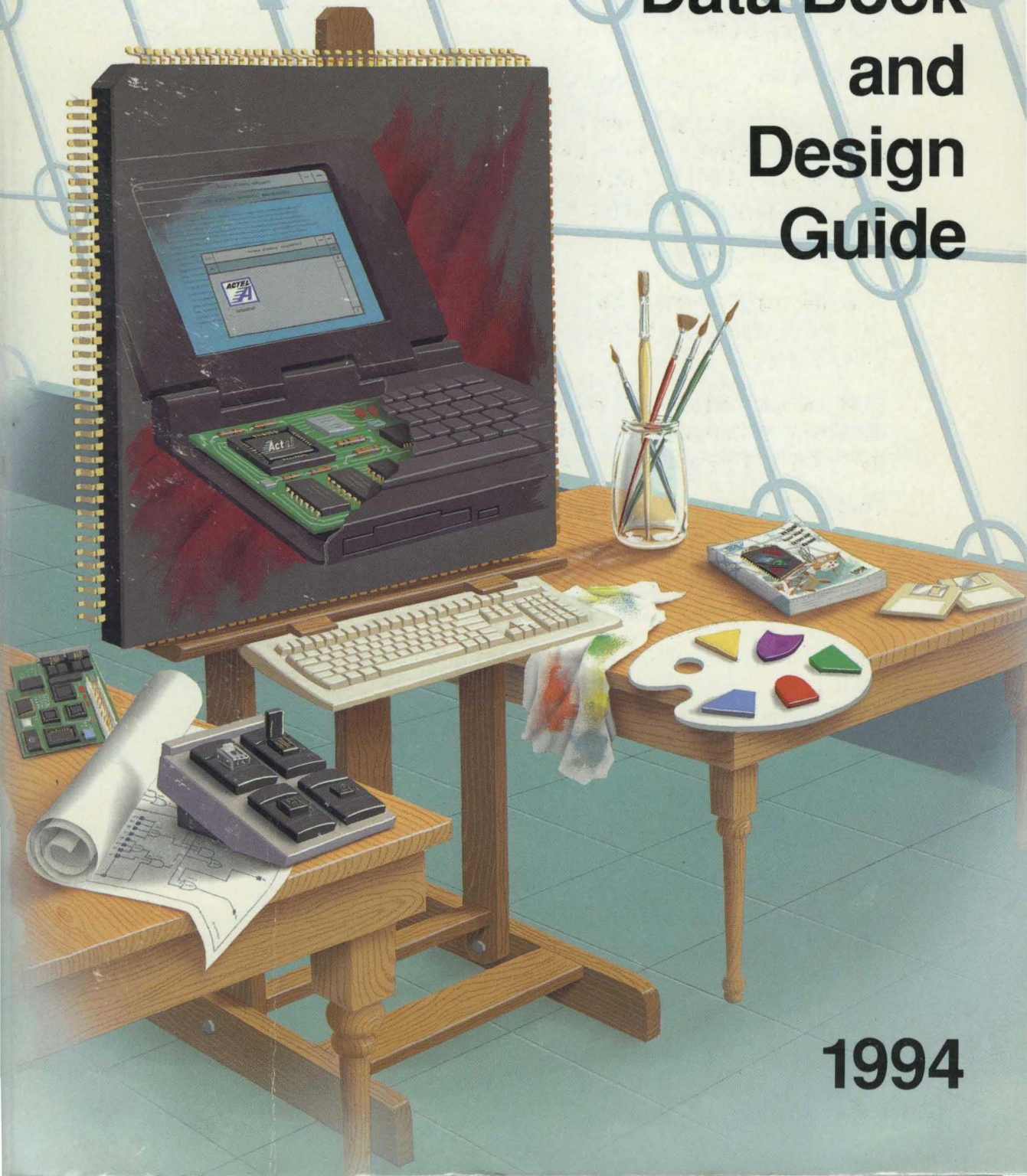
Actel

FPGA Data Book and Design Guide



1994

Are you an artist?

We think so.

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

We call that creative.

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

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

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

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

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

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

You have the ideas and the vision.

We provide the brushes, the paint and the canvas.

Fast Flexible Design

"The Actel design tools are like Hardware Heaven: you draw a schematic, simulate the logic, and view any and all internal nodes... You can place and route the chip in about 30 minutes, then analyze the post-layout timing on the same analyzer. And then, when you are satisfied with the design, you make a chip. All this can be done without ever leaving your PC. This is the way the world should be. Make your boss buy you these tools."

David Erickson
Vice President of Hardware Engineering
Datacube Corporation

Quote from *The Computer Applications Journal*, issue #34, May 1993

"Our i860-based numerical accelerator board incorporated four Actel A1460A devices, each implementing one of a wide variety of functions ranging from a DMA controller and four DRAM controllers to control registers and an internal bus controller. The Actel A1460A FPGAs were the only programmable devices available that met the performance, capacity and I/O requirements."

Ken Linton
Hardware Engineering Manager
SKY Computers

Quote from *Actel ACT 3 Press Release*, December 13, 1993

"All things considered, my opinion based on this project is that designing with an FPGA is actually easier than designing with SSI and MSI logic. I no longer wonder what applications FPGAs are useful for, but rather what applications still make sense for small- and medium-scale integration TTL and CMOS logic."

Doug Conner
Technical Editor
EDN Magazine

Quote from *Hands-on FPGA Project*, parts I and II, *EDN Magazine*, April 9 and 23, 1992.
For further details, see complete article reprint in Section 6.

Integrating PALs and TTL

"It simply wasn't feasible to implement that many shift registers in PALs... The choice was obvious, Actel was the clear winner. Actel's products are easy to work with, offer the highest available logic density and performance and can be relied upon to satisfy the most demanding design requirements."

Mike Casteloes
Senior Engineer
Interstate Electronics Corporation

For further details, see complete case history in Section 10.

"I was well aware of Actel's technology from the beginning. The Actel devices worked out exceptionally well. I was pleasantly surprised that we were able to integrate as much logic as we could into the parts."

David Kranzler
Engineering Project Leader
3COM Corporation

For further details, see complete case history in Section 10.

Computer Systems and Peripherals

RasterOps designed the Actel A1240, a 4,000-gate FPGA, into its PaintBoard Turbo family of graphics products for use with Apple Macintosh computers. The architectural flexibility of the A1240 made it suitable for implementing the varied functions required for the complex Macintosh graphics operations.

Telecommunications

Chipcom chose the Actel A1020 FPGA for use in a 24-port twisted pair Ethernet product. Actel believes that its device was selected over competitive SRAM-based FPGAs primarily because of the combination of effective design tools and performance. The Actel circuit architecture also proved more flexible than most PLD architectures in its ability to implement the required logic.

Military and Aerospace

Westinghouse military system engineers replaced 10 standard logic chips with one of Actel's FPGAs, which handled the entire error correction, registration and interface logic for Westinghouse's high speed, high dynamic range analog-to-digital converters.

Industrial Control Equipment

Siemens Medical Electronics cut its production cycle by an estimated two months by using Actel's FPGAs for both prototype development and production for a pulse oximetry system. Siemens' engineers produced a working prototype only two weeks after delivery of Actel's development system. All of the microprocessor interface logic was implemented on one of Actel's FPGAs.

Actel FPGA Data Book and Design Guide

1994

Contributors: Tara Anderson, Sam Beal, Larry Blessman, Sanjiv Desai, Tracy Fang, Gervais Fong, Steve Gurklys, Ken Hayes, Yousef Khalilollahi, Kamal Koraitem, Warren Miller, Kirk Owyang, Hitesh Patel, Quat Tran, and Bruce Weyer.

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

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

Actel FPGA Data Book and Design Guide

Order of Contents

Section 9: Application Examples and Design Techniques

Section 10: Customer Case Histories

Section 11: Technical Support Services

Introduction

Actel Company Background

Actel Corporation is a leading supplier of high performance field programmable gate arrays (FPGAs) and currently provides the dominant antifuse-based architecture in the FPGA market. As a Sunnyvale-based start-up in 1988, Actel Corporation was the first company to successfully develop and manufacture an antifuse-based FPGA, enabling its products to surpass the performance, density, and cost-per-gate of existing memory-based FPGAs. Actel's lead continues to escalate, and today Actel's advanced product lines are endorsed by electronics industry leaders it has partnered with. Actel's partners include Hewlett-Packard Company, Texas Instruments, and Matsushita Electronics Corporation (Panasonic).

The FPGA Market

In the past five years, the FPGA market has experienced tremendous growth and currently is the fastest growing segment of the $5.7 billion ASIC market. Researchers, such as In-Stat Inc., predict that the FPGA market will reach $887 million by 1997, representing a compound annual growth rate of 30 percent (see Figure 1).

Why is the FPGA market growing so fast?

Because a top priority for every engineer today is how to shorten the design cycle, FPGAs have become an important factor in new product development. Offering the design flexibility and capacity of gate arrays, plus the convenience and speed of desktop programming, FPGAs deliver a quicker time-to-market solution than any other method of logic integration.

At the same time, engineers are still facing the traditional demands to make new products cheaper and smaller. FPGAs provide an important solution to this problem, too. By integrating thousands of gates of random logic onto one chip, designers can reduce board space, component count, and power requirements at the same time.

FPGAs versus Conventional Logic

While discrete logic and programmable logic devices (PLDs) have long been a viable solution for high speed and high drive capability, conventional logic has relinquished part of the market to more integrated solutions such as FPGAs. Compared to conventional logic, FPGAs can:

- reduce system size (fewer devices)
- reduce system costs
- improve system performance
- improve system reliability (fewer devices on the board)

FPGAs versus Masked Gate Arrays

FPGAs also offer clear advantages over masked gate arrays, which are traditionally considered the high end of the logic

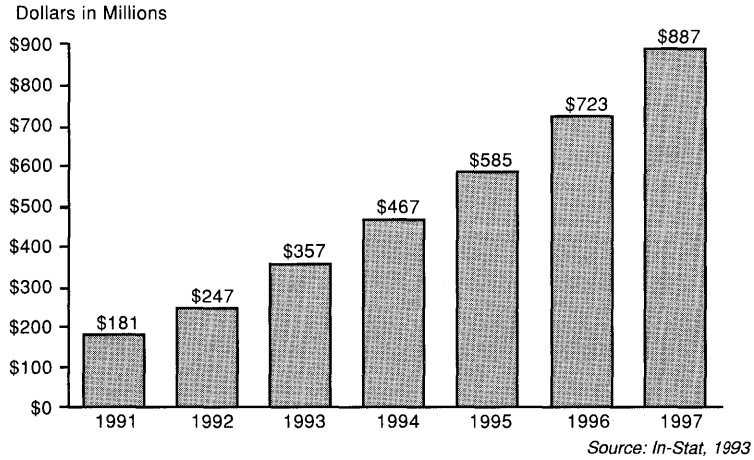


Figure 1. The FPGA Market

market. Since 1991, the category that includes FPGAs has realized more annual designs than masked gate arrays, according to Dataquest. Compared to masked gate arrays, FPGAs offer:

- reduced time-to-market
- increased flexibility
- lower cost for small and medium-size volumes
- low-risk design options (no NRE costs)

Actel FPGA Families

Actel currently offers the ACT™ 1, ACT 2, and ACT 3 families of FPGAs that span 1,200 to 10,000 gates in density (see Figure 2). These families of devices are based on proprietary architecture and interconnect technologies and offer users reliable, risk-free logic integration. The company's strong emphasis on architecture, routing, and programming technologies has enabled Actel to emerge as the technology leader in the FPGA market. The foundation of Actel's technology is the unique synergy created by the company's novel programming element, the PLICE® (Programmable Low-Impedance Circuit Element) antifuse and Actel's multiple-patented FPGA architecture. The advanced performance, capacity, low cost, and ease-of-use characteristics of the ACT families are highly dependent on this synergy. Indeed, Actel's antifuse and architecture must be used together to attain the benefits that Actel devices offer.

Actel's Patented PLICE Antifuse

The PLICE antifuse is a nonvolatile, two-terminal element that exhibits low "on" resistance when programmed and provides the same wire-to-wire interconnect functions as "vias" provide in mask-programmable gate arrays and that transistor-based EPROM and RAM cells and metal fuses provide in conventional programmable logic devices.

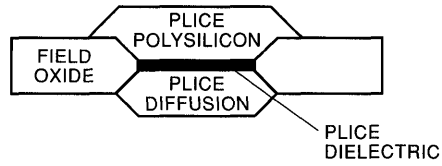


Figure 3. The PLICE Antifuse Element

The PLICE antifuse offers key advantages in both size and electrical properties (see Figure 3). The antifuse is small enough that it fits within the width of the channeled routing track. This means that there is basically no die size overhead created by the antifuse itself. The combination of small size and low delay characteristics of the PLICE antifuse has allowed Actel to make two key architectural breakthroughs:

- to offer abundant routing resources while offering very small die sizes
- to offer a highly flexible, highly granular architecture (small logic blocks)

Actel's Patented Segmented, Channeled Routing Architecture

Actel's segmented, channeled routing structure consists of various lengths of routing segments separated by antifuses. Some tracks consist of many short routing segments, while other tracks offer direct interconnect across the chip. Due to the abundant number of routing tracks per channel, a wide variety of segment lengths is offered. These routing tracks can be accessed by programming vertical antifuses located within the metal 1 to

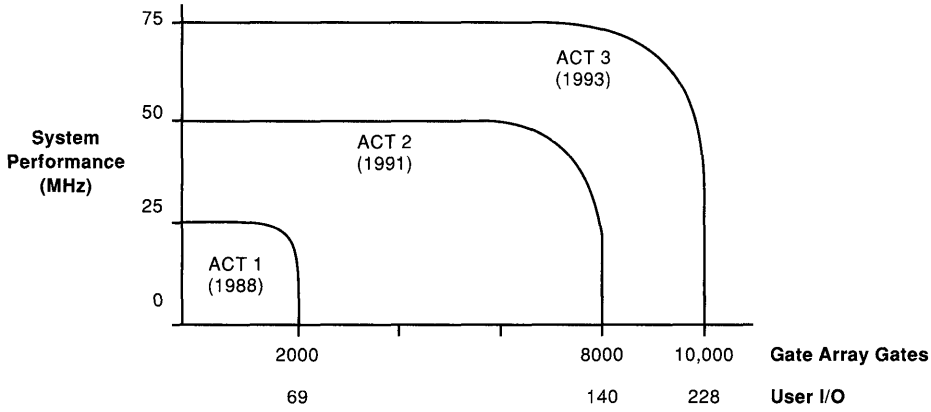


Figure 2. The FPGA Families

metal 2 vias. See Figure 4 for an example of a typical ACT 1 routing structure.

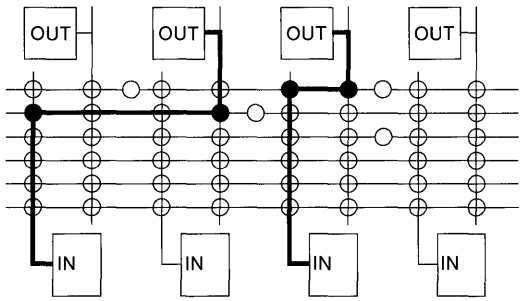


Figure 4. ACT 1 Routing Example

Actel's segmented routing architecture minimizes the number of antifuses in any path by offering various lengths of interconnect segments. For closely placed modules, short tracks are accessed. For more distant modules, long tracks are accessed. This flexible routing and interconnect architecture ensures that no more than four antifuses are used in any path, with 90 percent of all interconnects requiring only two antifuses.

Highest, Most Predictable Performance

Actel's ACT 3 family offers the highest performance in the FPGA industry: on-chip performance of up to 150 MHz; chip-to-chip speeds over 80 MHz; and 9 ns Clock-to-Out speeds. Figure 5 shows performance of the A1425A-1 per the Programmable Electronics Performance Corporation (PREP™) Benchmarks*, while using 100% automatic placement and routing tools.

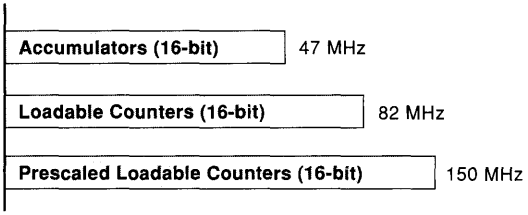


Figure 5. A1425A-1 Performance (Worst-Case Commercial)

The antifuse-based architectures have also demonstrated very tight distributions in performance. The tighter the distribution, the easier it is to design with. Figure 6 shows the average performance variability of antifuse-based and SRAM-based devices as derived from the PREP benchmarks* while using 100% automatic placement and routing software. The LOWER the variability, the HIGHER the predictability of your design. Clearly, antifuse-based FPGAs are much more highly predictable than SRAM-based FPGAs.

All PREP numbers refer to Suite #1, Version 1.2, dated 3-28-93; any analysis is not endorsed by PREP.

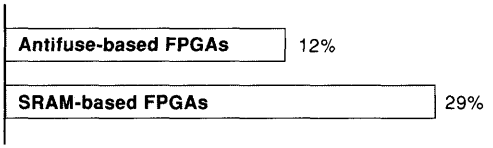


Figure 6. Performance Variability (LOWER variability means HIGHER predictability and ease-of-design)

This highly predictable performance is directly related to Actel's proprietary PLICE antifuse and segmented, channeled routing architecture. Total capacitance and resistance of each interconnect path has been minimized by:

- minimizing impedance of the antifuse
- minimizing the number of antifuses per interconnect path (no more than four, 90% of interconnects requiring only two antifuses)
- minimizing the length of the metal tracks used for interconnect

For more details on performance and performance predictability, see Section 2, which covers the PREP benchmark results.

Actel's Low-Cost Solution

Strong emphasis in process and architectural technologies, along with tight strategic partnerships, allows Actel to offer the most cost-competitive solutions in the programmable logic industry. Actel's technology leadership produces the industry's smallest die sizes for comparable density devices. Actel's strategic partnerships ensure continuous product availability, as well as new product innovation, at industry leading costs. Actel's overall price leadership can be attributed to:

- very small die sizes due to the PLICE antifuse technology
- competitive, multiple foundry fabrication
- second sourced products
- leadership in process technology and architectural design

Higher Performance for Lower Cost

Actel introduced the industry's first antifuse-based programmable architecture in 1988. The small size of the PLICE antifuse has allowed Actel to introduce devices 20% to 50% smaller in die size than comparable competitive devices. Actel's learning curve has been very steep, having manufactured and sold over 3 million FPGAs containing over 750 billion antifuses over the past five years. These economies of scale, paired with technological breakthroughs, have allowed Actel to reduce initial pricing by a magnitude, and to continue to offer higher performance solutions at lower prices (see Figure 7).

Actel's price leadership has been demonstrated in conjunction with industry standard PREP benchmarks. Actel ACT 2 devices are the best value in the mid-range and high-density classes. In fact, Actel antifuse-based FPGAs represent five of the seven best value programmable devices based on certified PREP results (see Figure 8).

For more information regarding Actel's value proposition, see the PREP Benchmarks in Section 2.

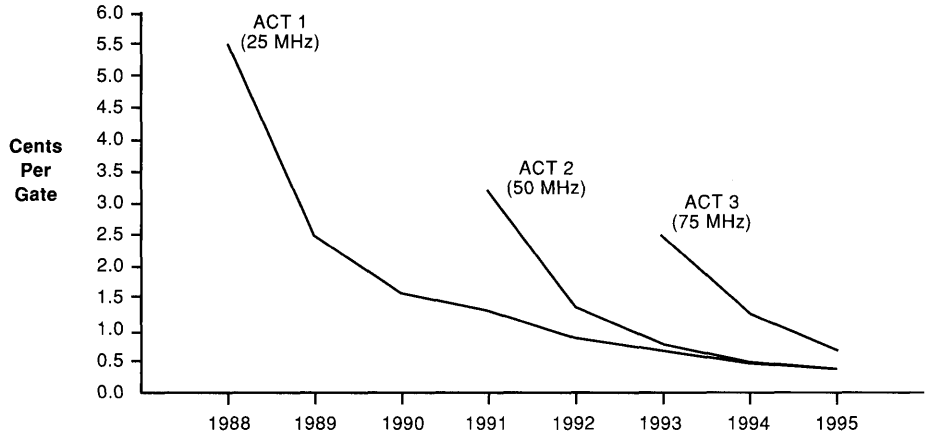


Figure 7. Actel Families: Cost Per Gate

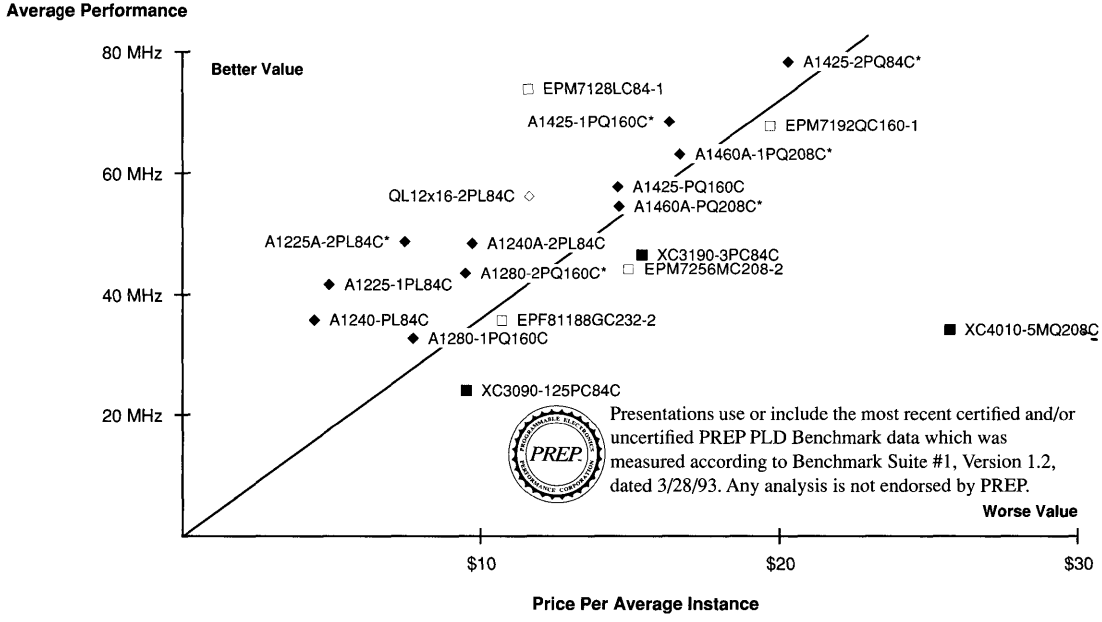


Figure 8. Price Per Average Instance

Actel's High-Density, High-Performance Solution

In 1991, Actel introduced the largest FPGA in the industry, the 8000-gate A1280. In 1993, Actel is introducing the industry's first high-performance, high-density programmable solutions, the A1460A and A14100A. These devices offer up to 10,000 gate array gates and up to 228 user I/Os while delivering on-chip performance to 100 MHz and clock-to-out speeds of 12 nanoseconds (see Table 1).

Table 1. Specifications for A1460A and A14100A

	High Density		High Performance	
	Gate Density	User I/O	On-Chip	Clock-Out
A14100A	10,000	228	100 MHz	12.0 ns
A1460A	6,000	168	100 MHz	11.6 ns

The introduction of the ACT 3 family punctuates the advantages of the antifuse technology. The low impedance of the antifuse supports high-performance designs even at high gate densities. No other programmable technology can offer this combination of high speed and high density (see Figure 9). This combination opens new design possibilities in computers, graphics, high-speed telecommunications and other high-speed and high-I/O applications.

In comparison, EPROM-based architectures offer high performance at low gate densities, but the architecture's performance dwindles and costs rise dramatically as gate capacity increases. The SRAM-based FPGA architectures have been shown to support high-density designs, but are lacking in performance and are quite expensive. Both the lack of performance and the high costs can be attributed to the large size and electrical characteristics of the SRAM interconnect elements. The small size and low impedance of Actel's antifuse-based architecture offers high speed at high densities, at the fraction of the cost of SRAM devices.

Actel Leads in Reliability

Actel builds the most reliable field programmable gate arrays (FPGAs) in the industry, with overall reliability ratings of 6 Failures-in-time (FITs at 70°C ambient, 0.9 EV), corresponding to a useful life of more than 40 years. Actel FPGAs have been production proven, 4 million devices shipped to date and over 700 billion antifuses manufactured. The architecture of the devices permits a highly testable structure for routing, logic resources, I/O pads, and programming circuits. As a result, all devices are fully tested prior to shipment, with an outgoing defect level of only 122 ppm. For more details on device reliability and testing, see Section 4.

Actel Offers Fast Flexible Design

The Actel design flow has been tuned to minimize the effort of bringing designs to market (see Figure 10). A windows-based interface allows users to complete ACT 1, ACT 2, and ACT 3 designs, from concept to silicon, within hours. Compatibility to industry standard design entry tools allows designers to leverage their existing expertise.

The fine granularity of the Actel architecture allows designers to implement simple to complex, random to highly structured designs while using 100% automatic placement and routing tools guaranteeing 80% to 95% gate utilization. Timing analysis software and backannotation simulation allow the designer to quickly verify performance and timing requirements. On-site programming and diagnostic tools provide quick-turn design implementation.

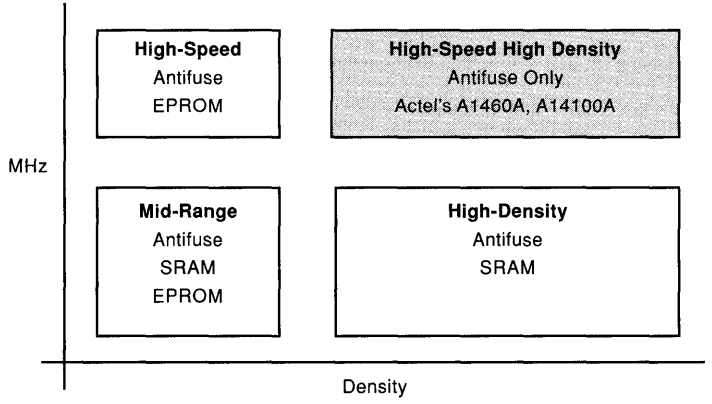


Figure 9. Speed and Density Classifications by Technology

Actel's Designer and Designer Advantage™ systems support schematic capture, Boolean high-level equation entry, state machine entry, design synthesis input as well as EDIF netlists. The systems offer schematic entry and simulation support for Cadence™, Mentor Graphics®, OrCAD™, and Viewlogic®. Additional CAE interfaces are supported through Actel's Industry Alliance Program, including Data I/O, Intergraph, and MINC. The systems are supported on the following platforms: PC 386/486 and Sun® and HP workstations.

For further information regarding Actel's fast, flexible design systems, please see Section 5.

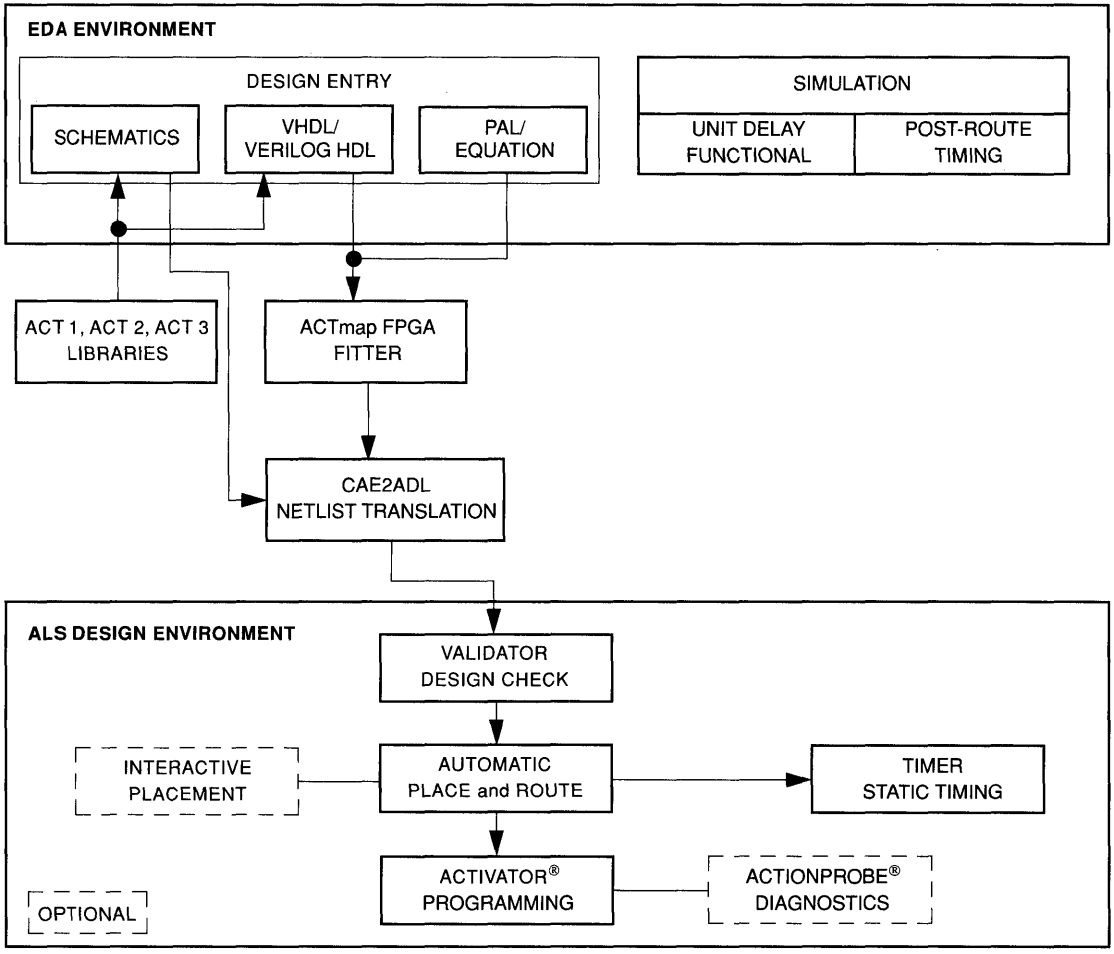
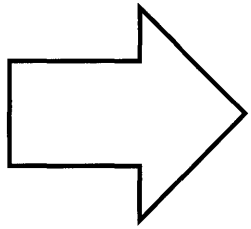


Figure 10. Typical Design Flow for an Actel FPGA

Component Data



Product Selector Guide

Device	Pkg[1]	# Pins	Speed Option[2]	Temp.[3]	User I/O	Gates	Flip–Flops		Equiv. Pkgs.	
							Dedicated	Max.	TTLs	20–Pin PALs
A1010B	PL	44	Std, –1, –2	C, I	34	1,200	0	147	30	12
	PL	68	Std, –1, –2	C, I	57	1,200	0	147	30	12
	VQ	80	Std, –1, –2	C	57	1,200	0	147	30	12
	PQ	100	Std, –1, –2	C, I	57	1,200	0	147	30	12
	PG	84	Std, –1	C, M, B	57	1,200	0	147	30	12
A1020B	PL	44	Std, –1, –2	C, I	34	2,000	0	273	50	20
	PL	68	Std, –1, –2	C, I	57	2,000	0	273	50	20
	PL	84	Std, –1, –2	C, I	69	2,000	0	273	50	20
	VQ	80	Std, –1, –2	C	69	2,000	0	273	50	20
	PQ	100	Std, –1, –2	C, I	69	2,000	0	273	50	20
	PG	84	Std, –1	C, M, B	69	2,000	0	273	50	20
	CQ	84	Std, –1**	C, M, B, E	69	2,000	0	273	50	20
A1225A	PL	84	Std, –1, –2	C, I	72	2,500	231	382	63	25
	PQ	100	Std, –1, –2	C, I	83	2,500	231	382	63	25
	PG	100	Std, –1, –2	C	83	2,500	231	382	63	25
A1240A	PL	84	Std, –1, –2	C, I	72	4,000	348	568	100	40
	PQ	144	Std, –1, –2	C, I	104	4,000	348	568	100	40
	PG	132	Std, –1, –2***	C, M, B	104	4,000	348	568	100	40
A1280A	PQ	160	Std, –1, –2	C, I	125	8,000	624	998	200	80
	PG	176	Std, –1, –2	C	140	8,000	624	998	200	80
	CQ	172	Std, –1**, –2***	C, M, B, E	140	8,000	624	998	200	80
A1415A*	PL	84	Std	C, I	70	1,500	264	312	38	15
	PQ	100	Std	C, I	80	1,500	264	312	38	15
	PG	100	Std	C	80	1,500	264	312	38	15
A1425A*	PL	84	Std, –1	C, I	70	2,500	360	435	63	25
	PQ	100	Std, –1	C, I	80	2,500	360	435	63	25
	PQ	160	Std, –1	C, I	100	2,500	360	435	63	25
	PG	133	Std, –1***	C, M, B	100	2,500	360	435	63	25
A1440A*	PQ	160	Std	C, I	130	4,000	568	706	100	40
	PG	175	Std	C	140	4,000	568	706	100	40
A1460A*	PQ	208	Std	C, I	167	6,000	768	976	150	60
	PG	207	Std	C, M, B	168	6,000	768	976	150	60
A14100A*	PG	257	Std	C, M, B	228	10,000	1,153	1,493	250	100

See below for table notes.
* Consult Actel for availability
** Extended Flow (E) not offered in –1 Speed
*** Offered for Commercial (C) devices only

Notes:

1. Package types:
   - CQ – Ceramic Quad Flat Packs
   - PG – Ceramic Pin Grid Arrays
   - PL – Plastic J–Leaded Chip Carriers
   - PQ – Plastic Quad Flat Packs
   - VQ – Very thin (1.0 mm) Quad Flatpacks
2. Speed Options:
   - Std – Standard Speed
   - –1 – approximately 15% faster than Standard
   - –2 – approximately 25% faster than Standard
3. Temperature Range:
   - C – Commercial Temperature (0 to +75 C)
   - I – Industrial (–40 to +85 C)
   - M – Military (–55 to +125 C)
   - B – MIL-STD-883C
   - E – Extended Flow

ACT™ 1 Field Programmable Gate Arrays

Features

- Up to 2000 Gate Array Gates (6000 PLD equivalent gates)
- Replaces up to 53 TTL Packages
- Replaces up to seventeen 20-Pin PAL® Packages
- Design Library with over 250 Macro Functions
- Gate Array Architecture Allows Completely Automatic Place and Route
- Up to 547 Programmable Logic Modules
- Up to 273 Flip-Flops
- Flip-Flop Toggle Rates to 100 MHz
- Two In-Circuit Diagnostic Probe Pins Support Speed Analysis to 25 MHz
- Built-In High Speed Clock Distribution Network
- I/O Drive to 10 mA
- Nonvolatile, User Programmable
- Logic Fully Tested Prior to Shipment

Description

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

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

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

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

Product Family Profile

Device	A1010B	A1020B
Capacity		
Gate Array Equivalent Gates	1,200	2,000
PLD Equivalent Gates	3,000	6,000
TTL Equivalent Packages	30	50
20-Pin PAL Equivalent Packages	12	20
Logic Modules	295	547
Flip-Flops (maximum)	147	273
Routing Resources		
Horizontal Tracks/Channel	22	22
Vertical Tracks/Column	13	13
PLICE Antifuse Elements	112,000	186,000
User I/Os (maximum)	57	69
Packages:	44 PLCC	44 PLCC
	68 PLCC	68 PLCC
		84 PLCC
	100 PQFP	100 PQFP
	80 VQFP	80 VQFP
	84 CPGA	84 CQFP
		84 CPGA
Performance		
Flip-Flop Toggle Rate (maximum)	100 MHz	100 MHz
System Speed (maximum)	40 MHz	40 MHz
CMOS Process	1.0 µm	1.0 µm

Note:
1. See Product Plan on page 1-6 for package availability.

The Designer and Designer Advantage Systems

The ACT 1 device family is supported by Actel's Designer and Designer Advantage Systems, allowing logic design implementation with minimum effort. The systems interface with the resident CAE system to provide a complete gate array design environment: schematic capture, simulation, fully automatic place and route, timing verification, and device programming. The systems are available for 386/486™ PC and for HP™ and Sun™ workstations and for running Viewlogic®, Mentor Graphics®, Cadence™, and OrCAD™.

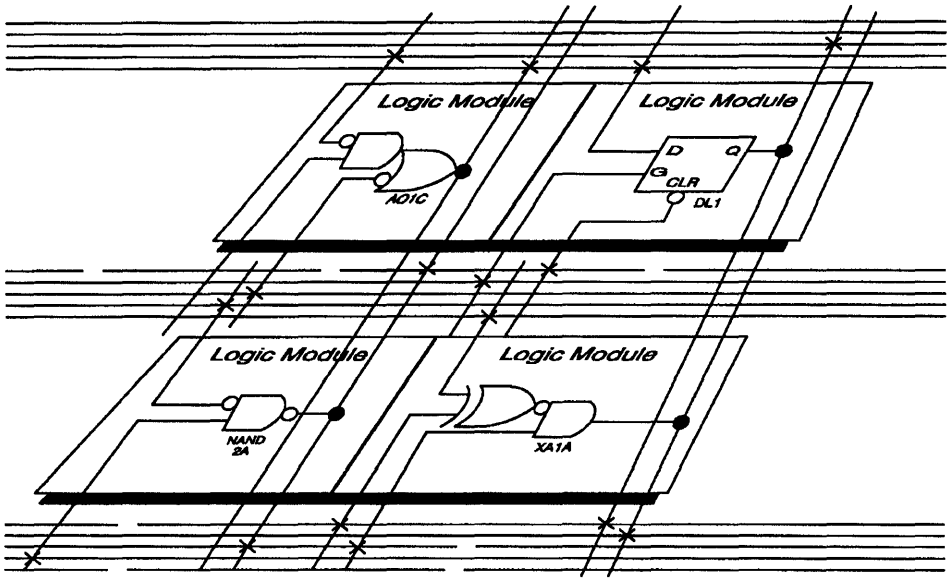


Figure 1. Partial View of an ACT 1 Device

ACT 1 Device Structure

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

The ACT 1 Logic Module

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

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

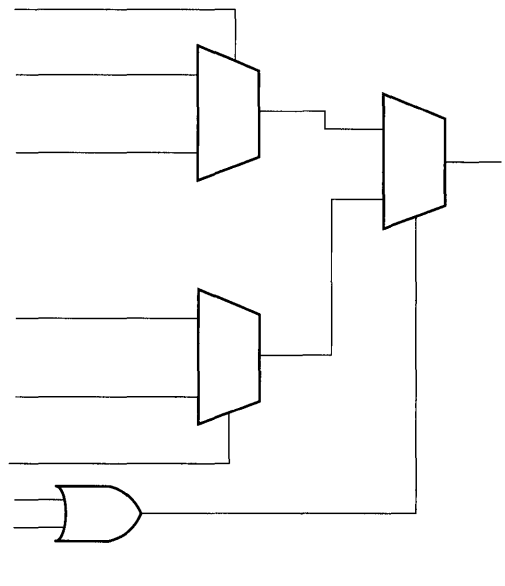


Figure 2. ACT 1 Logic Module

I/O Buffers

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

Device Organization

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

Probe Pin

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

ACT 1 Array Performance

Temperature and Voltage Effects

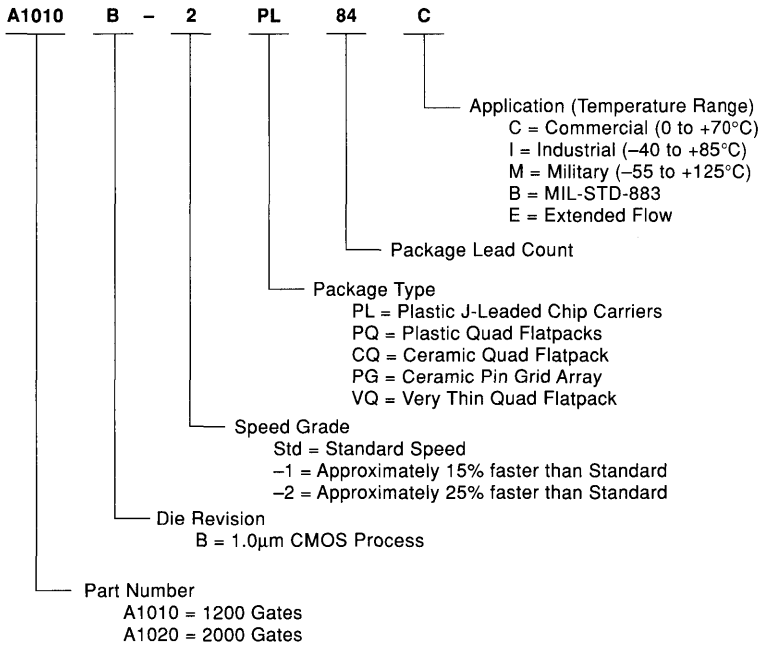
Worst-case delays for ACT 1 arrays are calculated in the same manner as for masked array products. A typical delay parameter is multiplied by a derating factor to account for temperature, voltage, and processing effects. However, in an ACT 1 array, temperature and voltage effects are less dramatic than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

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

Logic Module Size

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

Ordering Information



Product Plan

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

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

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

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

Device Resources

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

Pin Description

CLK **Clock (Input)**

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

DCLK **Diagnostic Clock (Input)**

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

GND **Ground**

Input LOW supply voltage.

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

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

MODE **Mode (Input)**

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

NC **No Connection**

This pin is not connected to circuitry within the device.

PRA **Probe A (Output)**

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

PRB **Probe B (Output)**

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

SDI **Serial Data Input (Input)**

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

V_{CC} **Supply Voltage**

Input HIGH supply voltage.

V_{PP} **Programming Voltage**

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

Absolute Maximum Ratings[1]

Free air temperature range

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

Note:

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

Recommended Operating Conditions

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

Note:

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

Electrical Specifications

Symbol	Parameter	Commercial		Industrial		Military		Units
		Min.	Max.	Min.	Max.	Min.	Max.	
V_{OH}[1]	(I_{OH} = –10 mA)[2]	2.4						V
	(I_{OH} = –6 mA)	3.84						V
	(I_{OH} = –4 mA)			3.7		3.7		V
V_{OL}[1]	(I_{OL} = 10 mA)[2]		0.5					V
	(I_{OL} = 6 mA)		0.33		0.40		0.40	V
V_{IL}		–0.3	0.8	–0.3	0.8	–0.3	0.8	V
V_{IH}		2.0	V_{CC} + 0.3	2.0	V_{CC} + 0.3	2.0	V_{CC} + 0.3	V
Input Transition Time t_R, t_F[2]			500		500		500	ns
C_{IO} I/O Capacitance[2, 3]			10		10		10	pF
Standby Current, I_{CC}[4]			3		10		20	mA
Leakage Current[5]		–10	10	–10	10	–10	10	µA
I_{OS} Output Short Circuit Current[6]	(V_O = V_{CC})		140		140		140	mA
	(V_O = GND)		–100		–100		–100	mA

Notes:

1. Only one output tested at a time. V_{CC} = min.
2. Not tested, for information only.
3. Includes worst-case 84-pin PLCC package capacitance. V_{OUT} = 0 V, f = 1 MHz.
4. Typical standby current = 1 mA. All outputs unloaded. All inputs = V_{CC} or GND.
5. V_O, V_{IN} = V_{CC} or GND.
6. Only one output tested at a time. Min. at V_{CC} = 4.5 V; Max. at V_{CC} = 5.5 V.

Package Thermal Characteristics

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

Maximum junction temperature is 150°C.

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

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

Package Type	Pin Count	θjc	θja Still Air	θja 300 ft/min	Units
Plastic J-Leaded Chip Carrier	44	15	52	40	°C/W
	68	13	45	35	°C/W
	84	12	44	33	°C/W
Plastic Quad Flatpack	100	13	55	47	°C/W
Very Thin (1.0 mm) Quad Flatpack	80	12	68	55	°C/W
Ceramic Pin Grid Array	84	8	33	20	°C/W
Ceramic Quad Flatpack	84	5	40	30	°C/W

Power Dissipation

The following formula is used to calculate total device dissipation.

Total Device Power (mW) = (0.20 x N x F1) + (0.085 x M x F2) + (0.80 x P x F3)

Where:

F1 = Average logic module switching rate in MHz

F2 = CLKBUF macro switching rate in MHz

F3 = Average I/O module switching rate in MHz

M = Number of logic modules connected to the CLKBUF macro

N = Total number of logic modules used in the design (including M)

P = Number of outputs loaded with 50 pF

Average switching rate of logic modules and of I/O modules is some fraction of the device operating frequency (usually CLKBUF). Logic modules and I/O modules switch states (from low-to-high or from high-to-low) only if the input data changes when the module is enabled. A conservative estimate for average logic module and I/O module switching rates (variables F1 and F3, respectively) is 10% of device clock driver frequency.

If the CLKBUF macro is not used in the design, eliminate the second term (including F2 and M variables) from the formula.

Sample A1020 Device Power Calculation

To illustrate the power calculation, consider a large design operating at high frequency. This sample design utilizes 85% of available logic modules on the A1020-series device (.85 x 547 = 465 logic modules used). The design contains 104 flip-flops (208 logic modules). Operating frequency of the design is 16 MHz. In this design, the CLKBUF macro drives the clock network. Logic modules and I/O modules are switching states at approximately 10% of the clock frequency rate (.10 x 16 MHz = 1.6 MHz). Sixteen outputs are loaded with 50 pF.

To summarize the design described above: N = 465; M = 208; F2 = 16; F1 = 4; F3 = 4; P = 16. Total device power can be calculated by substituting these values for variables in the device dissipation formula.

Total device power for this example =

(0.20 x 465 x 1.6) + (0.085 x 208 x 16) + (0.80 x 16 x 1.6) = 452 mW

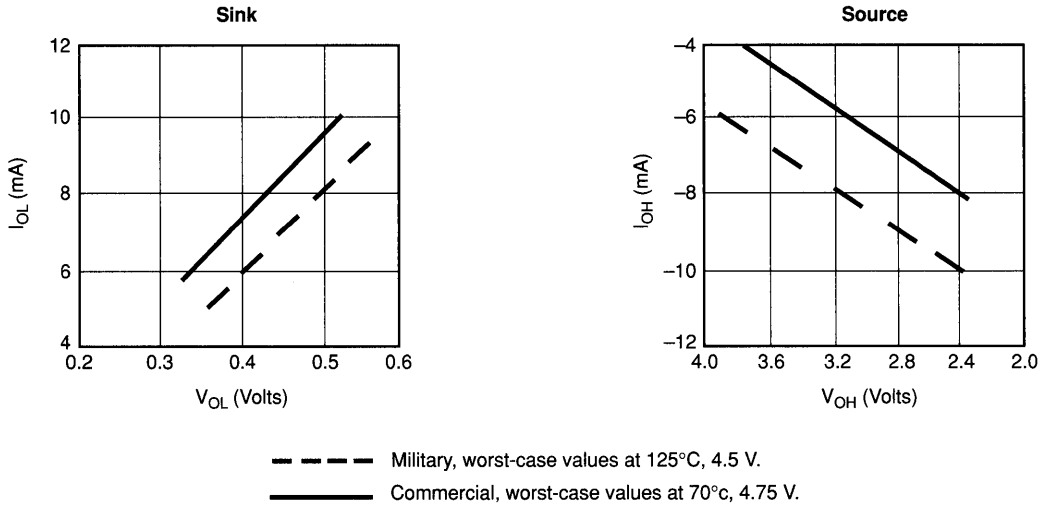
Functional Timing Tests

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

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

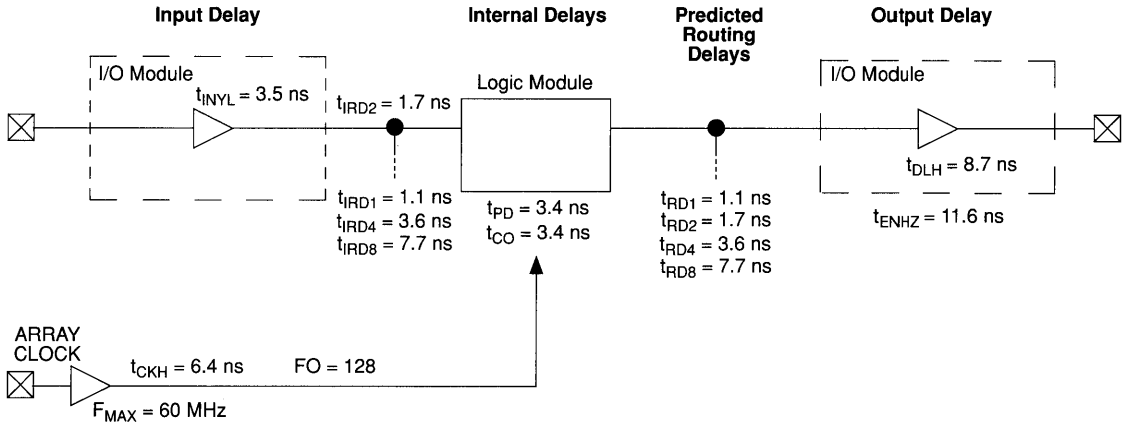
Propagation delay [$t_{PD} = (t_{PLH} + t_{PHL})/2$] is tested to the following AC test specifications.

Output Buffer Performance Derating



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

ACT 1 Timing Model*



*Values shown for ACT 1 '–2' speed devices at worst-case commercial conditions.

Predictable Performance: Tight Delay Distributions

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

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

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

Actel's patented PLICE antifuse offers a very low resistive/ capacitive interconnect. The ACT 1 family's antifuses, fabricated in 1.0 µm lithography, offer nominal levels of 500 ohms resistance and 7.5 femtofarad (fF) capacitance per antifuse.

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

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

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

Timing Characteristics

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

Critical Nets and Typical Nets

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

Long Tracks

Some nets in the design use long tracks. Long tracks are special routing resources that span multiple rows, columns, or modules. Long tracks employ three and sometimes four antifuse

connections. This increases capacitance and resistance, resulting in longer net delays for macros connected to long tracks. Typically, up to 6% of nets in a fully utilized device require long tracks. Long tracks contribute approximately 5 ns to 10 ns delay. This additional delay is represented statistically in higher fanout (FO=8) routing delays in the datasheet specifications section.

Timing Derating

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

Timing Derating Factor (Temperature and Voltage)

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

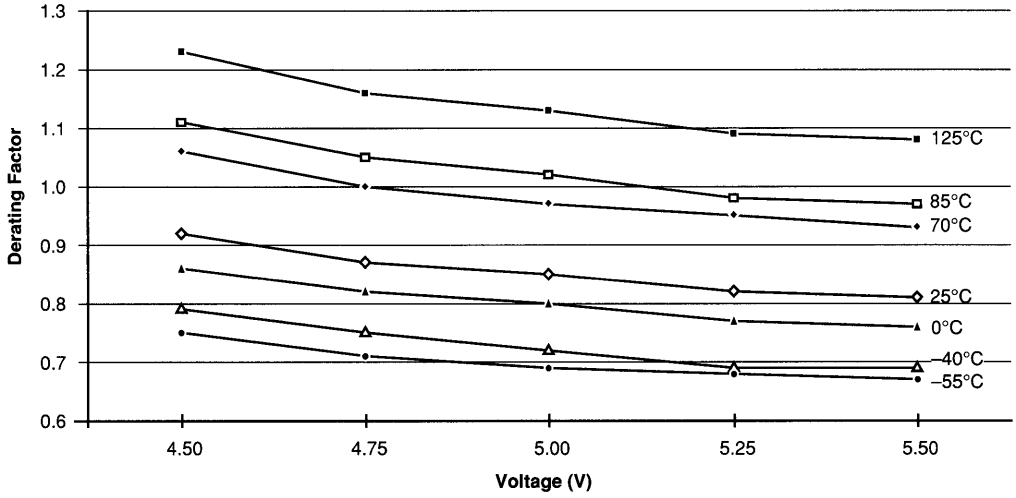
Timing Derating Factor for Designs at Typical Temperature (T_J = 25°C) and Voltage (5.0 V)

(Commercial Maximum Specification) x	0.85

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

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

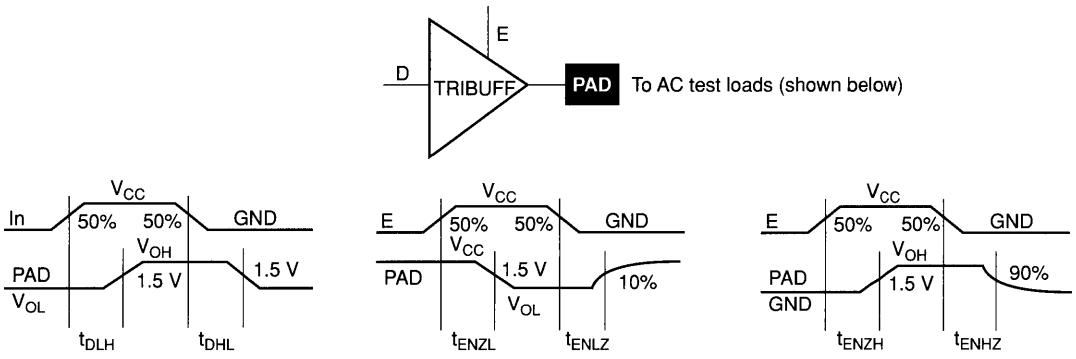
Junction Temperature and Voltage Derating Curves
(normalized to Worst-Case Commercial, T_J = 4.75 V, 70°C)



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

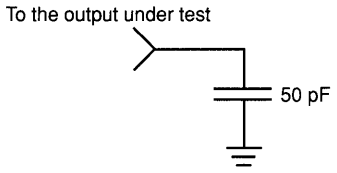
Parameter Measurement

Output Buffer Delays

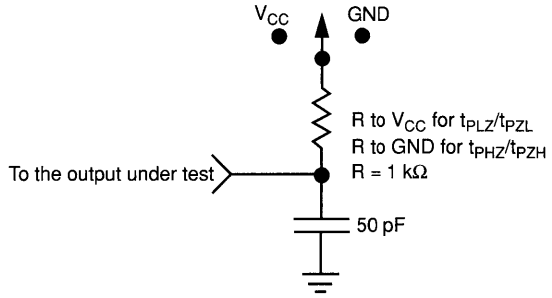


AC Test Loads

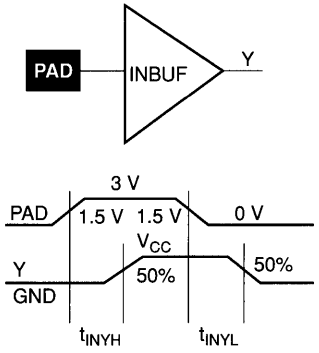
Load 1
(Used to measure propagation delay)



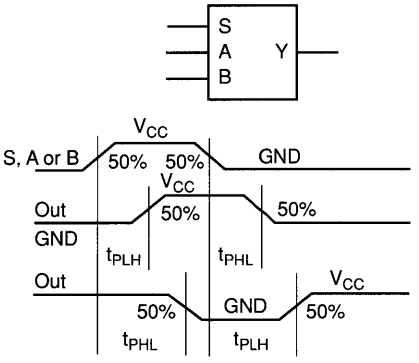
Load 2
(Used to measure rising/falling edges)



Input Buffer Delays

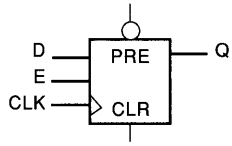


Module Delays

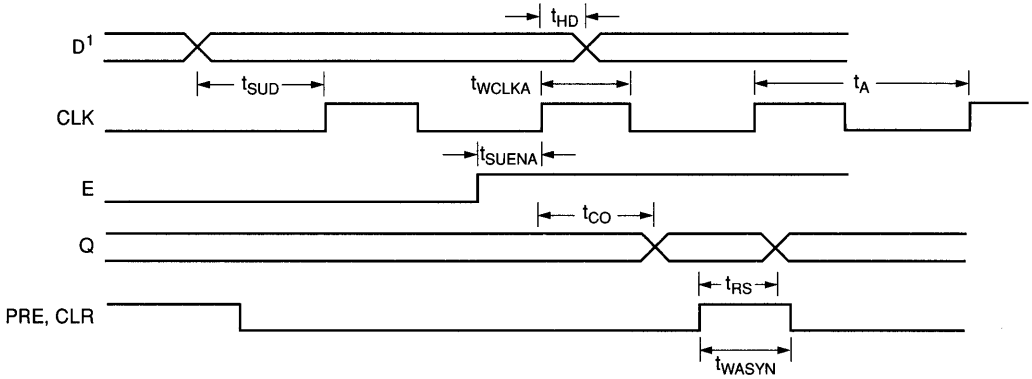


Sequential Timing Characteristics

Flip-Flops and Latches



(Positive edge triggered)



Note:

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

ACT 1 Timing Characteristics

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

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

Notes:

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

ACT 1 Timing Characteristics (continued)

(Worst-Case Commercial Conditions)

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

Note:

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

ACT 1Timing Characteristics (continued)

(Worst-Case Commercial Conditions)

Output Module Timing		'Std' Speed		'–1' Speed		'–2' Speed		
Parameter	**Description**	**Min.**	**Max.**	**Min.**	**Max.**	**Min.**	**Max.**	**Units**
TTL Output Module Timing[1]								
t_{DLH}	Data to Pad High		11.6		9.9		8.7	ns
t_{DHL}	Data to Pad Low		13.3		11.3		10.0	ns
t_{ENZH}	Enable Pad Z to High		11.5		9.8		8.6	ns
t_{ENZL}	Enable Pad Z to Low		14.0		11.9		10.5	ns
t_{ENHZ}	Enable Pad High to Z		15.4		13.1		11.6	ns
t_{ENLZ}	Enable Pad Low to Z		13.9		11.8		10.4	ns
d_{TLH}	Delta Low to High		0.09		0.08		0.07	ns/pF
d_{THL}	Delta High to Low		0.12		0.10		0.09	ns/pF
CMOS Output Module Timing[1]								
t_{DLH}	Data to Pad High		14.5		12.3		10.9	ns
t_{DHL}	Data to Pad Low		11.1		9.4		8.3	ns
t_{ENZH}	Enable Pad Z to High		11.5		9.8		8.6	ns
t_{ENZL}	Enable Pad Z to Low		14.0		11.9		10.5	ns
t_{ENHZ}	Enable Pad High to Z		15.4		13.1		11.6	ns
t_{ENLZ}	Enable Pad Low to Z		13.9		11.8		10.4	ns
d_{TLH}	Delta Low to High		0.15		0.13		0.11	ns/pF
d_{THL}	Delta High to Low		0.09		0.08		0.07	ns/pF

Note:
1. Delays based on 50 pF loading.

Macro Library

Hard Macros—Combinatorial

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

Hard Macros—Combinatorial (Continued)

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

Hard Macros—Combinatorial (Continued)

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

Hard Macros—Sequential

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

Hard Macros—Sequential (Continued)

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

Hard Macros—Sequential (Continued)

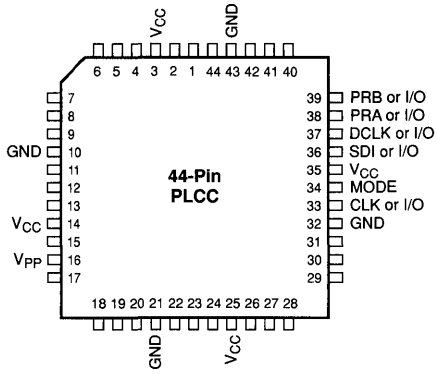
Function	Macro	Description	Modules C
Latch	DLE3A	Sequential, Data Latch with Enable	1
	DLE3B	Data Latch with active low Enable and clock and active low Preset	1
	DLE3C	Data Latch with active low Enable Preset and clock	1
	DLEA	Data Latch with active low Enable and active high clock	1
	DLEB	Data Latch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

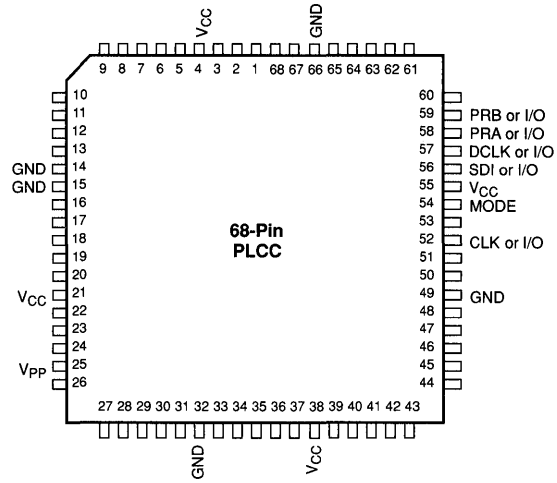
Input/Output Macros

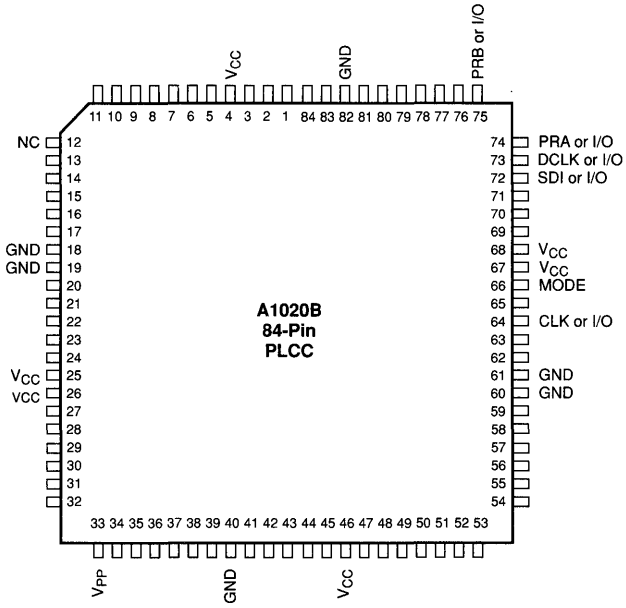
Function	Macro	Description	I/O Modules
Buffer	INBUF	Input Buffer	1
	OUTBUF	Output buffer, High Slew	1
Bidirectional	BIBUF	Bidirectional Buffer, High Slew (with hidden buffer at Y pin)	1
Input	CLKBUF	Input for Dedicated Routed Clock Network	1
Output	TRIBUFF	Tristate output, High Slew	1

Package Pin Assignments

(Top View)





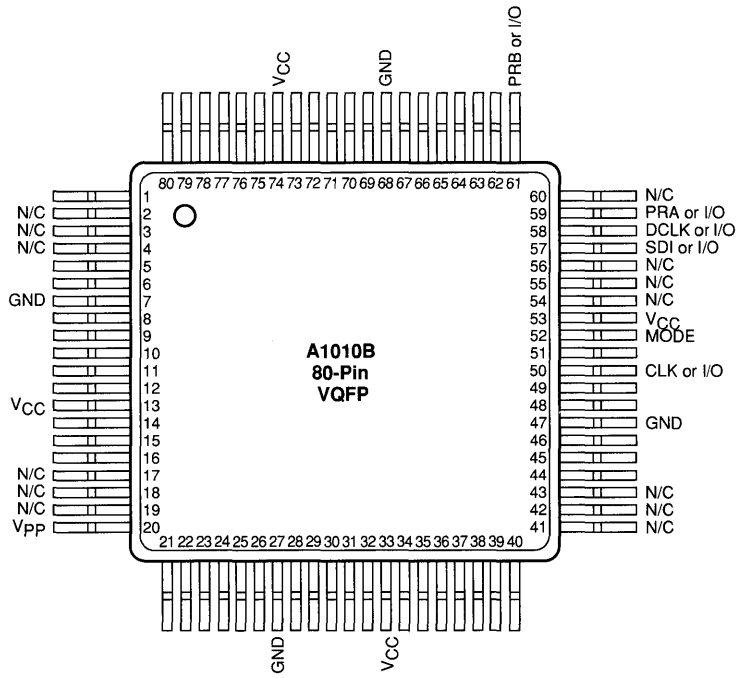


Notes:

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

Package Pin Assignments (continued)

(Top View)

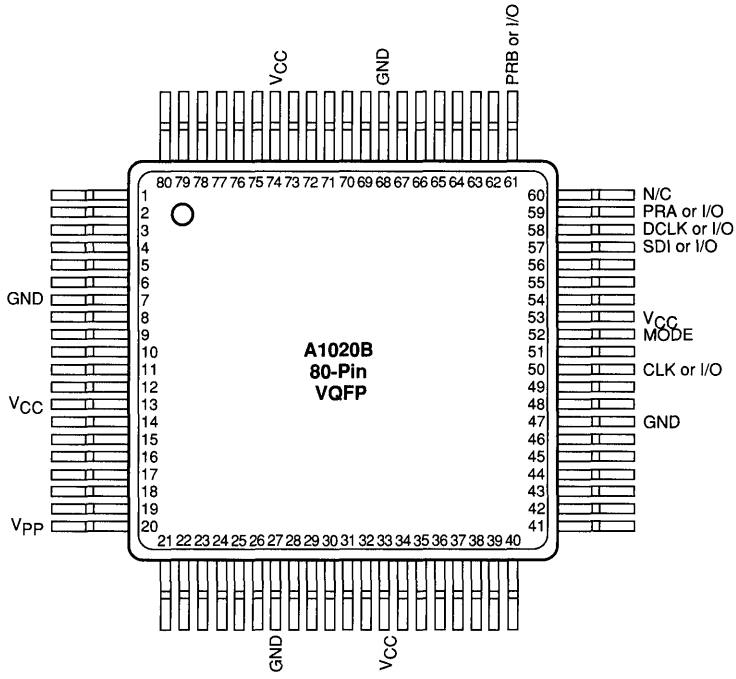


Notes:

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

Package Pin Assignments (continued)

(Top View)

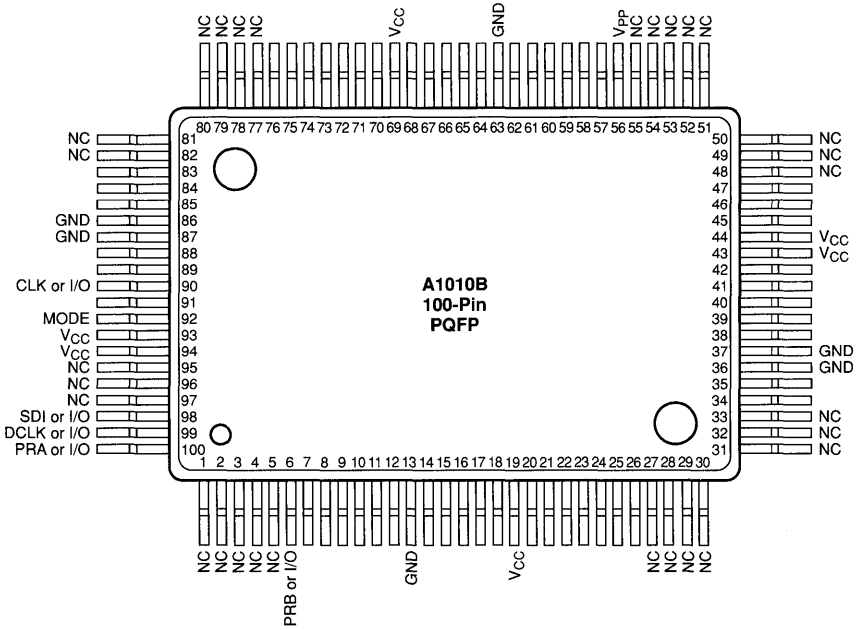


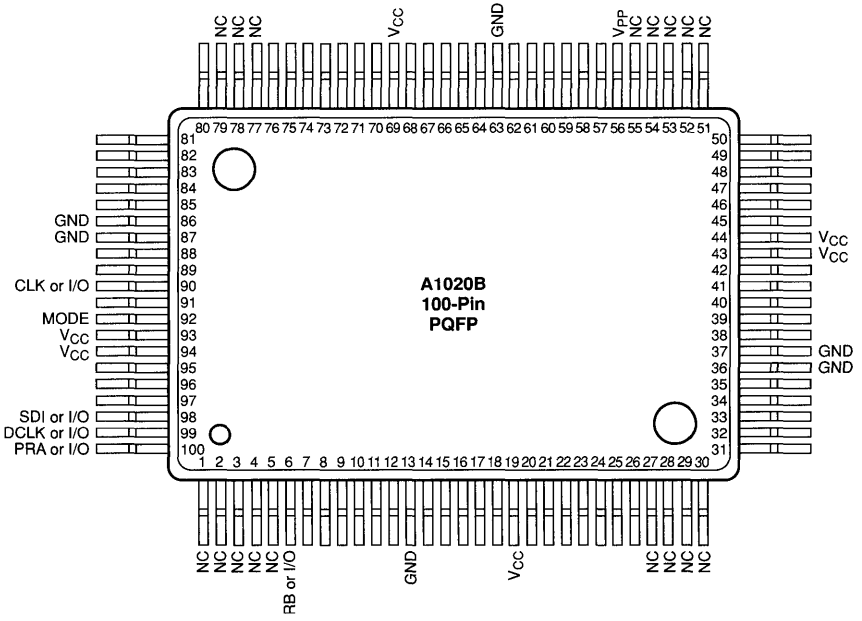
Notes:

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

Package Pin Assignments (continued)

(Top View)



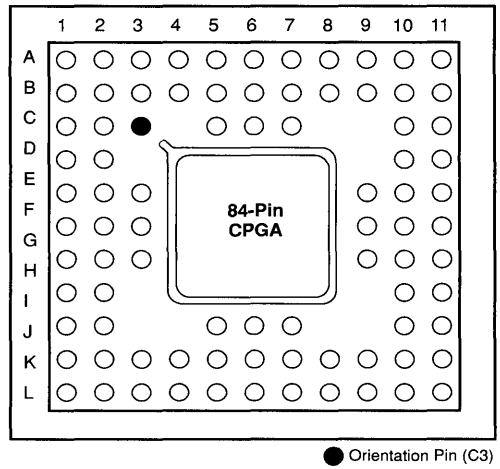


Notes:

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

Package Pin Assignments (continued)

(Top View)



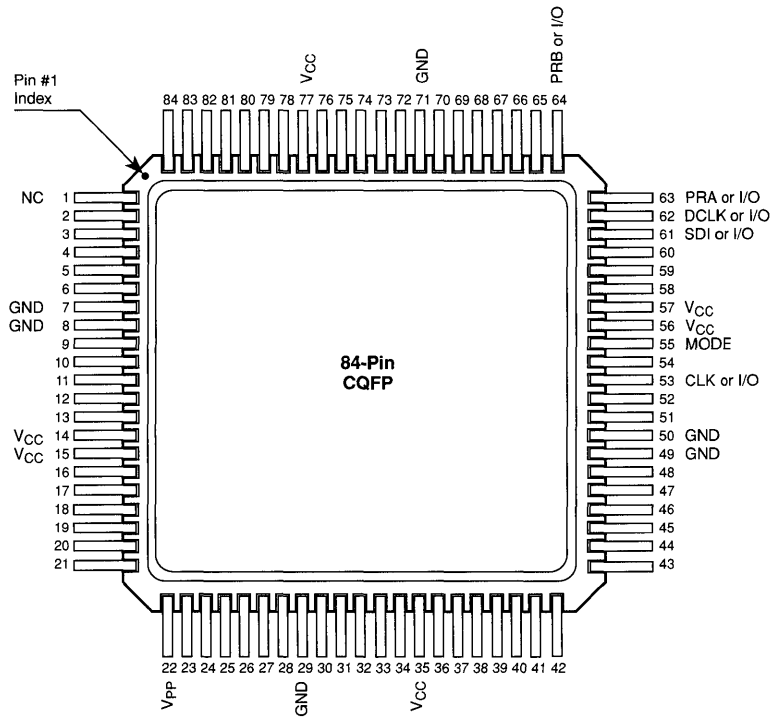
Signal	A1010B Devices	A1020B Devices
PRA	A11	A11
PRB	B10	B10
MODE	E11	E11
SDI	B11	B11
DCKL	C10	C10
V_{PP}	K2	K2
CLK or I/O	F9	F9
GND	B7, E2, E3, K5, F10, G10	B7, E2, E3, K5, F10,G10
V_{CC}	B5, F1, G2, K7, E9, E10	B5, F1, G2, K7, E9, E10
N/C (No Connection)	B1, B2, C1, C2, K1, J2, J10, K10, K11, C11,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

Notes:

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

Package Pin Assignments (continued)

(Top View)



Notes:

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

ACT™ 2 Field Programmable Gate Arrays

Features

- Up to 8000 Gate Array Gates (20,000 PLD equivalent gates)
- Replaces up to 200 TTL Packages
- Replaces up to eighty 20-Pin PAL® Packages
- Design Library with over 500 Macro Functions
- Single-Module Sequential Functions
- Wide-Input Combinatorial Functions
- Up to 1232 Programmable Logic Modules
- Up to 998 Flip-Flops
- Datapath Performance at 105 MHz
- 16-Bit Accumulator Performance to 42 MHz
- Two In-Circuit Diagnostic Probe Pins Support Speed Analysis to 50 MHz
- Two High-Speed, Low-Skew Clock Networks
- I/O Drive to 10 mA
- Nonvolatile, User Programmable
- Logic Fully Tested Prior to Shipment

Description

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

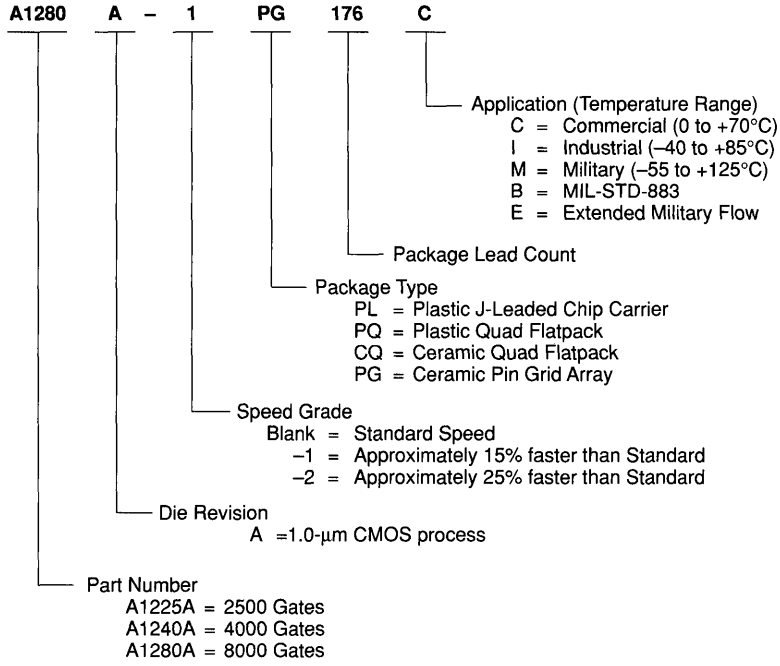
Product Family Profile

Device	A1225A	A1240A	A1280A
Capacity			
Gate Array Equivalent Gates	2,500	4,000	8,000
PLD Equivalent Gates	6,250	10,000	20,000
TTL Equivalent Packages	63	100	200
20-Pin PAL Equivalent Packages	25	40	80
Logic Modules	451	684	1,232
S-Modules	231	348	624
C-Modules	220	336	608
Flip-Flops (maximum)	382	568	998
Routing Resources			
Horizontal Tracks/Channel	36	36	36
Vertical Tracks/Channel	15	15	15
PLICE Antifuse Elements	250,000	400,000	750,000
User I/Os (maximum)	83	104	140
Packages[1]	100 CPGA 100 PQFP 84 PLCC	132 CPGA 144 PQFP 84 PLCC	176 CPGA 160 PQFP 172 CQFP
Performance[2]			
16-Bit Prescaled Counters	105 MHz	100 MHz	85 MHz
16-Bit Loadable Counters	66 MHz	63 MHz	59 MHz
16-Bit Accumulators	42 MHz	39 MHz	34 MHz
CMOS Process	1.0 µm	1.0 µm	1.0 µm

Note:

1. See product plan on page 1-33 for package availability.
2. Performance is based on '–2' speed devices at commercial worst-case operating conditions using PREP Benchmarks (mean frequency results), Suite #1, Version 1.2, dated 3-28-93, any analysis is not endorsed by PREP.

Ordering Information



Product Plan[1]

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

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

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

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

Note:

1. Please consult Actel representatives for current availability.

Device Resources

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

Pin Description

CLKA **Clock A (Input)**

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

CLKB **Clock B (Input)**

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

DCLK **Diagnostic Clock (Input)**

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

GND **Ground**

LOW supply voltage.

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

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

MODE **Mode (Input)**

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

NC **No Connection**

This pin is not connected to circuitry within the device.

PRA **Probe A (Output)**

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

PRB **Probe B (Output)**

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

SDI **Serial Data Input (Input)**

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

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

HIGH supply voltage.

