial coprocessor is discussed in detail under the heading "TI Serial Coprocessor Example" later in this section.

Design Entry

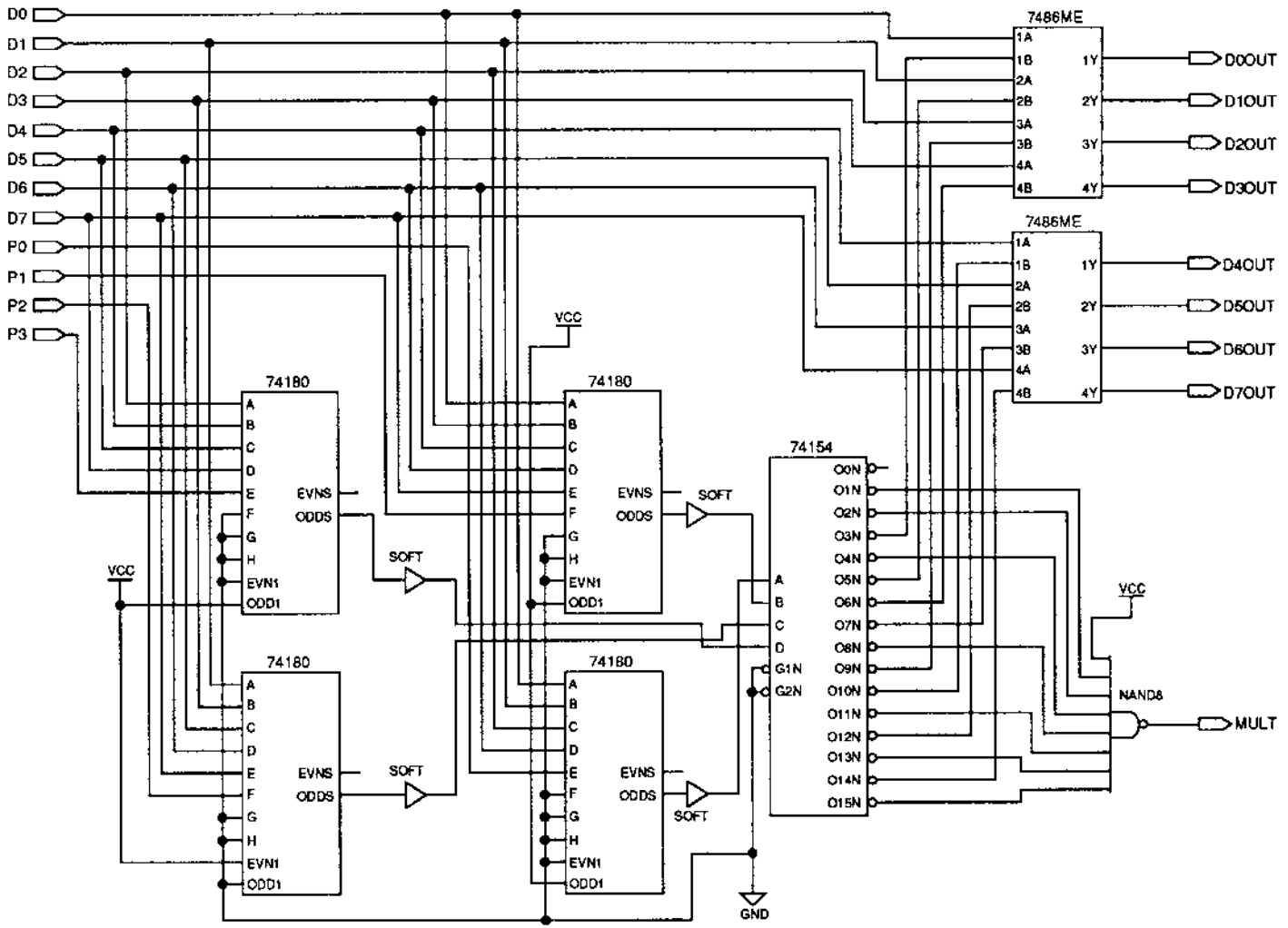
MAX+PLUS provides two methods of entering a design for the EPM5128 or any other MAX EPLD: hierarchical schematic capture and a high-level entry language that allows a mixture of state machines, Boolean equations, and truth tables. The Altera Hardware Description Language (AHDL), which supports advanced features such as automatic state bit assignments for state machine applications, replaces the Altera ADF and SMF entry language formats for MAX EPLD designs. However, for migration of EP-series designs into MAX parts the ADF2CNF and SMF2CNF conversion utilities are available. Detailed information about Altera ADF/SMF language format may be found in the ***MAX+PLUS Entry Language*** manual (MAX+PLUS versions 1.0 and 1.5). Detailed information about Altera Hardware Description Language (AHDL) may be found in the ***MAX+PLUS AHDL*** manual (MAX+PLUS version 2.0).

Schematic capture and language entry may be used either independently or together, with each function entered in the most appropriate form. The serial coprocessor design example described later in this Application Note uses primarily hierarchical schematic capture for design entry, but several control functions are implemented as Altera State Machine Files (SMFs).

Excerpts from the subsystem design example help illustrate the process of designing with the EPM5128. The logical error detection and correction (EDAC) function, shown in Figure 2, is used as the first example. (EDAC is the portion of the TI serial coprocessor that performs error correction and detection.)

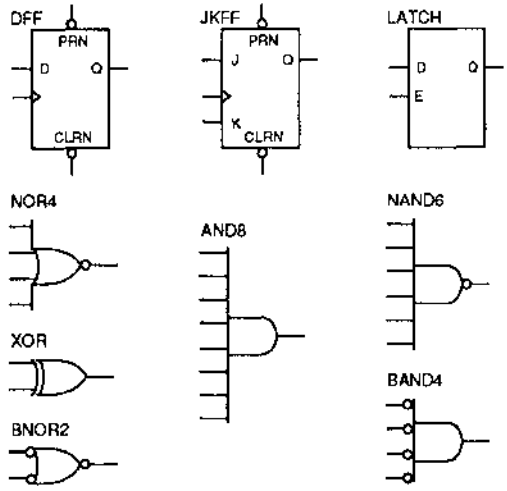
Figure 2. Error Detection and Correction (EDAC) Function

The custom logic function EDAC is implemented with primitive gates, TTL Macrofunctions, and the custom logic function 7486ME.



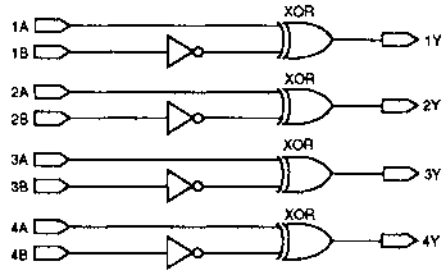
Symbols from multiple libraries are used to create the EDAC schematic. Basic logic functions, such as the NOT and NAND gates, are provided in the MAX+PLUS Primitive Library. Figure 3 shows some typical functions available from this library. More complex functions such as the 74180 parity generator/checker and the 74154 decoder used in EDAC are provided in the MAX+PLUS TTL MacroFunction Library, which contains over 200 common 7400-series TTL and bus functions. The EPM5128 has the asynchronous Preset and Clear functions for each register required for true emulation of TTL functions.

Figure 3. Examples of Altera Primitive Library Logic Functions



The user may also create customized functions in MAX+PLUS with any of the design entry methods. For example, the function 7486ME used in EDAC is a quad, two-input XOR function that is based on the standard TTL 7486 macrofunction, but with one input of each XOR gate inverted. See Figure 4.

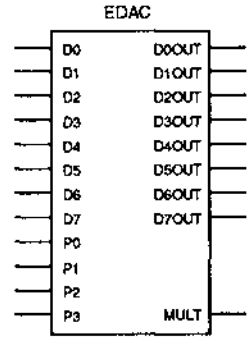
Figure 4. Custom Logic Function 7486ME



Symbols for custom functions, such as 7486ME, are automatically generated by MAX+PLUS when the logic schematic is saved. As a result, previously created custom designs may be quickly integrated into any design. Figure 5 shows the automatically generated symbol for the EDAC function, which is integrated into the top level of the serial coprocessor design.

Figure 5. EDAC Symbol

The symbol for EDAC is automatically generated by MAX+PLUS and may easily be integrated into other designs.



Some sections of designs, especially control functions, are represented most conveniently as state machines. For example, the state machine T1ST, which is integrated into both of the T1 serial transmitters, controls the serialization of data in the T1 transmitters. Figure 6 shows the T1ST State Machine File.

State machines may be entered with Altera's State Machine Language, generating a State Machine File (SMF). Or they may be entered with AHDL, generating a Text Design File (TDF). The MAX+PLUS Text Editor or any standard text editor may be used to enter state machines in MAX+PLUS. SMF and TDF syntaxes require input and output declarations, state declarations, and state transition information. Section 5 of the Altera *Applications Handbook* discusses these state machine design concepts further.

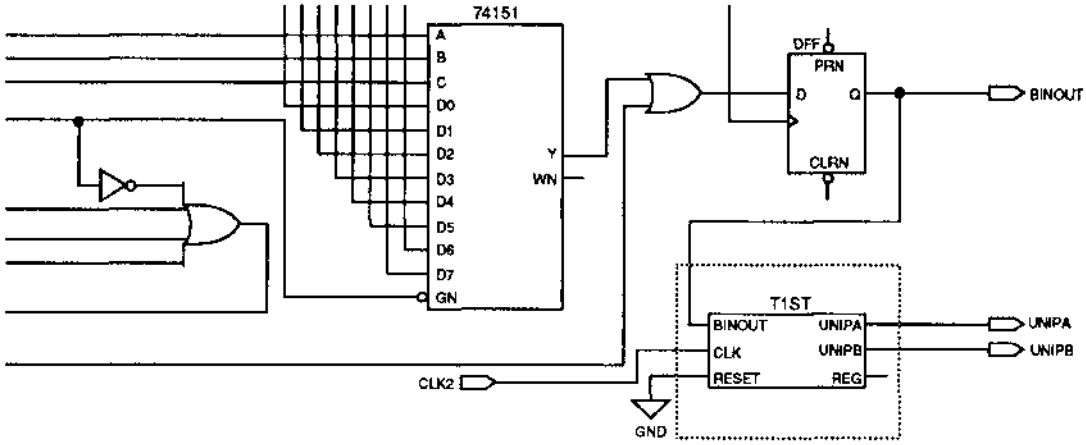
Figure 6. T1ST State Machine File *State Machines such as T1ST are created with Altera's State Machine Language.*

```
ALTERA
8/25/88
T1 STATE MACHINE

PART: EPM5032

INPUTS: BINOUT, CLK, RESET
OUTPUTS: UNIPA, UNIPB, REG

MACHINE: T1UNIP

CLOCK: CLK

CLEAR: RESET

STATES: [REG  UNIPB  UNIPA]
S0      [ 0     0      0  ]
S1      [ 0     0      1  ]
S2      [ 1     0      0  ]
S3      [ 1     1      0  ]
S4      [ 1     0      1  ]
S5      [ 1     1      1  ]
S6      [ 0     1      0  ]
S7      [ 0     1      1  ]

S0:  IF /BINOUT THEN S1
     S0
S1:  IF /BINOUT THEN S2
     S1
S2:  IF /BINOUT THEN S3
     S2
S3:  IF /BINOUT THEN S0
     S1
S4:  IF /BINOUT THEN S2
     S3
S5:  IF /BINOUT THEN S2
     S3
S6:  IF /BINOUT THEN S0
     S1
S7:  IF /BINOUT THEN S0
     S1
END
```

An SMF must be processed with the SMF2CNF converter provided with MAX+PLUS, which performs a syntax check and creates a symbol for the state machine. A TDF, however, may be submitted directly to the MAX+PLUS Compiler. The state machine symbol may be used to integrate the state machine into a schematic. Figure 7 shows state machine T1ST integrated with other TTL macrofunctions into a portion of the T1 transmitter schematic.

Figure 7. T1ST Integrated with Other TTL Macrofunctions

Automatic symbol generation allows state machines such as T1ST to be quickly integrated into schematics.



Design Processing

After a design for the EPM5128 has been entered, it is submitted to the MAX+PLUS Compiler for processing. Compilation typically takes five to fifteen minutes on an 80386-based PC. The Compiler extracts a netlist from the various design files, checks for conformity with design rules, and then performs logic synthesis, minimizing and optimizing the design for the EPM5128 architecture. If any errors are detected during compilation, the Error Report window shows the exact schematic location of the error, and highlights the incorrect node.

Once a design has been synthesized, it is mapped onto the EPM5128 by the Fitter. The Fitter uses a set of heuristic algorithms that frees the designer from routing and interconnect procedures. The Fitter also generates a Report File, providing device utilization information for design subsections as well as complete designs. Figure 8 shows a portion of the Report File with device utilization information for EDAC.

Figure 8. Report File for EDAC

The Report File is automatically generated during design compilation and contains information on device utilization

```
                                                    C:\MAXWORK\MAXAB\EDAC.RPT

**RESOURCE USAGE**

                                                                        External
Logic Array Block     Macrocells     I/O Pins      Expanders          Interconnect

A:     MC1 - MC16      8/16 (  8%)   8/ 8(  8%)    8/32(  8%)         8/24(  8%)
B:    MC17 - MC32      8/16 (  8%)   8/ 5(  8%)    8/32(  8%)         8/24(  8%)
C:    MC33 - MC48      8/16 (  8%)   8/ 5(  8%)    8/32(  8%)         8/24(  8%)
D:    MC49 - MC64      3/16 ( 18%)   1/ 8( 12%)    4/32( 12%)         1/24(  4%)
E:    MC65 - MC80      8/16 (  8%)   4/ 8( 50%)    8/32(  8%)         8/24(  8%)
F:    MC81 - MC96      8/16 (  8%)   8/ 5(  8%)    8/32(  8%)         8/24(  8%)
G:   MC97 - MC112      8/16 (  8%)   8/ 5(  8%)    8/32(  8%)         8/24(  8%)
H: MC113 - MC128      16/16 (100%)   8/ 8(100%)   12/32( 37%)         5/24( 20%)

Total Macrocells used:   19/128( 14%)
Total output pins used:   9/48( 18%)
Total input pins used:   12/51( 23%)
```

Timing Prediction and Simulation

After a design has been successfully compiled, AC analysis may be performed through delay prediction and timing simulation. Delay prediction, which takes place in the schematic environment, is used to quickly check critical timing paths from within the hierarchical schematic environment. It then provides the worst-case delay for a specified path. In EDAC, for example, if D1 is specified as the source node and D1OUT as the destination node, the worst-case delay path is through the 74180 parity generator/checkers.

Timing simulation is performed by the MAX+PLUS Simulator. The Simulator is an event-driven, interactive timing simulator that verifies performance and functionality of designs. It has resolution of 1/10 nanosecond, and may be configured to monitor the design for setup and hold time violations. Stimuli for the design may be defined in a vector format or created in an interactive graphical waveform environment. Simulation output can be displayed on screen or printed to hardcopy, in either table or waveform format. Figure 9 shows the table file for EDAC, containing input and the output vectors resulting from a typical simulation.

Figure 9. Table File for EDAC — *Simulation results may be viewed in table or waveform formats.*

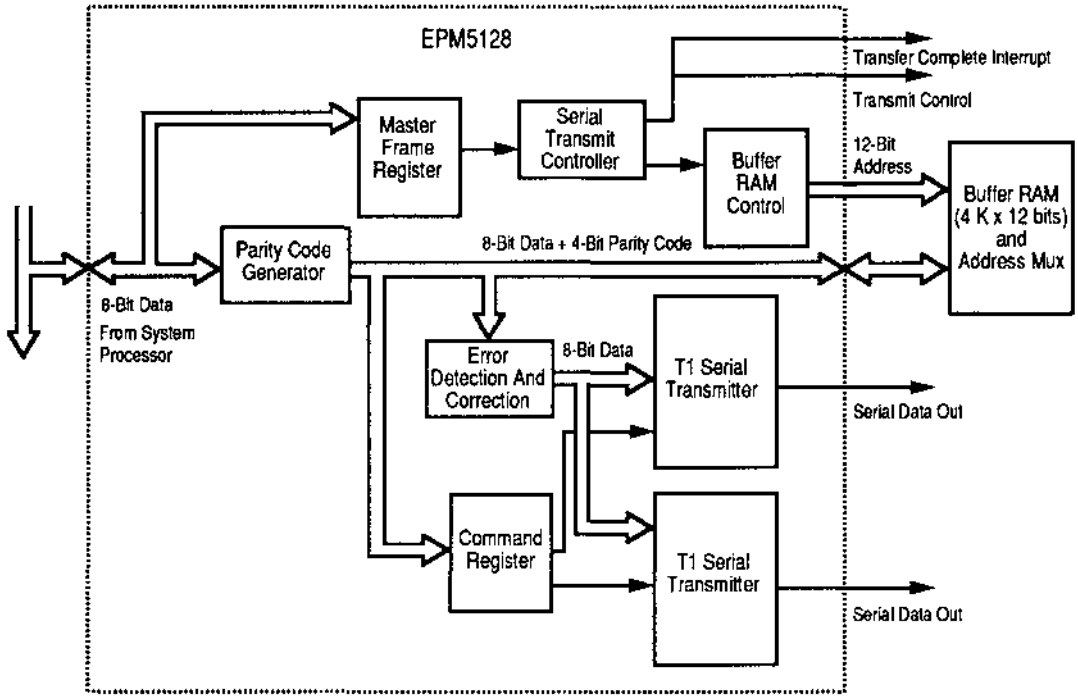
```
NODES  D7 D6 D5 D4 D3 D2 D1 D0 P3 P2 P1 P0 D7OUT
D6OUT D5OUS D4OUT D3OUT D2OUT D1OUT D0OUT MULT :
UNIT ns;
PATTERN
     0.0> 1 0 1 0 1 0 1 0 1 0 1 0 X X X X X X X X
    23.0> 1 0 1 0 1 0 1 0 1 0 1 0 X X X X X X X 1
    87.0> 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 1
  2500.0> 1 1 1 1 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 1
  2529.9> 1 1 1 1 1 0 1 0 1 0 1 0 1 1 1 0 1 0 1 1
  2587.0> 1 1 1 1 1 0 1 0 1 0 1 0 1 1 1 0 1 0 0 1
  2603.0> 1 1 1 1 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 1
  5000.0> 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 1
  5029.9> 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 1 1 0 1 1
  5086.9> 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 1 1 0 1 1
  5087.0> 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 1
  7500.0> 1 0 1 0 1 1 1 0 1 0 1 0 1 0 1 0 1 0 1 1
  7529.9> 1 0 1 0 1 1 1 0 1 0 1 0 1 1 1 1 1 0 1 1
  7587.0> 1 0 1 0 1 1 1 0 1 0 1 0 1 0 1 1 1 0 1 1
  7603.0> 1 0 1 0 1 1 1 0 1 0 1 0 1 0 1 0 1 0 1 1
 10000.0> 1 0 1 0 1 0 1 1 1 0 1 0 1 0 1 0 1 0 1 1
 10029.9> 1 0 1 0 1 0 1 1 1 0 1 0 1 0 1 1 1 1 1 1
 10086.9> 1 0 1 0 1 0 1 1 1 0 1 0 1 0 1 0 1 1 1 1
 10102.9> 1 0 1 0 1 0 1 1 1 0 1 0 1 0 1 0 1 0 1 1
 12500.0> 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 1
 12529.9> 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 1 1 1
 12586.9> 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 1
 17500.0> 0 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 1
 17529.9> 0 0 1 0 1 0 1 0 1 0 1 0 0 0 1 0 1 0 1 1
 17586.9> 0 0 1 0 1 0 1 0 1 0 1 0 0 0 1 0 0 0 1 1
 17587.0> 0 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 1
```

Design Methodology

Designs may be efficiently implemented in EPM5128s and other MAX EPLDs with hierarchical design entry. MAX+PLUS supports eight levels of hierarchy, and any entry format may be used for every design level.

For designing system-level functions within the EPM5128, MAX+PLUS supports both top-down and bottom-up design approaches. The serial coprocessor, for example, may be created by first using the top-down method to draw a functional block diagram as shown in Figure 10. Each top-level functional block is partitioned into smaller blocks. This process is repeated until the design consists entirely of simple functional blocks.

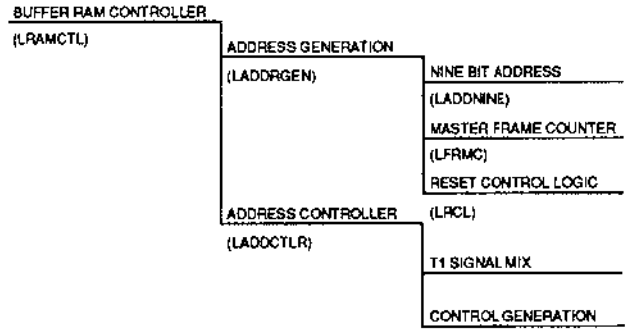
Figure 10. T1 Serial Coprocessor Block Diagram



These functional blocks may then be logically implemented in a bottom-up approach. As each simple logic function is designed, MAX+PLUS automatically creates a symbol to represent that function. Each of these low-level functions may then be independently compiled and simulated, and the simple blocks integrated into more complex blocks. The blocks are easily integrated by wiring the symbols together to create the next level of hierarchy. The designs that result from the integration of multiple low-level functions are similarly compiled and simulated before being integrated into the next level of the design. This bottom-up process is repeated until all blocks are integrated to create the complete T1 serial coprocessor.

The buffer RAM controller block of the T1 serial coprocessor is also implemented with this bottom-up method. Figure 11 shows the hierarchical definition of the buffer RAM controller, called LRAMCTL.

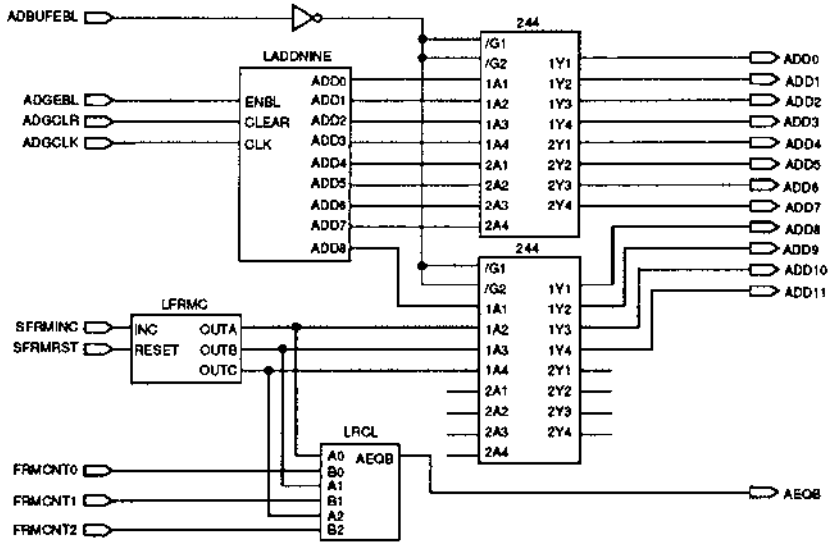
Figure 11. Hierarchical Definition of Buffer RAM Controller



The lowest-level functions of LRAMCTL—a 9-bit address generator, master frame counter, and reset control logic—are implemented first as the schematics LADDNINE, LFRMC, and LRCL, respectively. These schematics are then integrated to create the address generation function, LADDRGEN, by wiring the automatically generated symbols together. See Figure 12.