V_{KS} **Programming Voltage**

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

V_{PP} **Programming Voltage**

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

V_{SV} **Programming Voltage**

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

ACT 2 Architecture

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

Array Topology

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

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

Table 1. Array Sizes

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

Logic Modules

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

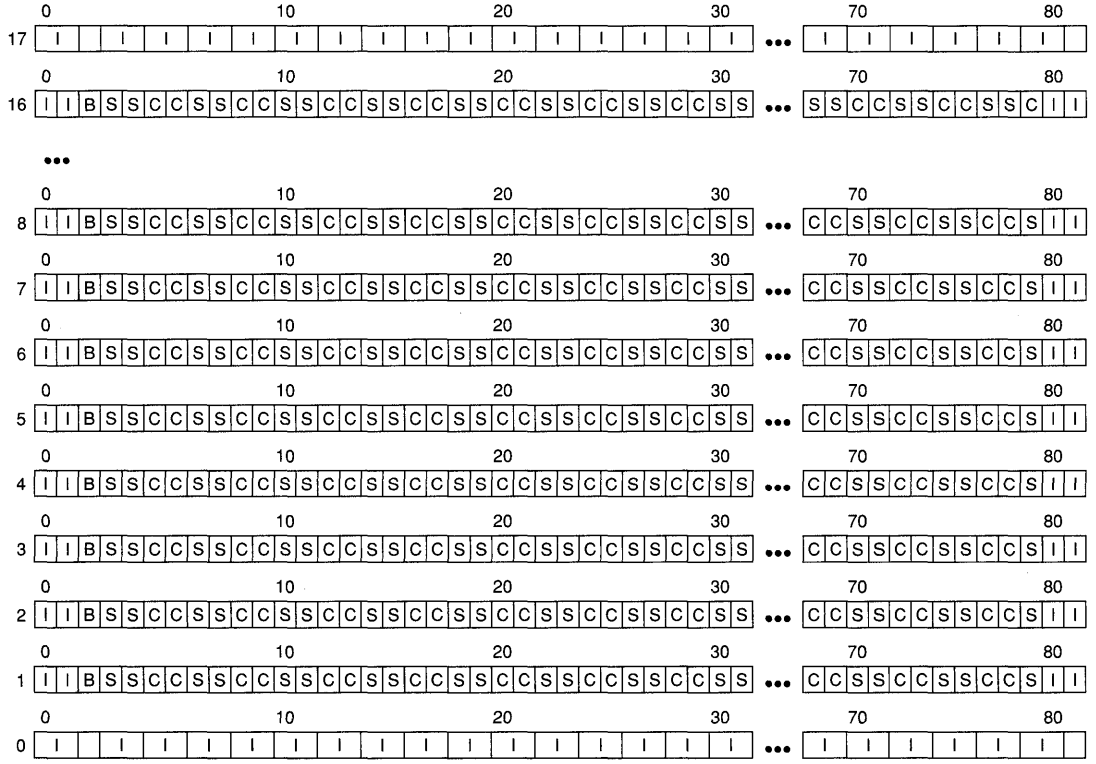


Figure 1. A1280A Simplified Floor Plan

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

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

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

where:

$S0 = A0 * B0$

$S1 = A1 + B1$

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

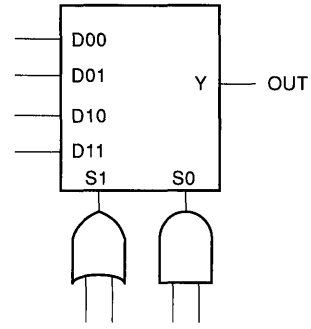
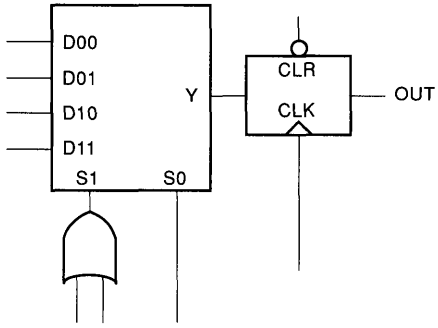
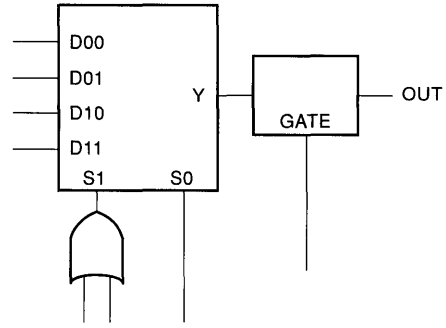


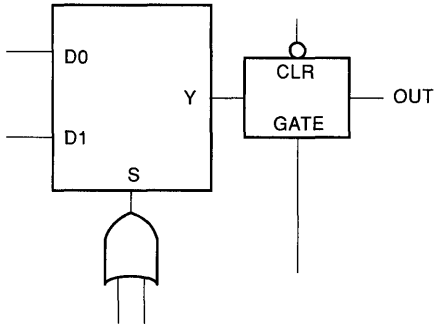
Figure 2. C-module Implementation



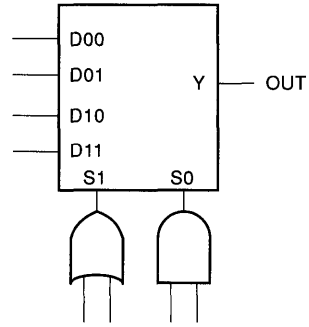
Up to 7-input function plus D-type flip-flop with clear



Up to 7-input function plus latch



Up to 4-input function plus latch with clear



Up to 8-input function (same as C-module)

Figure 3. S-module Implementations

I/Os

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

Special Purpose I/Os

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

Table 2. Special I/O Pads

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

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

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

I/O Pads

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

I/O Modules

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

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

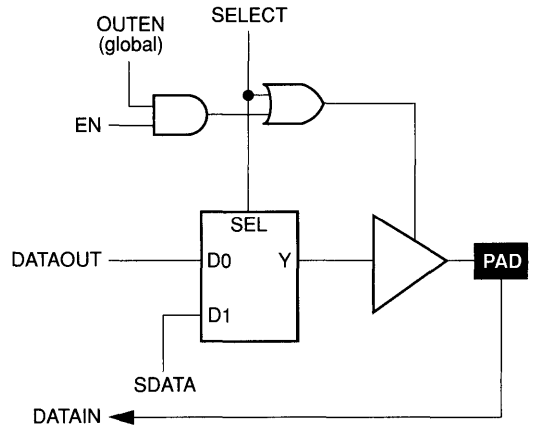


Figure 4. I/O Pad Signals

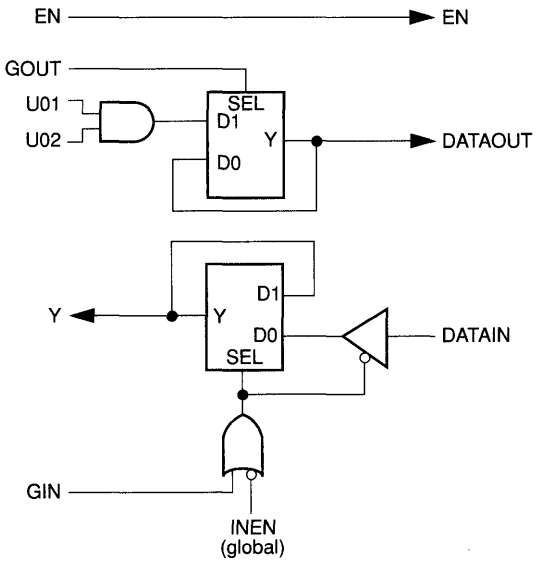


Figure 5. I/O Module

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

Hard Macros

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

Routing Structure

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

Horizontal Routing

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

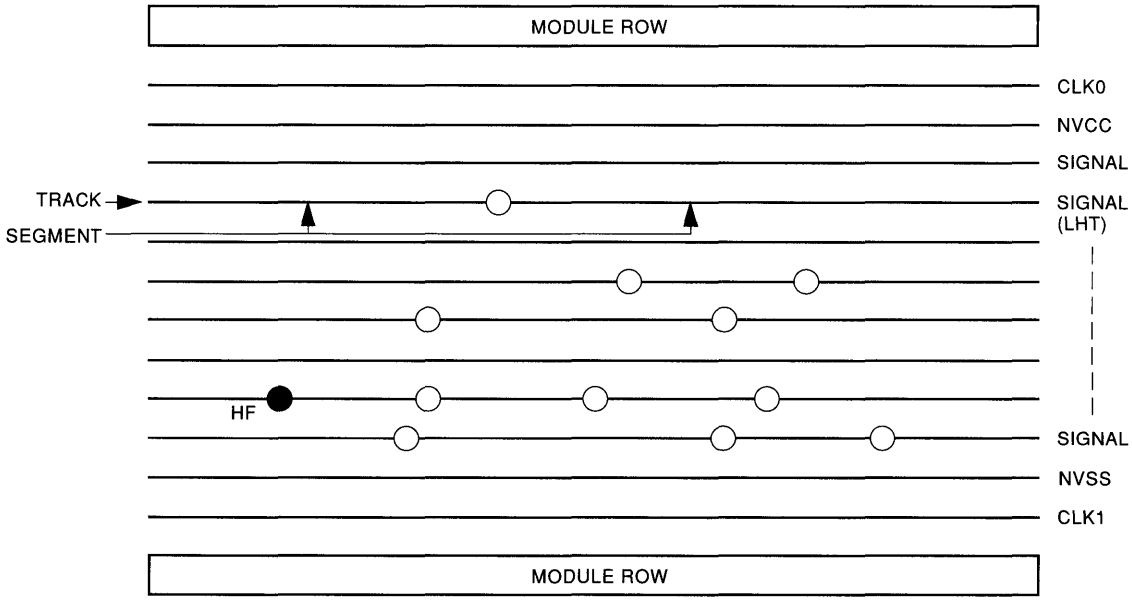


Figure 6. Horizontal Routing Tracks and Segments

Vertical Routing

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

Antifuse Structures

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

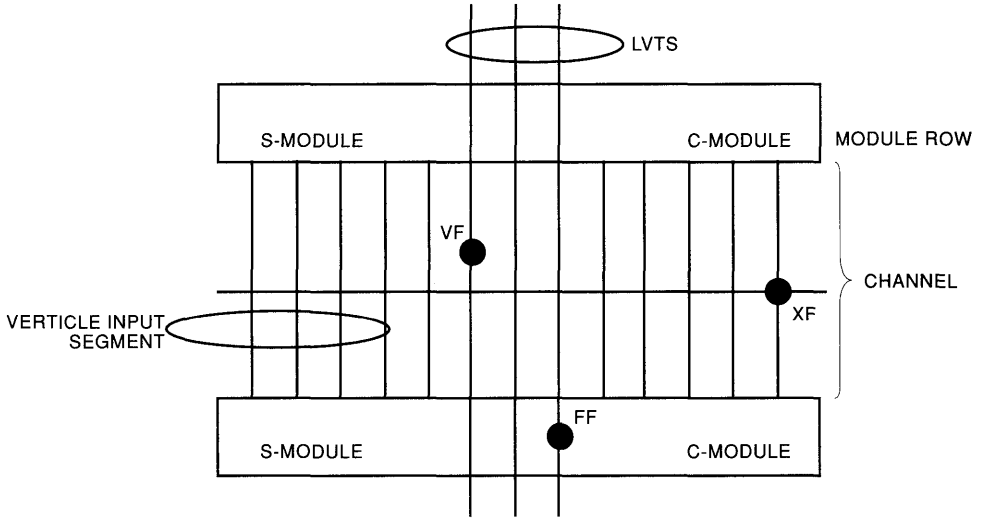


Figure 7. Vertical Routing Tracks and Segments

Antifuse Connections

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

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

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

Antifuse Programming

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

Clock Networks

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

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

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

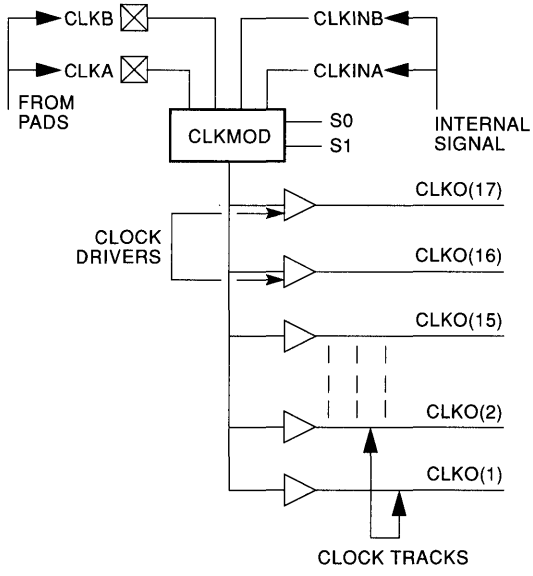


Figure 8. Clock Networks

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

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

Module Interface

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

Module Input Connections

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

Module Output Connections

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

LVT Connections

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

Antifuse Connections

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

Clock Connections

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

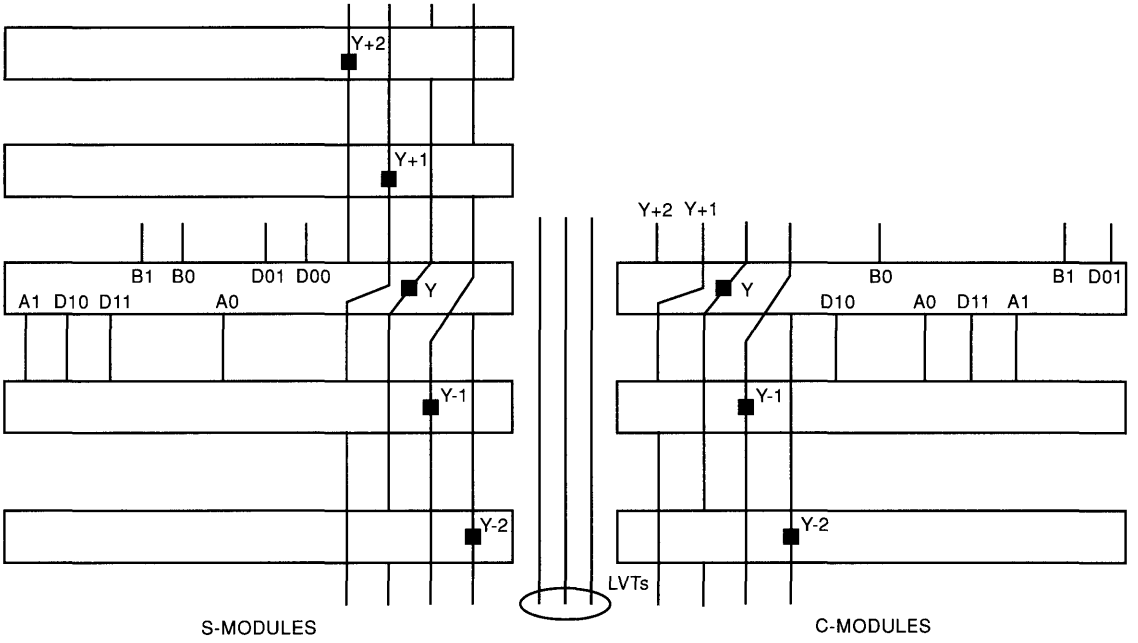


Figure 9. Logic Module Routing Interface

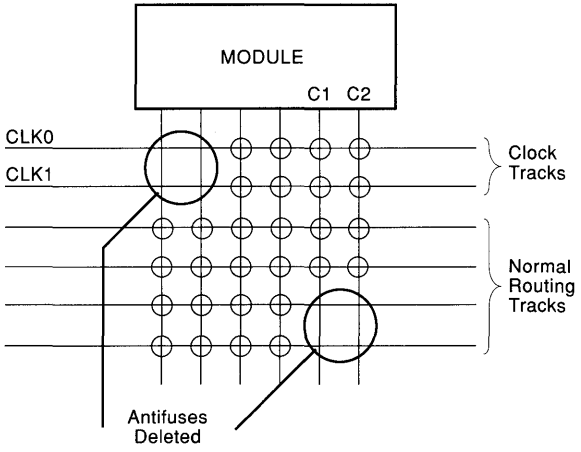


Figure 10. Fuse Deletion on Clock Networks

Programming and Test Circuits

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

Actionprobe

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

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

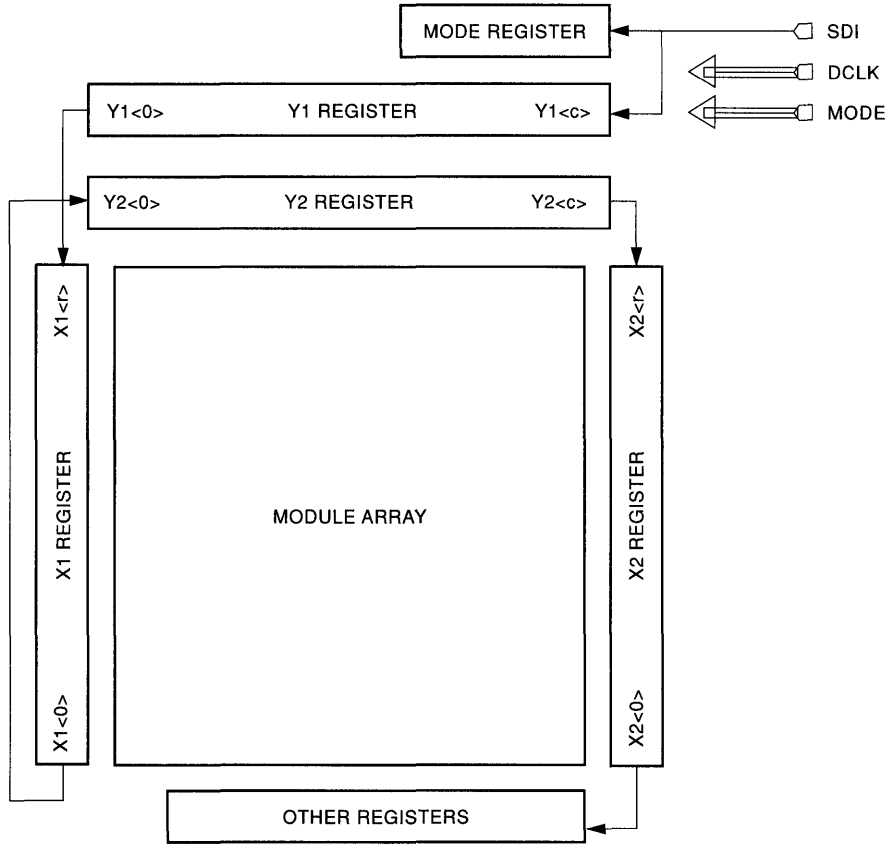


Figure 11. ACT 2 Shift Register

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

Table 3. Bit Stream Definitions for Actionprobe Diagnostics

Device	Probe_Mode	Filler (n)	Counter_Pattern	Mode_Register_Pattern	# of clocks
A1225A	Probe A only	308	1101011010	00000011000111110000	458
A1225A	Probe B only	308	1101011010	00000010100111110000	458
A1225A	Probe A and B	308	1101011010	00000011100111110000	458
A1240A	Probe A only	361	1111000001	00000011000111110000	545
A1240A	Probe B only	361	1111000001	00000010100111110000	545
A1240A	Probe A and B	361	1111000001	00000011100111110000	545
A1280A	Probe A only	443	0011011111	00000011000111110000	675
A1280A	Probe B only	443	0011011111	00000010100111110000	675
A1280A	Probe A and B	443	0011011111	00000011100111110000	675

For example: Selecting PRA for A1280 results in the following bit stream.
0011011111_00000011000111110000_
(433 zeros)_X2<0>...X2<17>_Y2<81>...Y2<0>_X1<0>...X1<0>...X1<17>_Y1<0>...Y1<81>_0,
where "_" is used for clarity only

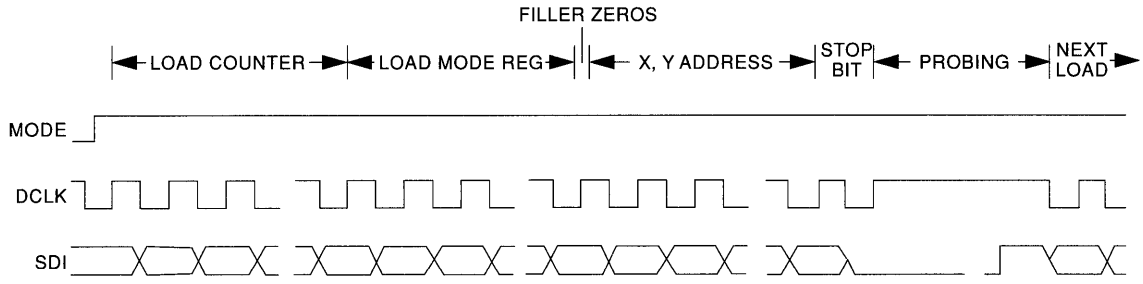


Figure 12. Timing Waveforms

Absolute Maximum Ratings[1]

Free air temperature range

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

Notes:

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

Recommended Operating Conditions

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

Note:

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

Electrical Specifications

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

Notes:

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

Package Thermal Characteristics

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

Maximum junction temperature is 150°C.

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

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

Package Type	Pin Count	θjc	θja Still Air	θja 300 ft/min	Units
Ceramic Pin Grid Array	100	5	35	17	°C/W
	132	5	30	15	°C/W
	176	8	23	12	°C/W
Ceramic Quad Flatpack	172	8	25	15	°C/W
Plastic Quad Flatpack[1]	100	13	55	47	°C/W
	144	15	35	26	°C/W
	160	15	33	24	°C/W
Plastic Leaded Chip Carrier[2]	84	12	44	33	°C/W

Notes:
1. Maximum Power Dissipation for PQFP packages is 2.0 Watts.
2. Maximum Power Dissipation for PLCC packages is 1.5 Watts.

Power Dissipation

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

Where:

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

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

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

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

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

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

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

Static Power

Static power dissipation is typically a small component of the overall power. From the values provided in the Electrical Specifications, the maximum static power (commercial) dissipation is:

2 mA * 5.25 V = 10.5 mW

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

Active Time

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

Equivalent Capacitance

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

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

Where:

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

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

f is the switching frequency in megahertz (MHz).

Equivalent capacitance is calculated by measuring I_{active} at a specified frequency and voltage for each circuit component of interest. The results for ACT 2 devices are:

	C_{EQ} (pF)
Modules	7.7
Input Buffers	18.0
Output Buffers	25.0
Clock Buffer Loads	2.5

To calculate the active power dissipated from the complete design, you must solve Equation 1 for each component. To do this, you must know the switching frequency of each part of the logic. The exact equation is a piece-wise linear summation over all components, as shown in Equation 2.

$$\text{Power } (\mu W) = [(m * 7.7 \times f_1) + (n * 18.0 \times f_2) + (p * (25.0 + C_L) * f_3) + (q \times 2.5 \times f_4)] * V_{CC}^2 \quad (2)$$

Where:

m = Number of logic modules switching at frequency f_1

n = Number of input buffers switching at frequency f_2

p = Number of output buffers switching at frequency f_3

q = Number of clock loads on the global clock network

f_1 = Average logic module switching rate in MHz

f_2 = Average input buffer switching rate in MHz

f_3 = Average output buffer switching rate in MHz

f_4 = Frequency of global clock

C_L = Output load capacitance in pF

Determining Average Switching Frequency

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

Module Utilization = 80% of combinatorial modules

Average Module Frequency = F/10

Inputs = 1/3 of I/O

Average Input Frequency = F/5

Outputs = 2/3 of I/Os

Average Output Frequency = F/10

Clock Net 1 Loading = 40% of sequential modules

Clock Net 1 Frequency = F

Clock Net 2 Loading = 40% of sequential modules

Clock Net 2 Frequency = F/2

Estimated Power

The results of estimating active power are displayed in Figure 13. The graphs provide a simple guideline for estimating power. The tables may be interpolated when your application has different resource utilizations or frequencies.

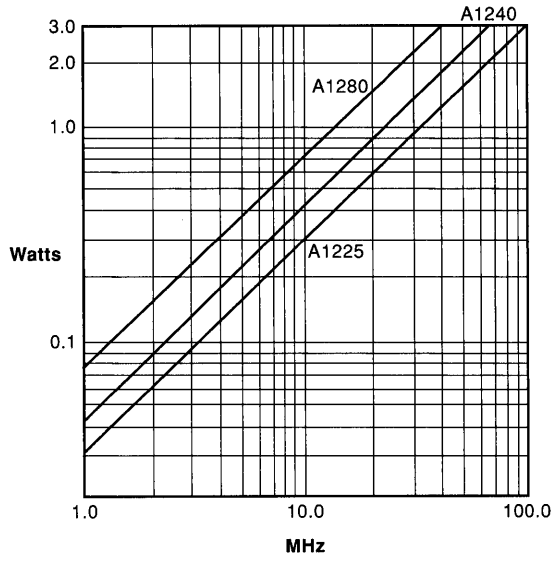
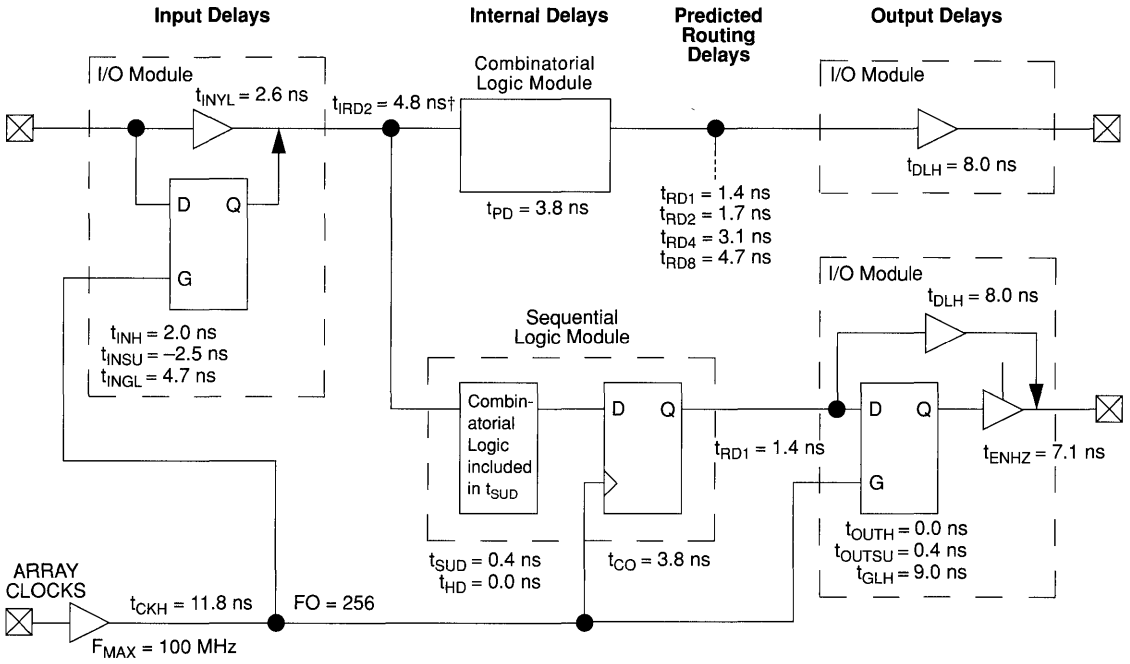


Figure 13. ACT 2 Power Estimates

ACT 2 Timing Model*

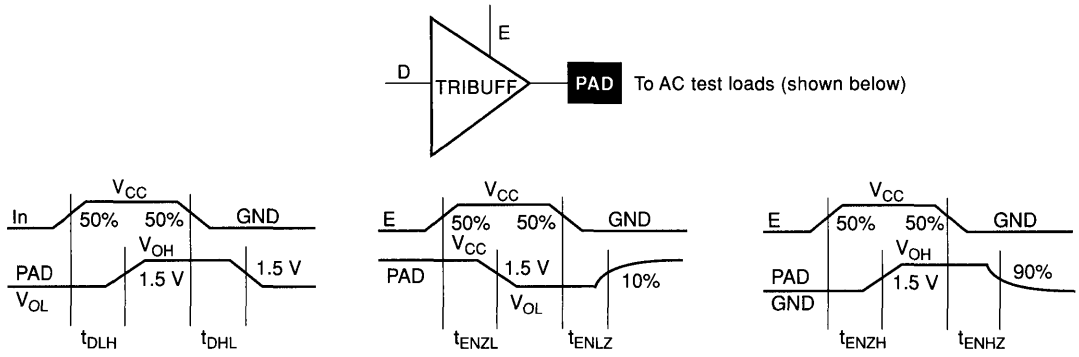


*Values shown for A1240A-2 at worst-case commercial conditions.

† Input Module Predicted Routing Delay

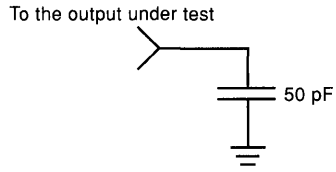
Parameter Measurement

Output Buffer Delays

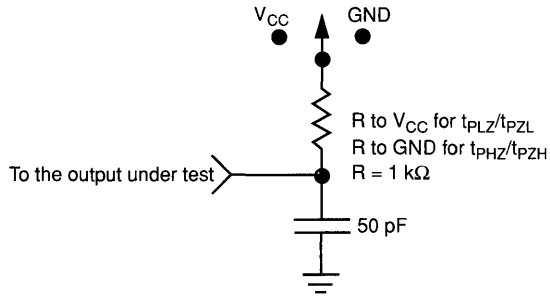


AC Test Loads

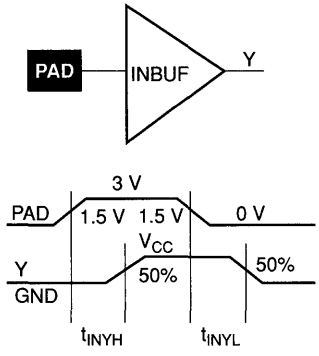
Load 1
(Used to measure propagation delay)



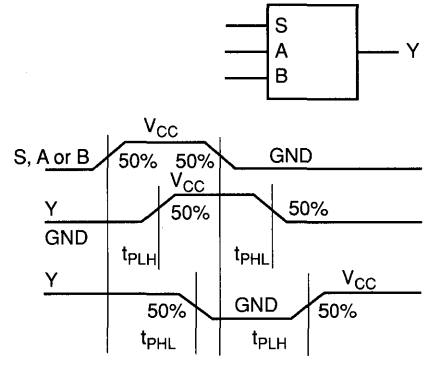
Load 2
(Used to measure rising/falling edges)



Input Buffer Delays

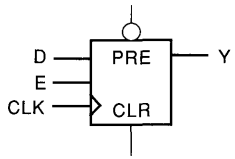


Combinatorial Macro Delays

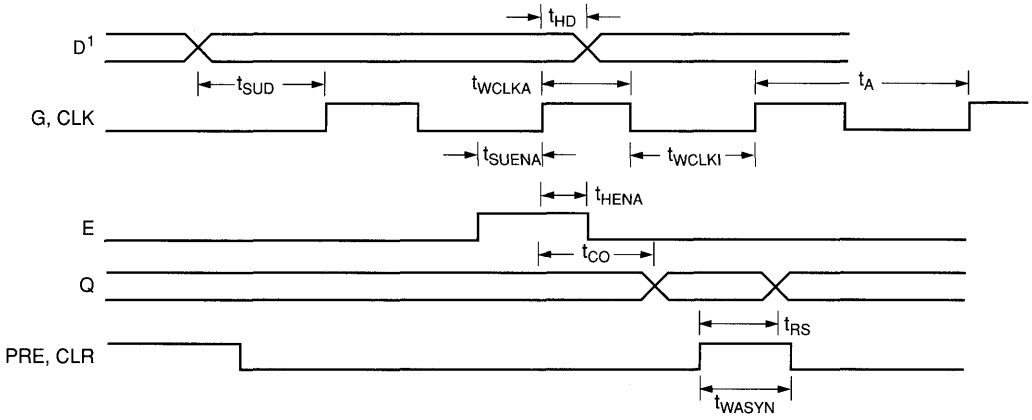


Sequential Timing Characteristics

Flip-Flops and Latches



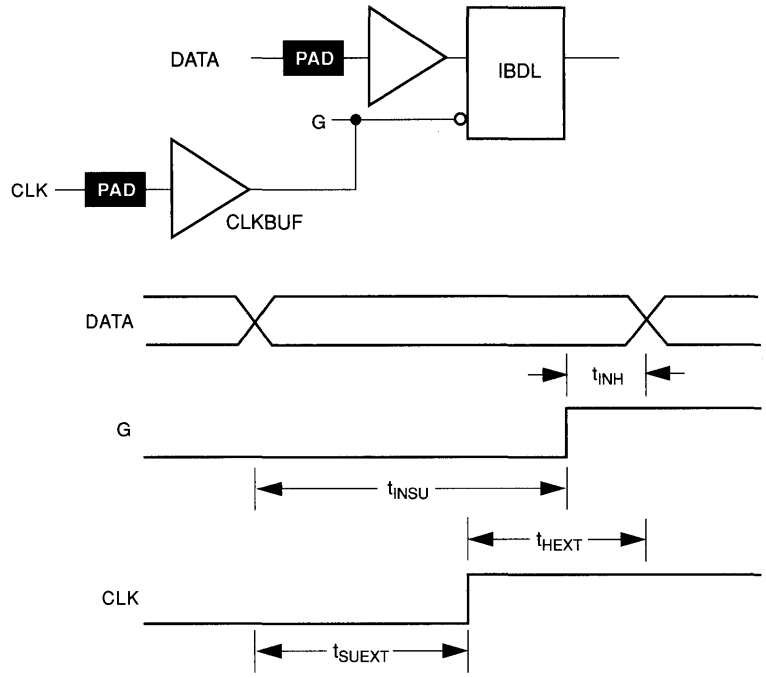
(Positive edge triggered)



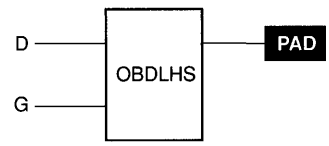
Note:
1. D represents all data functions involving A, B, and S for multiplexed flip-flops.

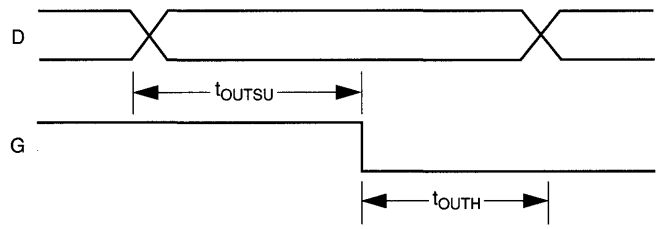
Sequential Timing Characteristics (continued)

Input Buffer Latches



Output Buffer Latches





Predictable Performance: Tight Delay Distributions

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

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

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

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

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

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

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

Timing Characteristics

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

Critical Nets and Typical Nets

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

Long Tracks

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

Timing Derating

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

Timing Derating Factor (Temperature and Voltage)

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

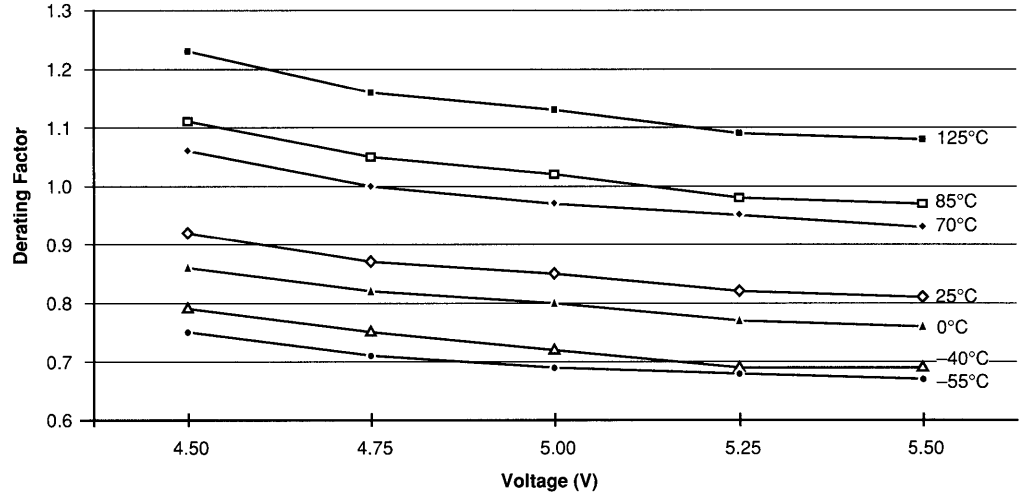
Timing Derating Factor for Designs at Typical Temperature (T_J = 25°C) and Voltage (5.0 V)

(Commercial Maximum Specification) x	0.85

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

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

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



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

A1225A Timing Characteristics

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

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

Notes:

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

A1225A Timing Characteristics (continued)

(Worst-Case Commercial Conditions)

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

Note:

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

A1225A Timing Characteristics (continued)
(Worst-Case Commercial Conditions)

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

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

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

A1240A Timing Characteristics

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

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

Notes:

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

A1240A Timing Characteristics (continued)

(Worst-Case Commercial Conditions)

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

Note:

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

A1240A Timing Characteristics (continued)

(Worst-Case Commercial Conditions)

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

Notes:

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

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

A1280A Timing Characteristics

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

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

Notes:

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

A1280A Timing Characteristics (continued)

(Worst-Case Commercial Conditions)

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

Note:

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

A1280A Timing Characteristics (continued)

(Worst-Case Commercial Conditions)

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

Notes:

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

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

Macro Library

Hard Macros—Combinatorial

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

Hard Macros—Combinatorial (continued)

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

Hard Macros—Combinatorial (continued)

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

Hard Macros—Sequential

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

Hard Macros—Sequential (continued)

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

Input/Output Macros

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

Soft Macros

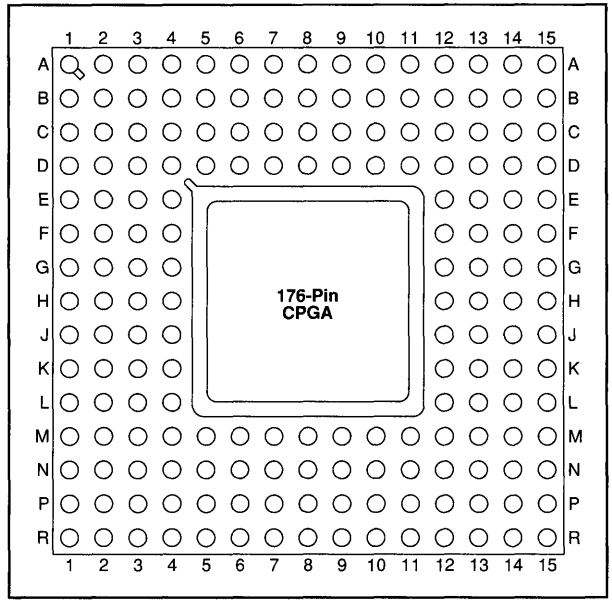
Function	Macro	Description	Maximum Logic Levels	Modules S	Modules C
Adder	FADD10	10-bit adder	3		56
	FADD12	12-bit adder	4		9
	FADD16	16-bit adder	5		97
	FADD8	8-bit adder	4		44
	FADD9	9-bit adder with active low carry out	3		49
	VAD16C	Very fast 16-bit adder, no Carry in	3		91
	VADC16C	Very fast 16-bit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

Soft Macros—TTL Equivalent

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

Package Pin Assignments

176-Pin CPGA (Top View)



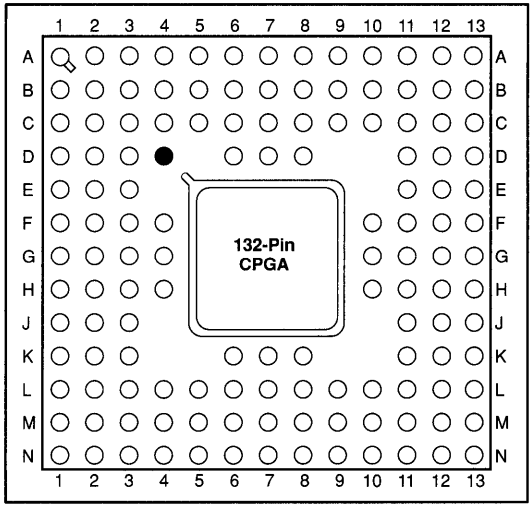
Signal	Pad Number	Location
PRA or I/O	152	C9
PRB or I/O	160	D7
MODE	2	C3
SDI or I/O	135	B14
DCLK or I/O	175	B3
CLKA or I/O	154	A9
CLKB or I/O	158	B8
GND	1, 8, 18, 23, 33, 38, 45, 57, 67, 77, 89 101, 106, 111, 121, 126, 133, 145, 156, 165	D4, E4, G4, H4, K4, L4, M4, M6, M8, M10, M12 K12, J12, H12, F12,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

Notes:

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

Package Pin Assignments (continued)

132-Pin CPGA (Top View)

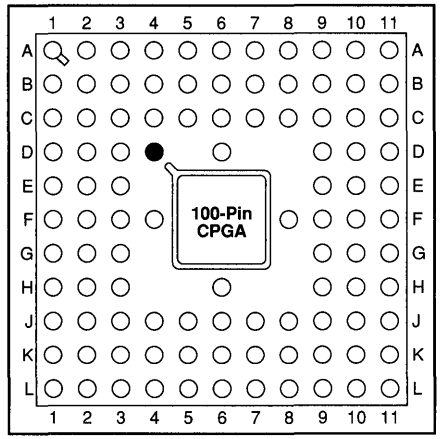


Signal	Pad Number	Location
PRA or I/O	113	B8
PRB or I/O	121	C6
MODE	2	A1
SDI or I/O	101	B12
DCLK or I/O	132	C3
CLKA or I/O	115	B7
CLKB or I/O	119	B6
GND	9, 10, 26, 27, 41, 58, 59, 73, 74, 92, 93, 107, 108, 125, 126	E3, F4, J2, J3, L5, L9, M9, K12, J11, E12, E11, C9, B9, B5, C5
V_{CC}	18, 19, 49, 50,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

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

Package Pin Assignments (continued)

100-Pin CPGA (Top View)



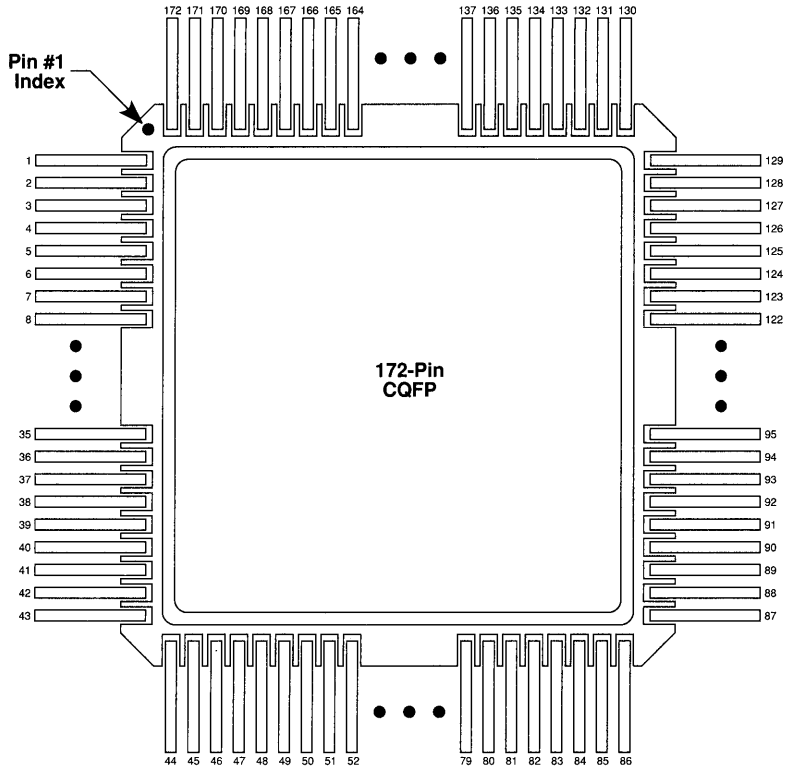
Signal	Pad Number	Location
PRA or I/O	85	A7
PRB or I/O	92	A4
MODE	2	C2
SDI or I/O	77	C8
DCLK or I/O	100	C3
CLKA or I/O	87	C6
CLKB or I/O	90	D6
GND	7, 20, 32, 44, 55, 70, 82, 94	E3, G3, J5, J7, G9, D10, C7, C5
V_{CC}	15, 38, 64, 88	F3, K6, F9, B6
V_{PP}	63	F10
V_{SV}	14, 65	G1, E11
V_{KS}	62	F11

Notes:

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

Package Pin Assignments (continued)

172-Pin CQFP (Top View)



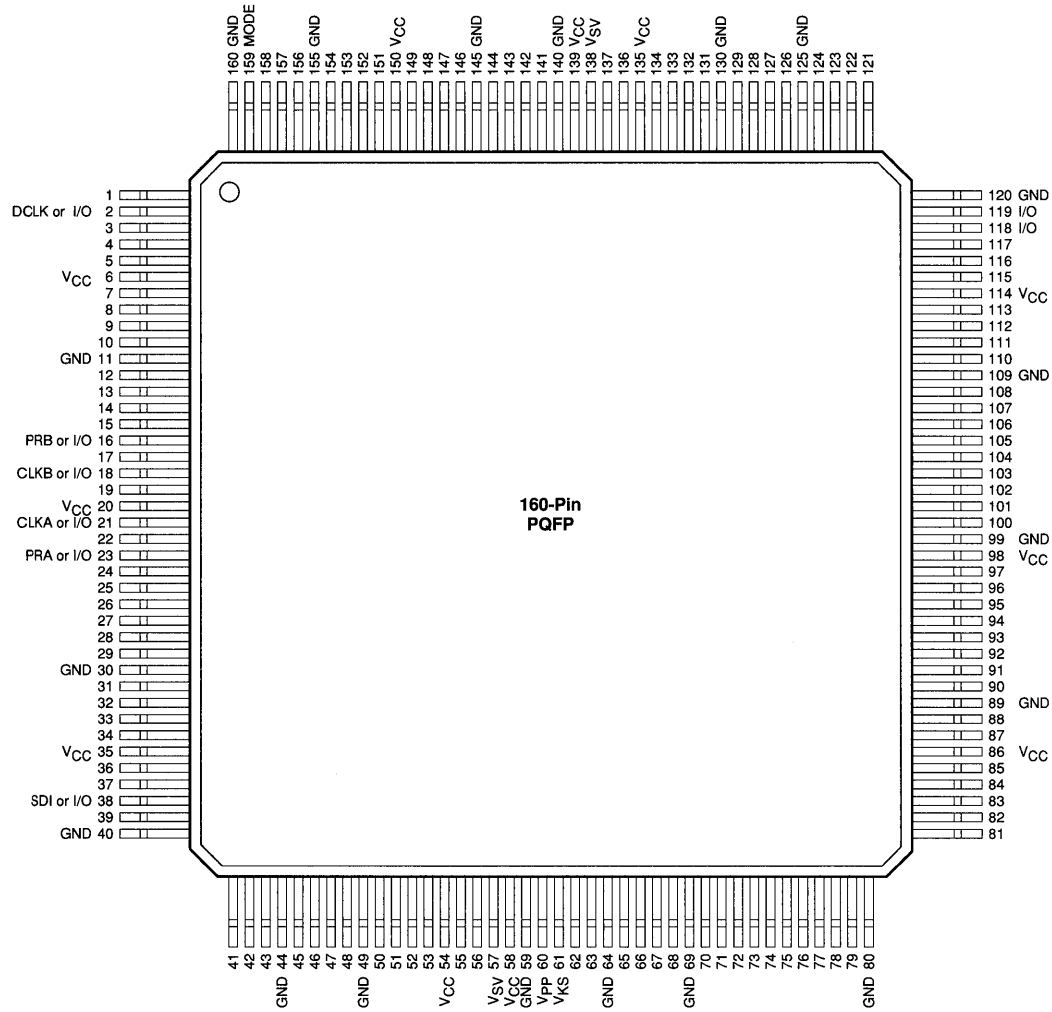
Signal	PIN Number
MODE	1
GND	7, 17, 22, 32, 37, 55, 65, 75, 98, 103, 108, 118, 123, 141, 152, 161
V_{CC}	12, 23, 27, 50, 66, 80, 109, 113, 136, 151, 166
V_{SV}	24, 110
V_{KS}	106
V_{PP}	107
SDI or I/O	131
PRA or I/O	148
PRB or I/O	156
CLKA or I/O	150
CLKB or I/O	154
DCLK or I/O	171

Notes:

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

Package Pin Assignments (continued)

160-Pin PQFP (Top View)

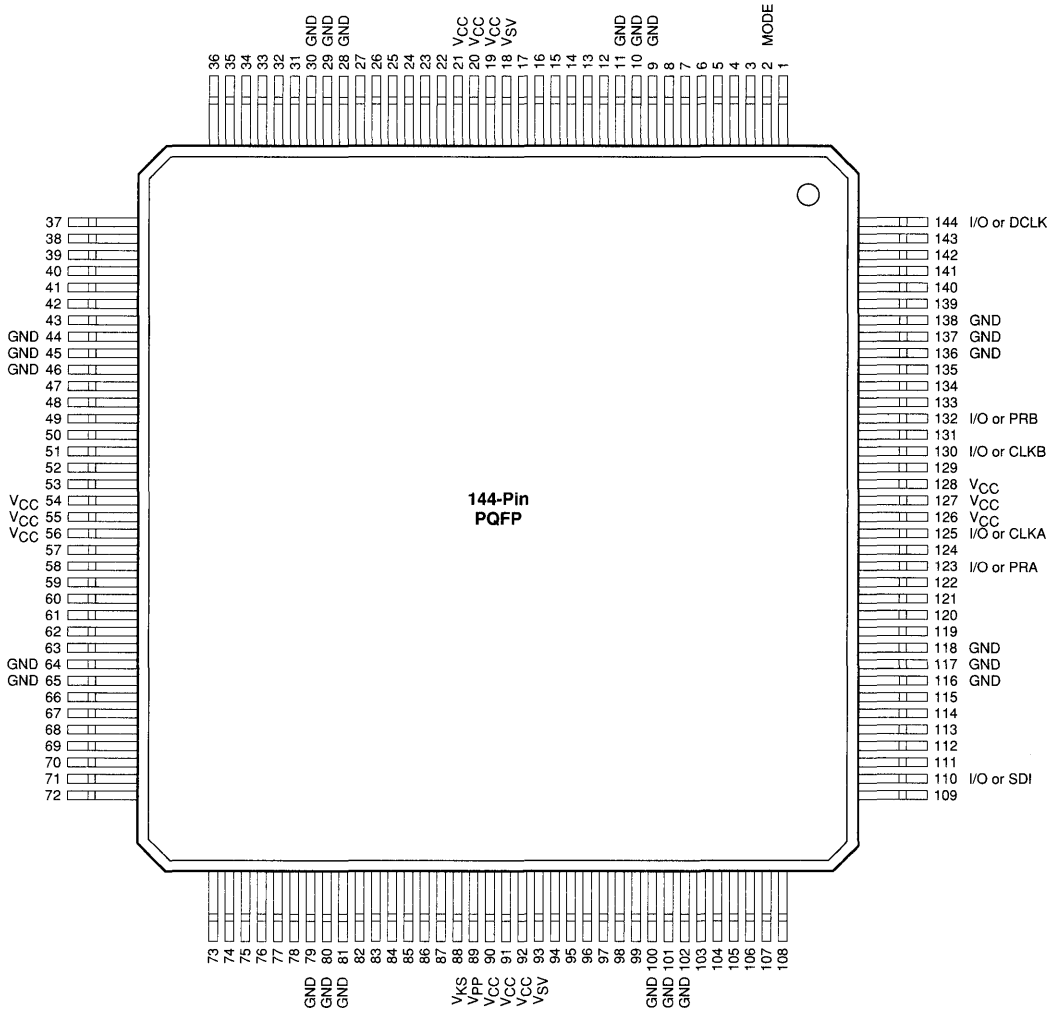


Notes:

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

Package Pin Assignments (continued)

144-Pin PQFP (Top View)

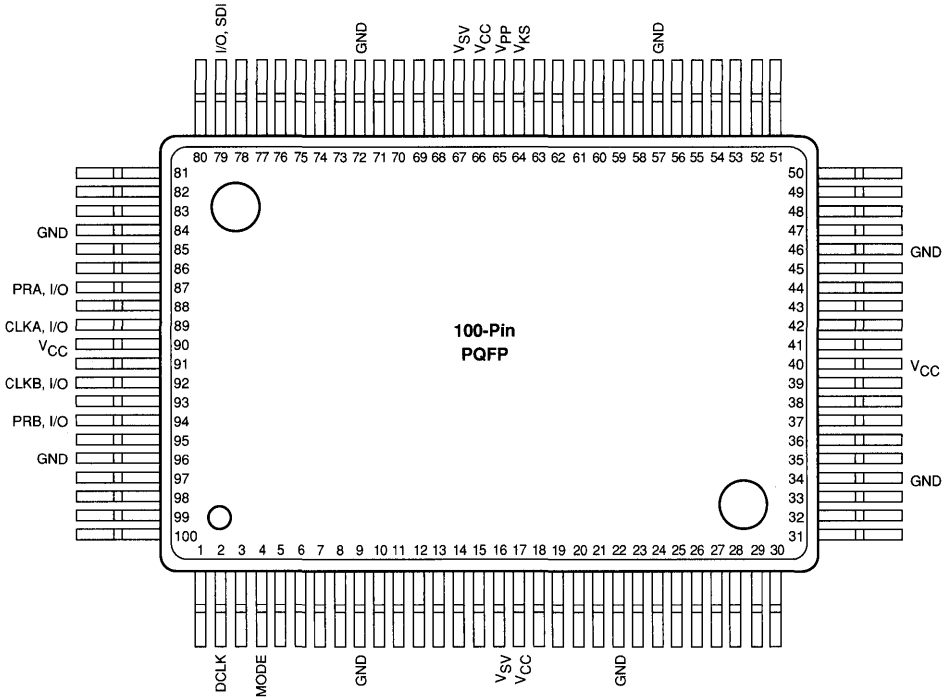


Notes:

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

Package Pin Assignments (continued)

100-Pin PQFP (Top View)

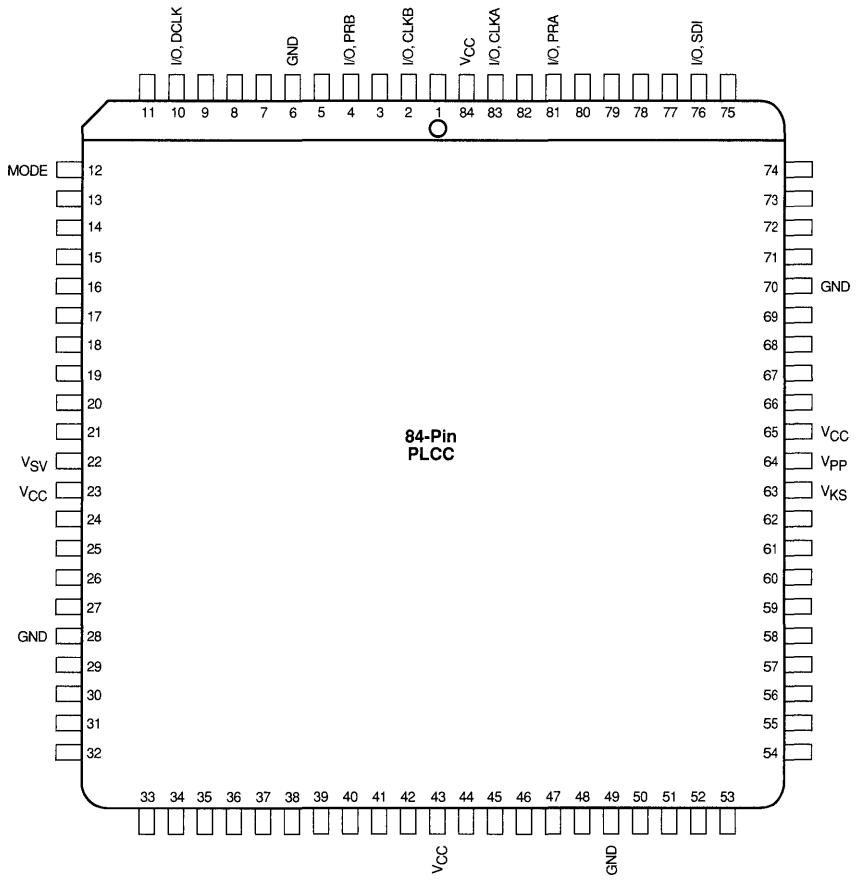


Notes:

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

Package Pin Assignments (continued)

84-Pin PLCC (Top View)



Notes:

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

ACT™ 3 Field Programmable Gate Arrays

Preliminary

Features

- Highly Predictable Performance with 100% Automatic Placement and Routing
- 9 ns Clock-to-Output Times
- Up to 150 MHz On-Chip Performance
- Up to 228 User-Programmable I/O Pins
- Four Fast, Low-Skew Clock Networks
- More Than 500 Macro Functions
- Up to 10,000 Gate Array Equivalent Gates (up to 25,000 equivalent PLD Gates)
- Replaces up to 250 TTL Packages
- Replaces up to 100 20-pin PAL® Packages
- Up to 1153 Dedicated Flip-Flops
- I/O Drive to 12 mA
- PQFP, PLCC, and CPGA Packages
- Nonvolatile, User Programmable
- Low-power 0.8 µm CMOS Technology
- Fully Tested Prior to Shipment

Description

The ACT 3 family, based on Actel's proprietary PLICE® antifuse technology and 0.8-micron double-metal, double-poly CMOS process, offers a high-performance programmable solution capable of 150 MHz on-chip performance and 9 nanosecond clock-to-output speeds. The ACT 3 family spans capacities from 1,500 to 10,000 gate array equivalent gates (up to 25,000 PLD gates), and offers very high pin-to-gate ratios, with up to 228 user I/Os for 10,000 gate designs.

Predictable Performance* (Worst-Case Commercial)	
Accumulators (16-bit)	46–47 MHz
Loadable Counters (16-bit)	76–82 MHz
Prescaled Loadable Counters (16-bit)	127–145 MHz
Shift Registers	150–150 MHz

*See page 1-82 for further details.

Product Family Profile

Device	A1415A	A1425A	A1440A	A1460A	A14100A
Capacity					
Gate Array Equivalent Gates	1,500	2,500	4,000	6,000	10,000
PLD Equivalent Gates	3,750	6,250	10,000	15,000	25,000
TTL Equivalent Packages (40 gates)	40	60	100	150	250
20-Pin PAL Equivalent Packages (100 gates)	15	25	40	60	100
Logic Modules	200	310	564	848	1,377
S-Module	104	160	288	432	697
C-Module	96	150	276	416	680
Dedicated Flip-Flops[1]	264	360	568	768	1,153
User I/Os (maximum)	80	100	140	168	228
Packages[2]					
CPGA	100-pin	133-pin	175-pin	207-pin	257-pin
PLCC	84-pin	84-pin	—	—	—
PQFP	100-pin	100-pin 160-pin	160-pin	208-pin	TBD
Performance[3] (maximum, worst-case commercial)					
Chip-to-Chip[4]	83 MHz	83 MHz	77 MHz	75 MHz	72 MHz
Accumulators (16-bit)	47 MHz	47 MHz	47 MHz	47 MHz	47 MHz
Loadable Counter (16-bit)	82 MHz	82 MHz	82 MHz	80 MHz	80 MHz
Prescaled Loadable Counters (16-bit)	145 MHz	145 MHz	145 MHz	115 MHz	115 MHz
Shift Registers	150 MHz	150 MHz	150 MHz	120 MHz	120 MHz
I/O, Clock-to-Output	10 ns	10 ns	11 ns	11.6 ns	12 ns
CMOS Process	0.8 µm	0.8 µm	0.8 µm	0.8 µm	0.8 µm

Notes:

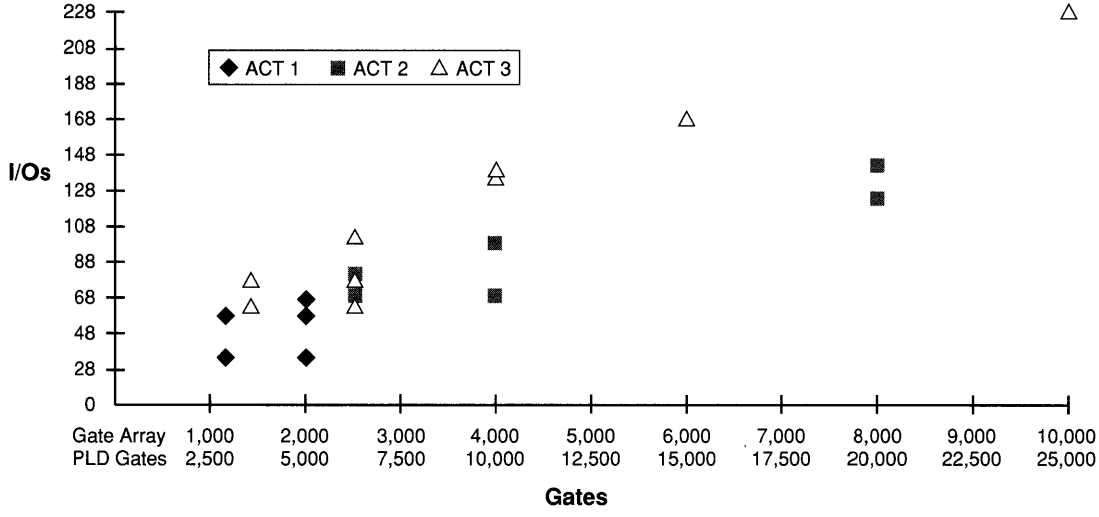
1. One flip-flop per S-Module, two flip-flops per I/O-Module.
2. See product plan on page 1-84 for package availability.
3. Based on A1425A-1 device.
4. Clock-to-Output + Setup

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

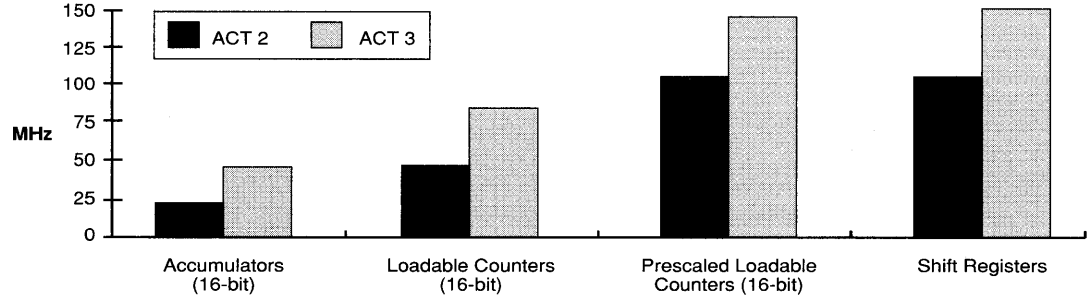
The ACT 3 family is supported by the Designer and Designer Advantage systems, which offers automatic or fixed pin assignment, automatic placement and routing with optional manual placement, timing analysis, user programming, and diagnostic probe capabilities. The system is supported on the following platforms: 386/486™ PC, Sun™ Microsystems, and HP™ workstations. The software provides CAE interfaces to Cadence, Mentor Graphics®, OrCAD™ and Viewlogic® design environments. Additional platforms and CAE interfaces are supported through Actel's Industry Alliance Program, including the CAD/CAM Group, DATA I/O® (ABEL™ FPGA), DAZIX, and MINC.

With the introduction of ACT 3, Actel extends its line of programmable devices. The ACT 1 family offers up to 2,000 gate array equivalent gates (to 6,000 PLD equivalent gates) at industry leading price-to-gate ratios. The ACT 2 family advances this price leadership into higher speed, higher I/O applications requiring 2,500 to 8,000 gate array equivalent gates (to 20,000 PLD equivalent gates). The ACT 3 family offers very high speed with very high I/O-to-gate ratios for designs requiring from less than 1,500 to 10,000 gate array equivalent gates (to 25,000 PLD equivalent gates).

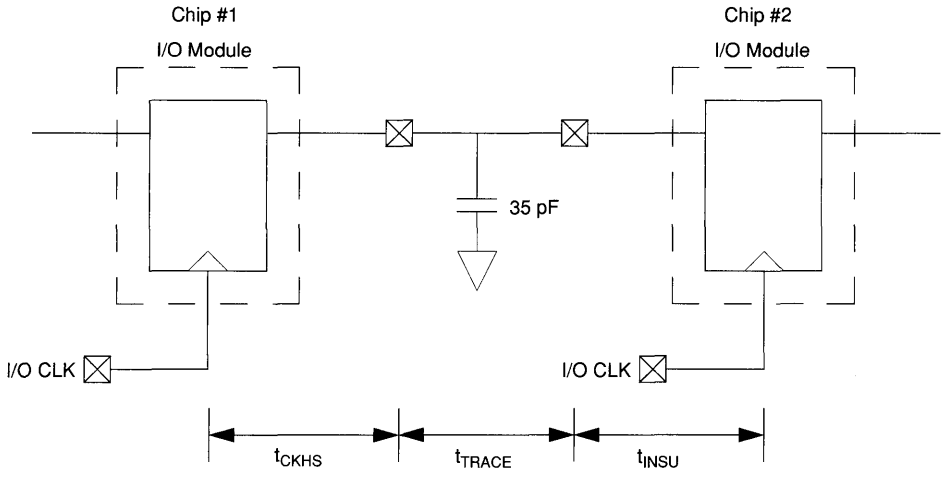
Actel Families: Gates Versus I/Os



Performance: ACT 3 Versus ACT 2 (Standard-Speed Devices)



Chip-to-Chip Performance



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

	t_{CKHS}	t_{TRACE}	t_{INSU}	Total	MHz
A1425A-1	9.0	1.0	3.0	13.0 ns	77
A1460A-1	10.4	1.0	3.0	14.4 ns	69

ACT 3 PREP Performance Examples

The ACT 3 family offers very high system performance. Typical application design building blocks have been developed and implemented to estimate and report ACT 3 system performance. These building blocks have been routed in multiple instances, replicated to fill a device in a step and repeat fashion. The average, minimum, and maximum performances were then determined, giving a realistic estimate of achievable performance. ACT 3 performance is very predictable, as observed by the small spread between maximum and minimum performance. The step and repeat methodology is illustrated in Figure 1.

16-bit Shift Registers

The 16-bit Shift Register Example is a parallel loadable shift register with clear, shift enable, serial in, and serial out. It is replicated by connecting parallel data in to parallel data out, and serial data in to serial data out.

16-bit Prescaled Counters

The 16-bit Prescaled Counter Example is a very high-speed loadable counter optimized for counting. The load requires multiple clock cycles (four), but counting and holding occur at the full clock rate. This counter is ideal for address generation and high-speed timing applications. It is replicated by connecting data inputs to data outputs.

16-bit Non-Prescaled Counters

The Non-Prescaled 16-bit Counter Example is the more traditional 16-bit loadable counter, where loading, counting, and hold all occur at the same clock rate. It is replicated by connecting inputs to counter outputs.

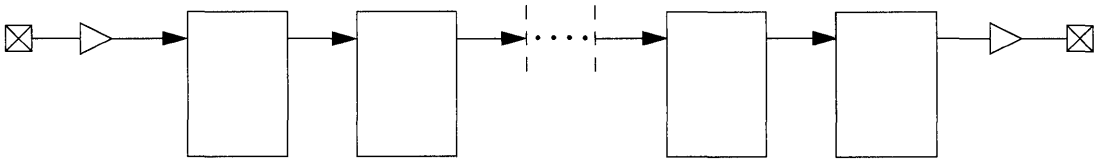
16-bit Accumulators

The 16-bit Accumulator adds a 16-bit number to the previous output value. It is replicated by connecting the data output to the data input.

Performance Results

These designs were completed using Actel's 100% automatic place and route software. No manual placement or routing was used when completing these designs. The performance measurements reflect worst-case commercial conditions.

Table 1 below presents the performance results for each design in minimum, maximum, and average measurements. The table also shows the number of design iterations completed within the device. Notice the tight distribution between minimum and maximum performance, in all cases within 1 ns, and in all cases automatic place and route was used exclusively.



- Step and Repeat Methodology
- Fully Utilized Device
- 100% Automatic Placement and Routing
- No Manual Placement or Routing

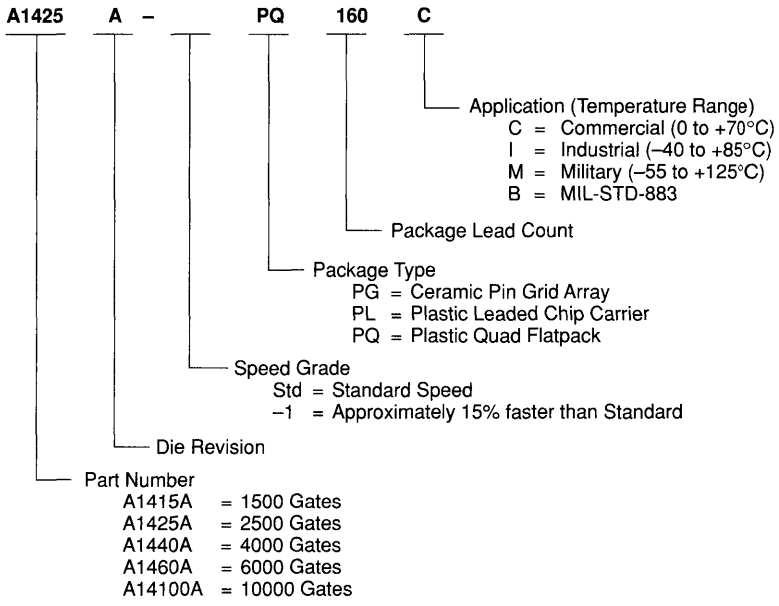
Figure 1. Layout of Performance Examples

Table 1. A1425A-1 Performance Results: Worst-Case Commercial Conditions

Design	Iterations	Performance		
		Minimum	Maximum	Average
16-bit Shift Registers	10	150 MHz	150 MHz	150 MHz
16-bit Prescaled Counters	3	127 MHz	145 MHz	134 MHz
16-bit Non-Prescaled Counters	6	76 MHz	82 MHz	80 MHz
16-bit Accumulators	3	46 MHz	47 MHz	47 MHz

Note:
For more information on the PREP Benchmarks, see Section 2 of this data book.

Ordering Information



Product Plan[1]

	Speed Grade*		Application				
	Std	–1	C	I	M	B	E
A1415A Device							
84-pin Plastic Leaded Chip Carrier (PL)	P	P	P	P	—	—	—
100-pin Plastic Quad Flatpack (PQ)	P	P	P	P	—	—	—
100-pin Ceramic Pin Grid Array (PG)	P	P	P	—	—	—	—
A1425, A1425A Devices							
84-pin Plastic Leaded Chip Carrier (PL)	✔	✔	✔	✔	—	—	—
100-pin Plastic Quad Flatpack (PQ)	✔	✔	✔	✔	—	—	—
133-pin Ceramic Pin Grid Array (PG)	✔	✔	✔	—	P	P	—
160-pin Plastic Quad Flatpack (PQ)	✔	✔	✔	✔	—	—	—
A1440A Device							
160-pin Plastic Quad Flatpack (PQ)	P	P	P	P	—	—	—
175-pin Ceramic Pin Grid Array (PG)	P	P	P	—	—	—	—
A1460A Device							
207-pin Ceramic Pin Grid Array (PG)	✔	P	✔	—	P	P	—
208-pin Plastic Quad Flatpack (PQ)	✔	P	✔	P	—	—	—
A14100A Device							
257-pin Ceramic Pin Grid Array (PG)	P	P	P	—	P	P	—

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

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

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

Note:

1. Availability as of January 1993. Please consult Actel Representatives for current availability.

Device Resources

			User I/Os								
			PLCC	PQFP			CPGA				
Device Series	Logic Modules	Gates	84-pin	100-pin	160-pin	208-pin	100-pin	133-pin	175-pin	207-pin	257-pin
A1415A	200	1500	70	80	—	—	80	—	—	—	—
A1425A	310	2500	70	80	100	—	—	100	—	—	—
A1440A	564	4000	—	—	130	—	—	—	140	—	—
A1460A	848	6000	—	—	—	167	—	—	—	168	—
A14100A	1377	10000	—	—	—	—	—	—	—	—	228

Pin Description

CLKA **Clock A (Input)**

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

CLKB **Clock B (Input)**

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

DCLK **Diagnostic Clock (Input)**

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

GND **Ground**

LOW supply voltage.

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

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

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

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

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

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

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

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

MODE **Mode (Input)**

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

NC **No Connection**

This pin is not connected to circuitry within the device.

PRA **Probe A (Output)**

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

PRB **Probe B (Output)**

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

SDI **Serial Data Input (Input)**

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

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

HIGH supply voltage.

V_{KS} **Programming Voltage**

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

V_{PP} **Programming Voltage**

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

V_{SV} **Programming Voltage**

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

Architecture

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

Topology

The ACT 3 family architecture is composed of six key elements: Logic modules, I/O modules, I/O Pad Drivers, Routing Tracks, Clock Networks, and Programming and Test Circuits. The basic structure is similar for all devices in the family, differing only in the number of rows, columns, and I/Os. The array itself consists of alternating rows of modules and channels. The logic modules and channels are in the center of the array; the I/O modules are located along the array periphery. A simplified floor plan is depicted in Figure 2.

Logic Modules

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

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

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

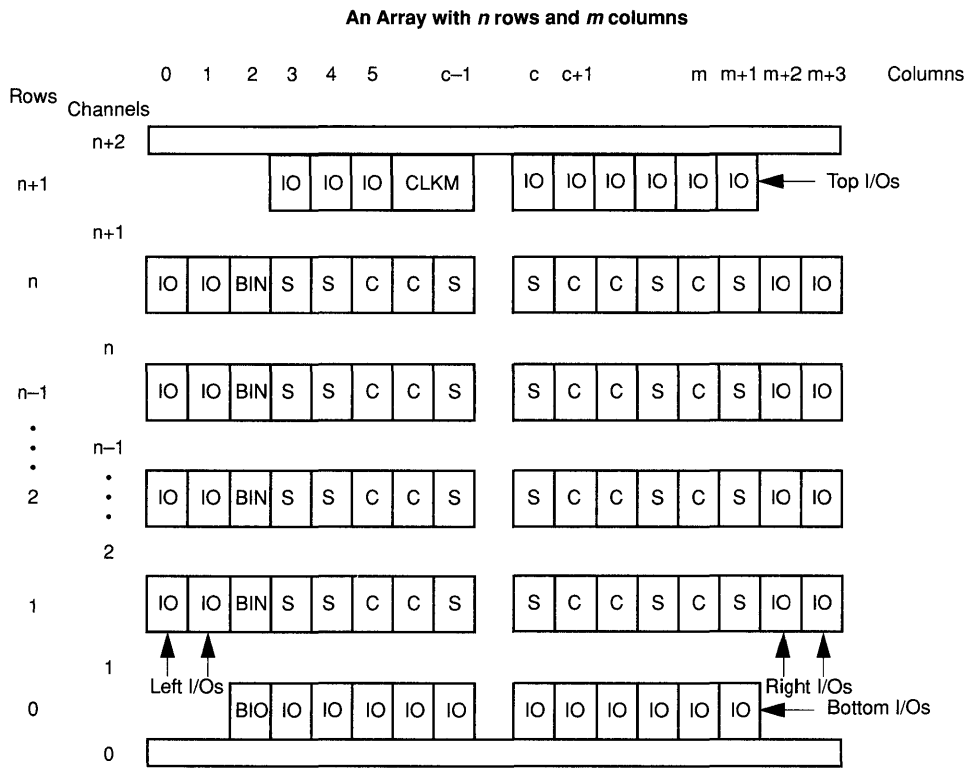


Figure 2. Generalized Floor Plan of ACT 3 Device

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

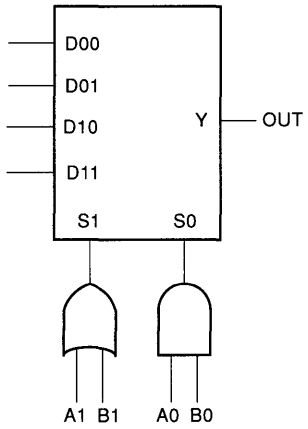


Figure 3. C-Module Diagram

The clear input CLR is accessible from the routing channel. In addition, the clock input may be connected to one of three clock networks: CLK0, CLK1, or HCLK. The C-module and S-module functional descriptions are shown in Figures 3 and 4. The clock selection multiplexor selects the clock input to the S-module.

I/Os

I/O Modules

I/O modules provide an interface between the array and the I/O Pad Drivers. I/O modules are located in the array and access the routing channels in a similar fashion to logic modules. There are two types of I/O modules: side and top/bottom. The I/O module schematic is shown in Figure 5. UO1 and UO2 are inputs from the routing channel, one for the routing channel above and one for the routing channel below the module. The top/bottom I/O modules interact with only one channel and therefore have only one UO input. The signals DataIn and DataOut connect to the I/O pad driver. Each I/O module contains two D-type flip-flops. Each flip-flop is connected to the dedicated I/O clock (IOCLK). Each flip-flop can be bypassed by nonsequential I/Os. In addition, each flip-flop contains a data enable input that can be accessed from the routing channels (ODE and IDE). The asynchronous preset/clear input is driven by the dedicated preset/clear network (IOPCL). Either preset or clear can be selected individually on an I/O module by I/O module basis.

The I/O module output Y is used to bring Pad signals into the array *or* to feed the output register back into the array. This allows the output register to be used in high-speed state machine applications. Side I/O modules have a dedicated output segment for Y extending into the routing channels above and below

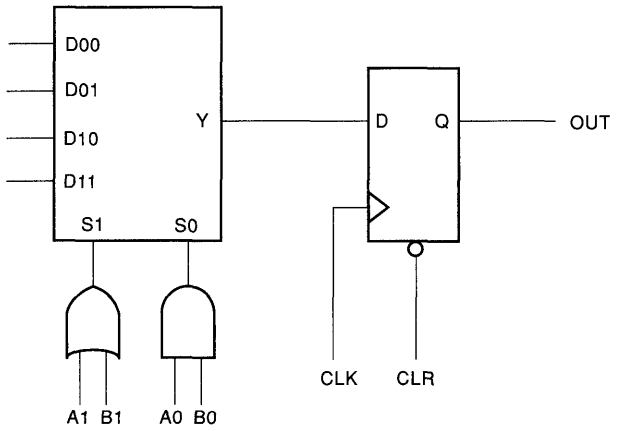


Figure 4. S-Module Diagram

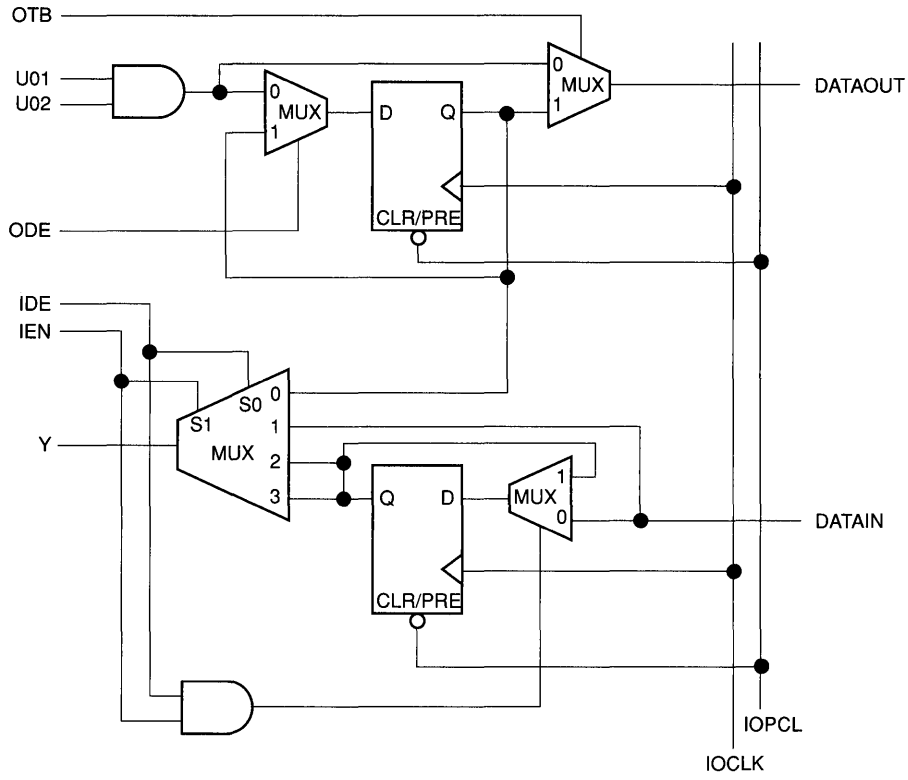


Figure 5. Functional Diagram for I/O Module

(similar to logic modules). Top/Bottom I/O modules have no dedicated output segment. Signals coming into the chip from the top or bottom are routed using F-fuses and LVTs (F-fuses and LVTs are explained in detail in the routing section).

I/O Pad Drivers

All pad drivers are capable of being tristate. Each buffer connects to an associated I/O module with four signals: OE (Output Enable), IE (Input Enable), DataOut, and DataIn. Certain special signals used only during programming and test also connect to the pad drivers: OUTEN (global output enable), INEN (global input enable), and SLEW (individual slew selection). See Figure 6.

Special I/Os

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

Clock Networks

The ACT 3 architecture contains four clock networks: two high-performance dedicated clock networks and two general purpose routed networks. The high-performance networks function up to 150 MHz, while the general purpose routed networks function up to 75 MHz.

Dedicated Clocks

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

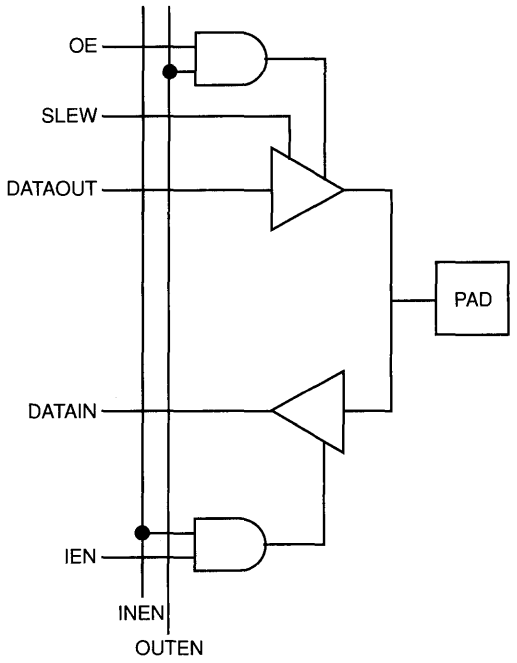


Figure 6. Function Diagram for I/O Pad Driver

Routed Clocks

The routed clock networks are referred to as CLK0 and CLK1. Each network is connected to a clock module (CLKMOD) that selects the source of the clock signal and may be driven as follows (see Figure 7):

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

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

Routing Structure

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

Horizontal Routing

Horizontal channels are located between the rows of modules and are composed of several routing tracks. The horizontal routing tracks within the channel are divided into one or more segments. The minimum horizontal segment length is the width of a module-pair, and the maximum horizontal segment length is the full length of the channel. Any segment that spans more than one-third the row length is considered a long horizontal segment. A typical channel is shown in Figure 8. Undedicated horizontal routing tracks are used to route signal nets. Dedicated routing tracks are used for the global clock networks and for power and ground tie-off tracks.

Vertical Routing

Other tracks run vertically through the modules. Vertical tracks are of three types: input, output, and long. Vertical tracks are also divided into one or more segments. Each segment in an input track is dedicated to the input of a particular module. Each segment in an output track is dedicated to the output of a particular module. Long segments are uncommitted and can be assigned during routing. Each output segment spans four channels (two above and two below), except near the top and bottom of the array where edge effects occur. LVTs contain either one or two segments. An example of vertical routing tracks and segments is shown in Figure 9.

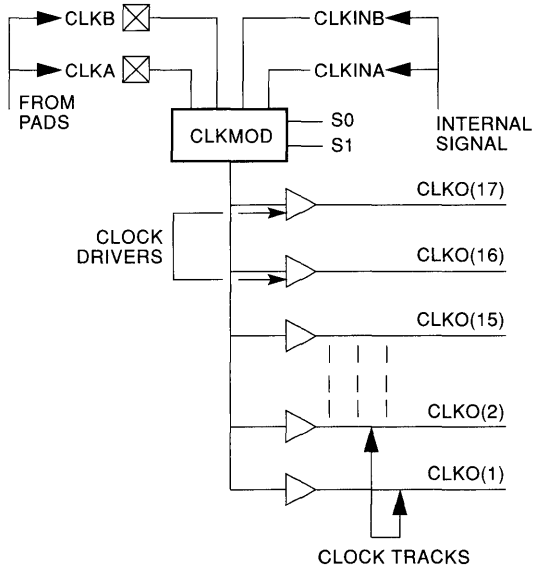


Figure 7. Clock Networks

Antifuse Connections

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

Four types of antifuse connections are used in the routing structure of the ACT 3 array. (The physical structure of the antifuse is identical in each case; only the usage differs.) Table 2 shows four types of antifuses.

Table 2. Antifuse Types

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

Examples of all four types of connections are shown in Figure 8 and Figure 9.

Module Interface

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

Module Input Connections

The tracks dedicated to module inputs are segmented by pass transistors in each module row. During normal user operation, the pass transistors are inactive, which isolates the inputs of a module from the inputs of the module directly above or below it. During certain test modes, the pass transistors are active to verify the continuity of the metal tracks. Vertical input segments span only the channel above or the channel below. The logic modules are arranged such that half of the inputs are connected to the channel above and half of the inputs to segments in the channel below as shown in Figure 10.

Module Output Connections

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

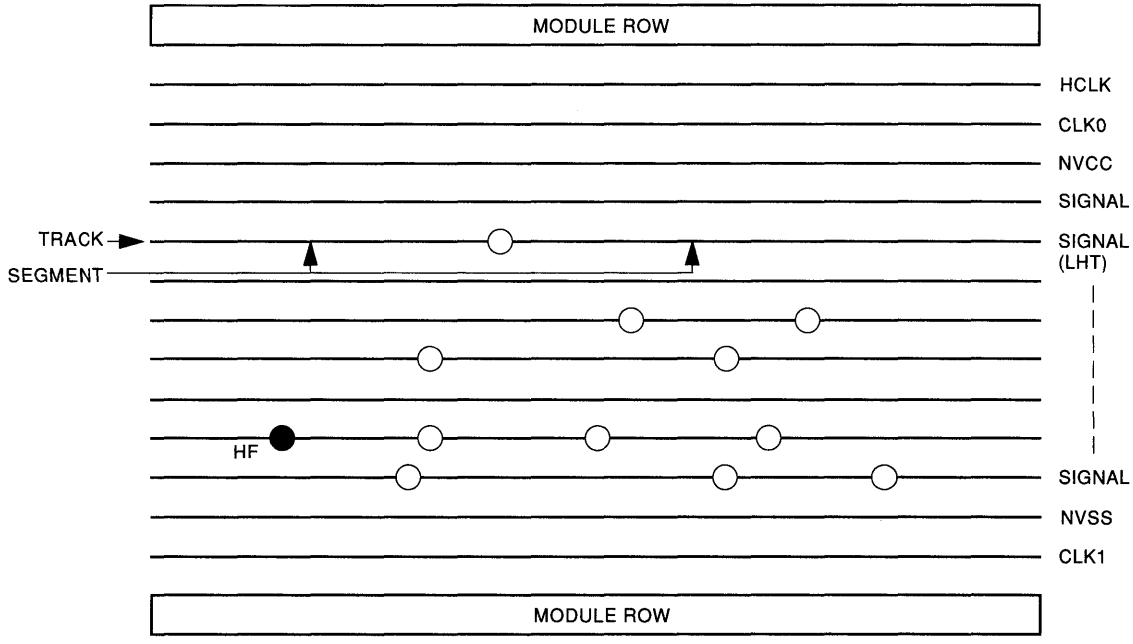


Figure 8. Horizontal Routing Tracks and Segments

LVT Connections

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

Antifuse Connections

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

Clock Connections

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

Programming and Test Circuits

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

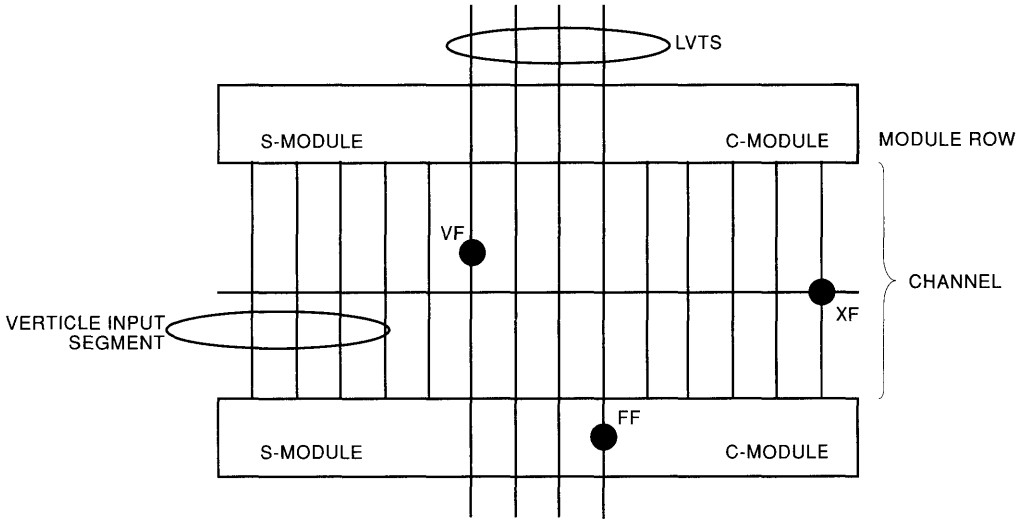


Figure 9. Vertical Routing Tracks and Segments

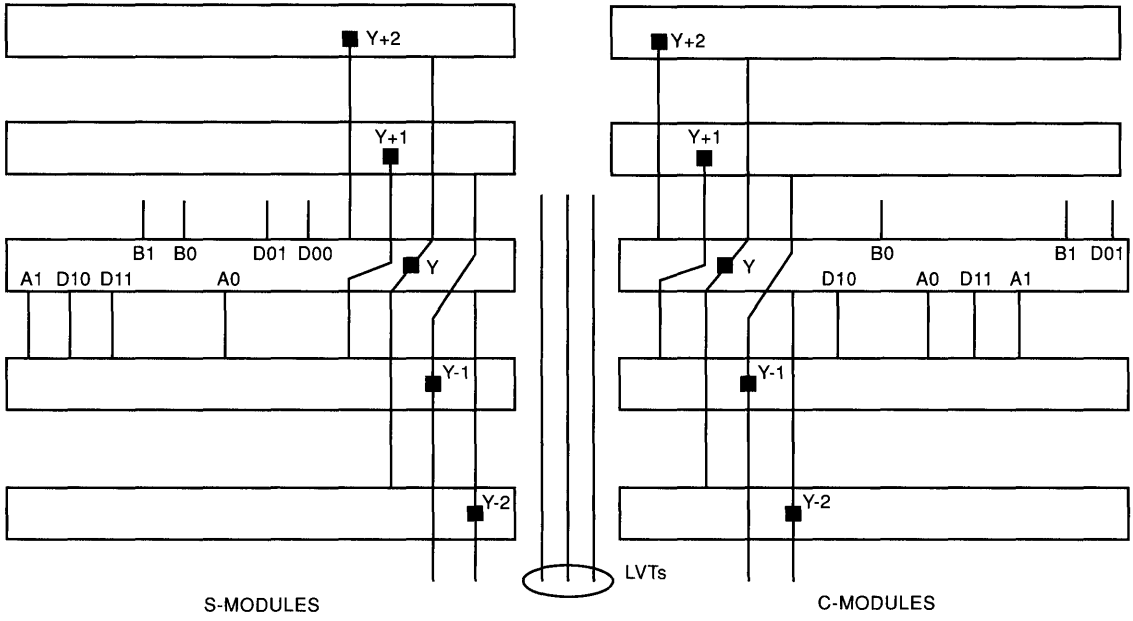


Figure 10. Logic Module Routing Interface

Absolute Maximum Ratings[1]

Free air temperature range

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

Notes:

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

Recommended Operating Conditions

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

Note:

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

Electrical Specifications

Symbol	Parameter	Test Condition	Commercial		Units
			Minimum	Maximum	
V_{OH}[1,2]	HIGH Level Output	I_{OH} = –4 mA (CMOS)	3.84		V
		I_{OH} = –8 mA (TTL)	2.40		V
V_{OL}[1,2]	LOW Level Output	I_{OL} = +6 mA (CMOS)		0.33	V
		I_{OL} = +12 mA (TTL)		0.50	V
V_{IH}	HIGH Level Input	TTL Inputs	2.0	V_{CC} + 0.3	V
V_{IL}	LOW Level Input	TTL Inputs	–0.3	0.8	V
I_{IN}	Input Leakage	$V_I = V_{CC}$ or GND	–10	+10	µA
I_{OZ}	3-state Output Leakage	$V_O = V_{CC}$ or GND	–10	+10	µA
$I_{CC(S)}$	Standby V_{CC} Supply Current	$V_I = V_{CC}$ or GND,			
		I_O = 0 mA		2	mA
$I_{CC(D)}$	Dynamic V_{CC} Supply Current	See "Power Dissipation" Section			

Notes:

1. Actel devices can drive and receive either CMOS or TTL signal levels. No assignment of I/Os as TTL or CMOS is required.
2. Tested one output at a time, V_{CC} = min.

Package Thermal Characteristics

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

Maximum junction temperature is 150°C.

A sample calculation of the absolute maximum power dissipation allowed for a CPGA 177-pin package at commercial temperature and still air is as follows:

$$\text{Absolute Maximum Power Allowed} = \frac{\text{Max. junction temp. (°C)} - \text{Max. ambient temp. (°C)}}{\theta ja\ (°C/W)} = \frac{150°C - 70°C}{18°C/W} = 4.4\ W$$

Package Type	Pin Count	θjc	θja Still Air	θja 300 ft/min	Units
Ceramic Pin Grid Array	100	8	35	17	°C/W
	133	8	30	15	°C/W
	175	8	25	14	°C/W
	207	8	22	13	°C/W
	257	2	15	8	°C/W
Plastic Quad Flatpack[1]	100	13	55	47	°C/W
	160	15	33	26	°C/W
	208	15	33	26	°C/W
Plastic Leaded Chip Carrier[2]	84	15	44	38	°C/W

Notes:
1. Maximum Power Dissipation for 160-pin PQFP package is 1.75 Watts, 208-pin PQFP package is 2.0 Watts, and 100-pin PQFP package is 1.0 Watt.
2. Maximum Power Dissipation for PLCC package is 1.5 Watts.

Power Dissipation

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

Where:

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

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

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

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

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

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

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

Static Power

Static power dissipation is typically a small component of the overall power. From the values provided in the Electrical Specifications, the maximum static power (commercial) dissipation is:

1 mA x 5.25 V = 5.25 mW

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

Active Time

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

Equivalent Capacitance

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

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

Where:

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

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

f is the switching frequency in megahertz (MHz).

Equivalent capacitance is calculated by measuring I_{active} at a specified frequency and voltage for each circuit component of interest. The results for ACT 3 devices are:

	C_{EQ} (pF)
Modules	8.2
Input Buffers	1.5
Output Buffers	2.3
I/O Clock Buffer Loads	0.4
Dedicated Array Clock Buffer Loads	0.5
Routed Array Clock Buffer Loads	0.5 + fixed/device

To calculate the active power dissipated from the complete design, you must solve Equation 1 for each component. To do this, you must know the switching frequency of each part of the logic. The exact equation is a piece-wise linear summation over all components, as shown in Equation 2.

$$\text{Power } (\mu W) = [(m \times 8.2 \times f_1) + (n \times 1.5 \times f_2) + (p \times (2.3 + C_L) \times f_3) + (q \times 0.5 \times f_4) + ((r_1 + 0.5\, r_2) \times f_5) + (s \times 0.4 \times f_6)] \times V_{CC}^2 \quad (2)$$

Where:

m = Number of logic modules switching at f_1

n = Number of input buffers switching at f_2

p = Number of output buffers switching at f_3

q = Number of clock loads on the dedicated array clock network

A1415A: q = 104

A1425A: q = 160

A1440A: q = 288

A1460A: q = 432

A14100A: q = 697

r_1 = Fixed capacitance due to routed array clock network

A1415A: r_1 = 60

A1425A: r_1 = 75

A1440A: r_1 = 105

A1460A: r_1 = 145

A14100A: r_1 = 195

r_2 = Number of clock loads on the routed array clock network

s = Number of clock loads on the dedicated I/O clock network

A1415A: s = 80

A1425A: s = 100

A1440A: s = 140

A1460A: s = 168

A14100A: s = 228

f_1 = Average logic module switching rate in MHz

f_2 = Average input buffer switching rate in MHz

f_3 = Average output buffer switching rate in MHz

f_4 = Average dedicated array clock rate in MHz

f_5 = Average routed array clock rate in MHz

f_6 = Average dedicated I/O clock rate in MHz

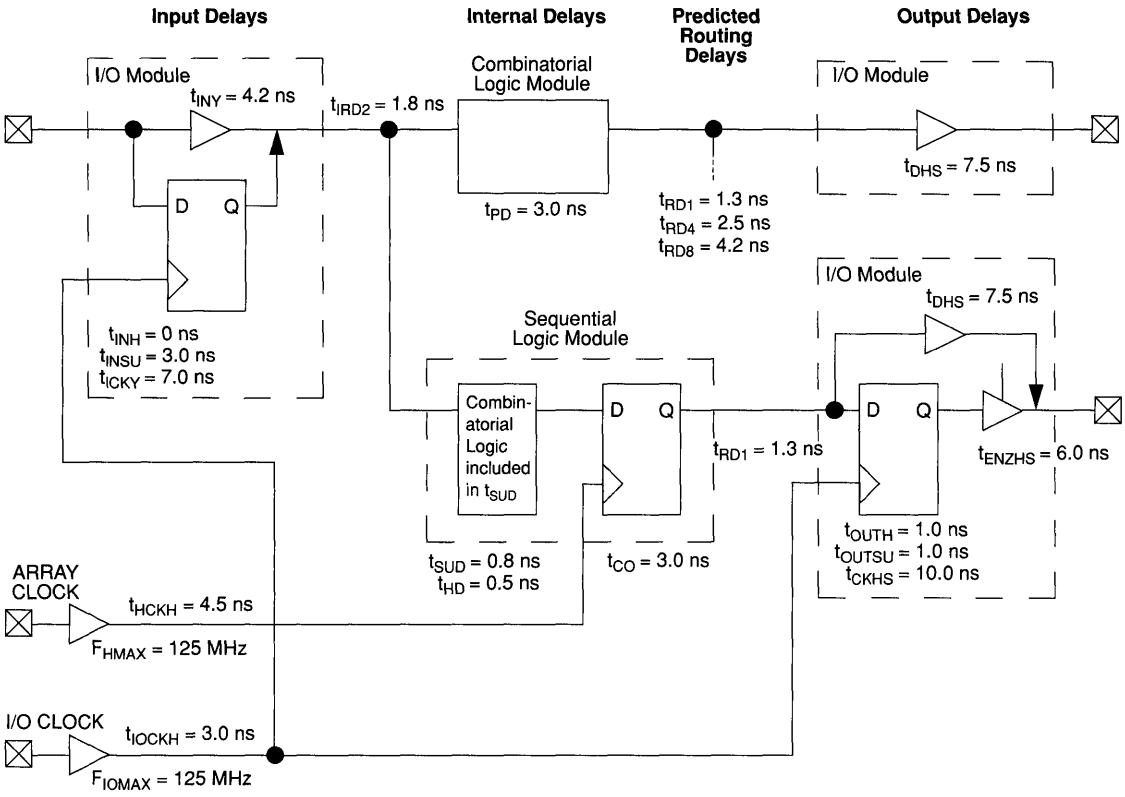
C_L = Output load capacitance in pF

Determining Average Switching Frequency

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

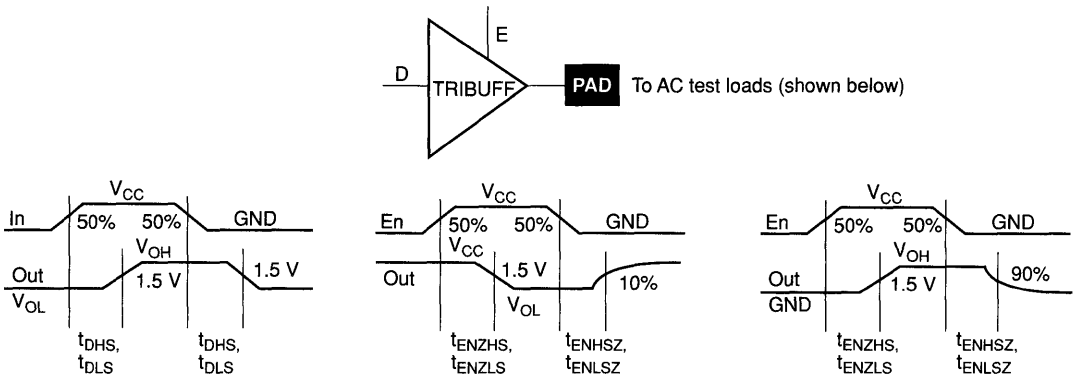
Logic Modules (m)	= 80% of modules
Average module switching rate (f_1)	= F/10
Inputs switching (n)	= # I/Os used/12
Average input switching rate (f_2)	= F
Outputs switching (p)	= # I/Os used/15
Output loading (C_L)	= 35
Average output switching rate (f_3)	= F/2
Dedicated array clock loads (q)	= fixed by device
Average dedicated array switching rate (f_4)	= F
Routed array fixed capacitance (r_1)	= fixed by device
Routed array clock loads (r_2)	= 40% of sequential modules
Average routed array switching rate (f_5)	= F/2
I/O clock loads (s)	= # I/Os used
Average I/O switching rate (f_6)	= F

ACT 3 Timing Model*



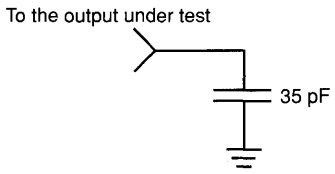
*Values shown for A1425A.

Output Buffer Delays

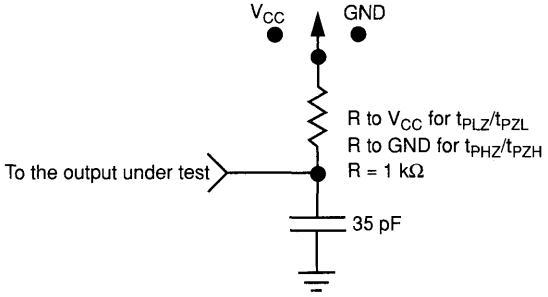


AC Test Loads

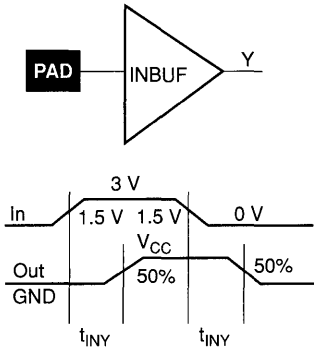
Load 1
(Used to measure propagation delay)



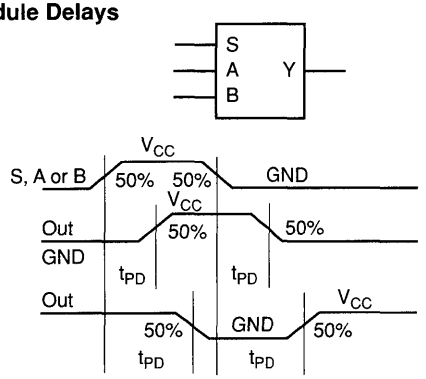
Load 2
(Used to measure rising/falling edges)



Input Buffer Delays

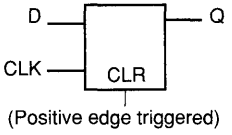


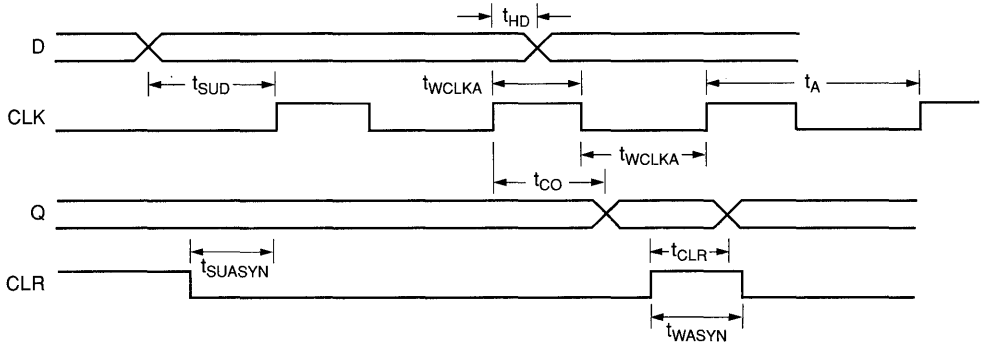
Module Delays



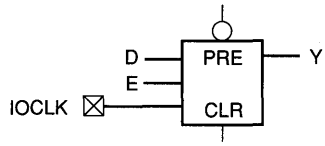
Sequential Module Timing Characteristics

Flip-Flops

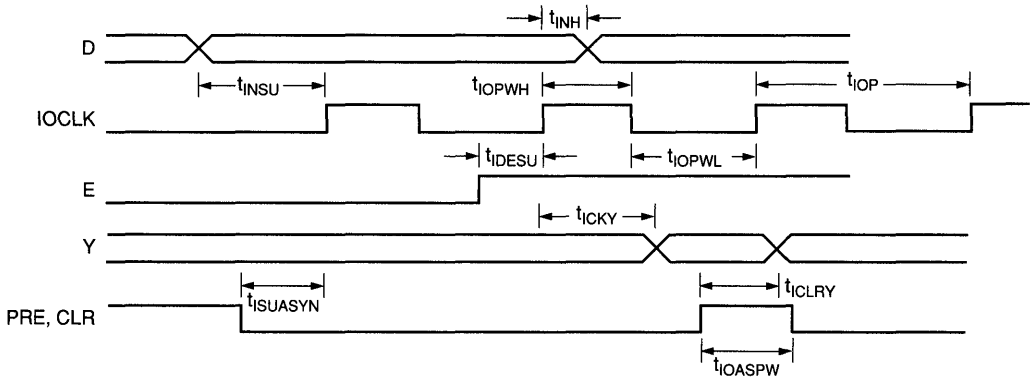




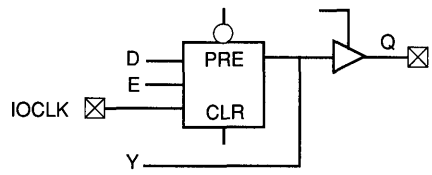
I/O Module: Sequential Input Timing Characteristics



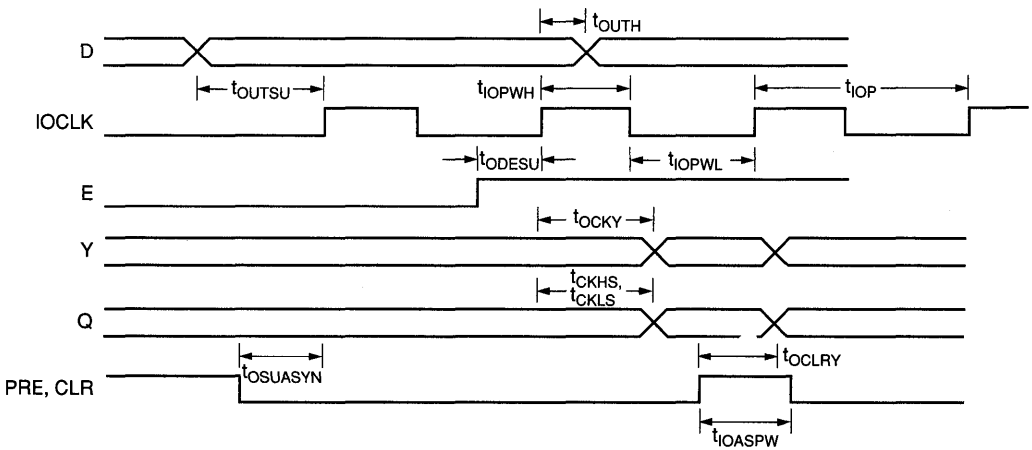
(Positive edge triggered)



I/O Module: Sequential Output Timing Characteristics



(Positive edge triggered)



Predictable Performance: Tightest Delay Distributions

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

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

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

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

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

Table 3. Logic Module + Routing Delay, by fanout (ns)[1] (Worst-Case Commercial Conditions)

Family	FO=1	FO=2	FO=3	FO=4	FO=8
ACT 1	5.0	5.7	6.6	7.9	12.5
ACT 2	5.5	6.2	6.9	7.4	9.2
ACT 3	3.7	4.2	4.4	4.8	6.2

Note:
1. '–1' Speed Devices Specified

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

Timing Characteristics

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

Critical Nets and Typical Nets

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

Long Tracks

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

Timing Derating

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

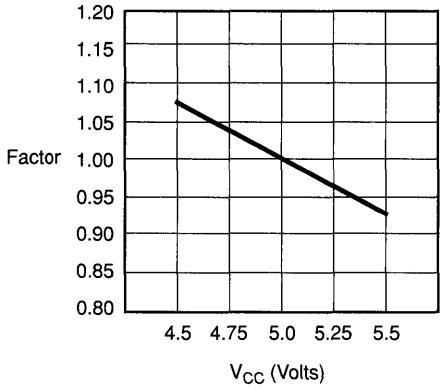
Timing Derating Factor, Temperature and Voltage

	Industrial		Military	
	Minimum	**Maximum**	**Minimum**	**Maximum**
(Commercial Minimum/Maximum Specification) x	0.85	1.07	0.81	1.16

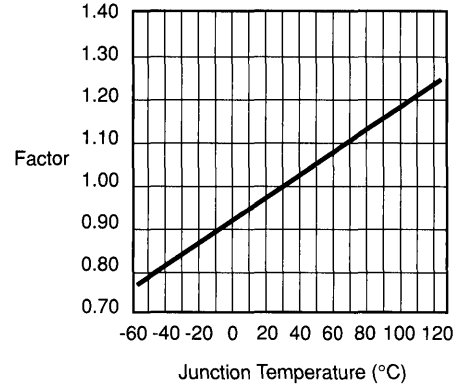
Timing Derating Factor for Designs at Typical Temperature (T_J = 25°C) and Voltage (5.0 V)

(Commercial Maximum Specification) x	0.87

Voltage Derating Curve



Temperature Derating Curve



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

A1415A Timing Characteristics

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

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

Notes:

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

**Actel is offering "Advanced Information" only on '–1' Speed devices. Consult Actel for '–1' device availability.*

A1415A Timing Characteristics (continued)

(Worst-Case Commercial Conditions)

		Preliminary Information		Advanced Information*		
I/O Module Input Propagation Delays		**'Std' Speed**		**'–1' Speed**		
Parameter	**Description**	**Min.**	**Max.**	**Min.**	**Max.**	**Units**
t_{INY}	Input Data Pad to Y		4.2		3.6	ns
t_{ICKY}	Input Reg IOCLK Pad to Y		7.0		6.0	ns
t_{OCKY}	Output Reg IOCLK Pad to Y		7.0		6.0	ns
t_{ICLRY}	Input Asynchronous Clear to Y		7.0		6.0	ns
t_{OCLRY}	Output Asynchronous Clear to Y		7.0		6.0	ns
Predicted Input Routing Delays[1]						
t_{IRD1}	FO=1 Routing Delay		1.3		1.1	ns
t_{IRD2}	FO=2 Routing Delay		1.8		1.6	ns
t_{IRD3}	FO=3 Routing Delay		2.1		1.8	ns
t_{IRD4}	FO=4 Routing Delay		2.5		2.2	ns
t_{IRD8}	FO=8 Routing Delay		4.2		3.6	ns
I/O Module Sequential Timing						
t_{INH}	Input F-F Data Hold (w.r.t. IOCLK Pad)	0.0		0.0		ns
t_{INSU}	Input F-F Data Setup (w.r.t. IOCLK Pad)	3.0		3.0		ns
t_{IDEH}	Input Data Enable Hold (w.r.t. IOCLK Pad)	0.0		0.0		ns
t_{IDESU}	Input Data Enable Setup (w.r.t. IOCLK Pad)	9.0		9.0		ns
$t_{ISUASYN}$	Input Asynchronous Setup	TBD		TBD		ns
t_{OUTH}	Output F-F Data Hold (w.r.t. IOCLK Pad)	1.0		1.0		ns
t_{OUTSU}	Output F-F Data Setup (w.r.t. IOCLK Pad)	1.0		1.0		ns
t_{ODEH}	Output Data Enable Hold (w.r.t. IOCLK Pad)	0.5		0.5		ns
t_{ODESU}	Output Data Enable Setup (w.r.t. IOCLK Pad)	2.0		2.0		ns
$t_{OSUASYN}$	Output Asynchronous Setup	TBD		TBD		ns

Note:

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

**Actel is offering "Advanced Information" only on '–1' Speed devices. Consult Actel for '–1' device availability.*

A1415A Timing Characteristics (continued)

(Worst-Case Commercial Conditions)

		Preliminary Information		Advanced Information*		
I/O Module – TTL Output Timing[1]		**'Std' Speed**		**'–1' Speed**		
Parameter	**Description**	**Min.**	**Max.**	**Min.**	**Max.**	**Units**
t_{DHS}	Data to Pad, High Slew		7.5		6.4	ns
t_{DLS}	Data to Pad, Low Slew		12.0		10.2	ns
t_{ENZHS}	Enable to Pad, Z to H/L, Hi Slew		6.0		5.1	ns
t_{ENZLS}	Enable to Pad, Z to H/L, Lo Slew		11.0		9.4	ns
t_{ENHSZ}	Enable to Pad, H/L to Z, Hi Slew		10.0		8.5	ns
t_{ENLSZ}	Enable to Pad, H/L to Z, Lo Slew		10.0		8.5	ns
t_{CKHS}	IOCLK Pad to Pad H/L, Hi Slew		10.0		9.0	ns
t_{CKLS}	IOCLK Pad to Pad H/L, Lo Slew		15.0		13.5	ns
d_{TLHHS}	Delta Low to High, Hi Slew		TBD		TBD	ns/pF
d_{TLHLS}	Delta Low to High, Lo Slew		TBD		TBD	ns/pF
d_{THLHS}	Delta High to Low, Hi Slew		TBD		TBD	ns/pF
d_{THLLS}	Delta High to Low, Lo Slew		TBD		TBD	ns/pF
I/O Module – CMOS Output Timing[1]						
t_{DHS}	Data to Pad, High Slew		9.3		7.9	ns
t_{DLS}	Data to Pad, Low Slew		17.5		14.9	ns
t_{ENZHS}	Enable to Pad, Z to H/L, Hi Slew		7.8		6.6	ns
t_{ENZLS}	Enable to Pad, Z to H/L, Lo Slew		13.3		11.3	ns
t_{ENHSZ}	Enable to Pad, H/L to Z, Hi Slew		10.0		8.5	ns
t_{ENLSZ}	Enable to Pad, H/L to Z, Lo Slew		10.0		9.0	ns
t_{CKHS}	IOCLK Pad to Pad H/L, Hi Slew		11.8		10.7	ns
t_{CKLS}	IOCLK Pad to Pad H/L, Lo Slew		17.3		15.6	ns
d_{TLHHS}	Delta Low to High, Hi Slew		TBD		TBD	ns/pF
d_{TLHLS}	Delta Low to High, Lo Slew		TBD		TBD	ns/pF
d_{THLHS}	Delta High to Low, Hi Slew		TBD		TBD	ns/pF
d_{THLLS}	Delta High to Low, Lo Slew		TBD		TBD	ns/pF

Note:
1. Delays based on 35pF loading.

**Actel is offering "Advanced Information" only on '–1' Speed devices. Consult Actel for '–1' device availability.*

A1415A Timing Characteristics (continued)

(Worst-Case Commercial Conditions)

		Preliminary Information		Advanced Information*		
Dedicated (Hard-Wired) I/O Clock Network		**'Std' Speed**		**'–1' Speed**		
Parameter	**Description**	**Min.**	**Max.**	**Min.**	**Max.**	**Units**
t_{IOCKH}	Input Low to High (Pad to I/O Module Input)		3.0		2.6	ns
t_{IOPWH}	Minimum Pulse Width High	3.8		3.3		ns
t_{IOPWL}	Minimum Pulse Width Low	3.8		3.3		ns
t_{IOSAPW}	Minimum Asynchronous Pulse Width	3.8		3.3		ns
t_{IOCKSW}	Maximum Skew		0.4		0.4	ns
t_{IOP}	Minimum Period	8.0		6.8		ns
f_{IOMAX}	Maximum Frequency		125		150	MHz
Dedicated (Hard-Wired) Array Clock Network						
t_{HCKH}	Input Low to High (Pad to S-Module Input)		4.5		3.9	ns
t_{HCKL}	Input High to Low (Pad to S-Module Input)		4.5		3.9	ns
t_{HPWH}	Minimum Pulse Width High	3.8		3.3		ns
t_{HPWL}	Minimum Pulse Width Low	3.8		3.3		ns
t_{HCKSW}	Maximum Skew		0.3		0.3	ns
t_{HP}	Minimum Period	8.0		6.8		ns
f_{HMAX}	Maximum Frequency		125		150	MHz
Routed Array Clock Networks						
t_{RCKH}	Input Low to High (FO=64)		5.5		4.7	ns
t_{RCKL}	Input High to Low (FO=64)		6.0		5.1	ns
t_{RPWH}	Min. Pulse Width High (FO=64)	4.9		4.2		ns
t_{RPWL}	Min. Pulse Width Low (FO=64)	4.9		4.2		ns
t_{RCKSW}	Maximum Skew (FO=128)		1.0		0.9	ns
t_{RP}	Minimum Period (FO=64)	10.0		8.7		ns
f_{RMAX}	Maximum Frequency (FO=64)		100		115	MHz
Clock-to-Clock Skews						
$t_{IOHCKSW}$	I/O Clock to H-Clock Skew	0.0	1.3	0.0	3.0	ns
$t_{IOHCKSW}$	I/O Clock to R-Clock Skew	0.0	3.0	0.0	3.0	ns
t_{HRCKSW}	H-Clock to R-Clock Skew	0.0	1.0	0.0	1.0	ns

Note:

1. Delays based on 35pF loading.

**Actel is offering "Advanced Information" only on '–1' Speed devices. Consult Actel for '–1' device availability.*

A1425A Timing Characteristics

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

		Preliminary Information		Preliminary Information		Advanced Information*		
Logic Module Propagation Delays[1]		**'Std' Speed**		**'–1' Speed**		**'–2' Speed**		
Parameter	**Description**	**Min.**	**Max.**	**Min.**	**Max.**	**Min.**	**Max.**	**Units**
t_{PD}	Internal Array Module		3.0		2.6		2.3	ns
t_{CO}	Sequential Clock to Q		3.0		2.6		2.3	ns
t_{CLR}	Asynchronous Clear to Q		3.0		2.6		2.3	ns
Predicted Routing Delays[2]								
t_{RD1}	FO=1 Routing Delay		1.3		1.1		1.0	ns
t_{RD2}	FO=2 Routing Delay		1.8		1.6		1.4	ns
t_{RD3}	FO=3 Routing Delay		2.1		1.8		1.6	ns
t_{RD4}	FO=4 Routing Delay		2.5		2.2		1.9	ns
t_{RD8}	FO=8 Routing Delay		4.2		3.6		3.2	ns
Logic Module Sequential Timing								
t_{SUD}	Flip-Flop Data Input Setup	0.8		0.8		0.8		ns
t_{HD}	Flip-Flop Data Input Hold	0.5		0.5		0.5		ns
t_{SUD}	Latch Data Input Setup	0.8		0.8		0.8		ns
t_{HD}	Latch Data Input Hold	0.5		0.5		0.5		ns
t_{SUASYN}	Asynchronous Input Setup	TBD		TBD		TBD		ns
t_{WASYN}	Asynchronous Pulse Width	3.8		3.2		2.9		ns
t_{WCLKA}	Flip-Flop Clock Pulse Width	3.8		3.2		2.9		ns
t_A	Flip-Flop Clock Input Period	8.0		6.8		6.0		ns
f_{MAX}	Flip-Flop Clock Frequency		125		150		167	MHz

Notes:

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

**Actel is offering "Advanced Information" only on '–2' Speed devices. Consult Actel for '–2' device availability.*

A1425A Timing Characteristics (continued)

(Worst-Case Commercial Conditions)

		Preliminary Information		Preliminary Information		Advanced Information*		
I/O Module Input Propagation Delays		**'Std' Speed**		**'–1' Speed**		**'–2' Speed**		
Parameter	**Description**	**Min.**	**Max.**	**Min.**	**Max.**	**Min.**	**Max.**	**Units**
t_{INY}	Input Data Pad to Y		4.2		3.6		3.2	ns
t_{ICKY}	Input Reg IOCLK Pad to Y		7.0		6.0		5.3	ns
t_{OCKY}	Output Reg IOCLK Pad to Y		7.0		6.0		5.3	ns
t_{ICLRY}	Input Asynchronous Clear to Y		7.0		6.0		5.3	ns
t_{OCLRY}	Output Asynchronous Clear to Y		7.0		6.0		5.3	ns
Predicted Input Routing Delays[1]								
t_{IRD1}	FO=1 Routing Delay		1.3		1.1		1.0	ns
t_{IRD2}	FO=2 Routing Delay		1.8		1.6		1.4	ns
t_{IRD3}	FO=3 Routing Delay		2.1		1.8		1.6	ns
t_{IRD4}	FO=4 Routing Delay		2.5		2.2		1.9	ns
t_{IRD8}	FO=8 Routing Delay		4.2		3.6		3.2	ns
I/O Module Sequential Timing								
t_{INH}	Input F-F Data Hold (w.r.t. IOCLK Pad)	0.0		0.0		0.0		ns
t_{INSU}	Input F-F Data Setup (w.r.t. IOCLK Pad)	3.0		3.0		3.0		ns
t_{IDEH}	Input Data Enable Hold (w.r.t. IOCLK Pad)	0.0		0.0		0.0		ns
t_{IDESU}	Input Data Enable Setup (w.r.t. IOCLK Pad)	9.0		9.0		9.0		ns
$t_{ISUASYN}$	Input Asynchronous Setup	TBD		TBD		TBD		ns
t_{OUTH}	Output F-F Data Hold (w.r.t. IOCLK Pad)	1.0		1.0		1.0		ns
t_{OUTSU}	Output F-F Data Setup (w.r.t. IOCLK Pad)	1.0		1.0		1.0		ns
t_{ODEH}	Output Data Enable Hold (w.r.t. IOCLK Pad)	0.5		0.5		0.5		ns
t_{ODESU}	Output Data Enable Setup (w.r.t. IOCLK Pad)	2.0		2.0		2.0		ns
$t_{OSUASYN}$	Output Asynchronous Setup	TBD		TBD		TBD		ns

Note:

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

**Actel is offering "Advanced Information" only on '–2' Speed devices. Consult Actel for '–2' device availability.*

A1425A Timing Characteristics (continued)

(Worst-Case Commercial Conditions)

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

Note:
1. Delays based on 35pF loading.

**Actel is offering "Advanced Information" only on '–2' Speed devices. Consult Actel for '–2' device availability.*

A1425A Timing Characteristics (continued)

(Worst-Case Commercial Conditions)

		Preliminary Information		Preliminary Information		Advanced Information*		
Dedicated (Hard-Wired) I/O Clock Network		**'Std' Speed**		**'–1' Speed**		**'–2' Speed**		
Parameter	**Description**	**Min.**	**Max.**	**Min.**	**Max.**	**Min.**	**Max.**	**Units**
t_{IOCKH}	Input Low to High (Pad to I/O Module Input)		3.0		2.6		2.3	ns
t_{IOPWH}	Minimum Pulse Width High	3.8		3.3		2.9		ns
t_{IOPWL}	Minimum Pulse Width Low	3.8		3.3		2.9		ns
t_{IOSAPW}	Minimum Asynchronous Pulse Width	3.8		3.3		2.9		ns
t_{IOCKSW}	Maximum Skew		0.4		0.4		0.4	ns
t_{IOP}	Minimum Period	8.0		6.8		6.0		ns
f_{IOMAX}	Maximum Frequency		125		150		167	MHz
Dedicated (Hard-Wired) Array Clock Network								
t_{HCKH}	Input Low to High (Pad to S-Module Input)		4.5		3.9		3.4	ns
t_{HCKL}	Input High to Low (Pad to S-Module Input)		4.5		3.9		3.4	ns
t_{HPWH}	Minimum Pulse Width High	3.8		3.3		2.9		ns
t_{HPWL}	Minimum Pulse Width Low	3.8		3.3		2.9		ns
t_{HCKSW}	Maximum Skew		0.3		0.3		0.3	ns
t_{HP}	Minimum Period	8.0		6.8		6.0		ns
f_{HMAX}	Maximum Frequency		125		150		167	MHz
Routed Array Clock Networks								
t_{RCKH}	Input Low to High (FO=64)		5.5		4.7		4.1	ns
t_{RCKL}	Input High to Low (FO=64)		6.0		5.1		4.5	ns
t_{RPWH}	Min. Pulse Width High (FO=64)	4.9		4.2		3.8		ns
t_{RPWL}	Min. Pulse Width Low (FO=64)	4.9		4.2		3.8		ns
t_{RCKSW}	Maximum Skew (FO=128)		1.0		0.9		0.8	ns
t_{RP}	Minimum Period (FO=64)	10.0		8.7		8.0		ns
f_{RMAX}	Maximum Frequency (FO=64)		100		115		125	MHz
Clock-to-Clock Skews								
$t_{IOHCKSW}$	I/O Clock to H-Clock Skew	0.0	1.3	0.0	3.0	0.0	3.0	ns
$t_{IOHCKSW}$	I/O Clock to R-Clock Skew	0.0	3.0	0.0	3.0	0.0	3.0	ns
t_{HRCKSW}	H-Clock to R-Clock Skew	0.0	1.0	0.0	1.0	0.0	1.0	ns

Note:
1. Delays based on 35pF loading.

**Actel is offering "Advanced Information" only on '–2' Speed devices. Consult Actel for '–2' device availability.*

A1440A Timing Characteristics

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

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

Notes:

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

**Actel is offering "Advanced Information" only on '–1' Speed devices. Consult Actel for '–1' device availability.*

A1440A Timing Characteristics (continued)

(Worst-Case Commercial Conditions)

		Preliminary Information		Advanced Information*		
I/O Module Input Propagation Delays		**'Std' Speed**		**'–1' Speed**		
Parameter	**Description**	**Min.**	**Max.**	**Min.**	**Max.**	**Units**
t_{INY}	Input Data Pad to Y		4.2		3.6	ns
t_{ICKY}	Input Reg IOCLK Pad to Y		7.0		6.0	ns
t_{OCKY}	Output Reg IOCLK Pad to Y		7.0		6.0	ns
t_{ICLRY}	Input Asynchronous Clear to Y		7.0		6.0	ns
t_{OCLRY}	Output Asynchronous Clear to Y		7.0		6.0	ns
Predicted Input Routing Delays[1]						
t_{IRD1}	FO=1 Routing Delay		1.3		1.1	ns
t_{IRD2}	FO=2 Routing Delay		1.8		1.6	ns
t_{IRD3}	FO=3 Routing Delay		2.1		1.8	ns
t_{IRD4}	FO=4 Routing Delay		2.5		2.2	ns
t_{IRD8}	FO=8 Routing Delay		4.2		3.6	ns
I/O Module Sequential Timing						
t_{INH}	Input F-F Data Hold (w.r.t. IOCLK Pad)	0.0		0.0		ns
t_{INSU}	Input F-F Data Setup (w.r.t. IOCLK Pad)	3.0		3.0		ns
t_{IDEH}	Input Data Enable Hold (w.r.t. IOCLK Pad)	0.0		0.0		ns
t_{IDESU}	Input Data Enable Setup (w.r.t. IOCLK Pad)	9.0		9.0		ns
$t_{ISUASYN}$	Input Asynchronous Setup	TBD		TBD		ns
t_{OUTH}	Output F-F Data Hold (w.r.t. IOCLK Pad)	1.0		1.0		ns
t_{OUTSU}	Output F-F Data Setup (w.r.t. IOCLK Pad)	1.0		1.0		ns
t_{ODEH}	Output Data Enable Hold (w.r.t. IOCLK Pad)	0.5		0.5		ns
t_{ODESU}	Output Data Enable Setup (w.r.t. IOCLK Pad)	2.0		2.0		ns
$t_{OSUASYN}$	Output Asynchronous Setup	TBD		TBD		ns

Note:

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

**Actel is offering "Advanced Information" only on '–1' Speed devices. Consult Actel for '–1' device availability.*

A1440A Timing Characteristics (continued)

(Worst-Case Commercial Conditions)

		Preliminary Information		Advanced Information*		
I/O Module – TTL Output Timing[1]		**'Std' Speed**		**'–1' Speed**		
Parameter	**Description**	**Min.**	**Max.**	**Min.**	**Max.**	**Units**
t_{DHS}	Data to Pad, High Slew		7.5		6.4	ns
t_{DLS}	Data to Pad, Low Slew		12.0		10.2	ns
t_{ENZHS}	Enable to Pad, Z to H/L, Hi Slew		6.0		5.1	ns
t_{ENZLS}	Enable to Pad, Z to H/L, Lo Slew		11.0		9.4	ns
t_{ENHSZ}	Enable to Pad, H/L to Z, Hi Slew		11.0		9.4	ns
t_{ENLSZ}	Enable to Pad, H/L to Z, Lo Slew		11.0		9.4	ns
t_{CKHS}	IOCLK Pad to Pad H/L, Hi Slew		11.0		10.0	ns
t_{CKLS}	IOCLK Pad to Pad H/L, Lo Slew		15.0		13.5	ns
d_{TLHHS}	Delta Low to High, Hi Slew		TBD		TBD	ns/pF
d_{TLHLS}	Delta Low to High, Lo Slew		TBD		TBD	ns/pF
d_{THLHS}	Delta High to Low, Hi Slew		TBD		TBD	ns/pF
d_{THLLS}	Delta High to Low, Lo Slew		TBD		TBD	ns/pF
I/O Module – CMOS Output Timing[1]						
t_{DHS}	Data to Pad, High Slew		9.3		7.9	ns
t_{DLS}	Data to Pad, Low Slew		17.5		14.9	ns
t_{ENZHS}	Enable to Pad, Z to H/L, Hi Slew		7.8		6.6	ns
t_{ENZLS}	Enable to Pad, Z to H/L, Lo Slew		13.3		8.5	ns
t_{ENHSZ}	Enable to Pad, H/L to Z, Hi Slew		11.0		9.4	ns
t_{ENLSZ}	Enable to Pad, H/L to Z, Lo Slew		11.0		9.4	ns
t_{CKHS}	IOCLK Pad to Pad H/L, Hi Slew		11.8		10.7	ns
t_{CKLS}	IOCLK Pad to Pad H/L, Lo Slew		17.3		15.6	ns
d_{TLHHS}	Delta Low to High, Hi Slew		TBD		TBD	ns/pF
d_{TLHLS}	Delta Low to High, Lo Slew		TBD		TBD	ns/pF
d_{THLHS}	Delta High to Low, Hi Slew		TBD		TBD	ns/pF
d_{THLLS}	Delta High to Low, Lo Slew		TBD		TBD	ns/pF

Note:
1. Delays based on 35pF loading.

**Actel is offering "Advanced Information" only on '–1' Speed devices. Consult Actel for '–1' device availability.*

A1440A Timing Characteristics (continued)

(Worst-Case Commercial Conditions)

		Preliminary Information		Advanced Information*		
Dedicated (Hard-Wired) I/O Clock Network		**'Std' Speed**		**'–1' Speed**		
Parameter	**Description**	**Min.**	**Max.**	**Min.**	**Max.**	**Units**
t_{IOCKH}	Input Low to High (Pad to I/O Module Input)		3.0		2.6	ns
t_{IOPWH}	Minimum Pulse Width High	3.8		3.3		ns
t_{IOPWL}	Minimum Pulse Width Low	3.8		3.3		ns
t_{IOSAPW}	Minimum Asynchronous Pulse Width	3.8		3.3		ns
t_{IOCKSW}	Maximum Skew		0.4		0.4	ns
t_{IOP}	Minimum Period	8.0		6.8		ns
f_{IOMAX}	Maximum Frequency		125		150	MHz
Dedicated (Hard-Wired) Array Clock Network						
t_{HCKH}	Input Low to High (Pad to S-Module Input)		4.5		3.9	ns
t_{HCKL}	Input High to Low (Pad to S-Module Input)		4.5		3.9	ns
t_{HPWH}	Minimum Pulse Width High	3.8		3.3		ns
t_{HPWL}	Minimum Pulse Width Low	3.8		3.3		ns
t_{HCKSW}	Maximum Skew		0.3		0.3	ns
t_{HP}	Minimum Period	8.0		6.8		ns
f_{HMAX}	Maximum Frequency		125		150	MHz
Routed Array Clock Networks						
t_{RCKH}	Input Low to High (FO=64)		5.5		4.7	ns
t_{RCKL}	Input High to Low (FO=64)		6.0		5.1	ns
t_{RPWH}	Min. Pulse Width High (FO=64)	4.9		4.2		ns
t_{RPWL}	Min. Pulse Width Low (FO=64)	4.9		4.2		ns
t_{RCKSW}	Maximum Skew (FO=128)		1.0		0.9	ns
t_{RP}	Minimum Period (FO=64)	10.0		8.7		ns
f_{RMAX}	Maximum Frequency (FO=64)		100		115	MHz
Clock-to-Clock Skews						
$t_{IOHCKSW}$	I/O Clock to H-Clock Skew	0.0	1.3	0.0	3.0	ns
$t_{IOHCKSW}$	I/O Clock to R-Clock Skew	0.0	3.0	0.0	3.0	ns
t_{HRCKSW}	H-Clock to R-Clock Skew	0.0	1.0	0.0	1.0	ns

Note:
1. Delays based on 35pF loading.

**Actel is offering "Advanced Information" only on '–1' Speed devices. Consult Actel for '–1' device availability.*

A1460A Timing Characteristics

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

		Preliminary Information		Advanced Information*		
Logic Module Propagation Delays[1]		**'Std' Speed**		**'–1' Speed**		
Parameter	**Description**	**Min.**	**Max.**	**Min.**	**Max.**	**Units**
t_{PD}	Internal Array Module		3.0		2.6	ns
t_{CO}	Sequential Clock to Q		3.0		2.6	ns
t_{CLR}	Asynchronous Clear to Q		3.0		2.6	ns
Predicted Routing Delays[2]						
t_{RD1}	FO=1 Routing Delay		1.3		1.1	ns
t_{RD2}	FO=2 Routing Delay		1.8		1.6	ns
t_{RD3}	FO=3 Routing Delay		2.1		1.8	ns
t_{RD4}	FO=4 Routing Delay		2.5		2.2	ns
t_{RD8}	FO=8 Routing Delay		4.2		3.6	ns
Logic Module Sequential Timing						
t_{SUD}	Flip-Flop Data Input Setup	0.8		0.8		ns
t_{HD}	Flip-Flop Data Input Hold	0.5		0.5		ns
t_{SUD}	Latch Data Input Setup	0.8		0.8		ns
t_{HD}	Latch Data Input Hold	0.5		0.5		ns
t_{SUASYN}	Asynchronous Input Setup	TBD		TBD		ns
t_{WASYN}	Asynchronous Pulse Width	4.8		4.1		ns
t_{WCLKA}	Flip-Flop Clock Pulse Width	4.8		4.1		ns
t_A	Flip-Flop Clock Input Period	10.0		8.5		ns
f_{MAX}	Flip-Flop Clock Frequency		100		120	MHz

Notes:

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

**Actel is offering "Advanced Information" only on '–1' Speed devices. Consult Actel for '–1' device availability.*

A1460A Timing Characteristics (continued)

(Worst-Case Commercial Conditions)

		Preliminary Information		Advanced Information*		
I/O Module Input Propagation Delays		**'Std' Speed**		**'–1' Speed**		
Parameter	**Description**	**Min.**	**Max.**	**Min.**	**Max.**	**Units**
t_{INY}	Input Data Pad to Y		4.2		3.6	ns
t_{ICKY}	Input Reg IOCLK Pad to Y		7.0		6.0	ns
t_{OCKY}	Output Reg IOCLK Pad to Y		7.0		6.0	ns
t_{ICLRY}	Input Asynchronous Clear to Y		7.0		6.0	ns
t_{OCLRY}	Output Asynchronous Clear to Y		7.0		6.0	ns
Predicted Input Routing Delays[1]						
t_{IRD1}	FO=1 Routing Delay		1.3		1.1	ns
t_{IRD2}	FO=2 Routing Delay		1.8		1.6	ns
t_{IRD3}	FO=3 Routing Delay		2.1		1.8	ns
t_{IRD4}	FO=4 Routing Delay		2.5		2.2	ns
t_{IRD8}	FO=8 Routing Delay		4.2		3.6	ns
I/O Module Sequential Timing						
t_{INH}	Input F-F Data Hold (w.r.t. IOCLK Pad)	0.0		0.0		ns
t_{INSU}	Input F-F Data Setup (w.r.t. IOCLK Pad)	3.0		3.0		ns
t_{IDEH}	Input Data Enable Hold (w.r.t. IOCLK Pad)	0.0		0.0		ns
t_{IDESU}	Input Data Enable Setup (w.r.t. IOCLK Pad)	9.0		9.0		ns
$t_{ISUASYN}$	Input Asynchronous Setup	TBD		TBD		ns
t_{OUTH}	Output F-F Data Hold (w.r.t. IOCLK Pad)	1.0		1.0		ns
t_{OUTSU}	Output F-F Data Setup (w.r.t. IOCLK Pad)	1.0		1.0		ns
t_{ODEH}	Output Data Enable Hold (w.r.t. IOCLK Pad)	0.5		0.5		ns
t_{ODESU}	Output Data Enable Setup (w.r.t. IOCLK Pad)	2.0		2.0		ns
$t_{OSUASYN}$	Output Asynchronous Setup	TBD		TBD		ns

Note:

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

**Actel is offering "Advanced Information" only on '–1' Speed devices. Consult Actel for '–1' device availability.*

A1460A Timing Characteristics (continued)

(Worst-Case Commercial Conditions)

		Preliminary Information		Advanced Information*		
I/O Module – TTL Output Timing[1]		**'Std' Speed**		**'–1' Speed**		
Parameter	**Description**	**Min.**	**Max.**	**Min.**	**Max.**	**Units**
t_{DHS}	Data to Pad, High Slew		7.5		6.4	ns
t_{DLS}	Data to Pad, Low Slew		12.0		10.2	ns
t_{ENZHS}	Enable to Pad, Z to H/L, Hi Slew		6.0		5.1	ns
t_{ENZLS}	Enable to Pad, Z to H/L, Lo Slew		11.0		9.4	ns
t_{ENHSZ}	Enable to Pad, H/L to Z, Hi Slew		11.6		9.9	ns
t_{ENLSZ}	Enable to Pad, H/L to Z, Lo Slew		11.0		9.4	ns
t_{CKHS}	IOCLK Pad to Pad H/L, Hi Slew		11.6		10.4	ns
t_{CKLS}	IOCLK Pad to Pad H/L, Lo Slew		17.0		15.3	ns
d_{TLHHS}	Delta Low to High, Hi Slew		TBD		TBD	ns/pF
d_{TLHLS}	Delta Low to High, Lo Slew		TBD		TBD	ns/pF
d_{THLHS}	Delta High to Low, Hi Slew		TBD		TBD	ns/pF
d_{THLLS}	Delta High to Low, Lo Slew		TBD		TBD	ns/pF
I/O Module – CMOS Output Timing[1]						
t_{DHS}	Data to Pad, High Slew		9.3		7.9	ns
t_{DLS}	Data to Pad, Low Slew		17.5		14.9	ns
t_{ENZHS}	Enable to Pad, Z to H/L, Hi Slew		7.8		6.6	ns
t_{ENZLS}	Enable to Pad, Z to H/L, Lo Slew		13.3		11.3	ns
t_{ENHSZ}	Enable to Pad, H/L to Z, Hi Slew		11.0		9.4	ns
t_{ENLSZ}	Enable to Pad, H/L to Z, Lo Slew		11.0		9.4	ns
t_{CKHS}	IOCLK Pad to Pad H/L, Hi Slew		13.8		12.4	ns
t_{CKLS}	IOCLK Pad to Pad H/L, Lo Slew		19.3		17.4	ns
d_{TLHHS}	Delta Low to High, Hi Slew		TBD		TBD	ns/pF
d_{TLHLS}	Delta Low to High, Lo Slew		TBD		TBD	ns/pF
d_{THLHS}	Delta High to Low, Hi Slew		TBD		TBD	ns/pF
d_{THLLS}	Delta High to Low, Lo Slew		TBD		TBD	ns/pF

Note:
1. Delays based on 35pF loading.

**Actel is offering "Advanced Information" only on '–1' Speed devices. Consult Actel for '–1' device availability.*

A1460A Timing Characteristics (continued)

(Worst-Case Commercial Conditions)

		Preliminary Information		Advanced Information*		
Dedicated (Hard-Wired) I/O Clock Network		**'Std' Speed**		**'–1' Speed**		
Parameter	**Description**	**Min.**	**Max.**	**Min.**	**Max.**	**Units**
t_{IOCKH}	Input Low to High (Pad to I/O Module Input)		3.5		3.0	ns
t_{IOPWH}	Minimum Pulse Width High	4.8		4.1		ns
t_{IOPWL}	Minimum Pulse Width Low	4.8		4.1		ns
t_{IOSAPW}	Minimum Asynchronous Pulse Width	3.8		3.3		ns
t_{IOCKSW}	Maximum Skew		0.8		0.7	ns
t_{IOP}	Minimum Period	10.0		8.5		ns
f_{IOMAX}	Maximum Frequency		100		120	MHz
Dedicated (Hard-Wired) Array Clock Network						
t_{HCKH}	Input Low to High (Pad to S-Module Input)		5.5		4.7	ns
t_{HCKL}	Input High to Low (Pad to S-Module Input)		5.5		4.7	ns
t_{HPWH}	Minimum Pulse Width High	4.8		4.1		ns
t_{HPWL}	Minimum Pulse Width Low	4.8		4.1		ns
t_{HCKSW}	Maximum Skew		0.8		0.7	ns
t_{HP}	Minimum Period	10.0		8.5		ns
f_{HMAX}	Maximum Frequency		100		120	MHz
Routed Array Clock Networks						
t_{RCKH}	Input Low to High (FO=256)		9.0		7.7	ns
t_{RCKL}	Input High to Low (FO=256)		9.0		7.7	ns
t_{RPWH}	Min. Pulse Width High (FO=256)	6.1		5.4		ns
t_{RPWL}	Min. Pulse Width Low (FO=256)	6.1		5.4		ns
t_{RCKSW}	Maximum Skew (FO=128)		1.8		1.6	ns
t_{RP}	Minimum Period (FO=256)	12.5		11.1		ns
f_{RMAX}	Maximum Frequency (FO=256)		80		90	MHz
Clock-to-Clock Skews						
$t_{IOHCKSW}$	I/O Clock to H-Clock Skew	0.0	3.5	0.0	3.5	ns
$t_{IOHCKSW}$	I/O Clock to R-Clock Skew	0.0	5.0	0.0	5.0	ns
t_{HRCKSW}	H-Clock to R-Clock Skew	0.0	3.0	0.0	3.0	ns

Note:
1. Delays based on 35pF loading.

**Actel is offering "Advanced Information" only on '–1' Speed devices. Consult Actel for '–1' device availability.*

A14100A Timing Characteristics

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

		Preliminary Information		Advanced Information*		
Logic Module Propagation Delays[1]		**'Std' Speed**		**'–1' Speed**		
Parameter	**Description**	**Min.**	**Max.**	**Min.**	**Max.**	**Units**
t_{PD}	Internal Array Module		3.0		2.6	ns
t_{CO}	Sequential Clock to Q		3.0		2.6	ns
t_{CLR}	Asynchronous Clear to Q		3.0		2.6	ns
Predicted Routing Delays[2]						
t_{RD1}	FO=1 Routing Delay		1.3		1.1	ns
t_{RD2}	FO=2 Routing Delay		1.8		1.6	ns
t_{RD3}	FO=3 Routing Delay		2.1		1.8	ns
t_{RD4}	FO=4 Routing Delay		2.5		2.2	ns
t_{RD8}	FO=8 Routing Delay		4.2		3.6	ns
Logic Module Sequential Timing						
t_{SUD}	Flip-Flop Data Input Setup	0.8		0.8		ns
t_{HD}	Flip-Flop Data Input Hold	0.5		0.5		ns
t_{SUD}	Latch Data Input Setup	0.8		0.8		ns
t_{HD}	Latch Data Input Hold	0.5		0.5		ns
t_{SUASYN}	Asynchronous Input Setup	TBD		TBD		ns
t_{WASYN}	Asynchronous Pulse Width	4.8		4.1		ns
t_{WCLKA}	Flip-Flop Clock Pulse Width	4.8		4.1		ns
t_A	Flip-Flop Clock Input Period	10.0		8.5		ns
f_{MAX}	Flip-Flop Clock Frequency		100		120	MHz

Notes:

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

**Actel is offering "Advanced Information" only on '–1' Speed devices. Consult Actel for '–1' device availability.*

A14100A Timing Characteristics (continued)

(Worst-Case Commercial Conditions)

		Preliminary Information		Advanced Information*		
I/O Module Input Propagation Delays		**'Std' Speed**		**'–1' Speed**		
Parameter	**Description**	**Min.**	**Max.**	**Min.**	**Max.**	**Units**
t_{INY}	Input Data Pad to Y		4.2		3.6	ns
t_{ICKY}	Input Reg IOCLK Pad to Y		7.0		6.0	ns
t_{OCKY}	Output Reg IOCLK Pad to Y		7.0		6.0	ns
t_{ICLRY}	Input Asynchronous Clear to Y		7.0		6.0	ns
t_{OCLRY}	Output Asynchronous Clear to Y		7.0		6.0	ns
Predicted Input Routing Delays[1]						
t_{IRD1}	FO=1 Routing Delay		1.3		1.1	ns
t_{IRD2}	FO=2 Routing Delay		1.8		1.6	ns
t_{IRD3}	FO=3 Routing Delay		2.1		1.8	ns
t_{IRD4}	FO=4 Routing Delay		2.5		2.2	ns
t_{IRD8}	FO=8 Routing Delay		4.2		3.6	ns
I/O Module Sequential Timing						
t_{INH}	Input F-F Data Hold (w.r.t. IOCLK Pad)	0.0		0.0		ns
t_{INSU}	Input F-F Data Setup (w.r.t. IOCLK Pad)	3.0		3.0		ns
t_{IDEH}	Input Data Enable Hold (w.r.t. IOCLK Pad)	0.0		0.0		ns
t_{IDESU}	Input Data Enable Setup (w.r.t. IOCLK Pad)	9.0		9.0		ns
$t_{ISUASYN}$	Input Asynchronous Setup	TBD		TBD		ns
t_{OUTH}	Output F-F Data Hold (w.r.t. IOCLK Pad)	1.0		1.0		ns
t_{OUTSU}	Output F-F Data Setup (w.r.t. IOCLK Pad)	1.0		1.0		ns
t_{ODEH}	Output Data Enable Hold (w.r.t. IOCLK Pad)	0.5		0.5		ns
t_{ODESU}	Output Data Enable Setup (w.r.t. IOCLK Pad)	2.0		2.0		ns
$t_{OSUASYN}$	Output Asynchronous Setup	TBD		TBD		ns

Note:

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

**Actel is offering "Advanced Information" only on '–1' Speed devices. Consult Actel for '–1' device availability.*

A14100A Timing Characteristics (continued)

(Worst-Case Commercial Conditions)

		Preliminary Information		Advanced Information*		
I/O Module – TTL Output Timing[1]		**'Std' Speed**		**'–1' Speed**		
Parameter	**Description**	**Min.**	**Max.**	**Min.**	**Max.**	**Units**
t_{DHS}	Data to Pad, High Slew		7.5		6.4	ns
t_{DLS}	Data to Pad, Low Slew		12.0		10.2	ns
t_{ENZHS}	Enable to Pad, Z to H/L, Hi Slew		6.0		5.1	ns
t_{ENZLS}	Enable to Pad, Z to H/L, Lo Slew		11.0		9.4	ns
t_{ENHSZ}	Enable to Pad, H/L to Z, Hi Slew		12.0		10.2	ns
t_{ENLSZ}	Enable to Pad, H/L to Z, Lo Slew		11.0		9.4	ns
t_{CKHS}	IOCLK Pad to Pad H/L, Hi Slew		12.0		10.8	ns
t_{CKLS}	IOCLK Pad to Pad H/L, Lo Slew		17.0		15.3	ns
d_{TLHHS}	Delta Low to High, Hi Slew		TBD		TBD	ns/pF
d_{TLHLS}	Delta Low to High, Lo Slew		TBD		TBD	ns/pF
d_{THLHS}	Delta High to Low, Hi Slew		TBD		TBD	ns/pF
d_{THLLS}	Delta High to Low, Lo Slew		TBD		TBD	ns/pF
I/O Module – CMOS Output Timing[1]						
t_{DHS}	Data to Pad, High Slew		9.3		7.9	ns
t_{DLS}	Data to Pad, Low Slew		17.5		14.9	ns
t_{ENZHS}	Enable to Pad, Z to H/L, Hi Slew		7.8		6.6	ns
t_{ENZLS}	Enable to Pad, Z to H/L, Lo Slew		13.3		11.3	ns
t_{ENHSZ}	Enable to Pad, H/L to Z, Hi Slew		12.0		10.0	ns
t_{ENLSZ}	Enable to Pad, H/L to Z, Lo Slew		11.0		9.4	ns
t_{CKHS}	IOCLK Pad to Pad H/L, Hi Slew		13.8		12.4	ns
t_{CKLS}	IOCLK Pad to Pad H/L, Lo Slew		19.3		17.4	ns
d_{TLHHS}	Delta Low to High, Hi Slew		TBD		TBD	ns/pF
d_{TLHLS}	Delta Low to High, Lo Slew		TBD		TBD	ns/pF
d_{THLHS}	Delta High to Low, Hi Slew		TBD		TBD	ns/pF
d_{THLLS}	Delta High to Low, Lo Slew		TBD		TBD	ns/pF

Note:
1. Delays based on 35pF loading.

**Actel is offering "Advanced Information" only on '–1' Speed devices. Consult Actel for '–1' device availability.*

A14100A Timing Characteristics (continued)

(Worst-Case Commercial Conditions)

		Preliminary Information		Advanced Information*		
Dedicated (Hard-Wired) I/O Clock Network		**'Std' Speed**		**'–1' Speed**		
Parameter	**Description**	**Min.**	**Max.**	**Min.**	**Max.**	**Units**
t_{IOCKH}	Input Low to High (Pad to I/O Module Input)		3.5		3.0	ns
t_{IOPWH}	Minimum Pulse Width High	4.8		4.1		ns
t_{IOPWL}	Minimum Pulse Width Low	4.8		4.1		ns
t_{IOSAPW}	Minimum Asynchronous Pulse Width	3.8		3.3		ns
t_{IOCKSW}	Maximum Skew		0.8		0.7	ns
t_{IOP}	Minimum Period	10.0		8.5		ns
f_{IOMAX}	Maximum Frequency		100		120	MHz
Dedicated (Hard-Wired) Array Clock Network						
t_{HCKH}	Input Low to High (Pad to S-Module Input)		5.5		4.7	ns
t_{HCKL}	Input High to Low (Pad to S-Module Input)		5.5		4.7	ns
t_{HPWH}	Minimum Pulse Width High	4.8		4.1		ns
t_{HPWL}	Minimum Pulse Width Low	4.8		4.1		ns
t_{HCKSW}	Maximum Skew		0.8		0.7	ns
t_{HP}	Minimum Period	10.0		8.5		ns
f_{HMAX}	Maximum Frequency		100		120	MHz
Routed Array Clock Networks						
t_{RCKH}	Input Low to High (FO=256)		9.0		7.7	ns
t_{RCKL}	Input High to Low (FO=256)		9.0		7.7	ns
t_{RPWH}	Min. Pulse Width High (FO=256)	6.1		5.4		ns
t_{RPWL}	Min. Pulse Width Low (FO=256)	6.1		5.4		ns
t_{RCKSW}	Maximum Skew (FO=128)		1.8		1.6	ns
t_{RP}	Minimum Period (FO=256)	12.5		11.1		ns
f_{RMAX}	Maximum Frequency (FO=256)		80		90	MHz
Clock-to-Clock Skews						
$t_{IOHCKSW}$	I/O Clock to H-Clock Skew	0.0	3.5	0.0	3.5	ns
$t_{IOHCKSW}$	I/O Clock to R-Clock Skew	0.0	5.0	0.0	5.0	ns
t_{HRCKSW}	H-Clock to R-Clock Skew	0.0	3.0	0.0	3.0	ns

Note:
1. Delays based on 35pF loading.

**Actel is offering "Advanced Information" only on '–1' Speed devices. Consult Actel for '–1' device availability.*

Macro Library

Hard Macros—Combinatorial

Function	Macro	Description	Modules S	Modules C
ACT 3 Combinatorial Logic Module	CM8	Combinational Module (Full ACT 3 Logic Module)		1
ACT 3 Sequential Logic Module	DFM8A	4-bit D-Type Flip-Flop with Multiplexed Data, active low Clear, and active high clock	1	
	DFM8B	4-bit D-Type Flip-Flop with Multiplexed Data, active low Clear, and active low clock	1	
Adder	FA1A	1-bit adder, carry in and carry out active low, A-input active low		2
	FA1B	1-bit adder, carry in and carry out active low		2
	FA2A	2-bit adder, carry in and carry out active low, A0 and A1 inputs active low		2
	HA1	Half-Adder		2
	HA1A	Half-Adder with active low A-input		2
	HA1B	Half-Adder with active low carry out and sum		2
	HA1C	Half-Adder with active low carry out		2
AND	AND2	2-input AND		1
	AND2A	2-input AND with active low A-input		1
	AND2B	2-input AND with active low inputs		1
	AND3	3-input AND		1
	AND3A	3-input AND with active low A-input		1
	AND3B	3-input AND with active low A- and B-inputs		1
	AND3C	3-input AND with active low inputs		1
	AND4	4-input AND		1
	AND4A	4-input AND with active low A-input		1
	AND4B	4-input AND with active low A- and B-inputs		1
	AND4C	4-input AND with active low A-, B-, and C-inputs		1
	AND4D	4-input AND with active low inputs		2
	AND5B	5-input AND with active low A- and B-inputs		1
AND-OR	AO1	3-input AND-OR		1
	AO10	5-input AND-OR-AND		1
	AO11	3-input AND-CR		1
	AO1A	3-input AND-OR with active low A-input		1
	AO1B	3-input AND-OR with active low C-input		1
	AO1C	3-input AND-OR with active low A- and C-inputs		1
	AO1D	3-input AND-OR with active low A- and B-inputs		1
	AO1E	3-input AND-OR with active low inputs		1
	AO2	4-input AND-OR		1
	AO2A	4-input AND-OR with active low A-input		1
	AO2B	4-input AND-OR with active low A- and B-inputs		1
	AO2C	4-input AND-OR with active low A- and C-inputs		1
	AO2D	4-input AND-OR with active low A-, B-, and C-inputs		1
	AO2E	4-input AND-OR with active low inputs		1
	AO3	4-input AND-OR		1
	AO3A	4-input AND-OR		1
	AO3B	4-input AND-OR		1
	AO3C	4-input AND-OR		1
	AO4A	4-input AND-OR		1
	AO5A	4-input AND-OR		1
	AO6	2-wide 4-input AND-OR		1

Hard Macros—Combinatorial (continued)

Function	Macro	Description	Modules S	Modules C
AND-OR	AO6A	2-wide 4-input AND-OR with active low D-input		1
	AO7	5-input AND-OR		1
	AO8	5-input AND-OR with active low C- and D-inputs		1
	AO9	5-input AND-OR		1
	AOI1	3-input AND-OR-INVERT		1
	AOI1A	3-input AND-OR-INVERT with active low A-input		1
	AOI1B	3-input AND-OR-INVERT with active low C-input		1
	AOI1C	3-input AND-OR-INVERT with active low A- and B-inputs		1
	AOI1D	3-input AND-OR-INVERT with active low inputs		1
	AOI2A	4-input AND-OR-INVERT with active low A-input		1
	AOI2B	4-input AND-OR-INVERT with active low A- and C-inputs		1
	AOI3A	4-input AND-OR-INVERT with active low inputs		1
	AOI4	2-wide 4-input AND-OR-INVERT		2
	AOI4A	2-wide 4-input AND-OR-INVERT with active low C-input		1
AND-XOR	AX1	3-input AND-XOR with active low A-input		1
	AX1A	3-input AND-XOR-INVERT with active low A-input		2
	AX1B	3-input AND-XOR with active low A- and B-inputs		1
	AX1C	3-input AND-XOR		1
Buffer	BUF	Buffer, with active high input and output		1
	BUFA	Buffer, with active low input and output		1
Clock Net	CLKINT	Clock Net Interface	0	0
	GAND2	2-input AND Clock Net		1
	GMX4	4-to-1 Multiplexor Clock Net		1
	GNAND2	2-input NAND Clock Net		1
	GNOR2	2-input NOR Clock Net		1
	GOR2	2-input OR Clock Net		1
	GXOR2	2-input Exclusive OR Clock Net		1
Inverter	INV	Inverter with active low output		1
	INVA	Inverter with active low input		1
Majority	MAJ3	3-input complex AND-OR		1
MUX	MX2	2-to-1 Multiplexor		1
	MX2A	2-to-1 Multiplexor with active low A-input		1
	MX2B	2-to-1 Multiplexor with active low B-input		1
MUX	MX2C	2-to-1 Multiplexor with active low output		1
	MX4	4-to-1 Multiplexor		1
	MXC1	Boolean		2
	MXT	Boolean		2
NAND	NAND2	2-input NAND		1
	NAND2A	2-input NAND with active low A-input		1
	NAND2B	2-input NAND with active low inputs		1
	NAND3	3-input NAND		1
	NAND3A	3-input NAND with active low A-input		1
	NAND3B	3-input NAND with active low A- and B-inputs		1
	NAND3C	3-input NAND with active low inputs		1
	NAND4	4-input NAND		2
	NAND4A	4-input NAND with active low A-input		1
	NAND4B	4-input NAND with active low A- and B-inputs		1
	NAND4C	4-input NAND with active low A-, B-, and C-inputs		1
	NAND4D	4-input NAND with active low inputs		1

Hard Macros—Combinatorial (continued)

Function	Macro	Description	Modules S	Modules C
NAND	NAND5C	5-input NAND with active low A-, B-, and C-inputs		1
NOR	NOR2	2-input NOR		1
NOR	NOR2A	2-input NOR with active low A-input		1
	NOR2B	2-input NOR with active low inputs		1
	NOR3	3-input NOR		1
	NOR3A	3-input NOR with active low A-input		1
	NOR3B	3-input NOR with active low A- and B-inputs		1
	NOR3C	3-input NOR with active low inputs		1
	NOR4	4-input NOR		2
	NOR4A	4-input NOR with active low A-input		1
	NOR4B	4-input NOR with active low A- and B-inputs		1
	NOR4C	4-input NOR with active low A-, B-, and C-inputs		1
	NOR4D	4-input NOR with active low inputs		1
	NOR5C	5-input NOR with active low A-, B-, and C-inputs		1
OR	OR2	2-input OR		1
	OR2A	2-input OR with active low A-input		1
	OR2B	2-input OR with active low inputs		1
	OR3	3-input OR		1
	OR3A	3-input OR with active low A-input		1
	OR3B	3-input OR with active low A- and B-inputs		1
	OR3C	3-input OR with active low inputs		1
	OR4	4-input OR		1
	OR4A	4-input OR with active low A-input		1
	OR4B	4-input OR with active low A- and B-input		1
	OR4C	4-input OR with active low A-, B-, and C-inputs		1
	OR4D	4-input OR with active low inputs		2
	OR5B	5-input OR with active low A- and B-inputs		1
OR-AND	OA1	3-input OR-AND		1
	OA1A	3-input OR-AND with active low A-input		1
	OA1B	3-input OR-AND with active low C-input		1
	OA1C	3-input OR-AND with active low A- and C-inputs		1
	OA2	2-wide 4-input OR-AND		1
	OA2A	2 wide 4-input OR-AND with active low A-input		1
	OA3	4-input OR-AND		1
	OA3A	4-input OR-AND with active low C-input		1
	OA3B	4-input OR-AND with active low A- and C-inputs		1
	OA4	4-input OR-AND		1
	OA4A	4-input OR-AND with active low C-input		1
	OA5	4-input complex OR-AND		1
	OAI1	3-input OR-AND-INVERT		1
	OAI2A	4-input OR-AND-INVERT with active low D-input		1
	OAI3	4-input OR-AND-INVERT		1
	OAI3A	4-input OR-AND-INVERT with active low C- and D-inputs		1
XNOR	XNOR	2-input XNOR		1
XNOR-AND	XA1A	3-input XNOR-AND		1
XNOR-OR	XO1A	3-input XNOR-OR		1
XOR	XOR	2-input XOR		1
XOR-AND	XA1	3-input XOR-AND		1
XOR-OR	XO1	3-input XOR-OR		1

Hard Macros—Sequential

Function	Macro	Description	Modules S	Modules C
D-Type	DF1	D-Type Flip-Flop	1	
	DF1A	D-Type Flip-Flop with active low output	1	
	DF1B	D-Type Flip-Flop with active low clock	1	
	DF1C	D-Type Flip-Flop with active low clock and output	1	
	DFC1	D-Type Flip-Flop with active high Clear	1	1
	DFC1A	D-Type Flip-Flop with active high Clear and active low clock	1	1
	DFC1B	D-Type Flip-Flop with active low Clear	1	
	DFC1D	D-Type Flip-Flop with active low Clear and clock	1	
	DFE	D-Type Flip-Flop with active high Enable	1	
	DFE1B	D-Type Flip-Flop with active low Enable	1	
	DFE1C	D-Type Flip-Flop with active low Enable and clock	1	
	DFE3A	D-Type Flip-Flop with Enable and active low Clear	1	
	DFE3B	D-Type Flip-Flop with Enable and active low Clear and clock	1	
	DFE3C	D-Type Flip-Flop with active low Enable and Clear	1	
	DFE3D	D-Type Flip-Flop with active low Enable, Clear, and clock	1	
	DFEA	D-Type Flip-Flop with Enable and active low clock	1	
	DFM	2-bit D-Type Flip-Flop with Multiplexed Data	1	
	DFM1B	2-bit D-Type Flip-Flop with Multiplexed Data and active low output	1	
	DFM1C	2-bit D-Type Flip-Flop with Multiplexed Data and active low clock and output	1	
	DFM3	2-bit D-Type Flip-Flop with Multiplexed Data and Clear	1	1
	DFM3B	2-bit D-Type Flip-Flop with Multiplexed Data and active low Clear and clock	1	
	DFM3E	2-bit D-Type Flip-Flop with Multiplexed Data, Clear, and active low clock	1	1
	DFM4C	2-bit D-Type Flip-Flop with Multiplexed Data and active low Preset and output	1	
	DFM4D	2-bit D-Type Flip-Flop with Multiplexed Data and active low Preset, clock, and output	1	
	DFM6A	4-bit D-Type Flip-Flop with Multiplexed Data, active low Clear, and active high Clock	1	
	DFM6B	4-bit D-Type Flip-Flop with Multiplexed Data, active low Clear, and clock	1	
	DFM7A	4-bit D-Type Flip-Flop with Multiplexed Data, active low Clear, and active high clock	1	
	DFM7B	4-bit D-Type Flip-Flop with Multiplexed Data, active low Clear and clock	1	
	DFMA	2-bit D-Type Flip-Flop with Multiplexed Data and active low clock	1	
	DFMB	2-bit D-Type Flip-Flop with Multiplexed Data and active low Clear	1	
	DFME1A	2-bit D-Type Flip-Flop with Multiplexed Data and active low Enable	1	
	DFP1	D-Type Flip-Flop with active high Preset		2
	DFP1A	D-Type Flip-Flop with active high Preset and active low clock		2
	DFP1B	D-Type Flip-Flop with active low Preset		2
	DFP1C	D-Type Flip-Flop with active high Preset and active low output	1	1
	DFP1D	D-Type Flip-Flop with active low Preset and clock		2
	DFP1E	D-Type Flip-Flop with active low Preset and output	1	
	DFP1F	D-Type Flip-Flop with active high Preset and active low clock and output	1	1
	DFP1G	D-Type Flip-Flop with active low Preset, clock, and output	1	
	DFPC	D-Type Flip-Flop with active high Preset, active low Clear, and active high clock		2
	DFPCA	D-Type Flip-Flop with active high Preset, and active low Clear and clock		2
J-K Type	JKF	JK Flip-Flop with active low K-input	1	

Hard Macros—Sequential (continued)

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

Input/Output Macros

Function	Macro	Description	I/O Modules
Buffer	BBHS	Bidirectional Buffer, High Slew	1
	BBUFTH	Bidirectional Buffer, Tristate Enable, High Slew	1
	BBUFTL	Bidirectional Buffer, Tristate Enable, Low Slew	1
	BIBUF	Bidirectional Buffer, High Slew (with hidden buffer at Y pin)	1
	HCLKBUF	Dedicated High-Speed S-Module Clock Buffer	1
	IBUF	Input Buffer	1
	INBUF	Input Buffer	1
	IOCLKBUF	Dedicated I/O Module Clock Buffer	1
	IOPCLBUF	Dedicated I/O Module IOPCL Buffer	1
	OBHS	Output buffer, High Slew	1
	OBUFTH	Output Buffer, Tristate Enable, High Slew	1
	OBUFTL	Output Buffer, Tristate Enable, Low Slew	1
	OUTBUF	Output Buffer, High Slew	1
Bidirectional	BRECTH	Bidirectional, Output Register with Clear, Data Enable, Tristate Enable, High Slew	1
	BRECTL	Bidirectional, Output Register with Clear, Data Enable, Tristate Enable, Low Slew	1
	BREPTH	Bidirectional, Output Register with Preset, Data Enable, Tristate Enable, High Slew	1
	BREPTL	Bidirectional, Output Register with Preset, Data Enable, Tristate Enable, Low Slew	1
	CLKBIBUF	Bidirectional with Input Dedicated to Clock Network	1
	DECETH	Bidirectional, Double Registered with Clear, Data Enable, Tristate Enable, High Slew	1
	DECETL	Bidirectional, Double Registered with Clear, Data Enable, Tristate Enable, Low Slew	1
	DEPETH	Bidirectional, Double Registered with Preset, Data Enable, Tristate Enable, High Slew	1
	DEPETL	Bidirectional, Double Registered with Preset, Data Enable, Tristate Enable, Low Slew	1
Input	CLKBUF	Input for Dedicated Routed Clock Network	1
	IREC	Input Register with Clear	1
	IREP	Input Register with Preset	1
Output	FECTMH	Output Register with Muxed Feedback, Clear, Data Enable, Tristate Enable, High Slew	1
	FECTML	Output Register with Muxed Feedback, Clear, Data Enable, Tristate Enable, Low Slew	1
	FEPTMH	Output Register with Muxed Feedback, Preset, Data Enable, Tristate Enable, High Slew	1
	FEPTML	Output Register with Muxed Feedback, Preset, Data Enable, Tristate Enable, Low Slew	1
	ORECTH	Output Register with Clear, Data Enable, Tristate Enable, High Slew	1
	ORECTL	Output Register with Clear, Data Enable, Tristate Enable, Low Slew	1
	OREPTH	Output Register with Preset, Data Enable, Tristate Enable, High Slew	1
	OREPTL	Output Register with Preset, Data Enable, Tristate Enable, Low Slew	1
	TBHS	Tristate output, High Slew	1
	TRIBUFF	Tristate output, High Slew	1

Soft Macros

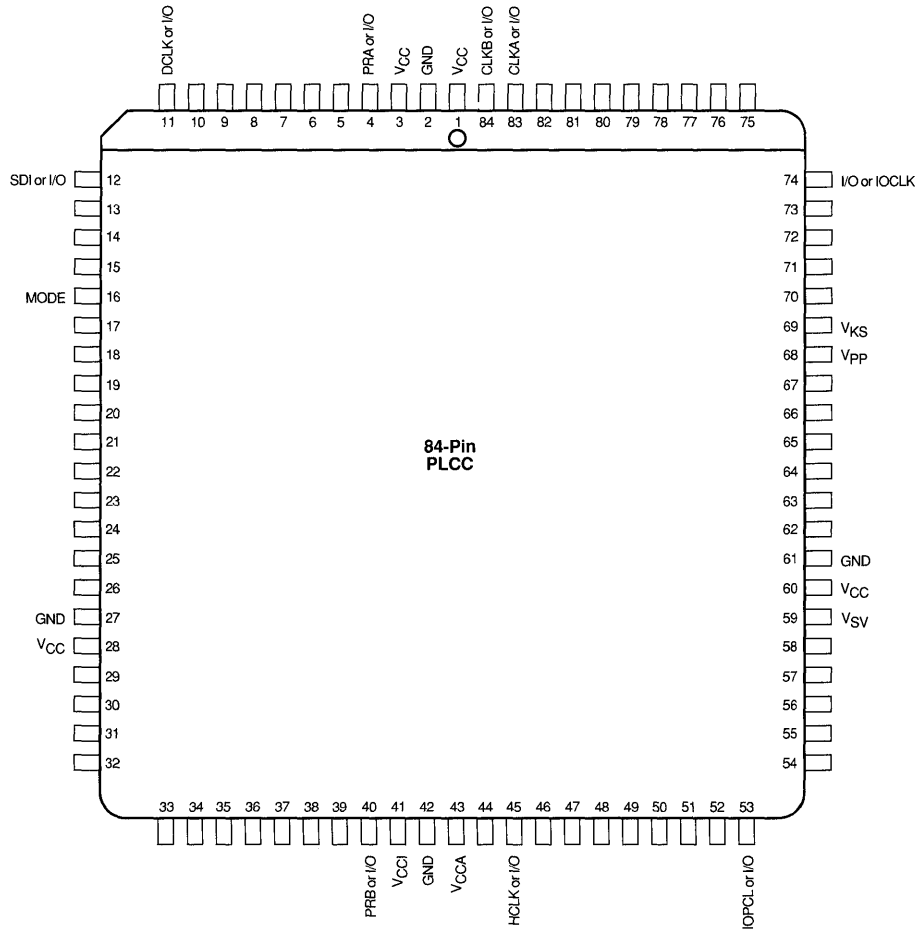
Function	Macro	Description	Maximum Logic Levels	Modules S	Modules C
Adder	FADD10	10-bit adder	3		56
	FADD12	12-bit adder	4		9
	FADD16	16-bit adder	5		97
	FADD8	8-bit adder	4		44
	FADD9	9-bit adder with active low carry out	3		49
	VAD16C	Very fast 16-bit adder, no Carry in	3		97
	VADC16C	Very fast 16-bit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

Soft Macros—TTL Equivalent

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

Package Pin Assignments

84-Pin PLCC (Top View)

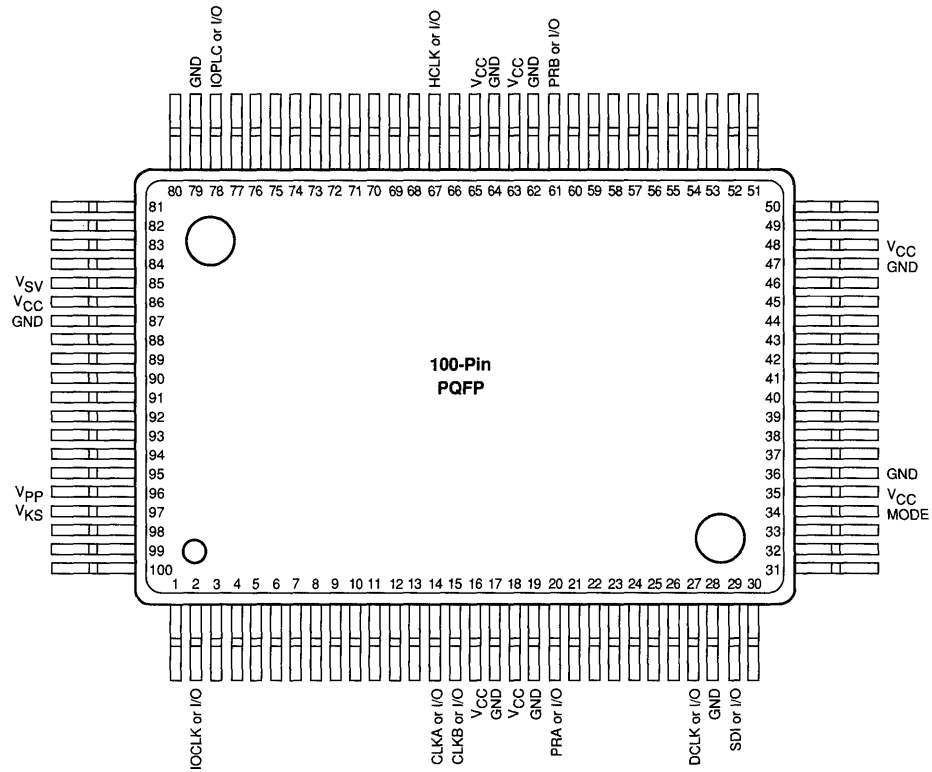


Notes:

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

Package Pin Assignments (continued)

100-Pin PQFP (Top View)

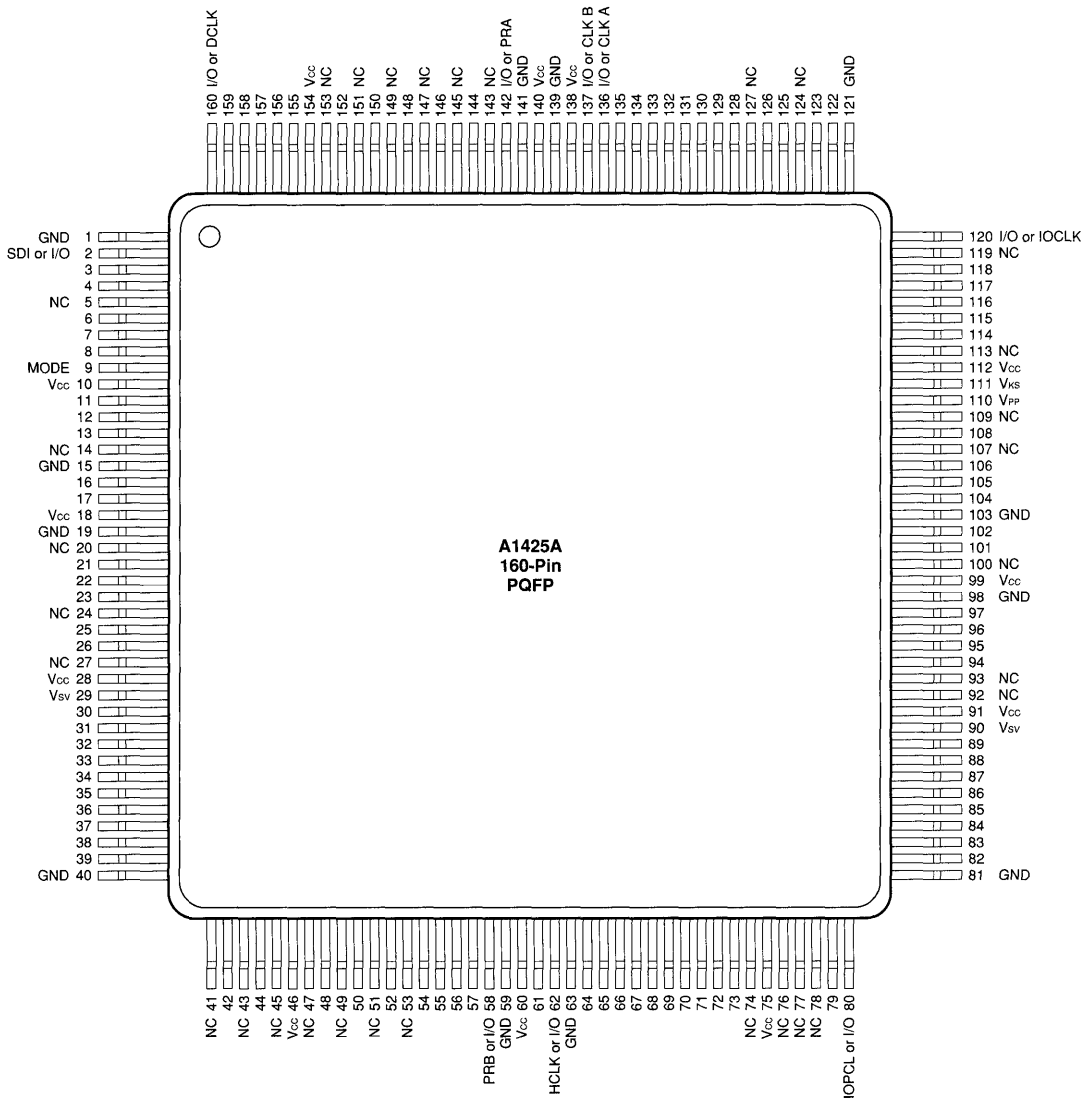


Notes:

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

Package Pin Assignments (continued)

160-Pin PQFP (Top View)