Figure 12. Address Generation (LADDRGEN) Function

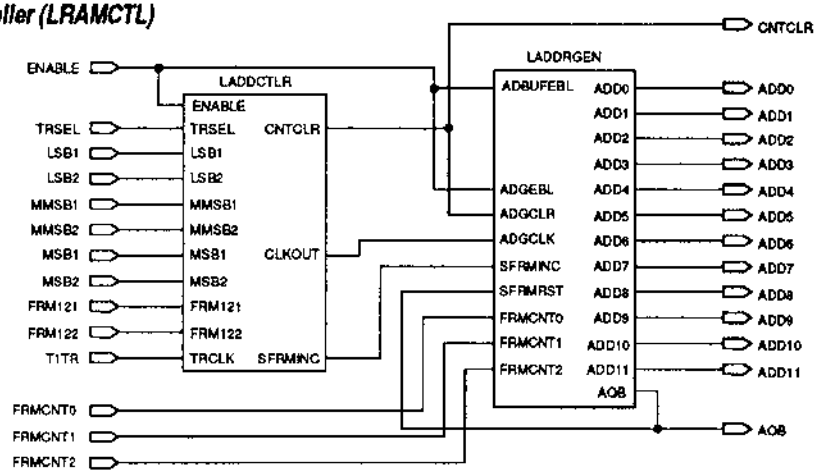
Functions such as the address generator for buffer RAM are individually compiled and simulated before integration into the next level of logic.



LADDRGEN is then compiled, simulated, and integrated with the address control generator, LADDCTLR, which is also built up from simple logic functions that had been independently compiled and simulated. The integration of LADDRGEN and LADDCTLR creates LRAMCTL, the buffer RAM controller. LRAMCTL, shown in Figure 13, is then simulated and integrated into the top level of the serial coprocessor.

Figure 13. Buffer Ram Controller (LRAMCTL)

The buffer RAM controller is built out of two custom logic functions.



After the design is fully integrated and compiled, it may be simulated. Figure 14 shows the logical hierarchy of the complete serial coprocessor.

T1 Serial Coprocessor Example

The T1 protocol for serial data transmission has been the basis for most digital voice communications since its introduction in the early 1960s. Recently, the protocol was revised to include support for digital data communications. The protocol is based on a pulse code modulation system in which multiple channels are time-division multiplexed onto a 1.544-MHz channel. The EPM5128's flexible architecture and the MAX+PLUS CAE development environment enable designs such as the T1 serial transmitter subsystem to be designed and integrated into a system in just days.

Figure 14. Hierarchical Definition of the T1 Serial Coprocessor

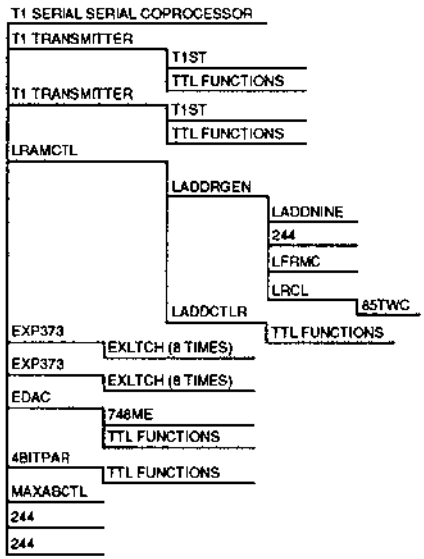


Figure 10 shows the functional block diagram of the T1 serial coprocessor. While the data bytes are written through the EPM5128 to RAM, a 4-bit parity code for error detection and correction is generated. The resulting 12 bits of data are stored in buffer RAM. The data for transmission is broken up into master frames. Each master frame consists of 12 frames, and each frame consists of 24 data bytes. The buffer RAM may hold 8 master frames at a time. The processor also writes 2 header command words for each master frame. After the system processor has transferred up to 8 master frames into buffer RAM, the processor writes the number of master frames into the master frame count register and issues a start command to the serial coprocessor.

The serial transmit controller is a state machine that controls functions of the coprocessor. Upon receiving the start

command, the serial transmit controller takes control of the data paths from the system processor and reads the command words for the first master frame. Buffer RAM addresses are generated by buffer RAM control. The command words specify the transmitter to be used for transmission and protocol information for the T1 transmitters. Once the command words are read, the data bytes of the first master frame are transferred from RAM to the specified T1 transmitter.

During the transfer from RAM to the T1 transmitter, the data passes through the error detection and correction (EDAC) circuitry. The circuitry uses a modified Hamming code. The data byte is then latched into the specified T1 serial transmitter, serialized, and transmitted. After a complete master frame has been transferred, the buffer RAM controller queues the next master frame so the next transfer may begin. This process continues until all of the frames have been transferred without processor intervention; then the control state machine issues an interrupt to the system processor, indicating that the transfer is complete, and the next cycle may begin.

The following T1 serial coprocessor parts are discussed here:

- ❒ Error Detection and Correction
- ❒ Buffer RAM Control
- ❒ Serial Transmit Controller
- ❒ MFREG and COMMAND LATCH
- ❒ T1 Transmitter

Error Detection and Correction

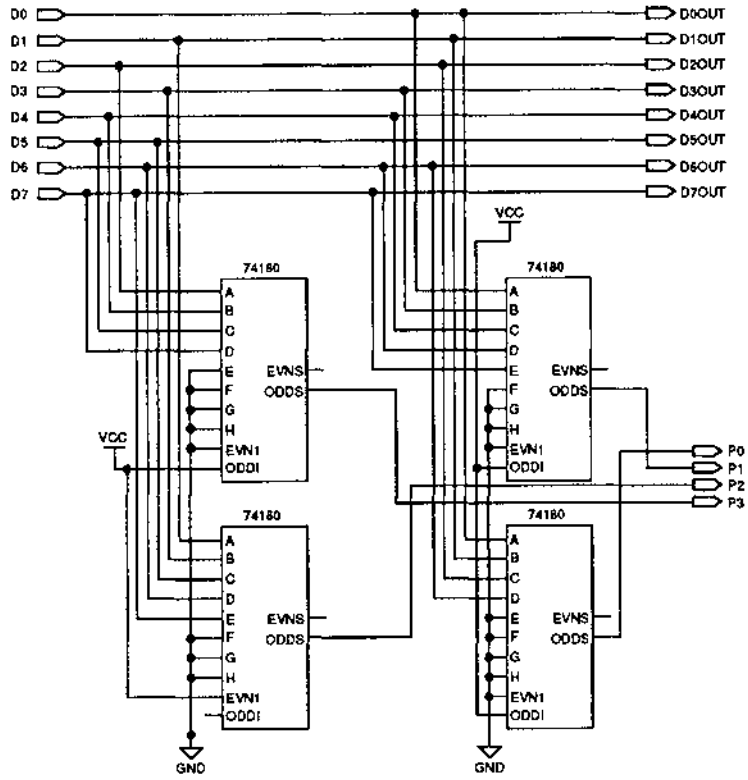
The error detection and correction (EDAC) circuitry consists of two functions: the 4-bit parity code generator and the EDAC function. The circuitry detects and corrects all single-bit errors and more than 75% of multiple-bit errors. (The multiple-bit error detection feature is not used in this design example.)

The parity code generator, 4BITPAR, is shown in Figure 15. The four-bit code is generated by four 74180 parity generator devices. Note that each parity code bit is generated by a different set of data bits, and each data bit is used in generating two of the code bits. When the system processor writes a data byte to the transmit buffer RAM, the 4-bit parity code is automatically generated and written into the RAM with the data byte. 4BITPAR consumes four macrocells and replaces four TTL MSI components.

The EDAC function, shown in Figure 2, uses the 4-bit parity code to detect and correct single-bit errors in the data word. The 74180s in EDAC generate a 4-bit code based on the 8 data bits and the 4 parity bits. This 4-bit code is decoded by the 74154. The 74154 output is then used to correct any single-bit errors in the data byte. While the serial transmit controller is transferring data to one of the T1 transmitters, error detection and correction is performed automatically. EDAC consumes 10 macrocells and replaces 7 TTL packages.

Figure 15. Four-Bit Parity Code Generator (4BITPAR)

4BITPAR generates error detection and correction codes. It is composed of four 74180s from Altera's TTL MacroFunction Library.



Buffer Ram Control

Figure 11 shows the schematic for the function LRAMCTL, the controller for the 4 K x 12-bit buffer RAM. LRAMCTL, which replaces 8 TTL MSI packages and consumes 15 macrocells, consists of 2 functions: LADDCTLR and LADDRGEN. LADDCTLR multiplexes control signals from the T1 serial transmitters to select the signals from the active transmitter, which are then decoded to generate control signals for LADDRGEN. LADDRGEN generates the address ADD0-ADD11 for the buffer RAM.

The Clear function for the ADD0-ADD9 address is implemented with complex combinatorial logic. Figure 16 shows the macrocells occupied by the address bits in a portion of the Report File produced by MAX+PLUS for this example. The Compiler's Logic Synthesizer allocates the expanders of the EPM5128 to implement the function. The total number of expanders used by each macrocell is shown in Column 5 of the Report File, and the number of shared expanders is shown in Column 6. Shared expanders are produced when the Logic Synthesizer scans designs for common product terms and places them on expanders, ensuring the most efficient design

implementation. The ADD0-ADD9 clear function is implemented with 10 expanders, which are shared by all macrocells associated with the function.

The fan-in for each macrocell is also shown in the Report File. ADD7, for example, has a total fan-in of 19 signals, illustrating MAX architecture's wide logic gating. Implementing such a function in an LCA device would require multiple logic levels, resulting in severe performance degradation. The EPM5128 may have any of 128 signals feeding into each macrocell.

Figure 16. Report File

The Report File shows expander usage and sharing, as well as macrocell fan-in. Expander sharing helps ensure efficient use of device resources.

C:\MAXWORK\MAXAB\ABMAX.RPT

OUTPUTS

				Expanders		Fan-In		
Pin	MCell	LAB	Primitive	Total	Shared	INP	FBK	Name
38	65	E	DFF	10	10	3	11	ADD0
39	66	E	DFF	10	10	3	13	ADD1
40	67	E	DFF	10	10	3	13	ADD2
41	68	E	DFF	10	10	3	14	ADD3
24	49	D	DFF	10	10	3	13	ADD4
25	50	D	DFF	10	10	3	15	ADD5
26	51	D	DFF	10	10	3	15	ADD6
27	52	D	DFF	11	10	3	16	ADD7
28	53	D	DFF	10	10	3	15	ADD8

Serial Transmit Controller

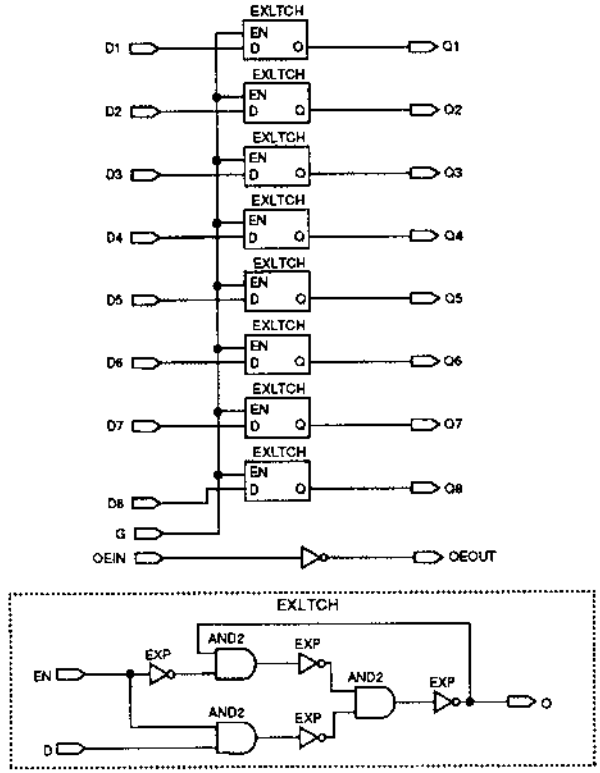
The serial transmit controller, MAXABCTL, is the controlling state machine for the T1 Serial I/O subsystem. Once the system processor has written the number of master frames for transfer into the register MFREG, the outputs of MAXABCTL are used to control the subsystem. INTRO is the interrupt to the system processor, indicating that all master frames are transferred. A symbol for MAXABCTL,entered with state machine language, is automatically generated by SMF2CNF supplied with MAX+PLUS. The symbol is then integrated into the top-level design. MAXABCTL uses nine macrocells and replaces four and a half 7474 macrofunctions.

MFREG and COMMAND LATCH

MFREG and COMMAND LATCH are latches used to control the subsystem. The system processor writes the number of master frames to be transmitted into MFREG. MAXABCTL then receives the signal SFRAME and the serial transmission cycle begins. MAXABCTL writes to COMMAND LATCH, using one of the command words that is stored for each master frame. This word controls certain aspects of the T1 transmission cycle. Both MFREG and COMMAND LATCH are implemented with EXP373s, which are input latches composed entirely of expanders (see Figure 17).

The flexible MAX architecture allows cross-coupling of expanders to form individual latches such as EXLTCH, which is also shown in Figure 17. EXP373 consists of 8 EXLTCH latches. The EPM5128 may implement up to

Figure 17. EXP373 Input Latch *The EPM5128 can implement 80 D-type latches on expander product terms without consuming a single macrocell.*



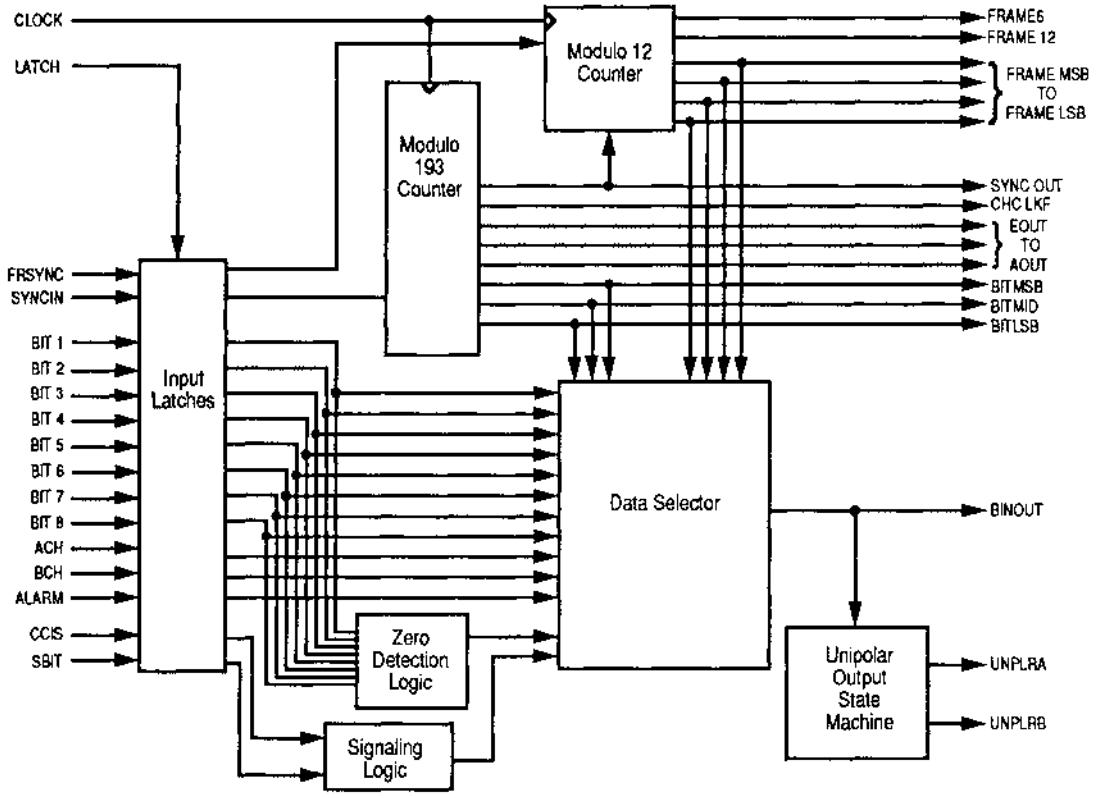
128 S-R latches on expanders or up to 80 flow-through latches without using a single macrocell. By implementing latches on expanders, macrocells are conserved for functions that require the full resources of the macrocell. The two EXP373 octal latches implement 11 latch bits and use 33 expanders and no macrocells.

T1 Transmitter

A block diagram of the T1 transmitter is shown in Figure 18. The T1 protocol uses pulse code modulation to time-multiplex 24 channels of data into a 1.544-MHz carrier frequency. The 1.544-MHz clock is used to generate bit and channel timing via a modulo 193 counter. A separate modulo 12 counter is used for frame timing. Each of the transmitters consumes 39 macrocells and replaces 13 TTL MSI components.

All inputs to the transmitter are latched during the transmission of the 8th bit of each data byte. The data selector, controlled by the bit counter and the frame counter, provides the proper sequence of bits on signal BINOUT.

Figure 18. Block Diagram of the T1 Serial Transmitter



The state machine T1ST uses BINOUT to provide unipolar outputs to create a single bipolar data transmission. BINOUT is a 12-product term equation with a fan-in of 20 signals. Consult the Altera *Applications Handbook* for further details about the T1 serial transmitter design.

Conclusion

Figure 19 shows the full complement of the TTL functions used in the T1 serial coprocessor. In this design, 53 TTL MSI functions were replaced, along with SSI gate functions such as AND, NAND, and OR gates of various widths. The number of packages required to implement these SSI functions is conservatively estimated at 22. This number is based on the width of the gates and ignores the fact that some of the functions, such as the wide NOR and AND gates, are not commercially available. These functions would have to be constructed out of several TTL SSI gates.

Since some SSI functions would have to be constructed out of TTL SSI gates, this design therefore replaces 75 or more TTL packages—equivalent to an entire board of TTL logic. This design may thus reduce the printed circuit boards used in a system. At the same time, system performance improves through the independent operation of the subsystem.

Figure 19. TTL Functions Used in the T1 Serial Coprocessor

75 TTL MSI and SSI functions are integrated into an EPM5128 to implement the T1 subsystem.

T1 Subsystem Component	TTL MSI or SSI Functions Used	Number Used
4BITPAR	74180	4
EDAC	74180	4
	7486	2
	74154	1
LRAMCTRL	74162	3
	74244	2
	74193	1
	7485	1
	74157	1
MAXABCTL	7474	4
Top Level Schematic	74244	2
	74373	2
TITR (2 times)	7474	12
	74161	6
	74373	4
	74153	2
	74151	2
		= 53 TTL Macrofunctions
TTL Gate Logic	inverter	25
	2-input gate	32
	3-input gate	14
	4-input gate	8
	8-input gate	3
		= 22 TTL SSI Packages
Total Functions Used = 75 Packages		

Furthermore, the entire process of functionally defining the serial coprocessor, entering the design, and simulating the logic at all levels, may take only one week. In contrast, laying out and routing tasks alone would take about the same time for a multi-layer PCB version of this I/O subsystem. Any changes in the EPM5128 design may also be made, often by just reprogramming the EPLD. Changes to the PCB version, on the other hand, would usually require many cuts and jumpers and would take considerably longer.

As the dual T1 serial transmitter subsystem shows, a single EPM5128 may integrate over 50 TTL MSI components plus a large amount of glue logic. The EPM5128's flexible chip architecture offers efficient implementation of both simple and complex logic functions, with true emulation of TTL devices. Integration is quick and efficient as a result of the chip architecture and advanced design environment offered by MAX+PLUS.

Further details about this design, including design and simulation files, are available upon request from Altera's Applications Department or the Altera Electronic Bulletin Board Service.

The following list shows the available files:

ABMAX.GDF	LRAMCTL.GDF	T1ST.CNF	74153.GDF
EXLTCH.GDF	LRCL.GDF	T1ST.SMF	74154.GDF
EXP373.GDF	LFRMC.GDF	T1TR.GDF	74157.GDF
EDAC.GDF	MAXABCTL.GDF	244.GDF	74161.GDF
LADDNINE.GDF	MAXABCTL.CNF	4BITPAR.GDF	74162.GDF
LADDRGEN.GDF	MAXABCTL.SMF	85TWC.GDF	74180.GDF
LADDCTLR.GDF	T1ST.GDF	7486ME.GDF	74193.GDF
		74151.GDF	74373.GDF

References

Altera Corporation. *Applications Handbook* (1988).

Understanding MAX EPLD Timing

January 1990 **Application Brief 75**

Introduction

The advanced technology of Altera's Multiple Array MatriX Erasable Programmable Logic Devices (MAX EPLDs) and the sophisticated MAX+PLUS Development System have made CMOS programmable logic much easier to use. Complex logic designs can be implemented by one MAX EPLD. In addition, the MAX+PLUS Delay Predictor and Simulator can identify critical delay paths and find each path's worst-case delays, which are the sum of discrete component delays inherent in the MAX architecture.

This Application Brief discusses these internal delay paths, their relationships to AC specifications (shown in the EPLD data sheets), and the calculated timing delays generated by MAX+PLUS. In addition, MAX timing models for analyzing delays calculated by the Simulator or Delay Predictor are discussed, and equations are used to calculate the delays.

MAX EPLD Architecture Basics

To accurately model timing characteristics, a designer must understand how logic is implemented in a MAX EPLD. The MAX architecture is based on a flexible Logic Array Block (LAB), which is composed of three parts: the macrocell array, the expander product-term (expander) array, and the I/O control block. The number of macrocells, expanders, and I/O control blocks varies, depending on the device used. Large MAX devices contain multiple LABs that are interconnected by a Programmable Interconnect Array (PIA). The PIA is fed by macrocell and I/O pin feedback, and globally routes signals within devices containing more than one LAB.

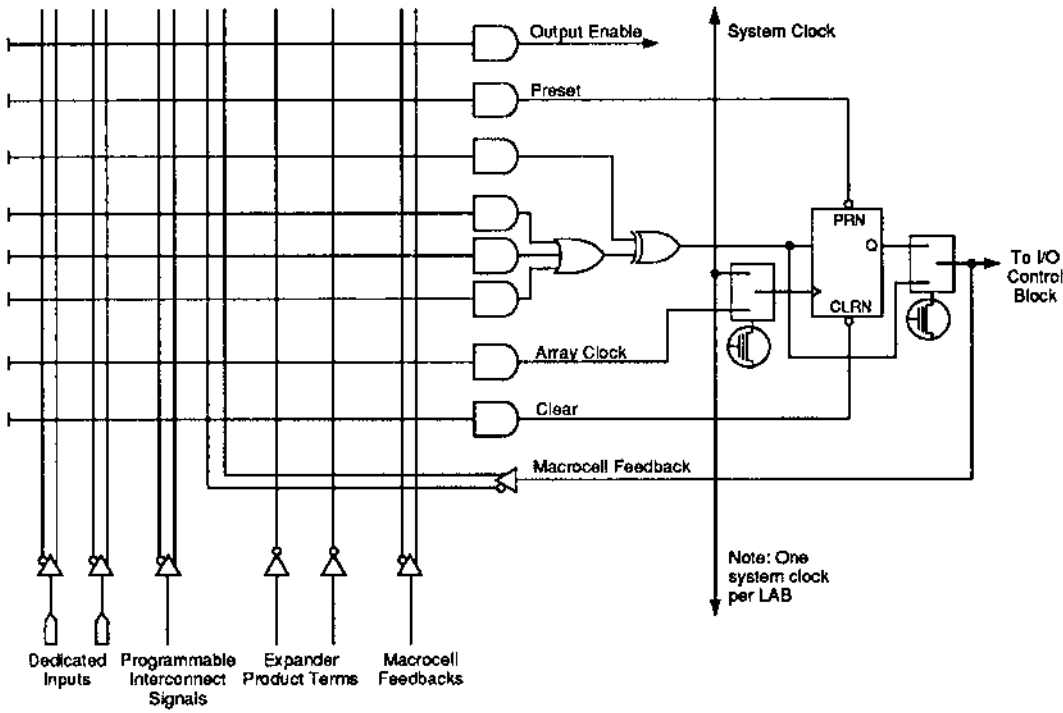
All gated and registered logic is implemented in MAX macrocells and expanders. The expander product-term array and the MAX macrocell contain product terms that are n-input AND gates (where n represents the number of connections). Depending on the implemented logic, a product term may be logically equivalent to one or more gates. (Refer to the MAX Device data sheets for more information.)

Macrocell Array

The macrocell structure of MAX devices, shown in Figure 1, has been optimized to handle variable product-term requirements. Each macrocell consists of a programmable AND, a fixed OR array, an XOR gate, a configurable register, and a number of inputs (varying from 80 to 144) that together enable the generation of very large product terms.

Combinatorial logic is implemented in the macrocell with three product terms ORed together, which feed the XOR gate. The second input to the

Figure 1. Macrocell Array



XOR gate is also controlled by a product term, providing the ability to control active-high or active-low logic. MAX+PLUS uses this gate to implement complex, mutually exclusive-OR logic, or to apply DeMorgan's inversion, reducing the number of product terms required to implement a function. The macrocell also includes additional product terms (secondary product terms) that are used for Output Enable, Preset, Clear, and Clock logic. If more product terms are required to implement a function, they may be added to the macrocell from the expander product-term array.

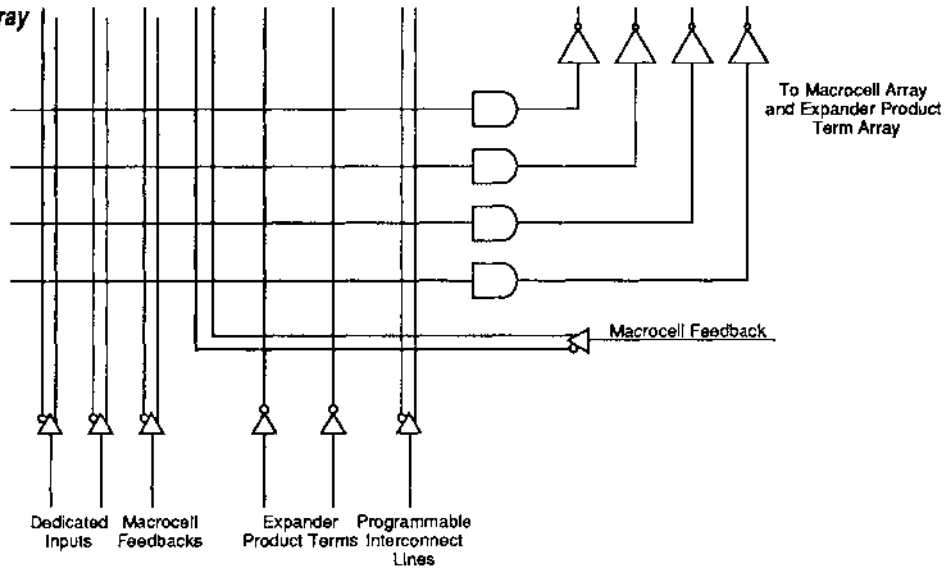
The MAX macrocell's AND array is an EPROM array; each product term generated by the AND array is a function of the EPROM bits within the array. The EPROM bits, initially erased, serve as electrical switches for the array inputs. An erased bit enables an input to a product term, while a programmed bit prohibits an input from reaching a product term. A product term is thus a function of only inputs connected by unprogrammed, non-erased EPROM bits.

The configurable register within a MAX macrocell may be programmed to D, T, JK, SR, or flow-through operation with independent, programmable Clock, Reset, and Preset options. Alternately, it may be bypassed entirely for purely combinatorial logic.

Expander Product Term Array

Additional product terms, called expanders, may feed the macrocell array's product terms to generate very complex Boolean logic. The expander product-term array, shown in Figure 2, is a programmable AND array with inversion. Expanders are fed by the dedicated input bus, the PIA, the macrocell feedback, expanders themselves, and the I/O pin feedbacks. The outputs of the expanders then go to each product term in the macrocell array. Since these expanders feed the secondary product terms (Preset, Clear, Clock, and Output Enable) of each macrocell, complex logic functions may be implemented without utilizing another macrocell.

Figure 2. Expander Product Term Array



These expanders are used and shared by the macrocells, allowing complex functions (more than 32 product terms) to be easily implemented in a single macrocell. Expanders may also feed other expanders, giving the user the ability to implement complex multi-level logic and input latches. As illustrated in Figure 2, the expander product-term array contains an AND array followed by inversion. It may supply from 32 to 64 additional product terms per LAB. Large MAX EPLDs such as the EPM5192, EPM5130, EPM5128, EPM5127, and EPM5064 provide up to 32 expanders per LAB; smaller MAX EPLDs, such as the EPM5032, EPM5024, and EPM5016 provide 64, 48, and 32 expanders, respectively. Inputs into the expander product-term array also vary from 80 to 144.

I/O Control Block

In the LAB, the I/O control block is separate from the macrocell array. It contains programmable tri-state buffers and I/O pins with optional feedbacks. Each I/O pin may be configured for dedicated input, or a

macrocell may be connected to an I/O pin, allowing the pin to be used as a dedicated output or as a bidirectional pin. The input of the tri-state buffer comes from a macrocell within the associated LAB. The feedback path from the I/O pin may feed other blocks within the LAB, as well as the PIA.

Programmable Interconnect Array

The EPM5192, EPM5130, EPM5128, EPM5127, and EPM5064 MAX EPLDs have multiple LABs, interconnected by a Programmable Interconnect Array (PIA) that globally routes all signals within devices containing more than one LAB. The PIA avoids interconnect limitations by routing only the signals needed by each LAB, effectively solving any routing problems that may arise in a design without degrading device performance. In addition, the PIA has a fixed delay from point to point. The PIA delay is constant because each signal that feeds into the PIA has its own dedicated metal line with multiple taps, one for each LAB. Undesired skews among logic signals, which may cause glitches in internal or external logic, are eliminated.

EPLD Delay Parameters

Internal delays within an EPLD are described by a number of AC parameters (called MicroParameters) that refer to the actual internal delay within the device. Figures 3 shows the timing model for single LAB devices.

Figure 3. EPM5032 / EPM5024 / EPM5016 Timing Model

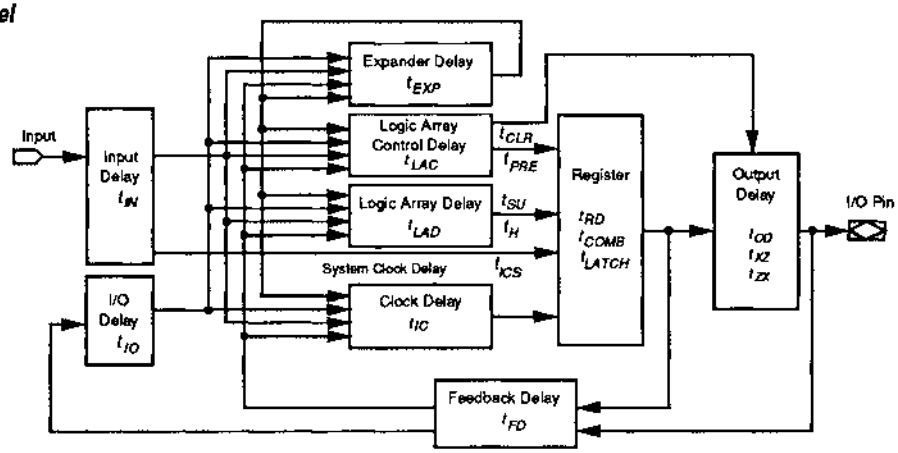
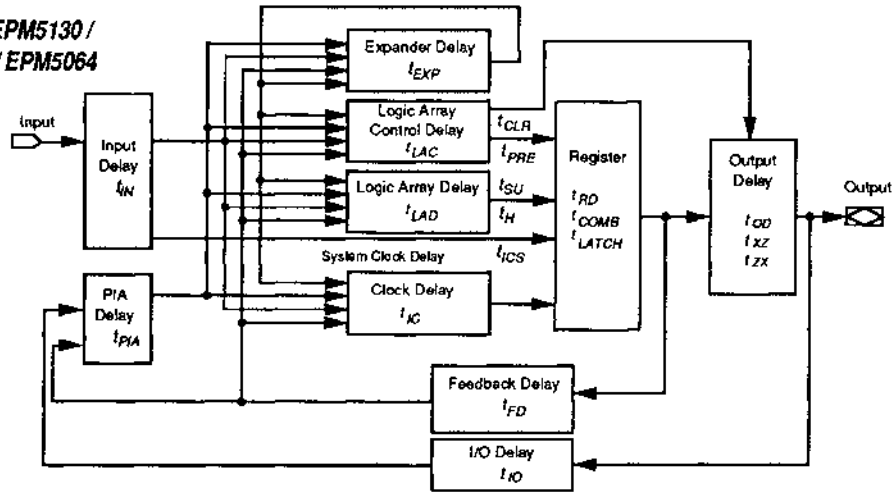


Figure 4 shows the timing model for multiple LAB devices. Following is a list of these MicroParameters and examples of how to predict timing delays with equations that use these MicroParameters.

Figure 4. EPM5192 / EPM5130 / EPM5128 / EPM5127 / EPM5064 Timing Model



t_{IN} Input pad and buffer delay that directs the true and complement input signals from the dedicated input pin into the LAB. Within the LAB, the signals may propagate to any of the four arrays: expander product-term array, logic array, logic array control, and clock array.

t_{IO} I/O input pad and buffer delay for I/O pins used as inputs. The t_{IO} delay value must be substituted for t_{IN} for the EPM5032, EPM5024, and EPM5016. When an I/O pin is used as an input, the t_{IO} delay value must also be added to t_{PIA} to obtain the total delay from the I/O pin to the LAB for the EPM5192, EPM5130, EPM5128, EPM5127, and EPM5064.

t_{EXP} Expander product-term array delay, which is the delay through the AND-INVERT structure of the expander product-term array. It is added to the delay already present in the four arrays when expanders are used, or added to itself when an expander feeds another expander.

t_{LAC} Logic array control delay. It is the delay through the AND array by Clear, Preset, and Output Enable signals, representing the time required to propagate through the AND array to the CLRN and PRN inputs of the register, and the OE signal to the tri-state buffer.

t_{CLR} Asynchronous register clear time, which represents the time required to reset a register output to a logical "low." It is the delta from the time the register CLRN input is asserted low to the time the register output stabilizes at logical "low."

t_{PRE} Asynchronous register preset time, representing the amount of time required to set a register output to a logical "high." It is the delta from the time the register PRN input is asserted low to the time the register output stabilizes at logical "high."

t_{LAD} Logic array delay. It is the time a signal requires to propagate through a macrocell's EPROM AND array, the three-input OR gate, and the two-input XOR gate.

t_{ICS} System clock delay. The t_{ICS} value measures the delay from the dedicated clock pin to a register's clock input.

t_{IC} Clock delay. The t_{IC} value is the delay through a macrocell's clock product term to the register clock input.

t_{FD} Feedback delay. For the EPM5032, EPM5024, and EPM5016, it is the propagation time from macrocell output or output pin to any of the LAB's four arrays. For the EPM5192, EPM5130, EPM5128, EPM5127, and EPM5064, it is the propagation time from macrocell output or output pin to any of the LAB's arrays, and the propagation time from macrocell output to PIA input or other macrocells in the LAB.

t_{SU} Setup time required for a signal to be stable at register input before the clock's rising edge.

t_{H} Hold time required at register input after the register clock's rising edge to ensure that the register stores the input data.

t_{RD} Delay from register clock's rising edge to the time that output appears at the register output.

t_{COMB} Combinatorial buffer delay, which is used only for combinatorial logic. It is the delay from the time the logic array's XOR output bypasses the programmable register to the time it becomes available for macrocell output.

t_{LATCH} Propagation delay through the latch from latch input to latch output.

t_{OD} Output pad and buffer propagation delay from macrocell output through the tri-state output buffer to the output pin.

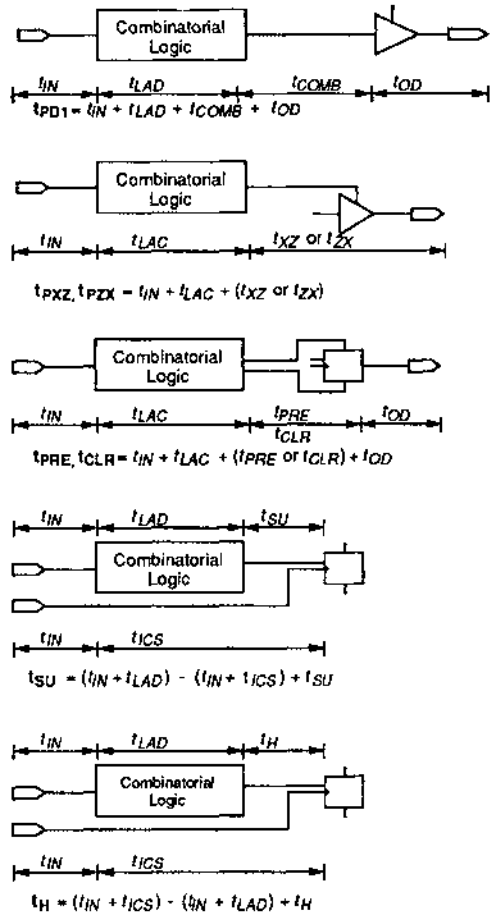
t_{XZ} Delay required for high-impedance to appear at the output pin after output buffer's active-high enable control is brought logically "low."

t_{ZX} Delay required for macrocell output to appear at the output pin after output buffer's active-high enable control is brought logically "high."

t_{PIA} Programmable Interconnect Array delay for multiple LAB devices. It is used for analyzing EPM5192, EPM5130, EPM5128, EPM5127, and EPM5064 designs that use a PIA for routing. The PIA delay path starts where the macrocell feedback or I/O delay path ends and ends where it enters the LAB and reaches any of its four arrays.

Critical pin-to-pin delays (denoted by a boldface t) are shown in Figure 5.

Figure 5. Critical Pin-to-Pin Delays



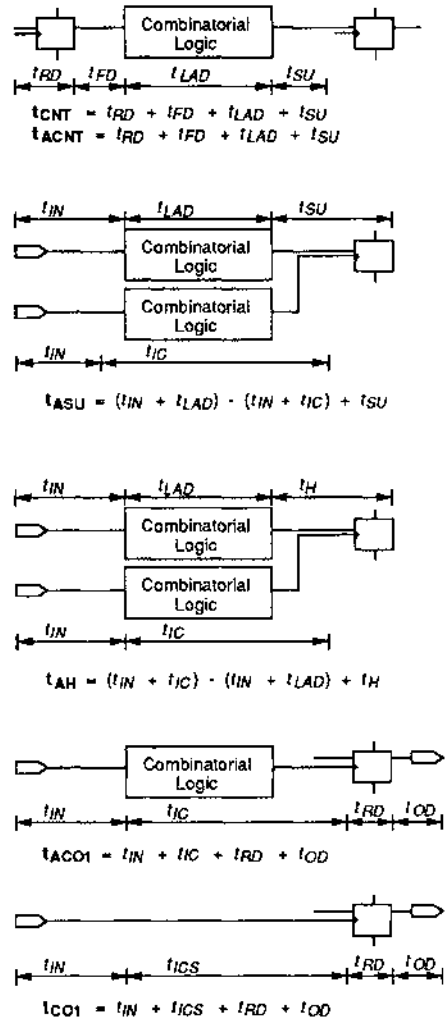
Notes:

1. *If an I/O is used for input:*
   a. *for EPM5032, EPM5024, and EPM5016, substitute t_{IO} for t_{IN}.*
   b. *for EPM5192, EPM5130, EPM5128, EPM5127, and EPM5064, substitute $t_{IO} + t_{PIA}$ for t_{IN}.*
2. *If expanders are used for complex logic, add t_{EXP} to delay path.*

Critical pin-to-pin delay calculations are shown in Figure 6. The calculations used to get these values from the MicroParameters listed in the data sheet are shown for each path. Note that these calculations assume that an input pin is used.

If the input would come from an I/O pin, t_{IO} would be substituted for the t_{IN} value. For multiple LAB MAX devices using an I/O pin as an input, $t_{IO} + t_{PIA}$ would be substituted for the t_{IN} value. If an expander is used in the path at any time, the t_{EXP} value must also be added to the total delay path.

Figure 6. Critical Pin-Delay Calculations



Notes:

1. *If an I/O is used for input:*
   a. *for EPM5032, EPM5024, and EPM5016, substitute t_{IO} for t_{IN}.*
   b. *for EPM5192, EPM5130, EPM5128, EPM5127, and EPM5064, substitute $t_{IO} + t_{PIA}$ for t_{IN}.*
2. *If expanders are used for complex logic, add t_{EXP} to delay path.*

Examples

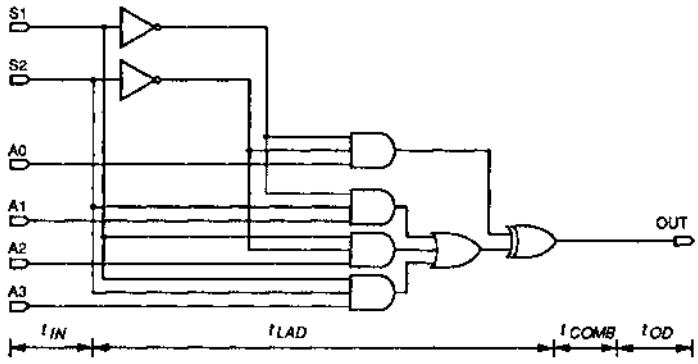
The MAX+PLUS Simulator and Delay Predictor can identify timing delays for any circuit. These delays may be separated into the MicroParameters already described and are provided in the MAX devices data sheets.

Example 1: 4-to-1 Multiplexer

The design shown in Figure 7 represents a 4-to-1 multiplexer, which is a combinatorial circuit. The circuit consists of four data inputs, two select controls, and one output, and is partitioned into four delay paths: input delay, logic array delay, combinatorial buffer delay, and output delay.

Figure 7. 4-1 MUX Logic Timing

This circuit can be partitioned into 4 delay paths: input delay, logic array delay, combinatorial buffer delay and output delay.



Input delay will be t_{IN} if only dedicated inputs are used. If I/O pins are used, the input delay will be t_{IO} for the EPM5032, EPM5024 and EPM5016, and will be $t_{IO} + t_{PIA}$ for the EPM5192, EPM5130, EPM5128, EPM5127, and EPM5064.

The propagation delay for combinatorial logic when bypassing the programmable register is t_{COMB}. For output data, the output pad and buffer delay is t_{OD}. The overall propagation delay is t_{PD1} when using all dedicated inputs and t_{PD2} when using any I/O pin for input.

The propagation delay from input pin to I/O pin is the sum of the input delay, logic array delay, combinatorial buffer delay, and output delays: $t_{IN} + t_{LAD} + t_{COMB} + t_{OD}$. This worst-case delay is also known as t_{PD1} (from input pin to output pin) or as t_{PD2} (from I/O pin to output pin). The maximum t_{PD1} and t_{PD2} are shown in the respective MAX data sheets, or may be determined with the following equations.

For MAX EPLDs without a PIA:

$$t_{PD1} = t_{IN} + t_{LAD} + t_{COMB} + t_{OD}$$
$$t_{PD2} = t_{IO} + t_{LAD} + t_{COMB} + t_{OD}$$

For MAX EPLDs with a PIA:

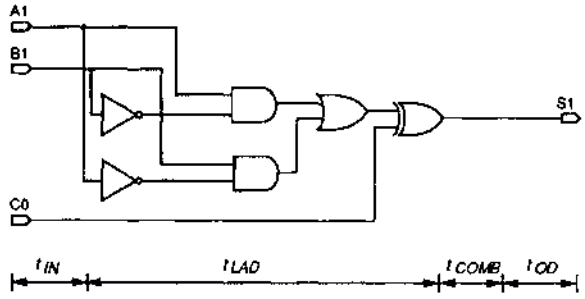
$$t_{PD2} = t_{IO} + t_{PIA} + t_{LAD} + t_{COMB} + t_{OD}$$

Example 2: 7483 TTL Macrofunction

The timing delays for macrofunctions subjected to logic synthesis can also be analyzed. The synthesized logic equations may be obtained from the "long" version of the Report File and have been structured so a designer can quickly determine the logic configuration. (Refer to *Report File* in the *MAX+PLUS User Guide*.) For example, the equations for **S1**, the least significant bit of the 7483 TTL Macrofunction (a 4-bit full adder) are:

```
S1      =  OUTPUT(_MC021, VCC);
_MC021  =  MCELL(_EQ026 $ C0);
_EQ026  =  B1 & A1'
        #  B1' & A1;
```

Figure 8. Adder Logic Timing



where **S1** is the output of macrocell 21 (**_MC021**), which contains combinatorial logic. The combinatorial logic, **MCELL(_EQ026 $ C0)**, represents the exclusive-OR of the intermediate equation (**_EQ026**) and the carry-in (**C0**). In turn, **_EQ026** represents logic equivalent to the exclusive-OR of inputs, **B1** and **A1**. See Figure 8.

Therefore, the timing delay for S1 is t_{IN} + t_{LAD} + t_{COMB} + t_{OD}, which is equal to t_{PD1}.

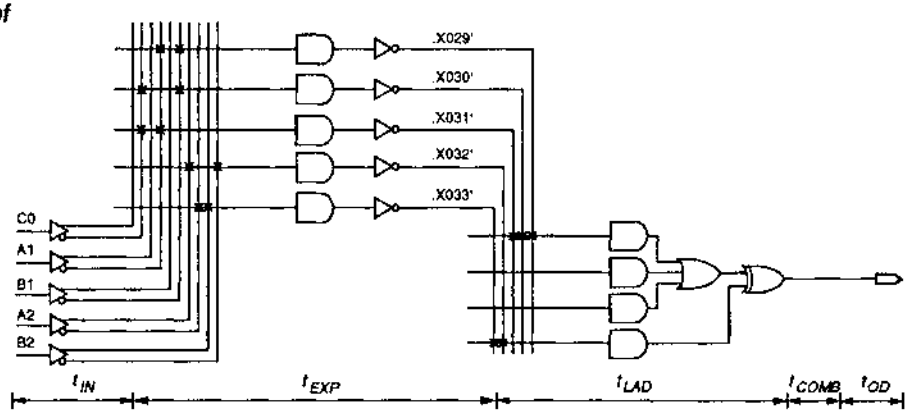
Example 3: S2

For complex logic that requires expanders (represented here by **_X**...), the expander-array delay, t_{EXP}, is added to the delay element. For instance, **S2**, the second bit of the full adder, requires expanders. The equations are:

```
S2     = _MC019;
_MC019 = MCELL(  _EQ023 $  _EQ024 );
_EQ023 = _X029 & _X030 & _X031;
_X029  = EXP( !B1 & !A1 );
_X030  = EXP( !B1 & !C0 );
_X031  = EXP( !A1 & !C0 );
_EQ024 = _X032 & _X033;
_X032  = EXP( !B2 &  A2 );
_X033  = EXP(  B2 &  A2 );
```

Using these equations, the logic structure can be mapped onto MAX architecture, as shown in Figure 9.