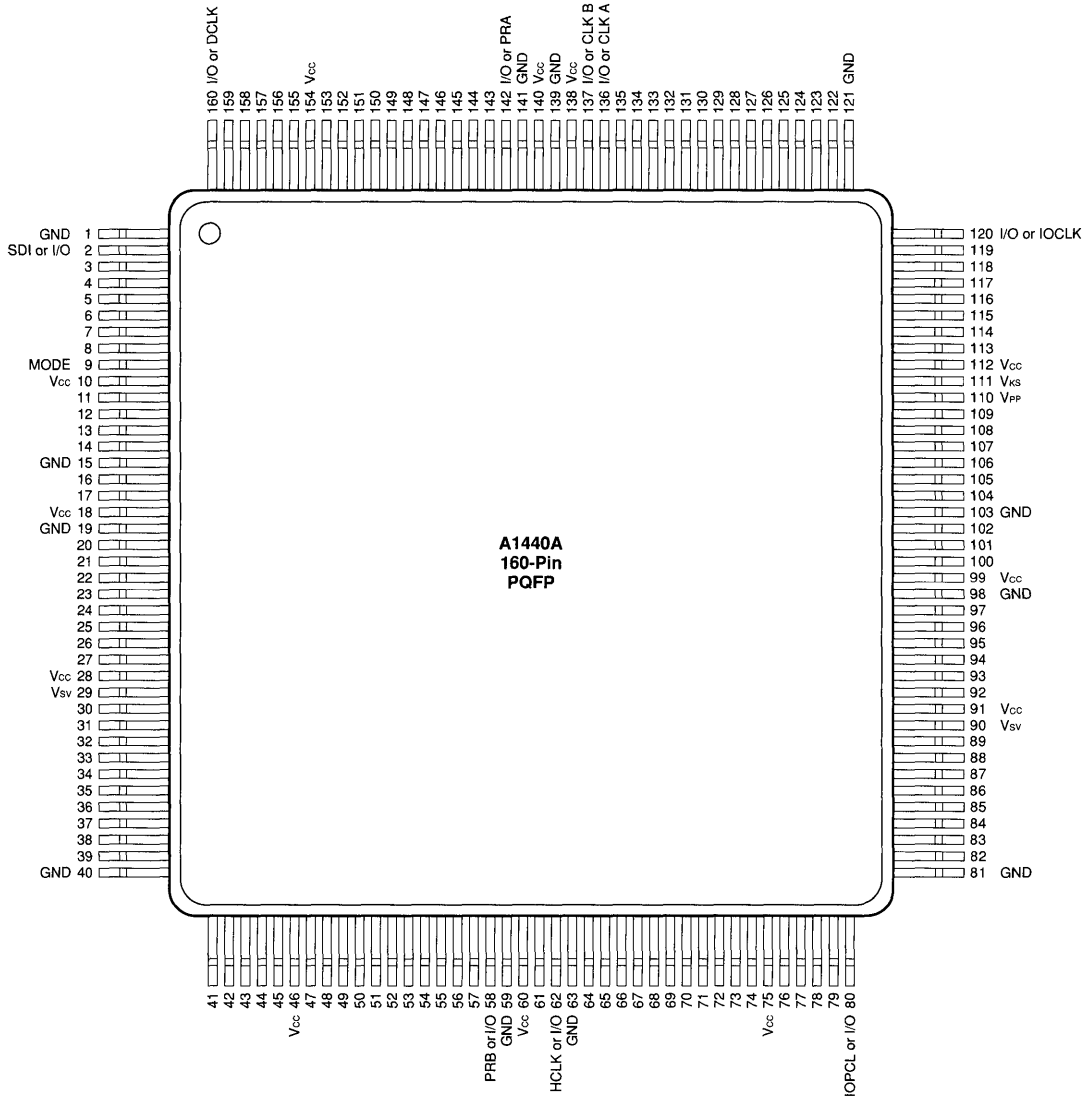


Notes:

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

Package Pin Assignments (continued)

160-Pin PQFP (Top View)

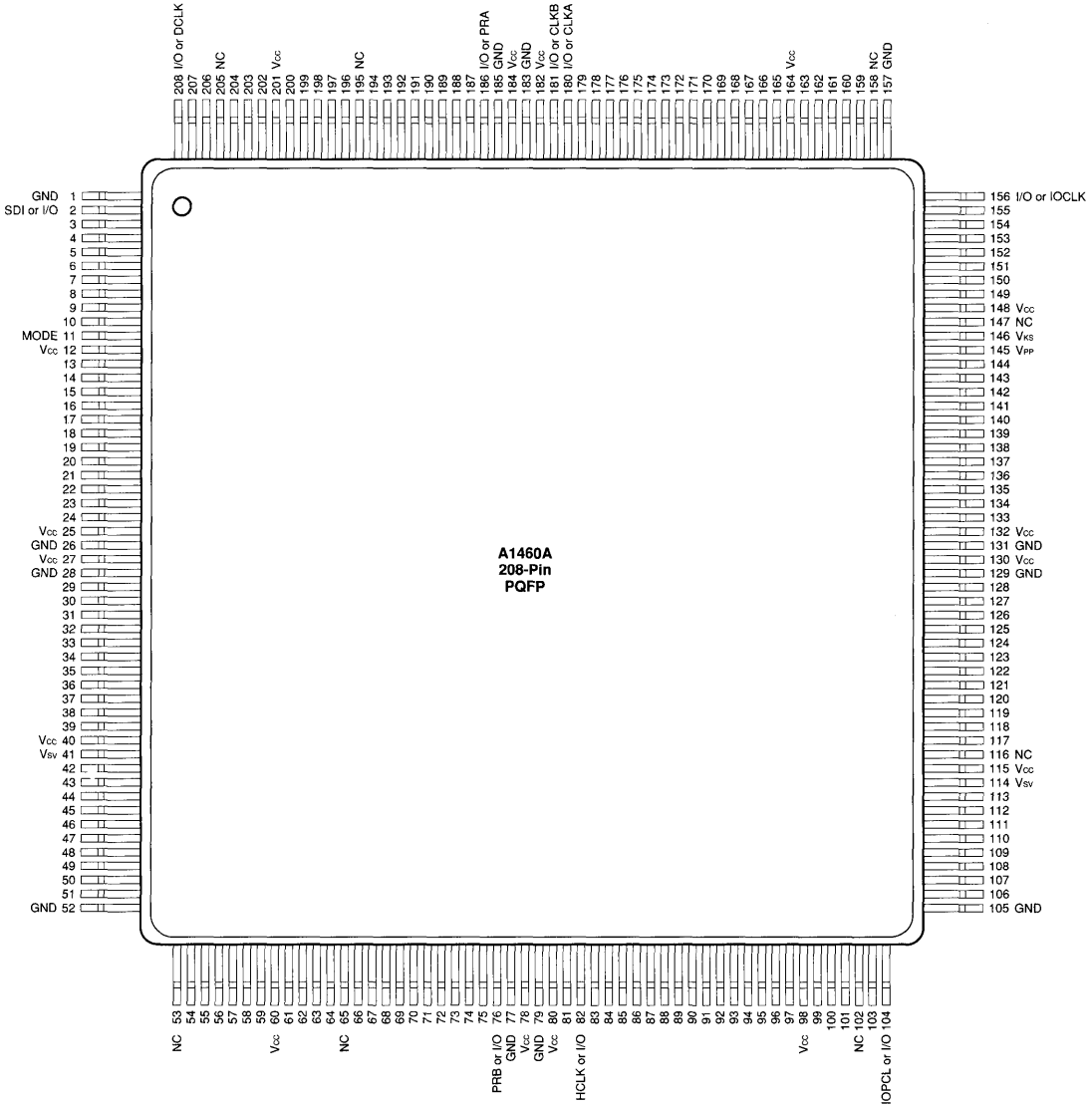


Notes:

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

Package Pin Assignments (continued)

208-Pin PQFP (Top View)

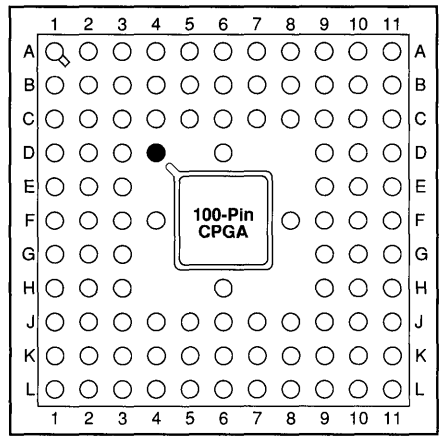


Notes:

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

Package Pin Assignments (continued)

100-Pin CPGA (Top View)



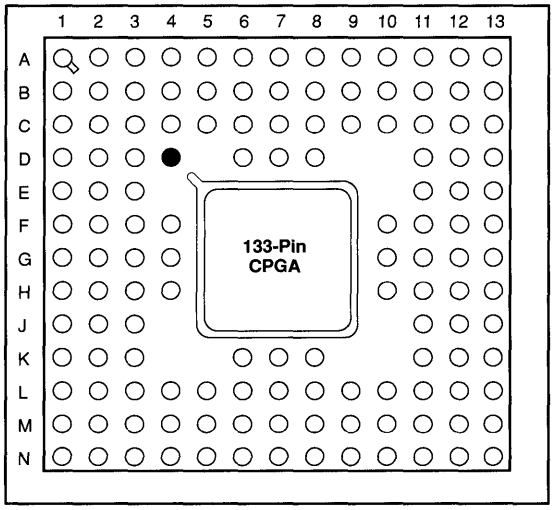
Signal	Pad Number	Location
CLKA or I/O	94	C7
CLKB or I/O	95	D6
DCLK or I/O	107	C4
GND	1, 9, 21, 37, 39, 49, 55, 63, 75, 87, 97, 99	C3, F3, J3, C6, J6, J8, C9, F9, J9
HCLK or I/O	42	H6
IOCLK or I/O	81	C10
IOPCL or I/O	54	K9
MODE	7	C2
PRA OR I/O	100	A6
PRB or I/O	36	L3
SDI or I/O	2	B3
V_{CC}	8, 14, 22, 38, 40, 62, 68, 76, 96, 98	F2, K2, B6, K6, B10, F10, K10
V_{KS}	74	E9
V_{PP}	73	E11
V_{SV}	23, 61	G2

Notes:

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

Package Pin Assignments (continued)

133-Pin CPGA (Top View)



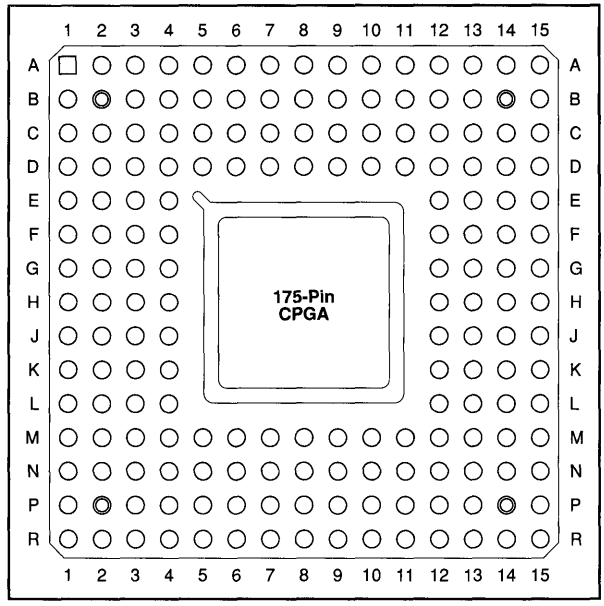
Signal	Pad Number	Location
CLKA or I/O	115	D7
CLKB or I/O	116	B6
DCLK or I/O	134	D4
GND	1, 10, 22, 35, 36, 48, 50, 64, 68, 79, 93, 101, 106, 118, 120, 132	A2, C3, C7, C11, C12, G3, G11, L3, L7, L11, M3, N1
HCLK or I/O	53	K7
IOCLK or I/O	100	C10
IOPCL or I/O	67	L10
MODE	8	E3
NC	—	A1, A7, A13, G1, G13, N1, N7, N13
PRA OR I/O	121	A6
PRB or I/O	47	L6
SDI or I/O	2	C2
V_{CC}	9, 16, 23, 39, 49, 51, 65, 78, 84, 94, 117, 119, 127	B2, B7, B12, G2, G12, M2, M7, M12
V_{KS}	92	F10
V_{PP}	91	E11
V_{SV}	24, 77	J2, J12

Notes:

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

Package Pin Assignments (continued)

175-Pin CPGA (Bottom View)

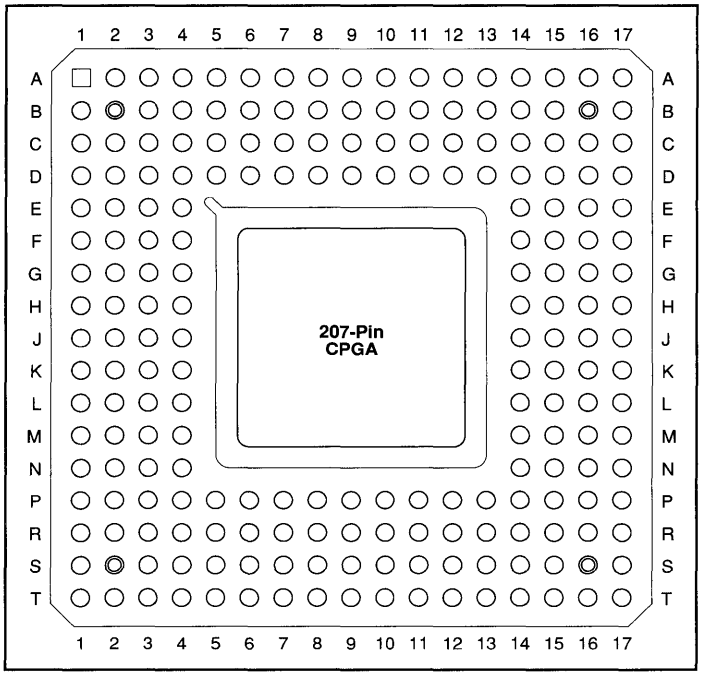


Signal	Pad Number	Location
CLKA or I/O	159	C9
CLKB or I/O	160	A9
DCLK or I/O	185	D5
GND	1, 12, 22, 24, 33, 46, 47, 57, 67, 69, 80, 90, 93, 104, 113, 115, 128, 138, 139, 151, 162, 164, 175, 183	D4, D8, D11, D12, E4, H4, H12, L4, L12, M4, M8, M12
HCLK or I/O	72	R8
IOCLK or I/O	137	E12
IOPCL or I/O	92	P13
MODE	10	F3
NC	—	A2, A15, B2, B3, P2, P14, R1, R2, R14, R15
PRA OR I/O	165	B8
PRB or I/O	66	R7
SDI or I/O	2	D3
V_{CC}	11, 21, 23, 34, 53, 68, 70, 86, 103, 114, 116, 129, 144, 161, 163, 178	C3, C8, C13, H3, H13, N3, N8, N13
V_{KS}	127	E14
V_{PP}	126	E15
V_{SV}	35, 102	L1, L14

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

Package Pin Assignments (continued)

207-Pin CPGA (Top View)



Signal	Pad Number	Location
CLKA or I/O	185	K1
CLKB or I/O	186	J3
DCLK or I/O	213	E4
GND	1, 13, 27, 29, 41, 55, 56, 68, 80, 82, 94, 106, 109, 120, 133, 135, 151, 161, 162, 175, 188, 190, 200, 210	C15, D4, D5, D9, D14, J4, J14, P3, P4, P9, P14, R1
HCKL or I/O	85	J15
IOCLK or I/O	160	P5
IOPCL or I/O	108	N14
MODE	11	D7
NC	—	A1, A2, A16, A17, B1, B17, C1, C2, S1, S3, S17, T1, T2, T16, T17
PRA OR I/O	191	H1
PRB or I/O	79	K16
SDI or I/O	2	C3
V_{CC}	12, 26, 28, 42, 63, 81, 83, 102, 119, 134, 136, 152, 168, 187, 189, 206	B2, B9, B16, J2, J16, S2, S9, S16
V_{KS}	150	P7
V_{PP}	149	T5
V_{SV}	43, 118	D11, P12

Notes:

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

ACT™ 1 and ACT 2 Military Field Programmable Gate Arrays

ACT 1 Features

- Up to 2000 Gate Array Gates (6000 PLD equivalent gates)
- Replaces up to 53 TTL Packages
- Replaces up to seventeen 20-Pin PAL® Packages
- Design Library with over 250 Macro Functions
- Single Logic Module Architecture
- Up to 547 Logic Modules
- Up to 273 Flip-Flops
- Two In-Circuit Diagnostic Probe Pins Support Speed Analysis to 25 MHz
- Built-In High Speed Clock Distribution Network
- I/O Drive to 6 mA
- Nonvolatile, User Programmable
- Logic Fully Tested Prior to Shipment

ACT 2 Features

- Up to 8000 Gate Array Gates (20,000 PLD equivalent gates)
- Replaces up to 200 TTL Packages
- Replaces up to eighty 20-Pin PAL Packages
- Design Library with over 500 Macro Functions
- Single-Module Sequential Functions
- Wide-Input Combinatorial Functions
- Up to 1232 Programmable Logic Modules
- Up to 998 Flip-Flops
- 16-Bit Counter Performance at 50 MHz (MIL Temp)
- 16-Bit Accumulator Performance to 25 MHz (MIL Temp)
- Two In-Circuit Diagnostic Probe Pins Support Speed Analysis to 50 MHz
- Two High-Speed, Low-Skew Clock Networks
- I/O Drive to 6 mA
- Nonvolatile, User Programmable
- Logic Fully Tested Prior to Shipment

Product Family Profile

Family	ACT 2		ACT 1	
Device	**A1280A**	**A1240A**	**A1020B**	**A1010B**
Capacity				
Gate Array Equivalent Gates	8,000	4,000	2,000	1,200
PLD Equivalent Gates	20,000	10,000	6,000	3,000
TTL Equivalent Packages	210	105	53	34
20-Pin PAL Equivalent Packages	69	34	17	12
Logic Modules	1,232	684	547	295
S-Modules	624	348	0	0
C-Modules	608	336	547	295
Flip-Flops (maximum)	998	565	273	147
Routing Resources				
Horizontal Tracks/Channel	36	36	22	22
Vertical Tracks/Channel	15	15	13	13
PLICE Antifuse Elements	750,000	400,000	190,000	110,000
User I/Os (maximum)	140	104	69	57
Packages[1]	176 CPGA 172 CQFP	132 CPGA	84 CPGA 84 CQFP	84 CPGA
CMOS Process	1.0 µm	1.0 µm	1.0 µm	1.0 µm

Note:
1. See product plan on page 1-142 for package availability.

High Reliability, Low Risk Solution

Actel builds the most reliable field programmable gate arrays (FPGAs) in the industry, with overall antifuse reliability ratings of less than 10 Failures-In-Time (FITs), corresponding to a useful life of more than 40 years. Actel FGPAs have been production-proven, with more than one million devices shipped and more than 130 billion antifuses manufactured. Actel devices are fully tested prior to shipment, with an outgoing defect level of only 122 ppm. (Further reliability data is available in the "Actel Reliability Report.")

100% Tested

Device functionality is fully tested before shipment and during device programming. Routing tracks, logic modules, and programming, debug, and test circuits are 100% tested before shipment. Antifuse integrity also is tested before shipment. Programming algorithms are tested when a device is programmed using Actel's Activator® 2 or Activator 2S programming stations.

Benefits

No Cost Risk—Once you have a Designer/Designer Advantage™ System, Actel's CAE software and programming package, you can produce as many chips as you like for just the cost of the device itself, with no NRE charges to eat up your development budget each time you want to try out a new design.

No Time Risk—After entering your design, placement and routing is automatic, and programming the device takes only about 5 to 15 minutes for an average design. You save time in the design entry process by using tools that are familiar to you. The Action Logic System software interfaces with popular CAE Cadence software such as Mentor Graphics®, OrCAD™, and Viewlogic® and runs on popular platforms such as HP™, Sun™, and 386/486™ PC compatible machines.

No Reliability Risk—The PLICE® antifuse is a one-time programmable, nonvolatile connection. Since Actel devices are permanently programmed, no downloading from EPROM or SRAM storage is required. Inadvertent erasure is impossible and there is no need to reload the program after power disruptions. Both the PLICE antifuses and the base process are radiation tolerant. Fabrication using a low-power CMOS process means cooler junction temperatures. Actel's non-PLD architecture delivers lower dynamic operating current. Our reliability tests show a very low failure rate of 66 FITs at 90°C junction temperature with no degradation in AC performance. Special stress testing at wafer test eliminates infant mortalities prior to packaging.

No Security Risk—Reverse engineering of programmed Actel devices from optical or electrical data is extremely difficult. Programmed antifuses cannot be identified from a photograph or by using a SEM. The antifuse map cannot be deciphered either electrically or by microprobing. Each device has a silicon signature that identifies its origins, down to the wafer lot and fabrication facility.

No Testing Risk—Unprogrammed Actel parts are fully tested at the factory. This includes the logic modules, interconnect tracks, and I/Os. AC performance is ensured by special speed path tests, and programming circuitry is verified on test antifuses. During the programming process, an algorithm is run to ensure that all antifuses are correctly programmed. In addition, Actel's Actionprobe® diagnostic tools allow 100% observability of all internal nodes to check and debug your design.

ACT 1 Description

The ACT™ 1 family of FPGAs offers a variety of package, speed, and application combinations. Devices are implemented in silicon gate, 1.2-micron two-level metal CMOS, and they employ Actel's PLICE antifuse technology. The unique architecture offers gate array flexibility, high performance, and instant turnaround through user programming. Device utilization is typically 95 percent of available logic modules.

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

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

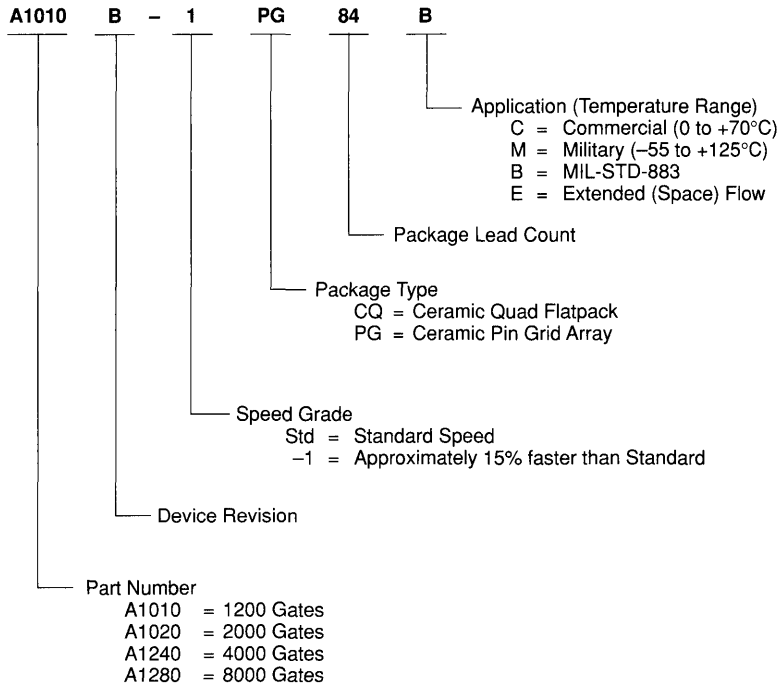
A security fuse may be programmed to disable all further programming and to protect the design from being copied or reverse engineered.

ACT 2 Description

The ACT 2 family represents Actel's second generation of FPGAs. The ACT 2 family presents a two-module architecture consisting of C-modules and S-modules. These modules are optimized for both combinatorial and sequential designs. Based on Actel's patented channeled array architecture, the ACT 2 family provides significant enhancements to gate density and performance while maintaining downward compatibility with the ACT 1 design environment. The devices are implemented in silicon gate, 1.0-µm, two-level metal CMOS, and employ Actel's PLICE antifuse technology. This revolutionary architecture offers gate array design flexibility, high performance, and fast time-to-production with user programming.

The ACT 2 family is supported by the ALS, which offers automatic pin assignment, validation of electrical and design rules, automatic placement and routing, timing analysis, user programming, and debug and diagnostic probe capabilities. The Action Logic System is supported on the following platforms: 386/486 PC, Sun, and HP workstations. It provides CAE interfaces to the following design environments: Cadence, Viewlogic, Mentor Graphics, and OrCAD.

Military Device Ordering Information



SMD Drawing Number at Actel Part Number Cross Reference

SMD Number	Cage Number	Actel Part Number
5962-9096401MZC	0J4Z0	A1010A-PG84B
5962-9096501MUC	0J4Z0	A1020B-PG84B
5962-9096502MUC	0J4Z0	A1020B-IPG84B
5962-9096501MTC	0J4Z0	A1020B-CQ84B
5962-9096502MTC	0J4Z0	A1020B-ICQ84B
5962-9215601MXC	0J4Z0	A1280A-PG176B
5962-9215601MYC	0J4Z0	A1280A-CQ172B
5962-9215602MXC	0J4Z0	A1280A-1PG176B
5962-9215602MYC	0J4Z0	A1280A-1CQ172B

Product Plan

	Speed Grade*		Application			
	Std	–1	C	M	B	E
A1280A Device						
176-pin Ceramic Pin Grid Array (PG)	✔	P	✔	✔	✔	—
172-pin Ceramic Quad Flatpack (PQ)	✔	P	✔	✔	✔	✔
A1240A Device						
132-pin Ceramic Pin Grid Array (PG)	✔	P	✔	✔	✔	—
A1020B Device						
84-pin Ceramic Pin Grid Array (PG)	✔	✔	✔	✔	✔	—
84-pin Ceramic Quad Flatpack (CQ)	✔	✔	✔	✔	✔	✔
A1010B Device						
84-pin Ceramic Pin Grid Array (PG)	✔	✔	✔	✔	✔	—

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

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

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

Device Resources

			User I/Os				
			CPGA			CQFP	
Device Series	Logic Modules	Gates	176-pin	132-pin	84-pin	172-pin	84-pin
A1280A	1232	8000	140	—	—	140	—
A1240A	684	4000	—	104	—	—	—
A1020B	547	2000	—	—	69	—	69
A1020B	295	1200	—	—	57	—	—

Pin Description

CLKA **Clock A (Input)**

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

CLKB **Clock B (Input)**

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

DCLK **Diagnostic Clock (Input)**

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

GND **Ground**

LOW supply voltage.

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

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

MODE **Mode (Input)**

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

NC **No Connection**

This pin is not connected to circuitry within the device.

PRA **Probe A (Output)**

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

PRB **Probe B (Output)**

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

SDI **Serial Data Input (Input)**

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

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

HIGH supply voltage.

V_{KS} **Programming Voltage**

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

V_{PP} **Programming Voltage**

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

V_{SV} **Programming Voltage**

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

Actel Military Product Flow

Step	Screen	833C—Class B 833C Method	833C—Class B Requirement	Military Datasheet Requirement
1.0	Internal Visual	2010, Test Condition B	100%	100%
2.0	Temperature Cycling	1010, Test Condition C	100%	100%
3.0	Constant Acceleration	2001, Test Condition E (min), Y1, Orientation Only	100%	100%
4.0	Seal	1014		
	a. Fine		100%	100%
	b. Gross		100%	100%
5.0	Visual Inspection		100%	100%
6.0	Pre Burn-in Electrical Parameters	In accordance with Actel applicable device specification	100%	N/A
7.0	Burn-in Test	1015 Condition D 160 hours @ 125°C Min.	100%	N/A
8.0	Interim (post burn-in) Electrical Parameters	In accordance with Actel applicable device specification	100%	100% (as final test)
9.0	Percent Defective Allowable	5%	All Lots	N/A
10.0	Final Electrical Test	In accordance with Actel applicable device specification		
	a. Static Tests (1) 25°C (Subgroup 1, Table I, 5005) (2) –55°C and +125°C (Subgroups 2, 3, Table I, 5005)		100%	100%
	b. Dynamic and Functional Tests (1) 25°C (Subgroup 7, Table I, 5005) (2) –55°C and +125°C (Subgroups 8A and 8B, Table I, 5005)		100%	100%
	c. Switching Tests at 25°C (Subgroup 9, Table I, 5005)		100%	100%
11.0	Qualification or Quality Confirmation Inspection Test Sample Selection (Group A)	5005	All Lots	N/A
12.0	External Visual	2009	100%	Actel specification

Actel Extended Flow[1]

	Screen	Method	Requirement
1.	Wafer Lot Acceptance	5007 with step coverage waiver	All Lots
2.	Destructive In-Line Bond Pull[2]	2011, condition D	Sample
3.	Internal Visual	2010, condition A	100%
4.	Temperature Cycling	1010, condition C	100%
5.	Constant Acceleration	2001, condition E (min), Y_1 orientation only	100%
6.	Visual Inspection	2009	100%
7.	Particle Impact Noise Detection	2020, condition A	100%
8.	Serialization		100%
9.	Pre Burn-in Test	In accordance with Actel applicable device specification	100%
10.	Burn-in Test	1015, 240 hours @ 125°C minimum	100%
11.	Interim (Post Burn-in) Electrical Parameters	In accordance with Actel applicable device specification	100%
12.	Reverse Bias Burn-in	1010, condition A or C, 72 hours @ 150°C minimum	100%
13.	Interim (Post Burn-in) Electrical Parameters	In accordance with Actel applicable device specification	100%
14.	Percent Defective Allowable (PDA) Calculation	5%, 3% functional parameters @ 25°C	All Lots
15.	Final Electrical Test	In accordance with Actel applicable device specification	100%
	a. Static Tests		100%
	(1) 25°C (Subgroup 1, Table1)	5005	
	(2) –55°C and +125°C (Subgroups 2, 3, Table 1)	5005	
	b. Dynamic and Functional Tests		100%
	(1) 25°C (Subgroup 7, Table 15)	5005	
	(2) –55°C and +125°C (Subgroups 5 and 6, 8a and b, Table 1)	5005	
	c. Switching Tests at 25°C (Subgroup 9, Table I, 5005)	5005	100%
16.	Seal a. Fine b. Gross	1014	100%
17.	Radiographic	2012	100%
18.	Qualification or Quality Conformance Inspection Test Sample Selection	5005	Per Group A
19	External Visual	2009	100%

Notes:

1. Actel offers the Extended Flow in order to satisfy those customers that require additional screening beyond the requirements of MIL-STD-883C, Class B. Actel is compliant to the requirements of MIL-STD-883C, Paragraph 1.2.1, and MIL-M-38510 Appendix A. Actel is offering this extended flow incorporating the majority of the screening procedures as outlined in Method 5004 of MIL-STD-883C Class S. The exceptions to Method 5004 are shown in Notes 2 to 4 below.
2. Method 5004 requires a 100%, Non-Destructive Bond Pull to Method 2023. Actel substitutes a Non-Destructive Bond Pull to Method 2011, condition D on a sample basis only.

Absolute Maximum Ratings[1]

Free air temperature range

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

Notes:

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

Recommended Operating Conditions

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

Note:

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

Package Thermal Characteristics

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

Maximum junction temperature is 150°C.

A sample calculation of the absolute maximum power dissipation allowed for a CPGA 176-pin package at military temperature is as follows:

$$\frac{\text{Max. junction temp. (°C)} - \text{Max. military temp.}}{\theta ja\ (°C/W)} = \frac{150°C - 125°C}{23°C/W} = 1.1\ W$$

Package Type	Pin Count	θjc	θja Still Air	θja 300 ft/min	Units
Ceramic Pin Grid Array	84	8	33	20	°C/W
	132	5	30	15	°C/W
	176	8	23	12	°C/W
Ceramic Quad Flatpack	84	5	40	30	°C/W
	172	8	25	15	°C/W

ACT 1 Electrical Specifications

Symbol	Parameter	Commercial		Military		Units
		Min.	Max.	Min.	Max.	
V_{OH}[1]	(I_{OH} = –6 mA)	3.84				V
	(I_{OH} = –4 mA)			3.7		V
V_{OL}[1]	(I_{OL} = 6 mA)		0.33		0.40	V
V_{IL}		–0.3	0.8	–0.3	0.8	V
V_{IH}		2.0	V_{CC} + 0.3	2.0	V_{CC} + 0.3	V
Input Transition Time t_R, t_F[2]			500		500	ns
C_{IO} I/O Capacitance[2, 3]			10		10	pF
Standby Current, I_{CC}[4]			3		20	mA
Leakage Current[5]		–10	10	–10	10	µA
I_{OS} Output Short Circuit Current[6]	(V_O = V_{CC})	20	140	20	140	mA
	(V_O = GND)	–10	–100	–10	–100	mA

Notes:
1. Only one output tested at a time. V_{CC} = min.
2. Not tested, for information only.
3. Includes worst-case 84-pin PLCC package capacitance. V_{OUT} = 0 V, f = 1 MHz.
4. Typical standby current = 3 mA. All outputs unloaded. All inputs = V_{CC} or GND.
5. V_O, V_{IN} = V_{CC} or GND.
6. Only one output tested at a time. Min. at V_{CC} = 4.5 V; Max. at V_{CC} = 5.5 V.

ACT 2 Electrical Specifications

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

Notes:
1. Only one output tested at a time. V_{CC} = min.
2. Not tested, for information only.
3. Includes worst-case 176 CPGA package capacitance. V_{OUT} = 0 V, f = 1 MHz.
4. All outputs unloaded. All inputs = V_{CC} or GND.
5. V_O , V_{IN} = V_{CC} or GND.

ACT 1 Power Dissipation

The following formula is used to calculate total device dissipation.

Total Device Power (mW) = (0.20 x N x F1) + (0.085 x M x F2) + (0.80 x P x F3)

Where:

F1 = Average logic module switching rate in MHz

F2 = CLKBUF macro switching rate in MHz

F3 = Average I/O module switching rate in MHz

M = Number of logic modules connected to the CLKBUF macro

N = Total number of logic modules used in the design (including M)

P = Number of outputs loaded with 50 pF

Average switching rate of logic modules and of I/O modules is some fraction of the device operating frequency (usually CLKBUF). Logic modules and I/O modules switch states (from low-to-high or from high-to-low) only if the input data changes when the module is enabled. A conservative estimate for average logic module and I/O module switching rates (variables F1 and F3, respectively) is 10% of device clock driver frequency.

If the CLKBUF macro is not used in the design, eliminate the second term (including F2 and M variables) from the formula.

Sample A1020 Device Power Calculation

To illustrate the power calculation, consider a large design operating at high frequency. This sample design utilizes 85% of available logic modules on the A1020-series device (.85 x 547 = 465 logic modules used). The design contains 104 flip-flops (208 logic modules). Operating frequency of the design is 16 MHz. In this design, the CLKBUF macro drives the clock network. Logic modules and I/O modules are switching states at approximately 10% of the clock frequency rate (.10 x 16 MHz = 1.6 MHz). Sixteen outputs are loaded with 50 pF.

To summarize the design described above: N = 465; M = 208; F2 = 16; F1 = 4; F3 = 4; P = 16. Total device power can be calculated by substituting these values for variables in the device dissipation formula.

Total device power for this example =

(0.20 x 465 x 1.6) + (0.085 x 208 x 16) + (0.80 x 16 x 1.6) = 452 mW

ACT 2 Power Dissipation

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

Where:

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

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

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

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

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

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

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

Static Power

Static power dissipation is typically a small component of the overall power. From the values provided in the Electrical Specifications, the maximum static power (commercial) dissipation is:

2 mA * 5.25 V = 10.5 mW

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

Active Time

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

Equivalent Capacitance

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

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

Where:

C_{EQ} is the equivalent capacitance expressed in picofarads(pF).
V_{CC} is power supply in volts (V).
f is the switching frequency in megahertz (MHz).

Equivalent capacitance is calculated by measuring I_{active} at a specified frequency and voltage for each circuit component of interest. The results for ACT 2 devices are:

	C_{EQ} (pF)
Modules	7.7
Input Buffers	18.0
Output Buffers	25.0
Clock Buffer Loads	2.5

To calculate the active power dissipated from the complete design, you must solve Equation 1 for each component. To do this, you must know the switching frequency of each part of the logic. The exact equation is a piece-wise linear summation over all components, as shown in Equation 2.

$$\text{Power } (\mu W) = [(m * 7.7 \times f_1) + (n * 18.0 \times f_2) + (p * (25.0 + C_L) * f_3) + (q \times 2.5 \times f_4)] * V_{CC}^2 \quad (2)$$

Where:

m = Number of logic modules switching at frequency f_1

n = Number of input buffers switching at frequency f_2

p = Number of output buffers switching at frequency f_3

q = Number of clock loads on the global clock network

f_1 = Average logic module switching rate in MHz

f_2 = Average input buffer switching rate in MHz

f_3 = Average output buffer switching rate in MHz

f_4 = Frequency of global clock

C_L = Output load capacitance in pF

Determining Average Switching Frequency

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

Module Utilization = 80% of combinatorial modules

Average Module Frequency = F/10

Inputs = 1/3 of I/O

Average Input Frequency = F/5

Outputs = 2/3 of I/Os

Average Output Frequency = F/10

Clock Net 1 Loading = 40% of sequential modules

Clock Net 1 Frequency = F

Clock Net 2 Loading = 40% of sequential modules

Clock Net 2 Frequency = F/2

Estimated Power

The results of estimating active power are displayed in Figure 1. The graphs provide a simple guideline for estimating power. The tables may be interpolated when your application has different resource utilizations or frequencies.

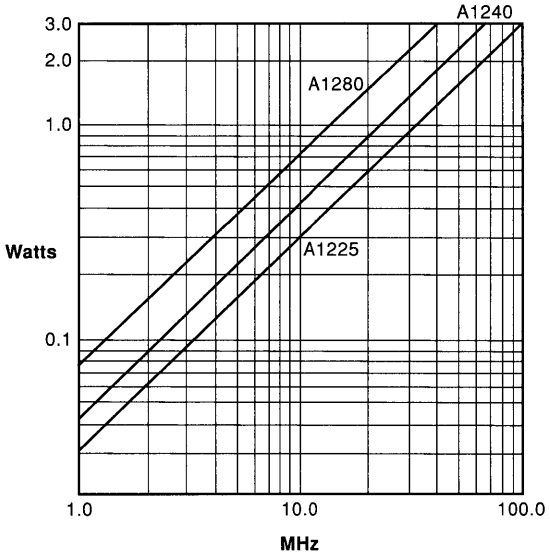
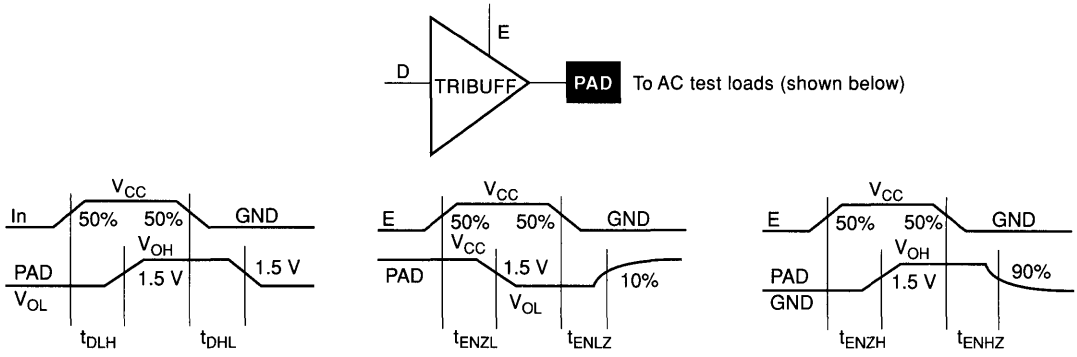


Figure 1. ACT 2 Power Estimates

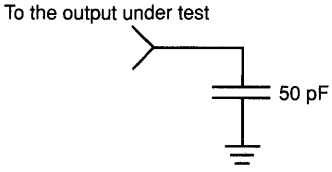
Parameter Measurement

Output Buffer Delays

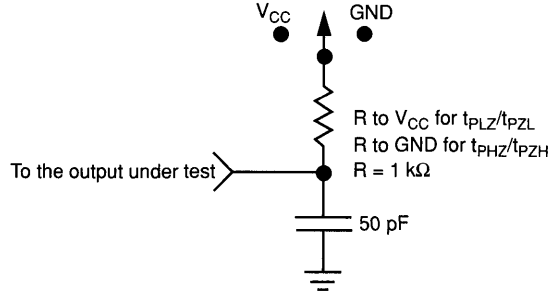


AC Test Load

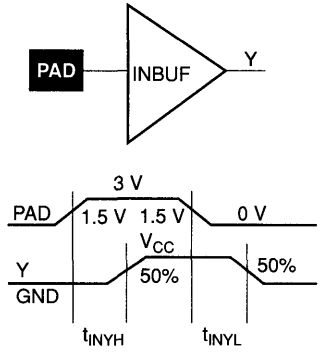
Load 1
(Used to measure propagation delay)



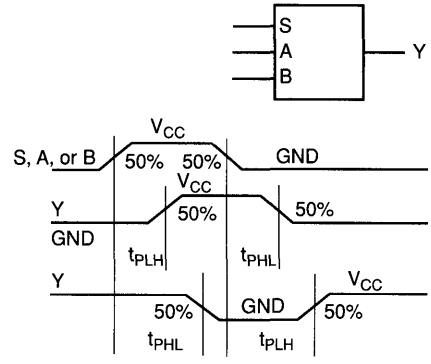
Load 2
(Used to measure rising/falling edges)



Input Buffer Delays

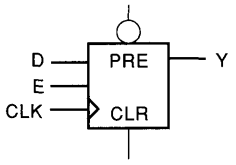


Combinatorial Macro Delays

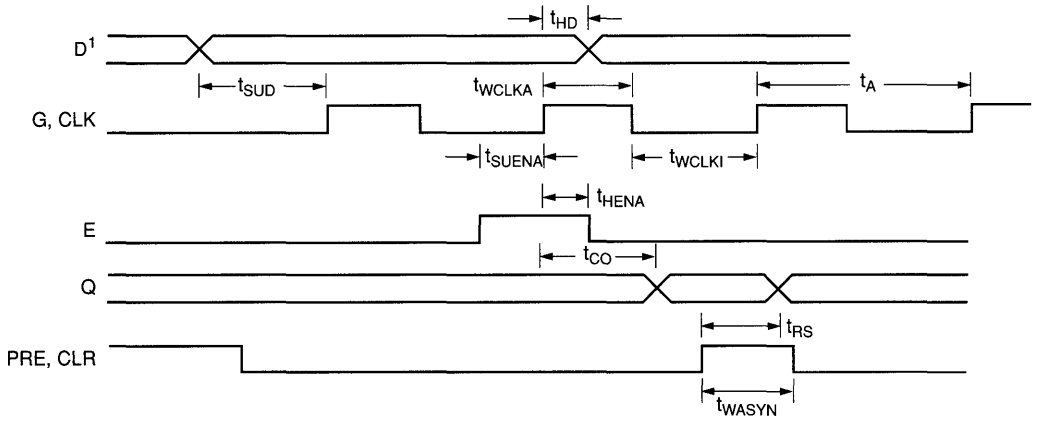


Sequential Timing Characteristics

Flip-Flops and Latches



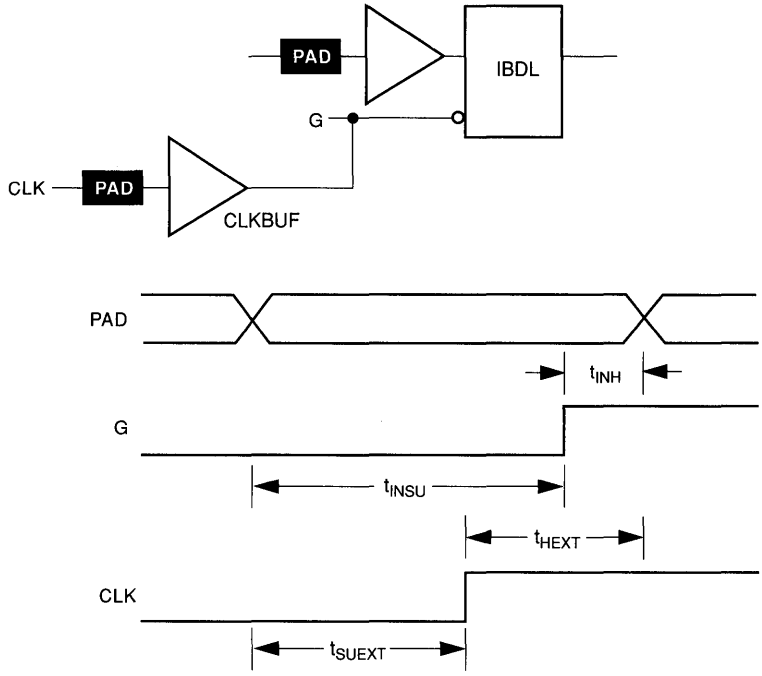
(Positive edge triggered)



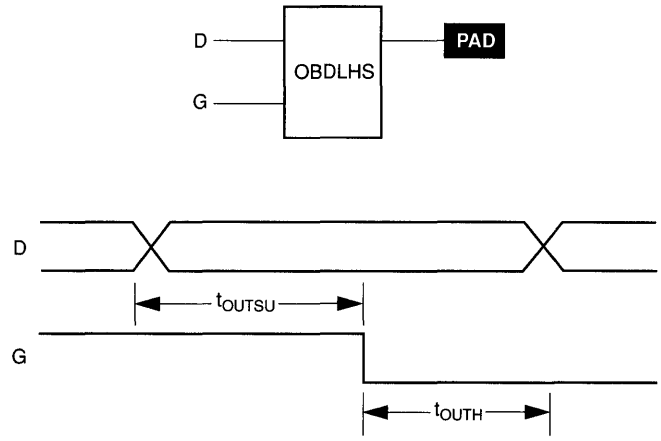
Note:
1. D represents all data functions involving A, B, and S for multiplexed flip-flops.

Sequential Timing Characteristics (continued)

Input Buffer Latches (ACT 2 only)



Output Buffer Latches (ACT 2 only)



ACT 1 Timing Characteristics

(Worst-Case Military Conditions)

Logic Module Propagation Delays		'Std' Speed		'–1' Speed		
Parameter	**Description**	**Min.**	**Max.**	**Min.**	**Max.**	**Units**
t_{PD1}	Single Module		5.5		4.7	ns
t_{PD2}	Dual Module Macros		12.7		10.8	ns
t_{CO}	Sequential Clk to Q		5.5		4.7	ns
t_{GO}	Latch G to Q		5.5		4.7	ns
t_{RS}	Flip-Flop (Latch) Reset to Q		5.5		4.7	ns
Predicted Routing Delays[1]						
t_{RD1}	FO=1 Routing Delay		1.7		1.5	ns
t_{RD2}	FO=2 Routing Delay		2.7		2.3	ns
t_{RD3}	FO=3 Routing Delay		4.0		3.4	ns
t_{RD4}	FO=4 Routing Delay		5.9		5.0	ns
t_{RD8}	FO=8 Routing Delay		12.5		10.6	ns
Sequential Timing Characteristics[2]						
t_{SUD}	Flip-Flop (Latch) Data Input Setup	10.4		8.8		ns
t_{HD}	Flip-Flop (Latch) Data Input Hold	0.0		0.0		ns
t_{SUENA}	Flip-Flop (Latch) Enable Setup	10.4		8.8		ns
t_{HENA}	Flip-Flop (Latch) Enable Hold	0.0		0.0		ns
t_{WCLKA}	Flip-Flop (Latch) Clock Active Pulse Width	12.9		10.9		ns
t_{WASYN}	Flip-Flop (Latch) Asynchronous Pulse Width	12.9		10.9		ns
t_A	Flip-Flop Clock Input Period	27.3		23.2		ns
f_{MAX}	Flip-Flop (Latch) Clock Frequency		37		44	MHz

Notes:

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

ACT 1 Timing Characteristics (continued)

(Worst-Case Military Conditions)

Input Module Propagation Delays			'Std' Speed		'–1' Speed		
Parameter	**Description**		**Min.**	**Max.**	**Min.**	**Max.**	**Units**
t_{INYH}	Pad to Y High			5.8		4.9	ns
t_{INYL}	Pad to Y Low			5.8		4.9	ns
Input Module Predicted Routing Delays[1]							
t_{IRD1}	FO=1 Routing Delay			1.7		1.5	ns
t_{IRD2}	FO=2 Routing Delay			2.7		2.3	ns
t_{IRD3}	FO=3 Routing Delay			4.0		3.4	ns
t_{IRD4}	FO=4 Routing Delay			5.9		5.0	ns
t_{IRD8}	FO=8 Routing Delay			12.5		10.6	ns
Global Clock Network							
t_{CKH}	Input Low to High	FO = 16 FO = 128		9.2 10.5		7.8 8.9	ns
t_{CKL}	Input High to Low	FO = 16 FO = 128		12.1 13.2		10.3 11.2	ns
t_{PWH}	Minimum Pulse Width High	FO = 16 FO = 128	12.2 12.9		10.4 10.9		ns
t_{PWL}	Minimum Pulse Width Low	FO = 16 FO = 128	12.2 12.9		10.4 10.9		ns
t_{CKSW}	Maximum Skew	FO = 16 FO = 128		2.2 3.4		1.9 2.9	ns
t_P	Minimum Period	FO = 16 FO = 128	25.6 27.3		21.7 23.2		ns
f_{MAX}	Maximum Frequency	FO = 16 FO = 128		40 37		46 44	MHz

Note:

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

ACT 1 Timing Characteristics (continued)

(Worst-Case Military Conditions)

Output Module Timing		'Std' Speed		'–1' Speed		
Parameter	**Description**	**Min.**	**Max.**	**Min.**	**Max.**	**Units**
TTL Output Module Timing[1]						
t_{DLH}	Data to Pad High		14.2		12.1	ns
t_{DHL}	Data to Pad Low		16.3		13.8	ns
t_{ENZH}	Enable Pad Z to High		14.1		12.0	ns
t_{ENZL}	Enable Pad Z to Low		17.1		14.6	ns
t_{ENHZ}	Enable Pad High to Z		18.8		16.0	ns
t_{ENLZ}	Enable Pad Low to Z		17.0		14.5	ns
d_{TLH}	Delta Low to High		0.11		0.09	ns/pF
d_{THL}	Delta High to Low		0.15		0.12	ns/pF
CMOS Output Module Timing[1]						
t_{DLH}	Data to Pad High		17.7		15.1	ns
t_{DHL}	Data to Pad Low		13.6		11.5	ns
t_{ENZH}	Enable Pad Z to High		14.1		12.0	ns
t_{ENZL}	Enable Pad Z to Low		17.1		14.6	ns
t_{ENHZ}	Enable Pad High to Z		18.8		16.0	ns
t_{ENLZ}	Enable Pad Low to Z		17.0		14.5	ns
d_{TLH}	Delta Low to High		0.18		0.16	ns/pF
d_{THL}	Delta High to Low		0.11		0.09	ns/pF

Note:
1. Delays based on 50 pF loading.

A1240A Timing Characteristics

(Worst-Case Military Conditions)

Logic Module Propagation Delays[1]		'Std' Speed		'–1' Speed		
Parameter	**Description**	**Min.**	**Max.**	**Min.**	**Max.**	**Units**
t_{PD1}	Single Module		6.1		5.2	ns
t_{CO}	Sequential Clk to Q		6.1		5.2	ns
t_{GO}	Latch G to Q		6.1		5.2	ns
t_{RS}	Flip-Flop (Latch) Reset to Q		6.1		5.2	ns
Predicted Routing Delays[2]						
t_{RD1}	FO=1 Routing Delay		2.2		1.9	ns
t_{RD2}	FO=2 Routing Delay		2.8		2.4	ns
t_{RD3}	FO=3 Routing Delay		3.7		3.1	ns
t_{RD4}	FO=4 Routing Delay		5.0		4.3	ns
t_{RD8}	FO=8 Routing Delay		7.7		6.6	ns
Sequential Timing Characteristics[3, 4]						
t_{SUD}	Flip-Flop (Latch) Data Input Setup	1.0		1.0		ns
t_{SUASYN}	Flip-Flop (Latch) Asynchronous Input Setup	2.0		2.0		ns
t_{HD}	Flip-Flop (Latch) Data Input Hold	0.0		0.0		ns
t_{SUENA}	Flip-Flop (Latch) Enable Setup	7.5		7.5		ns
t_{HENA}	Flip-Flop (Latch) Enable Hold	0.0		0.0		ns
t_{WCLKA}	Flip-Flop (Latch) Clock Active Pulse Width	8.4		7.1		ns
t_{WASYN}	Flip-Flop (Latch) Asynchronous Pulse Width	8.4		7.1		ns
t_A	Flip-Flop Clock Input Period	18.6		15.8		ns
t_{INH}	Input Buffer Latch Hold	2.5		2.5		ns
t_{INSU}	Input Buffer Latch Setup	–3.5		–3.5		ns
t_{OUTH}	Output Buffer Latch Hold	0.0		0.0		ns
t_{OUTSU}	Output Buffer Latch Setup	1.0		1.0		ns
f_{MAX}	Flip-Flop (Latch) Clock Frequency		54		63	MHz

Notes:

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

A1240A Timing Characteristics (continued)

(Worst-Case Military Conditions)

Input Module Propagation Delays			'Std' Speed		'–1' Speed		
Parameter	**Description**		**Min.**	**Max.**	**Min.**	**Max.**	**Units**
t_{INYH}	Pad to Y High			4.7		4.0	ns
t_{INYL}	Pad to Y Low			4.3		3.6	ns
t_{INGH}	G to Y High			8.1		6.9	ns
t_{INGL}	G to Y Low			7.7		6.6	ns
Input Module Predicted Routing Delays[1]							
t_{IRD1}	FO=1 Routing Delay			6.9		5.8	ns
t_{IRD2}	FO=2 Routing Delay			7.8		6.7	ns
t_{IRD3}	FO=3 Routing Delay			8.8		7.5	ns
t_{IRD4}	FO=4 Routing Delay			9.7		8.2	ns
t_{IRD8}	FO=8 Routing Delay			12.9		10.9	ns
Global Clock Network							
t_{CKH}	Input Low to High	FO = 32 FO = 256		15.7 19.2		13.3 16.3	ns
t_{CKL}	Input High to Low	FO = 32 FO = 256		15.7 19.5		13.3 16.5	ns
t_{PWH}	Minimum Pulse Width High	FO = 32 FO = 256	6.7 7.1		5.7 6.0		ns
t_{PWL}	Minimum Pulse Width Low	FO = 32 FO = 256	6.7 7.1		5.7 6.0		ns
t_{CKSW}	Maximum Skew	FO = 32 FO = 256		0.5 2.5		0.5 2.5	ns
t_{SUEXT}	Input Latch External Setup	FO = 32 FO = 256	0.0 0.0		0.0 0.0		ns
t_{HEXT}	Input Latch External Hold	FO = 32 FO = 256	7.0 11.2		7.0 11.2		ns
t_P	Minimum Period	FO = 32 FO = 256	13.5 14.3		11.5 12.2		ns
f_{MAX}	Maximum Frequency	FO = 32 FO = 256		74 70		87 82	MHz

Note:

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

A1240A Timing Characteristics (continued)

(Worst-Case Military Conditions)

Output Module Timing		'Std' Speed		'–1' Speed		
Parameter	**Description**	**Min.**	**Max.**	**Min.**	**Max.**	**Units**
TTL Output Module Timing[1]						
t_{DLH}	Data to Pad High		13.0		11.0	ns
t_{DHL}	Data to Pad Low		16.4		13.9	ns
t_{ENZH}	Enable Pad Z to High		14.4		12.3	ns
t_{ENZL}	Enable Pad Z to Low		19.0		16.1	ns
t_{ENHZ}	Enable Pad High to Z		11.5		9.8	ns
t_{ENLZ}	Enable Pad Low to Z		13.6		11.5	ns
t_{GLH}	G to Pad High		14.6		12.4	ns
t_{GHL}	G to Pad Low		18.2		15.5	ns
d_{TLH}	Delta Low to High		0.11		0.09	ns/pF
d_{THL}	Delta High to Low		0.20		0.17	ns/pF
CMOS Output Module Timing[1]						
t_{DLH}	Data to Pad High		16.5		14.0	ns
t_{DHL}	Data to Pad Low		13.7		11.7	ns
t_{ENZH}	Enable Pad Z to High		14.4		12.3	ns
t_{ENZL}	Enable Pad Z to Low		19.0		16.1	ns
t_{ENHZ}	Enable Pad High to Z		11.5		9.8	ns
t_{ENLZ}	Enable Pad Low to Z		13.6		11.5	ns
t_{GLH}	G to Pad High		14.6		12.4	ns
t_{GHL}	G to Pad Low		18.2		15.5	ns
d_{TLH}	Delta Low to High		0.20		0.17	ns/pF
d_{THL}	Delta High to Low		0.15		0.12	ns/pF

Note:
1. Delays based on 50 pF loading.

A1280A Timing Characteristics

(Worst-Case Military Conditions)

Logic Module Propagation Delays[1]		'Std' Speed		'–1' Speed		
Parameter	**Description**	**Min.**	**Max.**	**Min.**	**Max.**	**Units**
t_{PD1}	Single Module		6.1		5.2	ns
t_{CO}	Sequential Clk to Q		6.1		5.2	ns
t_{GO}	Latch G to Q		6.1		5.2	ns
t_{RS}	Flip-Flop (Latch) Reset to Q		6.1		5.2	ns
Predicted Routing Delays[2]						
t_{RD1}	FO=1 Routing Delay		2.8		2.4	ns
t_{RD2}	FO=2 Routing Delay		4.0		3.4	ns
t_{RD3}	FO=3 Routing Delay		4.9		4.2	ns
t_{RD4}	FO=4 Routing Delay		6.0		5.1	ns
t_{RD8}	FO=8 Routing Delay		10.8		9.2	ns
Sequential Timing Characteristics[3,4]						
t_{SUD}	Flip-Flop (Latch) Data Input Setup	1.0		1.0		ns
t_{SUASYN}	Flip-Flop (Latch) Asynchronous Input Setup	2.0		2.0		ns
t_{HD}	Flip-Flop (Latch) Data Input Hold	0.0		0.0		ns
t_{SUENA}	Flip-Flop (Latch) Enable Setup	7.5		7.5		ns
t_{HENA}	Flip-Flop (Latch) Enable Hold	0.0		0.0		ns
t_{WCLKA}	Flip-Flop (Latch) Clock Active Pulse Width	11.6		9.9		ns
t_{WASYN}	Flip-Flop (Latch) Asynchronous Pulse Width	11.6		9.9		ns
t_{A}	Flip-Flop Clock Input Period	24.5		20.8		ns
t_{INH}	Input Buffer Latch Hold	2.5		2.5		ns
t_{INSU}	Input Buffer Latch Setup	–3.5		–3.5		ns
t_{OUTH}	Output Buffer Latch Hold	0.0		0.0		ns
t_{OUTSU}	Output Buffer Latch Setup	1.0		1.0		ns
f_{MAX}	Flip-Flop (Latch) Clock Frequency		41		48	MHz

Notes:

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

A1280A Timing Characteristics (continued)

(Worst-Case Military Conditions)

Input Module Propagation Delays			'Std' Speed		'–1' Speed		
Parameter	**Description**		**Min.**	**Max.**	**Min.**	**Max.**	**Units**
t_{INYH}	Pad to Y High			4.7		4.0	ns
t_{INYL}	Pad to Y Low			4.3		3.6	ns
t_{INGH}	G to Y High			8.1		6.9	ns
t_{INGL}	G to Y Low			7.7		6.6	ns
Input Module Predicted Routing Delays[1]							
t_{RD1}	FO=1 Routing Delay			7.3		6.2	ns
t_{RD2}	FO=2 Routing Delay			8.4		7.2	ns
t_{RD3}	FO=3 Routing Delay			9.1		7.7	ns
t_{RD4}	FO=4 Routing Delay			10.5		8.9	ns
t_{RD8}	FO=8 Routing Delay			15.2		12.9	ns
Global Clock Network							
t_{CKH}	Input Low to High	FO = 32		15.7		13.3	ns
		FO = 384		21.1		17.9	
t_{CKL}	Input High to Low	FO = 32		15.7		13.3	ns
		FO = 384		21.4		18.2	
t_{PWH}	Minimum Pulse Width High	FO = 32	8.1		6.9		ns
		FO = 384	9.3		7.9		
t_{PWL}	Minimum Pulse Width Low	FO = 32	8.1		6.9		ns
		FO = 384	9.3		7.9		
t_{CKSW}	Maximum Skew	FO = 32		0.5		0.5	ns
		FO = 384		2.5		2.5	
t_{SUEXT}	Input Latch External Setup	FO = 32	0.0		0.0		ns
		FO = 384	0.0		0.0		
t_{HEXT}	Input Latch External Hold	FO = 32	7.0		7.0		ns
		FO = 384	11.2		11.2		
t_P	Minimum Period	FO = 32	16.2		13.7		ns
		FO = 384	18.9		16.0		
f_{MAX}	Maximum Frequency	FO = 32		62		73	MHz
		FO = 384		53		63	

Note:

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

A1280A Timing Characteristics (continued)

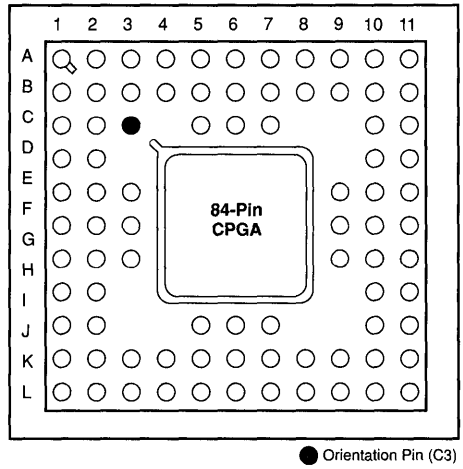
(Worst-Case Military Conditions)

Output Module Timing		'Std' Speed		'–1' Speed		
Parameter	**Description**	**Min.**	**Max.**	**Min.**	**Max.**	**Units**
TTL Output Module Timing[1]						
t_{DLH}	Data to Pad High		13.0		11.0	ns
t_{DHL}	Data to Pad Low		16.4		13.9	ns
t_{ENZH}	Enable Pad Z to High		14.4		12.3	ns
t_{ENZL}	Enable Pad Z to Low		19.0		16.1	ns
t_{ENHZ}	Enable Pad High to Z		11.5		9.8	ns
t_{ENLZ}	Enable Pad Low to Z		13.6		11.5	ns
t_{GLH}	G to Pad High		14.6		12.4	ns
t_{GHL}	G to Pad Low		18.2		15.5	ns
d_{TLH}	Delta Low to High		0.11		0.09	ns/pF
d_{THL}	Delta High to Low		0.20		0.17	ns/pF
CMOS Output Module Timing[1]						
t_{DLH}	Data to Pad High		16.5		14.0	ns
t_{DHL}	Data to Pad Low		13.7		11.7	ns
t_{ENZH}	Enable Pad Z to High		14.4		12.3	ns
t_{ENZL}	Enable Pad Z to Low		19.0		16.1	ns
t_{ENHZ}	Enable Pad High to Z		11.5		9.8	ns
t_{ENLZ}	Enable Pad Low to Z		13.6		11.5	ns
t_{GLH}	G to Pad High		14.6		12.4	ns
t_{GHL}	G to Pad Low		18.2		15.5	ns
d_{TLH}	Delta Low to High		0.20		0.17	ns/pF
d_{THL}	Delta High to Low		0.15		0.12	ns/pF

Note:
1. Delays based on 50 pF loading.

Package Pin Assignments

84-Pin CPGA (Top View)

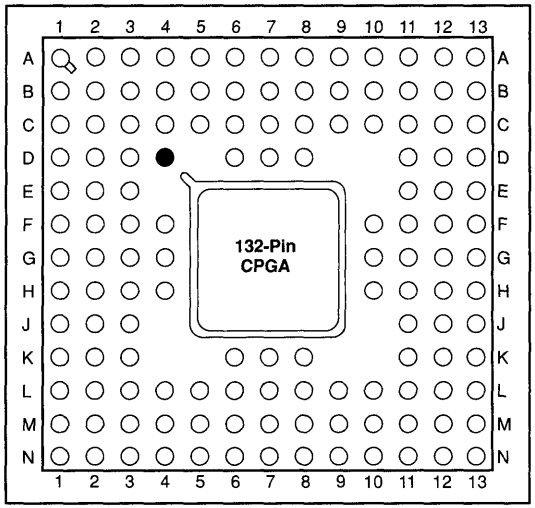


Signal	A1010B Devices	A1020B Devices
PRA	A11	A11
PRB	B10	B10
MODE	E11	E11
SDI	B11	B11
DCKL	C10	C10
V_{PP}	K2	K2
CLK or I/O	F9	F9
GND	B7, E2, E3, K5, F10, G10	B7, E2, E3, K5, F10,G10
V_{CC}	B5, F1, G2, K7, E9, E10	B5, F1, G2, K7, E9, E10
N/C (No Connection)	B1, B2, C1, C2, K1, J2, J10, K10, K11, C11,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

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

Package Pin Assignments

132-Pin CPGA (Top View)



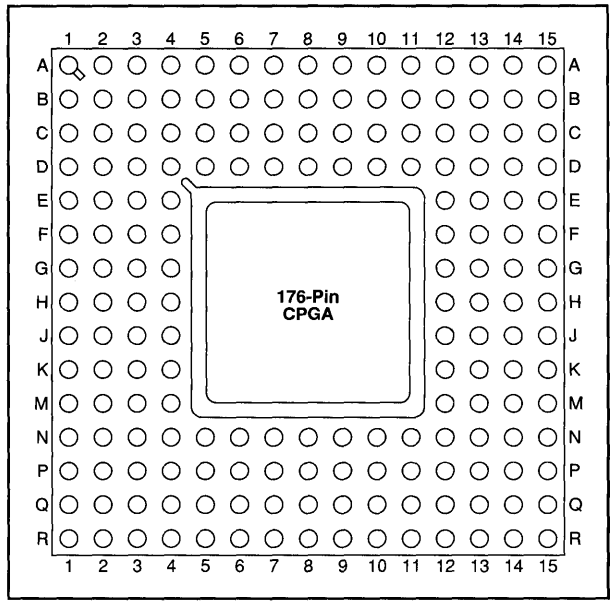
● Orientation Pin

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

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

Package Pin Assignments

176-Pin CPGA (Top View)

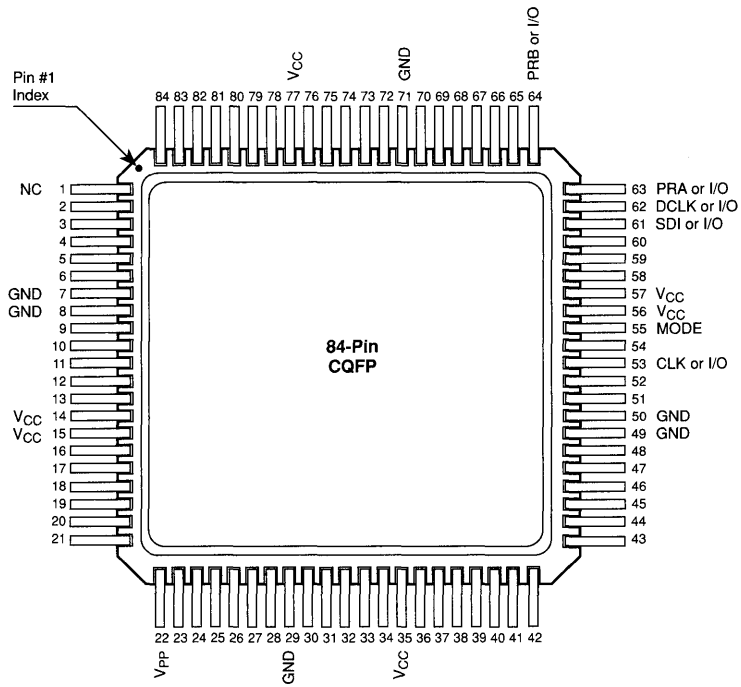


Signal	Pad Number	Location
PRA or I/O	152	C9
PRB or I/O	160	D7
MODE	2	C3
SDI or I/O	135	B14
DCLK or I/O	175	B3
CLKA or I/O	154	A9
CLKB or I/O	158	B8
GND	1, 8, 18, 23, 33, 38, 45, 57, 67, 77, 89 101, 106, 111, 121, 126, 133, 145, 156, 165	D4, E4, G4, H4, K4, L4, M4, M6, M8, M10, M12 K12, J12, H12, F12,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

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

Package Pin Assignments

84-Pin CQFP

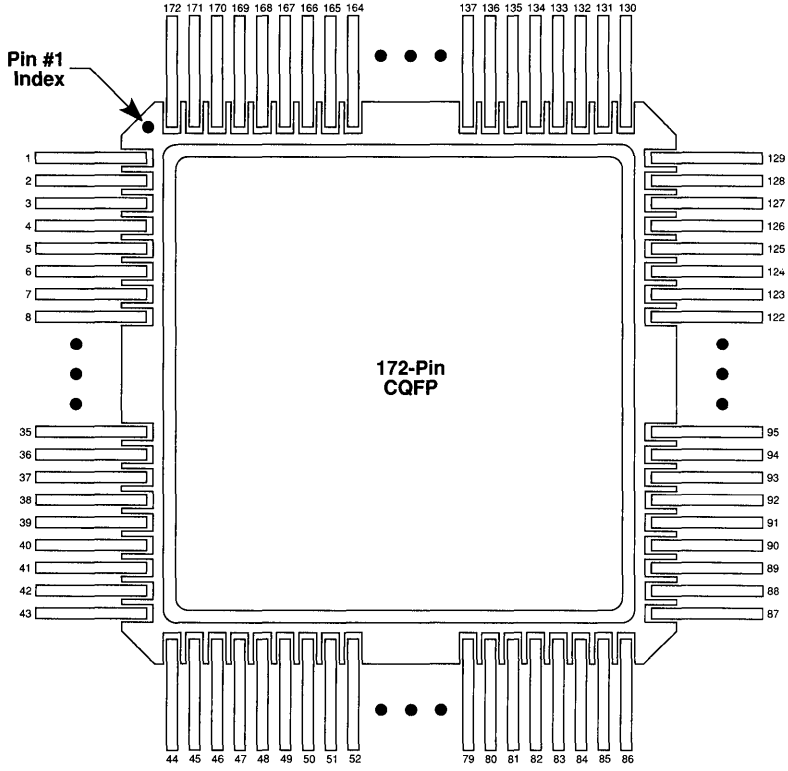


Notes:

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

Package Pin Assignments

172-Pin CQFP (Top View)

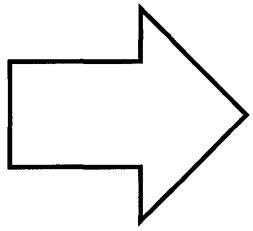


Signal	PIN Number
MODE	1
GND	7, 17, 22, 32, 37, 55, 65, 75, 98, 103, 108, 118, 123, 141, 152, 161
V_{CC}	12, 23, 27, 50, 66, 80, 109, 113, 136, 151, 166
V_{SV}	24, 110
V_{KS}	106
V_{PP}	107
SDI or I/O	131
PRA or I/O	148
PRB or I/O	156
CLKA or I/O	150
CLKB or I/O	154
DCLK or I/O	171

Notes:

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

PREP Data



Introduction to PREP™ Benchmarks

The PREP benchmarks measure the capacity and performance of high-capacity programmable logic devices. A benchmark circuit is repeated to completely fill the measured device in a step and repeat manner. Each circuit's input is connected to the output of the previous circuit. The first instance's inputs are connected to input pins, and the last instance's outputs are connected to output pins.

Capacity

Capacity is measured by reporting the number of instances of the benchmark circuit that fit in a particular device. Nine different benchmark circuits are used and thus capacity measurements are reported for all nine of the benchmarks for an individual device. For example, the first benchmark circuit is a datapath circuit and is shown in Figure 1. The circuits are repeated as shown in Figure 2, with inputs and outputs interconnected. Notice that some signals are global and go to all instances, while other signals are only used to interconnect instances. The first and last instances are connected to package inputs and outputs. In the A1425, ten instances of the datapath circuit can fit; thus, the capacity for benchmark 1 is reported as 10. It is important to point out that the current PREP methodology requires that each benchmark instance be implemented exactly the same. This makes it impossible to use the A1425 IO modules to the fullest extent thus reporting a fewer number of benchmark instances than what actually fits into the device. Real designs are able to utilize the A1425 IO modules, and the actual capacity of the device is higher than the PREP benchmarks indicate.

Performance

Performance is reported in the benchmarks by measuring the worst-case commercial maximum operating frequency of each benchmark instance. The maximum operating frequency of a single benchmark instance is determined by the slowest of the following:

- The longest path between a flip-flop in the benchmark instance and a flip-flop in the previous benchmark instance, including any combinatorial delays between the flip-flops
- The longest path between any two flip-flops in the same benchmark instance, including any combinatorial delays between the flip-flops

The first benchmark instance performance is not reported, since it has no previous instance.

The slowest, fastest, and mean frequency of the benchmark instances in a given device are then reported for each benchmark. For example, all of the datapath benchmark instances run at a worst-case frequency of 125 MHz, so the minimum frequency is 125 MHz, the maximum is 125 MHz, and the mean is 125 MHz.

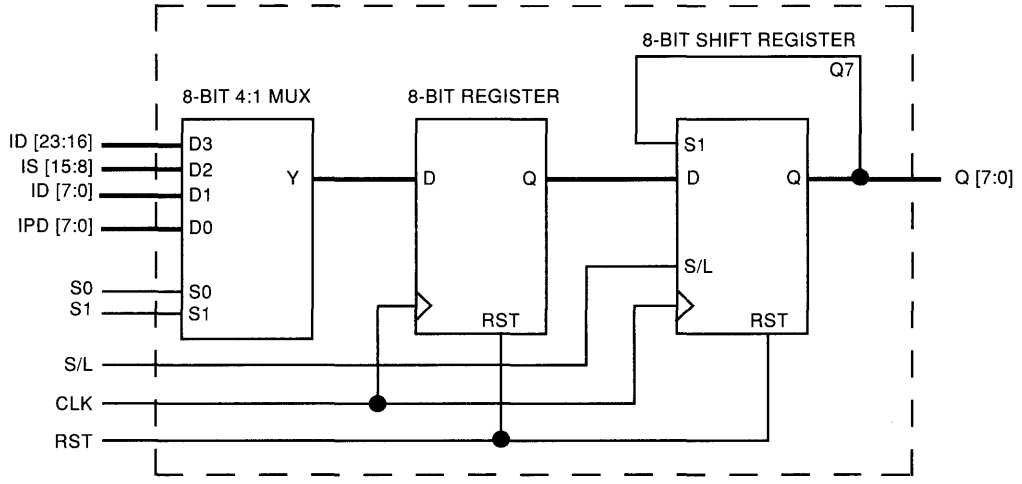


Figure 1. A Datapath Benchmark Circuit

External Performance

External frequency of operation is also reported by taking the output of the last benchmark instance and connecting it to the input of the first benchmark instance through the input and output pins of the device. A single number is reported since only a single instance exists for this measurement.

More Information

For more information on the PREP benchmarks, contact PREP Corp. at 408-356-2169 and ask for data on the PREP benchmark suite.

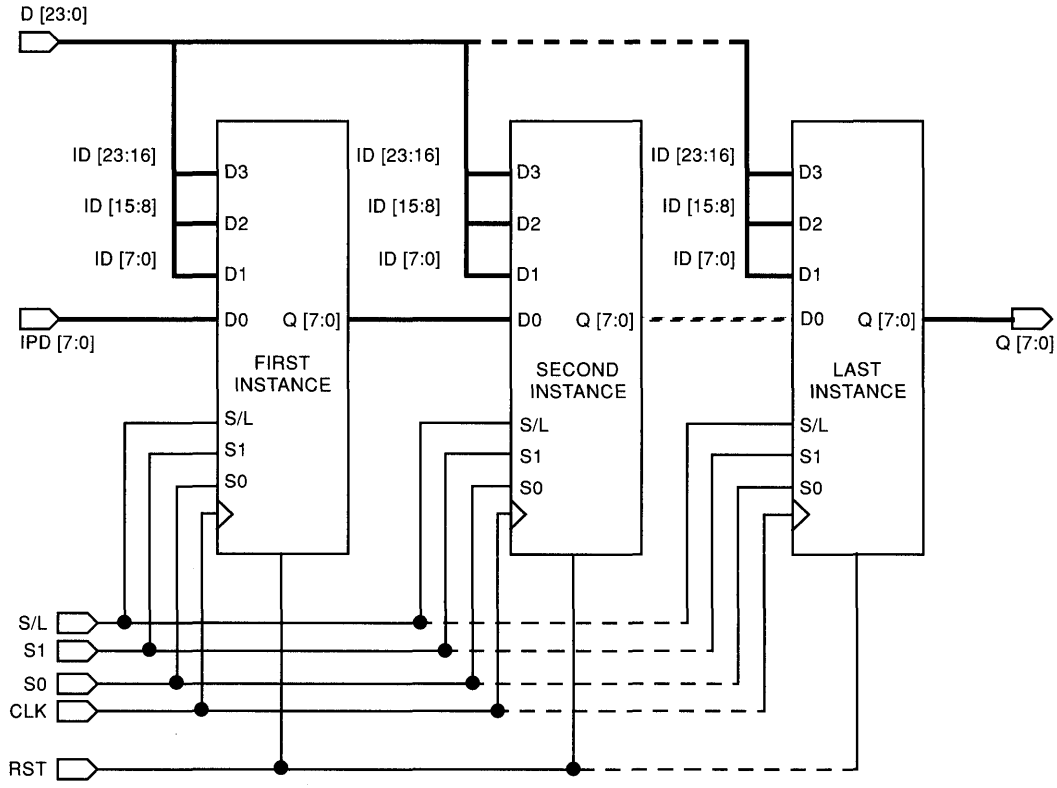


Figure 2. A Datapath Benchmark Circuit with Interconnected Inputs and Outputs

PREP™ Benchmarks Confirm Cost-Effectiveness of FPGAs

PREP benchmarks help determine the best value in programmable devices.

The Programmable Electronics Performance Corporation (PREP) has released the programmable logic industry's first performance and capacity benchmarks. In conjunction with published pricing, the PREP benchmarks offer excellent insight into the value of programmable architectures and devices. The following summarization of the PREP benchmarks introduces simple but effective methods for interpreting the PREP data and choosing an appropriate programmable device based on the value, performance, and predictability of the architecture.

Interpreting PREP Benchmark Results

The PREP benchmarks measure both capacity and performance for each of the nine benchmark circuits. The capacity of a device is expressed by the number of instances, or repetitions, of each benchmark circuit that fits in the programmable device. The performance of a device is reported for both internal and external operating frequencies for each benchmark circuit. All benchmarks are measured while automatically placing and routing the design circuits. Manual placement and routing capabilities are shown optionally by some manufacturers.

Since most designers in the evaluation stage have not used all the manufacturers' design tools, it would be wise to compare automatically placed and routed designs in similar capacity devices. A simple and effective way to compute capacity is to average the benchmark unit repetitions over all nine benchmarks; a good way to compare performance is to average the benchmark suite's mean internal operating frequency (Fmean) while automatically placeing and routing the designs.

Best Value

The PREP benchmarks have set realistic expectations for performance and capacity, and when used in conjunction with equitable pricing comparisons, they can provide an excellent measure of value.

A simple method of determining value is to plot the average performance of a device versus its price per unit of capacity, as shown in Figure 1. When viewing the chart, one will find that the less expensive devices are positioned towards the left and the

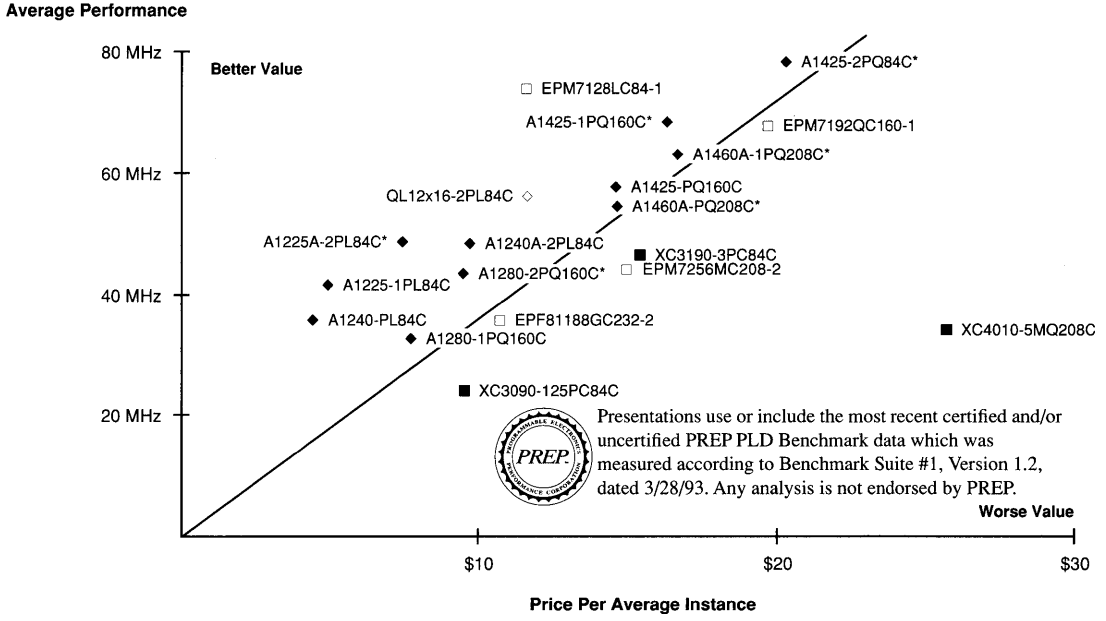


Figure 1. *The average performance, expressed in MHz, is the mean internal operating frequency averaged over the set of benchmark circuits. Price per unit of capacity, expressed as price per average instance, is the 100-piece distributor price quotation for the device (March 1993) divided by the average of the number of unit repetitions for the nine benchmarks. Projected values (*) are based on simulated perfomance and projected pricing.*

higher performance devices towards the top. A "value line" is drawn through the chart with best value (7 of 13 devices with multiple speed grades) appearing above the line. These provide either the same performance as more costly devices or better performance than similarly priced devices. The value line represents an increasing cost with increasing performance.

Figure 1 clearly shows that Actel ACT™ 2 devices (A12xx...) offer the highest value for designs with performance up to 50 MHz. Device comparisons show the Actel A1240A-2PL84C device offering 48.4 MHz average performance for $9.78 per average instance, the Xilinx XC3190-3PC84C device offering 46.2 MHz average performance for $15.41 per average instance, and the Altera EPM7256GC192-2 device offering 45.3 MHz average performance for $25.77 per average instance. The antifuse-based Actel device has the highest value, offering higher performance for a cost 37% less than the Xilinx device and 62% less than the Altera device.

The data in Figure 1 is also important in understanding the value of devices by architecture. The antifuse-based FPGA architectures (Axx, QLxx devices) offer the highest value across a wide range of performances and capacities. Indeed, six of the seven "high value" devices are antifuse-based FPGAs, with five of these six devices offered by Actel's ACT 2 (A12xx) and ACT 3 (A14xx) families. The fixed architecture of the Complex-PLD devices (EPMxx) offers high performance at low gate densities, but the architecture's performance dwindles and costs rise dramatically as gate capacity increases. The SRAM-based FPGA architecture (XCxx, EPFxx devices) is quite expensive and lacking in device performance, with all SRAM-based devices falling below the value line.

Conclusions:

1. Actel ACT 2 family devices are the best value in the mid-range and high-density classes.
2. Actel antifuse-based FPGAs represent five of the seven best programmable devices with the best value based on certified PREP results.
3. Actel ACT 2 family devices have a 40% to 70% cost advantage over similar Xilinx devices.

Classifying Programmable Devices

Determining the right programmable device for a design can be a difficult task. Information regarding the usable device capacity, performance, I/O count, and price is vital in choosing the most appropriate programmable device. Graphically displaying the devices by the PREP benchmark measurements of performance and capacity, as shown in Figure 2, can expedite this decision process.

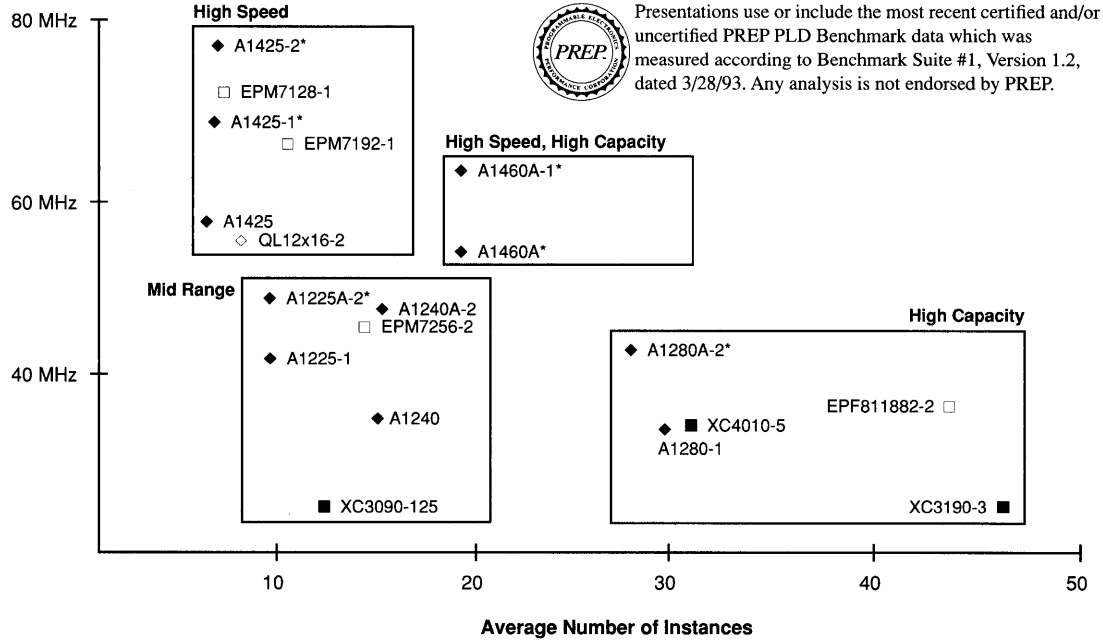


Figure 2. *The average performance, expressed in MHz, is calculated as the mean internal operating frequency averaged over the set of benchmark circuits while using 100% automatic placement and routing software. The device capacity, expressed as the average number of instances, is the average number of unit repetitions for the nine benchmarks. Projected values (*) are based on simulated perfomance and projected capacity.*

When viewing the chart, one will find distinctive device classifications. The high-speed devices congregate in the upper left-hand corner of the chart, the high-speed, high-capacity devices in the upper middle to right side of the chart, the high-capacity devices in the middle of the right side of the chart, and the remaining mid-range devices in the middle of the chart.

These four classifications offer designers a quick means of determining which devices will satisfy their performance and capacity requirements. For instance, if a design needs both high speed and high capacity, the designer would quickly realize that only the Actel A1460 device offers both characteristics. Once the required device classification is determined, the designer can then compare the devices' values by reviewing Figure 1. For instance, if a designer needed only moderate speed and density, the mid-range devices would suffice. Recalling the data from Figure 1, the best value in the mid-range devices can be realized by buying Actel A1225 or A1240 devices. Similar value, or cost, comparisons can be made for any classification. When comparing devices based on PREP benchmark data, it is wise to make performance and value comparisons only between devices within the same classification. In addition, the availability of user I/Os varies by package type. Designers should reference vendor package literature to determine which packaged devices provide the required interconnect resources.

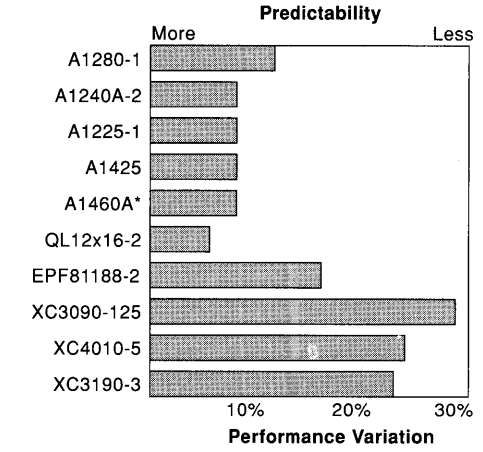
Conclusions:

1. Device-to-device comparisons are valid within each class, but are inappropriate between devices in different classes.
2. The Actel ACT 3 family is the only architecture that offers high-speed, high-capacity class devices. The ACT 3 family also offers the highest performance in the high-speed classification.
3. Actel ACT 2 family devices are the best value in the mid-range and high-density classes.

Performance Predictability

With time-to-market issues becoming a significant factor, the ability of a device to meet required performance goals without using manual optimization is vital. Design architectures that offer both design flexibility and highly predictable performance while using 100% automatic place and route software can ensure a designer that his or her design will be completed on time.

PLDs, known for their predictability, do not offer the design flexibility required in many time-to-market applications. FPGAs, known for their flexible design architecture, can also offer very predictable performance, as shown in Figure 3. The PREP benchmark data shows antifuse-based FPGAs (Axx, QLxx devices) to be much more predictable than SRAM-based FPGAs (EPFxx, XCxx devices). In the mid-range class, Actel A1240 and A1225 devices are three to four times more predictable than their Xilinx counterparts (XC3xx devices). In the high-density class, the Actel A1280 device is up to twice as predictable as its SRAM counterparts (XC4xx, EPFxx devices). Only antifuse-based FPGAs are classified as high-speed or high-speed, high-density devices.



Presentations use or include the most recent certified and/or uncertified PREP PLD Benchmark data which was measured according to Benchmark Suite #1, Version 1.2, dated 3/28/93. Any analysis is not endorsed by PREP.

Figure 3. A measure of predictability can be determined by computing the standard deviation of the best- and worst-case performance of each benchmark and dividing by the average benchmark performance. The resulting number measures the expected percentage spread around the average benchmark performance a device will exhibit while using 100% automatic placement and routing software. The lower the number, the more predictable the device performance is expected to be. Projected predictability (*) is based on simulated performance.

Conclusions:

1. Only antifuse-based FPGAs offer both high flexibility and highly predictable performance.
2. The Actel ACT 2 devices are 3 to 4 times more predictable than Xilinx devices in the mid-range class and are up to twice as predictable as other devices in the high-capacity class.

Many of the benchmarked devices exhibit a wide range of capacity depending on the characteristics of the benchmark designs. Some devices are more efficient at state machines, some more efficient at arithmetic functions, and so on. The ideal devices would be efficient for all types of applications and exhibit a narrow spread in gate capacity over the range of applications represented by the benchmark suite. A wide variation in gate count means it is difficult for users to predict whether a particular application will fit in a given device. A smaller variation in gate count makes it easier for the user to estimate if the application will fit, since gate count will be independent of application. Additionally, a device with a small variation will be more general purpose and thus able to address a wider set of applications.

A numeric measure of this predictability can be expressed by computing the standard deviation of the difference between the device capacity for an individual benchmark and the average capacity over the entire suite and dividing this number by the

average benchmark capacity. The resulting number measures the expected percentage spread around the average benchmark capacity a device will exhibit. The lower the number the more tightly around the average the capacity will fall, and the more predictable the device is expected to be. As shown in Figure 4, the Actel devices have the tightest spread of any of the benchmarked FPGA devices.

The narrow distribution in gate counts for Actel devices is the direct result of the fine-grained architecture and the antifuse interconnect element used in the devices. The logic building blocks are small, making them very efficient at implementing a wide variety of logic functions. Larger grained devices (like Xilinx devices) are less efficient if their logic blocks are under-utilized by a logic function. Conversely, if the logic application fits the logic building block just right, a much higher than expected gate count can result. Actel devices also contain abundant routing resources, thanks to the antifuse, which ensures that each logic block can be used (high logic module utilization) and that application dependent routing congestion can be eliminated. Routing congestion can significantly reduce the usable gates in architectures with fewer routing resources. Routing requirements are application dependent and sometimes difficult to judge at the beginning of the design phase. Sometimes the types of logic components needed at the beginning of a design can be estimated, but the interconnectivity between blocks and the interactions between various routing needs are very difficult to judge before completing the design. Congestion can make it difficult to use every logic module and thus can reduce device capacity significantly.

Conclusions:

1. The capacity of Actel devices is the most predictable of any benchmarked device because of its fine-grained architecture and abundant routing resources available when using an antifuse interconnect element and patented segmented channel routing.

Designers now have a resource to measure and predict FPGA performance by using PREP benchmarks. Actel's ACT 2 devices are production-proven FPGAs offering the highest value and predictability for designs running up to 50 MHz. The ACT 3 family introduces an entire new class of devices by offering the only high-speed, high-capacity programmable devices in the industry. The ACT 3 family also offers the highest performance in the high-speed class of programmable devices.

For detailed information on the ACT 2 and ACT 3 families of field programmable gate arrays or to obtain a copy of the complete PREP benchmarks on Actel devices, please call PREP Corp. at 408-356-2169 and ask for the Actel benchmark results.

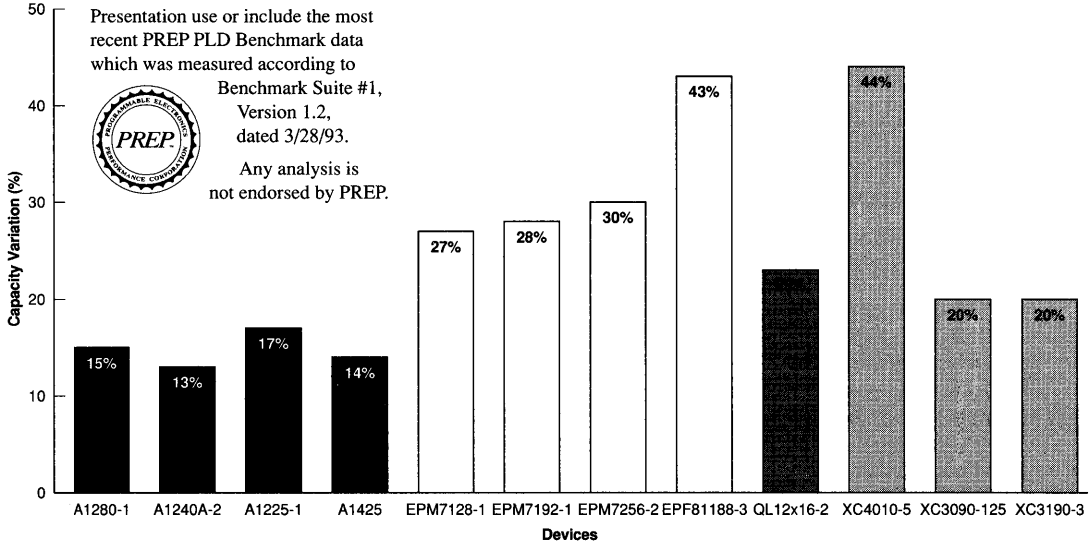


Figure 4. *The bar chart above shows a measure of capacity predictability of each benchmarked FPGA device. The standard deviation of the difference between the device gate capacity for a benchmark and the average gate capacity over the benchmark suite is divided by the average gate count over the benchmark suite [STDEVP(CapB1-CapAve, CapB2-CapAve,...)/CapAve]. The resulting percentage measures the spread in capacity of a device. The lower the percentage, the more predictable the capacity of the device. The Actel devices have the most predictable capacity of any benchmarked FPGA device.*

Actel PREP™ Benchmark Results

Datapath

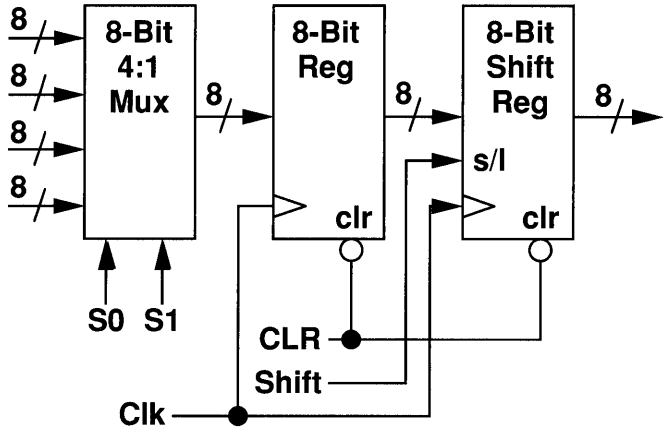
(Benchmark #1)

Functions

- Data Selecting/Holding
- Data Shifting
- 24-Bit Global Data Bus (Routing Intensive)

Actel Characteristics

- Single Logic Level
- Runs at Clock Rate
- Ample Routing Resources



Device/Grade	Capacity	Performance				Optimized	
	Reps	Worst	Best	Mean	Ext	Cap	Perf
A1280-1	48	29	41	34	12	X	
A1280-1	39	55	65	61	30		X
A1240	21	66	66	66	30		X
A1240A-2	27	49	61	57	24	X	
A1240	27	37	46	43	20	X	
A1240A-2	21	95	95	95	35		X
A1225-1	18	44	53	49	25	X	
A1225-1	14	85	85	85	35		X
A1425	10	125	125	125	38		
A1425 (I/Os only)	7	64	74	70	50		

100% Automatically placed and routed in ALS 2.2 software.

Presentations use or include the most recent PREP PLD Benchmark data which was measured according to Benchmark Suite #1, Version 1.2, dated 3/28/93.

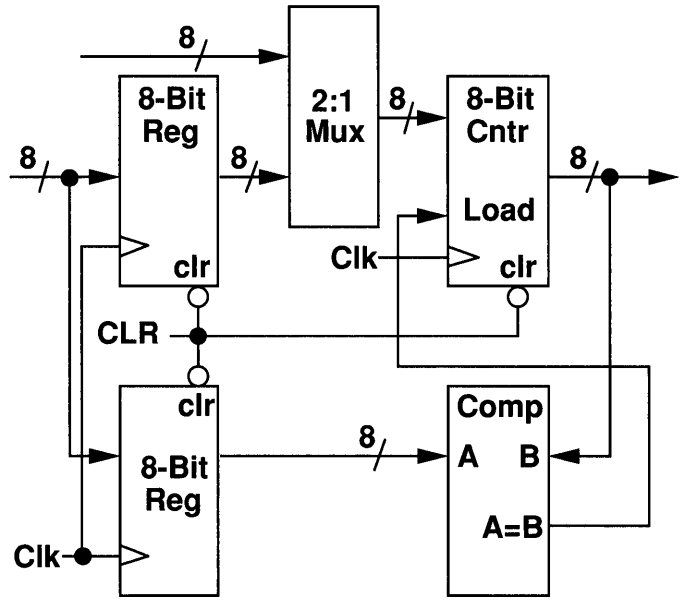
Any analysis is not endorsed by PREP.

Timer Counter

(Benchmark #2)

Functions
• Counting
• Data Comparing/Holding
• Bit Global Data Bus (Routing Intensive

Actel Characteristics
• Register Intensive
• Four Logic Level Compare
• High-Speed Counter
• Ample Routing Resources



Device/Grade	Capacity	Performance			
	Reps	Worst	Best	Mean	Ext
A1225-1	9	28	32	31	29
A1240A-2	14	35	36	35	24
A1280-1	20	19	23	21	12
A1425	6	39	44	41	45
A1240	14	26	27	27	18

100% Automatically placed and routed in ALS 2.2 software.

Presentations use or include the most recent PREP PLD Benchmark data which was measured according to Benchmark Suite #1, Version 1.2, dated 3/28/93.

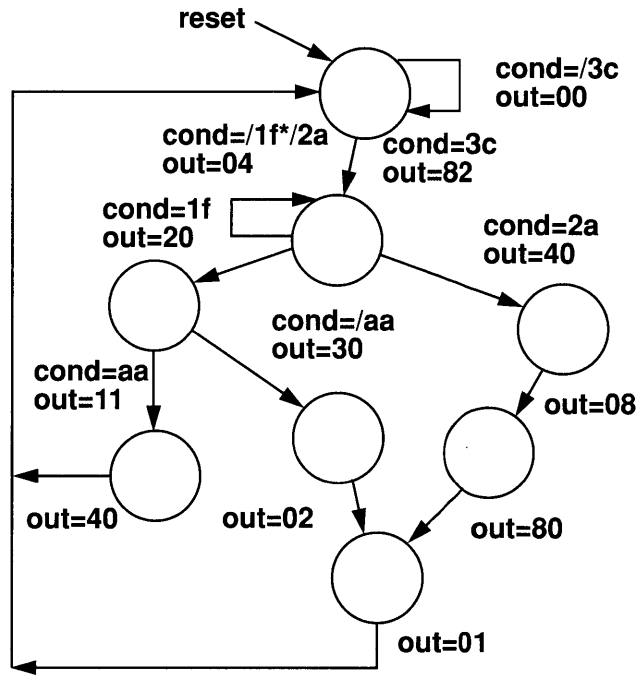
Any analysis is not endorsed by PREP.

State Machine

(Benchmark #3)

Functions
• Simple State Machine
• Eight States, Eight Inputs, Eight Outputs
• Mostly Single Transitions

Actel Characteristics
• Three Logic Levels
• Use Bit Per State



Device/Grade	Capacity	Performance			
	Reps	Worst	Best	Mean	Ext
A1225-1	15	32	36	34	21
A1240A-2	23	37	41	39	23
A1280-1	42	26	31	29	19
A1425	10	42	48	46	29
A1240	23	28	30	29	18

100% Automatically placed and routed in ALS 2.2 software.

Presentations use or include the most recent PREP PLD Benchmark data which was measured according to Benchmark Suite #1, Version 1.2, dated 3/28/93.

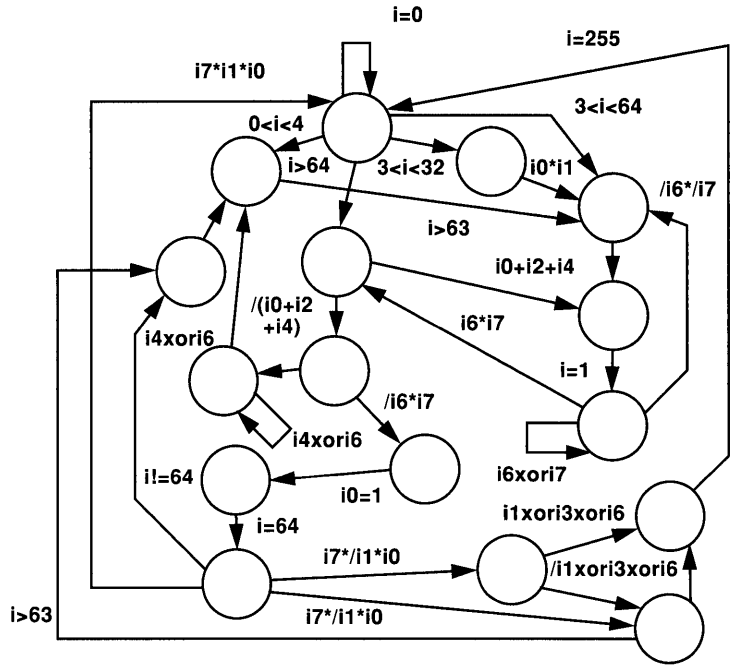
Any analysis is not endorsed by PREP.

Large State Machine

(Benchmark #4)

Functions
• State Machine
• 16 States, 8 Inputs, 8 Outputs
• Complex Transactions and Outputs

Actel Characteristics
• Five Logic Levels
• Use Bit Per State



Device/Grade	Capacity	Performance			
	Reps	Worst	Best	Mean	Ext
A1225-1	6	21	22	21	15
A1240A-2	10	22	25	24	17
A1280-1	18	16	19	17	14
A1425	4	26	29	28	20
A1240	10	17	19	18	13

100% Automatically placed and routed in ALS 2.2 software.

Presentations use or include the most recent PREP PLD Benchmark data which was measured according to Benchmark Suite #1, Version 1.2, dated 3/28/93.

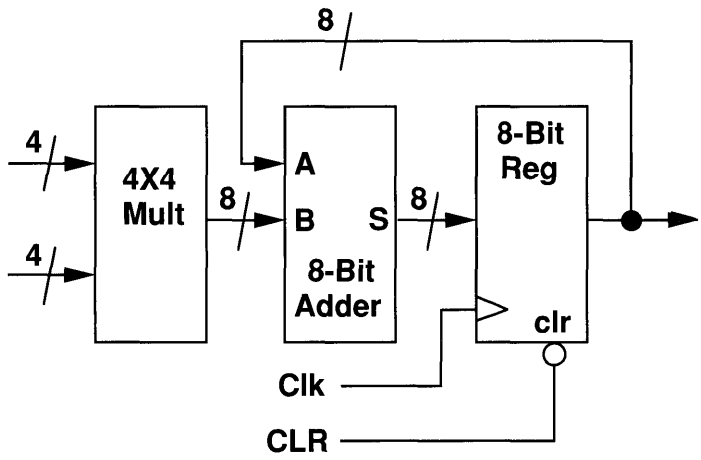
Any analysis is not endorsed by PREP.

Arithmetic

(Benchmark #5)

Functions
• Multiplying
• Adding
• Data Holding

Actel Characteristics
• 10 Logic Levels
• Module Speed Trade-Offs



Device/Grade	Capacity	Performance			
	Reps	Worst	Best	Mean	Ext
A1225-1	6	13	13	13	10
A1240A-2	9	14	15	15	12
A1280-1	16	11	12	11	9
A1425	4	18	18	18	14
A1240	9	11	11	11	9

100% Automatically placed and routed in ALS 2.2 software.

Presentations use or include the most recent PREP PLD Benchmark data which was measured according to Benchmark Suite #1, Version 1.2, dated 3/28/93.

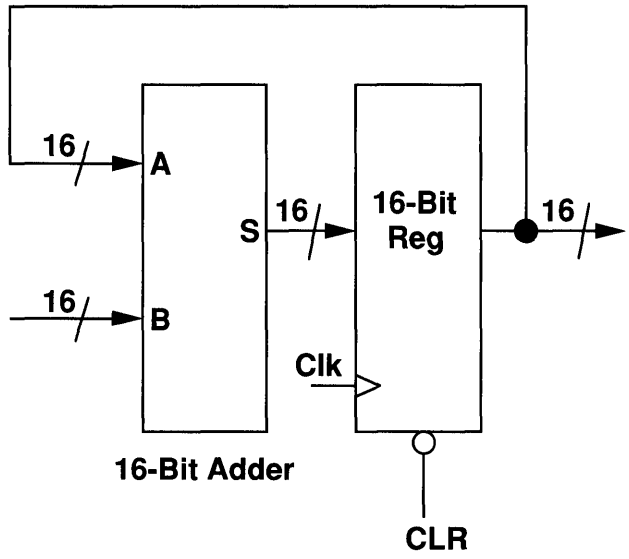
Any analysis is not endorsed by PREP.

Accumulator

(Benchmark #6)

Functions
• 16-Bit Adder
• 16-Bit Register

Actel Characteristics
• Three Logic Levels
• Parallel Implementation Faster than Competitor's Serial Approaches
• Module Speed Trade-Offs



Device/Grade	Capacity	Performance				Optimized	
	Reps	Worst	Best	Mean	Ext	Cap	Perf
A1225-1	5	23	25	24	16	X	
A1240A-2	8	27	30	29	20	X	
A1280-1	15	18	23	21	17	X	
A1425	3	36	37	36	26	X	
A1240	8	21	23	22	15	X	
A1225-1	4	32	33	32	20		X
A1240A-2	7	32	36	36	24		X
A1280-1	13	21	28	28	17		X
A1425	3	39	40	40	28		X
A1240	7	24	27	27	18		X

100% Automatically placed and routed in ALS 2.2 software.

Presentations use or include the most recent PREP PLD Benchmark data which was measured according to Benchmark Suite #1, Version 1.2, dated 3/28/93.

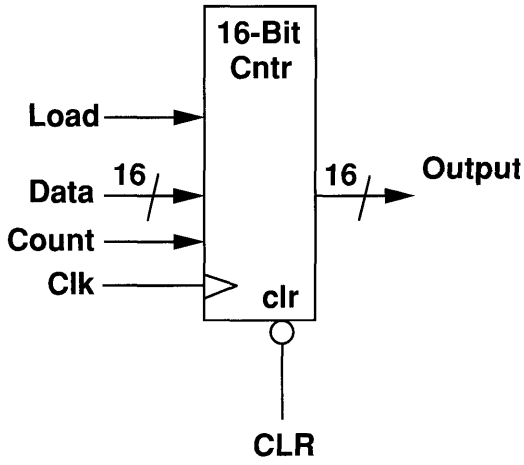
Any analysis is not endorsed by PREP.

16-Bit Counter

(Benchmark #7)

Functions
• Counting
• Loading
• Count After Load

Actel Characteristics
• Two Logic Levels
• Uses Look Ahead to Cut Logic Levels



Device/Grade	Capacity	Performance			
	Reps	Worst	Best	Mean	Ext
A1225-1	9	46	54	50	29
A1240A-2	13	52	64	58	32
A1280-1	25	33	47	41	21
A1425	6	65	70	68	40
A1240	13	39	48	44	24

100% Automatically placed and routed in ALS 2.2 software.

Presentations use or include the most recent PREP PLD Benchmark data which was measured according to Benchmark Suite #1, Version 1.2, dated 3/28/93.

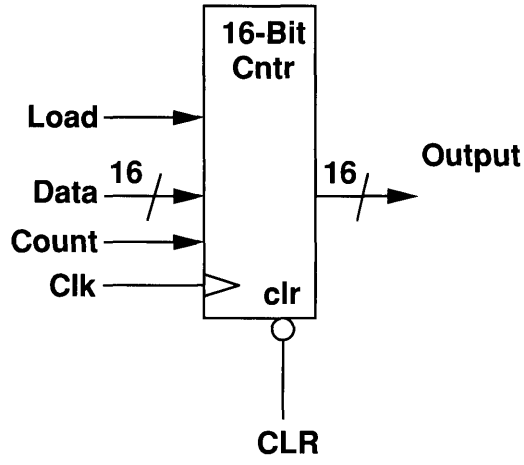
Any analysis is not endorsed by PREP.

16-Bit Prescaled Counter

(Benchmark #8)

Functions
• Counting
• Multicycle
• Uses Prescalar for LSBs

Actel Characteristics
• Single Logic Level
• Runs at Full Clock Rate



Device/Grade	Capacity	Performance			
	Reps	Worst	Best	Mean	Ext
A1225-1	6	85	85	85	18
A1240A-2	10	95	95	95	20
A1280-1	18	75	75	75	15
A1425	4	108	123	114	27
A1240	10	66	66	66	15

100% Automatically placed and routed in ALS 2.2 software.

Presentations use or include the most recent PREP PLD Benchmark data which was measured according to Benchmark Suite #1, Version 1.2, dated 3/28/93.

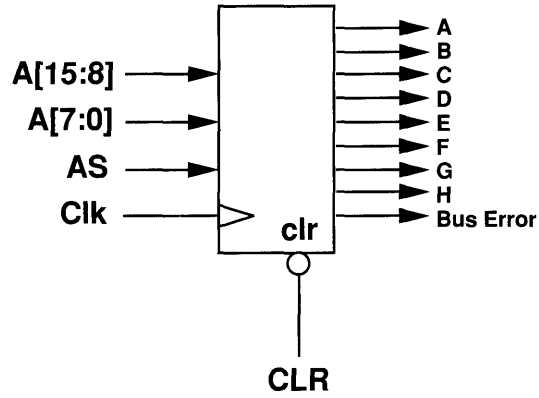
Any analysis is not endorsed by PREP.

Memory Map

(Benchmark #9)

Functions
• Address Decode
• Address Hold
• Wide Gating
• 8-Bit Global Address Bus (Routing Intensive

Actel Characteristics
• Three Logic Levels
• Abundant Routing



Device/Grade	Capacity	Performance			
	Reps	Worst	Best	Mean	Ext
A1225-1	20	33	36	34	21
A1240A-2	31	37	42	39	23
A1280-1	56	26	33	30	20
A1425	14	43	53	49	30
A1240	31	37	42	39	23

100% Automatically placed and routed in ALS 2.2 software.

Presentations use or include the most recent PREP PLD Benchmark data which was measured according to Benchmark Suite #1, Version 1.2, dated 3/28/93.

Any analysis is not endorsed by PREP.

Estimating FPGA Applications Performance Using PREP™ Benchmarks

Application Note

Introduction

The PREP Benchmarks provide data for estimating the performance of an application in an Actel field programmable gate array (FPGA). The PREP benchmarks were developed to help programmable logic users better understand the capabilities of programmable logic devices. Several common benchmark functions were selected to measure capacity and performance. The data is used to determine which device would suitably implement a particular application early in the design process. Before benchmarks, large portions of a design had to be completed to estimate the potential fit of the application in the device.

The nine PREP benchmarks are listed in Table 1. The performance of all nine benchmarks was measured in Actel FPGAs. The methodology requires that a benchmark circuit is continually repeated, connecting outputs of one instance to the inputs of another until a device is full. The device performance is determined by measuring the maximum operating frequency at worst-case commercial temperature and voltage conditions. The maximum operating frequency is the slowest of the following two cases:

- The longest path between a register in the benchmark instance and a register in the previous benchmark instance, including any combinatorial delays between the registers
- The longest path between any registers in the same benchmark instance, including any combinatorial delays between the registers

Table 1. PREP Benchmarks

Benchmark	Function
Datapath	8-bit 4:1 Multiplexer, 8-bit register, and 8-bit shift register
Timer/Counter	Counter, Load Register, Compare Register, and Comparator (8 bit)
Small State Machine	8-State Machine with simple transition terms
Large State Machine	16-State Machine with complex transition terms
Arithmetic	4x4 Multiplier, 8-bit Adder, and 8-bit Register
Accumulator	Registered 16-bit Adder
Counter	16-bit Loadable Counter
Pre-scaled Counter	16-bit Counter optimized for counting (Multiple cycle load)
Memory Mapper	Multiple address Decode with error detection

Measuring FPGA Performance

These benchmarks can be applied to individual circuits to estimate FPGA performance. The first step in estimating FPGA performance is to identify the key functions of the design. Drawing a block diagram will usually suffice. Choose the major functions from the block diagram and estimate the required performance for each critical section. Next, identify which benchmarks match the critical functions in the design. The benchmark results for several ACT™ 2 and ACT 3 devices are given in Table 2. In addition, notice that "levels of logic" data is reported for each benchmark. This information aids in choosing the most similar benchmark for the particular function. Figure 1 displays an example of two levels of logic. Comparing the performance requirements and the benchmark results shows which Actel FPGA best fits the target application.

Estimate Example

Figure 2 represents a typical block diagram of a design, including a datapath, shift register, counters, and a state machine. The other parts of the design, such as various combinatorial functions are not important to the performance because they operate at low speeds. For this design, the target performance is 32 MHz for the relatively simple state machine; 32 MHz for the ALU, address counter, and datapath; and 64 MHz for the shift registers.

The next step is to match the critical functions to a particular benchmark. For example, since the interface state machine is relatively simple, the small state machine benchmark can be used to predict its performance. The shift register is part of the datapath circuit, so the datapath benchmark is a good choice. Likewise, because the ALU implements only logical operations and nothing else arithmetic, the accumulator benchmark will provide a good, but conservative estimate. The address counters in the design require two levels of logic, so the simple counter benchmark will be used to estimate their performance. The benchmark results of this example for several Actel devices are listed in Table 3. This estimation shows that the A1280-1 device misses the desired performance of the ALU block. However, the A1225-1, A1240A-2, and the A1425 devices meet all the required performances. Any of these devices would be able to implement the required application at the specified performance.

PREP Benchmarks serve an important role in predicting FPGA application performance. The task is relatively simple and fast. Now many devices can be excluded from consideration before the details of the design are completed. This allows for flexibility, reduced risk, and overall success with an Actel FPGA.

Note: The benchmark suite is available from PREP Corp. For more information, contact PREP Corp. at 504 Mino Ave., Los Gatos, CA, 95032, 408-356-2169.

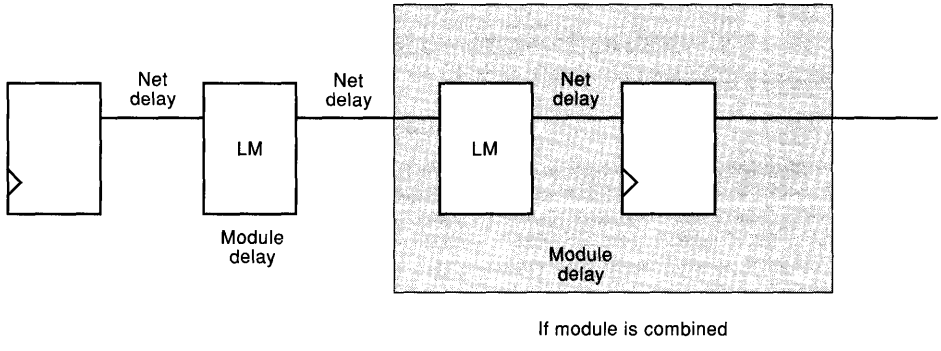


Figure 1. Logic Levels

Table 2. Benchnmark Performance of Actel Devices (Mean Performance)

Benchmark	Logic Levels	A1225-1	A1240A-2	A1280-1	A1425
Datapath	1	85 MHz	95 MHz	61 MHz	125 MHz
Timer/Counter	4	31 MHz	35 MHz	21 MHz	41 MHz
Small State Machine	3	34 MHz	39 MHz	29 MHz	46 MHz
Large State Machine	5	21 MHz	24 MHz	17 MHz	28 MHz
Arithmetic	10	13 MHz	15 MHz	11 MHz	18 MHz
Accumulator	3	32 MHz	36 MHz	28 MHz	40 MHz
Counter	2	50 MHz	58 MHz	41 MHz	68 MHz
Pre-scaled counter	1	85 MHz	95 MHz	75 MHz	114 MHz

Presentations use or include the most recent PREP PLD Benchmark data which was measured according to Benchmark Suite #1, Version 1.2, dated 3/28/93.

Any analysis is not endorsed by PREP.

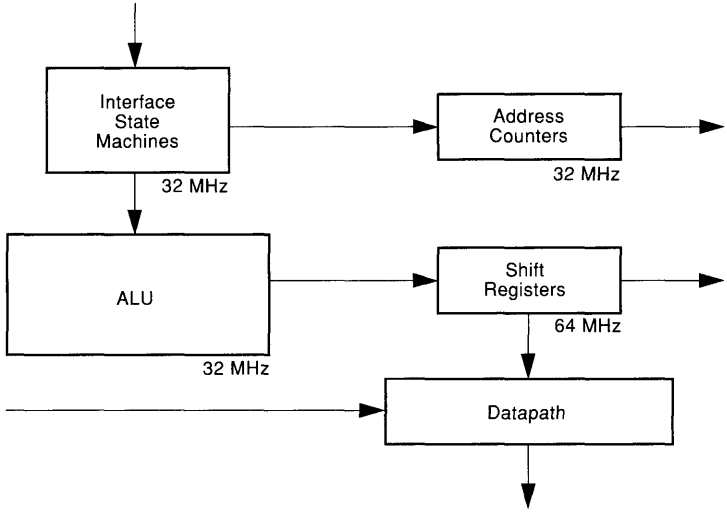


Figure 2. Example Design

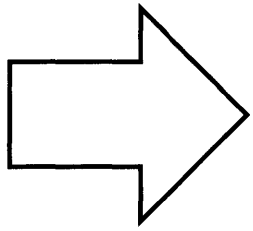
Table 3. Benchmark Estimates

Functions	Benchmark	Target Speed	A1225-1	A1240A-2	A1280-1	A1425
Address Counter	Counter	32 MHz	50 MHz	58 MHz	41 MHz	68 MHz
ALU	Accumulator	32 MHz	32 MHz	36 MHz	28 MHz	40 MHz
Shift Register	Datapath	64 MHz	85 MHz	95 MHz	61 MHz	125 MHz
State Machine	Small State Machine	32 MHz	34 MHz	39 MHz	29 MHz	46 MHz

Presentations use or include the most recent PREP PLD Benchmark data which was measured according to Benchmark Suite #1, Version 1.2, dated 3/28/93.

Any analysis is not endorsed by PREP.

Packaging and Mechanical Drawings



Package Options: User I/Os per Package

Package	Pins	ACT 1		ACT 2			ACT 3				
		A1010B	A1020B	A1225A	A1240A	A1280A	A1415A	A1425A	A1440A	A1460A	A14100A
PLCC	44	34	34								
	68	57	57								
	84	69	69	72	72		70	70			
PQFP	100	57	69	83			80	80			
	144				104						
	160					125		100	130		
	208									167	
VQFP	80	57	69								
CPGA	84	57	69								
	100			83			80				
	132/133				104			100			
	175/176					140			140		
	207									168	
	257										228
CQFP	84		69								
	172					140					

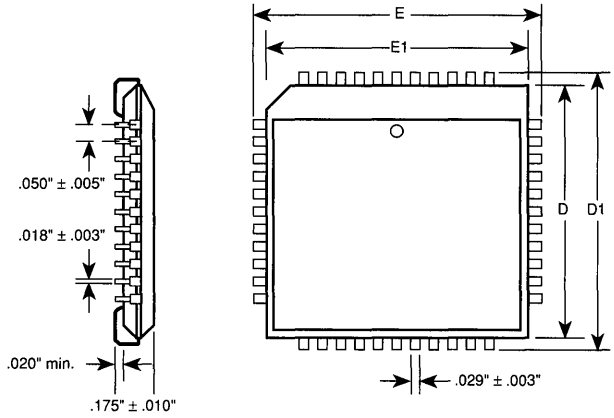
Package Thermal Characteristics

Package Type		Pin Count	Θ_{JC}	Θ_{JA} Still Air	Θ_{JA} 300 ft/min	Units
Plastic Leaded Chip Carrier	PLCC	44	15	52	40	°C/W
		68	13	45	35	°C/W
		84	15	44	38	°C/W
Plastic Quad Flatpack	PQFP	100	13	55	47	°C/W
		144	15	35	26	°C/W
		160	15	33	26	°C/W
		208	15	33	26	°C/W
Very Thin (1.0mm) Quad Flatpack	VQFP	80	12	68	55	°C/W
Cermaic Pin Grid Array	CPGA	84	8	33	20	°C/W
		100	8	35	17	°C/W
		132	5	30	15	°C/W
		133	8	30	15	°C/W
		175	8	25	14	°C/W
		176	8	23	12	°C/W
		207	8	22	13	°C/W
		257	2	15	8	°C/W
Ceramic Quad Flatpack	CQFP	84	5	40	30	°C/W
		172	8	25	15	°C/W

Package Mechanical Drawings

Plastic J-Leaded Chip Carrier

44-, 68-, and 84-Pin PLCC

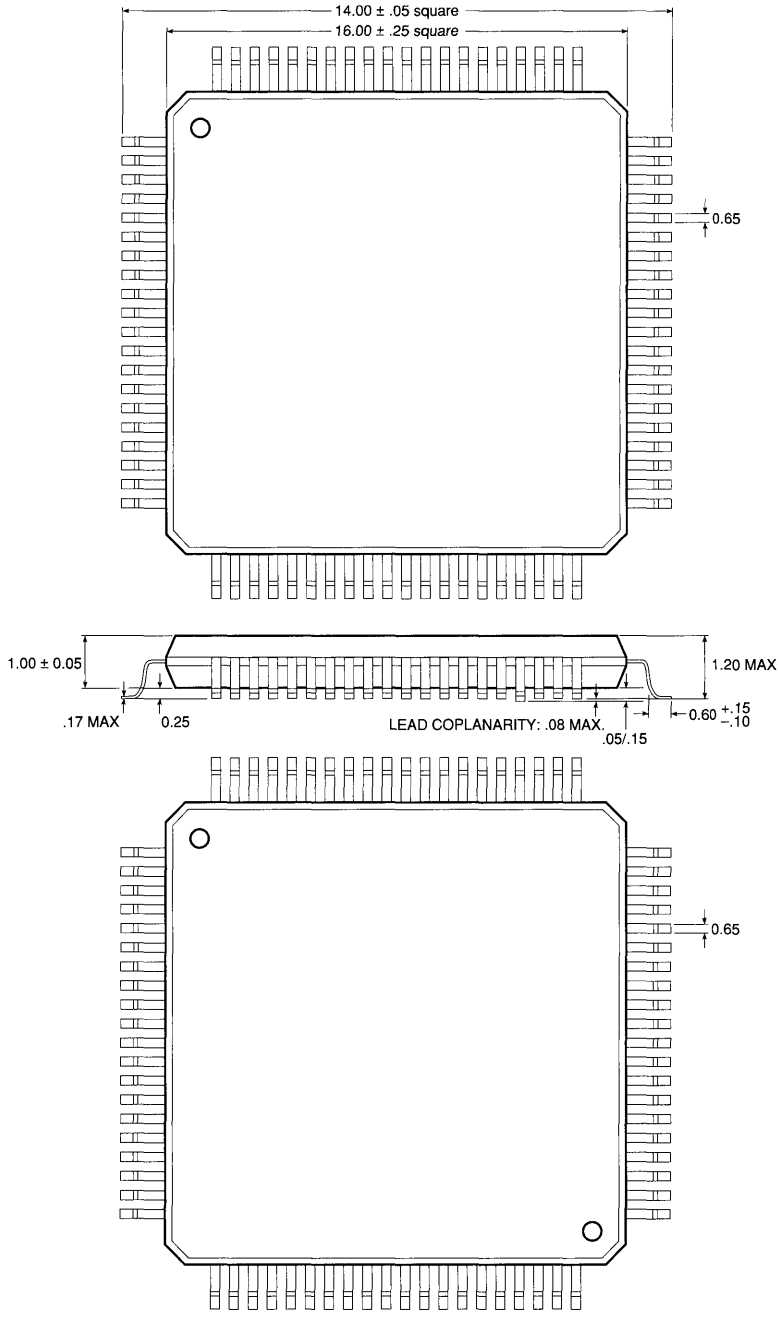


Lead Count	D,E	D1, E1
44	.690" ± .005"	.655" ± .005"
68	.990" ± .005"	.955" ± .005"
84	1.190" ± .005"	1.155" ± .005"

Very Thin Quad Flatpack

80-pin VQFP

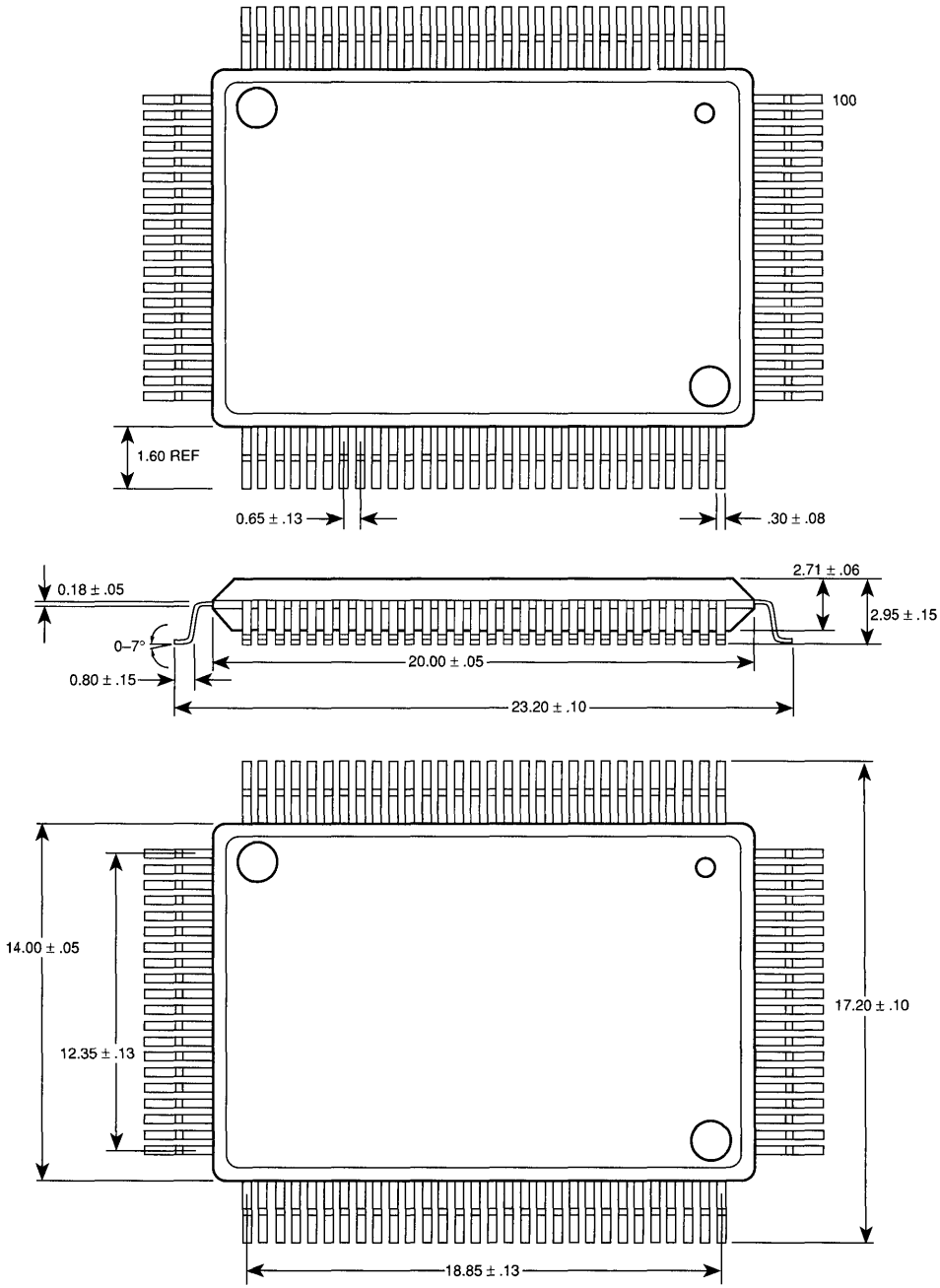
Dimensions in millimeters



Plastic Quad Flatpack

100-Pin PQFP

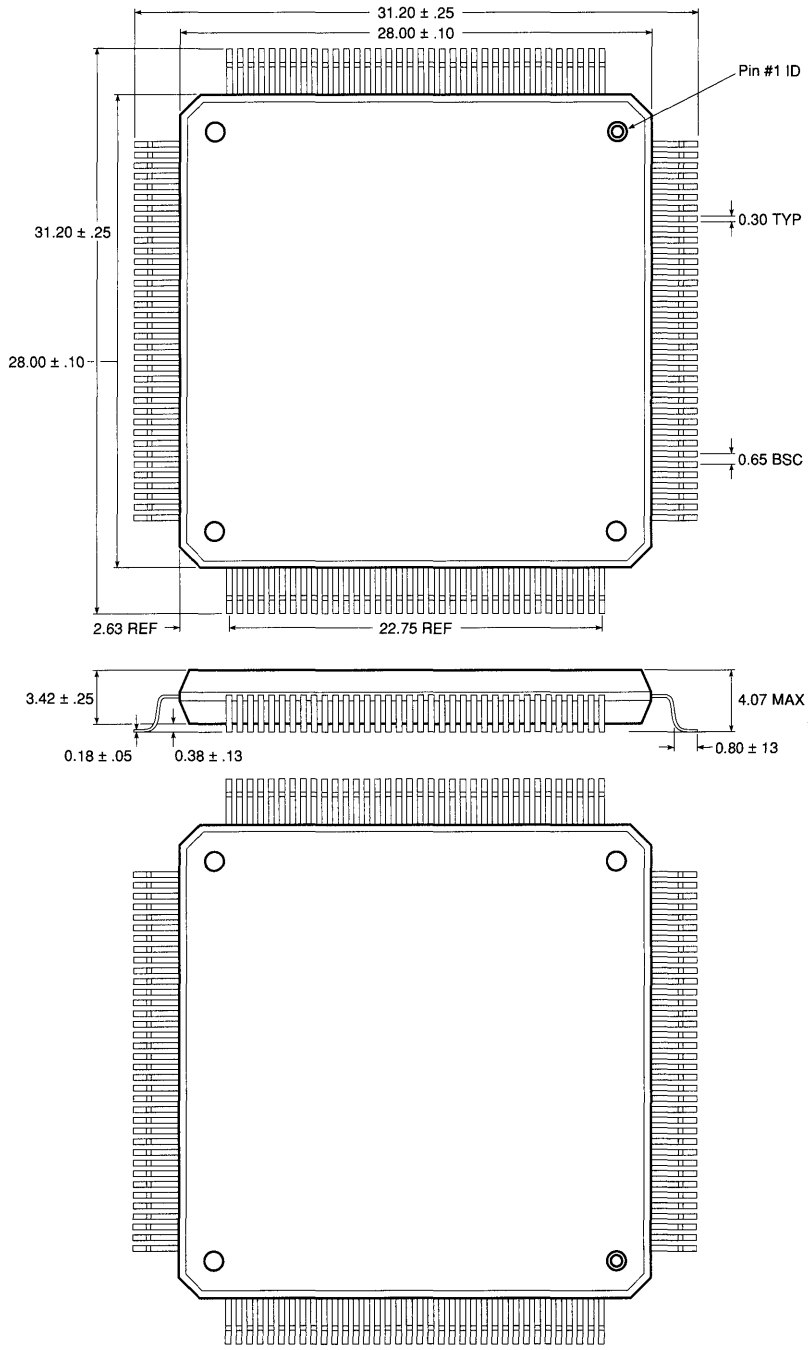
Dimensions in millimeters.



Plastic Quad Flatpack (continued)

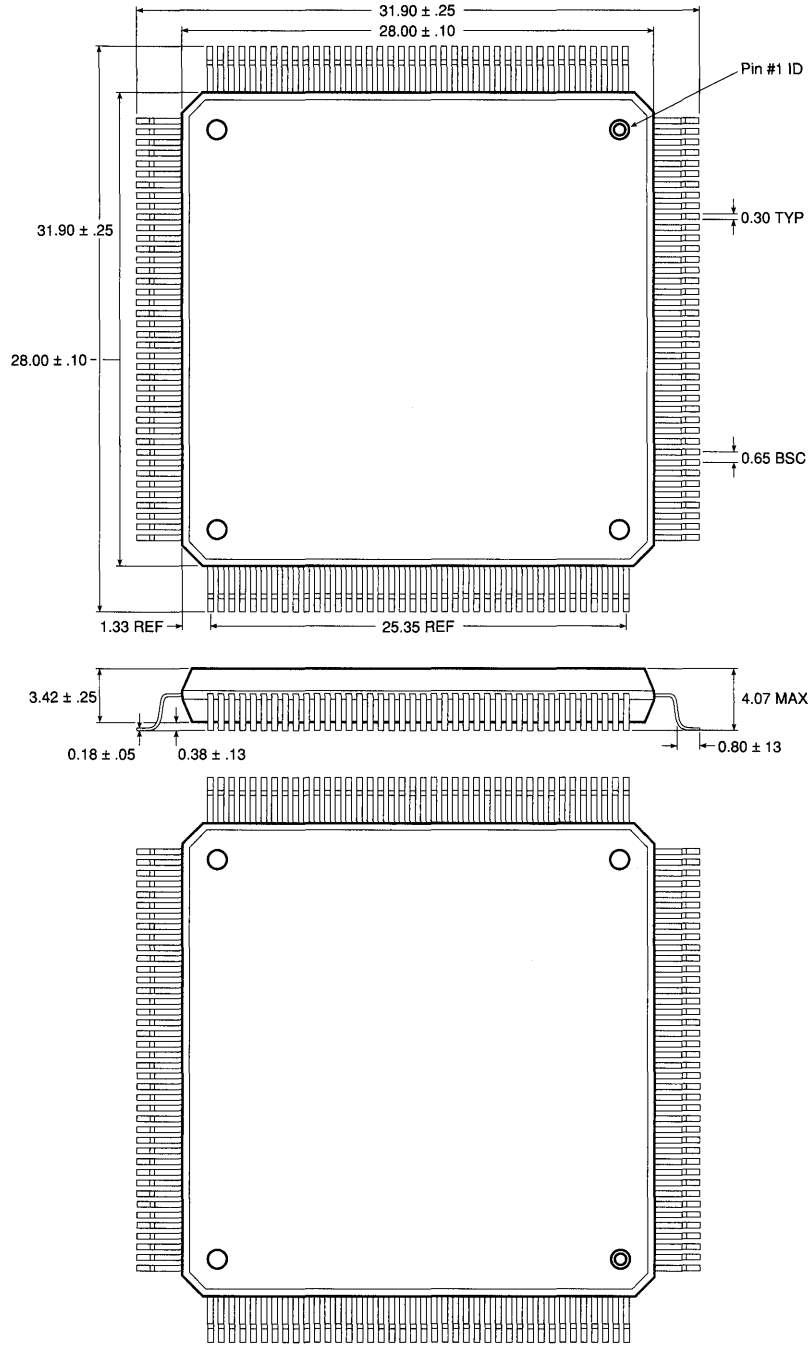
144-Pin PQFP

Dimensions in millimeters.



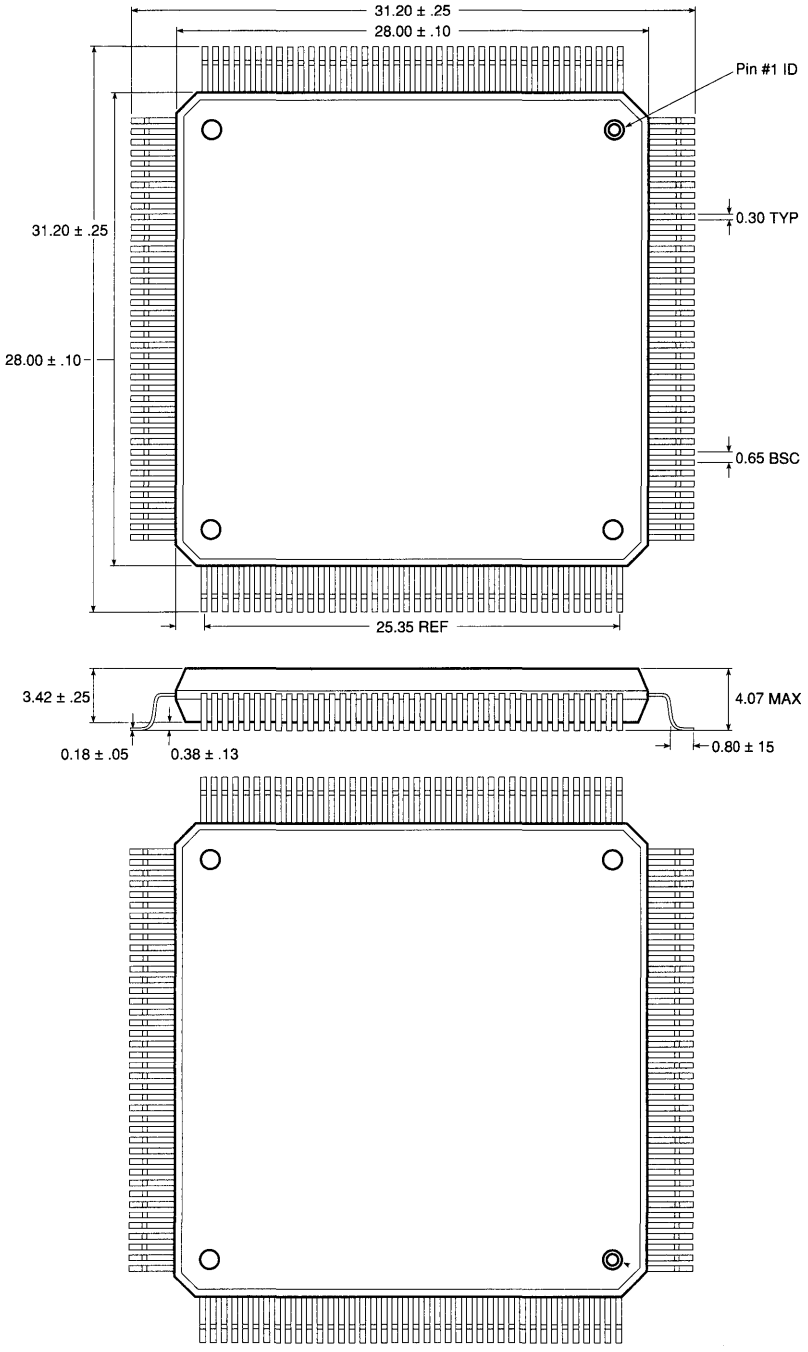
Plastic Quad Flatpack (continued)

160-Pin PQFP (ACT 2)
Dimensions in millimeters



Plastic Quad Flatpack (continued)

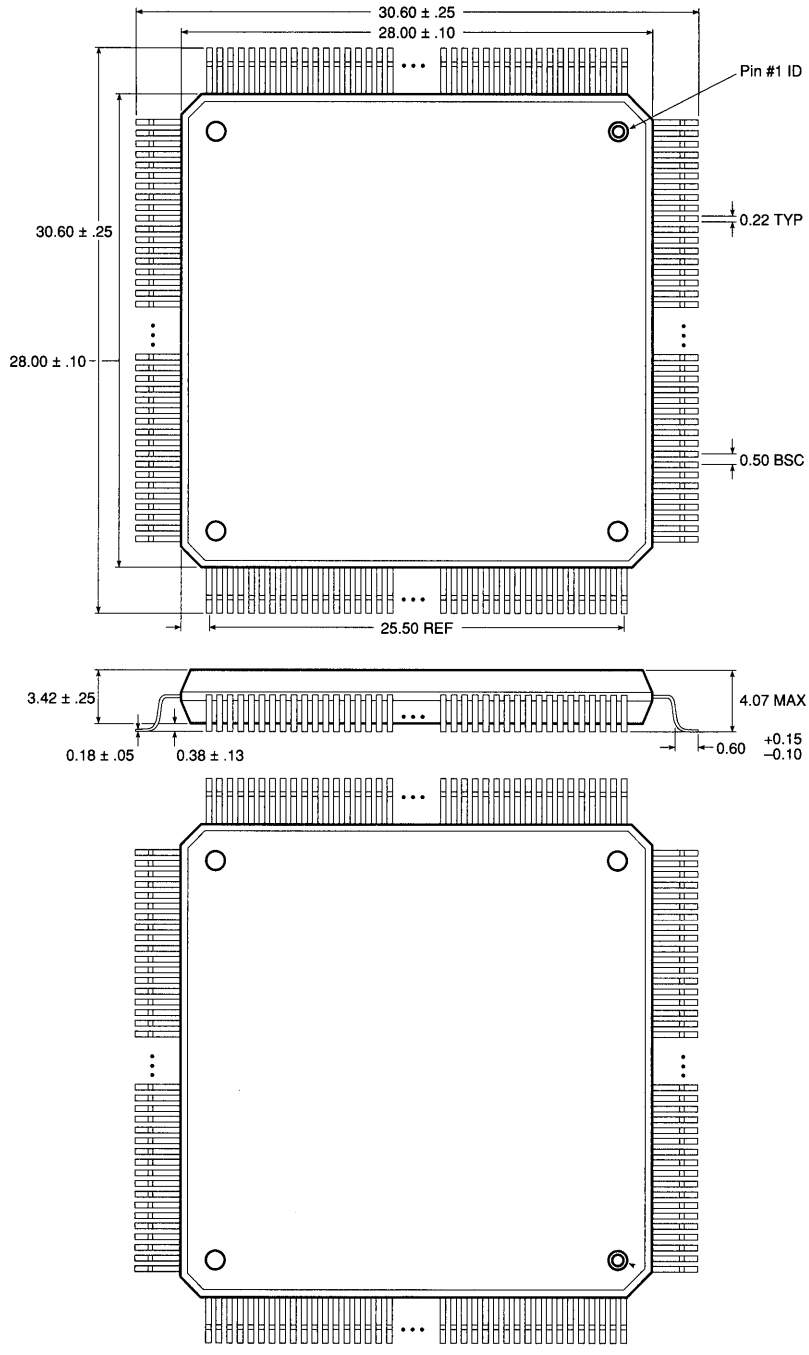
160-Pin PQFP (ACT 3)
Dimensions in millimeters



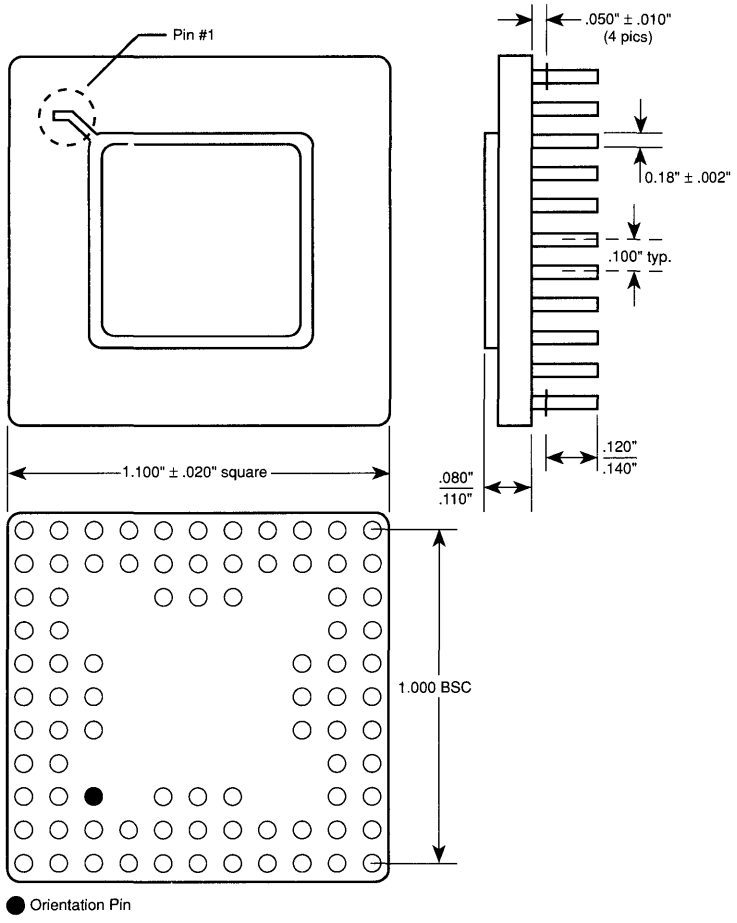
Plastic Quad Flatpack (continued)

208-Pin PQFP

Dimensions in millimeters

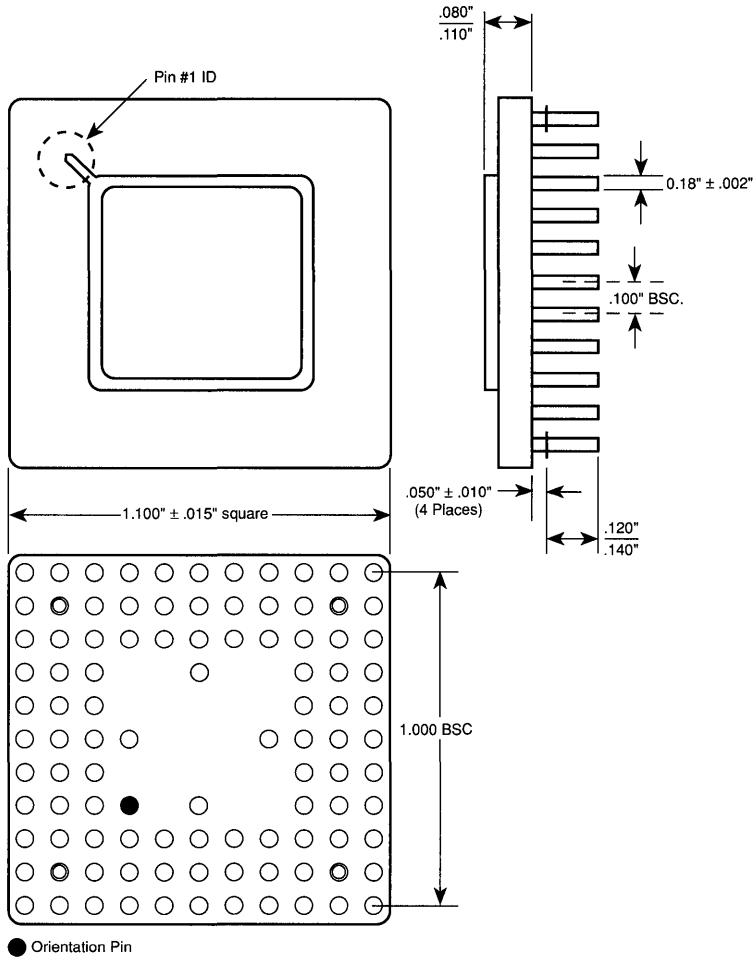


Ceramic Pin Grid Array
84-Pin CPGA



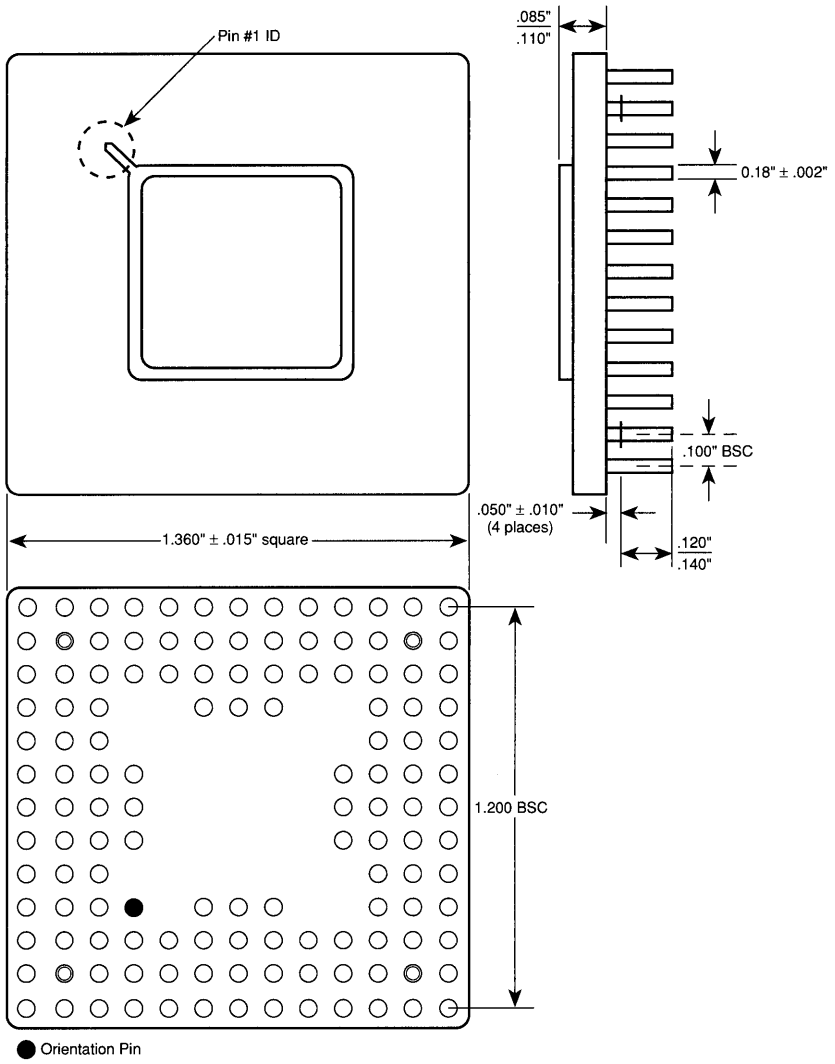
Ceramic Pin Grid Array (continued)

100-Pin CPGA



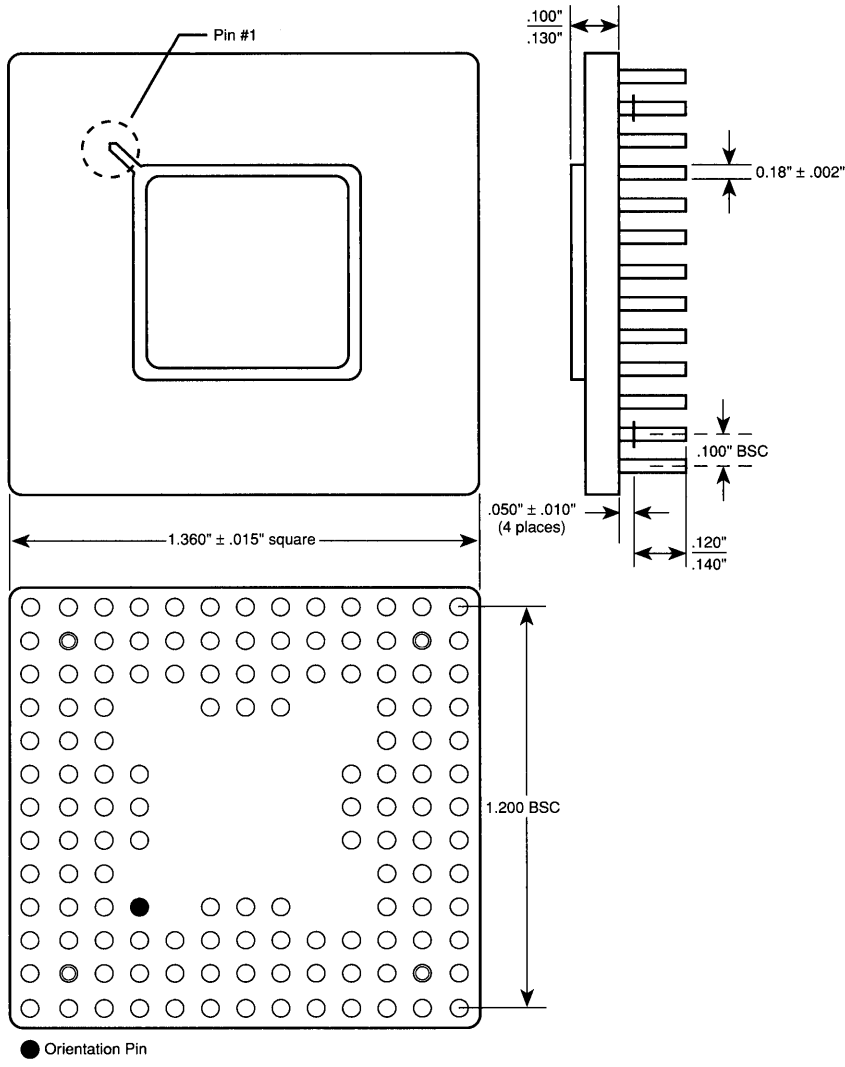
Ceramic Pin Grid Array (continued)

132-Pin CPGA



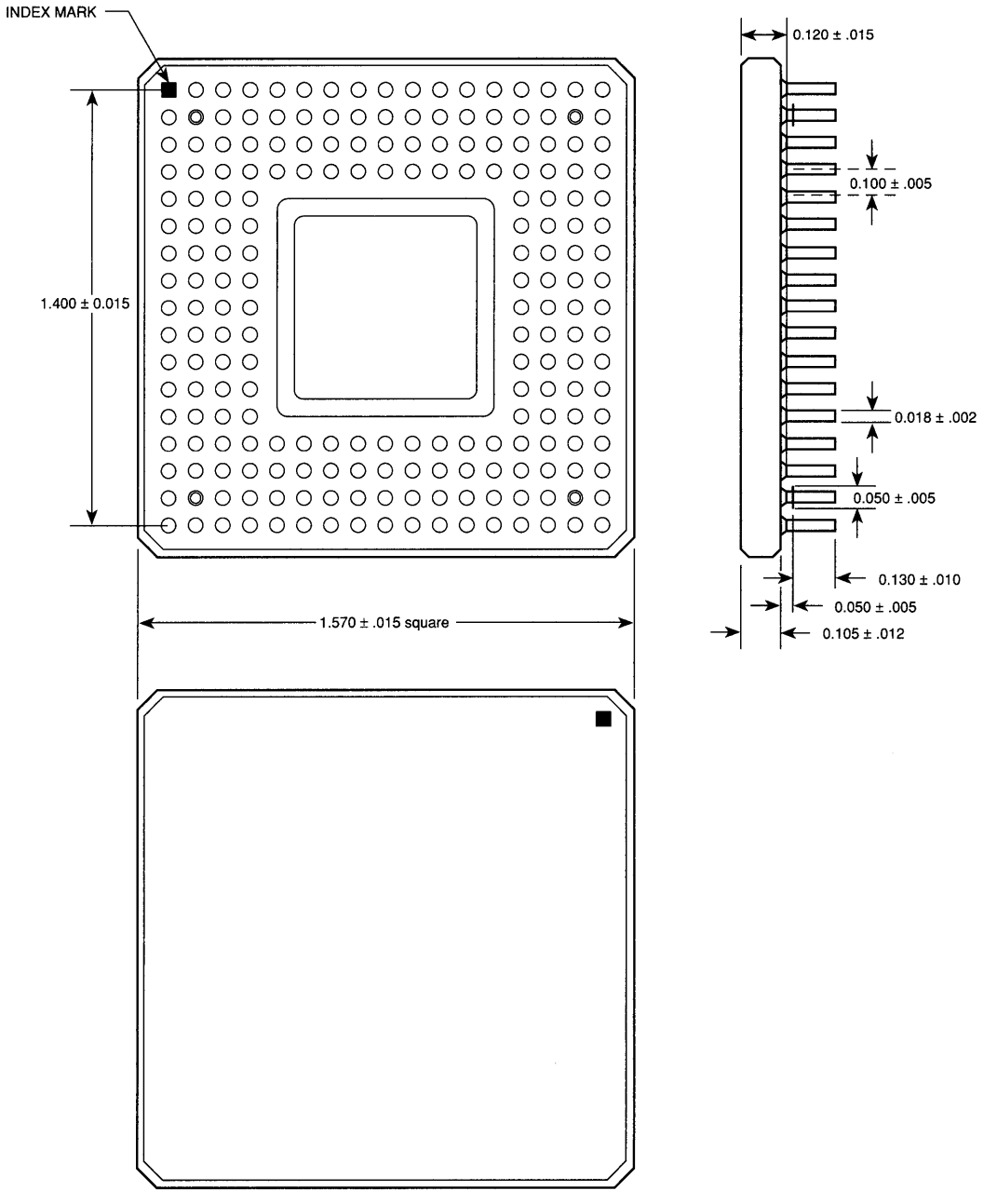
Ceramic Pin Grid Array (continued)

133-Pin CPGA



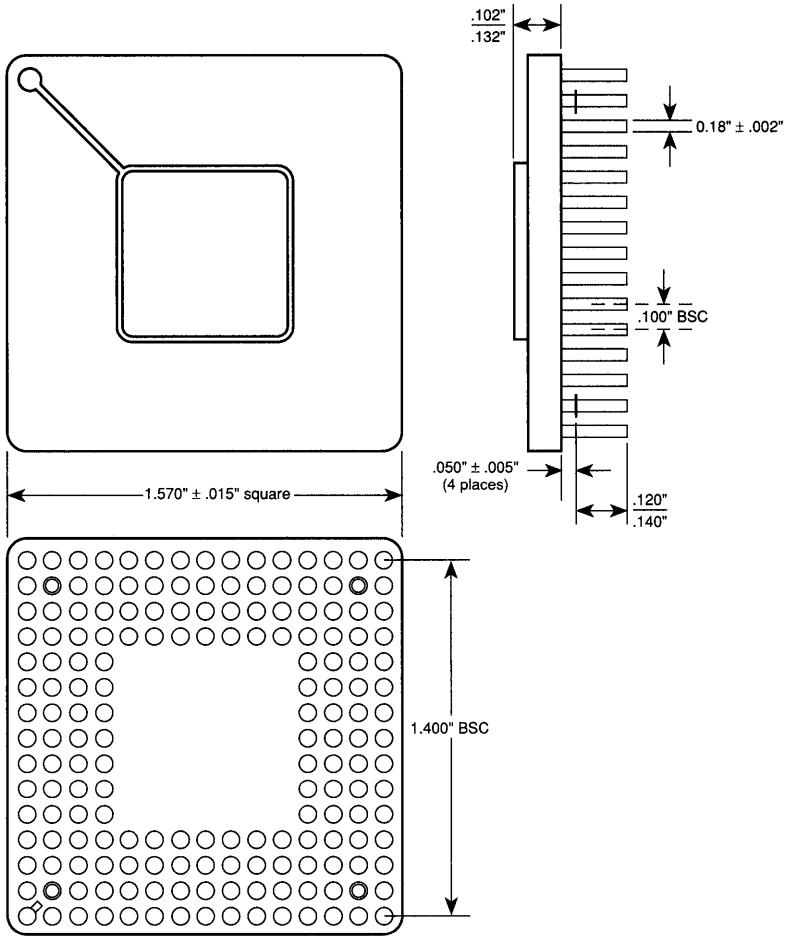
Ceramic Pin Grid Array (continued)

175-Pin CPGA



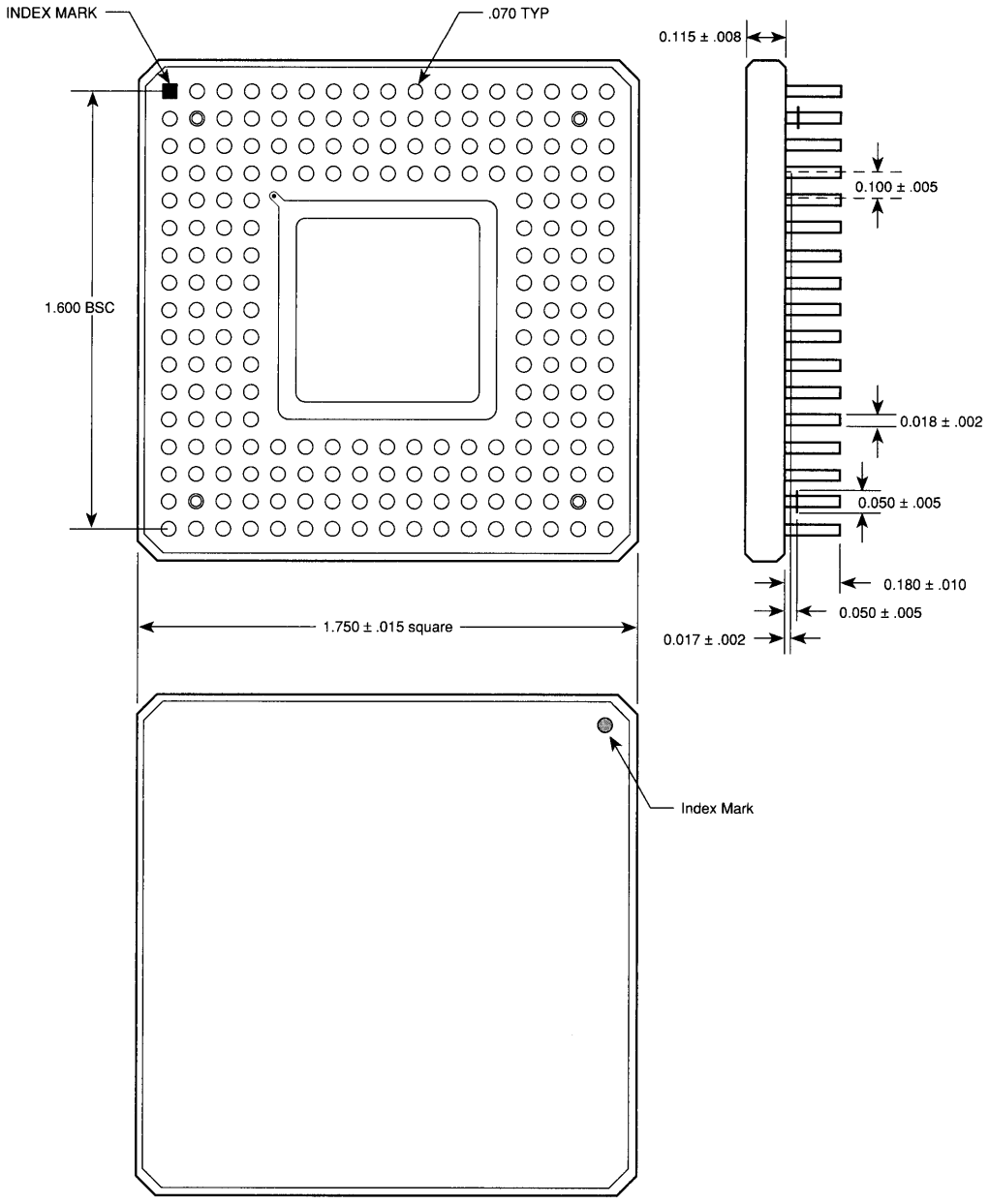
Ceramic Pin Grid Array (continued)

176-Pin CPGA



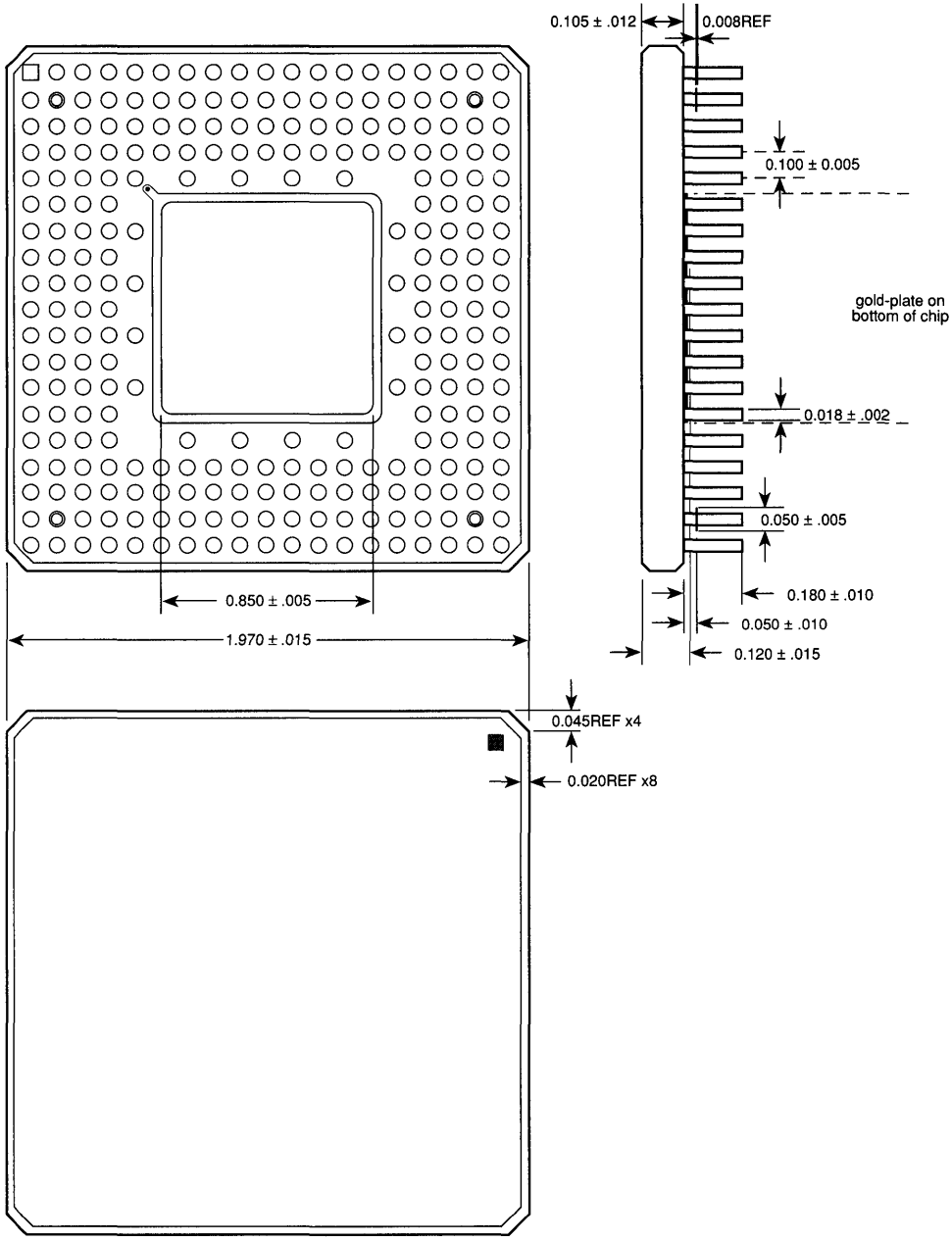
Ceramic Pin Grid Array (continued)

207-Pin CPGA



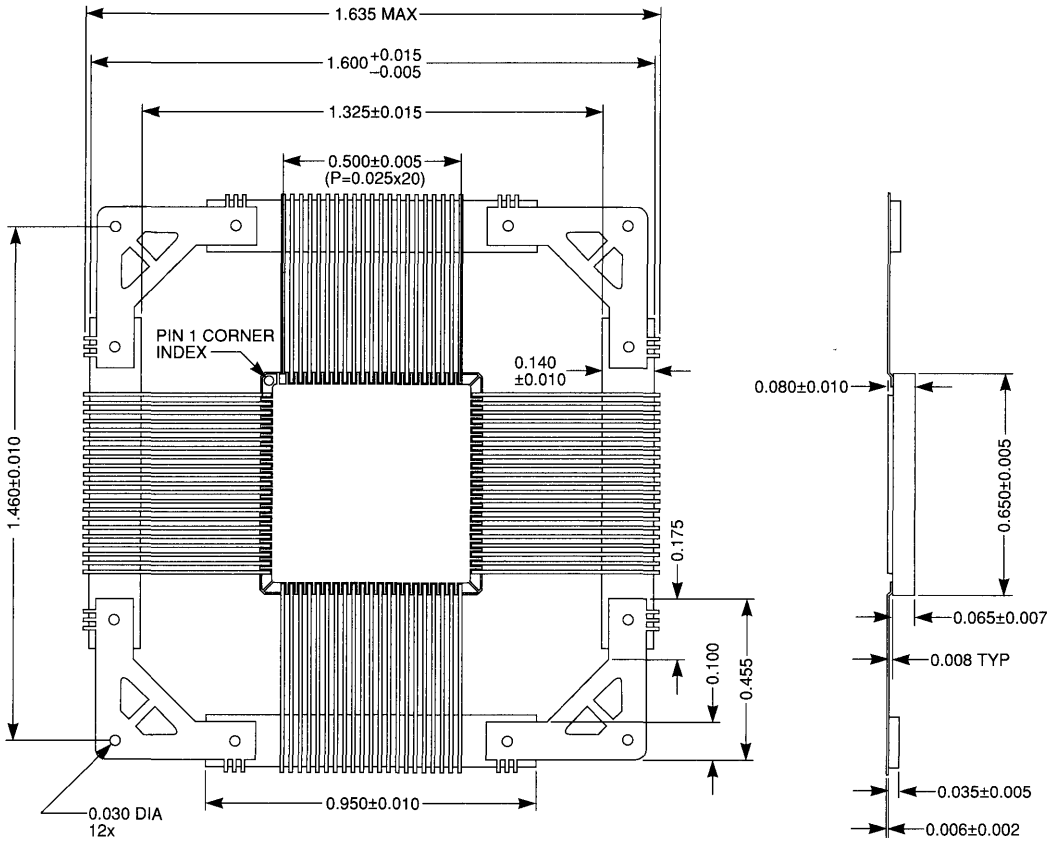
Ceramic Pin Grid Array (continued)

257-Pin CPGA



Ceramic Quad Flatpack

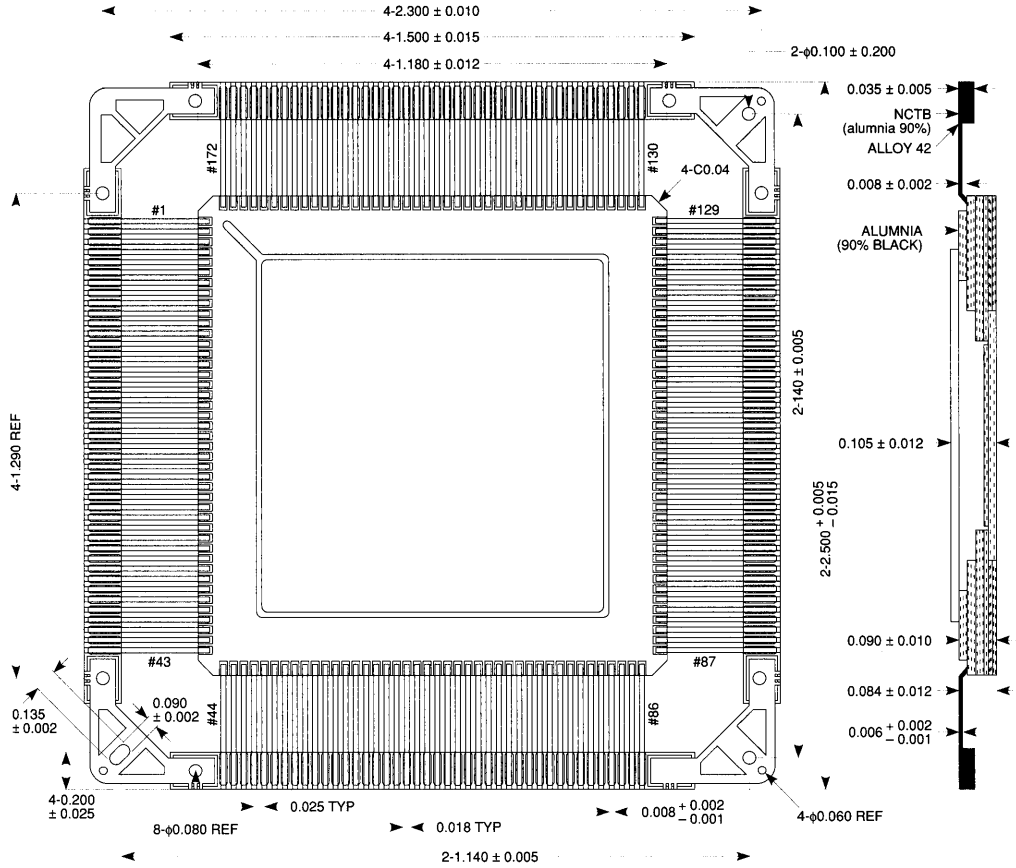
84-Pin CQFP



Ceramic Quad Flatpack (continued)

172-Pin CQFP

Dimensions in inches



Socket Recommendation for Actel FPGA Packages

Actel has compiled this list of know suppliers for the convenience of our customers. This is simply a list of suppliers that we are aware of rather than a list of recommended sockets, as we have not tested them for reliability. For information on these sockets, contact the manufacturers directly.