Figure 9. Mapping of Adder Equation to MAX Architecture



The overall delay for S2 is equivalent to $t_{IN} + t_{EXP} + t_{LAD} + t_{COMB} + t_{OD}$, which is equal to $t_{EXP} + t_{PD1}$.

Example 4: Asynchronous 4-Bit Counter

The last example (shown in Figure 10) evaluates an asynchronous 4-bit counter. The counter has one logic-controlled clock input and five outputs: CLK, QD, QC, QB, and QA. In addition, it has five inherent delays associated with registered logic (clock delay, input delay, array delay, feedback delay, and output delay) as well as setup time and hold time requirements for each register.

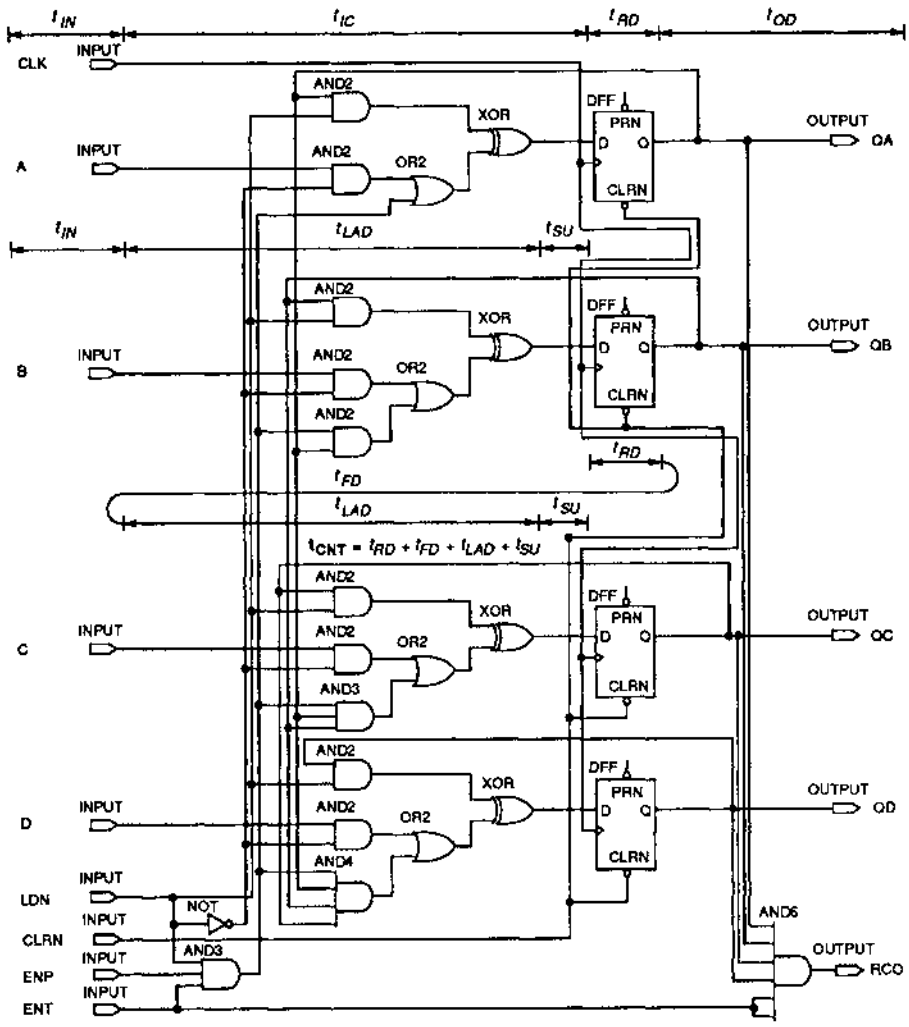
The propagation delay from input pin to the clock input of the register for the least significant bit QA is $t_{IN} + t_{IC}$ (see Figure 10). If an I/O pin is used for input, the propagation delay from I/O pin to the register's clock input is $t_{IO} + t_{IC}$, or $t_{IO} + t_{PIA} + t_{IC}$ for MAX EPLDs with multiple LABs. Since the delay from register to output pin is $t_{RD} + t_{OD}$, the total clock to output delay is $t_{IN} + t_{IC} + t_{RD} + t_{OD}$ for dedicated input to output, $t_{IO} + t_{IC} + t_{RD} + t_{OD}$ for I/O pin to output for MAX EPLDs with one LAB, or $t_{IO} + t_{PIA} + t_{RD} + t_{OD}$ for I/O pin to output for MAX devices with multiple LABs.

In addition, data input to the register must meet both setup and hold time requirements. The internal setup time is the time needed for the input data to stabilize before the triggering edge of the clock appears at the register input. The external setup time is the difference between the sum of the input, logic array and setup time, and the sum of the input and clock delay: $(t_{IN} + t_{LAD} + t_{SU}) - (t_{IN} + t_{IC})$. When expanders are used, t_{EXP} must be added, and when I/O pins are used, t_{IO} or $(t_{IO} + t_{PIA})$ must be added. As long as the external setup time is met at the inputs, the counter functions properly.

The maximum internal counter frequency (designated by f_{CNT}) represents the inverse of t_{CNT}, which is the worst-case delay for internal feedback. This frequency is the minimum internal clock period at which the counter can operate correctly. The delay is the sum of delay paths that the register feedback must traverse before reaching a register input and meeting the internal setup time: $(t_{RD} + t_{FD} + t_{LAD} + t_{SU})$. Once the clock triggers QB, data takes t_{RD} delay prior to appearing at the register output. The signal feeds back (t_{FD}) and flows through the logic array (t_{LAD}). Finally the signal reaches the register QC and meets the setup time of the register (t_{SU}).

When expanders are used, t_{EXP} must be added as the signal passes through the expander array before reaching the logic array. The t_{CNT} delay represents only internal circuit delays, while a circuit which depends on external and internal signals must also account for input and I/O delays.

Figure 10. Timing Analysis of 74161 Counter



Conclusion

MAX+PLUS development tools offer the ability to determine timing delays within MAX EPLDs. The Delay Predictor and Simulator allow the designer to analyze the delays for a given design. On the other hand, the designer may wish to hand-calculate these paths before ever entering the design, as this Data Sheet describes.

To understand timing relationships in MAX EPLDs, the designer must think of the total delay path in terms of the MicroParameters listed in the data sheet for the target EPLD. From these AC values, it is easy to determine accurate timing delay information by summing the appropriate combination of MicroParameters.

Using Expanders to Build Registered Logic in MAX EPLDs

January 1990 Application Brief 76

Introduction

MAX family EPLDs consist of high-density user-configurable devices with up to 192 macrocells. Each macrocell contains one register that can be selected for registered functions or bypassed for combinatorial functions. Since applications sometimes require more registers than are available in the macrocells, extra registers can be built with expander product terms (expanders). Expanders are unallocated product terms used to build complex combinatorial functions or latches and registers. This Application Brief explains how and when to use expander latches and registers, and describes timing considerations, specifically for an R–S latch, a transparent latch, and a synchronous register. Familiarity with MAX architecture and timing models is assumed.

Expander Product Terms

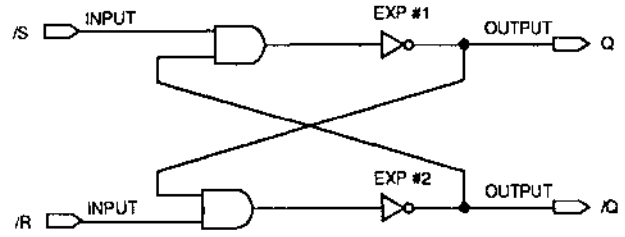
The MAX architecture provides expanders that have inverting outputs feeding the logic array. Each expander is fed by the same inputs that feed the macrocell: a global bus, macrocell feedbacks, other expanders, and I/O feedbacks. Since expanders feed themselves, they can be used to build latches and registers. Two expanders may be cross-coupled to generate an R–S latch, three may be used to build a transparent D latch, and six may be used to build a synchronous D-type flip-flop with asynchronous Preset and Clear. The expander circuits described here have been built into macrofunctions to optimize MAX performance. Each function is built with AND primitives followed by expanders to optimize fitting.

Asynchronous R-S Latch

Figure 1 shows an asynchronous R–S latch implemented with two expanders. Since expanders are product terms with inverted outputs, the latch is a NAND implementation, making the Set and Reset terms active low. If both inputs are simultaneously low, both outputs will become a logical high until one or both of the outputs go low again.

3

Figure 1. R-S Latch Designed Using Expanders



The functional output of an R–S latch is shown in Table 1. To implement the latch with active high inputs, as in a NOR latch, simply invert the inputs with NOT primitives. Asynchronous R–S latches are often used for debouncing input-switching circuits or for detecting edges in switching circuits.

Table 1. Functional Output of R-S Latch

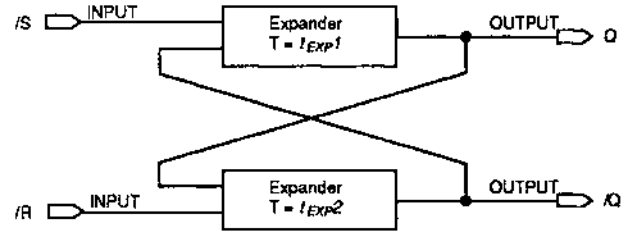
/R	/S	Q
H	H	Q
L	L	H
L	H	L
L	L	H*

* /R and /S low causes both Q and /Q to be high.

R-S Latch Timing

Each expander has a timing delay defined as t_{EXP}. When implementing latches and registers with expanders, it is important to be aware of the timing requirements to ensure that it functions properly. Consider the timing diagram of the R–S latch shown in Figure 2. The hold time is t_H. A low pulse at the S input must remain low long enough to propagate through Expander 1 and Expander 2 to latch the input. The propagation delay from the input of the latch to the output of the latch is t_{PD}. A low at the S input must travel through Expander 1 and Expander 2 before both Q and /Q become valid.

Figure 2. R-S Latch Timing Model



R-S Latch Timing Equations

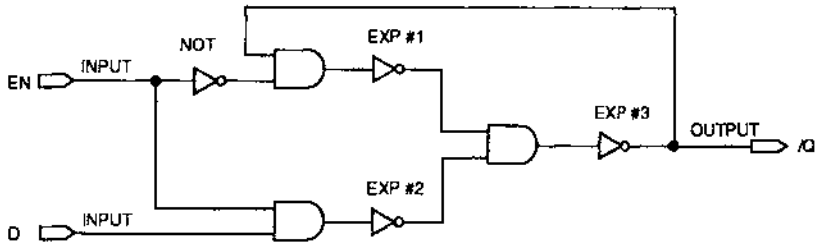
$$t_H = t_{EXP}1 + t_{EXP}2 = 2 * t_{EXP}$$

$$t_{PD} = t_{EXP}1 + t_{EXP}2 = 2 * t_{EXP}$$

Transparent Asynchronous Latch

The transparent asynchronous latch with Enable (EN) is implemented with three expander terms, as shown in Figure 3. This latch is comparable to a 74LS373, and is transparent while the EN line is a logical high. When EN becomes a logical low, the input is latched until EN goes high again. This latch is especially applicable for latching inputs from a bus. The functional output of the transparent asynchronous latch may be seen in Table 2.

Figure 3. Transparent Asynchronous Latch



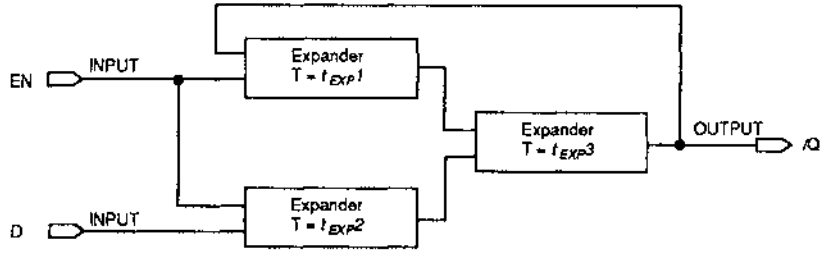
Transparent Asynchronous Latch Timing

The timing paths for the asynchronous latch are shown in Figure 4. The t_H value for this circuit is 0 ns because the paths from the D input and the EN input have equal delays to Expander 3 which latches the result. The setup time, t_{SU}, requires the D input to go through Expander 2 and Expander 3 to reach Expander 1 before the input can be latched. The delay through Expander 2 and Expander 3 to the output is t_{PD}.

Table 2. Functional Output of Transparent Latch

EN	D	Q
L	L	Q_O
L	H	Q_O
H	L	L
H	H	H

Figure 4. Asynchronous Transparent Latch Timing Model



Transparent Latch Timing Equations

$$t_H = 0$$

$$t_{SU} = t_{EXP}2 + t_{EXP}3 = 2 * t_{EXP}$$

$$t_{PD} = t_{EXP}2 + t_{EXP}3 = 2 * t_{EXP}$$

3

Synchronous Register

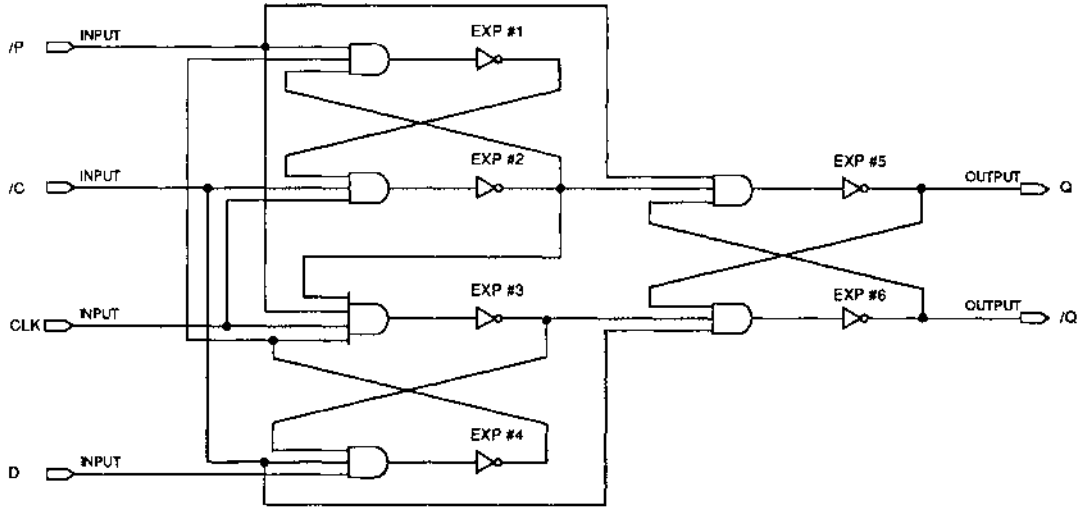
The function outputs for the synchronous register are shown in Table 3. A synchronous D register with asynchronous Preset and Clear can be built with six expanders as shown in Figure 5.

Table 3. Function Outputs for Synchronous Register

/P	/C	D	CLK	Q	/Q
H	H	L	⌐	L	H
H	H	H	⌐	H	L
H	H	X	H	Q_O	$/Q_O$
H	H	X	L	Q_O	$/Q_O$
L	H	X	X	H	L
H	L	X	X	L	H
L	L	X	X	H	H

The state at the D input is clocked into the latch with a rising edge at the clock input. Both the true and complement signals are available at the output. This output will remain latched until the next rising edge clock or until the Preset or Clear is activated with a logical low signal. The Preset and Clear can be made active high by placing NOT primitives in front of the two signals. Using expanders as registers will increase the total register count by 31 percent. For example, the EPM5192 can have up to 60 registers implemented with expanders, which gives it a total capacity of 252 registers.

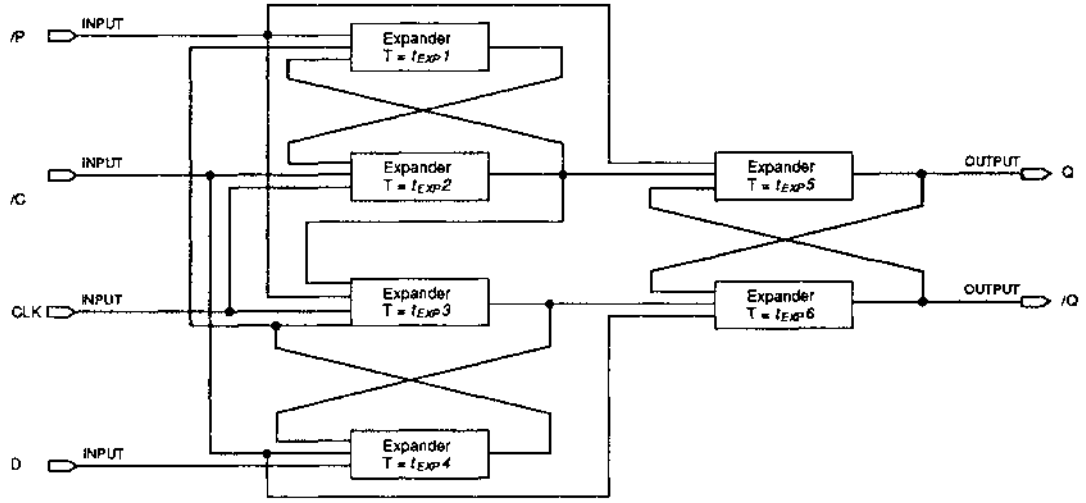
Figure 5. Synchronous Register with Preset and Clear.



Synchronous Register Timing

The timing paths for the synchronous D register are shown in Figure 6. Two different timing paths are determined by the D input, depending on whether the input is high or low. The worst-case path is described here. Before the input may be clocked, t_H is one expander delay through Expander 3. A valid path through Expander 4 and Expander 1 at the input of Expander 2 is the worst case for t_{SU}. The time required for clock to output (called t_{CO1}) has a worst-case path through Expander 3, Expander 6, and Expander 5 for both Q and /Q to produce valid outputs. Both t_{CLR} and t_{PRE} have a delay path through Expander 5 and Expander 6 to produce a valid output. The minimum time in which the register can be clocked in a pipeline register is t_{CNT}. In this case, t_{CNT} is simply t_{CO1} plus t_{SU}. Worst case for the synchronous register is five expander delays.

Figure 6. Timing Model for Synchronous Register



Synchronous Register Latch Timing Equations

t_H	=	$t_{EXP}2$					or	$t_{EXP}3$			=	t_{EXP}
t_{SU}	=	$t_{EXP}1$					or	$t_{EXP}1$	+	$t_{EXP}4$	=	2 * t_{EXP}
t_{CO1}	=	$t_{EXP}3$	+	$t_{EXP}6$	+	$t_{EXP}5$	or	$t_{EXP}2$	+	$t_{EXP}5$ + $t_{EXP}6$	=	3 * t_{EXP}
t_{CLR}	=	$t_{EXP}6$	+	$t_{EXP}5$							=	2 * t_{EXP}
t_{PRE}	=	$t_{EXP}5$	+	$t_{EXP}6$							=	2 * t_{EXP}
t_{CNT}	=	t_{CO1}	+	t_{SU}							=	5 * t_{EXP}

3

Fitting Expanders into MAX Designs

Expander latches and register should be placed strategically within a design to optimize fitting. Expanders are fed by three sources: inputs, macrocell outputs, and other expanders. Since other expanders are the only source of logic feeding expanders, complex logic should not feed the inputs of the expander macrofunctions. Instead, registers driven by complex logic should be placed in macrocells. On the other hand, registers that require logic after a register should use expander registers, since the output feeds directly into the logic within the macrocells.

Conclusion

All of the macrofunctions defined in this Application Brief are available through Altera's Electronic Bulletin Board Service (see *Electronic Bulletin Board Service* in Section 4 of this handbook). The files may also be obtained on disk from Altera representatives. The files are named as follows: the R–S latch is called **EXPRS.GDF**, the transparent latch is called **EXPLATCH.GDF**, and the synchronous register is called **EXPDFF.GDF**. These functions can also be built by copying the designs shown in this Application Brief.

Design Guidelines
for MAX EPLDs

January 1990 Application Brief 77

Introduction

MAX+PLUS uses a combination of advanced logic synthesis techniques and a heuristic fitter to efficiently map designs into MAX EPLDs. As a result, the MAX+PLUS Compiler typically synthesizes even the most complex designs in less than five minutes.

However, certain designs are more difficult to fit. These designs contain very complex combinatorial logic or require more macrocells than are available in the targeted MAX EPLD. MAX+PLUS has certain logic synthesis techniques specifically developed to quickly fit these designs. Thus, in most cases, MAX+PLUS will automatically fit even these complex designs.

Sometimes designs require subtle modifications to enable the Compiler to perform logic synthesis and obtain a fit. In these cases, the Compiler provides error messages to ensure that these design changes may be quickly recognized by the designer. This Application Brief discusses some of the synthesis techniques used to fit these complex designs and explains the most common Compiler error messages.

Logic Synthesis

The MAX+PLUS Compiler contains the Balancer module, which is specifically developed for synthesizing designs that may require too many resources. As part of the Logic Synthesis module, the Balancer is responsible for balancing the logical resources required by a design.

If a design contains too many macrocells for the specfied MAX EPLD, the Balancer can transform buried combinatorial macrocells into expander product terms (expanders). It also synthesizes designs containing too many expanders. For these designs, the Balancer transfers logical expressions implemented on expanders into macrocells. Up to three expanders may be transferred into each macrocell. By balancing resource usage, MAX+PLUS can fit designs that originally require too many of a particular logical resource. In this manner, most complex designs are automatically fitted.

Compiler Messages

If a design is too large or complex, the Compiler generates an error message specifying the problem and, if applicable, the error location. The most common Compiler error messages relating to synthesis and fitting of designs generally take one of three forms:

- **Design requires too many <#/#> macrocells**
  This message indicates that the current design contains too many macrocells for the specified MAX EPLD. The ratio **<#/#>** specifies the number of macrocells in the design divided by the number of macrocells available on the EPLD.

- **Logic too complex: for <node name> [<text>]**
  This message indicates that an equation for a particular node, in its current implementation, is too complex for a MAX EPLD. Logic synthesis may generate this message while processing very complex combinatorial (Boolean) expressions.

- **Design requires too many <#/#> expanders for <text>**
  This message indicates that the overall design is too complex and requires too many expanders, or that the design requires too many expanders for the targeted MAX EPLD. The ratio **<#/#>** specifies the number of expanders in the design divided by the number of available expanders. The **<text>** portion of the message specifies the particular macrocell or LAB that requires too many expanders. This message is typically generated by the Compiler Fitter module.

A number of variations on these messages also exist. Consult the *MAX+PLUS User Guide* for a complete listing of MAX+PLUS Compiler error messages.

Once the MAX+PLUS Compiler generates an error message, simple design modifications often deliver an efficient fit. The following sections provide suggestions to obtain or improve fitting results.

Fitting a Design with Too Many Macrocells

MAX+PLUS generates the first message when the design contains too many macrocells for the specified EPLD. At this point, the Balancer has unsuccessfully tried to transfer enough buried macrocells onto expanders. Thus, macrocells must be eliminated from the design. The following steps may help reduce the number of macrocells in a design.

- Any macrocell (MCELL) or soft (SOFT) buffers that have been manually placed into combinatorial logic may be eliminated.