Socket Sources for Actel FPGA Packages

Prototype/Production Sockets

	Lead Count	Source	Through-Hole	Source	Surface-Mount
PLCC	44	AMP METHODE	821551-3 213-44-101	AMP METHODE	821979-3 or 822035-3 213-044-602
	68	AMP METHODE	821574-3 213-068-101	AMP METHODE	822029-3 or 822073-3 213-068-602
	84	AMP METHODE	821573-3 213-084-101	AMP METHODE	821808-1 (high profile) 213-084-602
PQFP	100		N/A	YAMAICHI	IC149-100-014-S5
	144	AMP	822114-3 or 822115-3	AMP	LMMC prototype
	(ACT 2) 160	AMP	822114-4 or 822115-4	YAMAICHI	IC149-160-023-S5
	(ACT 3) 160		822114-4 or 822115-4	AMP	P93-1970-075
	208		N/A	AMP	P93-1970-076
VQFP	80		N/A	YAMAICHI	IC198-080-2001
PGA (11x11)	85	MILL-MAX McKENZE	510-91-085-11-041 PGA-85H-012B-1-1107		N/A N/A
PGA (11x11)	101	McKENZE	PGA-101M-012B-1-11B5		N/A
PGA (13x13)	133	McKENZE	PGA-133H-003B-1-13GO		N/A
PGA (15x15)	175/176	MILL-MAX McKENZE	510-91-176-15-061 PGA-177M-003B-1-1552		N/A N/A
PGA (17x17)	207	AMP	916227-6		N/A
PGA (19x19)	257	AMP	916229		N/A

Zero Insertion Sockets

	Lead Count	Source	Through-Hole
CQFP	84	WELLS	619-1000311-001
	172	ENPLAS	OTQ-172(196)-0.635-02
PQFP	100	YAMAICHI	IC51-1004-814-2
	144	YAMAICHI	IC51-1014-KS10418
	160	YAMAICHI	IC51-1604-845-1
	208	ENPLAS	OTQ-208-0.5-01
PGA	84	YAMAICHI	NP35-112-G4-BF85
	101	YAMAICHI	NP89-12110-G4-BF101
	132/133	NEPENTHE	NEP5-132-RS1311
	175/176	YAMAICHI	NP89-22508-G4-BF177
	207	AMP	55287-2
	257	AMP	55289-1
PLCC	44	YAMAICHI	IC51-0444-400
	68	YAMAICHI	IC51-0684-390-1
	84	YAMAICHI	IC51-0844-401-1
VQFP	80	YAMAICHI	IC51-0804-795

CONTACTS:

AMP	(408) 725-4914
ENPLAS	(415) 572-1683
METHODE	(408) 262-3812
MIL-MAX	(516) 922-6000
McKENZE	(510) 651-2700
NEPENTHE	(415) 856-9332
WELLS	(408) 559-8118
YAMAICHI	(408) 452-0797

PQFP Handling Instructions

Humidity Control

Follow the instructions on the packaging bag to keep devices dry prior to board assembly.

Handling Trays

The tray stack should be kept fastened with straps during transportation to prevent devices from coming out of their sockets and bending their leads. If a single loaded tray is to be moved, an empty tray should be placed over it as a cover and both trays should be taped to secure the devices in position (see Figure 1).

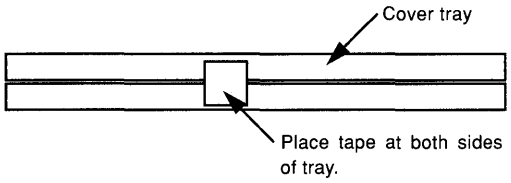


Figure 1. Temporary Tray Holding

Device Programming

Always use the vacuum wand provided to transfer devices.

Removing Devices from Tray

1. Position the vacuum needle at the proper angle so that your index finger can rest on the vacuum switch (see Figure 2).

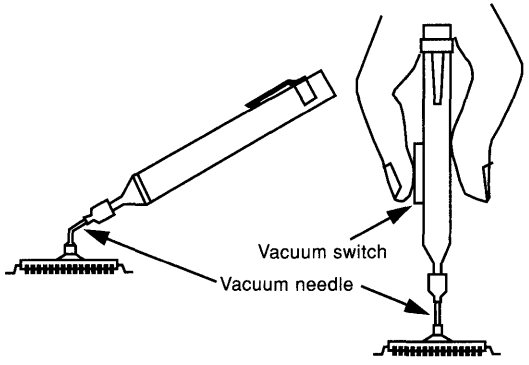


Figure 2. Vacuum Wand

2. Slightly depress the vacuum switch and then place the vacuum cup on the surface of the device.
3. Release the vacuum switch while holding the cup down to generate the vacuum that causes the cup to adhere to the device body.
4. Make certain to transfer a device within 5 to 8 seconds.

Loading the Device into the Programming Socket

1. Carefully align the device with the socket and gently lower the device into the socket (see Figure 3).

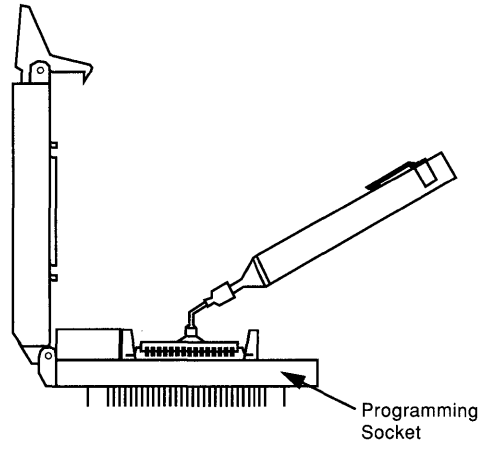


Figure 3. Inserting the Device in the Socket

2. Do not force the device into the socket; otherwise, the corner leads may be bent if the alignment is out.
3. Release the vacuum by depressing the vacuum switch. The device will be self-aligned to the socket.

Note: To completely release the vacuum, the switch should be pressed down further than when pressed to generate the vacuum. Otherwise, a partial vacuum will pull the device back and may cause bent leads.

4. Close the lid. The device is ready for programming.

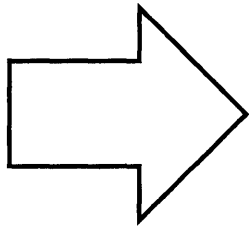
Removing the Device from the Programming Socket

When the programming of the device is complete, use the vacuum wand to remove the device from the socket and to place it in a tray labeled "programmed devices."

Cleaning the Vacuum Wand

The suction cup of the vacuum wand should be cleaned periodically using IPA to minimize possible leakage in vacuum.

Testing and Reliability



Testing and Programming Actel Field Programmable Gate Arrays (FPGAs)

Introduction

Testing has long been a struggle for users of masked gate arrays. To avoid board level, system level, or even possible field failures, the system designer must extend great effort in developing test vectors for gate array designs. Even after the vectors are developed, fault coverage for typical designs may be only about 70 percent, with 95 percent coverage about the best possible. With a 70 percent fault coverage, typical masked gate array designs are likely to have 2 to 5 percent devices defective[1].

In general, field programmable logic devices have allowed the user to avoid the need to develop test vectors. These devices allow the tests to be performed by the semiconductor vendor prior to programming. However, most one-time programmable logic devices have not yet achieved the functional quality levels of other semiconductor devices because they don't allow the chip manufacturer to access and test all internal gates. Early one-time programmable devices had poor test coverage, and users were often disappointed to see functional failure rates of more than 10 percent on parts that had passed programming. Over time, on-chip test circuits and testing techniques have greatly improved and now one-time programmable devices have functional defect rates in the range of 0.1 to 1 percent[2]. Although this failure rate is low for individual chips, putting 10 such chips on a single board can still mean a board failure rate of 5 to 10 percent.

Reprogrammable logic devices that use EPROM, EEPROM, or RAM technology however, have improved functional quality levels to nearly 100 percent. Since the semiconductor manufacturer can program these chips to any desired configuration, it is possible to test all internal gates. This can result in functional failure rates equivalent to most other semiconductor devices.

The Actel FPGA Product Family

There are three Actel FPGA product families. The ACT™1 family offers 1200 (A1010) and 2000 (A1020) gate products. The ACT 2 family consists of three products (A1225, A1240, and A1280) with 2500, 4000, and 8000 gate array equivalent gates. This family offers improved performance and number of I/O's compared to ACT 1. The ACT 3 family contains five products (A1415A, A1425A, A1440A, A1460A, and A14100A) with 1500, 2500, 4000, 6000, and 10,000 equivalent gates. The ACT 3 products offer the highest performance and number of I/Os of the three families.

Testability of Actel FPGAs

Although Actel's FPGA Family arrays use a one-time programmable technology, the device's unique architecture permits a degree of testability comparable to reprogrammable devices. Special test modes allow functional testing of unprogrammed devices at essentially 100 percent fault coverage. This testability is independent of the large number of equivalent gates in the A1010 (1200 gates) through the A14100 (10,000 gates). To show how this accomplished, we will first review the architecture of the Actel FPGAs and describe how they are programmed.

Architecture

The basic building block of all Actel FPGAs is the logic module. Each logic module is programmable and capable of implementing all two-input logic functions, most three-input functions, and many other functions up to eight inputs. With an architecture similar to a channeled gate array, logic modules are organized in rows and columns across the chip (Figure 1). Adjacent to each row of logic modules are routing channels. Horizontal routing channels are shown in the figure, but vertical channels also run through the logic modules. These are used to configure a logic module and connect inputs and outputs of logic modules together to implement a design. Surrounding the array of logic modules and routing channels are I/O buffers and test circuits.

Within the routing channels are programmable antifuse (PLICE™) elements. The antifuse is normally open and is programmed to form an electrical connection between routing elements. An antifuse that connects a horizontal routing track to a vertical track is called a cross antifuse. An example of a logic module interconnection (or a net) is shown in Figure 2. Here the output from Module 3 is connected to a horizontal routing track by programming a cross antifuse. Another cross antifuse is programmed to connect an input to Module 4. In a similar manner, the output of Module 3 is connected to the input of Module 2. Notice that not all horizontal tracks are continuous across the chip. Often tracks are broken into a series of smaller tracks called "segments." Segments are useful because it is often desirable to connect logic modules that are close to each other, and using a full horizontal track would waste routing resources and slow down circuit performance. Sometimes, however, it is necessary to connect two segments together to form a longer segment. This can be done by programming a special type of antifuse referred to as a horizontal antifuse. As an example, the output of Module 3 is also connected to the input of Module 1 by programming two cross antifuses and one horizontal antifuse. Vertical antifuses are used to connect two vertical segments (not shown).

A more detailed example of the Actel FPGA architecture is shown in Figure 3. Six logic modules (two rows, three columns) are shown. Between the two rows are six horizontal tracks. Down each column are five vertical tracks. Note that the products actually have 25 to 36 horizontal and 13 to 15 vertical tracks. The circles at the intersection of vertical and horizontal tracks represent cross antifuses. There are also circles at certain points

4

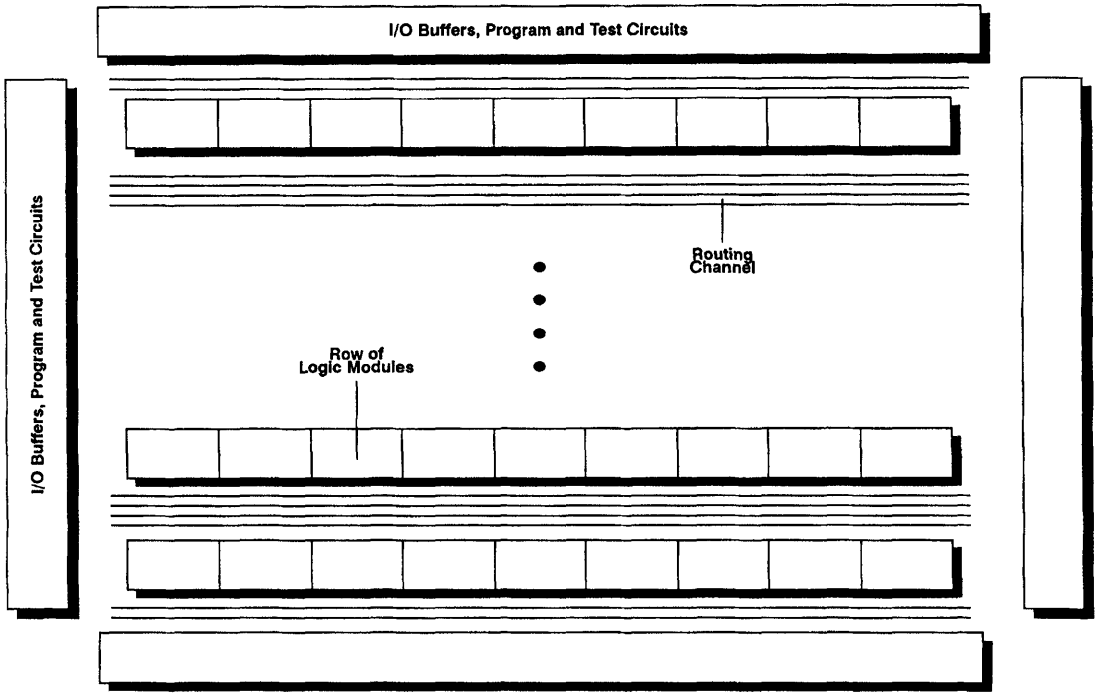


Figure 1. Architecture

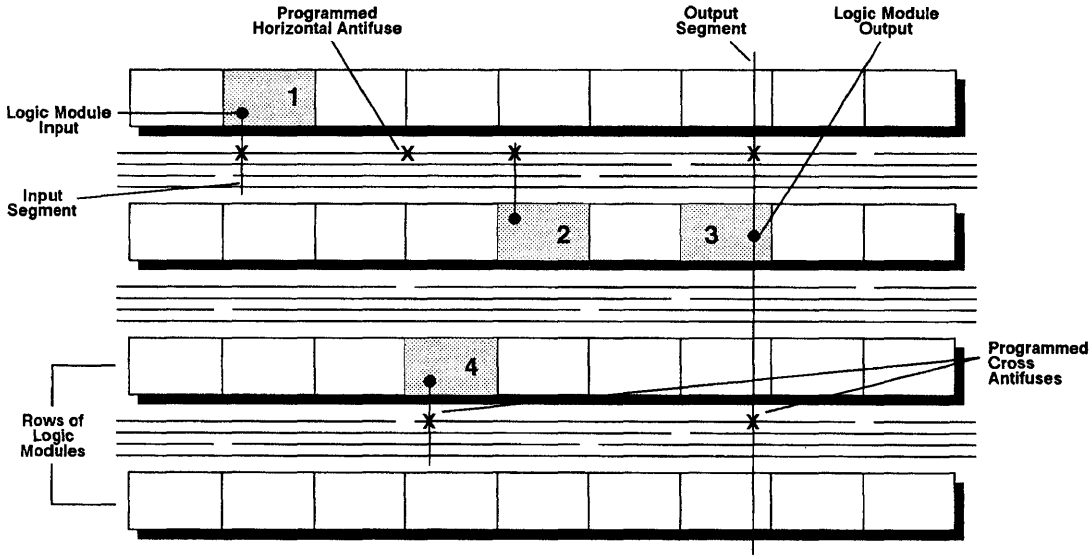


Figure 2. Routing

on the horizontal tracks that are horizontal antifuses. No vertical antifuses are shown. Notice the transistors that connect both horizontal and vertical tracks. By turning on selected transistors, various horizontal or vertical tracks can be connected together even though an antifuse has not been programmed. This ability to connect tracks in unprogrammed devices is used extensively during antifuse programming and is one of the key elements responsible for the excellent testability of the Actel FPGAs.

Logic configuration of modules is interesting because there are no dedicated antifuses in the module to accomplish this. Instead, the inputs (and outputs) of logic modules extend into the cross antifuse array. Each logic module has eight to ten inputs and one output. By programming appropriate antifuses, an input can be connected to a dedicated horizontal ground line, a Vcc line, or a horizontal routing track. The logic module implements a particular logic function by tying appropriate unused inputs to ground or Vcc.

Programming

The following discussions about programming and testing modes are specific to the ACT 1 family of FPGAs. However, basic concepts also apply to the ACT 2 and ACT 3 families.

An antifuse is programmed by applying a sufficiently high voltage across it. This voltage is referred to as Vpp. To access an antifuse deep inside the chip, it is necessary to create electrical paths from Vpp and ground to the antifuse. This is done by turning on the appropriate horizontal and vertical pass transistors (in normal chip operation, these transistors are always off). The transistors are turned on by applying Vpp to their gates. In Figure 4, we see an example of programming a typical cross antifuse. Vpp is applied to a vertical track at the top of the chip and ground is applied to a horizontal track on the right side. The design of the A1010/A1020 actually allows Vpp or ground to be applied from the top, bottom, left, or right as is appropriate to best access a particular antifuse. Notice that Vpp is also applied to the gates of

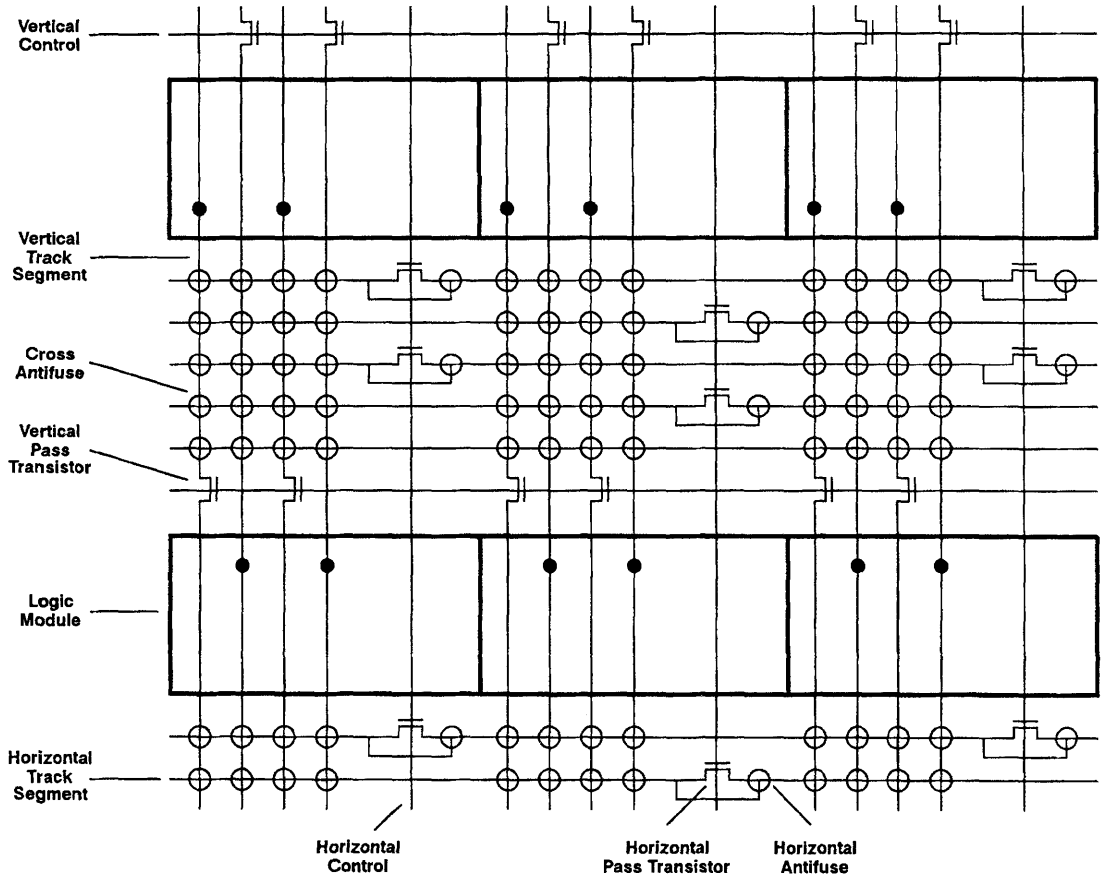


Figure 3. Programmable Interconnect

the horizontal and vertical pass transistors on the tracks accessing the cross antifuse. The circled cross antifuse now has Vpp applied to it on one side and ground on the other. This voltage breaks down the antifuse's dielectric and creates an electrical connection between the horizontal and vertical routing tracks.

There is one other important consideration when programming an antifuse. Notice that the cross antifuses in the same vertical track as the antifuse to be programmed also have Vpp applied to them on one side. This is true until the track is broken by a vertical pass transistor below it that is turned off. However, the potential on the other side of the antifuses is not being driven. Should this potential be at ground, the other cross antifuses on the vertical segment could be accidentally programmed.

The same logic applies to other antifuses on the same horizontal track. Here, one side of the antifuse is being driven to ground and if the other side were at Vpp, extra antifuses could program.This problem is solved by first applying what is referred to as a "precharge cycle." During the precharge cycle, all horizontal and vertical tracks are charged to Vpp/2. As a result, there is no voltage across the antifuses. The appropriate vertical track is then driven to Vpp and a horizontal track to ground (Figure 5). At this point, other antifuses on the vertical track have a potential of Vpp/2 across them (Vpp on one side and Vpp/2 on the other). This Vpp/2 voltage is not sufficient to program the antifuses. Other antifuses on the same horizontal track also have Vpp/2 across them (Vpp/2 on one side and ground on the other). Most other antifuses in the chip still have Vpp/2 on both sides and will not program.

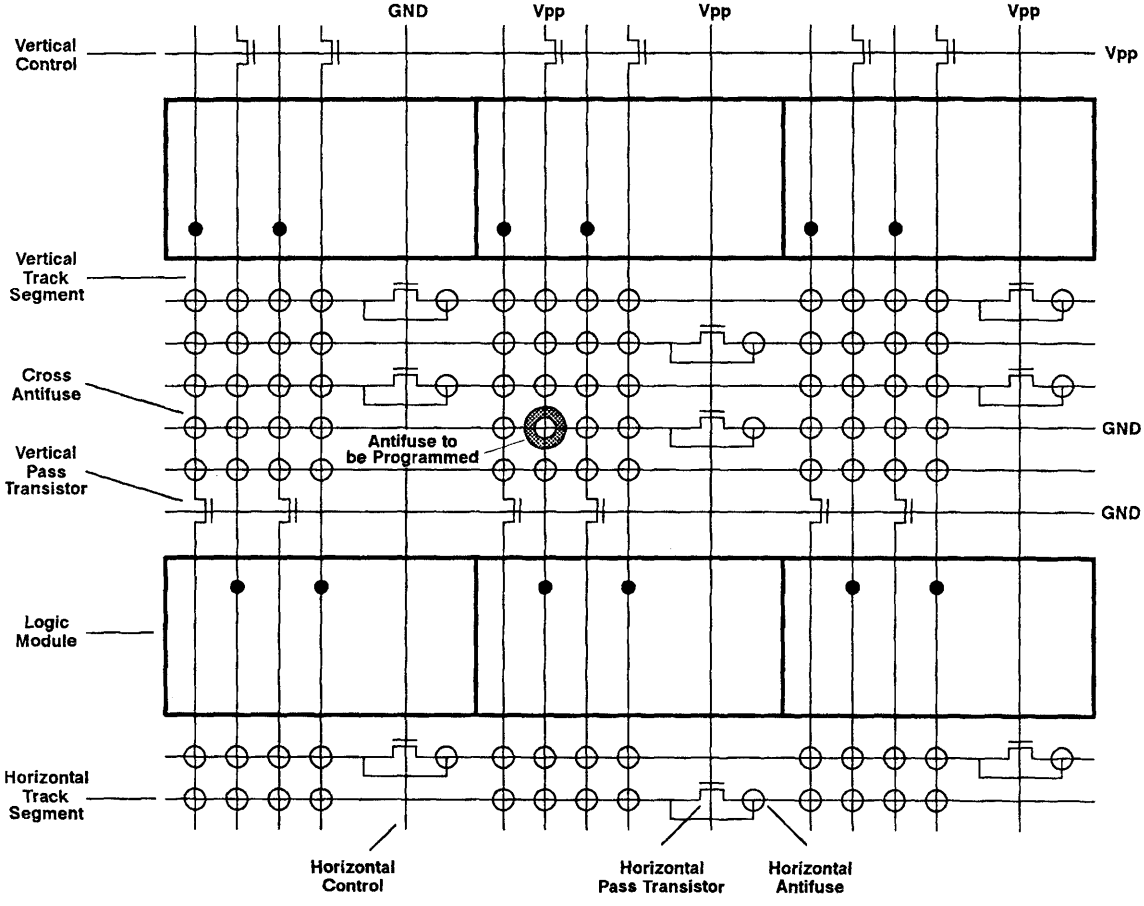


Figure 4. Programmable Interconnect

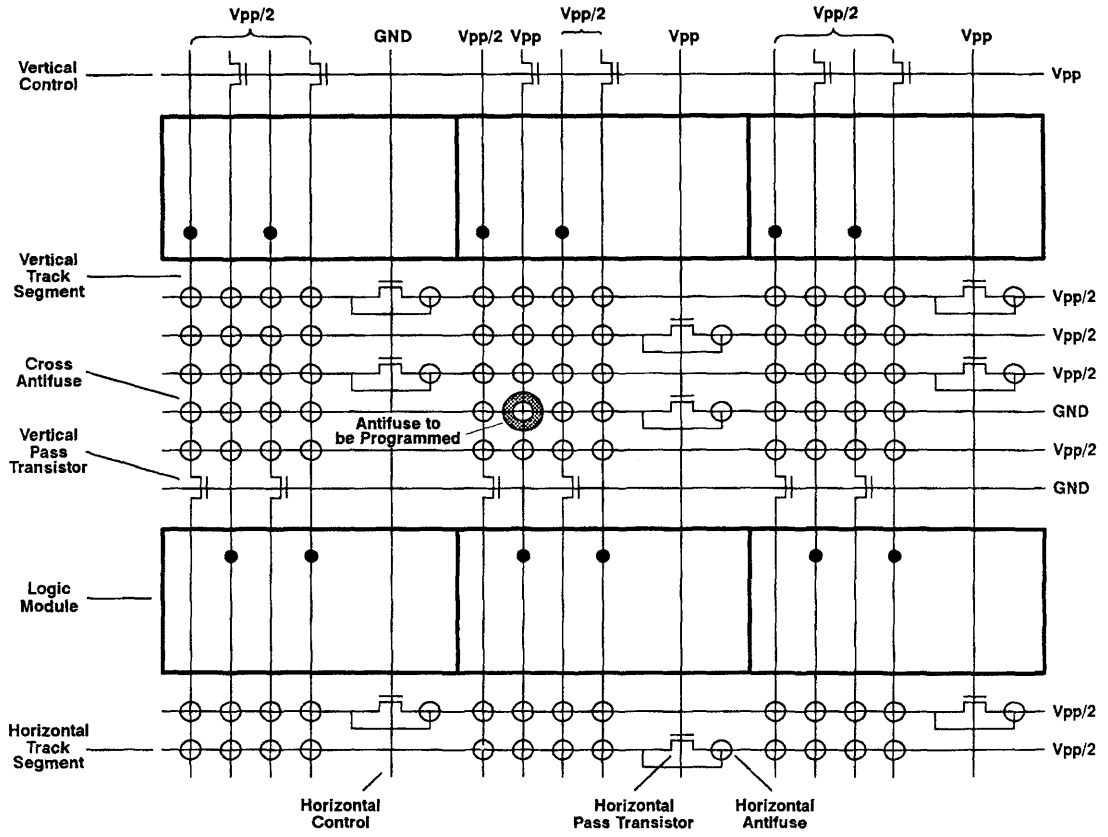


Figure 5. Programmable Interconnect

Programming Algorithm

In concept, the Actel FPGAs are programmed in a manner very similar to many other programmable logic devices as well as memories such as EPROMs. The programming algorithm consists of the following steps:

1. An addressing sequence to select the antifuse to be programmed.
2. A programming sequence where Vpp is applied in pulses until the antifuse programs.
3. A soak or "overprogram" step to ensure uniform, low antifuse resistance.
4. A verify step to make sure the antifuse was properly programmed.

Unlike a memory where an antifuse is addressed by applying a parallel address, the FPGAs are addressed in a serial manner by using the special DCLK (Data Clock) and SDI (Serial Data In) pins. There is a large shift register that travels around the periphery of the chip. Bits in this shift register can be used to drive tracks to ground, Vcc, Vpp, or float. It is also possible to sense the level on the track (high or low) and load this information into the shift register. By shifting in the correct address, any antifuse can be selected for programming. The shift register also plays a key role in testing the chip. This will be discussed later.

The programming sequence starts with the precharge pulse where Vpp/2 is applied to the Vpp pin. This is followed by a programming pulse where Vpp is applied to the pin. Following the program pulse, the voltage on the Vpp pin is returned to a nominal value (about 6 V). See Figure 6 for a typical Vpp waveform. The precharge/program pulse sequence is repeated until either the selected antifuse programs or a maximum number of pulses is exceeded (in which case the antifuse is considered nonprogrammable and the device is rejected).

Confirmation that an antifuse has programmed is determined by monitoring the current on the Vpp pin. This current is very low (typically < 10 µa) until an antifuse programs. Once an antifuse is programmed, an electrical connection is made between Vpp and ground in which case currents in the range of 3 to 15 mA may be observed on Vpp. Once this current is observed, the antifuse is considered programmed and enters the soak or "overprogram" cycle. Here, extra pulses are applied to the antifuse to achieve minimum antifuse resistance.

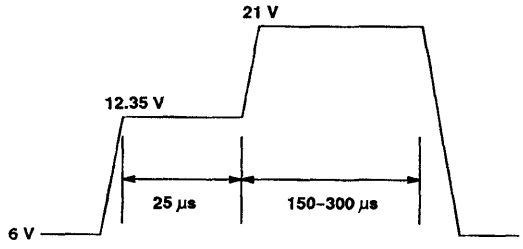


Figure 6. Vpp Waveform

ACT 1 Programming Algorithm

Current Parameters		
V Program	=	21 V
V Precharge	=	12.35 V
V Verify	=	6.0 V
t Program	=	150–300 µs
t Precharge	=	25 µs
I Threshold	=	~2.5 mA (to detect programmed antifuse)
I Max	=	15 mA (clamp current)
# Soak	=	30–800 pulses
Maxpulses	=	60,000

Test Modes of Actel FPGAs

The unique architecture described above allows outstanding testability of unprogrammed devices at the factory. Details of the various test modes available are as follows:

1. The shift register circling the periphery of the chip can be both downloaded and uploaded. This allows the use of various test patterns to ensure that the shift register is fully functional.
2. All vertical and horizontal tracks can be tested for continuity and shorts. There are several ways to implement these tests. One way of doing continuity testing is to precharge the array, turn on all vertical or horizontal pass transistors on a track, drive the track low from one side of the chip, and read a low on the other side. Shorts can be detected by driving every other track low after precharge and reading back on the other side. Note that these tests also confirm that the vertical and horizontal pass transistors will turn on.
3. It is important for programming to make sure that all tracks can hold the precharge level. By charging a track, floating it, and waiting a predetermined amount of time, the track can be read back and confirmed to be still high.
4. Leakage of vertical and horizontal pass transistors can be tested for by driving one side of a track to a voltage via the Vpp pin and grounding the other side. All pass transistors except the one being tested are turned on. If excess current is detected on the Vpp pin, the pass transistor is considered defective.
5. There are one or two dedicated clock buffers that travel across all horizontal channels. This buffer can be tested by driving with the clock pin and reading for the proper levels at the sides of the array.
6. There are two special pins referred to as Probe A and Probe B (Actionprobes™). By entering a test mode, the shift register can be made to address the internal output of any logic module. This output is then directed to one of two dedicated vertical tracks, which in turn can be observed externally on the Probe A or B pins. This ability to observe internal signals (even on unprogrammed parts) allows Actel to perform a large number of functional tests. The first such test is the Input Buffer Test. Input buffers on all I/O pins can be tested for functionality by driving at the input pad and reading the internal I/O output node through the probe pins.
7. Test modes exist to drive all output buffers low, high, or tristate. This allows testing of Vol, Voh, Iol, Ioh, and leakage on all I/Os.
8. One of the key tests is the ability to functionally test all internal logic modules. By turning on various vertical pass transistors and driving from the top or bottom of the chip, any of the eight to ten module inputs can be forced to a high or low. The output of the module can then be read through the Actionprobe pins. The logic module test allows 100 percent fault coverage of each module. In addition, the architecture allows modules to be tested in parallel for reduced test time.
9. Actel FPGAs have one or two dedicated columns on the chip that are transparent to the user and used by the factory for speed selection. These columns are referred to as the "Binning Circuit." Modules in the columns are connected to each other by programming antifuses. The speed of the completed test circuit can then be tested. The Binning Circuit allows the separation of units into different speed categories. It also allows the speed distribution within each category to be minimized.
10. There are several tests to confirm that the programming circuitry is working. The first such test is a basic junction stress/leakage test. The program mode is enabled and Vpp voltage plus a guardband is applied to the Vpp pin. All

vertical and horizontal tracks are driven to Vpp; thus, no voltage is applied across the antifuses. The Ipp current is then measured. If it exceeds its normal value, the device is rejected.

11. There is a test to ensure that all antifuses are not programmed. This is referred to as the Antifuse Shorts Test (or Blank Test). The array is precharged and then the vertical tracks are driven to ground. The horizontal tracks are then read to confirm that they are still high (a programmed or leaky antifuse would drive a horizontal track low). The test is repeated by driving horizontal tracks low and reading vertical tracks.
12. The functionality of the programming circuitry can be verified by programming various extra antifuses on the chip that are transparent to the user. Some of these antifuses were already described earlier when the Binning Circuit was discussed. Actel FPGAs also have a Silicon Signature™. In the ACT 1 family, the Silicon Signature consists of four words of data, each word 23 bits in length. The first word is hard wired (no antifuses) and contains a manufacturer ID number as well as a device ID number. These numbers can be read by a programmer and the proper programming algorithm would be automatically selected. The other words contain antifuses and are programmable. Actel is currently using bits in these words to store information such as the chip's run number and wafer number. Thus, each Actel FPGA has traceability down to the wafer level. By programming this information, the functionality of the programming circuitry is also tested. Actel software also allows the user to program a design ID and checksum into the Silicon Signature. By later reading this back, the user can verify that the chip is correctly programmed to a given design.
13. The most important antifuse test is the stress test. When this test is enabled, a voltage applied to the Vpp pin can be applied across all antifuses on the chip (the other side is grounded). The voltage applied is the precharge voltage plus a significant guardband. After the voltage is applied, the Antifuse Shorts Test is again used to make sure no antifuses have programmed. The antifuse stress test is effective at catching antifuse defects. Becasue the reliability of the antifuse is much more voltage dependent than it is temperature dependent, this test is also an effective antifuse infant mortality screen. See the Actel Reliability Report for details.

Burn-In of Actel FPGAs

As mentioned above, Actel has found that antifuse infant mortality failures can be effectively screened out during electrical testing, and it is thus unnecessary to do any kind of burn-in for standard commercial production units to screen out antifuse infant mortality failures. However, burn-in is still an effective screen for standard CMOS infant mortality failure mechanisms, and it is required for all military 883D products. MIL-883D Method 1005 allows several types of burn-in screens. These can be divided into two categories: steady-state (static) and dynamic. Static burn-in applies DC voltage levels to the pins of the device under test. The device may or may not be powered up. Dynamic burn-in applies AC signals to device inputs. These signals are selected so that the device receives internal and external stresses similar to what it may see in a typical application.

Static burn-in is by far the simplest to implement. By choosing appropriate biasing conditions and load resistors, it is possible to design a single burn-in circuit that could be used for both unprogrammed and programmed devices. It would not matter what pattern is programmed into the device. Static burn-in can be an effective screen for some types of failure modes, particularly those that may happen at device inputs or outputs (such as screening for mobile ionic contamination). It is not, however, very effective at stressing internal device circuits. Many internal nodes may be biased at ground without receiving any voltage or current stress. Signal lines will not toggle, and it may not be possible to screen failure modes such as metal electromigration.

A properly designed dynamic burn-in can effectively stress inputs, outputs, and internal circuits. However, dynamic burn-in of ASIC products can be very expensive because customer-specific burn-in circuits and burn-in boards must be designed and built to properly stress each design implemented in the ASIC. This results in large NRE costs and long lead times to design and build these boards. From the standpoint of burn-in, a programmed FPGA is essentially the same as a mask-programmed ASIC, and it would require similar custom burn-in circuits to do a dynamic burn-in. However, Actel has been able to use the testability features of its FPGA products to allow effective dynamic burn-in of unprogrammed devices. This dynamic burn-in allows us to stress circuits in a way that static burn-in would be unable to duplicate.

During burn-in of unprogrammed units, test commands are serially shifted into each device using the SDI pin and clocked using the DCLK pin. There are three test modes shifted into each device. The first test stresses each cross antifuse with a voltage of Vpp–2 V (Vpp is normally set at 7.5 V so that each antifuse gets 5.5 V across it). This voltage is applied to all vertical tracks while the horizontal tracks are grounded. Once enabled, the stress mode is held for 10 ms.

The second test mode is identical to the first except that the horizontal tracks are driven to Vpp–2 V while the vertical tracks are grounded. Note that both of these modes are similar to the antifuse stress test described earlier (although the stress voltage is lower during burn-in). Not only do these tests stress the antifuses, but they also toggle all routing tracks in the chip to Vpp–2 V and ground. All input and output tracks to the logic modules are also toggled.

The third test drives several I/O pins on the chip to a low state. Prior to this, they are at high impedance state and held at Vcc through pull-up resistors. This test confirms that the burn-in is being properly implemented by looking at these I/O pins to see if they display the proper waveform. It also passes current through each I/O as it toggles low.

Although the chip is unprogrammed, these tests allow us to apply stresses to the inputs, outputs, and internal nodes that are similar

to what a programmed device may see in normal operation. Once burn-in is completed, post burn-in testing as specified by MIL-883D is performed (including PDA) to ensure that fully compliant devices are shipped to the customer.

Conclusion

The description of the Actel FPGA architecture and the numerous test modes attest to the outstanding testability of these devices. All internal logic gates can be tested without programming antifuses other than the few for the Binning Circuit and Silicon Signature. Because Actel FPGAs are one-time programmable, the only item that is not fully tested at the factory is the programmability of all the individual antifuses. However, this is done on the programmer while the units are being programmed. Being able to test all internal gates allows Actel to achieve functional yields superior to other one-time programmable devices and equivalent to reprogrammable parts.

References

1. Henshaw, "User Requirements for Fault Coverage," Wescon Proceedings, 1990, P. 179
2. AMD PAL Device Data Book, 1988, P. 3–106

ACT Family Reliability Report

Actel's field programmable gate arrays (FPGAs) are currently available in three product families—ACT™ 1, ACT 2, and ACT 3. The ACT 1 family consists of the A1010 and A1020, which are 1200- and 2000-gate FPGAs respectively. The ACT 2 family includes the A1225, A1240, and A1280 FPGAs, which offer 2500, 4000, and 8000 gates respectively. The ACT 3 family is the newest family of FPGAs and contains products ranging from 1500 to 10,000 gates (A1415, A1425, A1440, A1460, and A14100). This report will cover the ACT 1 and ACT 2 families; a future addendum will include ACT 3 products.

The programming element for all Actel FPGAs is an Actel-invented PLICE® (Programmable Low-Impedance Circuit Element) antifuse. An antifuse is a device that is normally open and in which an electrical connection is established by applying programming voltage. Although Actel FPGAs are one-time programmable devices, their unique architecture includes complete functional testability.

ACT 1 products were originally manufactured using 2 µm design rules (A1010/A1020). Later they were linearly shrunk in size by 20% to 1.2 µm (A1010A/A1020A). Recently, the family has been shrunk again to 1.0 µm (A1010B/A1020B). ACT 2 products were first introduced with 1.2 µm design rules (A1225/A1240/A1280) and as a result of a 20% linear shrink are now available in 1.0 µm versions (A1225A/A1240A/A1280A). ACT 3 products are manufactured using a 0.8 µm process. In all cases, the technology is a standard double metal, twin well, CMOS process in which three additional masking steps have been added to implement the PLICE antifuse. The main process parameters are shown in Table 1. Because Actel FPGAs are manufactured with a conventional CMOS process, normal CMOS failure modes are observed. However, the addition of the antifuse adds another structure that could affect the device's reliability.

To quantify the reliability of the antifuse, Actel has completed numerous studies. These studies lead to the conclusion that the time-to-failure of the antifuse is substantially more than 40 years under normal operating conditions and that the combined contribution of all antifuses to the gate array product's hard failure rate is less than 10 FITs (Failures-in-Time or 0.001% failures per 1000 hours).

4

Table 1. ACT 1 and ACT 2 Process Description

Dimensions	2.0 µm Process		1.2 µm Process		1.0 µm Process		0.8 µm Process	
	Width (µm)	Space (µm)	Width (µm)	Space (µm)	Width (µm)	Space (µm)	Width (µm)	Space (µm)
N +	4.0	2.0	3.2	1.6	2.4	1.2	1.8	1.4
p +	4.0	2.0	3.2	1.6	2 4	1.2	1.8	1.4
Cell PolySilicon	2.0	3.6	2.1	2.4	1.5	0.9	1.4	1.0
Gate PolySilicon	1.6	2.4	1.6	1.6	1.2	1.2	1.0	1.2
Metal I	4.0	2.0	3.2	1.6	2.2	1.2	1.5	1.3
Metal II	4.8	2.2	3.8	1.8	2.1	1.6	1.5	1.5
Contact	1.8 x 1.8	2.0	1.2 x 1.2	1.8	1.0 x 1.0	1.0	1.0 x 1.0	1.2
Via	2.0 x 2.0	2.0	1.3 x 1.3	1.9	1.0 x 1.0	1.0	1.0 x 1.0	1.2
Thickness	**2.0 µm Process**		**1.2 µm Process**		**1.0 µm Process**		**0.8 µm Process**	
Normal Gate Oxide	25 nm		25 nm		20 nm		18 nm	
High Voltage Gate Oxide	40 nm		40 nm		32.5 nm		35 nm	
Cell PolySilicon	450 nm		450 nm		450 nm		400 nm	
Gate PolySilicon	450 nm		450 nm		450 nm		400 nm	
Metal I	900 nm		900 nm		750 nm		750 nm	
Metal II	1000 nm		1000 nm		1050 nm		1050 nm	
Passivation	1000 nm		1000 nm		1100 nm		1100 nm	
Compositions								
Metal I	Al – Sl (1%) – Cu (5%)		Al – Sl (1%) – Cu (5%)		Ti/TiN/Ai – Si – Cu		Ti/TiN/Ai – Si – Cu	
Metal II	Al – Sl (1%) – Cu (5%)		Al – Sl (1%) – Cu (5%)		Ti/Al – Si – Cu		Ti/Al – Si – Cu	
Passivation	300 nm SiO_2, 800 nm SiN		300 nm SiO_2, 800 nm SiN		300 nm SiO_2, 800 nm SiN		300 nm SiO_2, 800 nm SiN	

The PLICE Antifuse

The antifuse is a vertical, two-terminal structure. It consists of a polysilicon layer on top, N+ doped silicon on the bottom, and an ONO (oxide-nitride-oxide) dielectric layer in between. A Scanning Electron Microscope (SEM) cross section of the antifuse is shown in Figure 1. The size of the antifuse is 3.2 μm^2, 1.4 μm^2, and 1.0 μm^2 for the 2.0 µm, 1.2 µm, and 1.0 µm processes respectively. This small size, along with a low programmed on-resistance (typically 500 ohms) makes the PLICE antifuse an attractive alternative to EPROM, EEPROM, or RAM as a programming element in a large programmable gate array. In the unprogrammed state, the resistance of the antifuse is more than 100 megohms. Actel FPGAs may contain anywhere from 112,000 antifuses (A1010) to 750,000 (A1280). However, typical applications that utilize 85% of the available gates require you to program only 2% to 3% of the available antifuses.

The Unprogrammed Antifuse

To evaluate antifuse reliability, Actel has developed models and collected data for both unprogrammed and programmed antifuses.[1,2] We'll consider the unprogrammed antifuse first. Since the antifuse is a dielectric sandwiched between polysilicon and silicon, the model to evaluate its reliability in the unprogrammed condition is the same as that used for MOS transistor gate oxides.[3] The parameter to evaluate is the dielectric time-to-breakdown (t_{bd}). This parameter is a function of the electric field across the dielectric, as well as temperature. It has the following relationship.[3]

$$t_{bd} = t_0 * \exp(G/E) \quad (1)$$

Where t_{bd} is the time-to-breakdown in seconds, t_0 is a constant in seconds, E is the electric field in Mv/cm, and G is the field acceleration factor in Mv/cm. (G is temperature dependent and will be discussed later.)

By taking the log of both sides of equation 1, we have:

$$\ln(t_{bd}) = G * (1/E) + \ln(t_0) \quad (2)$$

From experimental data, we can plot the log of the time-to-breakdown of the antifuse at various temperatures versus the reciprocal of the electric field across it and derive G from the slope and t_0 from the y-intercept. Actel has done this on single antifuses, large antifuse capacitors, test arrays of 28,000 antifuses, and actual FPGA products. These antifuse areas range from 3.2 μm^2 to 0.35 mm^2. Figure 2 shows plots of data collected on antifuses of different sizes.

There is some discussion in the literature regarding whether time-to-breakdown depends on E or 1/E. To verify the validity of equation 2, we conducted the following experiment. Large 200 µm by 200 µm (0.04 mm^2) area capacitors were packaged and then stressed at more than 11 different voltages. Capacitors with thicknesses ranging from a low of 8.0 nm to a high of 9.5 nm were chosen from two different wafer runs. A total of 610 capacitors were used in the experiment. The test splits and sample sizes are summarized in Table 2. The distribution of time-to-breakdown of the dielectric at each voltage is shown in Figure 3. In Figure 4, the median of the cumulative failure percentage rates (t_{50}) from Figure 3 is plotted versus 1/E. In Figure 5 the median failure percentage is plotted versus E. By comparing the two figures, the validity of the 1/E model is clearly established. A more detailed statistical verification of the 1/E model for the ONO antifuse is given by S. Chiang, etal.[2]

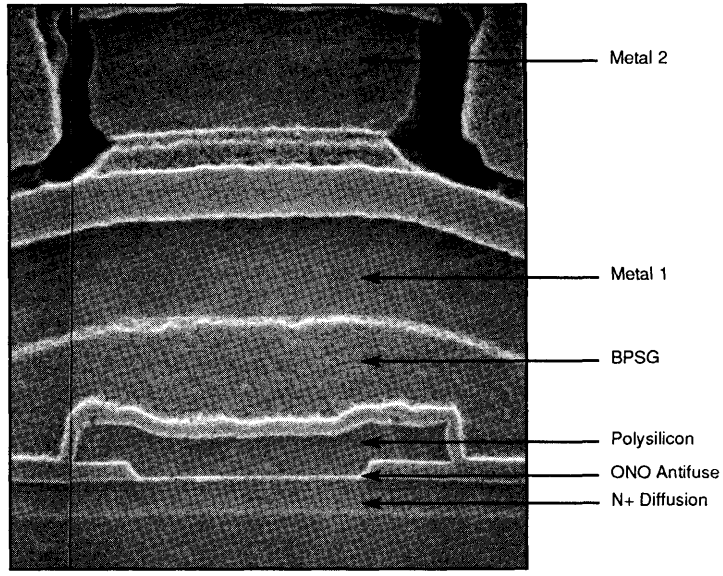


Figure 1. Antifuse SEM Cross Section

Table 2. Field Accelerated Test Data for Two Lots with Thicknesses Ranging from 8 nm to 9.5 nm. (The test was done on a 0.04 mm^2 area capacitor.)

Lot A					Lot B				
Voltage (V)	Tox (nm)	E-Field (MV/cm)	# of Capacitors	t_{50} (sec)	Voltage (V)	Tox (nm)	E-Field (MV/cm)	# of Capacitors	t_{50} (sec)
13.5	8.3	16.2	22	4.2 e-3	14.0	8.7	15.9	25	9.8 e-3
12.5	8.3	15.1	22	3.7 e-2	13.0	8.7	14.9	25	5.0 e-2
12.0	8.3	14.4	22	1.5 e-1	12.5	8.7	14.3	25	2.4 e-1
11.5	8.3	13.8	22	8.6 e-1	12.0	8.7	13.7	25	1.3 e0
11.0	8.4	13.1	22	4.7 e0	11.4	8.7	13.1	25	9.0 e0
10.5	8.4	12.5	9	5.8 e1	11.2	8.7	13.1	45	8.0 e1
10.0	8.3	12.0	6	3.2 e2	10.8	8.7	12.5	45	3.52 e2
9.5	8.3	11.4	6	2.5 e3	10.2	9.0	12.0	45	2.88 e3
9.0	8.3	10.7	36	2.5 e5	9.7	9.0	11.3	45	2.88 e3
8.5	8.3	10.2	15	2.3 35	9.0	8.7	10.3	32	3.35 e5
8.0	8.3	9.6	59	1.5 e6	9.0	9.3	9.7	32	2.22 e6
Subtotal of tested capacitors:			241					369	
Total of tested capacitors:								610	

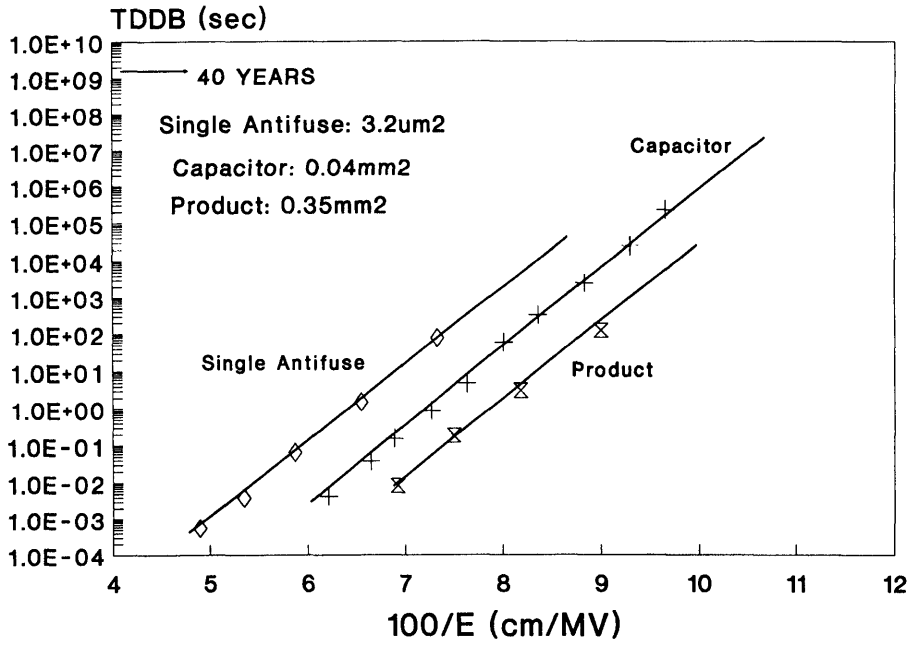


Figure 2. Field Accelerated Test

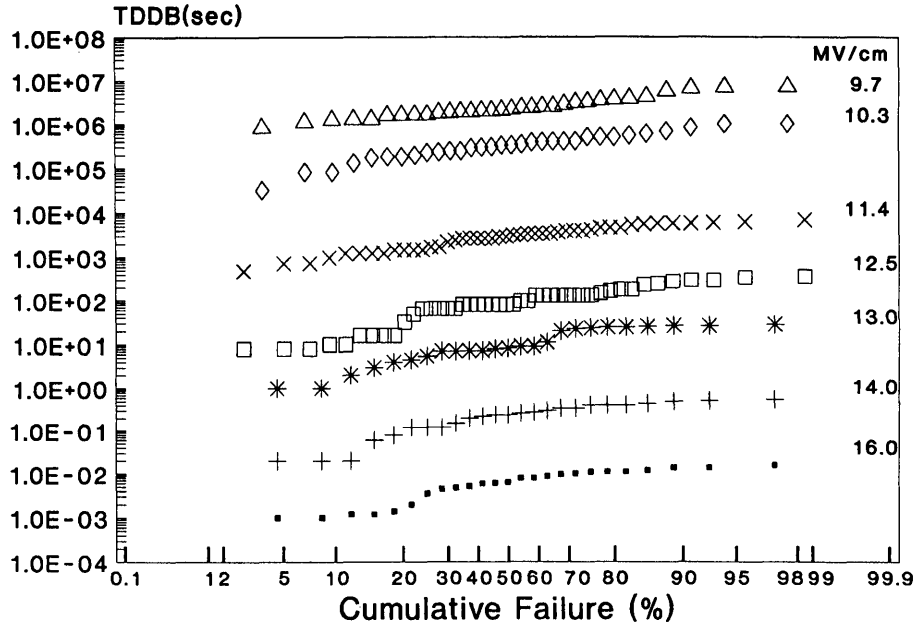


Figure 3. TDDB Distribution

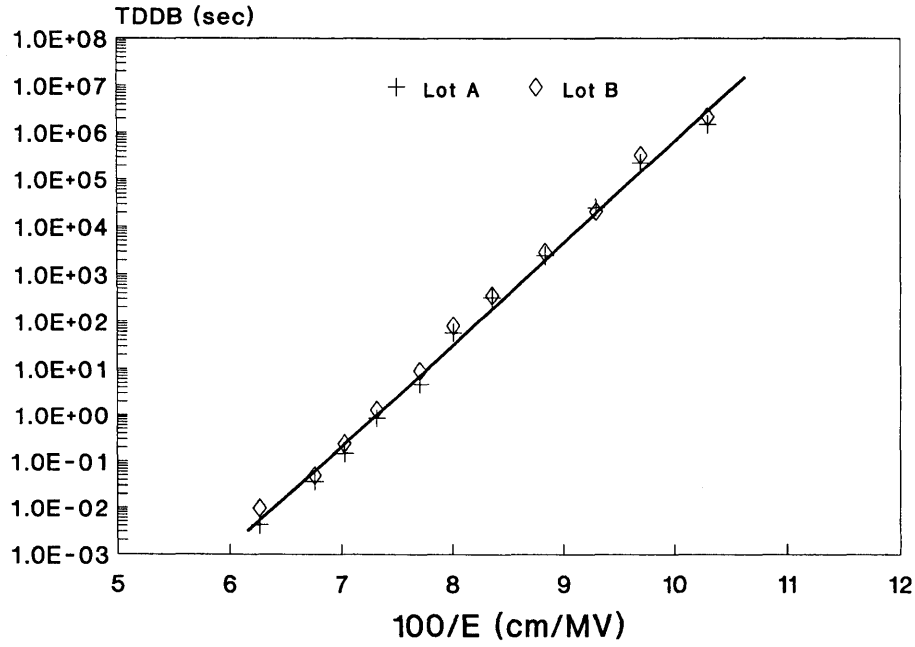


Figure 4. ONO Reliability (1/E Model)

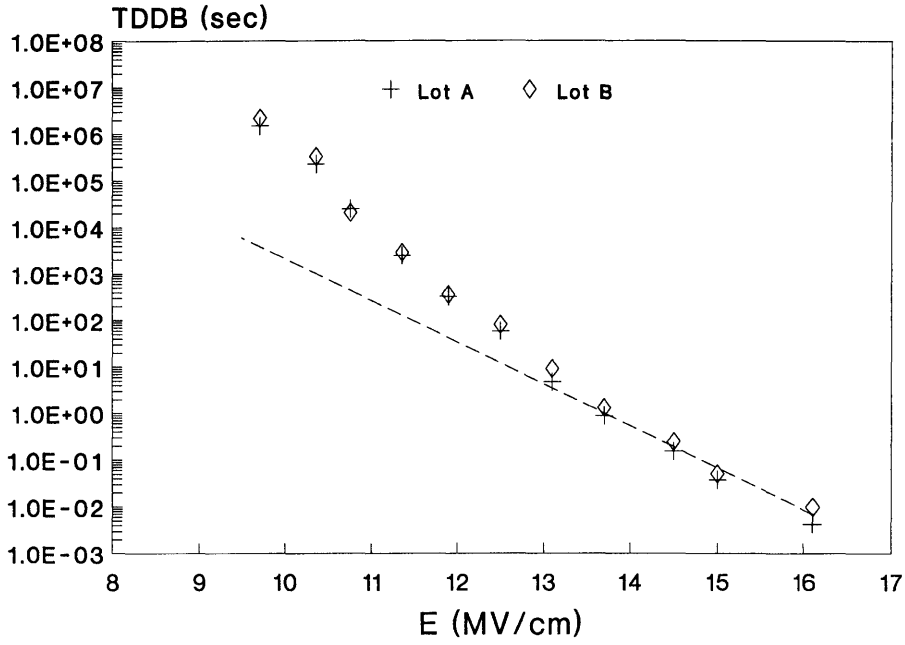


Figure 5. ONO Reliability (E Model)

To quantify the temperature dependence of the time-to-breakdown, we use the Arrhenius equation to determine the semiconductor failure rate of a given process (failure mode) over temperature:

$$R = R_0 * \exp(E_a/kT) \quad (3)$$

where R is the failure rate, R_0 is a constant for a particular process, T is the absolute temperature in degrees Kelvin, k is Boltzmann's constant (8.62×10^{-5} eV/°K) and E_a is the activation energy for the process in electron volts. To determine the acceleration factor for a given failure mode at temperature T_2 as compared with temperature T_1 we use equation 3 to derive:

$$A(T_1, T_2) = \exp[(E_a/k) * \{(1/T_1)-(1/T_2)\}] \quad (4)$$

where A is the acceleration factor.

In Figure 6 we plot t_{50} at different temperatures and electric fields. This plot shows that time-to-breakdown is dependent on temperature as well as electric field. For a given time-to-breakdown of a dielectric, the expression,

$$E_a = k * d\ln(tbd)/d(1/T) \quad (5)$$

gives us the activation energy[2]. The slope in Figure 6 represents the activation energy E_a. E_a also shows a linear correlation with 1/E as shown in Figure 7. The field acceleration factor, G, is also temperature dependent, that is,

$$G(T) = G(298) * [1 + \delta/k * \{1/T - 1/298\}] \quad (6)$$

where G(298) is the field acceleration factor at room temperature (25°C = 298°K) and δ (in eV) characterizes the temperature dependence of G.

By combining equations 1, 5, and 6, E_a can be related to G(T) by:

$$E_a = G(298) * \delta/E - E_b \quad (7)$$

where δ and E_b are treated as fitting parameters between E_a and G.

From the data shown in Figures 3, 4, 6, and 7, we can obtain values of G(298), δ, E_b, and E_a regardless of antifuse area. Typical values of G(298), δ, E_b, and E_a at 5.5 V are 480 MV/cm, 0.014 eV, 0.43 eV, and 1×10^{-16}. By combining equations 1, 6, and 7, we obtain an overall equation for the time-to-breakdown for a given temperature and E field:

$$t_{bd} = t_0 * \exp\{(G(298)/E)[1+(\delta/k) * (1/T-1/298)] - (E_b/k) * (1/T-1/298)\} \quad (8)$$

By applying the values for the constants as defined above, the time-to-breakdown for the antifuse dielectric can be derived for a given temperature and electric field. In Table 3, we have used equation 8 to solve for the acceleration factors associated with powering up a device at high voltage or temperature or both. Then we compare the failure rate with more typical voltages or temperatures. We can see the effect of temperature by comparing 125°C at 5.5 V with 55°C at 5.5 V. The Actel model (equation 8) gives us an acceleration factor of 55.3, or 6.3 equivalent years for

a 1000 hour burn-in at 125°C. Note that this acceleration factor of 55.3 is close to the value of 41.8 derived from the Arrhenius equation (equation 4) using an activation energy of 0.6eV and the same temperatures. We use 0.6 eV (and 0.9 eV) as the activation energy of a general semiconductor failure mode when calculating failure rates based on high temperature operating life (HTOL) later in this report.

We can also see from Table 3 that a small change in voltage is a much more effective reliability screen than is a change in temperature. For example, if we compare 25°C at 5.75 V to 25°C at 5.25 V, we see that a change of just a half volt yields an acceleration factor of 1092.6, or 124.7 equivalent years per 1000 hours at 5.75 V. This strong dependence on voltage allows Actel to screen antifuse infant mortality failures during normal wafer sort testing simply by applying a higher than normal voltage across all antifuses. Because antifuse infant mortality failures can be detected and effectively screened, Actel FPGAs have as high a level of reliability as standard CMOS-processed products.

To establish that the antifuse contributes less than 10 FITs (at 5.5 V, 125°C) to the overall product reliability, Actel has calculated the product failure rates due to the antifuse using three different techniques. In the first case, we evaluated the t_{bd} distribution of 125 A1010 units in which the antifuses received an 11 V stress. Using E_a = 0.9 eV and extrapolating to 5.5 V, we found that the 1% antifuse failure lifetime at 5.5 V is well over 40 years and less than 10 FITs.

The second method of determining product reliability was to look at the results of production wafer sorts. As mentioned earlier, all antifuses receive a high voltage stress at wafer sort to screen out infant mortality failures. Specifically, all antifuses receive the equivalent of two 10 V stresses for one second each. The first stress is to screen out clearly defective antifuses. The second stress is to catch weaker antifuses that could cause problems with product programming yield or infant mortality failures. Actual failure rates observed on the A1010 over ten runs for these two stresses (FS-1 and FS-2) demonstrate average yield loss at the second stress screen of less than 0.3%. By extrapolating this yield loss to a normal 5.5 V operating voltage, we thus conclude that the contribution of the antifuse to the product's lifetime is less than 10 FITs.

The third technique of determining the product's antifuse failure rate is by performing an accelerated burn-in of Actel FPGA products. The acceleration is accomplished by using both higher voltage (5.75 V to 6.0 V) and higher temperatures (125°C to 150°C). Units are programmed to a specific design and exercised in a manner similar to what may occur in a real application. For a detailed description of the test and the results, see "High Temperature Operating Life" later in this report. The results indicate that the antifuse contributes less than 10 FITs to the product's overall failure rate; it is thus an insignificant factor in a product lifetime of 40 years at 5.5 V and 125°C.

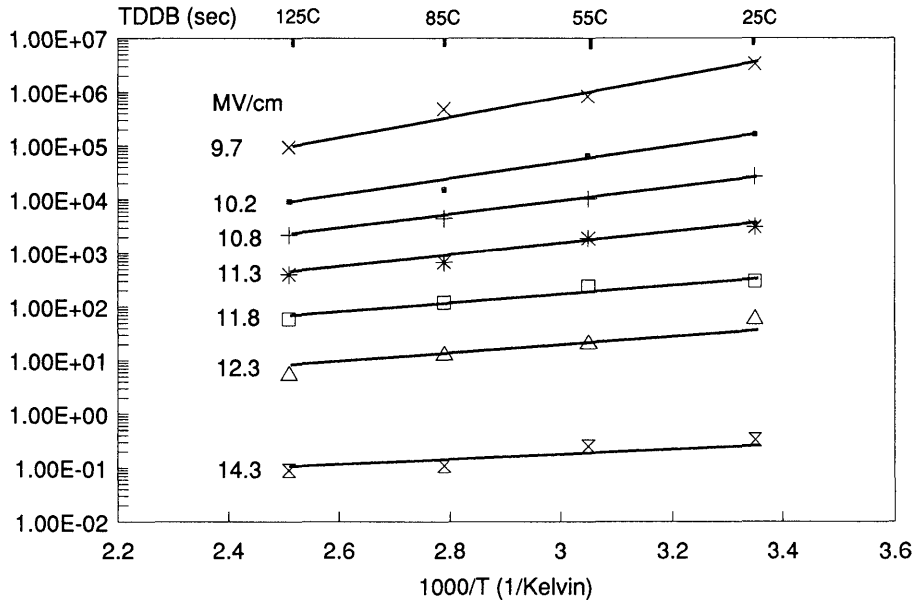


Figure 6. Dependence of E-Field Acceleration on Temperature

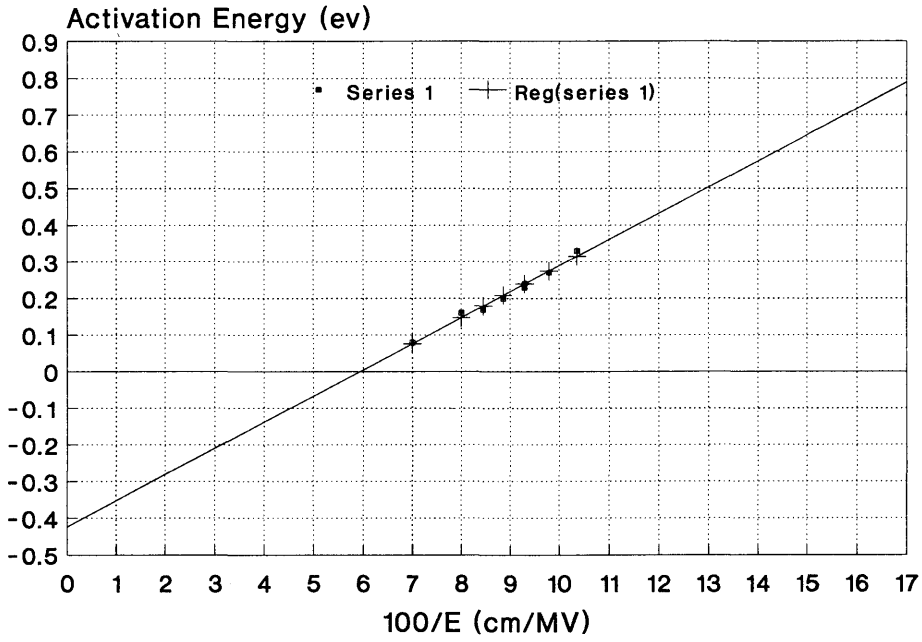


Figure 7. Activation Energy versus 1/E

4

Table 3. Acceleration Factor versus Operating Conditions (Unprogrammed Antifuse)

$t_0 = 1 \times 10^{-16}$ sec., G = 480 MV/cm, f = 0.014 eV, $E_b = 0.43$ eV.

Model	Temperature/Voltage		Acceleration Factor	Equivalent Years for 1000 Hour 125°C Burn-In
	High	Typical		
Fixed Voltage	125°C/5.5 V	55°C/5.5 V	55.3	6.3
	125°C/5.5 V	90°C/5.5 V	6.1	0.7
Fixed Temperature	25°C/5.5 V	25°C/5.25 V	38.8	4.4
	25°C/5.75 V	25°C/5.25 V	1092.6	124.7
	25°C/5.75 V	25°C/5.5 V	28.2	3.2
	25°C/6.0 V	25°C/5.25 V	23321	2662
	25°C/6.0 V	25°C/5.5 V	601.8	68.7
Varied Temperature and Voltage	125°C/5.5 V	55°C/5.25 V	1787.2	204.0
	125°C/5.75 V	55°C/5.5 V	987.8	112.8
	125°C/5.75 V	90°C/5.5 V	109.4	12.5
	125°C/6.0 V	55°C/5.5 V	13865	1583
	125°C/6.0 V	90°C/5.5 V	1535.9	175.3
Fixed 0.6 eV Activation Energy Voltage—Independent	150°C/5.5 V	55°C/5.5 V	117.6	13.4
	150°C/5.5 V	90°C/5.5 V	15.2	1.7
	125°C/5.5 V	55°C/5.5 V	41.8	4.8
	125°C/5.5 V	90°C/5.5 V	5.4	0.6

The Programmed Antifuse

A Kelvin test structure as shown in Figure 8 was used to evaluate the reliability of a programmed antifuse. Here, a strip of polysilicon crosses an N+ diffusion. The antifuse is located at the intersection of the two. There are metal-to-poly contacts at nodes 1 and 3, as well as metal-to-N+ contacts at nodes 2 and 4. A four-terminal Kelvin structure is useful should a failure occur, because antifuse opens can be separated from other problems (such as polysilicon or contact opens) simply by checking for continuity on appropriate pairs of nodes.

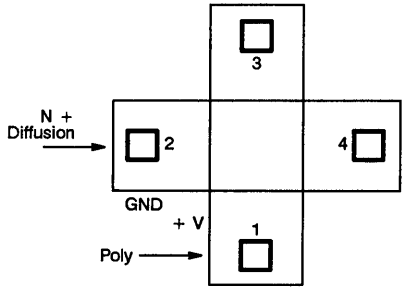


Figure 8. Antifuse Kelvin Structure

Test devices were stressed by forcing a constant 5 mA current from polysilicon to N+ through the antifuse at 250°C. Note that this stress is far greater than a programmed antifuse would be subject to in a device under normal operating conditions. Because the antifuse is used to connect two networks, there is usually no voltage across it; hence, no current passes through. A voltage will appear across the antifuse only momentarily while a network switches from low-to-high or high-to-low.

During the 5 mA, 250°C stress, the voltage across the antifuse was monitored. Figure 9 is a plot of the voltage as a function of stress time. A sudden increase in voltage indicates that an open occurred. As can be seen from the figure, failures occurred at about 300 hours of stress. However, by probing on nodes 3 and 4 of the Kelvin structure, we were able to measure continuity and determine that the cause of failure was not the antifuse. The failed units were then examined on an SEM, where the cause of failure was revealed as metal-to-poly contact electromigration. This is a well-known failure mode in CMOS, which has been determined to have an activation energy of 0.9 eV. Using equation 4, we can predict a lifetime under normal operating conditions in excess of 40 years for this failure mode. The lifetime of the programmed antifuse is even longer.

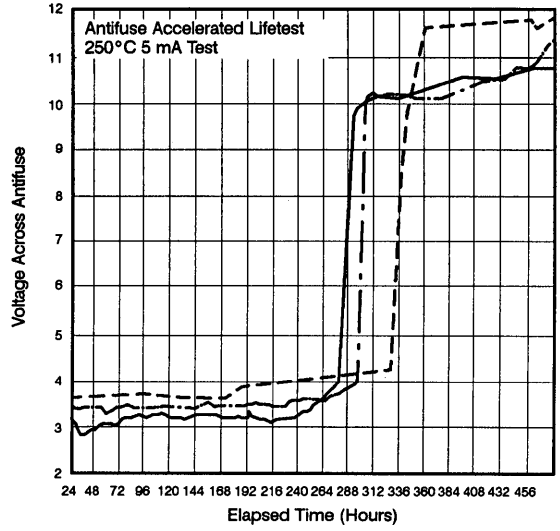


Figure 9. Voltage Across Antifuse versus Stress Time

Actel FPGA Product Reliability

Product reliability was evaluated on fourteen Actel products: a 64K PROM (PROM64), a 300-gate FPGA (1003), 1200-gate FPGAs (A1010/A1010A/A1010B), 2000-gate FPGAs (A1020/A1020A/A1240B), 2500-gate FPGAs (A1225/A1225A), 4000-gate FPGAs (A1240/A1240A), and 8000-gate FPGAs (A1280/A1280A). The PROM64 product uses the same process and antifuse as the FPGAs. The 1003 is a test device that is a smaller version of the A1010/A1020; it was used for early characterization and qualification. As mentioned earlier, the "A" and "B" versions of the FPGAs are linear shrinks of the original 2.0 µm (ACT 1) and 1.2 µm (ACT 2) products. Reliability tests were conducted on units assembled in many different package types as listed in Tables 6 through 11. Package characteristics for Actel FPGAs are shown in Table 4.

High Temperature Operating Life (HTOL)

The intent of an HTOL test is to operate a device dynamically at a high temperature (usually 125°C or 150°C) and extrapolate the failure rate to typical operating conditions. This test is defined by Military Standard-883 in the Group C Quality Conformance Tests. The Arrhenius relationship in equations 3 and 4 is used to do the extrapolation. To use the Arrhenius equation, we need to know the activation energy of the failure mode. Activation energies of antifuse failure modes were discussed previously. Table 5 gives the activation energies of general semiconductor failure modes.

Table 4. Actel FPGA Package Characteristics

Characteristics	PLCC	PQFP
Molding Compound	Sumitomo 6300 H	Sumitomo 6300 H
Filler Material	Fused silicon 70% by weight	Fused silicon 70% by weight
Lead Frame Material	Copper (Olin 150 or equiv.)	Copper (Olin 150 or equiv.)
Lead Plating Composition	Solder, 300–800 micro inches (µin)	Solder, 300–800 micro inches (µin)
Die Attach Material	Silver Epoxy	Silver Epoxy
Flame Retardance	UL-94, V-0	UL-94, V-0
Bond Wire	Gold, 1.3 mil diameter	Gold, 1.3 mil diameter
Bond Attach Method	Thermosonic	Thermosonic
Characteristics	**JQCC**	**PGA/CQFP**
Body Material	Ceramic	Alumina
Lid Material	Ceramic	Kovar, 50 µin gold plated
Sealant	Glass	Au/Sn Solder
Lead Frame Material	Alloy 42 (40% Nickel, 60% Iron)	N/A
Bond Wire	99% Aluminum, 1% Si, 1.25 mil dia.	99% Aluminum, 1% Si, 1.25 mil. dia.
Bond Attach Method	Ultrasonic	Ultrasonic
Lead Finish	Solder dip, 200 µin. min.	A42 or Kovar
Die Attach Material	Silver loaded glass	Silver loaded glass

Thermal Resistance (°C/Watt)

Package	44 PLCC	44 JQCC	68 PLCC	68 JQCC	80 VQFP	84 PLCC	84 JQCC	84 PGA	84 CQFP
θ_{JC}	15	8	13	8	12	12	8	8	5
θ_{JA} (still air)	52	38	45	35	68	44	34	33	40
Package	**100 PQFP**	**100 PGA**	**132 PGA**	**144 PQFP**	**160 PQFP**	**172 CQFP**	**176 PGA**	**207 PGA**	**208 PQFP**
θ_{JC}	13	5	5	15	15	8	8	8	15
θ_{JA} (still air)	55	35	30	35	33	25	23	22	33

Six different data patterns are programmed into the 64K PROMs for HTOL testing: a diagonal of zeros (98% programmed); a diagonal of ones (2% programmed); a topological checkerboard pattern (50%); all zeros (100%); all ones (0%); and an incrementing pattern (50%). During burn-in, all addresses are sequenced through at a 1 MHz clock rate. The outputs are enabled and loaded with a 100 ohm resistor to a 2 V supply. This results in an output loading of equal to or greater than the specified limits of Ioh = –4 mA and Iol = 16mA. In most cases, the PROMs are burned in at Vcc = 5.5 V and at 125°C. However, voltage acceleration experiments are also done at 7 V, 125°C and 8 V, 25°C.

The PROM is useful for antifuse reliability studies for several reasons. First, it allows us to program anywhere from 0% to 100% of the antifuses, although we program only 2% to 3% of the antifuses for a given design in an FPGA device. Also, an antifuse failure on the PROM is very noticeable because the antifuse is directly addressed. A weak fuse would show an AC speed drift, and a failed antifuse would read the wrong data.

For evaluating the 1003/A1010/A1020/A1225/A1240/A1280, we programmed an actual design application into most devices (some units were burned-in unprogrammed) and performed a dynamic burn-in by toggling the clock pins at 1 MHz. The designs selected utilized 85% to 97% of the available logic modules and 85% to 94% of the I/Os. Outputs were usually loaded with 1.2 K ohm resistors to Vcc, which results in greater than 4 mA of sink current as each I/O toggled low. Under these conditions, each unit typically draws a minimum of 100 mA during dynamic burn-in. Most of this current comes from the output loading while about 5 mA is from the device supply current. For a 125°C burn-in, this results in junction temperatures of at least 150°C for plastic packages and 145°C for ceramic packages (depending on package type). Most burn-in was done at 5.75 V or 6.0 V (for voltage acceleration of the antifuse) and 125°C or 150°C.

Table 5. CMOS Failure Modes

Failure Mechanism	Activation Energy
Ionic Contamination	1.0 eV
Oxide Defects	0.3 eV
Hot Carrier Trapping in Oxide (Short Channels)	–0.06 eV
Silicon Defects	0.5 eV
Aluminum-Silicon-Copper Electromigration	0.6 eV
Contact Electromigration	0.9 eV
Electrolytic Corrosion	0.54 eV

As mentioned previously, some units are burned in unprogrammed. To accomplish this, we use a special burn-in circuit that allows us to use the product's test features to serially shift in commands to the chip during burn-in. All internal routing tracks are toggled between Vss and Vcc. When vertical tracks are at Vcc, horizontal tracks are held at Vss and vice versa. Thus, all antifuses that can connect vertical and horizontal tracks receive a full Vcc stress in both directions. Since vertical tracks connect to logic module inputs and outputs, these are also toggled between Vss and Vcc. Finally, a command is sent to the chip to toggle some external I/O pins between Vss and Vcc. This special dynamic burn-in circuit is the same one used by Actel to screen unprogrammed products to MIL–883D requirements. Since virtually all antifuses receive a full Vcc stress, this screen is much more effective at catching unprogrammed antifuse infant mortality failures than is burning in programmed devices where only a fraction of the antifuses are stressed.

A summary of the HTOL data collected by Actel is shown in Table 6. A failure is defined as any device that shows a functional failure, exceeds datasheet DC limits, or exhibits AC speed drift. Among the parts tested, no speed drift, faster or slower, was observed within the accuracy of the test set-up. Failure rates at 55°C, 70°C, and 90°C were extrapolated by using the Arrhenius equation and general activation energies of 0.6 eV and 0.9 eV. Poisson statistics were used to derive a calculated failure rate with a 60% confidence level. Using Poisson statistics is valid for a failure rate that is low and a failure mode that occurs randomly with time. At 55°C, the calculated failure rate with 60% confidence level was found to be 26 FITs or 0.0026% failures per 1000 hours. This number was derived from over 8.6 million device hours (125°C) of data. There were eight total failures out of 6598 tested units. Only one occurred in the first 80 hours of burn-in (of a 1003 product). Seven of the eight failures observed were due to common CMOS failure modes (gate oxide failure, silicon defects, or open via). Only one unit failed because the antifuse failed. This unit was burned in in the unprogrammed state to stress all antifuses. It was stressed at 6 V, 150°C. It passed at 168 hours and failed at 650 hours because an antifuse became programmed. By passing at 168 hours, the unit received a total stress well in excess of 100 years of operation at 5.5 V, 125°C (using equation 8). With only one antifuse related failure in 8.66 million device hours at 125°C, we use equation 8 to derive that this one antifuse failure at 6 V is significantly less than 10 FITs at 5.5 V.

Unbiased Pressure Pot Test

This test is used to qualify products in plastic packages. Units are placed in an autoclave (pressure pot) and exposed to a saturated steam atmosphere at 121°C and 15 psi. Problems with bonding, molding compounds, or wafer passivation can cause metal corrosion to occur in this atmosphere. The existence of metal corrosion is detected during a full electrical test of the device following exposure in the autoclave.

A total of 1771 units from 44 wafer runs were evaluated. Read points were taken at 96, 168, 240, and 336 hours. There were a total of seven failures (Table 7). Five failures were found to be due to bond wires lifting off bond pads. These were caused by assembly problems that occurred only in our first lots of plastic units. The failures were caused by temperature and not by metal corrosion. The assembly problems have been corrected, with no further failures observed. Two A1010A units failed at 336 hours because of metal corrosion at the bond pads, but both units had passed at 168 hours.

Biased Moisture Life Test (85/85)

In this test, the units are placed in a chamber at a temperature of 85°C and a relative humidity of 85%. A voltage of 5.5 V is applied to every other device pin while other pins are grounded. 5.5 V is applied to Vcc while Vss is grounded. This test effectively detects die-related and plastic package-related problems.

As shown in Table 8, a total of 1277 units have been stressed. There have been three failures. Two failures were due to lifted bond wires; these units came from the same lot in which we saw failures in the steam pressure pot. The 1000-hour failure is nonfunctional due to an open Metal 1 line.

Highly Accelerated Stress Test (HAST) at 131°C/85% Humidity

As in the 85/85 test, units receive an alternate pin bias (5.5 V and 0 V) but are exposed to a higher pressure and higher temperature environment. Fifty hours of HAST is generally considered to be equivalent to 1000 hours of 85/85. As summarized in Table 9, 716 units from 18 wafer runs have been tested with no failures.

Temperature Cycling

This test checks for package integrity by cycling units through temperature extremes. Data was taken for cycles of 0°C to 125°C, –40°C to 125°C, and –65°C to 150°C. Both programmed and unprogrammed units were placed on temperature cycle. As shown in Table 10, of 3151 units tested to date, there have been no failures.