- SOFT buffers from Altera-provided combinatorial TTL macrofunctions in the design may be removed. SOFT buffers are placed in complex

Table 1. Complex Combinatorial TTL Macrofunctions

Function	Macrofunctions with Soft Buffers	Macrofunctions without Soft Buffers
Arithmetic Function	8FADD, MULT4, 7488, 7482, 7483, 74181	MULT2, MULT24, 74183
Comparator	8MCOMP, 7485	74518
Converter		74184, 74185
Decoder		7446, 7447, 7448, 7449
Latch	74116	
Multiplexer		74151
Parity Checker	74180, 74280	

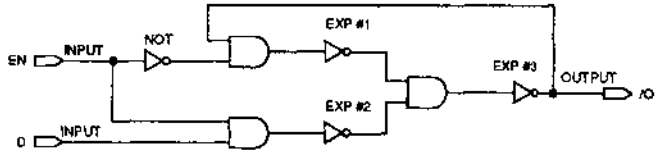
combinatorial macrofunctions to partition the logic into multiple macrocells. Normally, logic synthesis automatically removes the SOFT buffers that are not required inside these macrofunctions. In some cases, however, macrocell count may be reduced by removing one or more buffers. Table 1 lists the macrofunctions with and without SOFT buffers.

The 7485 macrofunction, for example, contains four SOFT buffers. One or more of these SOFT buffers may removed, particularly if the inputs or outputs are single product terms.

- Buried registers or latches may be moved onto expanders to reduce the macrocell count. Expanders can be cross-coupled to form a variety of register and latching functions for use as buried logic. Figure 1 shows a flow-through latch that has been placed on cross-coupled expanders to reduce macrocell count. Tips on building registers with expanders are presented in *Application Brief 76 (Use of Expanders for Registered Logic in MAX EPLDs)*. Expander-based macrofunctions are available from the Altera Electronic Bulletin Board Service or can be built directly from the schematics in *Application Brief 76*.

Figure 1. Flow-Through Latch with Cross-Coupled Expanders

If a design contains too many macrocells, buried register functions may be placed on cross-coupled expanders such as this flow-through latch.



If none of these suggestions yields the desired fit, the Altera Applications Department may be contacted for assistance at (408) 984-2805 ext. 102. As a last resort, logic may have to be removed from the design to make it fit into the target device, or the design may have to be partitioned into two MAX EPLDs. Altera's Applications Department can quickly help determine the proper path for each design.

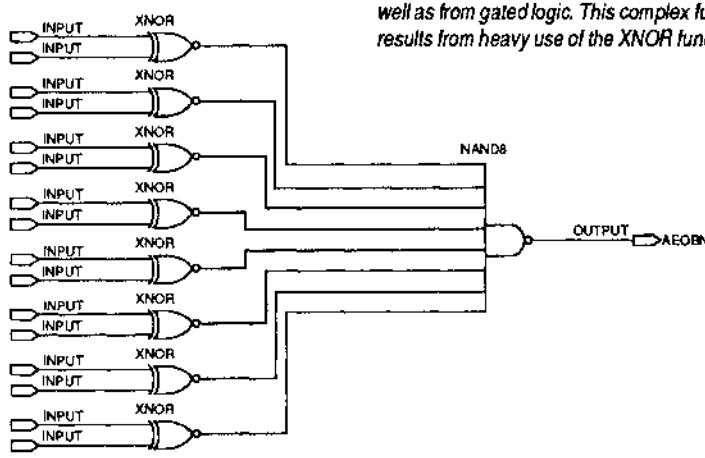
Designing Complex Combinatorial Functions

The second and third messages from the list typically result from complex combinatorial logic, which, in turn, can result from cascading several combinatorial macrofunctions, such as adders and comparators, or from heavy use of XOR functions. Figure 2 shows complex combinatorial logic resulting from heavy use of the XNOR function.

To resolve Compiler errors related to complex combinatorial logic, one or more SOFT or MCELL buffers must be inserted to separate complex combinatorial expressions. By placing a SOFT or MCELL buffer, a macrocell is used to implement a portion of a complex logic expression. The complex expression is distributed over two or more macrocells and simplified. As a result, inserting SOFT buffers in a design may affect the timing of a design. However, proper placement of SOFT buffers can significantly simplify a design, resulting in reductions in both the number of expanders and the number of macrocells required by a design.

Figure 2. Complex Combinatorial Logic Resulting from Heavy Use of XNOR

Complex combinatorial logic can result from cascading several combinatorial macrofunctions such as adders and comparators, as well as from gated logic. This complex function results from heavy use of the XNOR function.



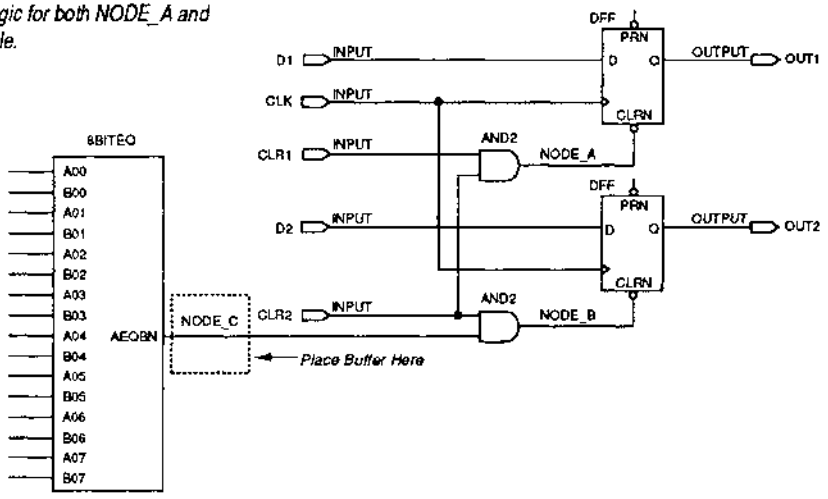
Placing SOFT and MCELL Buffers

The best location for a SOFT buffer may not be obvious. Although the second error message specifies a location, this node name indicates the output of the complex expression which is usually not the correct location for a SOFT buffer. To find the best location, the logic feeding the node must be analyzed to determine the cause of the expression's complexity.

A SOFT buffer can often be placed so that it minimizes the complexity of many combinatorial expressions. In Figure 3, for example, the Compiler has flagged NODE_A as an expression that is too complex. Since NODE_B has the same structure (and thus the same complexity) as NODE_A, a SOFT buffer could be inserted at NODE_C to simplify the complex expressions of both NODE_A and NODE_B.

Figure 3. SOFT Buffer Minimizing Multiple Complex Expressions

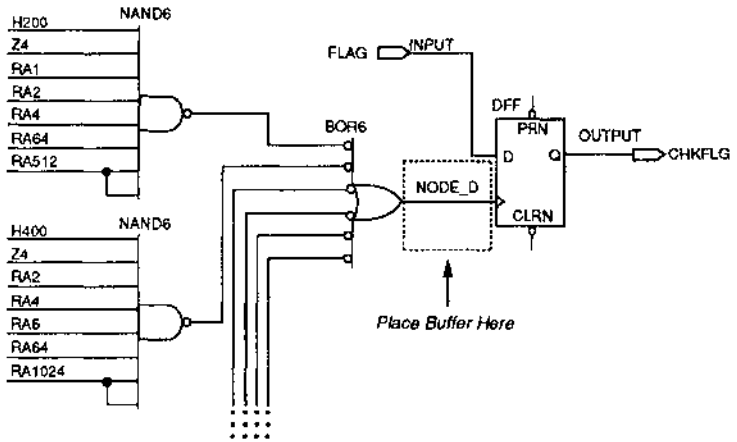
Inserting a SOFT buffer at location NODE_C reduces the complexity of the logic for both NODE_A and NODE_B in this example.



Conversely, Figure 4 shows an example where the SOFT buffer is best placed at the location NODE_D, as specified by the Compiler. The logic expression for NODE_D is an AND-OR function feeding the Clock input of a D-type register. The Clock input has a single product term dedicated to it, and thus would use a great number of expanders to implement the logic. By placing a SOFT buffer at NODE_D, the AND-OR function is efficiently implemented in a macrocell, and the output of the macrocell feeds the Clock input of the D-type register.

Figure 4. SOFT Buffer Minimizing Complex Expression That Feeds a DFF Clock Input

The control functions for flip-flops have a single dedicated product term. Therefore, complex functions on the flip-flop Clock, Preset, and Clear terms should be simplified with a SOFT buffer. In this example, the SOFT buffer should be inserted at location NODE_D.



3

The following list gives suggestions for the placement of SOFT buffers This information should be used to prevent compilation problems as wel as to resolve them. All of the points on the list are good design practices Not only can fitting problems be avoided if these suggestions are followed but often a more efficient fit will result, allowing integration of additiona logic into the targeted MAX EPLD. The following four steps should be taken in order:

- The *Design Guidelines* section of the ***MAX+PLUS Graphic Editor*** manual contains basic guidelines for designing efficiently with the MAX architecture. This section should be read before any other steps are taken.

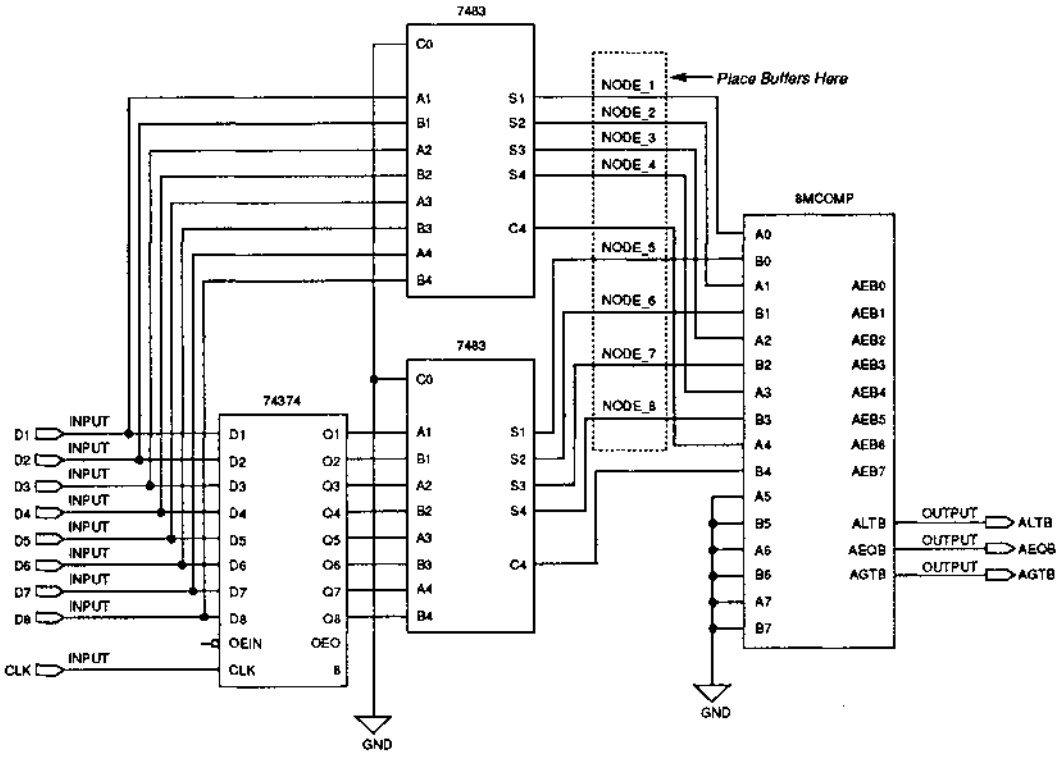
- Complex combinatorial expressions feeding flip-flop controls and tri-state buffers may be good locations for SOFT buffers. The Output Enable tri-state input and the Preset, Clear, and Clock inputs to flip-flops all have a single product term associated with them. If the expression feeding a control input is complex or feeds multiple locations, it requires expanders. Placing a SOFT buffer will reduce the number of expanders used.

- Complex combinatorial outputs of macrofunctions may be good locations for SOFT buffers, if they do not feed a flip-flop input or an I/O pin. These complex expressions may need to be isolated before being routed into another macrofunctions. Inputs to macrofunctions may also be good locations for SOFT buffers if they are being fed by complex expressions.

  Figure 5 shows the outputs of two 7483 adders feeding an 8-bit magnitude comparator. Both the adder and the comparator contain complex logic, but placing SOFT buffers at NODE_1 through NODE_8 significantly reduces the complexity of the overall function.

Figure 5. SOFT Buffers Minimizing Complex Expressions That Feed an 8-Bit Magnitude Comparator

Cascading complex combinatorial macrofunctions may sometimes require SOFT buffers. In this example, NODE_1 through NODE_8 require SOFT buffers.



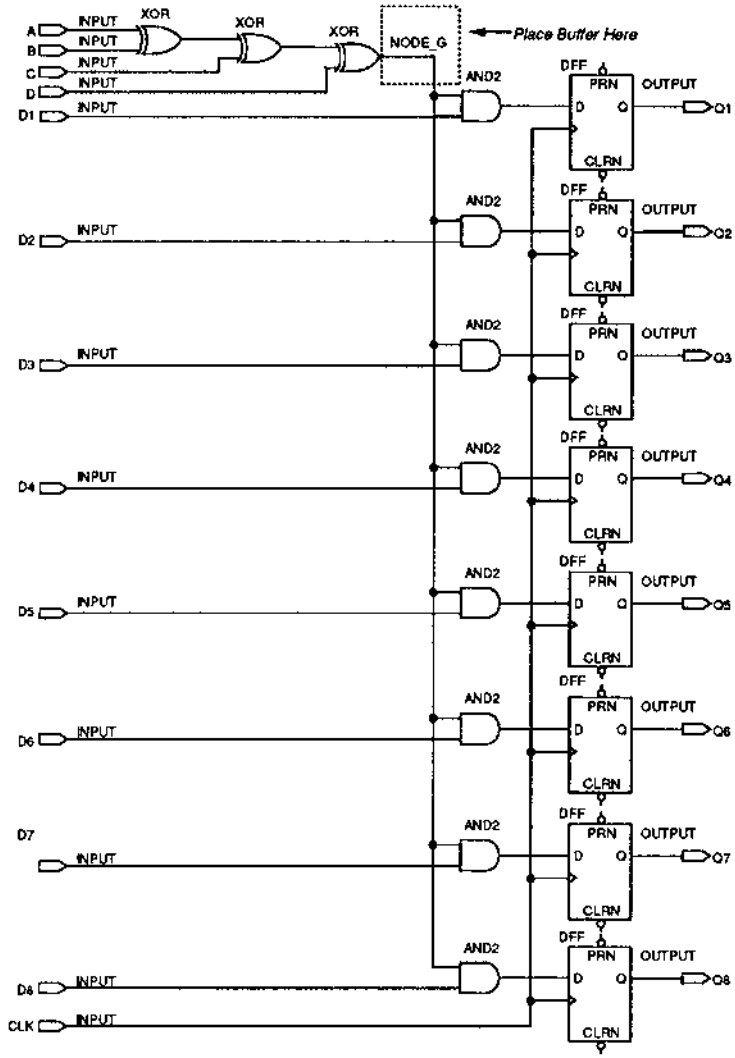
- Complex combinatorial expressions that feed many different locations may be good locations for SOFT buffers. If a complex expression feeds multiple Logic Array Blocks (LABs), the logic associated with the expression is duplicated in each LAB fed by the expression, and the number of expanders is increased. Placing a SOFT buffer at the complex combinatorial output reduces the number of expanders. The Compiler Report File contains a list of all duplicated expanders.

  Figure 6 shows a design that has 1 output in each of the 8 LABs of an EPM5128. As implemented, the design will consume 5 expanders in each LAB, a total of 40 expanders. Inserting a SOFT buffer at NODE_G, however, reduces the total expanders used to just 3.

In general, consult the ***MAX+PLUS User Guide*** when the Compiler generates an error message. If help is required for a particular error, note the *exact* wording of the error message, have a copy of the Compiler Error File ready, and call the Altera Applications Department for assistance.

Figure 6. SOFT Buffer Minimizing Complex Expression That Feeds Multiple LABs

If a logical expression feeds multiple LABs, it may be a good place for a SOFT buffer. In this example, inserting a SOFT buffer at NODE_G reduces expanders from 40 to 3.



Conclusion

This Application Brief describes common MAX+PLUS Compiler error messages and how to efficiently modify complex designs so they fit into a MAX EPLD. The guidelines, while general in scope, have been developed for MAX+PLUS versions through 2.0. Designing with the guidelines in this Application Brief will improve design efficiency for MAX EPLDs.

Optimizing Memory for MAX+PLUS

January 1990 **Application Brief 78**

Introduction

Altera's MAX+PLUS Development System contains a Compiler and a Simulator that efficiently utilize the memory resources available to MS-DOS-based systems. If a system does not contain enough memory, MAX+PLUS may report out-of-memory error messages when implementing large designs. A system that contains more than one Mbyte of RAM may also cause these kinds of error messages if the memory is not configured for optimum usage by MAX+PLUS. This Application Brief discusses the different types of memory supported by DOS systems, how to determine the memory configuration of a system, and how to configure a system to allow even the most complex designs to be processed by MAX+PLUS.

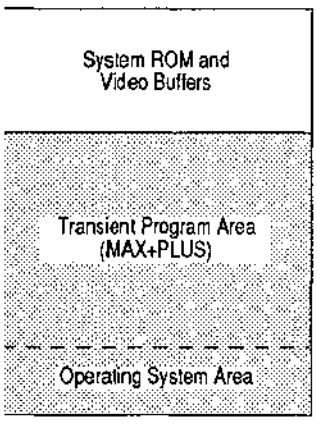
Memory Types

Computers that run DOS may have three different types of memory: conventional, expanded, and extended.

Conventional Memory

Conventional memory consists of the 1 Mbyte of memory directly addressable by the 80286 or 80386 microprocessor running in real mode (also referred to as 8086-emulation mode). Figure 1 shows how DOS allocates conventional memory.

Figure 1. Allocation of Conventional Memory in DOS Systems



Memory ranging from address 640 Kbytes up to 1 Mbyte is used for system ROM (device handlers) and expansion board buffers. DOS and programs under DOS (e.g., MAX+PLUS) occupy the lower 640 Kbytes of conventional memory. This area is further divided into an operating system area and a transient program area (TPA).

The operating system area contains the memory-resident portion of DOS and all installed device drivers specified in the **CONFIG.SYS** file. The amount of memory this area consumes depends on the version of DOS, the number of disk buffers, and the cumulative size of the installed drivers. Minimizing the size of this area will increase the memory available to the TPA, and therefore to MAX+PLUS. The size of the operating system area when running MAX+PLUS at peak efficiency is typically 60 to 70 Kbytes.

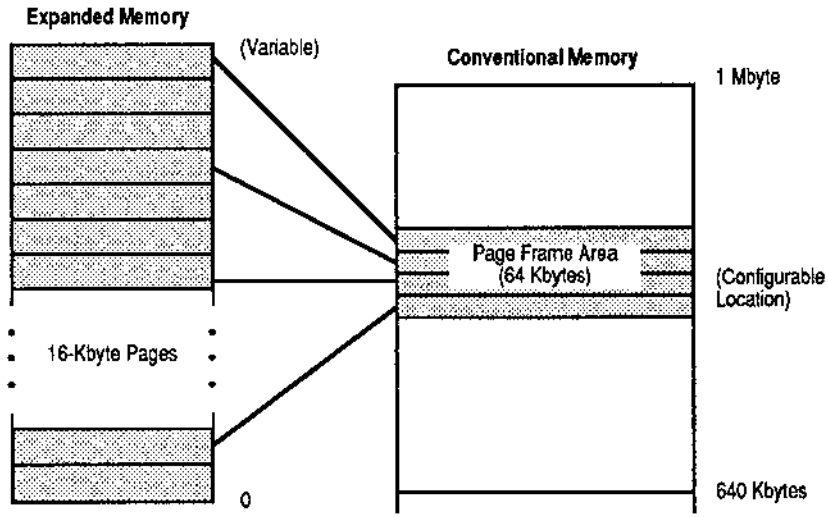
The TPA starts immediately above the operating system area and extends up to the the limit of either the installed RAM or the 640-Kbyte boundary, whichever is smaller. All programs running under DOS are loaded into this area for execution. For peak efficiency, MAX+PLUS requires 570 Kbytes of RAM for the TPA. This area may also contain programs that run simultaneously with MAX+PLUS. These programs, commonly referred to as background programs, memory resident, or TSR (terminate and stay resident), occupy memory that could otherwise be used by MAX+PLUS.

Expanded Memory

Expanded memory in 80286/80386 DOS systems provides access to additional Mbytes of memory along with the one Mbyte of conventional memory. The recommended configuration for MAX+PLUS includes a minimum of one Mbyte of expanded memory. An expanded memory subsystem consists of expanded memory hardware and a resident driver program. The hardware is either a memory expansion board plugged into an expansion slot or RAM located on the computer mother board. The driver program, called the expanded memory manager (EMM), is installed via a device directive (**device = EMM.SYS**) in the **CONFIG.SYS** file. A specific EMM, supplied by the expansion board manufacturer, must be used to ensure proper operation. Figure 2 shows how the EMM makes expanded memory available to application software.

Figure 2. Mapping of Expanded Memory

The expanded memory manager maps at least four 16-Kbyte pages at any one time into a contiguous 64-Kbyte page frame area in conventional memory.



The EMM dynamically maps at least four 16-Kbyte pages of expanded memory into a contiguous 64-Kbyte page frame area in conventional memory. The exact location of the page frame is user-configurable to avoid hardware conflicts, and is typically placed in the upper 384 Kbytes of conventional memory. Placing the page frame in the lower 640 Kbytes will decrease the size of the TPA. Since MAX+PLUS requires 570 Kbytes of TPA for peak efficiency, the page frame must be placed in the upper 384 Kbytes to obtain the optimum configuration for MAX+PLUS.

The 80286 and the 80386 systems have different ways of using expanded memory. Memory mapping capabilities built into the 80386 processor allow it to use RAM as expanded memory with no additional hardware. The 80286 does not have this capability, and requires expanded memory hardware to provide the memory mapping function. Some extended memory boards may therefore not be compatible with 80286 machines. The user should consult the board manufacturer for information on system compatibility.

Extended Memory

Extended memory consists of RAM located above one Mbyte that can be linearly addressed by 80286/80386 systems running in protected mode. DOS does not support extended memory except for ROM BIOS routines, which allow extended memory to be used as RAM Disks via the IBM VDISK software. Since extended memory cannot be used to load and execute programs, it also cannot be used to increase the usable memory for MAX+PLUS. Methods for converting extended memory to expanded memory are discussed under the heading "Expanded Memory for MAX+PLUS."

Determining Memory Configuration

The MAX+PLUS AFD (Altera Field Diagnostics) utility can determine the memory configuration of a system. AFD may be invoked by typing **afd <Enter>** from the **MAXPLUS** directory. After pressing the space bar to advance past the first screen warning message, the memory configuration is displayed, as shown in Figure 3.

3

Four items appear under **Memory Configuration**:

- **Normal** refers to the amount of conventional memory installed for use with DOS and programs under DOS.
- **Available** refers to the amount of normal memory in the TPA available to be used by MAX+PLUS.
- **Expanded** refers to the amount of expanded memory available to MAX+PLUS.
- **Extended** refers to the amount of extended memory in the system.