Table 6. High Temperature Operating Life (HTOL) Summary

Product	Total Units	Total Wafer Runs	Device Hours at 125°C (0.6 eV)	Failures	Equiv. Dev. Hrs. at 55°C (0.6 eV)	Equiv. Dev. Hrs. at 70°C (0.6 eV)	Equiv. Dev. Hrs. at 90°C (0.6 eV)
64K PROM	295	4	6.18E+05	0	2.59E+07	1.02E+07	3.34E+06
1003	238	3	3.60E+05	1	1.50E+07	5.94E+06	1.94E+06
A1010	703	10	1.17E+06	1	4.87E+07	1.93E+07	6.29E+06
A1010A	618	13	8.20E+05	1	3.43E+07	1.36E+07	4.43E+06
A1010B	536	5	5.08E+05	1	2.12E+07	8.39E+06	2.74E+06
A1020	334	5	2.16E+05	0	9.04E+06	3.57E+06	1.17E+06
A1020A	1720	29	2.61E+06	3	1.09E+08	4.32E+07	1.41E+07
A1020B	612	10	5.72E+05	0	2.39E+07	9.46E+06	3.09E+06
A1225	207	1	2.07E+05	0	8.66E+06	3.42E+06	1.12E+06
A1225A	80	2	8.00E+04	0	3.35E+06	1.32E+06	4.32E+05
A1240	485	7	6.11E+05	0	2.55E+07	1.01E+07	3.30E+06
A1240A	130	3	1.30E+05	0	5.44E+06	2.15E+06	7.02E+05
A1280	586	12	7.03E+05	1	2.94E+07	1.16E+07	3.80E+06
A1280A	54	2	5.40E+04	0	2.26E+06	8.93E+05	2.92E+05
Totals	**6598**	**103**	**8.66E+06**	**8**	**3.62E+08**	**1.43E+08**	**4.68E+07**

Overall FITs

Ambient Temperature	Activation Energy	Observed	60% Confidence
90°C	0.6 EV	171	202
70°C	0.6 EV	56	66
55°C	0.6 EV	22	26
90°C	0.9 EV	66	78
70°C	0.9 EV	12	15
55°C	0.9 EV	3	4

High Temperature Operating Life

Product	Run #	Package	# Units	Pattern	# Hours	# Failures	Temp.	V_{CC}
64KPR	DG1060	24 SB	59	6PATS	2000	0	125	5.50
64KPR	DG1064	24 SB	36	6PATS	2500	0	125	5.50
64KPR	JB13	24 SB	40	6PATS	2000	0	125	5.50
64KPR	JB13	24 SB	40	6PATS	2000	0	125	7.00
64KPR	JB13	24 SB	10	6PATS	2000	0	25	8.00
64KPR	JB14	24 SB	50	6PATS	2500	0	125	5.50
64KPR	JB14	24 SB	50	6PATS	2500	0	125	7.00
64KPR	JB14	24 SB	10	6PATS	2500	0	25	8.00
1003	DG1063	84 JLCC	25	AL7C	2000	0	125	5.75
1003	DG1065	84 JLCC	32	AL7C	2000	0	125	5.75
1003	DG1065	84 JLCC	159	AL7C	500	1	150	7.00
1003	DG1067	84 JLCC	22	AL7C	1000	0	125	5.75

Table 6. High Temperature Operating Life (HTOL) Summary (Continued)

High Temperature Operating Life								
Product	**Run #**	**Package**	**# Units**	**Pattern**	**# Hours**	**# Failures**	**Temp.**	**V_{CC}**
1010	DG1042	84 JLCC	3	4BCNTR	1000	0	125	5.75
1010	DG1047	84 JLCC	2	4BCNTR	1000	0	125	5.75
1010	DG1073	84 JLCC	10	AL9B	2000	0	125	5.75
1010	DG1077	84 JLCC	15	AL9B	2000	0	125	5.75
1010	JB13	84 PLCC	32	AL9B	2000	0	125	5.75
1010	JB13	68 PLCC	126	AL9B	2000	1	125	5.75
1010	JB13	84 JLCC	64	AL9B	2000	0	125	5.75
1010	JB14	68 PLCC	100	AL9B	2000	0	125	5.75
1010	JB14	68 JLCC	50	AL9B	2000	0	125	5.75
1010	JB22	68 PLCC	100	AL9B	1000	0	125	5.75
1010	JB42	68 PLCC	47	VFUSE	1000	0	125	5.75
1010	JB44	68 PLCC	40	VFUSE	1000	0	125	5.75
1010	JB46	68 PLCC	49	VFUSE	1000	0	125	5.75
1010	TI24	68 PLCC	65	VFUSE	2000	0	125	5.75
1010A	JG03	68 PLCC	59	VFUSE	2000	0	125	5.75
1010A	JG03	68 PLCC	117	VFUSE	1000	0	125	5.75
1010A	TI24	68 PLCC	74	VFUSE	2000	0	125	5.75
1010A	TI29	68 PLCC	53	AL9B	1000	1	125	5.75
1010A	TI31	68 PLCC	26	AL9B	2000	0	125	5.75
1010A	TI32	68 PLCC	9	AL9B	2000	0	125	5.75
1010A	TI33	68 PLCC	19	AL9B	2000	0	125	5.75
1010A	TI35	68 PLCC	15	AL9B	2000	0	125	5.75
1010A	TI1104 TI1243 TI1263 TI1297	68 PLCC	107	AL9B	1000	0	125	5.75
1010A	E01-1	68 PLCC	69	AL9B	1000	0	125	5.75
1010A	E02-1	68 PLCC	70	AL9B	1000	0	125	5.75
1010B	TI2072857 TI2072858 TI2072860	68 PLCC	400	AL9B	1000	1	125	5.75
1010B	U1G-01	68 PLCC	79	AL9B	1000	0	125	5.75
1010B	U1G-02	68 PLCC	57	AL9B	500	0	125	5.75
1020	DG1133	84 JLCC	29	AL11	2000	0	125	5.75
1020	JB22	84 PLCC	16	AL11	1000	0	125	5.75
1020	JB22	84 JLCC	32	AL11	1000	0	125	5.75
1020	JB25	84 PLCC	16	AL11	1000	0	125	5.75
1020	JB26	84 PLCC	32	AL11	500	0	125	5.75
1020	JE03	84 JQCC	104	AL11	186	0	150	5.75
1020	JB33	84 JLCC	105	AL11	80	0	150	5.75

Table 6. High Temperature Operating Life (HTOL) Summary (Continued)

High Temperature Operating Life								
Product	**Run #**	**Package**	**# Units**	**Pattern**	**# Hours**	**# Failures**	**Temp.**	**V_{CC}**
1020A	JF01	84 JLCC	25	AL11	2000	0	125	5.75
1020A	JF01	84 PLCC	15	VFUSE	2000	0	125	5.75
1020A	JF02	84 JLCC	44	AL11	2000	0	125	5.75
1020A	JF02	84 JLCC	41	VFUSE	2000	0	125	5.75
1020A	JF04	84 PLCC	77	VFUSE	1000	0	125	5.75
1020A	JF04	84 PLCC	20	VFUSE	500	0	125	5.75
1020A	JF14	84 PLCC	58	AL11	417	0	150	6.00
1020A	JF14	84 PLCC	100	BLANK	417	1	150	6.00
1020A	JF37	84 PLCC	14	AL11	300	1	150	6.00
1020A	JF37	84 PLCC	20	BLANK	300	0	150	6.00
1020A	JF39	84 PLCC	32	AL11	300	0	150	6.00
1020A	JF39	84 PLCC	29	BLANK	300	0	150	6.00
1020A	JF42	84 PLCC	49	BLANK	650	0	150	6.00
1020A	JF42	84 PLCC	30	VFUSE	650	1	150	6.00
1020A	JF66	84 PLCC	33	AL11	1000	0	125	5.75
1020A	JF66	84 PLCC	39	BLANK	1000	0	125	5.75
1020A	JF67	84 PLCC	49	AL11	1000	0	125	5.75
1020A	JF67	84 PLCC	45	BLANK	1000	0	125	5.75
1020A	TI S#1	84 PLCC	79	AL11	1000	0	125	5.75
1020A	E-14	84 PLCC	45	AL11	2000	0	125	5.75
1020A	E-15	84 PLCC	44	AL11	2000	0	125	5.75
1020A	E-17	84 PLCC	45	AL11	2000	0	125	5.75
1020A	JF-207	100 PQFP	129	AL11	1000	0	125	5.75
1020A	D1J1815	84 PGA	51	ICE20	1000	0	125	5.75
	D2B2704							
1020A	E-01	84 PLCC	45	AL11	1000	0	125	5.75
1020A	E-02	84 PLCC	45	AL11	1000	0	125	5.75
1020A	E-03	84 PLCC	45	AL11	1000	0	125	5.75
1020A	ADK29X	84 PLCC	45	AL11	2000	0	125	5.75
1020A	ADA72X	84 PLCC	45	AL11	2000	0	125	5.75
1020A	ADC21X	84 PLCC	45	AL11	2000	0	125	5.75
1020A	TI1130	84 PLCC	223	AL11	1000	0	150	6.00
1020A	TI1139							
1020A	TI1210							
1020A	TI1800	84 PLCC	34	AL11	648	0	150	6.00
1020A	TI1803							
1020A	UP-04	84 PLCC	40	AL11	1000	0	125	5.75
	UP-05	84 PLCC	40	AL11	1000	0	125	5.75

Table 6. High Temperature Operating Life (HTOL) Summary (Continued)

High Temperature Operating Life								
Product	**Run #**	**Package**	**# Units**	**Pattern**	**# Hours**	**# Failures**	**Temp.**	**V_{CC}**
1020B	JJ-14	84 PLCC	45	AL11	1000	0	125	5.75
1020B	JJ-15	84 PLCC	45	AL11	1000	0	125	5.75
1020B	JJ-17	84 PLCC	45	AL11	1000	0	125	5.75
1020B	JJ-13	84 PGA	30	ICE20	1000	0	125	5.75
1020B	JJ-13	84 PGA	80	AL11	500	0	125	5.75
1020B	JJ-16	84 PLCC	80	AL11	1000	0	125	5.75
1020B	U1P-01	84 PLCC	40	AL11	1000	0	125	5.75
1020B	U1P-02	84 PLCC	40	AL11	1000	0	125	5.75
1020B	JJ-24	84 PLCC	87	AL11	1000	0	125	5.75
1020B	EBFJ004	84 PLCC	80	AL11	1000	0	125	5.75
1020B	EBFI004	84 PLCC	40	AL11	1000	0	125	5.75
1225	UJ-01	100 PGA	80	SPEED26	1000	0	125	5.75
1225	UJ-01	100 PQFP	127	PQFP26	1000	0	125	5.75
1225A	U1J-02	100 PQFP	80	PQFP26	1000	0	125	5.75
1240	TI3257	132 PGA	7	CHI1240	500	0	125	5.75
1240	TI3257	144 PQFP	129	CHI1240	1000	0	125	5.75
1240	TI1045571	132 PGA	38	CHI1240	2000	0	125	5.75
1240	TI1053933	132 PGA	55	CHI1240	2000	0	125	5.75
1240	TI1053932	132 PGA	36	CHI1240	2000	0	125	5.75
1240	TI1220494	132 PGA	90	CHI1240	1000	0	125	5.75
1240	UI-01	132 PGA	50	CHI1240	1000	0	125	5.75
1240	UI-03	84 PLCC	80	ACT40	1000	0	125	5.75
1240A	E-02 E-03	144 PQFP	100	CHI1240	1000	0	125	5.75
1240A	E-04	84 PLCC	30	ACT40	1000	0	125	5.75
1280	JH05	176 PGA	15	BETA12	2000	0	125	5.75
1280	JH06	176 PGA	15	BETA12	2000	0	125	5.75
1280	JH03(K)	176 PGA	25	BETA12	2000	0	125	5.75
1280	JH03(SB)	176 PGA	25	BETA12	2000	0	125	5.75
1280	TI1143649	176 PGA	44	BETA12	1000	1	125	5.75
1280	TI1143650	176 PGA	44	BETA12	1000	0	125	5.75
1280	TI1136307	176 PGA	42	BETA12	1000	0	125	5.75
1280	UH-01	176 PGA	26	BETA12	1000	0	125	5.75
1280	UH-02	176 PGA	26	BETA12	1000	0	125	5.75
1280	UH-05	176 PGA	40	SPEED9	1000	0	125	5.75
1280	UH-04	160 PQFP	79	SPEED12	1000	0	125	5.75
1280	UH-10 UH-14	176 PGA	75	SPEED13	1487	0	125	5.75
1280	ADC18X	160 PQFP	130	SPEED12	1000	0	125	5.75
1280A	UHI-01	160 PQFP	27	SPEED12	1000	0	125	5.75
1280A	UH1-02	160 PQFP	27	SPEED12	1000	0	125	5.75

Table 6. High Temperature Operating Life (HTOL) Summary (Continued)

HTOL Failure Analysis			
Product	**Run #**	**Hours**	**Cause**
1003	DG1065	80	I_{CC} = 40 mA. Fails at High V_{CC}. Functional at Low V_{CC}. Not antifuse related
1010	JB13	500	I_{CC} = 30 to 40 mA. Not Functional. Liquid crystal shows hot spot in chip periphery. Silicon defect. Not antifuse related.
1010A	TI29	500	Functional at V_{CC} > 4.5 V. All nets slow. Damaged isolation transistor oxide. Not antifuse related. Unit was marginal at T_0.
1010B	TI2072857	144	I_{DDH} failure at 144 hours. Gate poly to drain short due to gate oxide breakdown in I/O.
1020A	JF14	417	Open Metal I to Metal II via.
1020A	JF37	300	Gross functional failure due to shorted gate oxide in the logic module. Unit was good at 112 hours.
1020A	JF42	650	Failed fuse shorts. Cause is shorted antifuse. Unit was good at 168 hours. Blank pattern at 6 V is much stronger stress than normal application.
1280	TI1143649	500	Tristate LKG pin 28. No cause determined due to bond wire damage during decapsulation.

Table 7. 121°C, 15 PSI Steam Pressure Pot (Unbiased Autoclave)

				Number of Failures			
Product	**Run #**	**Package**	**# Units**	**96 Hours**	**168 Hours**	**240 Hours**	**336 Hours**
1010	JB13	84 PLCC	34	0	3	—	0
1010	JB13	68 PLCC	71	1	0	—	0
1010	JB14	68 PLCC	71	0	0	—	0
1010	JB22	68 PLCC	50	0	0	—	0
1010	JB27	68 PLCC	50	0	0	—	0
1010	JB28	68 PLCC	42	1	0	—	0
1010A	TI15	68 PLCC	77	0	0	—	0
1010A	TI24	68 PLCC	129	0	0	—	0
1010A	TI1104 TI1243 TI1263 TI1297	68 PLCC	77	0	—	0	0
1010A	E01-1	68 PLCC	30	0	0	—	0
1010A	E02-1	68 PLCC	30	0	0	—	0
1010A	E03-1	68 PLCC	30	0	0	—	2
1010B	TI2072857 TI2072858 TI2072860	68 PLCC	45	0	0	0	
1010B	U1G-01	68 PLCC	40	—	0	—	—
1020A	TI1800 TI1859 TI2156	84 PLCC	77	0	—	0	—

Table 7. 121°C, 15 PSI Steam Pressure Pot (Unbiased Autoclave) (Continued)

				Number of Failures			
Product	**Run #**	**Package**	**# Units**	**96 Hours**	**168 Hours**	**240 Hours**	**336 Hours**
1020A	E-01	84 PLCC	26	0	0	—	0
	E-02	84 PLCC	28	0	0	—	0
	E-03	84 PLCC	26	0	0	—	0
1020A	ADK29	84 PLCC	27	—	0	—	
	ADC21X	84 PLCC	27	—	0	—	
	ADA72X	84 PLCC	27	—	0	—	
1020A	JF-207	100 PQFP	80	—	0	—	—
1020A	S-1702A	84 PLCC	25	—	0	—	0
	S-1702B		26	—	0	—	0
	S-1702C		25	—	0	—	0
1020B	JJ14-17	84 PLCC	81	—	0	—	
1020B	U1P-01	84 PLCC	40	—	0	—	
1225	UJ-01	100 PQFP	80	—	0	—	0
1240	TI10301	144 PQFP	79	—	0	—	0
1240	UI-03	84 PLCC	79	—	0	—	0
1240A	E-02	144 PQFP	25	—	0	—	
1240A	E-03	144 PQFP	30	—	0	—	
1240A	E-04	84 PLCC	25	—	0	—	
1280	JH-14	160 PQFP	80	—	0	—	0
1280	ADC18X	160 PQFP	82	—	0	—	

Failure Analysis:

Three 1010 JB13 failures at 168 hours due to lifted bond wires.
Corrective action implemented at assembly vendor.

1010 JB13 failure at 96 hours; same problem as above.

1010 JB28 failure at 96 hours due to open bond wire caused by lifted die paddle. This was the first qual lot from a new assembly vendor and corrective action was implemented.

Two 1010A E03-1 failed cont. at 336 hours. Both units had corroded pads.

Table 8. 85°C/85% Humidity with DC Alternate Pin Bias of 0 V to 5.5 V

				Number of Failures			
Product	**Run #**	**Package**	**# Units**	**168 Hours**	**500 Hours**	**1000 Hours**	**2000 Hours**
1010	JB13	68 PLCC	80	—	2	1	0
1010	JB14	68 PLCC	81	—	0	0	0
1010	JB22	68 PLCC	54	—	0	0	—
1010	JB26	68 PLCC	54	—	0	0	—
1010	JB27	68 PLCC	19	—	0	0	—
1010A	E01-1	68 PLCC	79	—	0		
1010A	E02-2	68 PLCC		—	0		
1010A	E03-1	68 PLCC		—	0		
1010A	TI1104	68 PLCC	80	0	0	0	—
	TI1243						
	TI1263						
	TI1297						
1010B	TI2072857	68 PLCC	201	0	0	0	—
	TI2072858						
	TI2072860						—
1020	JF48	84 PLCC	34	0	0	—	—
	JF49	84 PLCC	49	0	0	—	—
1020A	E-01	84 PLCC	64	—	0	0	
	E-02	84 PLCC					
	E-03	84 PLCC					
1020A	ADK29	84 PLCC	27	—	0	0	
	ADC21X	84 PLCC	27	—	0	0	
	ADA72X	84 PLCC	27	—	0	0	
1020A	E14	84 PLCC	24	—	0	0	
	E15	84 PLCC	29	—	0	0	
	E17	84 PLCC	32	—	0	0	
1020A	UP-06	100 PQFP	77	—	0	0	
1020B	JJ-14	84 PLCC	27	—	—	0	
	JJ-15	84 PLCC	27	—	—	0	
	JJ-17	84 PLCC	27	—	—	0	
1240	TI3256	144 PQFP	78	—	0	0	
1280	UH-04	160 PQFP	80	—	0	0	

Failure Analysis:

1010	JB13	Two failures at 500 hours. Open pins due to bond lifting. Corrective action implemented at assembly vendor.
		One failure at 1000 hours. Horizontal track open (Metal I).

Table 9. Biased Humidity (HAST) 131°C/85% Humidity

				Number of Failures					
Product	Run #	Package	# Units	50 Hours	100 Hours	200 Hours	240 Hours	300 Hours	400 Hours
1020	E18, 22	84 PLCC	29		0	0	—	0	0
1020A	TI1130	84 PLCC	77	—	0	0	0	—	
	TI1139								
	TI1210								
1020B	EBFJ001	84 PLCC	44	—	0				
1020B	EBFI004	84 PLCC	36	—	0				
1225	UJ-03	100 PQFP	80	0	0				
1240	UI-03	84 PLCC	80	0	0				
1240A	E-04	84 PLCC	77	0	—				
1240A	E-02,03	144 PQFP	53	0	—				
1280	ADC20X	160 PQFP	80	0					
1280A	U1H-01	160 PQFP	40	0	—				
1280A	U1H-02	160 PQFP	40	0	—				
1280A	EBFJ002	160 PQFP	30	0	—				
1280A	EBFJ003	160 PQFP	30	0	—				
1280A	EBFJ004	160 PQFP	20	0	—				

Table 10. Temperature Cycle

				Number of Failures			
Product	Run #	Package	# Units	100 Cycles	500 Cycles	1000 Cycles	2000 Cycles
0°C to 125°C Cycles							
1010	DG1077	84 JLCC	20	0	—	—	—
1010	JB13	68 PLCC	158	0	—	0	—
1010	JB14	68 PLCC	28	0	—	0	—
1010	JB26	68 PLCC	21	0	—	0	—
1010	JB28	68 PLCC	31	0	—	0	—
1010A	TI15	68 PLCC	125	0	—	0	—
1010A	TI24	68 PLCC	176	0	—	0	—
1010A	TI1104	68 PLCC	129	0	0	0	0
	TI1243						
	TI1263						
	TI1297						
1020	JB22	84 JLCC	17	0	—	0	—
1020A	TI1800	84 PLCC	129	0	0	0	0
	TI1859						
	TI2156						
–40°C to 125°C Cycles							
1010A	TI1104	68 PLCC	129	0	0	0	0
	TI1243						
	TI1263						
	TI1297						
1020A	TI1800	84 PLCC	129	0	0	0	0
	TI1859						
	TI2156						

Table 10. Temperature Cycle (Continued)

				Number of Failures			
Product	**Run #**	**Package**	**# Units**	**100 Cycles**	**500 Cycles**	**1000 Cycles**	**2000 Cycles**
–65°C to 1505°C Cycles							
1010A	TI1104	68 PLCC	129	0	0	0	0
	TI1243						
	TI1263						
	TI1297						
1010B	TI2072857	68 PLCC	201	0	0	0	
	TI2072858						
	TI2072860						
1010B	U1G-01,02	68 PLCC	40	—	—	0	
1020A	TI1800	84 PLCC	129	0	0	0	0
	TI1859						
	TI2156						
1020A	JF-71	100 PQFP	129	—	—	0	
1020A	E01	84 PLCC	85	—	—	0	
	E02						
	E03						
1020A	S-1702A,B,C	84 PLCC	144	0	—	0	
1020B	JJ14-17	84 PLCC	81	—	—	0	
1020B	U1P-01,02	84 PLCC	40	—	—	0	
1020B	EBFJ001	84 PLCC	80	—	—	0	
1225	UJ-01	100 PGA	80	0	—	0	
1225	UJ-01	100 PQFP	80	—	—	0	
1240	TI1053932	132 PGA	15	0	0	0	
	TI1045571		20				
	TI1053933		42				
1240	TI 3256	144 PQFP	80	—	0	0	
1240	UI-01	132 PGA	50	0	—	0	
1240	UI-02	132 PGA	50	0	—	0	
1240	UI-03	84 PLCC	80	—	—	0	
1240A	E-02	144 PQFP	25	—	—	0	
1240A	E-03	144 PQFP	28	—	—	0	
1240A	E-04	84 PLCC	77	—	—	0	
1280	TI1136307	176 PGA	71	0	—	0	
	TI1143650	176 PGA	5	0	—	0	
	TI1143649	176 PGA	1	0	—	0	
1280	UH-01	176 PGA	25	0	—	0	
1280	UH-02	176 PGA	26	0	—	0	
1280	UH-04	160 PQFP	34	0	—	0	
1280	JH-14	160 PQFP	48	0	—	0	
1280	ADE16X	160 PQFP	27	—	—	0	
1280	ADC18X	160 PQFP	30	—	—	0	
1280	ADC20X	160 PQFP	27	—	—	0	
1280A	UH1-01	160 PQFP	40	—	—	0	
1280A	UH1-02	160 PQFP	40	—	—	0	

Other Tests

Electrostatic Discharge (ESD)

All Actel products contain static electricity protection circuitry and are tested for sensitivity to static electricity by using the human body model as described in MIL-STD-883D (100 pf discharged through 1.5 Kohms). Three positive and three negative pulses are discharged into each pin tested at each voltage level. For inputs and I/Os, these six pulses are applied with three different grounding conditions: Vss only grounded, Vcc only grounded, and all other I/Os grounded. Thus, each pin receives a total of eighteen pulses for each test voltage. After pulsing, the units are then tested on a VLSI tester. Leakage currents are measured at 0 V and 5.5 V. Any pin showing more than 1 µA of leakage is considered to be a failure. To date, all Actel products pass ESD testing at 1500 V or better. For further information about specific products and packages, please contact Actel.

Latch-Up

Latch-up is a well known cause of failure in CMOS circuits. Parasitic bipolar transistors are created by the P-Channel transistors, the N-channel transistors, the N-Well, and the P-Substrate. These transistors are connected in a manner that effectively creates an SCR. If a voltage on an external pin were to forward bias to the substrate, the parasitic SCR can be latched to the on state, thereby creating a low impedance path between Vcc and ground. A large amount of current then flows through this path. This current can, at best, temporarily make the device nonfunctional and, at worst, cause permanent damage.

There are several techniques used by CMOS designers to reduce the chance of latch-up. One of the most common techniques is using guard rings to isolate P-Channel and N-Channel transistors. The disadvantage of this method is that it requires additional silicon die area. Another method is to use a substrate bias generator. Creating a negative substrate bias means that an input must go even more negative to cause latch-up. A third technique is to use EPI wafers to achieve low substrate resistance, which lowers the chance of triggering latch-up. Actel uses both guard ring and EPI wafer techniques for all FPGA devices.

The latch-up test method is defined by JEDEC Standard No. 17. Each I/O pin on a tested device was forward biased in both directions (to Vss and Vcc) by forcing negative and positive currents ranging from ±50 mA to ±250 mA in 50 mA increments. Following each stress, the device Icc current was measured. If the current exceeded the datasheet limit of 10 mA, the unit was rejected. The device was also functionally tested.

Six units from three different wafer lots are tested to qualify each Actel product. Testing is done at room temperature as well as at a worst-case temperature of 135°C. All device I/Os and power supplies are tested. To date, no failures have been detected up through 250 mA.

Radiation Hardness

A programmed antifuse makes a connection between an upper layer of polysilicon and an N+ diffusion on the bottom. This connection is very similar to a "buried contact" used in some MOS processes. Many other programmable logic products rely on a stored charge to make their connections (for example, RAM, EPROM, or EEPROM). This stored charge can be susceptible to degradation from radiation exposure. The Actel antifuse makes a hard contact and does not rely on a stored charge. As a result, one would expect the Actel products to have superior radiation tolerance compared to products that use a stored charge.

Although Actel has not yet performed radiation testing, several customers and independent laboratories have performed tests and shared their data. This data shows that the A1010/A1020 devices can withstand a total radiation dose of more than one million RADs. Upsets/bit-day have been measured at 1×10^{-6}. Single Event Upset (SEU) sensitivity measurements gave an asymptotic cross section of 3.6×10^{-6} cm^2/bit. The threshold for Linear Energy Transfer (LET) was 22 MeV–cm^2/mg. For further information, please contact Actel.

Summary

The data presented in this report establishes the excellent reliability of Actel FPGAs. Both Actel models and actual device testing show that the antifuse is highly reliable and that the combined contribution of all antifuses to the gate array product's hard failure rate is less than 10 FITs or 0.001% failures per 1000 hours.

References

1. E. Hamdy, et al, "Dielectric Based Antifuse for Logic and Memory ICs," IEDM paper, p. 786–789, 1988
2. S. Chiang, et al, "Oxide-Nitride-Oxide Antifuse Reliability," Proc. Int. Rel. Phys. Symp., 1990
3. J. Lee, I. Chen, and C. Hu, "Modeling and Characterization of Gate Oxide Reliability," IEEE Trans. of Elec. Dev., Dec. 1988.

Antifuse Field Programmable Gate Arrays

JONATHAN GREENE, MEMBER, IEEE, ESMAT HAMDY, SENIOR MEMBER, IEEE, AND SAM BEAL

Invited Paper

An antifuse is an electrically programmable two-terminal device with small area and low parasitic resistance and capacitance. Field programmable gate arrays (FPGA's) using antifuses in a segmented channel routing architecture now offer the digital logic capabilities of an 8000-gate conventional gate array and system speeds of 40–60 MHz. A brief survey of antifuse technologies is provided. The antifuse technology, routing architecture, logic module, design automation, programming, testing and use of ACT™ antifuse FPGA's are described. Some inherent tradeoffs involving the antifuse characteristics, routing architecture and logic module are illustrated.

I. Introduction

A decade ago the designer of an application specific integrated circuit (ASIC) was constrained chiefly by the performance, logic complexity, and fabrication cost attainable with state-of-the-art semiconductor process technology. Today, the capabilities of leading edge silicon have outstripped the requirements of typical applications. Gate arrays with over 100 000 usable gates and propagation delays well below 1 nsec are now available. Yet estimates show that over half the volume of the gate array market is in designs below 10 000 gates [1], and that over half the systems using gate arrays run at clock frequencies below 25 MHz [2]. At the same time, pressures to reduce product development time have become intense. Gate arrays require a customized mask for the metal wiring and hence take weeks to months to produce.

This environment motivated the development of field programmable gate arrays (FPGA's). For the many users who do not require the speed and gate capacity of conventional arrays, it is preferable to program an off-the-shelf FPGA and have chips available within minutes after their design is completed.

A key characteristic of an FPGA is the programmable switch technology used to configure it. Many such technologies have been considered for use in FPGA's, including laser programming [3], [4], pass transistors controlled by either SRAM [5] or EPROM [6] memory cells and antifuses [7]–[10]. (See [11] for a list of recent references.)

An antifuse is an electrically programmable two-terminal device. It irreversibly changes from a high to a low resistance when a programming voltage (in excess of normal signal levels) is applied across its terminals. Antifuses offer several unique features for FPGA's, most notably a relatively low "on" resistance of 100–600 ohms and a small size. The layout area of an antifuse cell is generally smaller than the pitch of the metal lines it connects; it is about the same size as a via connecting metal lines in a mask programmed array. Nearly 750 000 antifuses can now be integrated on a single FPGA chip, facilitating the development of routing architectures approaching the flexibility and scaling potential of conventional gate arrays. Current antifuse FPGA's offer complexity equivalent to an 8000-gate conventional gate array and typical system clock speeds of 40–60 MHz.

This paper describes the technology, architecture and use of the ACT FPGA families, which are based on the PLICE(R) antifuse [8]. Similar principles will most likely apply to other antifuse-based FPGA's now emerging as the field undergoes rapid development of both antifuse process technology and FPGA architecture. The ACT 1 family [12] [13] ranges from 1200 to 2000 gates and the ACT 2 family [14] from 2500 to 8000 gates.[1] An ACT 3 family supporting higher speeds over a range of 1000–10 000 gates is planned.

Figure 1 shows the basic structure of an ACT FPGA. Rows of logic modules are interspersed with horizontal routing channels containing predefined wiring segments of various lengths and horizontal positions. Other wiring segments run vertically through the modules and across the channels. (The figure shows only a few suggestive vertical

Manuscript received December 30, 1990; revised January 30, 1992.

J. Greene is with the BioCAD Corporation, Mountain View, CA 94043.

E. Hamdy and S. Beal are with the Actel Corporation, Sunnyvale, CA 94086.

IEEE Log Number 9210746.

[1] Logic capacities in this paper are based on the stated capacity in "gates" (cells of four transistors) of an equivalent conventional channeled gate array. The estimates are derived as follows using various benchmark circuits and realistic application designs. First one determines the number of replications of a circuit that fit in a given FPGA; placement and routing are done automatically and must complete. The number of replications is then multiplied by the number of gates each instance of the circuit would require in a conventional channeled gate array, assuming the 80% gate utilization typical of such arrays. The benchmarks include data path, counter, state machine and arithmetic circuits. See [15] for details.

0018–9219/93$03.00 © 1993 IEEE

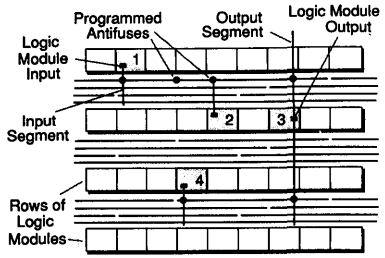


Fig. 1. Basic structure of an ACT FPGA.

segments.) Each logic module computes a single-output function of several inputs. Each module input is connected to a dedicated vertical wiring segment spanning either the channel just above or below the module. Each output signal appears on a dedicated vertical wiring segment of somewhat longer length. An antifuse is provided at each intersection of a horizontal and vertical segment, permitting them to be connected. The output of the driver, module 3, is connected by programmed antifuses to horizontal segments, which in turn are connected to input segments. In the top channel, an antifuse is used to link two adjacent horizontal segments end-to-end, making it possible to reach an input of module 1 as well as 2.

The central array of modules and channels is surrounded by input/output pads and buffers. Each I/O buffer can be connected to the internal logic through a special module near the edge of the array.

The remainder of this paper is organized as follows. Section II begins with a brief survey of the history and current status of antifuse technology in general, and then describes the physical structure, manufacture and reliability of the PLICE antifuse. Section III describes some basic principles of programmable routing applicable to any programmable switch technology, and the "segmented routing channel" model. Section IV describes the routing architecture of ACT FPGA's, which employ antifuses to implement segmented channels. Section V discusses the factors influencing the choice of the basic logic module and gives a full description of the modules used in the ACT families. Section VI briefly describes how input/output and clock distribution are handled. Section VII gives an overview of programming and testing methods, while Section VIII describes how a design is created and the design automation tools used. The design examples given in Section IX convey the speed and integration capabilities of antifuse FPGA's. By way of summary, Section X discusses how antifuse FPGA's have been used in practice and are expected to evolve in the future.

II. Antifuse Technology

A high-performance FPGA requires a programmable interconnect switch having small area and low parasitic resistance and capacitance. The switch technology must be manufacturable and reliable.

Laser-programmed switches [3], [4] can offer good performance, but require costly equipment having direct access to the unpackaged die for programming. Although the switch itself may be small, some approaches require a buffer zone around it to protect adjacent structures from being damaged by the laser.

Early PROM's and PLD's employed electrically blown fuses made of various types of material, e.g., polysilicon, platinum silicide, tungsten-titanium, and nickel-chromium. These have proven difficult to manufacture and program with sufficient reliability for state-of-the-art integrated circuits. The most common difficulty is that an inadequately blown fuse can "grow back" (reconnect) over time.

A pass transistor can also be used as a programmable switch [5]. Although this approach is widely used (see [11] for references), the significant resistance and capacitance of the switch transistor and the large area of the SRAM or EPROM cell controlling it constrain the design of the routing architecture and the performance of the circuit.

An antifuse switch technology offers these advantages for FPGA's:

- Antifuses have a significantly lower "on" resistance and parasitic capacitance than switch transistors, reducing RC delays in the routing.
- Antifuses are small, typically fitting within the minimum pitch of metal wires. This allows the use of regular and flexible routing architectures.
- Antifuses are "normally off" devices. Only the small fraction of the total that need to be turned on must be programmed (about 2% for a typical application). So other things being equal, programming is faster with antifuses than with "normally on" devices.

The concept of antifuses dates back at least to 1957, when they were considered for use in memories [16]. Antifuses developed since then fall into two categories: amorphous silicon and dielectric.

A layer of amorphous silicon placed between two metal layers undergoes a phase change when current is passed through it, becoming conductive. Devices based on this principle have been the subject of research for many years [7], [9], [10], [17]–[20], and were considered for an early FPGA design [21]. Their use has been hampered by two difficulties. First, application of a reverse current can return the amorphous silicon in a programmed antifuse to a nonconductive state. Second, even unprogrammed devices pass a small but significant current, termed *leakage current*. In a memory, where only a few bits must be active simultaneously, these problems can be avoided by careful design. They are more significant in an FPGA since the supply voltage is present across about half the antifuses at any given time.

Recent efforts at developing an amorphous silicon antifuse for FPGA's report the following results. Resistance is inversely proportional to the programming current, and is 50–110 ohms with a mode of 80 ohms at a programming current above 10 mA [22]. The capacitance contributed by each antifuse is 1.3 femtofarads in a 1.0 micron CMOS

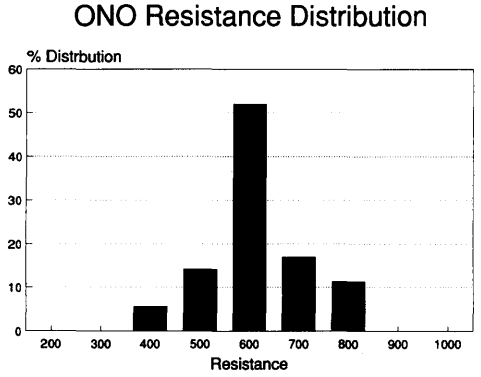


Fig. 2. Resistance distribution of PLICE antifuses programmed with 5mA.

process [22]. (This figure does not account for the capacitance of the metal lines themselves, which contribute several times this amount per antifuse.) Preprogramming leakage current is under 10 nanoamperes at 5.5 volts [10].

Dielectric antifuses consist of a layer of dielectric material placed between N+ diffusion and polysilicon. Upon application of sufficient voltage, the dielectric breaks down. Early dielectric antifuses used a single-layer oxide dielectric. The remainder of this section focuses on the "programmable low impedance circuit element" (PLICE), a multilayer oxide-nitride-oxide (ONO) dielectric antifuse developed for use in FPGA's [8]. FPGA's integrating as many as 750 000 PLICE antifuses on a single chip are currently in volume production.

Application of a 16 V programming pulse across the PLICE melts the dielectric, creating a conductive link of polycrystalline silicon between the electrodes. Typically, a single link is observed. The radius of the link increases with programming current, hence lowering the resistance. As the programming current flows, dopant atoms flow from both electrodes into the link, providing a controllable low resistance. A pulse of under one msec suffices.

For a minimum area PLICE programmed with 5 mA, the resistance is distributed as shown in Fig. 2, with a mode of 600 ohms. With a programming current of 15 mA, the mode of the distribution is shifted down to about 100 ohms. Capacitance is 10 femtofarads per antifuse in a 1.2 micron CMOS process; this includes the contribution of the polysilicon and diffusion electrodes and the metal lines used to connect them. An unprogrammed PLICE has a leakage current of about one femtoampere, so even for the largest FPGA's the total leakage is negligible.

The PLICE structure is shown in Fig. 3. A thin layer of oxide is thermally grown on top of the N+ surface, followed by low pressure chemical vapor deposition (LPCVD) nitride and the reoxidized top oxide. Use of the ONO structure tightens the resistance distribution and also improves both the yield and the reliability compared to simple oxide antifuses [23]. The PLICE adds three masks

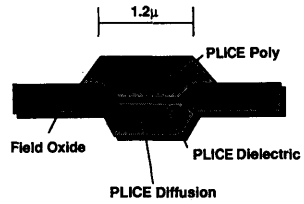


Fig. 3. PLICE antifuse structure.

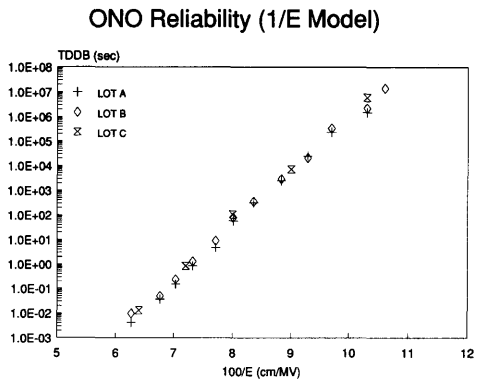


Fig. 4. Time to dielectric breakdown versus reciprocal of electric field.

to a conventional double-metal CMOS process. It can be fabricated in a typical CMOS facility using standard material, processing equipment and techniques.

Antifuse reliability must be considered for both the unprogrammed and programmed states. For an unprogrammed antifuse, with ONO less than 10 nm thick, time dependent dielectric breakdown (TDDB) reliability over 40 years is an important consideration. Ordinary accelerated testing using electric field and temperature stress has been done in order to extrapolate the dielectric's lifetime under normal operating conditions. Figure 4 shows a plot of the time-to-breakdown vs. the reciprocal of the electric field applied to the dielectric, which has been shown to be the proper model [23]. Based on this data, one can extrapolate a lifetime for an ONO antifuse of well over 40 years of normal operation at 5.5 V and 125 C.

It is equally important that the resistance of a programmed antifuse remain low during the life of the part. Single-layer oxide dielectrics are known to be susceptible to "self healing," which would increase the resistance with time. This is not the case for ONO dielectrics. Temperature-accelerated measurements reveal no intrinsic failure mechanism; the programmed antifuse resistance remains unchanged in all cases. The true lifetime of a programmed antifuse has yet to be determined since normal CMOS electromigration failures destroy the test structure first.

Of course, the reliability of the FPGA is affected by the reliability of the base CMOS process as well as the PLICE. PLICE-based FPGA product reliability studies show failure levels encountered with normal CMOS circuits. [24]

Fortuitously, the ONO dielectric is highly radiation resistant. Initial results show that products containing ONO antifuses can withstand 1 500 000 rads [25].

III. Principles of Programmable Routing

A routing architecture for an FPGA must meet two criteria: routability and speed. Routability refers to the capability of an FPGA to accommodate all the nets of a typical application, despite the fact that the wiring segments must be defined at the time the blank FPGA is made. Only the switches connecting the wiring segments can be customized (by programming) for a specific application, not the numbers, lengths or locations of the wiring segments themselves. While sufficient wiring segments for good routability must be provided, excess wiring segments will waste chip area. It is also important that the routing of an application can be determined by an automatic program with little or no manual intervention.

Propagation delay through the routing is a major factor in FPGA performance. In any efficient gate array architecture, whether mask or field programmable, it is inevitable that some nets require longer routings than others. After routing, the exact segments and switches used to establish the net are known and the delay from the driving output to each input can be computed accordingly. The resulting post-layout delays will deviate from prelayout estimates according to some statistical distribution. If the distribution of the delays for nets of a particular fanout is too broad, a user will have difficulty estimating delays when designing an application. And, of course, if the average of the distribution is too high the chip will be slow.

The delay distribution for mask-programmed arrays is sufficiently tight that variation isn't a major difficulty for the designer, since the resistance and capacitance of the metal wires are low.[2] This problem is more challenging for FPGA architectures. Any programmable switch (EPROM, MOS pass device, or antifuse) has a significant resistance and capacitance. Each time a signal passes through a programmable switch, another RC stage is added to the propagation path. For a fixed R and C, the propagation delay mounts quadratically with the number of series RC stages. This tends to increase the average of the net delays and to broaden the post-layout delay distribution.

The use of a low resistance switch, such as an antifuse, helps to keep the delay low and its distribution tight. Of equal significance is optimization of the routing architecture. Some tradeoffs involving the length of wiring segments in a channel, the area required and the the resistance and capacitance of the switch are illustrated in Fig. 5.

[2] As feature size shrinks, however, metal resistance increases faster than capacitance decreases. Even with current masked gate arrays, it is becoming important to model the distributed RC delay in the final routing to get accurate delays.

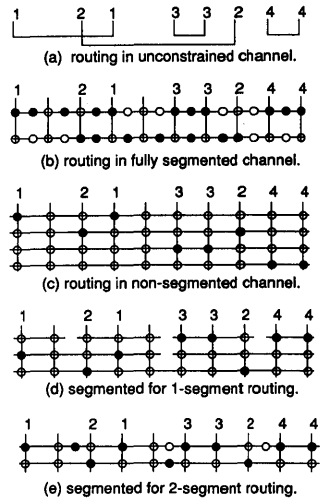


Fig. 5. Segmented routing tradeoffs. Dark circles denote programmed antifuse, open circles unprogrammed antifuses.

Figure 5(a) shows a set of nets routed in a conventional channel. With the complete freedom to configure the wiring afforded by mask programming, the positions and lengths of the horizontal wires can be customized for the particular set of nets. The "left edge algorithm" [26] shows how to do this using a number of tracks equal to the *channel density*, the maximum number of nets passing through any cut across the channel [27]. (This assumes there are no "vertical constraints" [27]. Vertical constraints do not occur in FPGA's since each signal enters or leaves the channel on its own vertical segment.)

In an FPGA, achieving this complete freedom would require switches at every cross point. More switches would be required between each two cross points along a track to allow the track to be subdivided into segments of arbitrary length [Fig. 5(b)]. Since the number of RC stages encountered by a net is proportional to its length, the delay of long nets becomes unacceptable.

Another alternative would be to provide continuous tracks in sufficient number to accommodate all nets [Fig. 5(c)]. This is the approach used in many types of programmable logic arrays and in the interconnect array portion of certain programmable logic devices (e.g., [6], [28]). Advantages are that only two RC stages are encountered and that the delay of each net is identical and predictable. However, even short nets incur the capacitance of a full track length. Furthermore the area is excessive, growing quadratically with the number of nets.

A segmented routing channel offers an intermediate approach. The tracks are divided into segments of varying lengths [Fig. 5(d)], allowing each net to be routed using a single segment of the appropriate size. Greater routing flexibility is obtained by allowing multiple adjacent seg-

ments in the same track to be joined end-to-end by switches [Fig. 5(e)]. Enforcement of simple limits on the number of segments joined or their total length guarantees that the delay will not be unduly increased.

The problem of routing a segmented channel, i.e., determining which segment(s) to assign to each net, is solvable in linear time for single-segment routing [the model of Fig. 5(d)]. When a net is allowed to use multiple segments [Fig. 5(e)], the general routing problem becomes more difficult (in particular, NP-complete [29]). However many important special cases can be solved in polynomial time, and practical problems can be routed in a few minutes on a personal computer using heuristic methods. Reference [29] gives a more thorough review of algorithms for segmented channel routing.

How does one design a segmented channel? In a conventional channel the number of tracks, or *channel width*, must be chosen to accommodate most applications. Statistical models have been developed to estimate the required channel width (e.g., [30]). In a segmented channel, both the channel width and the segment lengths and positions must be chosen carefully to suit the statistics of anticipated applications. Analytical solutions to this problem are as yet unknown, but experience indicates that even channels designed in an ad hoc manner can be quite efficient (with limited use of multiple-segment routing).

A well-designed segmented channel does not require many more tracks than would be needed in a conventional channel. This is a surprising finding given the considerable restrictions segmented routing imposes, but it can be supported both experimentally and analytically. A distribution giving the probability of occurrence of a connection as a function of length and starting point was derived from placements of 510 channels from 34 applications. A segmentation was designed for a channel with 32 tracks, taking this distribution into account and assuming that at most two segments will be used for each net. Then sets of nets with various densities were chosen randomly from the distribution, and an attempt was made to route each set in the channel. Figure 6 shows the results. In a conventional channel, any problem with density 32 or less can always be routed. A high probability of routing in the segmented channel is observed when the density is only three or four below the number of tracks. (Further details of this study are given in [31].)

An asymptotic analysis [29] has confirmed that the number of tracks required in a segmented channel grows linearly with the expected channel density, just as with conventional channels. This is true even for single-segment routing.

IV. Routing Architecture of Act FPGAs

We now describe how segmented routing is applied in the routing architecture of the ACT 2 and 3 FPGA's. Figure 7 shows a simplified view of a four row by two column section of the array of modules. Segmented channels extend horizontally between the rows.

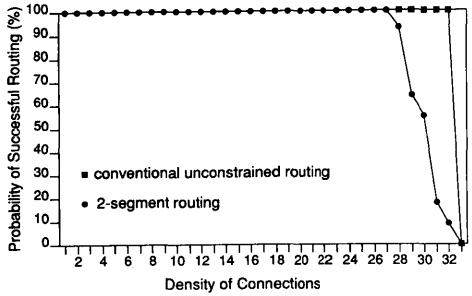


Fig. 6. Probability of successful routing in a 32-track channel vs. density of connections.

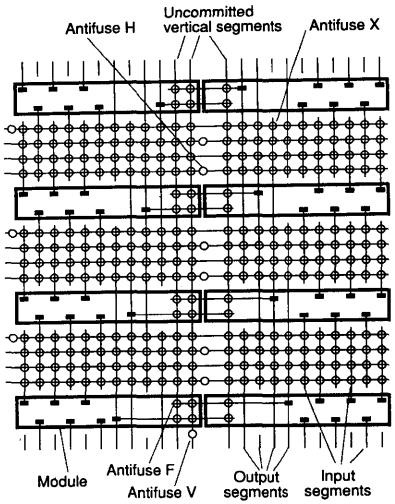


Fig. 7. Detailed view of routing architecture and modules. (Only a representative sample of input and uncommitted segments are shown.)

A module has several inputs, each of which appears on a dedicated vertical input segment. (The figure shows six inputs per module, but the actual number varies according to the function of the module; it is typically eight for a basic logic module.) The input segments each span only one channel in order to minimize their capacitance and the total wiring area.

Each module has a single output, which appears on a dedicated vertical output segment. Output segments span two channels above and two channels below the module. (Those reaching the top or bottom channel may be slightly longer or shorter). In this way a module can distribute its output to more than one nearby channel without the need for inserting more antifuses in the signal path. The added capacitance of the output segment is a small price to pay since the segment is driven directly by the module, not through any antifuses.

Each input or output segment can connect to any of the uncommitted horizontal segments in the channels it crosses through an antifuse such as the one marked X. Antifuses such as the one marked H allow connection of the horizontal segments end-to-end to support multiple-segment routing. The number of tracks per channel varies with the size of the array, according to the need for routing resources. Each channel also contains one full length horizontal segment that is grounded and another tied high so that any input can be programmed to logical 0 or 1.

Uncommitted segments are also provided in the vertical direction. These span several rows and channels, in many cases the full height of the array. Typically either one or two tracks of uncommitted vertical segments are provided per column of modules (as shown in the figure), but again this varies with array size. The uncommitted vertical segments can be joined end-to-end by antifuses such as the one marked V, and thus constitute a vertical segmented channel. However we do not refer to them as a channel since in the actual layout the vertical segments pass over the modules rather than between them (like the "feed-throughs" in a conventional channeled gate array). Each uncommitted vertical segment can connect to the horizontal segments it crosses. Also, special antifuses such as the one marked F connect a module's output and each uncommitted vertical segment passing over the module or its neighbor; two horizontally adjacent modules thus share one set of uncommitted vertical segments. These antifuses are located within the area of the module itself, but are considered part of the routing.

(The earlier ACT 1 architecture is similar to the above description except in the following respects. Additional antifuses permit certain pairs of inputs of a module to be joined; this increases routing flexibility. Also, antifuses such as the one marked F are not provided in ACT 1.)

The routing shown in Fig. 1 is typical of many nets. Note that in order to minimize the number of series RC stages, each channel's horizontal segments are driven directly by the dedicated output segment. This style of global routing is classified as a "Steiner Tree with Trunk" [32]. The chosen output segment length is sufficient to permit this routing to be used for most nets.

Although this favorable routing can be assured for all speed critical nets, another 5–10% of the nets usually must be placed with an input in some channel beyond the reach of the dedicated output segment. In this case, a suitable uncommitted vertical segment is selected to provide an alternate trunk for the channels not reached by the output. The vertical segment is driven by the output through one "F" antifuse, as shown in Fig. 8. Since this antifuse may be called upon to drive nearly the whole capacitive load of a widely dispersed net, the delay is very sensitive to its resistance. By programming these antifuses with higher currents and providing them with additional strapping contacts, the resistance is reduced to about 100 ohms, compared to 600 ohms for the other antifuses. The "F" antifuses thus require greater layout area but since there are few of them the cost is negligible.

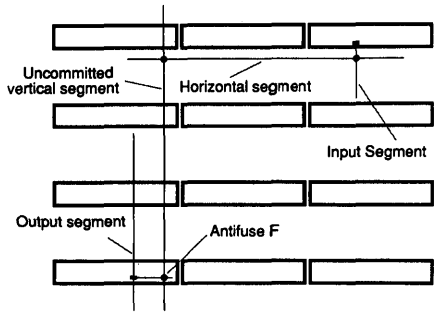


Fig. 8. Routing using directly driven uncommitted vertical segment.

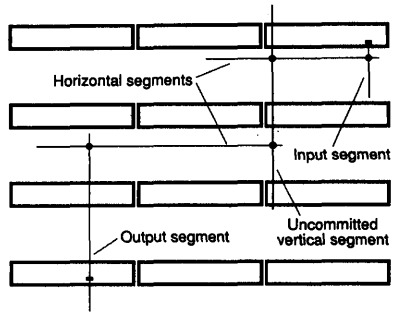


Fig. 9. Routing using indirectly driven uncommitted vertical segment.

In rare instances an uncommitted vertical segment is needed but there is none available that is accessible to the driving module's column and that extends through the required channels. A segment from some other column must be used instead. In this case the uncommitted segment is driven through one of the horizontal channels spanned both by it and the module's output, as shown in Fig. 9. This route involves an extra RC stage, and so it is reserved for nets whose speed is not critical.

Assignment of uncommitted segments in each horizontal and vertical channel to those nets that need them constitutes a segmented channel routing problem. Algorithms for solving these are discussed in [29].

V. Logic Module

FPGA architectures are often categorized by the complexity of their basic logic module, termed *granularity*. A simpler module will have less internal delay and, since each module takes less area, more of them can be provided on the chip. Furthermore, a fine-grained architecture tends to be more flexible. For example, a wide variety of functions can be built with equal efficiency out of two-input NAND gates, but eight-input NAND gates are much better at some functions than others. With simple modules there is also often more than one way to implement a function, allowing beneficial tradeoffs between area and delay.

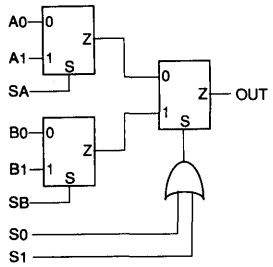


Fig. 10. ACT 1 logic module.

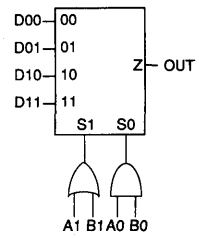


Fig. 11. "C" module.

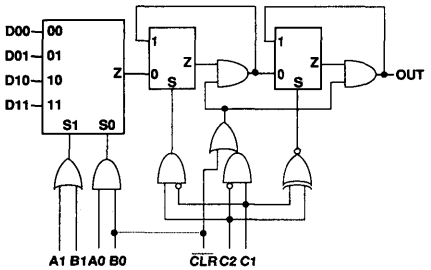


Fig. 12. "S" module. In ACT 2 chips, the B0 and $\overline{CLR}$ inputs are joined.

On the other hand, using a module that is too simple can overburden the routing network. If a function that could be packed into a few complex modules must instead be distributed among many simple modules, more connections must be made through the programmable routing network.

As a rule of thumb, an FPGA should be as fine-grained as possible while maintaining good routability and routing delay for the given switch technology. The module should be chosen to implement a wide variety of functions efficiently, yet have minimum layout area and delay.

Our approach to selecting a module began by collecting statistics on usage of various logic functions, or *macros*, in actual gate array applications. These were then used to evaluate candidates for the function of the module itself. The idea is to choose a module into which the most commonly used macros can be efficiently embedded.[3]

The ACT 1 family uses one general-purpose logic module [12], shown in Fig. 10. The module is composed of three two-to-one multiplexors and a gate. Various macrofunctions (AND, NOR, flip-flops, etc.) can be implemented by applying each input signal to the appropriate module input(s) and tying other module inputs to 0 or 1. The module can implement all combinational functions of two inputs, all functions of three inputs with at least one positive unate input, many functions of four inputs, and others ranging up to eight inputs. AND gates and OR gates up to four inputs wide are accommodated (for one or more combination of inversions at the inputs). In all, 702 distinct combinational macros are possible. Any sequential macro can be configured from one or more modules using appropriate feedback routings. Over a range of designs, an ACT 1 FPGA accommodates just over three gates of logic per module. This value is fairly consistent independent of the ratio of combinational to sequential macros in the design.

For larger chips, we can rely more strongly on the law of averages to keep the fraction of sequential macros in a design within a given range. Furthermore, statistics indicate a significant proportion of the nets driving the data input of a flip-flop have no other fanout. This motivated the use of a mixture of two specialized modules for the ACT 2 and 3 families.

The *C module* (Fig. 11) is a modified version of the ACT 1 module reoptimized to better accommodate high-fan-in combinational macros, notably some five-wide AND and OR gates, though with some loss in ability to build sequential functions. It is composed of a four-to-one multiplexor and two gates, and can implement a total of 766 distinct combinational macros.

The *S module* consists of a front end equivalent to the C module followed by a sequential block built around two latches; Fig. 12 gives a functional diagram. The sequential block can be used as a rising- or falling-edge D flip-flop, or a transparent-high or transparent-low latch, by tying the C1 and C2 inputs to a clock signal, logical zero or logical one in various combinations. For example, tying C1 to 0 and clocking C2 implements a rising-edge D flip-flop. The block can also be set permanently transparent (by tying C1 to 1 and C2 to 0), making the S module equivalent to a simple C module with a small additional delay. Figure 13 shows the function categories that can be implemented using the S module. Toggle or enabled flip-flops can be built using the combinational front end in addition to the D flip-flop. Other less commonly used flip-flops (such as JK or set/reset) not supported by the sequential block can be configured from one or more C or S modules using external feedback connections.

In ACT 2 chips, the clear input is joined to the B0 input of the front end to limit the total number of inputs at some

[3]Given the parameters of antifuse routing, our efforts focused on modules not more complex than a typical gate array macro. Thus statistics from a design targeted for implementation in a gate array were taken to reflect designs targeted at any candidate module. The possibility of simple local optimizations (such as relocating inversions or combining a macro with a subsequent D flip-flop) was accounted for, but nothing beyond that. Ideally one would optimally reorganize the logic of each sample design for the module under consideration using logic synthesis and technology mapping, as has been done in more recent work [33], [34]. Such reoptimization becomes more important for more complex modules.

minor cost in logic flexibility. ACT 3 FPGA's will provide independent clear and B0 inputs for greater regularity, primarily for the benefit of logic synthesis programs.

A chip with an equal mixture of C and S modules provides sufficient flip flops for most designs, plus an extra margin to assure flexibility in placement. Over a range of designs, the ACT 2 mixture provides about 1.4–2.0 times the logic capacity per module of the ACT 1 module.[4]

Figure 14 shows a typical critical path in a state machine implemented using four C modules and one S module. The ability to fit a five-wide gate in one C module saves one routed net compared to an ACT 1 module implementation. The use of the S module eliminates one more routed net (which is now internal to the module in the second stage).

The choice of module also greatly influences the routability of the chip. Because each input is accessible from only one of the two channels adjacent to a module, it would appear that routability is degraded compared to a conventional *double-entry* gate array cell, which a signal can enter from either channel at the convenience of the router. However, there is nearly always more than one way to implement a macro. For an N-input macro, there may be as many as 2^N different assignments of the input signals to the two channels. This corresponds to full double-entry symmetry. Even if not all of the 2^N assignments are possible, a sufficiently sophisticated router can take advantage of whatever flexibility exists.

For the C module, a given signal can be routed from either side of the module an average of 70% of the time (weighted by macro usage), quite sufficient for good routability. This figure is affected by both the module function and the assignment of the module inputs to sides. It is another important criterion in selecting a module function.

Finally, we note that the user is insulated from the details of the module function. Logic is entered as a schematic using a familiar gate-array-style macro library or a logic synthesis program. Software automatically determines how each module should be connected to realize the desired macro. (Methods of design entry are described more fully in Section VIII.)

VI. Input/Output and Clock Distribution

Each I/O pad has an adjacent bidirectional buffer, which connects to the array through an associated I/O module. These modules fit in the outer columns and rows of the array next to the logic modules, and interface to the routing channels in the same way. Each I/O module has inputs for outgoing data and enable, which are sent to the buffer, and an output for incoming data which is driven from the pad.

[4] It is interesting to note the related work of Rose *et al.* [33]. They found that the addition of a D flip-flop to a module increased the logic capacity by a factor of 1.4–2.3. The results are not exactly comparable, however, since their model differs from ours in several respects: their module is a three- or four-input look-up table; they add flip-flops to all modules rather than to half of them; our results include a slight change in the combinational capabilities of the module as well.

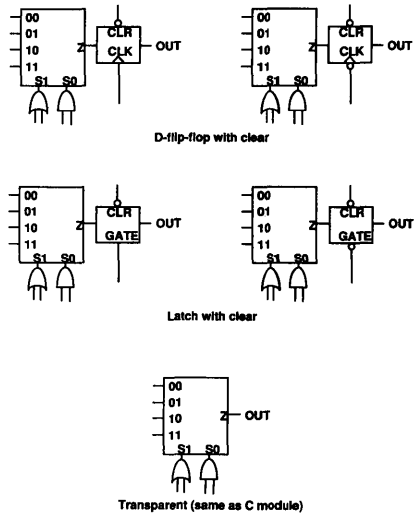


Fig. 13. "S" module function categories.

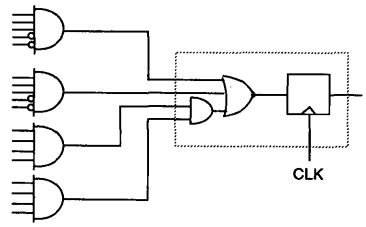


Fig. 14. Sample critical path in an ACT 2 FPGA.

Thus the I/O module can be programmed to provide input, output, tristate or bidirectional capability.

In order to keep clock-to-output delay to a minimum, the ACT 2 family has a dedicated transparent-high latch for each pad, which can serve as the slave stage of a flip-flop. The latch is controlled by a gate input to the I/O module. If flow-through operation is desired, the gate is simply tied high to make the latch transparent. A dedicated transparent-low latch is similarly provided on each input path. The polarities of the input and output latches are chosen so they can be combined with each other, and possibly with other internal latches or flip-flops, to form a path that is functionally equivalent to a chain of rising-edge flip-flops.

Clock signals present unusual requirements: they have high fanout and yet allow only minimal delay and skew. Rather than attempt to route clocks like any other signal, dedicated clock distribution networks are provided. Each network is driven directly from a special input pad for high speed. The signal passes through a buffer tree, and appears on a dedicated full-length horizontal track in each channel. Thus the network reaches every logic and IO module in the

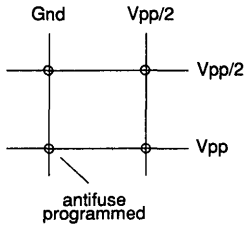


Fig. 15. Application of programming voltage.

array. The ACT 2 family provides two clock distribution networks. Each network can be driven from user-defined internal logic as well as from an input pad.

To minimize the capacitive load on the clock network, antifuses are provided to connect the clock track to only certain module inputs, specifically the clock inputs of the S and IO modules and a subset of the combinational inputs on all the logic modules. Skew is further reduced by having the automatic placement program attempt to balance the loading on each branch of the distribution tree.

The clock inputs to the S module, like other inputs, can also be connected to the uncommitted horizontal segments. This accommodates designs with several local asynchronous clocks routed in the normal fashion.

VII. Programming and Testing

One of the great puzzles in the development of antifuse FPGA's was to find an efficient way to address each of the two-terminal antifuses uniquely. Diodes in series with each antifuse would allow unique addressing, but block signal flow. Early schemes required the use of an individual control line for each routing track, doubling the number of lines required in each direction [21]. However, methods for programming and testing antifuse FPGA's that use only a few control lines for an entire channel have been developed [12], [14]. Although a full description is beyond the scope of this paper, the following explanation conveys the basic concepts.

Consider an array of antifuses at the intersection of some horizontal and vertical segments, as shown in Fig. 15. An antifuse is programmed by applying a programming voltage, Vpp, across it. This is done by precharging all segments to an intermediate voltage of about Vpp/2. Then a selected vertical segment is grounded and a selected horizontal segment is driven to Vpp. Other segments are left floating at Vpp/2. Only the single antifuse at the intersection of the selected segments sees the full Vpp.

We now show how the segments can be selectively driven. Figure 16 illustrates the *series pass transistor addressing* method. A sample of horizontal and vertical segments are shown. Each pair of adjacent segments in the same track is connected by a pass transistor. (In tracks containing uncommitted segments there is already an antifuse joining adjacent segments, and so the transistor is hooked in parallel with the antifuse.) These transistors are

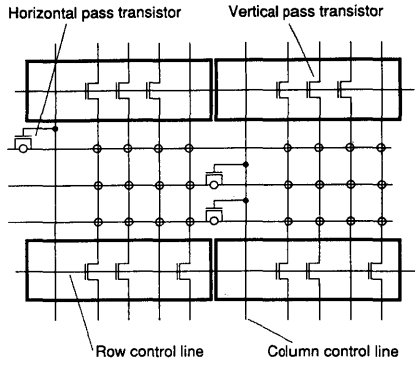


Fig. 16. Pass transistor addressing.

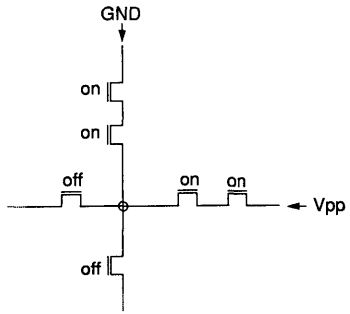


Fig. 17. *Programming an antifuse between a horizontal and a vertical segment.*

used only for programming and testing and are all shut off during normal operation of the programmed part. The transistors in each row of modules are gated by a control line running along the row, and the transistors in each column of modules are gated by a control line running along the column. These lines in turn are driven by special logic at the periphery of the array. Note that only one control line per row or column is required, regardless of the number of tracks.

Figure 17 shows how this circuitry is used to program an antifuse at the intersection of a horizontal and vertical segment. The vertical track containing the antifuse is driven to ground from one end by special logic at the periphery of the array. The pass transistors in all rows from the driving periphery to the antifuse are turned on. The horizontal track is driven to Vpp in a similar manner. Figure 18 shows how an antifuse between two adjacent segments in the same track is programmed. All columns of pass transistors except the one bypassing the antifuse are turned on, and the track is grounded at one end and driven to Vpp at the opposite end.

In certain cases *direct addressing* circuitry offers a favorable alternative to series pass transistors. Figure 19 shows how this method can be used to address horizontal seg-

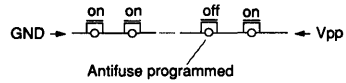


Fig. 18. Programming an antifuse between two adjacent segments in the same track.

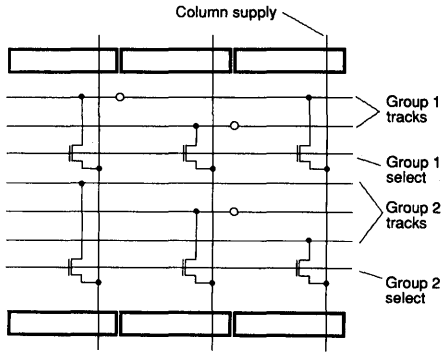


Fig. 19. Direct addressing circuitry.

ments. Column supply lines run vertically through the array. Each segment along a track can be connected through its own addressing transistor to a supply line. These transistors are gated by a horizontal select line. Each select line serves all the segments in a group of one or more adjacent tracks in the same channel. Activating one supply line and one select line uniquely addresses a particular horizontal segment. The number of segments in a channel that can be addressed is limited by the number of supply lines times the number of group select lines. It follows from this that the ratio of tracks to select lines is at most the average segment length. Also note that the supply path from the periphery to the segment includes only one transistor, rendering the programming current independent of the position of the segment and the segmentation of its track. Thus the direct addressing method is most efficient for irregularly segmented channels with long segments.

ACT 1 chips use only the pass transistor method. ACT 2 and 3 chips use the pass transistor method for the vertical tracks, which contain many short input segments, and the direct address method for the horizontal tracks.

In either scheme, some care is required to assure that a unique antifuse is addressed once other antifuses have already been programmed. Figure 20 gives an example of how improper addressing might allow programming current to divert along a *sneak path* through previously programmed antifuses 3 and 4, programming antifuse 1 instead of the intended antifuse 2. Fortunately, we are not interested in programming an arbitrary pattern of antifuses (an FPGA is not a PROM!). For example, we are not called upon to program a pattern connecting two outputs together since this does not form a useful net. For the relevant patterns, it can be proved that there is always an order

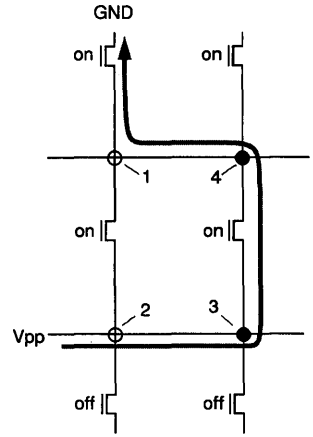


Fig. 20. A sneak path. The arrow shows the path of programming current through antifuse 1, bypassing antifuse 2.

in which the antifuses can be programmed such that sneak paths *never* occur.

Special care is also required to protect the module circuitry from the voltages present on the segments during programming. Transistors that are in contact with routing segments must be specially designed to withstand the programming voltage.

Next we turn to testing. This takes place in three phases: before, during and after programming. Preprogramming tests check for shorted or open segments, shorted or even weak antifuses, and proper module and I/O operation. Continuity of the segments is easily verified by turning on all pass transistors, using the peripheral circuits to drive the segment at one end of a track and read the segment at the other end. Testing for the absence of shorts between segments in adjacent tracks is done in a similar way by applying a pattern of alternating zeros and ones. Weak antifuses can be screened out by applying the proper stress voltage (higher than normal operating voltage but lower than Vpp) across groups of antifuses in parallel using the programming circuits. Breakdown of an antifuse is detected by passage of excessive current.

To verify the functionality of the modules, we need to apply test vectors to their inputs and read their outputs. A vector can be applied simultaneously to all modules by turning on all vertical pass transistors and applying the vector to the vertical tracks from the peripheral logic. A simple row-select and column-sense scheme conveys the output of each module in turn to an output pad for monitoring. (This same circuitry allows users to probe internal nodes during normal operation, as described in the next section).

Proper closure of the programmed antifuses is verified during programming by sensing the passage of programming current. A complete test for unintended connections between any two segments can be done after programming

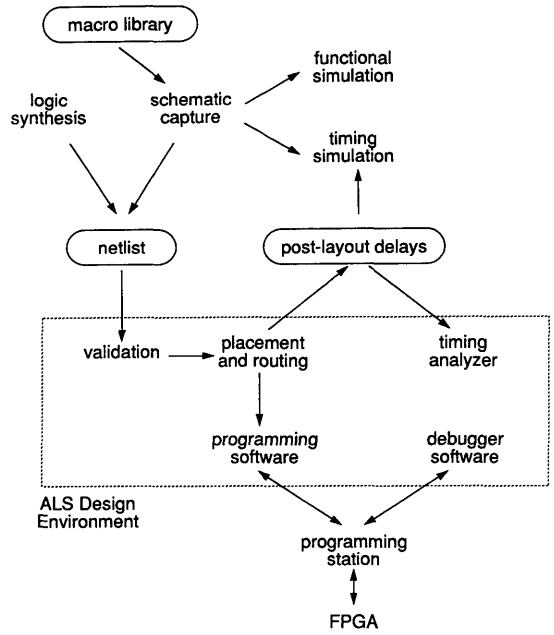


Fig. 21. Design process.

using the programming circuitry to precharge, drive and read the segments. This is true despite the fact that it is no longer possible to address each individual antifuse uniquely once programming commences. The solution to this paradox is that detection (though not location) of shorts can be accomplished simultaneously for many antifuses in parallel.

Taken together, these tests insure correct functioning of the programmed part. (Further details of testing in ACT 1 FPGA's are found in [35].)

VIII. Design Process and Software

Figure 21 diagrams the process of embedding a design in an ACT FPGA. This process begins with capture of the design in a computer readable format. Currently, most users enter their designs as schematics built of macros from a library. Any of several standard schematic capture programs can be used. Since the ACT 1 and C modules have about the same complexity as a typical gate array library macro, the problem of selecting which groups of macros should share one module is avoided; each macro is usually assigned its own module. The larger capabilities of the S module are handled by software that automatically combines a flip-flop and a preceding combinational macro into one S module where possible. This process is transparent to the user and does not require modifying the schematic.

Another increasingly popular alternative is to enter designs in terms of Boolean equations, state machine descriptions, or functional (rather than structural) schematics. Different portions of a design can be described in different ways, compiled separately, and the results merged according to a top-level hierarchical schematic. Various industry standard formats are supported.

High-level synthesis tools such as MIS-II [36] or Synopsys [37] can be used with ACT FPGA's by providing a suitable library. This can encompass only the standard schematic library macros, or the complete set of all 700 or so functions embeddable in the module can be included. Recent research has investigated other more efficient ways to allow synthesis tools access to the full flexibility of the module. A method for rapid searching of large libraries using Boolean matching is given in [38]. Other approaches take advantage of the multiplexer structure of the module, using either binary decision diagrams [39] or "if-then-else DAG's" [40].

Several guidelines are suggested for reliable logic design with FPGA's. The general goal is to make proper circuit function independent of shifts in timing from one part to the next.

- Use synchronous logic design where possible.
- Avoid gated clocks, using enabled flip-flops instead.
- Avoid race conditions.
- Limit fanout by buffering.

These guidelines are the same as those for users of mask programmed gate arrays [41]. In addition, designs for ACT

	logic module utilization	pins per logic module
8000 gate FPGA		
a) design done in an 8K TI gate array	99.5%	4.28
b) 32 bit data path, 16x16 mult, state machine	99.4	4.30
c) 2901 ALU (x4)	98.1	4.57
d) DMA controller (x3)	97.1	3.99
e) asynchronous serial ECC	97.0	4.78
f) pipelined fixed point mult, div, sqrt	94.5	3.37
g) state mach., mult/add, datapath, counter	92.7	4.34
h) color crt controller (x3)	87.3	3.83
i) 32 bit data path with sum, compare (x3)	86.8	5.25
j) 40 bit floating point adder/subtractor	86.7	4.33
4000 gate FPGA		
k) 16 bit datapath, 16x16 mult, state machine	98.1	4.86
l) 2901 (x2)	93.2	4.68
m) DRAM, DMA & SCSI controllers, UART	92.6	4.73

Fig. 22. Statistics for some placed and routed designs.

FPGA's must use multiplexers rather than tristate drivers since internal tristate is not supported.

Once the design has been entered, it can be simulated either functionally or using prelayout delay estimates. The netlist is checked by a validation program for problems such as undriven nets, outputs shorted together, excessive fanout, etc.

The next step is to map the netlist into the ACT architecture. The placement problem (selecting a module for each macro) is similar to that for a conventional gate array, with a few additional considerations. I/O pins not constrained by the user must be assigned to locations that minimize delay and routing congestion inside the chip. Macros must be placed in modules of the appropriate type (C or S). Macros hooked to a clock network should be distributed so as to balance the load on the network's branches. Routing congestion within each horizontal channel must be limited. Whenever possible, the dedicated output segments should be used in preference to the uncommitted vertical segments, especially for nets identified by the user as speed critical. Uncommitted vertical segments must be assigned to any remaining nets. Routing of the segmented channels is done as previously described. The automatic placement and routing software takes 45–60 min to complete an 8000-gate FPGA on a 68030 microprocessor workstation.

Once placement and routing are completed, the propagation delay to each input pin is estimated. The calculation accounts for the module internal delay as well as the delay through each RC stage in the routed net. The estimates are about as accurate as what would be obtained with a SPICE circuit simulation. The delays can be back-annotated to a simulator, or checked with a static timing analyzer program.

A list of antifuses to be programmed is generated and downloaded to a programming station into which the FPGA is plugged. Programming time is about 5–10 min, depending on the size of the FPGA and the design. The station allows up to four chips to be programmed simultaneously in this time. The standard tests applied before, during and after programming, described in Section VII, assure correct function of the programmed part. No design-specific test vectors are required from the user.

The programming system can also access the previously described test circuitry inside the programmed FPGA, causing the FPGA to sense any selected module output and present it on an external pin in real time. This virtual probe can be used and its address changed even as the chip is operating in the user's system. The probe provides a useful back-up when simulation is difficult or impossible, such as for highly asynchronous designs or when the design must accommodate a poorly documented interface. Debugger software is also provided to enable the programmer to be used as a functional tester, presenting stimuli at the FPGA's pins and reading data back via the pins and the probe.

IX. Density and Performance

In this section we report results with various design examples to convey the density and performance of the ACT 2 architecture.

The logic capability of an FPGA depends on more than the number and capacity of it modules. The fraction of the modules that can be used without running out of routing capacity must also be considered. A simple design such as one long shift register can use all modules with very little routing, but realistic designs are more demanding.

Figure 22 summarizes results for a variety of designs in 8000- and 4000-gate FPGA's having 1232 and 649 logic modules, respectively. The designs were placed and routed automatically with no manual intervention. The table includes the number of routed pins per logic module to demonstrate the realistic burdens the designs impose on the routing. Nearly all designs with module utilization under 85%, most designs with utilization under 95%, and many with utilization up to 100%, route completely.

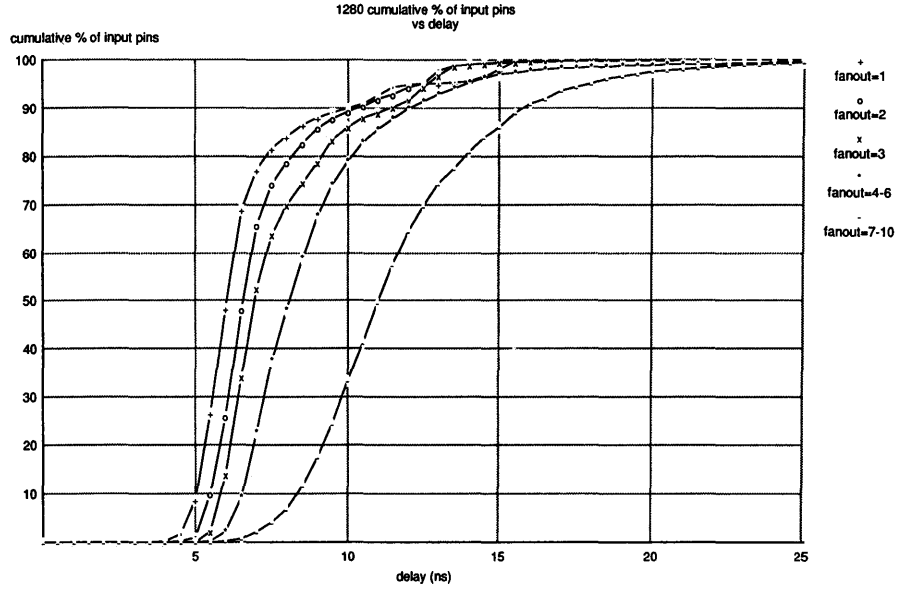


Fig. 23. Cumulative distribution of pin delays grouped by the fanout of their net. Delays were computed by the postlayout extraction program for conditions of 5V, 25C, and worst-case processing. Data based on 29 designs in an 8000-gate FPGA (A1280–1). The designs were automatically placed and routed without manual intervention.

How successful is the architecture in controlling routing delay? Figure 23 shows the distribution for the routing delay to each input pin for various ranges of net fanout. The figure includes all nets in the design. Pins on nets identified by the user as speed critical would generally be among the fastest 30%, and nearly always among the fastest 90%. Especially for low fanouts and critical nets, the routing delay distributions are tight, offering the user predictable delays.

Finally, we give some results regarding some basic circuit blocks. Table 1 shows the number of C and S modules used, number of levels of modules in the critical path, and nominal speed at 5V and 25C including routing delay. The counters are binary, with synchronous parallel load and asynchronous clear. The accumulator consists of an adder feeding into a register with asynchronous clear. Area-delay tradeoffs are possible using different types of adders. The state machine handles arbitration for a DRAM controller in a 68030 microprocessor system. It has 3 state bits.

Table 1. Implementations of Some Typical Subcircuits

Example	Modules	Levels	Clock period (nsec)
8-bit counter, 1-cycle load	16	2	17
16-bit counter, 1-cycle load	50	2	18
8-bit counter, 2-cycle load	40	1	12
16-bit counter, 2-cycle load	80	1	12
8-bit ripple-carry accumulator	17	8	60
8-bit carry-select accumulator	40	3	28
16-bit carry-select accumulator	89	3	33
state machine	16	2	16

X. The Practical Impact of Antifuse FPGA's

The major impact of FPGA's is the ability to get a design into production quickly and cheaply, even if iterations are required. This is demonstrated in the following case history from commercial practice.

Example 1: A digital video signal processing system was designed for the consumer market. Four 2000-gate FPGA's were used to integrate registers, multipliers, adders and digital phase-locked-loops; the pipelined data paths are 12–16 bits wide, and operate at a 16MHz clock rate. It is estimated that the use of FPGA's saved two to three months in product development time. Subsequently, it became necessary to add

features to the product. A single 8000-gate FPGA replaced the four 2000-gate FPGA's and provided the new circuitry.

A surprising fraction of conventional gate array designs (estimates indicate about half) never reach high-volume production. This may be because either the lifetime of the product is short or because new features or other changes are required. In such cases use of FPGA's for all production saves the nonrecurring engineering costs associated with developing a mask programmed gate array.

For the lucky few whose products are required in high volume, the FPGA design can be transferred to a conventional gate array. Due to the small granularity of the module, an antifuse FPGA netlist can be efficiently converted to a gate array library by substituting the appropriate realization for each macro in the FPGA library. Test vectors can often be generated automatically. Several vendors support this conversion path. The design in Example 1 was eventually converted to an 8000-gate conventional array.

Even when designs require the greater speed or density of conventional arrays, use of FPGA's for functional prototyping accelerates product development. This was true in the next example.

Example 2: A multiprocessor file server for a PC network was under development. The system employed multiple 80486 microprocessors, banks of DRAM, and multiple 64- and 32-bit busses. Several 8000-gate FPGA's implemented the random logic in a functional prototype using synchronous logic design. Since the logic was composed of clearly defined subblocks, principally state machines and simple datapaths, partitioning of the logic among the FPGA's was easy. For this complex system, the design process was incremental and somewhat experimental. Because it was anticipated that the design would be modified or extended several times, a fast design flow was necessary. The logic for each FPGA was synthesized with a Synopsys hardware description language (HDL) compiler using an Actel macro library, and then checked with an HDL simulator. Each design was automatically placed and routed, and resimulated with postlayout delays. This methodology allowed software to be developed and tested concurrently with the completion of the logic design.

We conclude with some comments on the future evolution of antifuse FPGA's. Research into antifuse technologies is gaining momentum due to the commercial importance of FPGA's. Reductions in the parasitic resistance or capacitance will help improve speeds. Reductions in the programming voltage or current will reduce the overhead area required for pass transistors and other programming circuitry, and hence cost. On the architecture front, better logic modules and improved methods of designing segmented routing channels may be developed. FPGA architectures could be specialized for certain classes of applications. Of course, antifuse FPGA's will also directly benefit from anticipated improvements in the underlying CMOS technology. Thus one would expect antifuse FPGA's to evolve more rapidly than conventional arrays, reducing the current gap between them. Ultimately it is possible that FPGA's will supplant conventional gate arrays for application-specific logic to the same extent that EPROM's have supplanted ROM's for application-specific memory.

ACKNOWLEDGMENT

The authors would like to thank C.-L. Chan, R. Gopisetty, S. Kaptanoglu, D. McCarty, W. Miller, W. S[illegible]u, and T. Whitney for their help in preparing data for this paper; J. Birkner for his kind discussions with the authors regarding the programming currents for the resistances reported in [22]; and A. Haines, D. How, J. Schlageter, and the reviewers for their suggested improvements to the manuscript.

REFERENCES

[1] G. Worchel, "Analysis of the CMOS gate array merchant market," *In-stat Services*, Mar. 1991.
[2] "Programmable logic user study," *Electronic Eng. Times*, May 1990.
[3] J. F. Smith, *et al.*, "Laser-induced personalization and alterations of LSI and VLSI circuits," in *Proc. 1st Int. Laser Processing Conf.*, Anaheim, Calif., Laser Institute of America, Nov. 16, 1981.
[4] D. Allen and R. Goldenberg, "Design aids and test results for laser-programmable logic arrays," in *Proc. Int. Conf. Computer Design*, pp. 386–390, 1990.
[5] W. Carter *et al.*, "A user programmable reconfigurable gate array," in *Proc. Custom Integrated Circuits Conf.*, pp. 233–235, 1986.
[6] S. Wong, H. So, J. Ou, and J. Costello, "A 5000-gate CMOS EPLD with multiple logic and interconnect arrays," in *Proc. Custom Integrated Circuits Conf.*, pp. 5.8.1–5.8.4, 1989.
[7] L. Gerzberg. U.S. Patent 4,590,589, 1986.
[8] E. Hamdy, J. McCollum, S. Chen, S. Chiang, S. Eltoukhy, J. Chang, T. Speers, and A. Mohsen, "Dielectric based antifuses for logic and memory ICs," *IEDM Tech. Digest*, pp. 786–789, 1988.
[9] "ICC takes the antifuse one step further," *Electronics Mag.*, p. 87, Mar. 1989.
[10] R. Whitten, R. Bechtel, M. Thomas, H.T. Chua, A. Chan, and J. Birkner, European Patent Application No. 90309731.9, May 9, 1990.
[11] J. Rose, A. El Gamal, and A. Sangiovanni-Vincentelli, "A classification and survey of field-programmable gate array architectures," *Proc. IEEE*, vol. 81, no. 7, July 1993..
[12] A. El Gamal, J. Greene, J. Reyneri, E. Rogoyski, K. El-Ayat, and A. Mohsen. "An architecture for electrically configurable gate arrays." *IEEE J. Solid-State Circuits*, vol. 24, no. 2, pp. 394–398, Apr. 1989.
[13] K. El Ayat, *et al.*, "A CMOS Electrically Configurable Gate Array," *IEEE J. Solid-State Circuits*, Vol. 24, No. 3, pp. 752–762, June 1989.
[14] M. Ahrens, *et al.*, "An FPGA family optimized for high densities and reduced routing delay," in *Proc. Custom Integrated Circuits Conf.*, 1990.
[15] B. Osann and A. El Gamal. "Compare ASIC capacities with gate array benchmarks," *Electronic Design*, pp. 93–98, Oct. 13, 1988.
[16] U.S. Patent 2,784,389, 1957.
[17] B. Roesner, U.S. Patents 4,424,579 and 4,442,507, 1984.
[18] Holmberg, *et al.*, U.S. Patents 4,499,557, 1985 and 4,599,705, 1986.
[19] Lim *et al.* ,U.S. Patent 4,569,121, 1986. Stacy et al. U.S. Patent 4,569,120, 1986.
[20] H. Stopper, *et al.* ,U.S. Patent 4,847,732, 1989.
[21] H. Graham and D. Seltz. "Electronically programmable gate array having programmable interconnect lines," U.S. Patent 4,786,904, Nov. 22, 1988.
[22] J. Birkner, A. Chan, H.T. Chua, A. Chao, K. Gordon, B. Kleinman, P. Kolze, and R. Wong, "A very high-speed field programmable gate array using metal-to-metal antifuse programmable elements," in 1991 IEEE Custom Integrated Circuits Conf., San Diego, CA, May 12, 1991.