WARNING: If **available** is below 570 Kbytes or **expanded** is below 1024

Figure 3. AFD Memory Configuration Screen

```
ALTERA applications engineering diagnostics
Version 5.0   May 23 1989   18:13:08
Copyright (C) 1986-1989 ALTERA Corporation

IBM Personal Computer DOS Version 3.30
Computer type: AT or equivalent.
BIOS release date: '02/19/88'
Memory Configuation:
Normal     :  640k bytes.
Available  :  566k bytes.
Expanded   : 2048k bytes.
Extended   : 1084k bytes.
BIOS revision: 0.
MicroChannel not available.
system configuration: model: 252 sub-model: 1
```

Kbytes, MAX+PLUS may run out of memory. AFD also allows testing of Altera programming hardware.

Increasing Available Memory for MAX+PLUS

Available memory for MAX+PLUS may be increased by removing all background programs and unnecessary device drivers. Removing background programs frees the entire TPA for use by MAX+PLUS, while removing device drivers decreases the size of the operating system area, thereby maximizing TPA size. The following procedure describe how to obtain the optimum configuration for MAX+PLUS and how to quickly switch between system configurations:

1. Save the current configuration by copying the **AUTOEXEC.BAT** file to **ORIGINAL.BAT** and the **CONFIG.SYS** file to **ORIGINAL.SYS**.

2. Edit **AUTOEXEC.BAT** and **CONFIG.SYS** to remove the background programs and unneeded device drivers (EMM must *not* be removed).

3. Reboot the system and run AFD again to verify that available memory has increased.

4. Copy the edited **AUTOEXEC.BAT** to **MAXPLUS.BAT** and **CONFIG.SYS** to **MAXPLUS.SYS** to save the MAX+PLUS configuration after maximizing available memory.

5. Create a DOS batch file (**SWITCH.BAT**) to provide a means of switching between the MAX+PLUS configuration and the original configuration.

Figure 4 shows **SWITCH.BAT** and possible **MAX.BAT** and **MAX.SYS** configuration files.

Figure 4. SWITCH.BAT, MAX.BAT, and MAX.SYS

SWITCH.BAT

```
c:
cd \
Copy %1.bat AUTOEXEC.BAT
Copy %1.sys CONFIG.SYS
```

MAX.BAT

```
Path = c:\;c:\maxplus;c:\dos;
prompt $p$g
```

MAX.SYS

```
Device = EMM.SYS AT D000 258 ND
Files = 12
Buffers = 12
```

SWITCH.BAT is used to switch automatically to a desired configuration. By typing **switch max <Enter>**, **MAX.BAT** is copied to **AUTOEXEC.BAT** and **MAX.SYS** is copied to **CONFIG.SYS**. The system may then be rebooted to load the MAX+PLUS configuration. Typing **switch original <Enter>** followed by a reboot switches back to the original configuration.

Note that **MAX.BAT** and **MAX.SYS** are examples; actual files may differ.

ncreasing Expanded Memory for MAX+PLUS

The amount of expanded memory available to MAX+PLUS may be increased in four ways:

- By configuring MAX+PLUS to use all the system's expanded memory
- By reconfiguring extended memory to expanded memory
- By emulating expanded memory with extended memory (80386 only)
- By adding RAM to the system

MAX+PLUS is initially set to use all the expanded memory in a system when MAX+PLUS is installed. The **OE (Options : Expanded Memory)** command described in *Appendix B—MAX+PLUS Configuration File* of the *MAX+PLUS User Guide* displays the current amount of expanded memory available to MAX+PLUS in the bottom portion of the screen. If this value differs from the amount of expanded memory in the system reported by AFD, the value may be increased to make all expanded memory in the system available to MAX+PLUS.

3

If the system contains extended memory, it may be possible to reconfigure this memory to be expanded. Some plug-in memory expansion cards or system mother boards allow the memory to be divided between conventional, expanded, and extended memory. Typically, the memory on plug-in is configured via on-card dip switches or a setup program supplied with the card. The system mother board is usually configured with the setup program supplied with the computer. If the system contains extended memory, the card or computer documentation may provide information about configurability.

If the system has extended memory that is not directly configurable, it may be possible to emulate expanded memory with an appropriate device driver. The memory-mapping capabilities of the 80386 allow generic device drivers to convert extended memory to expanded memory. Two such drivers for 80386 machines are QEMM from Quarter Deck and 386MAX from Qualitas. Because of the hardware-dependent interface to expanded memory, however, generic drivers for 80286 machines are not available. The user should consult the computer manufacturer for an appropriate device driver for an 80286 machine with expanded memory capability on the system mother board.

If the system does not contain memory that can be used to provide one Mbyte of expanded memory, the user must add RAM in one of the forms described above to the system, and the associated EMM must be installed to add expanded memory to the system.

After increasing expanded memory in a system, AFD should be run again to verify that expanded memory has been installed properly. Selecting the **OE** (**Options : Expanded memory**) command then allows MAX+PLUS access to any newly installed expanded memory.

Conclusion

To avoid out-of-memory messages when running MAX+PLUS, a system must be configured to provide 570 Kbytes of available program memory and 1 Mbyte of expanded memory. Batch files, such as the ones shown in this Application Brief, allow quick loading of this configuration for operating MAX+PLUS. If out of memory error messages persist, Altera's Applications Department should be contacted at (408) 984-2805 ext. 102 for assistance.

Simulating Internal Nodes

October 1989 Application Brief 79

Introduction

The MAX+PLUS Simulator and Waveform Editor allow graphical validation of logic, both at the device pins and at nodes internal to the device. This Application Brief discusses a variety of methods used to simulate internal nodes for MAX+PLUS 2.0 software. First, it describes the four ways of naming internal nodes. Next, it shows how to easily locate nodes within state machines as well as combinatorial nodes. Finally, it describes how to incorporate nodes into an input file to perform a quick and thorough simulation after the appropriate nodes have been identified.

Logic Synthesis Affects Simulation

During compilation, the advanced algorithms in MAX+PLUS minimize logic and perform logic synthesis to fit designs into MAX architecture. This process retains the functionality of the design, but it may eliminate some of the original nodes that defined the circuit. As a result, not all of the original nodes can be simulated. The nodes that can be simulated are device inputs and outputs, MCELLs, and all of the nodes on registers and latches. SOFT buffers and TRI buffers can also be simulated if they are not eliminated during logic synthesis.

Before simulating a MAX design, it is helpful to generate a History File containing a list of all nodes that can be simulated:

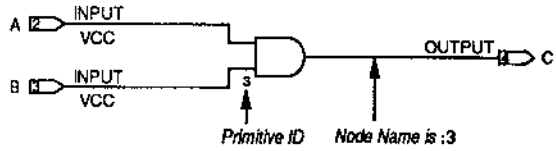
1. Invoke the Simulator and load the netlist for the design.
2. Select **FHC** (**File : History : Create**) and type the name of the file. Press **<Enter>** to create the History File (with extension **.HST**).
3. Select **CNL** (**Control : Node : List**) * and press **<Enter>**. The wildcard character (*) is used to list all nodes in the design that can be simulated.
4. Quit the Simulator and MAX+PLUS.
5. Print the file with the extension **.HST**.

Nodes That Can Be Simulated

All nodes in the History File are assigned a name and defined to be of type IN, I/O, or OUT. Nodes defined as IN correspond to inputs to the chip from device pins; they have the same node names as the pins they represent. Nodes defined as I/O are the I/O pins on the chip; they have the same name as their corresponding input, output, or bidirectional pin. Nodes defined as OUT are internal to the device; they have no direct pin connection. Internal node names specify the location of a node within a design and refer to primitives only. The section titled *MAX+PLUS Primitives* in the *MAX+PLUS Graphic Editor* manual shows the complete list of primitives.

All primitives, except TITLE blocks and OUTPUT primitives, have a single output. Figure 1 shows a design with a single internal node fed by the AND3 primitive, and lists all nodes in this design that can be simulated.

Figure 1. Nodes That Can Be Simulated in a Simple Design



Type	Name
I/O	C
OUT	:3
IN	A
IN	B

The MAX+PLUS Graphic Editor assigns a unique identification number to every symbol entered into a schematic (the number appears in the lower left-hand corner of the symbol). The only internal node in Figure 1 is :3, defined as type OUT. This node corresponds to the output of the AND primitive with symbol ID 3.

Node Names in Levels of Hierarchy

Complex designs with multiple hierarchical levels use node names incorporating the hierarchical path of the node to define the location of an internal node. Internal node names may take four forms, and the user may use more than one of these forms for a particular internal node. However, the History File lists each node only once using the highest priority node name found. The four kinds of node names, listed from highest to lowest priority, are as follows:

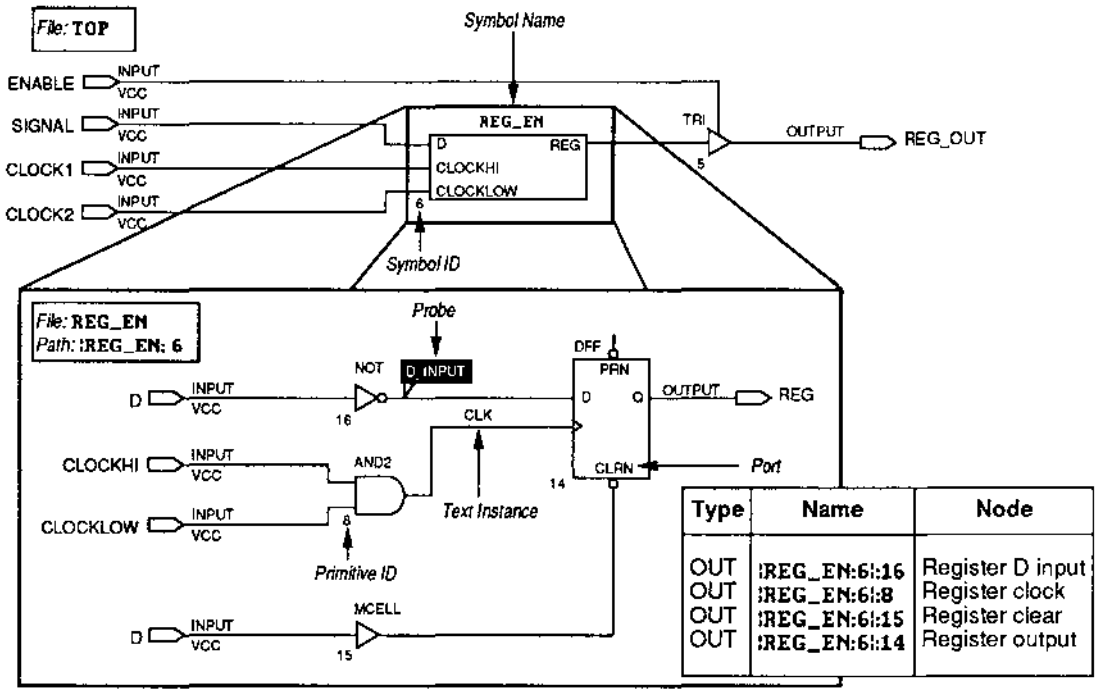
Priority 1: Probes
Priority 2: Text instances
Priority 3: Ports
Priority 4: Hierarchical pathnames

All internal nodes have hierarchical pathnames and most may also be specified with a port name. Text instances and probes may be assigned by

the user to further simplify the finding and simulating of nodes. For an overview of internal node name syntax, see "Hierarchical Node Names & Probes" in the *Simulator Reference* section of the *MAX+PLUS Simulator* manual.

Figure 2 shows a hierarchical design example with two levels. The title of each schematic file in a hierarchical design indicates which symbol it defines. This design has four accessible internal nodes that feed the CLK, CLRN, and D inputs to the register, and the register output.

Figure 2. Node Names in a Hierarchical Design



Priority 4: Hierarchical Pathnames

All internal nodes can be identified by a hierarchical pathname, which traces the path of the node through multiple levels of hierarchy by using the following format:

```
|<symbol name>:<symbol ID>...|:<primitive ID>
```

The pipe (|) indicates that the symbol name following it is the name of a file in a lower level of the hierarchy. The colon (:) precedes the symbol ID, which provides further distinction between common symbols in the same level of hierarchy. Thus, the hierarchical pathname for the clock of the

register in Figure 2 is **!REG_EN:6!:8**. This name defines the AND2 primitive (with primitive ID **8**) driving the CLK input, which is inside the symbol REG_EN (with symbol ID **6**). Designs with multiple levels of hierarchy follow this pattern, with a pipe (!) separating each level of hierarchy.

Priority 3: Ports

Internal nodes connected to primitives that convert directly into hard nodes (i.e., nodes that will not be eliminated by logic synthesis) use ports as their internal node names. A port is an extension to the hierarchical pathname that defines the specific input or output of the primitive. Ports are identified by a period (.) and a port name following a hierarchical pathname. For example, the D input to the register in Figure 2 is **!REG_EN:6!:14.D**. Table 1 shows all port names associated with the register in Figure 2. A list of all primitives with ports may be found under the heading "Primitives" in *AHDL Elements* in the *MAX+PLUS AHDL* manual.

Table 1. Port Names for Register in Figure 2

Type	Name	Node
OUT	:5.OE	Tri-state control
OUT	!REG_EN:6!:14.CLK	Register clock
OUT	!REG_EN:6!:14.CLRN	Register clear
OUT	!REG_EN:6!:14.D	Register D input
OUT	!REG_EN:6!:14.Q	Register output
OUT	!REG_EN:6!:15.IN	MCELL input
OUT	!REG_EN:6!:15.OUT	MCELL output

Priority 2: Text Instances

Hierarchical pathnames can be enhanced by using text instances. Text instances are created in a schematic by inserting text above a node, causing the primitive ID to be replaced with the inserted text. For example, the node feeding the clock in Figure 2 has the text **CLK** inserted above it. This text replaces the primitive ID **8**, changing the name of this internal node to **!REG_EN:6!CLK**.

Figure 3. TDF Using Text Instances

```
TITLE "4-Bit Counter With A 74161";

FUNCTION
74161(A,B,C,D,LDn,ENP,ENT,CLRn,CLK)
    RETURNS (QA,QB,QC,QD,RCO);

DESIGN IS 4_bit

SUBDESIGN 4_bit
( ... )

VARIABLE
    counter : 74161;

BEGIN
...
END;
```

Text instances are automatically created in AHDL Text Design Files (TDFs) when variables are assigned to nodes that can be simulated. Figure 3 shows part of a TDF with the variable **counter** assigned to a 74161 counter in the **VARIABLE** section. The default node name of the first register's output is **!74161:33!QA**. The node name after making the variable assignment is **!COUNTER!QA**.

Priority 1: Probes

Internal nodes can be fully customized by using probes. Probes can be entered into schematics at any level of hierarchy on primitives only. The probe connects to the output of the primitive it is placed on. The following steps describe how to place a probe on the appropriate primitive to connect to the clock node of the register in Figure 2.

1. Go to **REG_EN**.
2. Position the cursor on the AND2 and type **SPE** (**Symbol : Probe : Enter**).
3. Type a name for the probe and press **<Enter>**.

A pointer with the assigned probe name is attached to the primitive output. During compilation, the probe name is put into the Simulator node list in place of the hierarchical pathname. For example, in Figure 2 a probe called **D_INPUT** placed on the NOT gate feeding the D input to the register will produce the node name **D_INPUT** in the Simulator node list, thus eliminating all the hierarchy information for the node name.

MAX+PLUS allows the user to name internal nodes in a variety of ways. The Simulator lists each node only once, using the simplest name for the node. If the node has a probe attached to it, the node is listed under the probe name. If not, the node is listed under a text instance, or under a port name if no text instance exists. Finally, if no other higher priority name exists, the node is listed under its hierarchical pathname. Table 2 shows the four ways of naming the D input to the register in Figure 2.

Table 2. Four Possible Names for D Input

Internal Node Name	Type of Name	Description
D_INPUT	Probe	Probe named D_INPUT placed on NOT gate
:REG_EN:6:D_INPUT	Text instance	Node feeding D input labeled D_INPUT
:REG_EN:6:14.D	Port	D input port
:REG_EN:6:16	Hierarchy pathname	Hierarchical pathname of NOT primitive feeding D input

3

Simulating State Machines

Since state machines often control an entire design, simulation is especially useful in state machine debug. Four items in state machines may need to be simulated: inputs, decoded outputs, state register bits, and present states.

State Machine Inputs

State machine input lines feed combinatorial logic, which, in turn, feeds the D inputs to state register bits. However, state machine inputs can only be simulated when they are fed by device input pins or by hard nodes within a design. To allow simulation of state machine inputs when these inputs are fed by combinatorial logic, MCELLs should be placed in front of all inputs to a state machine. MCELLs are hard nodes; therefore, their outputs (the inputs to the state machine) can be simulated. Note, however, that inserting MCELLs affects the timing of the state machine, and MCELLS should therefore be removed after simulation.

To make MCELLS more accessible during simulation, probes with the same names as the inputs to the state machine may be attached to MCELLs in the Graphic Editor.

Decoded Outputs

Decoded outputs are also combinatorial, and therefore may not allow simulation. If decoded outputs do not feed hard nodes, such as MCELLs or output pins, then they can not be simulated. As with state machine inputs, decoded outputs may simulated if an MCELL with a probe on it is placed after the decoded outputs.

State Register Bits

State register bits are assigned names in a state machine declaration in the **VARIABLE** section of AHDL files. The syntax of this declaration is:

```
<name> : MACHINE OF BITS ( <state registers> )
         WITH STATES ( <state assignments> )
```

The Simulator extracts the state register names from the TDF to define the register bits. A state machine in a lower level of a hierarchical design incorporates the state machine name into the node name to distinguish it from state machines at the same level. Figure 4 shows a TDF with a state machine. If this machine were loaded into the top level of a schematic design, the state register outputs for this design would be **!4_STATE:n!Q0** and **!4_STATE:n!Q1** (where **:n** is the ID for the symbol representing the state machine).

Figure 4. Text Design File with State Machine

```
TITLE "A Simple State Machine";

DESIGN IS 4_state;

SUBDESIGN 4_state
(
     clk, BUS[3..0]          : INPUT;
     CNT[3..0], decode       : OUTPUT;
)

VARIABLE
     state : MACHINE OF BITS (Q[1..0])
                  WITH STATES ( ONE, TWO, THREE, FOUR );

BEGIN
     state.clk = clk;
     CNT[ ] = Q[ ];

     CASE (state) IS
          WHEN ONE =>
               state = TWO;
               decode = (BUS[] == h"A");
          WHEN TWO =>
               state = THREE;
          WHEN THREE =>
               state = FOUR;
          WHEN FOUR =>
               state = ONE;
               decode = (BUS[] == h"7");
     END CASE;
END;
```

AHDL state machines provide automatic state assignments. When this feature is used, it is more useful to monitor the state name rather than the individual state registers. The state name can be monitored easily in the Waveform Editor by defining a bus with the same name as the state machine. The waveforms will have the associated state names displayed in a text format inside the bus. Figure 5 shows the waveforms for the TDF in Figure 4.

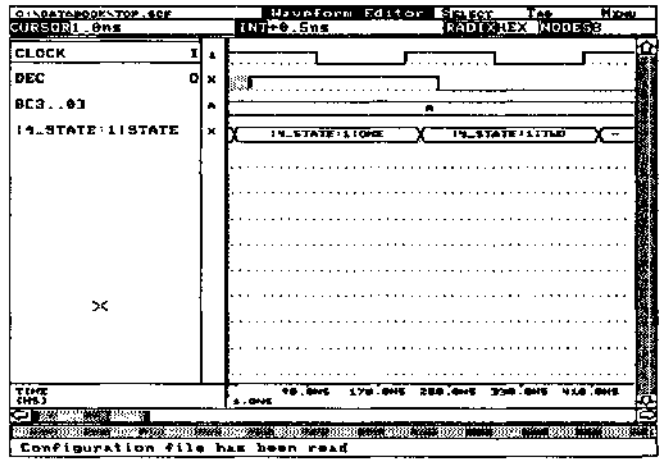
Present States

There are three methods of setting up the Waveform Editor to show the present state of a state machine. The easiest method is to use the default channel file which automatically places the state machine bus in the Simulator Channel File (**.SCF**). Another method is to enter the state machine into an ASCII Vector File (**.VEC**), which requires putting the state machine name (including its hierarchy information) into the **OUTPUTS** section. The third method, entering the state machine into a Simulator Channel File, requires the following two steps:

1. Enter the state register bits individually into the Waveform Editor, with each node corresponding to a state register bit.

2. Group the nodes together and assign the group name the same name as the state machine.

Figure 5. State Names in the Waveform Editor



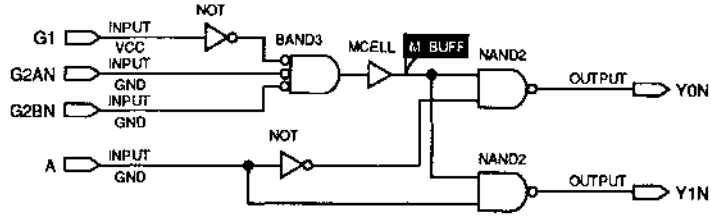
Whichever method is used, the appropriate state information will appear in the Simulator Channel File after simulation.

Simulating Inaccessible Combinatorial Nodes

Some combinatorial nodes cannot be directly simulated. To simulate buried combinatorial logic, feed the node into an MCELL primitive. Place a probe on the MCELL and name it. Recompile the design to generate an updated Simulator Netlist File. The MCELL should be removed once the node is shown to be functionally correct.

Figure 6 shows a combinatorial circuit for a decoder with three enable lines called G1, G2AN, and G2BN. To activate the outputs, all three of the enable lines feeding the BAND3 gate must be active. To simulate the combinatorial logic in this design, an MCELL may be placed in the circuit between the AND3 primitive and the BAND3 primitive. Be aware that the timing of the circuit will change. This technique should be used to verify functionality, not timing.

Figure 6. Placing an MCELL for Simulation of Combinatorial Logic



Finding Nodes

Once all nodes that can be simulated have been identified, it may be useful to find their locations in the schematic or text file. The **LNF (Line : Node : Find)** command in both the Graphic Editor and the Text Editor helps locate nodes that have hierarchical node names:

1. Type **LNF (Line : Node : Find)**.
2. Type the internal node name (for example, **:REG_EN:6:11** in Figure 2) and press **<Enter>**.

The schematic or file containing the node is then loaded and the node is highlighted.

A quick way to find a node is to use the **LNF (Line : Node : Find)** within the schematic containing the node. For example, if **LNF (Line : Node : Find)** is selected while in **REG_EN**, the prompt line displays **:REG_EN:6**. Press **<End>** to move the cursor to the end of the line, type **:8**, and press **<Enter>** to highlight the AND2 gate. If a node is identified with a probe, only the probe name must be entered.

Using Internal Nodes in Simulation

After all internal nodes that need to be simulated have been identified, they can quickly be placed into a Simulator Channel File or Vector File for simulation. Internal nodes to be simulated are added to **OUTPUTS** section of a Vector File. This can be done easily by importing the names from the node list in the History File. Figure 7 shows an example of a Vector File created from a History File node list with I/O pins and internal nodes grouped together to form buses.

The Waveform Editor automatically creates a default channel file which contains the device I/O pins, internal nodes with probes, and nodes associated with state machines. Additional nodes that need to be simulated can be appended to this channel file.

3

Figure 7. Creating Groups in a Vector File from an Internal Node List

```
--- NODE LIST ---

Type        Name
----        ----
IN          CLOCK
IN          DIR
IN          S
IN          D0
IN          D1
IN          D2
IN          D3
I/O         B0
I/O         B1
I/O         B2
I/O         B3
OUT         :5
OUT         :6
OUT         :7
OUT         :8
OUT         |21MUX:9|:5
OUT         |21MUX:12|:5
OUT         |21MUX:10|:5
OUT         |21MUX:11|:5
```

```
--- VECTOR FILE ---

GROUP CREATE B_BUS   = B3 B2 B1 B0;
GROUP CREATE D_BUS   = D3 D2 D1 D0;

GROUP CREATE INT_BUS = |21MUX:9|:5
                       |21MUX:12|:5
                       |21MUX:10|:5
                       |21MUX:11|:5;

OUTPUTS B_BUS INT_BUS;
```

Conclusion

The MAX+PLUS Simulator (MAX+PLUS version 2.0) incorporates many new features. With the addition of an integrated text language (AHDL), all nodes that can be simulated may be easily identified. Additionally, both AHDL Text Design Files and Graphic Design Files use a parallel method to identify internal nodes, making simulation quick and simple.

Choosing the Right EPLD
for a State Machine Application

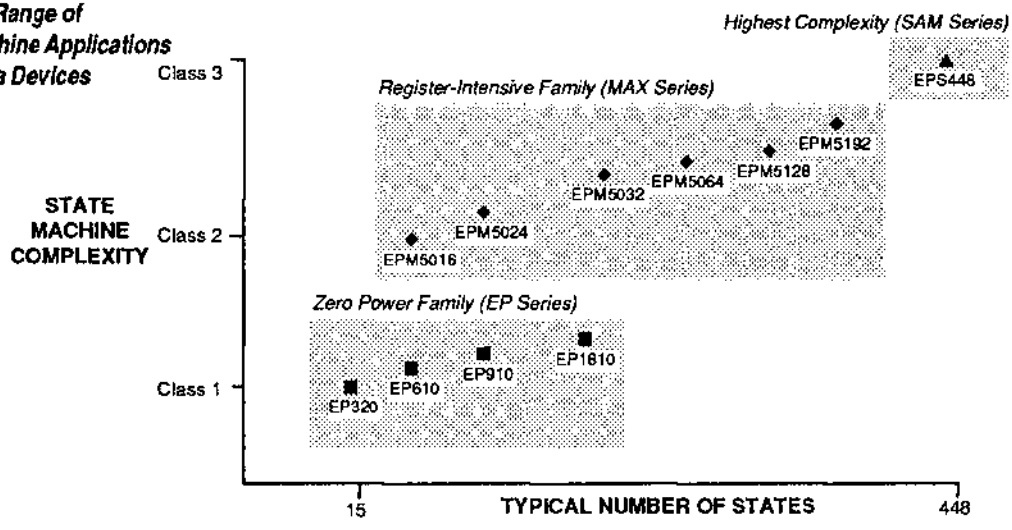
January 1990 **Application Brief 80**

Introduction

A common application for Programmable Logic Devices (PLDs) is the implementation of state machines. Compared to TTL-implemented state machines, PLDs provide many benefits, including significant integration density and ease of design. Altera Erasable Programmable Logic Devices (EPLDs), in particular, are well-suited for state machines since their architectures combine combinatorial logic with registers and since they are supported by a choice of high-level state machine input languages. Details on state machine designs appear in the *State Machine and Microsequencer Applications* section of Altera's ***Applications Handbook*** and in the ***MAX+PLUS AHDL*** manual.

Figure 1 lists the EPLDs available for implementing state machines.

Figure 1. Range of State Machine Applications with Altera Devices



These devices are divided into three families: (1) the EP series (EP320, EP610, EP910, and EP1810); (2) the MAX series, consisting of parts using a Multiple Array MatriX architecture; and (3) the SAM series, i.e., the Stand-Alone Microsequencer family. Faced with all these alternatives, the designer must ask, "Which device best suits the needs of my state machine?"

This Application Brief helps answer this question by examining some of the pros and cons of implementing state machine designs in the various EPLDs. The decision, of course, is influenced not only by important system-level

factors such as cost, power, operating performance, and board area requirements/restrictions, but also by state machine complexity. Each of the EPLD families is suited for a specific class of state machine designs.

Designing State Machines

All three families support the State Machine File (SMF) format that consists of state definitions followed by the appropriate transition equations, as shown in the sample file in Figure 2. SMFs targeted for EP-series devices are submitted directly to the A+PLUS Design Processor (ADP); SMFs for designs targeted for the SAM EPS448 device are submitted directly to the SAM+PLUS Design Pro-cessor (SDP). Design files targeted for MAX devices require that the SMF be converted into a Compiler Netlist File (CNF) with the MAX+PLUS SMF2CNF utility.

Figure 2. 68020 Bus Arbiter State Machine File

```
PART: EP320

% Pin Assignments (an option) are
made by the designer %
INPUTS: REQUEST@2 ACK CLK
OUTPUTS: GRANT@12 TRISTATE@13 OS0
OS1 OS2 OS3 OS4 OS5
MACHINE: BUSARBITER
CLOCK: CLK
% STATES gives the output value
mapping %
STATES: [GRANT TRISTATE OS0 OS1 OS2
OS3 OS4 OS5]
  S0 [0 0 1 0 0 0 0 0]
  S1 [1 1 0 1 0 0 0 0]
  S2 [1 1 0 0 1 0 0 0]
  S3 [1 1 0 0 0 1 0 0]
  S4 [1 1 0 0 0 0 1 0]
  S5 [0 1 0 0 0 0 0 1]
  S6 [0 1 0 0 0 0 0 0]
% Transition Specifications follow%
S0:
  IF REQUEST*/ACK THEN S1
  IF ACK THEN S5
  S0
S1:
  S2
S2:
  IF /REQUEST*/ACK  + ACK THEN S6
  S2
S3:
  IF /REQUEST THEN S6
  IF REQUEST*/ACK THEN S2
  S3
S4:
  S3
S5:
  IF REQUEST THEN S4
  IF /REQUEST*/ACK THEN S0
  S5
S6:
  S5
END0
```

The Altera Hardware Description Language (AHDL), which is the most powerful and efficient entry method, supports the full range of MAX EPLDs and eliminates the conversion process. AHDL language consists of a Text Design File (TDF) that is directly submitted to the MAX+PLUS Compiler.

Designs targeted for a MAX device should be entered with the advanced AHDL, which supports features such as hierarchical designs and automatic state bit assignments. However, in cases where it is unclear which device—i.e., which device family—is most suitable for a state machine design, the design may be entered in the traditional SMF format so that it may be submitted to each compiler to determine the most appropriate device.

Choosing the Right EPLD

Which device is most appropriate for a design? This decision is guided by three factors: power, input/output pin count, and state machine complexity.

Power For low-power (e.g., battery-powered) applications, the EP series offers a zero-power mode resulting in current consumption of 10 to 20 microamps when the circuit is static and a few milliamps when it is active.

Input and Output Pin Count If the number of outputs exceeds the number of I/O pins on a device, the device can be eliminated as a possible EPLD. For instance, if a design requires low power and has 14 outputs, the EP320 should be disregarded. Also, the number of inputs into the state machine indicates whether a part is suitable for a design. For example, the EP1810, EPB1400, and MAX devices, which support dual-feedback, yield the best device utilization when a design is input-intensive, i.e., when it has more than 12 inputs. (Dual feedback is the ability to bury a state register, feed the register output back to the logic array, and use the I/O pin as input.)

State Machine Complexity The complexity of a state machine also influences the EPLD choice. For this discussion, state machines have been split into three classes. Figure 1 shows class type vs. state machine size for each of the EPLDs. Class 1 includes "simple" state machines, which usually have no more than two **If** statements per state and two or three product terms per transition statement. Class 2 machines have many possible next states and multiple product terms in the transition statements. Class 3 machines are large machines with many multi-way branch locations and/or complex transition statements. Altera EPLDs provide solutions for each of these classes while maintaining a common design language. See Table 1.

Table 1. Altera EPLD Density and Performance Range for State Machine Implementation

Device	Macrocells	Inputs	I/O	Typical ICC Active/Standby	Maximum State Machine Frequency†
EP SERIES					
EP320-1	8	10	8	18 mA / 10 µA	28.6 MHz
EP610-25	16	4	16	32 mA / 20 µA	40.0 MHz
EP910-30	24	12	24	45 mA / 20 µA	33.3 MHz
EP1810-35	48	16	48	120 mA / 35 µA	28.6 MHz
MAX SERIES					
EPM5016-1	16	8	8	85 mA	100.0 MHz
EPM5024	24	8	12		
EPM5032-1	32	8	16	125 mA	76.9 MHz
EPM5064-1	64	8	28	95 mA	50.0 MHz
EPM5127	128	8	28		
EPM5128-1	128	8	52	155	50.0 MHz
EPM5192	192	8	68		
SAM SERIES					
EPS448-30	N/A	10	16	90 mA	30.0 MHz

†Includes time to calculate conditional branches.

3

EP Series

EP family devices are ideal for Class 1 state machines. These devices implement the standard sum-of-products notation by providing a programmable AND/fixed OR architecture. Eight product terms feed a programmable register. This architecture makes the EP series natural for implementing state machines because the dedicated register in the macrocell acts as the state register. Feedback from the register feeds the logic array, where the eight dedicated product terms in the macrocell determine the next state.

To further understand why the EP series is best suited to Class 1 state machines, it is important to understand how state transition equations are prioritized and minimized when state machines are mapped into EPLDs. The following example shows a state and its potential next states:

```
State0:  If Input1 * Input2 then State1
         If Input3 * /Input4 then State2
         Else State3
```

To go to **State2**, (**Input1*Input2**) must be false and (**Input3*/Input4**) must be true. To get to **State3**, (**Input1*Input2**) and (**Input3*/Input4**) must both be false.

Because a branch decision is dependent on all previous branch possibilities, product-term count requirements can grow beyond the eight dedicated product terms in a particular macrocell. Three major factors can contribute to this growth. (1) First, if a transition statement is complex (i.e., contains multiple product terms) and particularly if this complex transition equation is placed in the first **If** statement in a state, the final logic may get very complex. (2) Second, if the number of possible destinations in a particular state is large, the required product-term count will increase. (3) Finally, if state variable definitions are chosen poorly (see the *Applications Handbook*, *Introduction to State Machine Design* under "Selecting State Assignments"), product-term counts can increase beyond the limitations of the EPLD. If more than eight product terms are required for any particular state register equation, the state machine will not directly map into the EPLD. To resolve these problems and make the design fit, the following steps are recommended.

(1) To solve the first problem, additional macrocells could be inserted to provide additional product terms. However, this solution is not practical if speed is vital, because every additional macrocell adds delay.

(2) Large product-term counts resulting from a large number of destinations may be more evenly distributed by manipulating the state machine. For example, transitions to "dummy" states placed half-way through the **If** statements can be inserted. The dummy state would then, in turn, contain the remaining possible next states. This solution, shown in Figure 3,

effectively reduces the number of possible destinations from any one state. This approach, however, will slow the state machine and result in "dead states" and wasted clock cycles.

(3) To counteract the third potential problem (poor state definitions), state variables can be manipulated to minimize the number of state-variable transitions. The goal is to reduce the number of 1s among the state definitions if D-type flip-flops are used as state registers, or to reduce the number of 0-to-1 or 1-to-0 transitions if T-type flip-flops are used (see Altera's *Applications Handbook, Selecting State Assignments*).

Figure 3. Inserting States to Reduce Product Term Counts

Original State Machine

```
S0:   If ( A ) then SA
      If ( B ) then SB
      If ( C ) then SC
      If ( D ) then SD
      If ( E ) then SE
      If ( F ) then SF
      Else SG
```

After Partition

```
S0:       If ( A ) then SA
          If ( B ) then SB
          If ( C ) then SC
          Else Dummy
Dummy:    If ( D ) then SD
          If ( E ) then SE
          If ( F ) then SF
          Else SG
```

Because of these issues, the EP series is best suited for Class 1 machines. However, these devices are not necessarily limited to small state machines. As a basic guideline, the EP320 can typically implement a state machine with about 15 states. The EP1810, on the other hand, can implement a much larger state machine and leave plenty of the device free for implementing other logic. If the states each have one or two **If** statements, the EP1810 can typically implement a machine with at least 50 states, operating at 28.6 MHz.

MAX Series

MAX family devices are typically used to implement Class 2 state machines. These devices avoid the limitations of fixed product-term macrocells by providing expander product terms (expanders). Expanders are freely allocatable product terms that can be used to supplement the product terms within the macrocells for very complex combinatorial functions, allowing more than 8 product terms in any particular macrocell. In fact, the EPM5032 (in a 28-pin package) can accommodate as many as 66 product terms in one expression. Therefore, even when product-term counts grow very large, the state machine will still directly fit into the MAX architecture.

As stated earlier, designs targeted for a MAX device should be entered with AHDL, which supports many more features than the traditional State Machine Entry Language. However, the SMF2CNF utility is provided to allow migration of EP-series designs into MAX devices.

Figure 4 shows part of an AHDL state machine with transitions from one of the many states in a Buffer Memory Controller. The state, **RAS2**, has complex transition statements and many possible next states. The entire state machine was compiled and a Report File containing device utilization information and minimized equations was generated.

Figure 4. AHDL State Machine Transition

```
WHEN ras2 =>
  IF !bmreq & (!nxtmst1 # nxtmst0 # !csbm)
   & (nxtmst1 # nxtmst0)
   & (!mstchg # !dirchg
      # (!nxtmst1 & nxtmst0 & nbpcmp)
      # (nxtmst1 & !nxtmst0 & oppcmp)
      # (nxtmst1 & nxtmst0 & dmapcmp)) THEN
         buf_mem_control =  newras;
  ELSIF !bmreq & nxtmst1 & !nxtmst0 & csbm THEN
         buf_mem_control =  op3;
  ELSIF !bmreq & !nxtmst1 & !nxtmst0 THEN
         buf_mem_control =  rfsh1;
  ELSIF !bmreq & (!nxtmst1 # nxtmst0 # !csbm) & bmwr
   & (nxtmst1  # nxtmst0)  & mstchg & dirchg
   & (nxtmst1  # !nxtmst0  # !nbpcmp)
   & (!nxtmst1 # nxtmst0   # !oppcmp)
   & (!nxtmst1 # !nxtmst0  # !dmapcmp) THEN
         buf_mem_control =  read1;
  ELSIF !bmreq & (!nxtmst1 # nxtmst0 # !csbm) & !bmwr
   & (nxtmst1  # nxtmst0)  & mstchg & dirchg
   & (nxtmst1  # !nxtmst0  # !nbpcmp)
   & (!nxtmst1 # nxtmst0   # !oppcmp)
   & (!nxtmst1 # !nxtmst0  # !dmapcmp) THEN
         buf_mem_control =  write1;
  ELSE bmreq THEN
         buf_mem_control =  ras2;
  END IF;
```

!	=	Inversion Operator
&	=	AND Operator
#	=	OR Operator

Figure 5, part of this design's Report File, lists the state registers. Under the column labeled **Expanders** is a sub-column, **Total Expanders**, showing the number of expanders required for each of the state variables. Note that **DATAEN** requires 14 expanders.

Figure 5. Efficient Use of Resources with Expanders

				Expanders		Fan-In		
Pin	MCell	LAB	Primitive	Total	Shared	INP	FBK	Name
12	15	A	DFF	5	1	9	10	ACKEN
11	13	A	DFF	4	0	4	10	ARBDIS
5	5	A	DFF	5	4	9	9	BMOE
9	9	A	DFF	10	6	9	11	BMRAS
6	7	A	DFF	2	0	8	10	BMWE
4	3	A	DFF	0	0	1	8	CASEN
10	11	A	DFF	14	9	4	14	DATAEN
3	1	A	DFF	4	4	9	9	RCASEN
17	17	A	DFF	10	5	9	9	SB0

The other sub-column, labeled **Shared Expanders**, shows the number of expanders that are shared by different macrocells. The MAX+PLUS Compiler automatically places common logic on expanders to minimize the net amount of logic required. This feature is particularly beneficial for state machines, because transition equations between states are often repeated, which means that an identical set of conditions is required to cause a transition in a number of different states. Common transition terms for different state variables are placed on expanders and shared among all state registers requiring the logic. Consequently, a MAX device has better overall device utilization because logic is not duplicated.

MAX devices also provide speed for state machines. The EPM5032-2, for instance, can toggle at over 71 MHz, and clock counters and Class 1 state machines at 62.5 MHz. Even very complex state machines, such as the one shown in Figure 4, will run with input transitions at better than 38 MHz.

EPS Series

The EPS448, a Stand-Alone Microsequencer (SAM) device, can implement Class 3 state machines. EPS448 architecture, optimized for large state machines, contains 768 product terms allocated specifically for transition equations. In addition to the dedicated product terms, it provides 448 unique state locations, i.e., SAM can implement a state machine with at least 448 states. Actually, state machines much larger than this can be implemented because an on-chip stack and an 8-bit counter allow looping and subroutine calls.

Figure 6 shows a portion of a sample SMF. State outputs are assigned and then the state transitions are specified. This state code unconditionally forces an output string (alternating 0s and 1s) to appear at the pins for 7 consecutive clock cycles. With the SMF format, this process requires 7 states. A functionally identical result could be emulated with the Algorithmic State Machine (ASM) entry language for SAM devices. This entry format uses 13 different opcodes to allow access to the stack and counter.

The ASM file, shown next to the SMF in Figure 6, provides the same functionality as the SMF, but uses fewer states. The first line of the ASM specifies the appropriate outputs in brackets, and the **LOADC 5D** loads the counter with five outputs. At the next clock, a transition to **state1** occurs, and the outputs remain at the pins. The **LOOPNZ** instruction compares the counter value to zero. Since the counter value is not zero, it is decremented, and loops back to **state1**. This process of checking for zero in the counter, decrementing the counter, and looping back continues until the counter value is zero. When it is finally zero, the loop ends and a jump to **nextstate** occurs.

Figure 6. State Machine Design for EPS448 Entered with SMF and ASM Format

SMF Entry

```
STATES: [OUT15 ..... OUT0]
state0  [0101010101010101]
state1  [0101010101010101]
state2  [0101010101010101]
state3  [0101010101010101]
state4  [0101010101010101]
state5  [0101010101010101]
state6  [0101010101010101]

state0: state1
state1: state2
state2: state3
state3: state4
state4: state5
state5: state6
state6: nextstate
```

ASM Entry

```
PROGRAM:
state0: [0101010101010101] LOADC 5D;
state1: [0101010101010101] LOOPNZ state1
                           GOTO nextstate;
```

Even in this simple example, the functionality of seven states consumes only two states. The stack can be used in a similar manner to store address values for subroutines and nested loops, or it could be used to extend the length of the counter.

In addition to implementing very large state machines, the EPS448 can be used to replace state machines that consume the resources of other, more expensive general-purpose PLDs. Often the 28-pin EPS448 can replace a state machine that consumes the logic within a 40- or 68-pin device, unless, of course, extra I/O pins are required. EPS448 architecture is optimized for state machine applications and will therefore implement large state machines more efficiently than general-purpose devices, which translates into overall board area reduction and, therefore, cost reduction.

Conclusion

Altera EPLD architectures and sizes allow implementation of the full range of state machines. Architectures offer general-purpose structures as well as structures that are optimized specifically for state machine designs. As a result, simple state machines as well as very large, complex state machines can be implemented efficiently.

Troubleshooting Programming Problems

January 1990 **Application Brief 81**

Introduction

Programming devices with Altera development systems is usually fast and easy. However, difficulties occasionally do arise. This Application Brief describes Altera programming software and hardware, helps identify and solve common programming problems, and discusses problem-solving methods. Consult *Application Brief 56A (Production Programming Specifications (Altera))* for more general information about production programming, or *Application Brief 56D (Production Programming Specifications (Data I/O))* for programming with Data I/O hardware.

Programming Software & Hardware

Programming a device from the Altera EP, EPB, and EPS families requires LogicMap software (part of A+PLUS and SAM+PLUS); an LP3, LP4, or LP5 programming card; and a programming unit. An adapter may also be required to program a device from these families. Programming a device from the Altera EPM series requires MAX+PLUS software, an LP4 or LP5 programming card, a programming unit, and an adapter. The LP3 card does not support production versions of MAX devices.

LogicMap and MAX+PLUS Software

Several different versions of programming software have been released. To find the LogicMap version number, run it and look under the word *LogicMap* on the main menu or on the LogicMap distribution diskette label. To find the version number of MAX+PLUS, run it and look at the bottom of the screen.

Table 1 lists the earliest version of LogicMap that is compatible with Altera EP, EPB, and EPS family devices. Table 2 lists the earliest version of the MAX+PLUS Programmer that is compatible with EPM family devices.

LogicMap version 4.7 and earlier versions will not work correctly with PCs that run at 10 MHz or faster since these earlier versions were created before 10-MHz PCs were available. LogicMap version 5.1 and later versions will interface with LP5 (PS/2 MicroChannel) cards. Of course, LP3s and LP4s also work with these later releases of LogicMap.

The current version of LogicMap or MAX+PLUS programming software should ***always*** be used to ensure correct device programming. The annual software maintenance agreement ensures

Table 1. Hardware/Software Programming Compatibility for EP, EPB, and EPS Family Devices

Part	Adapter	Programming Unit	Programming Card	Earliest Compatible Software Version
EP310	None	PLE3-12 or -12A	LP3, LP4, LP5	LogicMap 4.61
EP320	None	PLE3-12 or -12A	LP3, LP4, LP5	LogicMap 4.61
EP512D	PLED512	PLE3-12 or -12A	LP3, LP4, LP5	LogicMap 6.1
EP600D	PLED600	PLE3-12 or -12A	LP3, LP4, LP5	LogicMap 4.61
	PLED600-610	PLE3-12 or -12A	LP3, LP4, LP5	LogicMap 4.61
EP600J	PLEJ600	PLE3-12 or -12A	LP3, LP4, LP5	LogicMap 4.61
	PLEJ600-610	PLE3-12 or -12A	LP3, LP4, LP5	LogicMap 4.61
EP610D	PLED600	PLE3-12A	LP3, LP4, LP5	LogicMap 5.0
	PLED600-610	PLE3-12 or -12A	LP3, LP4, LP5	LogicMap 5.0
EP610J	PLEJ600	PLE3-12A	LP3, LP4, LP5	LogicMap 5.0
	PLEJ600-610	PLE3-12 or -12A	LP3, LP4, LP5	LogicMap 5.0
EP630D	PLED600-610	PLE3-12 or -12A	LP4, LP5	LogicMap 6.5
EP630J	PLEJ600-610	PLE3-12 or -12A	LP4, LP5	LogicMap 6.5
EP900D	PLED900	PLE3-12 or -12A	LP3, LP4, LP5	LogicMap 4.61
EP900J	PLEJ900	PLE3-12 or -12A	LP3, LP4, LP5	LogicMap 4.61
EP910D	PLED900	PLE3-12 or -12A	LP3, LP4, LP5	LogicMap 5.0
EP910J	PLEJ900	PLE3-12 or -12A	LP3, LP4, LP5	LogicMap 5.0
EP1210J	PLEJ1210	PLE3-12 or -12A	LP3, LP4, LP5	LogicMap 4.61
EP1800J	PLEJ1800	PLE3-12 or -12A	LP3, LP4, LP5	LogicMap 4.61
	PLEJ1800-1810	PLE3-12 or -12A	LP3, LP4, LP5	LogicMap 4.61
	PLEJ1810	PLE3-12 or -12A	LP3, LP4, LP5	LogicMap 4.61
EP1800G	PLEG1800	PLE3-12 or -12A	LP3, LP4, LP5	LogicMap 4.61
	PLEG1810	PLE3-12 or -12A	LP3, LP4, LP5	LogicMap 4.61
EP1810J	PLEJ1800	PLE3-12A	LP3, LP4, LP5	LogicMap 5.61
	PLEJ1800-1810	PLE3-12 or -12A	LP3, LP4, LP5	LogicMap 5.61
EP1810G	PLEG1810	PLE3-12A	LP3, LP4, LP5	LogicMap 5.61
EPB1400D	PLED1400	PLE3-12 or -12A	LP3, LP4, LP5	LogicMap 5.0
EPB1400J	PLEJ1400	PLE3-12 or -12A	LP3, LP4, LP5	LogicMap 5.0
EPB2001J	PLEJ2001	PLE3-12 or -12A	LP4, LP5	LogicMap 6.5
EPS448D (Rev E)	PLED448	PLE3-12A	LP3, LP4, LP5	LogicMap 6.1
EPS448J (Rev E)	PLEJ448	PLE3-12A	LP3, LP4, LP5	LogicMap 6.1

that the user will always have the current version. Older versions may not always program devices correctly.

Programming Card

The programming card plugs into an IBM XT, AT, PS/2, or compatible machine and provides the interface between the software and the rest of the hardware. Three different cards are currently supported by Altera

Table 2. Hardware/Software Programming Compatibility For EPM Devices

Part	Adapter	Programming Unit	Programming Card	Earliest Compatible Software Version
EPM5016D	PLED5016	PLE3-12 or -12A	LP4, LP5	MAX+PLUS 2.01
EPM5032D	PLED5032A	PLE3-12 or -12A	LP4, LP5	MAX+PLUS 2.01
EPM5032J	PLEJ5032A	PLE3-12 or -12A	LP4, LP5	MAX+PLUS 2.01
EPM5064J	PLEJ5064	PLE3-12 or -12A	LP4, LP5	MAXPROG 2.02
EPM5128J	PLEJ5128A	PLE3-12 or -12A	LP4, LP5	MAX+PLUS 2.01
EPM5128G	PLEG5128A	PLE3-12 or -12A	LP4, LP5	MAX+PLUS 2.01

development systems: (1) LP3 cards, which can be identified by the *LP3* silkscreen on the left-hand side of the card; (2) LP4 cards, which are the current production model cards for PCs and PC-compatible machines and can be identified by the *LP4* silkscreen in the upper left-hand corner of the card; and (3) LP5 cards, which are the current production model cards for PS/2s and can be identified by the *LP5* silkscreen also in the upper left-hand corner of the card. LP3 cards and LP4 cards run in any machine speed, but early LP5 cards are not fast enough for 20-MHz PS/2s. Altera's Applications Department should be contacted at (408) 984-2805 ext. 102 if difficulties arise in programming with a PS/2.

Programming Unit

The programming unit plugs into the back of the programming card (outside of the computer) and delivers the programming signals to the Altera device. The computer must be turned off before the programming unit is connected.

☞ WARNING: Plugging a printer into the programming card connector *may* permanently damage the programming card. Plugging a software guard into the programming card *will* permanently damage the guard.

Currently compatible programming units are the PLE3-12 and PLE3-12A. The PLE3-12 has *PLE3-12* silkscreened on the upper left-hand corner of the unit; the PLE3-12A has a *PLE3-12A* silkscreen in this location. EP320s and DIP EP1210s plug directly into the unit and can be programmed by either model. However, other devices require an adapter that maps their particular pinout and package types to that of the programming unit. The PLE3-12A works with all adapters and devices, and the PLE3-12 works with most. Refer to Table 1 for the exceptions.

Solving a Programming Problem

Several steps may help to identify and solve programming problems. If the software issues an error message, the first step is to check all connections between adapter, programming unit, and programming card. The next step is to check the appropriate documentation for clues to the problem. If the part is an EP, EPB, or EPS device, the *LogicMapII* manual and the A+PLUS **READ.ME** file should be consulted; if the part is an EPM device, the *MAX+PLUS Programmer* manual and the MAX+PLUS **READ.ME** file should be consulted.

Programming Card Problems

Some programming difficulties occur because the software cannot locate the programming card. When this happens, LogicMap typically generates the message **Programmer self test failed**, and MAX+PLUS generates the message **No programming hardware is installed**. In either case, it may help to take the card out of the computer and plug it back in firmly, making sure that the card is well seated. The machine *must* be unplugged before putting in or taking out any cards.

It is also possible that the DIP switches on the card may be set to the wrong address. LP3 and LP4 card addresses should be set to 280 Hex (i.e., all four switches are ON). However, the card may have been set to another address to avoid conflicts with other cards in the computer. To find exactly which address corresponds to the current switch settings on your card, refer to *Appendix A* in the *LogicMapII* manual or the *Installation* section of the *MAX+PLUS User Guide*. To change the default address, run the installation program for the software. For LP5 cards, the address is 420 Hex. This address is configured by the PS/2 operating system software. For information on changing this configuration, consult the *Installation* section of the *MAX+PLUS User Guide*.

Part-Related Problems

Programming difficulties may be caused by part-related problems. Some possible sources of part-related problems are detailed in the following list:

1. The part to be programmed must match the part specified by the JEDEC file. Table 1 indicates which components of a programming setup are compatible.

2. The 10-series parts (e.g., the EP610) require a PLE3-12A programming unit, or an adapter that specifically lists the part number on the front.

3. The part must be secure in the socket. Make sure it is not in backward. If the part is J-leaded, try closing the lid on the adapter and carefully pressing down on the device with a pencil eraser.

4. If the device then programs successfully, or if the error message changes, there is probably a contact problem between the device and the adapter. After examining the adapter carefully, call the Altera Applications Department about contact problems that cannot be resolved.

A good debugging technique for programming problems is to program a second part and see if the same problem arises. Programming a different type of part (an EP600 is a good choice) may also help. If no parts program, then the problem is most likely hardware-related. If one type of part does not program, but a different type does, then the problem is most likely related to software or to the specific adapter for that part.

Altera Field Diagnostics Utility

Altera provides a diagnostic utility called Altera Field Diagnostics (AFD) for programming issues. AFD, which is provided on the **INSTALL** distribution diskette that comes with development systems, is also available from the Altera Applications Department and through the Altera Electronic Bulletin Board Service. (See *Electronic Bulletin Board Service* in Section 4 of this handbook for more information.) AFD runs internal diagnostics, and then asks the user to check some voltages on the programming unit. To run the utility, type **afd**. Follow the prompts and answer the questions as they come up.

If AFD can not find the programming card, the card may not be inserted properly or the DIP switches may be set to an incorrect address as described earlier in this Application Brief.

Sometimes AFD indicates a bad card, which may result from two common mistakes: (1) plugging the printer into the programming card, or (2) "hot socketing" the programming unit—plugging the unit into the card while the computer is turned on. The first action nearly always blows an NPN transistor called Q9, which can be located by looking for a discolored or melted plastic spacer in the middle of the card. The second action blows a 2A picofuse (not slowblow) on the programming card. On LP3 cards, the fuse is labeled R23 or F1, and is located on the upper right-hand side of the card. On LP4 cards, the fuse is labeled F1 and is located on the lower right-hand side of the card. (LP5 cards do not have the fuse.) Call the Altera Applications Department for help if either of these two components is bad.

Conclusion

Altera development systems typically provide trouble-free programming. Sometimes, though, problems do occur. Most of these fall into one of the categories mentioned in this Application Brief, and many can be resolved by the user.

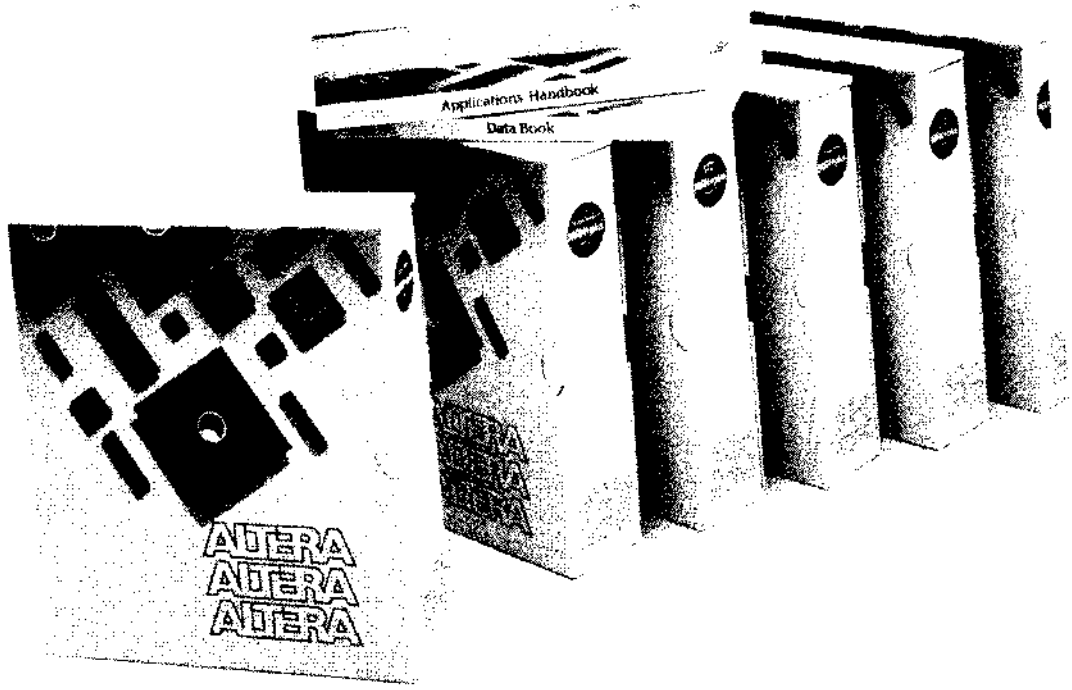
If none of the above procedures solves the problem, call the Altera Applications Department at (408) 984-2805 ext. 102 for assistance. Please have the following information at hand when calling, to help speed problem solving:

1. Results of the AFD utility analysis
2. The version number of LogicMap or the MAX+PLUS Programmer
3. The type and serial numbers of the programming unit and card
4. The exact wording of any error messages
5. The types of devices that program and that do not program

Contents

ection 4 General Information

4


Applications Handbook
Data Book
ALTERA
ALTERA
ALTERA

ntroduction

Altera provides an Electronic Bulletin Board System (BBS) for continuous access to up-to-date device and development tool information, electronic application briefs, and useful utility programs. In addition, the Bulletin Board transfers files to and from the Altera Applications Department. Owners of A+PLUS and MAX+PLUS Development Systems may refer to *Appendix E—Electronic Design Support Service* of the ***A+PLUS Reference Guide*** or ***MAX+PLUS User Guide*** for additional information. The Bulletin Board can be accessed at:

(408) 249-1100

To connect to the BBS via modem, the following items are required:

- ❒ Baud rate of 300, 1200, or 2400
- ❒ Bell Standard 212A modem or compatible
- ❒ Data format: 8-bit data; 1 stop bit; no parity

The following file transfer protocols are supported: Crosstalk, Xmodem, Kermit, ASCII, Minitel, Modem7, and Telink.

ervices and nformation

After the connection has been established, the session will begin when the user presses the space bar twice. Non-registered Altera users may log on using the account name **GUEST** and the password **EPLD**. Once a customer logs onto the service, menus guide the way and on-line help is available. The BBS provides the following services and information:

To-Altera **(File Area 1)**

This file area is used to upload customer files when a problem requires analysis or correction by an Applications Engineer.

From-Altera **(File Area 2)**

This file area is used to download files to the customer. The files downloaded from this area may be the solution to a problem file uploaded to the **To-Altera** file area or any other file.

Application Notes and Briefs **(File Area 3)**

Electronic Application Briefs (EABs) and Notes (EANs), which provide up-to-date information on using Altera EPLDs effectively, are found in this file area.

4

Application Utilities **(File Area 4)**

Electronic Application Utilities (EAUs), which assist in the design of EPLDs are available in this file area.

Applications Newsletters **(File Area 5)**

This file area contains helpful information from Altera Newsletters.

A+PLUS MacroFunction Exchange Library **(File Area 6)**

This file area is used to publicly exchange A+PLUS macrofunctions Customers may download any macrofunctions found in this file area and upload any macrofunctions they would like to share with other users.

MAX+PLUS MacroFunction Exchange Library **(File Area 7)**

This area is used to publicly exchange MAX+PLUS macrofunctions Customers may download any macrofunctions from this file area and upload any macrofunctions they would like to share with other users.

Available Products

MAX Family products available for ordering at the time of publication:

EPM5016	EPM5032	EPM5064	EPM5128
EPM5016 DC	EPM5032 DC	EPM5064 JC	EPM5128 JC
EPM5016 DC-2	EPM5032 DC-2	EPM5064 JC-2	EPM5128 JC-2
EPM5016 DC-1	EPM5032 JC		EPM5128 GC
	EPM5032 JC-2		EPM5128 GC-2
	EPM5032 DI		
	EPM5032 DM		
	EPM5032 JI		
	EPM5032 JM		

For availability information on MAX products, contact Altera Marketing at (408) 984-2800. MIL-STD-883-compliant product specifications will be provided in military product drawings available on request from Altera Marketing. These military drawings should be used to prepare source control drawings.

Product Code Summary

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
Plastic Quad Flat Pack	Q
Plastic Small Outline	S

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-883C Class B	-55° C to +125° C	883B

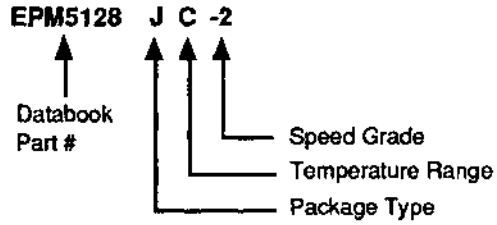
For currently available package/grade/speed combinations, please refer to product listings or call Altera Marketing at (408) 984-2805 ext. 101.

4

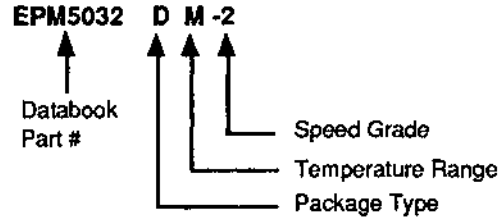
Examples

Examples of package types, and electrical and temperature grades are shown below.

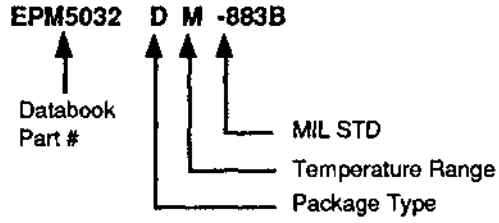
Level-1 Product:



Level-2 Product:



Level-B, MIL-STD-883C:



Development Systems/ Software

All development systems and software products should be ordered by their data sheet nomenclature. No other codes are assigned. Some examples are listed.

System Designation	Order By
PLDS-MAX	Same Designation
PLS-MAX	Same Designation
PLDVS	Same Designation
PLSA	Same Designation
PLS-EDIF	Same Designation
PLDS-ENCORE	Same Designation

For information on package outlines, please see the Altera *Data Book* (September 1988), pp. 238-244.

The following list provides the titles of Altera *Application Notes* (AN) and *Briefs* (AB). It also indicates whether they are available in the Altera Applications Handbook and/or as a stand-alone brochure. For an up-to-the-minute list and access to application design data, call the Altera Electronic Bulletin Board at (408) 249-1100.

NO.	REV.	TITLE	Applications Handbook	Printed Stand-Alone
AN1	1.0	Introduction to EPLDs	YES	NO
AN2	2.0	Replacing 20 Pin PALs with the EP320	YES	NO
AN3	3.0	Memory and Peripheral Interfacing	YES	NO
AN4	1.0	EPLD Simulation	YES	NO
AN6	2.0	Custom UART Design	YES	NO
AN7	1.0	Introduction to State Machine Design	YES	YES
AN8	1.0	EPLD Technology	YES	YES
AN9	1.0	Metastability Characteristics of EPLDs	YES	YES
AN10	1.0	SAM Applications Using State Machine Entry	YES	YES
AN11	1.0	SAM Applications Using Micro Assembler Entry	YES	YES
AN12	1.0	Microprocessor Peripheral Design with EPB1400	YES	YES
AN14	1.0	PS/2 Add-On Card Interfacing with EPB2001/EPB2002	YES	(1)
AN15	1.0	PS/2 Adapter Installation and Software Support	NO	(1)
AN16	2.0	Integrating PAL and PLA Devices with the EPM5032	(2)	YES
AN17	2.0	Integrating an Intelligent I/O Subsystem with a Single EPM5128	(2)	YES
AN18	1.0	Using the PLDVS Design Verification System	NO	YES
AN19	1.0	DSP/Imaging Applications with EPS448 SAM Microsequencer	NO	YES
AB3	2.0	Manchester Decoder/Encoder	YES	NO
AB4	2.0	T-1 Serial Transmitter	YES	NO
AB8	2.0	Efficient Counter Design with Toggle Flip-Flops	YES	NO
AB9	2.0	Designing Asynchronous Latches	YES	NO
AB14	2.0	State Machine Design Entry	YES	NO
AB15	3.0	Building Oscillators	YES	NO
AB17	2.0	State Machine Guidelines	YES	NO
AB18	2.0	Partitioning State Machines	YES	NO
AB19	2.0	Implementing Schmitt Triggers	YES	NO
AB21b	1.0	Data I/O Support	NO	YES
AB21d	1.0	EPLD Developement Tools and Programming Support	YES	NO
AB23	1.0	Multipler Circuits in EPLDs	NO	YES
AB24	4.0	Functional Simulation (PLFSIM)	YES	NO
AB26	1.0	Serial Data FIFO Design in the EP1810	NO	YES
AB27	2.0	EP1810 as Bar Code Decoder	YES	NO
AB34	2.0	Designing with MacroFunctions	YES	NO
AB39	1.0	Designing Asyn Preset with Asyn Clear	NO	YES
AB45	1.0	Testing OTP Plastic	NO	YES
AB46	2.0	Selecting Sockets for Altera J-Lead Packages	YES	NO
AB51	1.0	Total Dose Gamma Radiation	YES	NO
AB52	1.0	Low Power EPLD Design Guidelines	NO	YES
AB54	3.0	EPLD Simulation	YES	NO
AB55	1.0	Using Dual Feedback	YES	NO
AB56A	1.0	Production Programming Specifications (Altera)	YES	NO

[1]Available in *The Micro Channel Design Handbook*
[2]Available in *The MAXimalist Handbook*

Application Notes and Briefs (continued)

NO.	REV.	TITLE	Applications Handbook	Printed Stand-Alone
AB56D	2.0	Production Programming Specifications (Data I/O)	NO	YES
AB57	1.0	Serial Transmitter Using the EPB1400	YES	NO
AB58	1.0	Emulating the 74245 with EPB1400	YES	NO
AB59	1.0	Basic Building Block Design with EPB1400	YES	NO
AB60	2.0	Estimating a Design Fit	YES	NO
AB61	1.0	Design Guidelines for the EP1800/EP1810	YES	NO
AB62	1.0	Post-Programming Testing	YES	NO
AB63	1.0	Multi-Way Branching with SAM	YES	NO
AB64	1.0	Converting Meally State Machines to Moore Machines	YES	NO
AB65	1.0	Vertically Cascading SAMs	YES	NO
AB66	1.0	Input Reduction for SAM	YES	NO
AB67	1.0	DMA Controller with EPB1400	NO	YES
AB68	1.0	25 MBIT Serial Transmitter Receiver EPB1400	NO	YES
AB69	1.0	High Performance Bus Coupler Using the EPB1400	NO	YES
AB70	1.0	AVEC-How to Get A+PLUS JEDEC Files with Test Vectors	NO	YES
AB71	1.0	Boolean Equation Design Entry	YES	NO
AB72	1.0	PS/2 Master Slave Adapter Design	NO	(1)
AB73	1.0	Utility Software Programs	YES	NO
AB74	1.0	Using SAM Opcodes	NO	YES
AB75	2.0	Understanding MAX-EPLD Timing	(2)	YES
AB76	2.0	Using Expanders to Build Registered Logic in MAX EPLDs	(2)	YES
AB77	2.0	Design Guidelines for MAX EPLDs	(2)	YES
AB78	2.0	Optimizing Memory for MAX+PLUS	(2)	YES
AB79	2.0	Simulating Internal Nodes	(2)	YES
AB80	2.0	Choosing the Right EPLD for a State Machine Application	(2)	YES
AB81	2.0	Troubleshooting Programming Problems	(2)	YES

Available in *The Micro Channel Design Handbook*
Available in *The MAXimalist Handbook*

Application Utilities

AU000	Functional Overview of Electronic Application Utilities
AU002	PAL2EPLD JEDEC File Conversion Utility
AU003	Conversion Utility for EP310 to EP320 JEDEC Files
AU004	LogiCaps Plotting Utility for Houston Instrument Plotters
AU005	JEDPACK, JEDEC File Compactor
AU006	ADF Address Decoder Generator
AU007	JEDSUM, JEDEC Checksum Generator
AU008	AVEC Test Vector Generator
AU009	BACKPIN, Back Annotator for LogiCaps
AU010	PC-CAPS File Converter
AU011	DASH (FutureNet) File Converter
AU012	LEF2ABEL File Converter
AU013	PAL2ADF Conversion Utility
AU015	JED2ADF Conversion Utility
AU016	LCA2ADF (LCA) Conversion Utility

CALIFORNIA (Headquarters)
ALTERA CORPORATION
2610 Orchard Parkway
San Jose, CA 95134
Telephone: (408) 984-2800
Telex: 888496
FAX: (408) 248-7097

SOUTHERN CALIFORNIA
ALTERA CORPORATION
17100 Gillette Avenue,
Irvine, CA 92714
Telephone: (714) 474-9616
FAX: (714) 474-7355

ILLINOIS
ALTERA CORPORATION
200 W. Higgins Road, Suite 216,
Schaumburg, IL 60195
Telephone: (708) 310-8522
Telex: 5101011409
FAX: (708) 310-0909

MASSACHUSETTS
ALTERA CORPORATION
945 Concord Street,
Framingham, MA 01701
Telephone: (508) 626-0181
Telex: 948477
FAX: (508) 879-0698

GEORGIA
ALTERA CORPORATION
1080 Holcomb Bridge Road
Suite 300, Bldg. 100
Roswell, GA 30076
Telephone: (404) 594-7621
Telex: 382207
FAX: (404) 998-9830

TEXAS
ALTERA CORPORATION
Plaza Executive Suites
15770 N. Dallas Parkway
Suite 600
Dallas, TX 75248
Telephone: (214) 233-1491
FAX: (214) 233-1493

EUROPE
ALTERA CORPORATION
25, Av Beaulieu
B-1160 Bruxelles
Belgium
Telephone: 02 660 2077
Telex: (886) 270 87901
FAX: 02 660 5225

ALTERA CORPORATION
21 Broadway
Maidenhead,
Berkshire SL6 1JK
England
Telephone: 0628 32516
Telex: (851) 94016389
FAX: 0628 770892

ALTERA GMBH
Ismaninger Straße 21
D-8000 München 80
West Germany
Telephone: 089/41300614
Telex: 5213250 (IBCM)
FAX: 089 4707228

Altera Sales Representatives & Distributors

For a complete list of Sales Representatives and Distributors, please consult the Altera ***Data Book*** (September 1988), pp. 246-252.

4

Notes:

Altera Corporation
2610 Orchard Parkway
San Jose, CA 95134-2020
(408)984-2800 Telex 888496