[23] S. Chiang, R. Wang, J. Chen, K. Hayes, J. McCollum, E. Hamdy, and C. Hu, "Oxide-nitride-oxide antifuse reliability," *Int. Reliability Physics Symp.*, pp. 186–192, Mar. 1990.
[24] S. Chiang and K. Hayes, Act 1010/1020 Reliability Report, Actel Corporation, Sunnyvale, CA, April 1990.
[25] Preliminary data from a military system manufacturer. Test done on total dose Gamma Irradiation Survivability Test.
[26] A. Hashimoto, J. Stevens, "Wire routing by optimizing channel assignment within large apertures," in *Proc. 8th IEEE Design Automation Workshop*, 1971.
[27] M. Lorenzetti and D. Baeder, "Routing," in *Physical Design Automation of VLSI Systems*,B. Preas and M. Lorenzetti, Eds. Benjamin Cummings, Chapter 5, 1988.
[28] C. Marr, "Logic array beats development time blues," *Electronic System Design Mag.*, pp. 38–42, Nov. 1989.
[29] A. El Gamal, J. Greene, and V. Roychowdhury, "Segmented channel routing is nearly as efficient as channel routing (and just as hard)," in *Proc. Conf. Advanced Research in VLSI*, Santa Cruz, CA, Mar. 1991.
[30] A. El Gamal, "Two dimensional stochastic model for interconnections in master slice integrated circuits," *IEEE Trans. Circuits Syst.*, CAS-28, pp. 127–138, Feb. 1981.
[31] J. Greene, V. Roychowdhury, S. Kaptanoglu, and A. El Gamal, "Segmented channel routing," in *Proc. ACM/IEEE Design Automation Conf.*, June 1990.
[32] B. Preas and P. Karger, "Placement, assignment and floorplanning," in *Physical Design Automation of VLSI Systems*, B. Preas and M. Lorenzetti, Eds. Benjamin Cummings, Chapter 4, 1988.
[33] J. Rose, R. Francis, D. Lewis, and P. Chow, "Architecture of programmable gate arrays: the effect of logic block functionality on area efficiency," *IEEE J. Solid State Circuits*, vol. 25, no. 5, pp. 1217–1225, Oct. 1990.
[34] J. Kouloheris and A. El Gamal, "FPGA performance versus cell granularity," in *Proc. Custom Integrated Circuits Conf.*., pp. 6.2.1–6.2.4, May 1991.
[35] K. El Ayat, A. Haines, and K. Hayes, "Testing antifuse-based FPGAs," *Electronic Eng. Times*, May 15, 1989.
[36] R. Brayton, R. Rudell, A. Sangiovanni-Vincentelli, and A. Wang, "MIS: A multiple-level logic optimization system," *IEEE Trans. CAD*, Nov. 1987.
[37] R. Rudell and R. Segal, "Logic synthesis can help in exploring design choices," *1989 Semicustom Design Guide*, CMP Publications, Manhasset, NY.
[38] S. Ercolani and G. de Micheli, "Technology mapping for electrically programmable gate arrays," in *Proc. 28th ACM/IEEE Design Automation Conf.*, pp. 234–239, 1991.
[39] R. Murgai, Y. Nishizaki, N. Shenoy, R. Brayton, and A. Sangiovanni-Vincentelli. "Logic synthesis for programmable gate arrays," in *Proc. 27th ACM/IEEE Design Automation Conf.*, 1990.
[40] K. Karplus, "Amap: a technology mapper for selector-based field-programmable gate arrays," in *Proc. 28th ACM/IEEE Design Automation Conf.*, pp. 244–247, 1991.
[41] *HCMOS Gate Array Databook and Design Manual*, LSI Logic Corp., Oct. 1986.

Jonathan Greene (Member, IEEE) received the Sc.B. degree in biology from Brown University in 1979, and the Ph.D. degree in electrical engineering from Stanford University in 1983.

He then worked on various aspects of computer-aided IC design at Hewlett-Packard Laboratories and LSI Logic Systems Research Lab. From 1986 to 1990 he was with Actel Corporation, where he helped develop the architecture and software for their field-programmable gate arrays and became Director of System Architecture. He is currently with BioCAD Corp., Mountain View, Calif., applying computer-aided design and information theoretic techniques to the process of drug discovery.

Esmat Hamdy (Senior Member, IEEE) was born in Cairo, Egypt, of February 2, 1950. He received the B.Sc. and M.Sc. degrees from Cairo University in 1971 and 1975, respectively, and the M.A.Sc. and Ph.D. degrees from the University of Waterloo, Ontario, Canada, in 1977 and 1980, respectively, all in electrical engineering.

From 1971 to 1975 he served an Instructor in the Department of Electrical Engineering, Helwan University, Helwan, Egypt. From 1975 to 1980 he was a Research and Teaching Assistant in the Department of Electrical Engineering, University of Waterloo, where he was engaged in the analysis, modeling, and design of high-density bipolar and MOS integrated structures for LSI/VLSI technologies. He joined Intel Corp., Aloha, Oreg., in 1981, where he was a Senior Staff Engineer/Project Manager in the Technology Department involved in device physics and circuit design of the world first 256K CMOS DRAM and submicrometer CMOS for Microprocessors. In Actel he managed and coinverted the PLICE antifuse technology used in the world's first antifuse FPGA. He has authored or coauthored over 20 papers on LSI/VLSI circuits including contributions to IIL, single-device-well (SDW) MOSFET's CMOS latch-up and PLICE antifuse. He is the winner of the Best Student paper at the 1979 IEDM meeting. He also has 10 pending or granted patents.

Sam Beal received B.S., M.S., and Ph.D. degrees in electrical engineering from Southern Methodist University, Dallas, Tex., in 1975, 1977, and 1979, respectively, having done research in AlGaAs solar cells.

He joined Texas Instruments in 1980, working in the CMOS Technology Department, where his responsibilities included CMOS process development and ASIC design development. In 1984 he assumed responsibility for Texas Instruments' ASIC design center in Santa Clara, Ca. He joined Actel in 1988 as Applications Engineering Manager. He is currently Applications and Product Planning Manager for Development Systems.

Oxide-Nitride-Oxide Antifuse Reliability

Article Reprint

Oxide-Nitride-Oxide Antifuse Reliability

Steve Chiang, Roger Wang, Jacob Chen, Ken Hayes, John McCollum, Esmat Hamdy, Chenming Hu*

Actel Corp. 955 E. Arques Ave. Sunnyvale, CA 94086
Phone:(408)739-1010

*Department of Electrical Engineering and Computer Science U.C. Berkeley, Berkeley, CA 94720

Abstract

Compact, low-resistance oxide-nitride-oxide antifuses are studied for TDDB, program disturb, programmed antifuse resistance stability, and effective screen. ONO antifuse is superior to oxide antifuse. No ONO antifuse failures were observed in 1.8 million accelerated burn-in device-hours accumulated on 1108 product units. This is in agreement with the 1/E field acceleration model.

Introduction

Field programmable gate arrays has been a fast growing field only recently [1]. The key to their configurability is the development of a programmable interconnect element. This element should have small area, low post programming resistance, and be reliable. Many known interconnect elements have been used, including SRAMs, EPROMs, and EEPROMs. Problems encountered using these elements are: large area, high resistance, or inefficient utilization due to circuit complexity. The antifuse approach, however, has some unique and attractive features. Since it is only a two terminal device, the area required is small, and the simple two terminal resistor structure allows simple and efficient routing schemes [2]. The programmed antifuse has very low resistance. It was found that oxide-nitride-oxide (ONO) antifuses have a lower and tighter resistance distribution than that of oxide antifuses (Fig. 1). The choice of antifuse material has further improved both the yield and the reliability over that of oxide antifuses. In addition, ONO is highly radiation resistant. Initial evaluation results indicate that products containing ONO antifuses can withstand 1.5 million rads [3]. The technology and performance characteristics of the ONO antifuse has been previously described [4]. In this paper, we will report the reliability characterization of the ONO antifuse.

We will discuss three different types of antifuse reliability. The first is that the unprogrammed antifuse has to survive a 5.5V 40 year operating condition. The second is that during programming, all unprogrammed antifuses are subject to a momentary stress of half the programming voltage (Vpp/2). The programming yield is required to match or exceed PAL yields which are in excess of 99%. The third is that programmed antifuses should have a very low resistance, which will not increase in value over the life of the part. As will be shown below, the unprogrammed antifuse is reliable, well in excess of the 40 year lifespan, the programming yield is excellent, and the programmed antifuse is not subject to any measurable electromigration. The weakest link in the technology is not the antifuse, but typical CMOS process limitations.

Antifuse Structure

The ONO antifuse is sandwiched between N+ diffusion and N+ poly-silicon gate to form a very dense array with density limited by metal pitches (Fig. 2). A thin layer of oxide is thermally grown on top of the N+ surface, followed by LPCVD nitride, and the re-oxidized top oxide. The target electrical thickness of the combined layer is equivalent to 9nm of silicon dioxide.

TDDB of Unprogrammed Antifuses at 5.5V

For the sub 10 nm ONO thickness, time-dependent-dielectic-breakdown (TDDB) reliability over 40 years is an important consideration. The very first task in determining the feasibility of the antifuse was to examine its TDDB reliability. Typical electrical field and temperature accelerated tests were done in order to extrapolate the dielectric lifetime under normal operating conditions. Based on the oxide study [5], it was reported that there may be different field dependencies of lifetime in high field (>6MV/cm) and in low field (<5MV/cm) regimes. In the case of ONO antifuses, the 5.5V operating field is already over 6MV/cm. The extrapolated data from the high field regime was therefore assumed accurate. This assumption was later confirmed with device burn-in data.

Field Acceleration (E vs 1/E model for ONO)

200X200 um^2 ($0.04mm^2$) area capacitors were packaged and then stressed at different voltages. ONO thickness ranging from 8nm to 9.5nm were studied. The test splits and sample sizes are summarized in Table 1. The TDDB distribution at each voltage condition is shown in Fig. 3.

In the literature, the oxide intrinsic lifetime has been observed to have an exp(1/E) dependence, which is explained mainly with the Fowler-Nordheim tunneling mechanism [6]. Oxide Log(I) curves and lifetime Log(t_{50}) curves exhibit a linear function of 1/E behavior. On the other hand, nitride Log(I) has been shown to follow the Frenkel Poole behavior ($\sqrt{E}$) [7]. Log(I) of ONO is not a linear function of 1/E (Fig. 4a). Rather, it more closely follows E

4

(Fig. 4b). Also, several studies have fitted lifetime of ONO to an E model [8,9].

Nevertheless a careful examination of our data revealed that TDDB lifetime of ONO follows the 1/E model (Fig. 5) better than the E model (Fig. 6). This is in agreement with conclusions from one study [10], but in contradiction with others which did not examine the fit between data and the 1/E model [8,9]. Based on our observation, we found that the E model can fit the data well over 4 to 5 orders of magnitude of time span. However, as time span increases to 7 orders of magnitude, the E model is clearly inadequate (Fig. 6).

Since there is no theoretical basis for ONO to follow the 1/E model or the E model, we tried a statistical approach to find out which model can best fit the data. First, the data is fitted to different field dependent models of $\exp(E^n)$ with n ranging from -1.5 to 1 at 0.5 intervals. Then the correlation coefficient is compared for different models in Fig. 7. The residual comparison is shown in Fig. 8. Again, the E model (n=1) turns out not to be a good fit for the data. The best fit appears to be n = -.5 or -1. This seems to suggest that ONO behavior is similar to oxide (n = -1). But, the addition of nitride (n=0.5) has changed the n to between -0.5 to -1. Which of the two exponents, n=-0.5 or -1, should be used may depend on the ONO processing conditions. The difference between extrapolated lifetime based on these two models is not nearly as dramatic as the choice between E and 1/E. At 5.5V, the difference in the extrapolated lifetime between n= -0.5 and -1 is one order of magnitude in time. On the other hand, the difference between n=1 and -1 is 5 orders of magnitude. In the subsequent analysis, we will use 1/E model exclusively for simplicity. The conclusion reached will not change much if the $1/\sqrt{E}$ model were to be used.

Besides the 0.04mm^2 area capacitor data, we also did a TDDB study on single antifuses (3.2um^2) and ACT 1010 product antifuse arrays (0.36mm^2). Results are shown in Fig. 9. Again, the data follows the 1/E model well for all different area sizes.

Temperature Acceleration

The temperature effect on the 0.04mm^2 ONO area capacitor lifetime is shown in Fig. 10. The activation energy as a function of the electrical field is shown in Fig. 11. A field dependent activation energy has been reported for oxide TDDB lifetime, as well.

For the 5.5 volt lifetime estimate, the activation energy is close to 0.9eV (1/E model). Using this estimate and the product TDDB defect distribution (Fig. 12), the 1% failure lifetime at 5.5V is well over 40 years. The projected product antifuse failure rate (containing 100K to 200K antifuses) is less than 50 FITS at 125°C.

Program Disturb and Screen

During programming, all antifuse electrodes are precharged at a given voltage, Vpre. To program the antifuse, its poly-silicon electrode is raised to Vpp while its N+ diffusion is grounded. The unselected antifuses are subjected to the stress of either Vpre to ground or Vpp to Vpre for an average of 100 times the single antifuse programing time. Usually the Vpre is set such that stress is approximately Vpp/2. If defective unselected antifuses fail (become programmed) due to this stress, they will show up as programming failures. These defective antifuses can be screened out at wafer sort by a 1 second stress at 10 volts (10V/1s). This screen is done twice during sort. The first 10v/1s (FS-1) screens out the defective dielectric distribution. The second 10V/1s stress (FS-2) simulates the percent yield loss during programming. In Fig. 13, it shows a typical wafer trend on the failure rate of both first and second stress. The 10 run average of FS-2 is 0.3 %. This suggests that the programming failure loss due to antifuse defects after the screen should be less than 0.3%.

Unlike floating gate EPROMs and EEPROMs, latent ONO defects can be easily screened out with a voltage stress as described above. This is one more advantage of the antifuse structure as a programming element. Once an antifuse has passed the voltage screen at sort, it is very reliable. Based on either 1/E or $1/\sqrt{E}$ model, the 10V/1s stress is equivalent to a stress time at 5.5V well over 40 years. We have calculated the equivalent product failure rate at 5.5V as a function of the FS-2 screen yield loss. It shows that for an FS-2 of 0.5%, the equivalent FITs at 5.5V 125°C is less than 50, which is consistent with the results mentioned at the end of the previous section.

Programmed Antifuse Reliability

Once the antifuse is programmed and forms a low resistance path, the resistance should remain low. In the case of oxide, it is a known fact that they are susceptible to self healing [11] or an increase in resistance with time. This is not the case for ONO as will be shown in the following section.

A four terminal Kelvin structure was used for the reliability study (Fig. 14). A constant 5mA current, which is much larger than the operating current, was passed through the antifuse at 250°C (through terminals A,B) while the voltage across the antifuse was monitored between terminals A and B. A typical voltage vs time graph is shown in Fig. 15. A sudden increase in voltage indicates that an open circuit has formed. Prior to that, there is no significant change in the voltage across the antifuse indicating that the resistance remained low.

Next, electric continuity measurement and scanning electron microscopy (SEM) were done on the Kelvin structure. It was found that

the antifuse resistance still remained low when measured from the other two unstressed terminals C and D. This is the case for all samples tested under this condition. SEM analysis showed that the open circuit was related to the metal to poly contact electromigration failure (Fig. 16). The activation energy (based on 250°C and 200°C data) for the contact electromigration is 1.1eV, which is in agreement with typical values obtained from contact electromigration failures [12]. The extrapolated lifetime of contacts in these circuits under normal operating conditions is well in excess of 40 years. The real lifetime of a programmed antifuse itself is yet to be determined.

High Temperature Product Burn-in Life Data

In previous sections, it was demonstrated that the extrapolated ONO antifuse lifetime follows the 1/E model instead of the E model. Product burn-in data supports this conclusion. 1108 units (including PROMs, ACT1010, and ACT1020) containing an average of about 100K antifuses per unit, about 5% of which were programmed, underwent dynamic burn-in at 125°C and 5.5V with roughly an accumulated 1.8 million device hours. No antifuse failure has been observed while two CMOS circuit failures have been observed and identified in the peripheral circuitry. This data is consistent with the failure rate projection based on 1/E extrapolation, while the E model extrapolation based on TDDB test data would have projected 90 unit failures (out of 1108 units) due to ONO antifuses.

Conclusions

We have investigated three reliability aspects of the ONO antifuses. During operation, the lifetime of the ONO antifuse is well in excess of 40 years at elevated temperatures. It has been further demonstrated that the E model is not adequate for lifetime extrapolation. Results indicate that 1/E is a better choice. The key to successful extrapolation is that data should span over seven orders of magnitude in time. Based on the 1/E model, the extrapolated lifetime is well over 40 years at 5.5V. To screen out the programming yield loss due to breakdowns of defective unselected antifuses, a screen was developed. This is not a yield limiting factor in the typical process as the yield loss due to the screen on the average is 1%. After the screen, the programming yield is higher than 99%. The reliability of programmed ONO antifuses was also studied. It was found that the lifetime is limited by the contact electromigration, not by the ONO antifuse. In addition, the resistance remains low throughout the test indicating the antifuse resistance does not increase. Finally, more than 1100 product units and over 1.8 million unit hours of burn-in data have shown no failure at all that can be attributed to the ONO antifuses. This is in agreement with the prediction based on wafer-level tests and the 1/E model.

References

[1] A. Haines , "Field-Programmable Gate Array with Non-Volatile Configuration", Microprocessors and Microsystems, Vol. 13, No. 5, pp. 305-312, June, 1989.

[2] A. El Gamal, J. Greene, J. Reyneri, E. Rogoyski, K. El-ayat, and A. Mohsen, "An Architecture for Electrically Configurable Arrays", IEEE J. Solid-State Circuits. Vol 24, No. 2 pp. 394-398, Apr. 1989.

[3] Preliminary data from a military system manufacturer. Test done on total dose Gamma Irradiation Survivability Test.

[4] E. Hamdy, J. McCollum, S. Chen, S. Chiang, S. ELtoukhy, J. Chang, T. Speers, A. Mohsen, " Dielectric Based Antifuses for Logic and Memory ICs", IEDM Tech. Digest, pp. 786-789, 1988.

[5] K. Boyko, D. Gerlach, "Time Dependent Dielectric Breakdown of 210 A Oxides", Proc. Int. Rel. Phys. Symp. pp. 1-8, 1989.

[6] I. Chen, S. Holland, C. Hu, "Electric Breakdown in Thin Gate and Tunneling Oxides", IEEE Trans. Electron Devices, ED-32, No. 2, pp. 413-422, Feb. 1985.

[7] S. Sze, "Physics of Semiconductor devices", 2nd Edition, John Wiley and Sons,Inc. pp 402-407, 1981.

[8] A. Nishimura, S. Murata, S. Kuroda, O. Enomoto, H. Kitagawa, and S. Hasegawa, "Long Term Reliability of SiO2/SiN/SiO2 Thin Layer Insulator Formed in 9 um Deep Trench on High Boron Concentrated Silicon", Proc. Int. Rel. Phys. Symp. pp. 158-162, 1989.

[9] Y. Ohji, T. Kusaka, I. Yoshida, A. Hiraiwa, K. Yagi, and K. Mukai, and O. Kasahara, "Reliability of Nano-Meter Thick Multi-Layer Dielectric Films on Poly-Crystalline Silicon", Proc. Int. Rel. Phys. Symp. pp. 55-59, 1987.

[10] P. Hiergeist, A. Spitzer, and S. Rohl, "Lifetime of Thin Oxide-Nitride-Oxide Dielectrics within Trench Capacitors for DRAM's", Trans. Electron devices, Vol. 36, No. 5, pp. 913-919, May, 1989.

[11] D. Walters, J. van der School, "Dielectric Breakdown in MOS Devices, Part I,II, III", Phillips J. Res. Vol. 40, pp. 115-192, 1985.

[12] D.S. Peck and O.D. Trapp, "Accelerated Testing Handbook", pp. 5-36 to 5-37, 1987.

Table 1 Field accelerated test data for two lots with thickness ranging from 8nm to 9.5nm. The test was done on 0.04mm2 area capacitor.

Lot A					Lot B				
Voltage (V)	Tox (nm)	E-field (MV/cm)	# of cap	t_{50} (sec)	Voltage (V)	Tox (nm)	E-field (MV/cm)	# of cap.	t_{50} (sec)
13.5	8.3	16.2	22	4.2e-3	14.0	8.7	15.9	25	9.8e-3
12.5	8.3	15.1	22	3.7e-2	13.0	8.7	14.9	25	5.0e-2
12.0	8.3	14.4	22	1.5e-1	12.5	8.7	14.3	25	2.4e-1
11.5	8.3	13.8	22	8.6e-1	12.0	8.7	13.7	25	1.3e0
11.0	8.4	13.1	22	4.7e0	11.4	8.7	13.1	25	9.0e0
10.5	8.4	12.5	9	5.8e1	11.2	8.7	12.5	45	8.0e1
10.0	8.3	12.0	6	3.2e2	10.8	9.0	12.0	45	3.52e2
9.5	8.3	11.4	6	2.5e3	10.2	9.0	11.3	45	2.88e3
9.0	8.3	10.7	36	2.5e4	9.7	9.0	10.8	45	2.07e4
8.5	8.3	10.2	15	2.3e5	9.0	8.7	10.3	32	3.35e5
8.0	8.3	9.6	59	1.5e6	9.0	9.3	9.7	32	2.22e6
Sub-total of tested cap.			241					401	
Total of tested cap.			642						

Table 2 High temperature operating life test data (HTOL).

Device	# of units	# of fuse per unit	Device Hours @ 125°C/5.5V*	# fuse Fail	Equivalent Device Hours @ 55°C
PROM64	275	65,536	450,000	0	18.8 Million
1003JLCC	238	40,000	359,400	0	15.0
1010JLCC	144	112,000	283,000	0	11.8
1020JLCC	61	186,000	90,000	0	3.8
1010PLCC	358	112,000	616,000	0	25.8
1020PLCC	32	186,000	5,300	0	0.2
Total	1108	701,536	1,804,100	0	75.5 Million

* All PLCC, 114/144 of 1010 JLCC and 32/61 of 1020 JLCC have 5.75V.

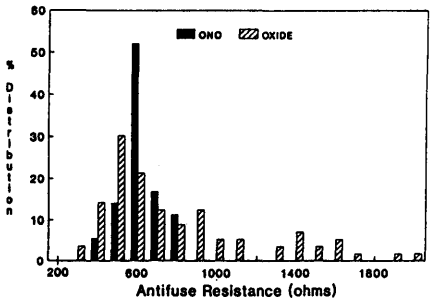


Fig. 1 ONO antifuse has a tighter resistance distribution than oxide antifuse.

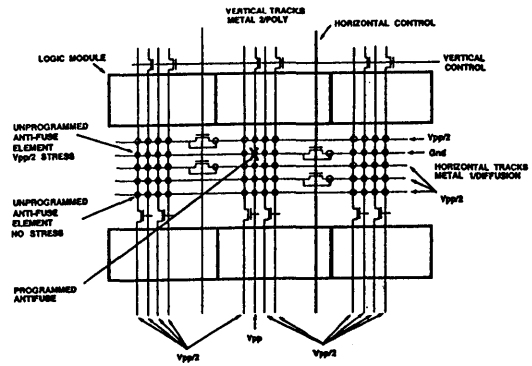


Fig. 2 Simplified product architecture showing logic modules, routing tracks, and antifuse arrays. Vpp is applied to program a selected antifuse. Unselected antifuses have Vpp/2 or 0V stress.

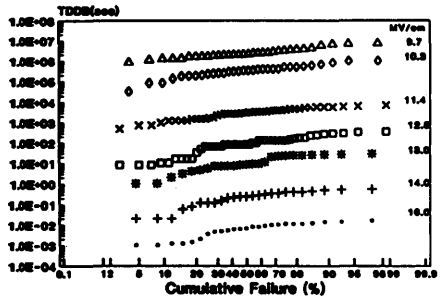


Fig. 3 Cumulative percentage failure verses time on a log-normal scale.

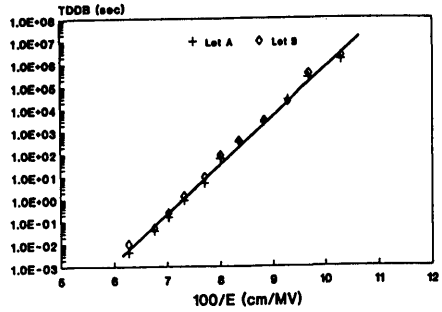
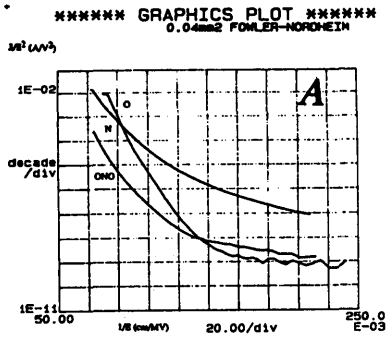


Fig. 5 Log t_{50} vs 1/E for two lots. 0.04mm^2 area capacitor was used.



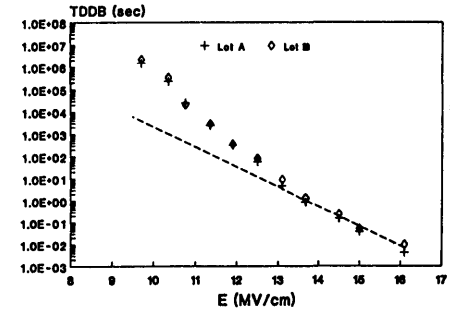
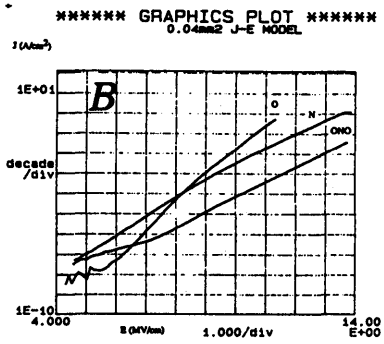


Fig. 6 Log t_{50} vs E for two lots. Log t_{10} has a similar E dependence.



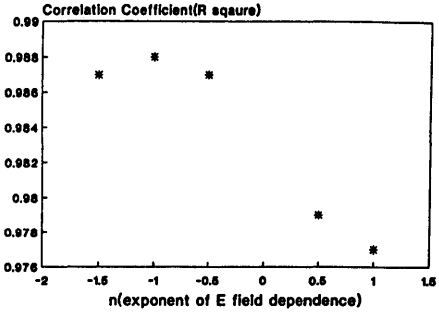


Fig. 4 I-V characteristics of oxide, nitride, and ONO. (a) Fowler-Nordheim tunneling plot. (b) J vs E plot. Log(J) of ONO is not a linear function of 1/E I-V.

Fig. 7 t_{50} was fitted to 5 different distributions. The fitting correlation coefficient (R^2) is plotted against the field exponent n in [exp(E^n)]. 1/E(n= -1) has the best fit with the largest correlation coefficient.

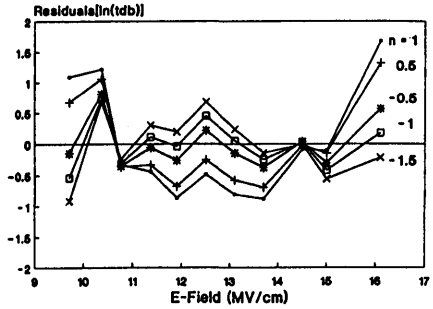


Fig. 8 Residuals at each data point are plotted for 5 different n's. E field dependence has the largest residuals. 1/E and $1/\sqrt{E}$ have the smallest residuals

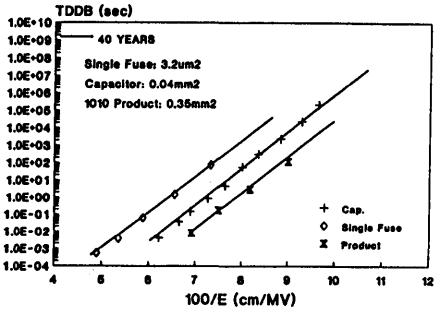


Fig. 9 Log t_{50} can be fit as a linear function of 1/E with the same slope for three different structures: single fuse ($3.2um^2$), area capacitor ($0.04mm^2$), and product array ($0.35mm^2$).

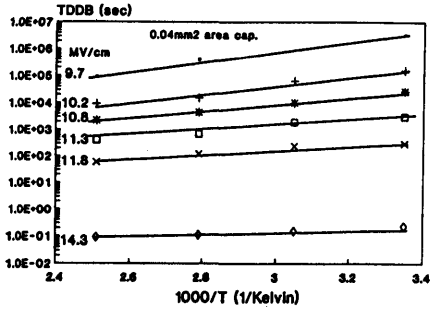


Fig. 10. Field effect on t_{50} at different temperatures ranging from 25°C to 150°C.

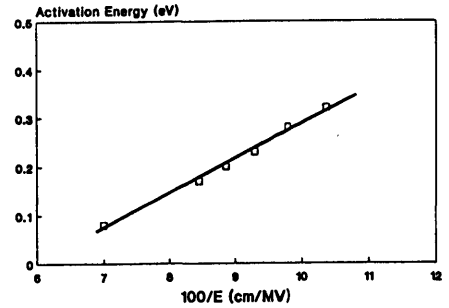


Fig. 11. Field dependence of ONO activation energy.

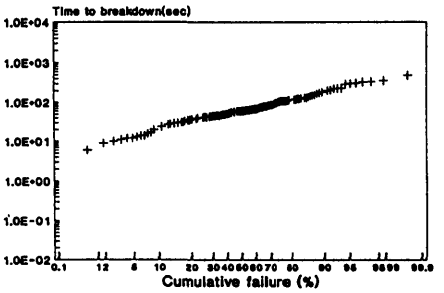


Fig. 12. Cumulative percentage failure of product antifuse array ($0.35mm^2$) vs breakdown time at 11V stress.

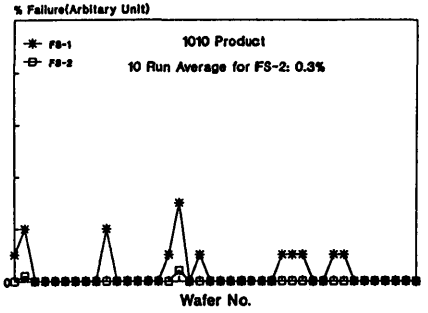


Fig. 13 A typical wafer sort yield loss plot for one lot. After the screen, the 10 run average yield loss is less than 0.3%. Since the screen is more severe than 5.5V/40 years, the product will be very reliable throughout the operating lifetime.

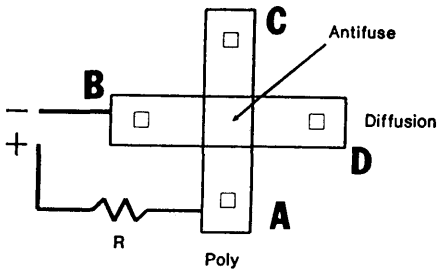


Fig. 14 A four terminal Kelvin structure is used for programmed antifuse reliability test. When an open failure is detected through two stressed terminals, A and B, the antifuse resistance remains low as measured through two unstressed terminals, C and D.

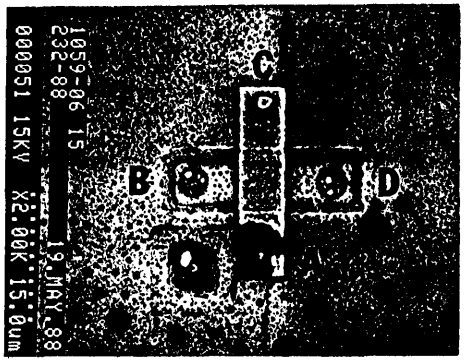


Fig. 16 SEM photograph of the Kelvin structure after showing an open circuit. The open is identified to be at poly to metal contact due to contact electromigration.

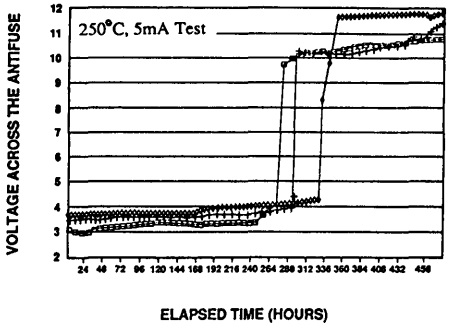


Fig. 15 Voltage across antifuse versus stress time, with 5mA current. Antifuse resistances remains little changed prior to contact failure.

4

Conductive Channel in ONO Formed by Controlled Dielectric Breakdown

*S. Chiang, R. Wang, T. Speers, J. McCollum, E. Hamdy and C. Hu**

Actel Corporation
955 East Arques Avenue, Sunnyvale, California 94086

*University of California
Berkeley, California 94720

Abstract – For the first time, cross section TEM photos capture the conductive channel of Oxide-Nitride-Oxide (ONO) films after electric breakdown. It reveals a single crystal or polycrystal channel with a dome-shaped cap depending on the breakdown current. The implication of this structure on electric characteristics is analyzed with a spherical thermal-electric model. The use of ONO as antifuses in FPGA's is also discussed.

INTRODUCTION

ONO as well as NO or ON has been well developed for DRAM capacitor and FPGA antifuse applications[1][2]. Its low defect density, scalability, and reliability make it very attractive[3]. One main difference between DRAM capacitor and FPGA antifuse application is that besides good dielectric integrity for both applications, understanding how the conductive channel is formed during breakdown is of considerable importance in controlling the resistance of the conductive channel in antifuse applications. Since the breakdown spot is very small and randomly located, capturing this spot in cross section TEM with high degree of repeatability has not been reported before.

STRUCTURE OF THE CONDUCTIVE PATH

Fig.1 Shows the TEM cross section of an ONO dielectric with an equivalent of 9nm oxide thickness. ONO is sandwiched between N+ poly and N+ diffusion. Programming, or breakdown of the ONO film, was achieved with controlled high voltage pulses. External resistor is used to limit the current going through the breakdown spot.

Fig.2 shows the SEM top view of the breakdown spot after the polysilicon gate is removed by wet etching. Etching stops on ONO film. Fig.2 revealed a dome-shaped loose network of the dielectric over the breakdown spot. The many channels through the loose network are apparently the conductive paths of the breakdown dielectric. The diameter of the dome is roughly 3000Å for 16mA programming current and the link resistance is 100Ω.

This dome-shaped structure is confirmed by cross sectional TEM as shown in Fig.3. Fig.3(a) shows a 200nm diameter dome produced by programming current of 12mA. In addition, the material inside the breakdown spot is single crystal. Fig.3(b) shows a small dome produced by a programming current of 3.5mA. Close examination of the photo shows that there are defects inside the N+ silicon near the breakdown channel and the material inside the conductive channel now is polycrystalline.

Apparently, during breakdown (programming), programming current produces sufficiently high temperature to melt the silicon and ONO film over a small volume centered around the point of breakdown. ONO film breaks up into a loose network and protrudes toward anode, probably due to the drift momentum of the electrons in the molten silicon. Upon cooling at the end of programming, nitride (melting point 1900°C) and oxide (melting point 1700°C) solidify into the dome first, then followed by silicon (melting point 1400°C) solidification. High quality epitaxial growth of silicon takes place when the melt volume is large, eg. at 16mA current. When the melt volume is small, such as at 3.5mA current, the temperature gradient and cooling rate are large – resulting in defects in the breakdown spot below ONO film. This prevents formation of epitaxial single crystal inside the channel. In addition, a small-grain polysilicon crest is formed over the dome. It shows that molten silicon region has a large diameter than the molten ONO because of silicon's lower melting point.

RESISTANCE CHARACTERISTICS AND MODEL

It was mentioned above that when the programming current is 3.5mA, the conductive channel is constituted with polysilicon. Temperature dependence of the resistance of a 5mA programmed antifuse, shown in Fig.4 is indeed similar to that of polysilicon resistor. Fig.5 shows that the conductive channel resistance decreased with increasing programming current.

Without the dome-cap (ONO part) in Fig.3(a), the resistance through the opening of 100nm radius, r, in the ONO film would have been $R=\rho/2r$, where ρ is the resistivity (2mΩ-cm) of the surrounding N+ silicon, i.e. R=100Ω. The measured resistance is about 200Ω, from which the resistivity of the dome material can be estimated to be 15mΩ-cm. Empirically, the dome cap contributes about half of the resistance over the range of programming current studied, i.e. it raised the effective ρ by 2X. To understand what determines r and R, assume that the temperature is spherically symmetrical and equal to 1900°C, ONO melting point, at radius r during programming and that most of the heat I^2R is dissipated within r. The result is

$$r = I \cdot \left(\frac{\rho}{8\pi \cdot 1900\,\kappa}\right)^{1/2} \quad (1)$$

$$R = \frac{\rho}{2r} = \frac{(1900 \cdot 2\pi\kappa\rho)^{1/2}}{I} \simeq \frac{2.5\ (V)}{I} \quad (2)$$

where ρ is resistivity of the surrounding N+ silicon (2mΩ-cm), κ is the silicon thermal conductivity (0.25W/cm°C), and I is the programming current.

Fig.6 shows the antifuse resistance versus measurement current after programming at 5mA. Resistance read at 100uA is very similar to that read at 5mA. As the read current increases, the resistance also increases due to heating and increase in resistivity. The trend reverses when silicon begins to melt. The melting occurs before the read current reaches 5mA since the melting of silicon can take place at lower current than melting of the initial dielectric. Beyond 5mA measurement current, the radius of the opening, r, in ONO increases and resistance, R, continues to decrease. On the second scan, R would start at the new lower value corresponding to the larger r.

When ONO films are used as antifuse in FPGA product, the resistance of the antifuse can be controlled by choosing a sufficiently large programming current level and the resistance remains stable during 1000 hours of burn-in at 125°C and 5.75V. Negligible change in delay time along many different data paths were observed as shown in Fig.7.

92CH3172-4/92/0000-0020$3.00 © 1992 IEEE

REFERENCES

[1] A. Nishimura, S. Murata, S. Kuroda, O. Enomoto, H. Kitagawa, and S. Hasegawa, "Long term reliability of $SiO_2/SiN/SiO_2$ thin layer insulator formed in 9μm deep trench on high boron concentrated silicon," *IEEE IRPS Proc.*, pp.158-162, 1989.

[2] A. El Gamal, J. Greene, J. Reynerl, E. Rogoyski, K. El-Ayat, and A. Mohsen, "An architecture for electrically configurable gate arrays," *IEEE J. Solid-State Circuits*, Vol.SC-24, no.2, pp.394-398, Apr. 1989.

[3] S. Chiang, R. Wang, J. Chen, K. Hayes, J. McCollum, E. Handy, and C. Hu, "Oxide-nitride-oxide antifuse reliability," *IEEE IRPS Proc.*, pp.186-192, 1990.

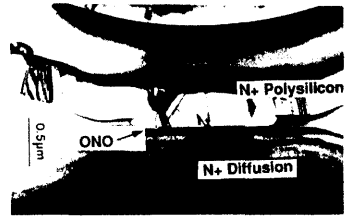


Fig. 1. TEM cross-section of ONO antifuse.

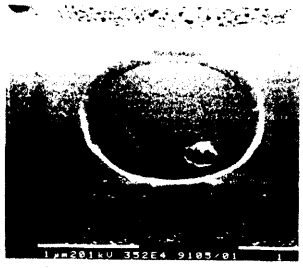


Fig. 2. SEM shows that 16mA breakdown pulse produced a 3000Å opening in the ONO film with a dome cap of a loose network of dielectrics.

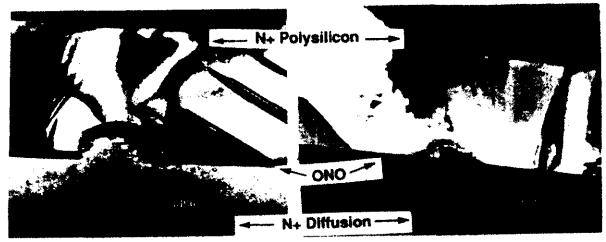


Fig. 3. TEM photos of (a) 2000Å diameter opening in the ONO produced by 12mA breakdown and (b) 700Å diameter opening after 3.5mA breakdown.

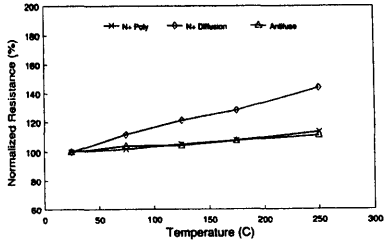


Fig. 4. Temperature dependence of antifuse, N+ diffusion, and polysilicon resistance.

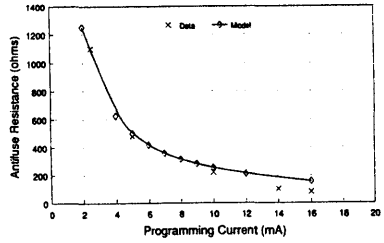


Fig. 5. Antifuse resistance decreases with programming current.

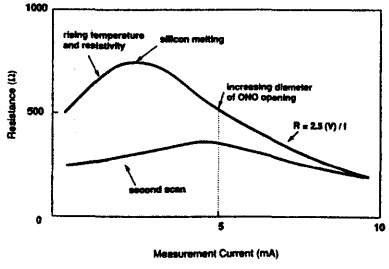


Fig. 6. Resistance versus measurement current after programming with 5mA of breakdown current.

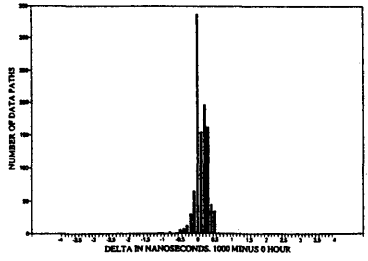


Fig. 7. Change in delay time from a 1000 gate programmable gate array after 1000 hours dynamic burn-in at 125°C and 5.75V.

Metastability of ACT 1 Devices

Actel Metastability Characteristics

Designers often have asynchronous signals coming into synchronous systems. Typically a flip-flop is used to synchronize the incoming signal with the system clock.

If the asynchronus incoming signal does not meet the setup time requirement for the flip-flop, a window of time exists where the incoming signal may cause the flip-flop to develop an unknown, or metastable, logic condition. Figure 1 shows this window as t_W. Actual clock setup time of the flip-flop is shown as t_{SU}; propagation delay of the flip-flop is shown as t_{CQ}. Resolution time (t_{res}) between flip-flop output and the next clocked device is the amount of time required for the metastable condition to stabilize.

The duration of a metastable condition is probabilistic. But the designer can calculate how often a metastable state will last longer than a given duration. Mean time between failures (MTBF) can be calculated from the following equation:

$$MTBF = \frac{1}{f_{clk} \cdot f_{dat} \cdot C1\, e^{-C2 * tres}}$$

where the constants depend on the ACT™ 1 device characteristics, and:

MTBF = Mean time between failures (s)

f_{clk} = System clock frequency (Hz)

f_{dat} = Incoming data rate (Hx)

e = Natural log base

t_{res} = Resolution time (ns)

C1 = 10^{-9}/Hz

C2 = 4.6052/ns

(Value of C2 derived from circuit simulations.)

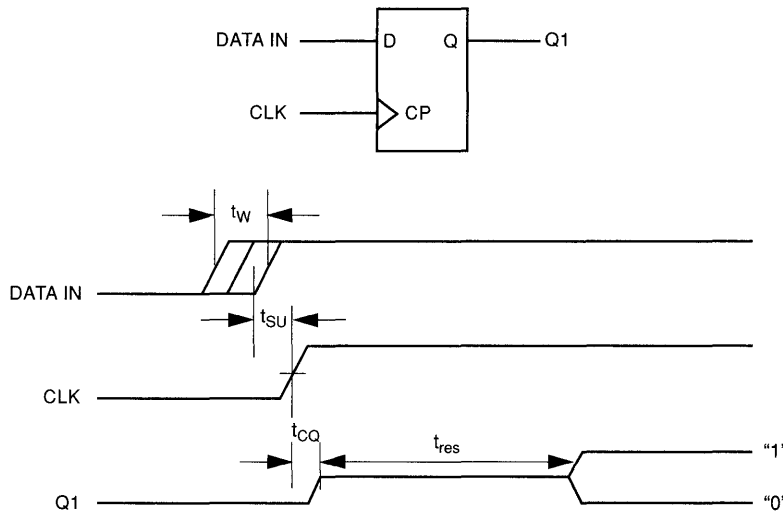


Figure 1. Metastable Condition

Sample Calculation

Using the MTBF equation for a design with a system clock frequency of 10 MHz and a data rate of 1 MHz, various resolution times produce the results shown in Figure 2. Linear increases in resolution time cause exponential increases in MTBF.

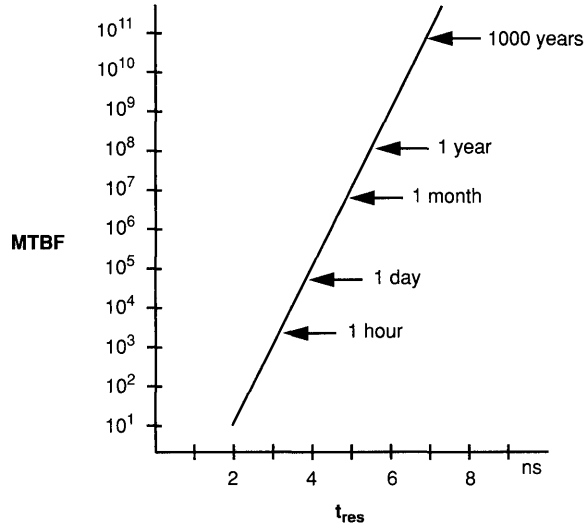
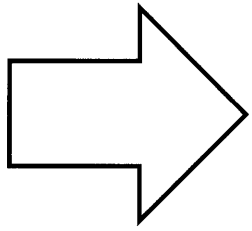


Figure 2. Metastable MTBF as a Function of Resolution Time

Development Tools



System Product Selector Guide

PC Development Systems

Development System	Design Kit	Software Options			Hardware Options			ChipEdit
		Schematic Entry	PLD Mapping	Simulation	Programming		Actionprobe® Diagnostics	
					Activator® 2	Activator 2S		
Designer Development Systems (up to 2,500 gates)								
DS-PC-OR	OrCAD™	Supports SDT 386+	ACTmap™ FPGA Fitter (included)	Supports VST 386+	Optional (ALS-219)	Optional (DS-P2S-PC)	Optional (ALS-218)	Optional (DS-CE-PC)
DS-PC-VL	Viewlogic®	Optional (VL-005)	ACTmap FPGA Fitter (included)	Optional (ALS-017)	Optional (ALS-219)	Optional (DS-P2S-PC)	Optional (ALS-218)	Optional (DS-CE-PC)
DS-PC	Generic	Supports EDIF-based tools	ACTmap FPGA Fitter (included)	Supports EDIF-based tools	Optional (ALS-219)	Optional (DS-P2S-PC)	Optional (ALS-218)	Optional (DS-CE-PC)
Designer Advantage™ Development Systems (up to 10,000 gates)								
DA-PC-OR	OrCAD	Supports SDT 386+	ACTmap FPGA Fitter (included)	Supports VST 386+	Optional (ALS-219)	Optional (DS-P2S-PC)	Optional (ALS-218)	Optional (DA-CE-PC)
DA-PC-VL	Viewlogic	Optional (VL-005)	ACTmap FPGA Fitter (included)	Optional (ALS-016)	Optional (ALS-219)	Optional (DS-P2S-PC)	Optional (ALS-218)	Optional (DA-CE-PC)
DA-PC	Generic	Supports EDIF-based tools	ACTmap FPGA Fitter (included)	Supports EDIF-based tools	Optional (ALS-219)	Optional (DS-P2S-PC)	Optional (ALS-218)	Optional (DA-CE-PC)

Workstation Development Systems

Development System	Design Kit	Software Options				Hardware Options			ChipEdit
		Schematic Entry	PLD Mapping	Simulation	Synopsys Synthesis Libraries	Programming		Actionprobe® Diagnostics	
						Activator® 2	Activator 2S		
Designer Advantage Development Systems (up to 10,000 gates)									
SUN™ Workstations									
DA-SN-CD	Cadence™	Supports Composer	ACTmap FPGA Fitter (included)	Supports Verilog	Optional (ALS-SYN-S4)	Optional (ALS-249)	Optional (DS-P2S-SN)	Optional (ALS-218)	Optional (DA-CE-SN)
DA-SN-MG	Mentor Graphics®	Supports NETED™	ACTmap FPGA Fitter (included)	Supports Quicksim II™	Optional (ALS-SYN-S4)	Optional (ALS-249)	Optional (DS-P2S-SN)	Optional (ALS-218)	Optional (DA-CE-SN)
DA-SN-VL	Viewlogic	Supports Viewdraw®	ACTmap FPGA Fitter (included)	Supports Viewsim®	Optional (ALS-SYN-S4)	Optional (ALS-249)	Optional (DS-P2S-SN)	Optional (ALS-218)	Optional (DA-CE-SN)
DA-SN	Generic	Supports EDIF-based tools	ACTmap FPGA Fitter (included)	Supports EDIF-based tools	Optional (ALS-SYN-S4)	Optional (ALS-249)	Optional (DS-P2S-SN)	Optional (ALS-218)	Optional (DA-CE-SN)
HP 700® Workstations									
DA-HP7-MG	Mentor Graphics	Supports NETED	ACTmap FPGA Fitter (included)	Supports Quicksim II	Optional (ALS-SYN-HP7)	Optional (DS-P2-HP7)	Optional (DS-P2S-HP7)	Optional (ALS-218)	Optional (DA-CE-HP7)
DA-HP7	Generic	Supports EDIF-based tools	ACTmap FPGA Fitter (included)	Supports EDIF-based tools	Optional (ALS-SYN-HP7)	Optional (DS-P2-HP7)	Optional (DS-P2S-HP7)	Optional (ALS-218)	Optional (DA-CE-HP7)

Designer and Designer Advantage System Environment

Overview

The Designer and Designer Advantage™ systems are high-productivity computer-aided engineering environments used with Actel's family of field programmable gate array (FPGA) devices. A new, user-friendly Microsoft Windows™ (PC) and X Window™ (workstation) graphical user interface allows completion of Actel designs from concept to silicon in hours without costly nonrecurring engineering expenses. The systems consist of placement and routing, timing verification, and programming software with interfaces to many popular CAE platforms such as Cadence™, Mentor Graphics®, OrCAD™, and Viewlogic®. In addition, the Designer and Designer Advantage systems support programming on Data I/O's Unisite and 3900 series programmers. With minimal investment in the Designer and Designer Advantage systems, designers can use their existing environments to enter PAL® or Boolean equations, to capture schematics, to simulate, verify, and place and route, to perform timing analysis, to program, and to debug their chips.

Designer for 386™ PC and 486™ PC platforms supports all ACT™ 1, ACT 2, and ACT 3 devices up to and including 2500 gates. In addition, Designer customers who maintain their system with annual software support will receive free software upgrades for any devices up to 2500 gates in future Actel families.

Designer Advantage for 386 PC, 486 PC, Sun™, and HP® platforms covers the entire range of Actel FPGAs. It also supports all Actel devices from 1200 to 10,000 gates.

The following sections describe the various modules that make up the Designer and Designer Advantage systems. This information will provide the background information necessary to decide which Designer product is correct for your design requirements.

PAL and PLD Entry

Actel fully supports PAL and PLD design entry methodology through the ACTmap™ FPGA Fitter. This Fitter is part of the design software that allows PAL descriptions, Boolean equations, and state machines to be quickly entered and optimized into any Actel device. Actel architecture-specific algorithms are used to maximize device utilization and performance. The user can specify whether the design should be optimized for area or speed, depending on design requirements.

Schematic Capture and Simulation

Users enter their design into the ACT device by drawing a schematic using the Actel Macro Library. Once schematic capture has been completed, functional simulation can be performed on the design. Actel provides the Macro Library for popular schematic capture and simulation systems such as Cadence, Mentor Graphics, OrCAD, and Viewlogic.

Logic Synthesis

Actel FPGA designs can be synthesized and optimized from hardware description language specifications such as VHDL or Verilog HDL. Popular logic synthesis tools such as Mentor Graphics Autologic and Viewlogic's VHDL Synthesis support Actel FPGAs. Mentor Graphics and Viewlogic offer their own synthesis libraries for ACT family FPGAs, while the optional Synopsys library can be obtained directly from Actel.

Macro Library

The Actel Macro Library contains the building blocks of logic functions necessary to create a design. The library includes macros ranging in complexity from simple gates to complex functions. Each macro has a graphic symbol, which is used to enter it on a schematic.

For added design flexibility, soft macros such as counters, adders, and decoders can be created from hard macros. The Actel Macro Library contains more than 500 different hard and soft macro functions.

Design Validator

The Actel Design Validator examines an ACT design for adherence to Actel device-specific design rules. Validation calculates routability and performs design rule checks prior to routing. The Validator produces error and information messages and system warnings regarding electrical rule violations such as excessive fanout, shorted outputs, and unconnected inputs. For example, a warning message is issued if a net exceeds a fanout of ten. An unconnected module input, however, produces an error message.

The Validator also provides statistical information about routability, logic module count, average fanout per net, and array utilization. After passing through the Validator, the design is free of electrical rule violations.

Automatic Place and Route

Fully automatic place and route software minimizes design delay by assigning macros to optimal locations on the chip. The system uses the design's netlist, critical net information, and I/O assignments to automatically place and route all the logic blocks within the circuit. No manual intervention is required, even at high device utilization.

During automatic place and route, users have the choice of either automatic I/O placement or manual placement. Typically, automatic I/O pin assignment is preferred because it does a better job than manual assignment, and it is much faster. However, for those situations requiring manual pin assignment, the user can view the chip in the ChipView window (described in a later section) and easily assign I/O pin placement using a graphical or list box entry.

The place and route software provides the user with data on actual interconnect delay. This data can be used in the Timer timing verification tool or backannotated to a simulation netlist for accurate post-route timing simulations. To minimize delay, the route program assigns the shortest possible connections between logic modules, routing 100% of the nets automatically for up to 95% logic module utilization. Because of the abundance of routing resources in the Actel architecture, the nets typically have tight delay distributions, making the uesign's performance more predictable.

For greater control, users can create speed critical nets by assigning a criticality value to the net. The place and route software will minimize the net delay. The user can choose from four levels of criticality, and the place and route software will minimize the net delay accordingly.

Incremental Place and Route

Incremental Place and Route software allows quick iterations of an existing design by making small logic changes without changing the entire placement and routing. After the initial automatic placement and routing, the user can freeze or select varying levels of incremental strength and layout the design again to include the changes. This tool saves time because incremental place and route makes use of the initial automatic place and route file. Design flexibility is enhanced because changes can be made to the design while preserving most of the original performance characteristics.

ChipView

During placement and routing, the actual I/O and logic module placement can be displayed in ChipView, which is a window in the Designer or Designer Advantage system running Microsoft Windows (PC) or X Window (workstation) interfaces. ChipView allows users to zoom, resize, pan, and show multiple windows of the I/O and logic module placement. By identifying critical design paths and viewing the associated logic modules, users can gain insight as to the impact of placement on their design's performance.

ChipEdit

ChipEdit is an optional placement tool that allows users to achieve optimal design performance by manually placing critical logic modules. ChipEdit allows the preplacement of key design sections, as well as the ability to change the existing placement, just by pointing and clicking in the ChipEdit display. Logic modules corresponding to specific design functions can be quickly located and edited by highlighting the function in a hierarchical list box that shows all design macros. After the desired modules are manually placed, the automatic tools are used to place and route the remaining logic modules.

Timing Analysis

The timing analyzer (Timer) is an interactive tool that determines and analyzes the performance of all critical and noncritical paths within a specified design. The Timer performs a static timing analysis, using the post-route delays that were extracted from the layout. Using this information, designs are optimized to meet timing specifications. Delay reports generated by the timing analyzer using post-route numbers provide the final AC specifications for the design. Also, timing analysis can be performed by using an optional CAE simulator supplied by vendors such as Cadence, Mentor Graphics, OrCAD, and Viewlogic. The Designer system generates a post-route backannotation file that can be used by these simulators to provide accurate timing information.

Device Programming and Functional Test

After final timing analysis is completed, the Designer and Designer Advantage systems produce device programming files that can be used by either Data I/O or Actel Activator® series programmers. Existing Data I/O customers can utilize the Unisite or series 3900 programmers to program all Actel ACT 1, ACT 2, and ACT 3 devices when certified by Actel.

Alternatively, Actel offers the Activator 2 and Activator 2S programmers, which program all Actel FPGAs. The Activator 2 programs up to four devices of the same design at a time, while the Activator 2S is a single-device programmer. Users can mount the appropriate programming adapter for their device and package selection.

Designer and Designer Advantage system software generates a fuse map for the ACT device, which is used by the Data I/O or Activator programmers. The fuse map contains information that specifies the programming of the PLICE® antifuse, which determines interconnections within the FPGA. A programming sequence is then initiated. It programs antifuses to customize the FPGA so that it implements the user's design.

In-Circuit Test and Debug

Once the device is programmed and functionally verified, it is ready for operation in the system. Optional Actionprobe® diagnostic tools may be used then to further evaluate circuit integrity. The Actionprobe diagnostic tool works as an adapter to the Activator programmer. It connects to the FPGA diagnostic probe pins in the target system. Under full control of the Actionprobe software, any two internal design signals can be selected and analyzed at the two diagnostic probe pins during system operation. Timing and waveform analysis is then accomplished with an oscilloscope or a logic analyzer. The Actionprobe diagnostic system works with the Activator 2 and Activator 2S programming systems.

Optional Features

Synopsys Libraries

Actel supplies a technology library for the Synopsys logic synthesis environment. It allows designers to capture Actel designs by entering VHDL/HDL format source files into the Synopsys Design Compiler or FPGA Compiler. The compilers create an EDIF netlist, which is read by the Actel supplied EDIF reader in the Designer Advantage environment. Versions of the Synopsys libraries are available for the Sun and HP workstation platforms.

Annual Support Program

The Annual Support Program ensures that the Designer and Designer Advantage software is updated to support the most recently released devices and packages.

Software updates are typically released twice a year and are sent to Annual Support customers at no additional charge.

Another additional benefit of the Annual Support program is that it provides access to technical expertise through the Actel Technical Hotline. Applications engineers are available any time during Actel's regular working hours to answer questions. Also, users have access to Actel's on-line Bulletin Board System (BBS). Services available on the BBS are:

- Software corrections and updates
- Design file uploading and downloading for troubleshooting
- A message service for communicating with applications engineers

The Software Support program also provides access to the Action Facts system. This system allows fast access to Actel's latest application notes, software bug documentation and workarounds, and new product information by means of a dial-up phone system that faxes back the selected document.

Design Platforms

Versions of the Designer and Designer Advantage systems are available to run on a variety of platforms. These systems and options can be bought in various configurations to meet design requirements. Brief descriptions follow for typical system configurations using combinations of the following design styles: PAL/Boolean, equation entry, schematic entry, and HDL or VHDL synthesis.

Designer and Designer Advantage systems support these platforms:

Viewlogic on 386 PC and 486 PC—This system is designed to run on a 386 or 486 IBM-compatible PC running DOS and, optionally, Windows. It integrates with Viewlogic's PROcapture schematic capture and PROsim simulator tools. In addition, it works with Viewlogic's Workview® Plus and Powerview® software packages, which are sold directly by Viewlogic. Actel provides the macro libraries for these tools and the placement and routing, timing, and programming software. Available options are Viewlogic's PROcapture schematic capture, PROsim simulator, ChipEdit placement editor, Activator 2 and Activator 2S programmers, and Actionprobe diagnostic tools.

OrCAD on 386 PC and 486 PC—This system is designed to run on a 386 or 486 IBM-compatible PC running DOS and, optionally, Windows. It integrates with OrCAD's SDT 386+ schematic capture and VST 386+ simulation software packages. Actel provides the macro libraries for these tools and the placement and routing, timing, and programming software. Available options are ChipEdit placement editor, Activator 2 and Activator 2S programmers, and Actionprobe diagnostic tools.

Mentor Graphics on HP and Sun Workstations—These packages integrate with existing design systems utilizing Mentor Graphics' Design Architect schematic capture and QuickSim II™ simulator running on HP or Sun workstations. Actel provides the macro libraries for these tools and the placement and routing, timing, and programming software. Available options are ChipEdit placement editor, Activator 2 and Activator 2S programmers, Actionprobe diagnostic tools, and the Synopsys technology library.

Cadence Design Systems on Sun Workstations—This package integrates with existing design systems utilizing Cadence's Composer™ schematic capture and Verilog® simulator running on Sun workstations. Actel provides the macro libraries for these tools and the placement and routing, timing, and programming software. Available options are ChipEdit placement editor, Activator 2 and Activator 2S programmers, Actionprobe diagnostic tools, and the Synopsys technology library.

Viewlogic on Sun Workstations—This package integrates with existing design systems utilizing Viewlogic's Powerview schematic capture and simulator running on Sun workstations. Actel provides the macro libraries for these tools and the placement and routing, timing, and programming software. Available options are ChipEdit placement editor, Activator 2 and Activator 2S programmers, Actionprobe diagnostic tools, and the Synopsys technology library.

Typical System Configurations

Table 1 lists four major design methodologies and typical system configurations used to support these approaches.

A System Product Selector Guide was provided at the front of this chapter to help you select the version of the software that best suits your needs. Complete data sheets for each version of the software follow.

Table 1. Major Design Methods and Recommended System Configurations

Design Method	System Configuration
PAL/Boolean equation	Designer or Designer Advantage system
	Data I/O or optional Activator 2S/Activator 2 Programmer
Mixed PAL/Boolean equation/Schematic entry/ Simulation	Designer or Designer Advantage system
	Schematic capture tool with EDIF write capability (for example, Viewlogic ViewDraw®)
	CAE simulator (for example, Viewlogic ViewSim®)
	Data I/O or optional Activator 2S/Activator 2 Programmer
Schematic entry/Simulation	Designer or Designer Advantage system
	Schematic Capture tool with EDIF write capability (for example, Viewlogic Viewdraw)
	CAE simulator (for example, Viewlogic Viewsim)
	Data I/O or optional Activator 2S/Activator 2 Programmer
Logic synthesis	Designer Advantage system
	Actel technology libraries for logic synthesis (for example, Synopsys)
	Data I/O or optional Activator 2S/Activator 2 Programmer

Designer and Designer Advantage System with Cadence Composer/Verilog Design Kit

Development System Capabilities

Actel's Designer Advantage™ system for the Cadence™ design environment is a low-cost field programmable gate array (FPGA) design, programming, and verification software system for the ACT™ 1, ACT 2, and ACT 3 families. The Designer Advantage system supports all ACT FPGA devices including the 10K gate A14100. The system consists of Actel's design software and design kit for the Cadence environment, which contains macro libraries and simulation models for all three families of Actel FPGAs. Included in the design software is the ACTmap™ FPGA Fitter that optimizes schematic or behavioral design descriptions for Actel FPGAs. Available options are the Activator® 2 and Activator 2S Programmer and Actionprobe® diagnostic tools.

Cadence Design Flow

Figure 1 shows how the ACT design kit is integrated with Cadence's Composer schematic capture (1) and Verilog simulator packages (2 and 9) to provide all the elements for a complete FPGA design, simulation, and programming environment. Design files can be exported (3) from Cadence directly into the Designer Advantage environment, which runs as a Unix application on the Sun Workstation®.

After importing the files into the Designer Advantage system, the design can be optimized with the ACTmap FPGA Fitter (4). PALs®, Boolean equation descriptions, or state machines can be merged into the design. A specific Actel device is selected (5), and the Design Validator then verifies design rule compliance by completing an electrical rules check and provides statistical information such as utilization percentage and average fanout. After validation, the automatic place and route software (6) implements the engineer's design within the FPGA. The physical placement of the FPGA can be viewed using ChipView, a graphical placement viewer, which operates in the X Window™ system. After automatic placement and routing, the design can be optimized by either utilizing the incremental place and route tool or the optional ChipEdit manual placement tool. After placement and routing, the Timer (7), a static timing analysis tool, verifies circuit timing and delays. Alternately, post-route simulations (8 and 9) can be done with a backannotated delay file to the Cadence Verilog simulator. Once the design is complete, the Activator programmer (10) programs the proper antifuses to configure the FPGA. The Actionprobe diagnostic hardware and software (11) provides 100 percent real-time observability of internal nodes while the FPGA is in the target system.

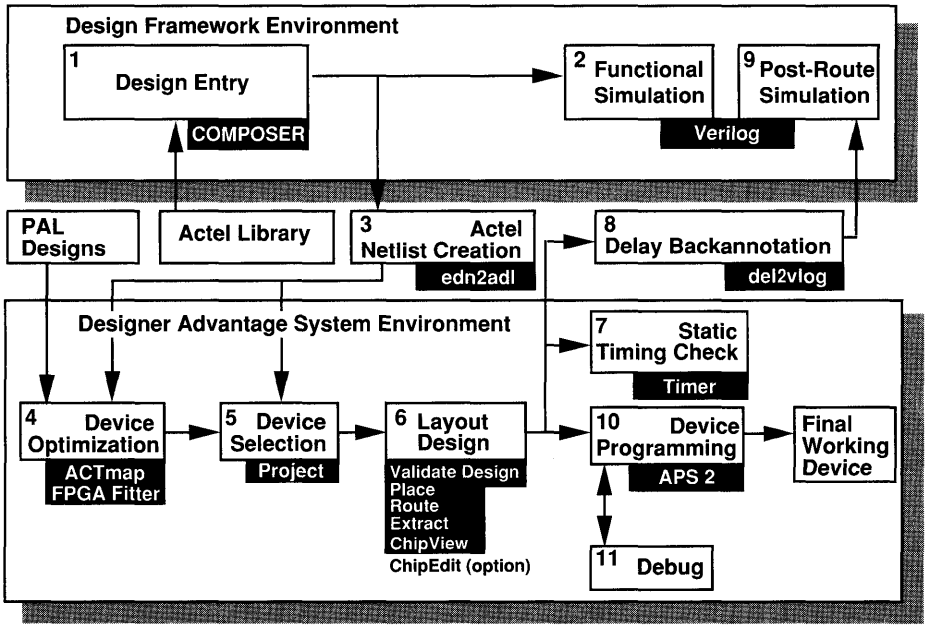


Figure 1. Designer Advantage with Cadence's Composer/Verilog

Software Requirements

Sun Workstation

- Sun OS 4.1.1 or 4.1.2
- Sun OpenWindows version 3.0 (or later) or SunView
- Cadence Composer version 4.2.1 or later
- Cadence Verilog version 1.6a.7.1 or 1.6b

Hardware Requirements

Sun Workstation

- Sun SPARC or SPARC 2
- 16 MB RAM (minimum)
- High-density cartridge tape drive

Activator Programmers

DS-P2S-SN

The Activator 2S programmer is used for ACT 1, ACT 2, and ACT 3 FPGA families. It programs one device at a time, making it ideal for prototyping and low to medium volume production applications. Its modular approach allows for different packages to be programmed by switching programming adapters. Each Activator 2S requires at least one programming adapter.

ALS-249

The Activator 2 programmer is used for ACT 1, ACT 2, and ACT 3 FPGA families. It can program up to four devices simultaneously, making it ideal for medium to high volume production applications. Its modular approach allows for different packages to be programmed by switching programming adapters. Each Activator 2 requires at least one programming adapter.

Activator Programming Adapters

ACT 1 Programming Adapters	Package	ACT 2 Programming Adapters	Package	ACT 3 Programming Adapters	Package
ALS-280	100 QFP	ALS-286	132 PGA	MA3-PL84	84 PLCC
ALS-281	44 PLCC	ALS-287	176 PGA	MA3-QF100	100 QFP
ALS-282	68 PLCC	ALS-288	84 PLCC	MA3-QF160	160 QFP
ALS-283	84 PLCC	ALS-289	100 PGA	MA3-QF208	208 QFP
ALS-284	84 PGA	ALS-290	100 QFP	MA3-PG100	100 PGA
ALS-285	84 QFP	ALS-292	144 QFP	MA3-PG133	133 PGA
MA1-VQ80	80 VQFP	ALS-293	160 QFP	MA3-PG177	177 PGA
		ALS-294	172 QFP	MA3-PG207	207 PGA
				MA3-PG257	257 PGA

Actionprobe Diagnostic Tools

ALS-218

Actionprobe diagnostic tools provide 100 percent real-time observability of internal nodes while the FPGA is running in the target system. This observability is a unique feature that reduces the time required for design verification and debugging.

ChipEdit Placement Editor

DA-CE-SN

The ChipEdit placement editor allows design optimization after automatic placement and routing. Designs are viewed in ChipView, a graphical placement viewer operating in X Window on the Sun Workstation, and then optimized by manual placement of logic modules.

Designer Advantage System Selector Table

System Part Number	ACT 1, ACT 2, and ACT 3 design software up to 10,000 gates	Activator 2S Programmer	Activator 2 Programmer	Actionprobe Diagnostics	ChipEdit
DA-SN-CD	Included	Optional	Optional	Optional	Optional

Designer and Designer Advantage System with Cadence Concept/RapidSIM Design Kit

Development System Capabilities

Actel's Designer Advantage™ system for the Cadence™ design environment is a low-cost field programmable gate array (FPGA) design, programming, and verification software system for the ACT™ 1, ACT 2, and ACT 3 families. The Designer Advantage system supports all ACT FPGA devices including the 10K gate A14100. The system consists of Actel's design software and design kit for the Cadence environment, which contains macro libraries and simulation models for all three families of Actel FPGAs. Included in the design software is the ACTmap™ FPGA Fitter that optimizes schematic or behavioral design descriptions for Actel FPGAs. Available options are the Activator® 2 and Activator 2S Programmer and Actionprobe® diagnostic tools.

Cadence Design Flow

Figure 1 shows how the ACT design kit is integrated with Cadence's Concept™ schematic capture (1) and RapidSIM™ simulator packages (2 and 9) to provide all the elements for a complete FPGA design, simulation, and programming environment. Design files can be exported (3) from Cadence directly into the Designer Advantage environment, which runs as a Unix application on the Sun Workstation™.

After importing the files into the Designer Advantage system, the design can be optimized with the ACTmap FPGA Fitter (4). PALs®, Boolean equation descriptions, or state machines can be merged into the design. A specific Actel device is selected (5), and the Design Validator then verifies design rule compliance by completing an electrical rules check and provides statistical information such as utilization percentage and average fanout. After validation, the automatic place and route software (6) implements the engineer's design within the FPGA. The physical placement of the FPGA can be viewed using ChipView, a graphical placement viewer, which operates in the X Window™ system. After automatic placement and routing, the design can be optimized by either utilizing the incremental place and route tool or the optional ChipEdit manual placement tool. After placement and routing, the Timer (7), a static timing analysis tool, verifies circuit timing and delays. Alternately, post-route simulations (8 and 9) can be done with a backannotated delay file to the Cadence RapidSIM simulator. Once the design is complete, the Activator programmer (10) programs the proper antifuses to configure the FPGA. The Actionprobe diagnostic hardware and software (11) provides 100 percent real-time observability of internal nodes while the FPGA is in the target system.

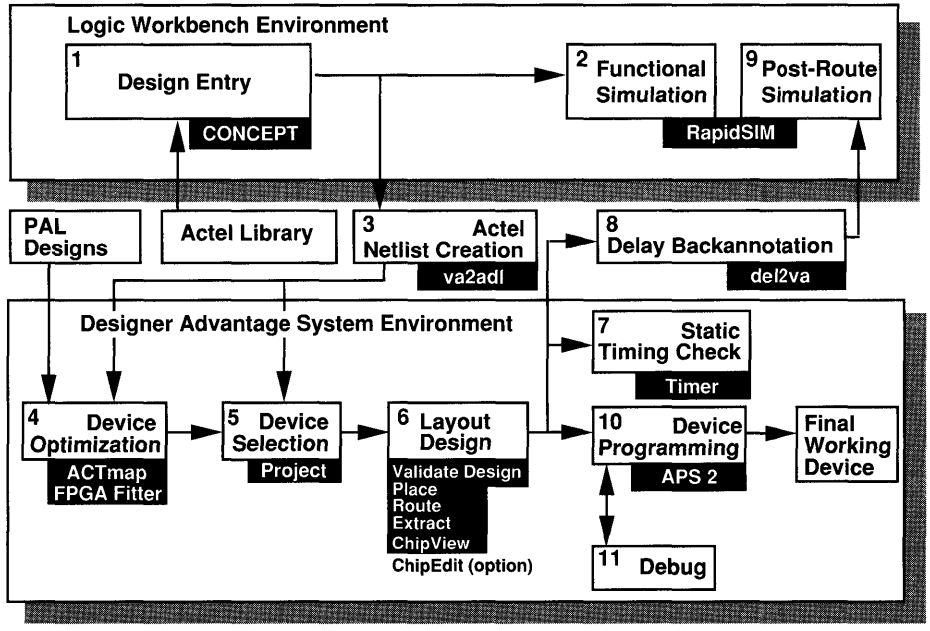


Figure 1. Designer Advantage with Cadence's Concept/RapidSIM

Software Requirements

Sun Workstation

- Sun OS 4.1.1 or 4.1.2
- Sun OpenWindows version 3.0 (or later)
- Cadence Logic Workbench™ version 1.0 (or later)
- Cadence RapidSIM version 1.0 (or later)

Hardware Requirements

Sun Workstation

- Sun SPARC or SPARC 2
- 16 MB RAM (minimum)
- High-density cartridge tape drive

Activator Programmers

DS-P2S-SN

The Activator 2S programmer is used for ACT 1, ACT 2, and ACT 3 FPGA families. It programs one device at a time, making it ideal for prototyping and low to medium volume production applications. Its modular approach allows for different packages to be programmed by switching programming adapters. Each Activator 2S requires at least one programming adapter.

ALS-249

The Activator 2 programmer is used for ACT 1, ACT 2, and ACT 3 FPGA families. It can program up to four devices simultaneously, making it ideal for medium to high volume production applications. Its modular approach allows for different packages to be programmed by switching programming adapters. Each Activator 2 requires at least one programming adapter.

Activator Programming Adapters

ACT 1 Programming Adapters	Package	ACT 2 Programming Adapters	Package	ACT 3 Programming Adapters	Package
ALS-280	100 QFP	ALS-286	132 PGA	MA3-PL84	84 PLCC
ALS-281	44 PLCC	ALS-287	176 PGA	MA3-QF100	100 QFP
ALS-282	68 PLCC	ALS-288	84 PLCC	MA3-QF160	160 QFP
ALS-283	84 PLCC	ALS-289	100 PGA	MA3-QF208	208 QFP
ALS-284	84 PGA	ALS-290	100 QFP	MA3-PG100	100 PGA
ALS-285	84 QFP	ALS-292	144 QFP	MA3-PG133	133 PGA
MA1-VQ80	80 VQFP	ALS-293	160 QFP	MA3-PG177	177 PGA
		ALS-294	172 QFP	MA3-PG207	207 PGA
				MA3-PG257	257 PGA

Actionprobe Diagnostic Tools

ALS-218

Actionprobe diagnostic tools provide 100 percent real-time observability of internal nodes while the FPGA is running in the target system. This observability is a unique feature that reduces the time required for design verification and debugging.

ChipEdit Placement Editor

DA-CE-SN

The ChipEdit placement editor allows design optimization after automatic placement and routing. Designs are viewed in ChipView, a graphical placement viewer operating in X Window on the Sun Workstation, and then optimized by manual placement of logic modules.

Designer Advantage System Selector Table

System Part Number	ACT 1, ACT 2, and ACT 3 design software up to 10,000 gates	Activator 2S Programmer	Activator 2 Programmer	Actionprobe Diagnostics	ChipEdit
DA-SN-CR	Included	Optional	Optional	Optional	Optional

Designer Advantage System with Mentor Graphics Design Kit

Development System Capabilities

Actel's Designer Advantage™ system for the Mentor Graphics® design environment is a low-cost field programmable gate array (FPGA) design, programming, and verification software system for the ACT™ 1, ACT 2, and ACT 3 families. The Designer Advantage system supports all ACT FPGA devices including the 10K gate A14100. The system consists of Actel's design software and design kit for the Mentor Graphics environment, which contains macro libraries and simulation models for all three families of Actel FPGAs. Included in the design software is the ACTmap™ FPGA Fitter that optimizes schematic or behavioral design descriptions for Actel FPGAs. Available options are the Activator® 2 and Activator 2S Programmer and Actionprobe® diagnostic tools.

Mentor Graphics Design Flow

Figure 1 shows how the ACT design kit is integrated with Mentor Graphics' Design Architect schematic capture (1) and QuickSim II™ simulator packages (2 and 9) to provide all the elements for a complete FPGA design, simulation, and programming environment. Design files can be exported (3) from Mentor Graphics directly into the Designer Advantage environment, which runs as a Unix application on the Sun™ or HP 700® workstation.

After importing the files into the Designer Advantage system, the design can be optimized with the ACTmap FPGA Fitter (4). PALs®, Boolean equation descriptions, or state machines can be merged into the design. A specific Actel device is selected (5), and the Design Validator then verifies design rule compliance by completing an electrical rules check and provides statistical information such as utilization percentage and average fanout. After validation, the automatic place and route software (6) implements the engineer's design within the FPGA. The physical placement of the FPGA can be viewed using ChipView, a graphical placement viewer, which operates in the X Window™ system. After automatic placement and routing, the design can be optimized by either utilizing the incremental place and route tool or the optional ChipEdit manual placement tool. After placement and routing, the Timer (7), a static timing analysis tool, verifies circuit timing and delays. Alternately, post-route simulations (8 and 9) can be done with a backannotated delay file to the Mentor Graphics QuickSim II simulator. Once the design is complete, the Activator programmer (10) programs the proper antifuses to configure the FPGA. The Actionprobe diagnostic hardware and software (11) provides 100 percent real-time observability of internal nodes while the FPGA is in the target system.

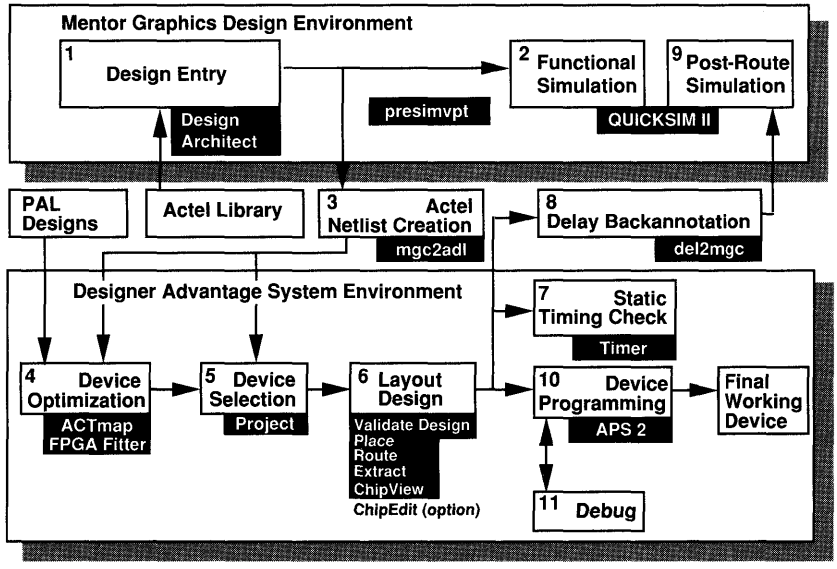


Figure 1. Designer Advantage with Mentor Graphics

Software Requirements

Sun Workstation

- Sun OS version 4.1.1 or 4.1.2
- Sun OpenWindows version 2.0 (or later) or SunView
- Mentor Graphics version 8.2

HP700 Workstation

- HPUX version 9.01
- Mentor Graphics version 8.2

Hardware Requirements

Sun Workstation

- Sun SPARC or SPARC 2
- 32 MB RAM (minimum)
- High-density cartridge tape drive

HP700 Workstation

- HP700 series workstation
- 32 MB RAM (minimum)
- High-density cartridge tape drive

Activator Programmers

DS-P2S-SN (Sun Workstation)

DS-P2S-HP7 (HP700 Workstation)

The Activator 2S programmer is used for ACT 1, ACT 2, and ACT 3 FPGA families. It programs one device at a time, making it ideal for prototyping and low to medium volume production applications. Its modular approach allows for different packages to be programmed by switching programming adapters. Each Activator 2S requires at least one programming adapter.

ALS-249 (Sun Workstation)

DS-P2-HP7 (HP700 Workstation)

The Activator 2 programmer is used for ACT 1, ACT 2, and ACT 3 FPGA families. It can program up to four devices simultaneously, making it ideal for medium to high volume production applications. Its modular approach allows for different packages to be programmed by switching programming adapters. Each Activator 2 requires at least one programming adapter.

Activator Programming Adapters

ACT 1 Programming Adapters	Package	ACT 2 Programming Adapters	Package	ACT 3 Programming Adapters	Package
ALS-280	100 QFP	ALS-286	132 PGA	MA3-PL84	84 PLCC
ALS-281	44 PLCC	ALS-287	176 PGA	MA3-QF100	100 QFP
ALS-282	68 PLCC	ALS-288	84 PLCC	MA3-QF160	160 QFP
ALS-283	84 PLCC	ALS-289	100 PGA	MA3-QF208	208 QFP
ALS-284	84 PGA	ALS-290	100 QFP	MA3-PG100	100 PGA
ALS-285	84 QFP	ALS-292	144 QFP	MA3-PG133	133 PGA
MA1-VQ80	80 VQFP	ALS-293	160 QFP	MA3-PG177	177 PGA
		ALS-294	172 QFP	MA3-PG207	207 PGA
				MA3-PG257	257 PGA

Actionprobe Diagnostic Tools

ALS-218 (Sun and HP700 Workstations)

Actionprobe diagnostic tools provide 100 percent real-time observability of internal nodes while the FPGA is running in the target system. This observability is a unique feature that reduces the time required for design verification and debugging.

ChipEdit Placement Editor

DA-CE-SN (Sun Workstation)

DA-CE-HP7 (HP700 Workstation)

The ChipEdit placement editor allows design optimization after automatic placement and routing. Designs are viewed in ChipView, a graphical placement viewer operating in X Window on the Sun or HP700 workstation, and then optimized by manual placement of logic modules.

Designer Advantage System Selector Table

System Part Number	ACT 1, ACT 2, and ACT 3 design software up to 10,000 gates	Activator 2S Programmer	Activator 2 Programmer	*Actionprobe* Diagnostics	ChipEdit
DA-SN-MG	Included	Optional	Optional	Optional	Optional
DA-HP7-MG	Included	Optional	Optional	Optional	Optional

Designer and Designer Advantage System with OrCAD Design Kit

Development System Capabilities

Actel's Designer and Designer Advantage™ system for the OrCAD™ design environment is a low-cost field programmable gate array (FPGA) design, programming, and verification software system for the ACT™ 1, ACT 2, and ACT 3 families. The Designer Advantage system supports all ACT FPGA devices including the 10K gate A14100. The system consists of Actel's design software and design kit for the OrCAD environment, which contains macro libraries and simulation models for all three families of Actel FPGAs. Included in the design software is the ACTmap™ FPGA Fitter that optimizes schematic or behavioral design descriptions for Actel FPGAs. Available options are the Activator® 2 and Activator 2S Programmer and Actionprobe® diagnostic tools.

OrCAD Design Flow

Figure 1 shows how the ACT design kit is integrated with OrCAD's Schematic Design Tools 386+ schematic capture (1) and Design Simulation Tools 386+ simulator packages (2 and 9) to provide all the elements for a complete FPGA design, simulation, and programming environment. Design files can be exported (3) from OrCAD directly into the Designer or Designer Advantage environment, which runs under Microsoft Windows™.

After importing the files into the Designer Advantage system, the design can be optimized with the ACTmap FPGA Fitter (4). PALs, Boolean equation descriptions, or state machines can be merged into the design. A specific Actel device is selected (5) and the Design Validator then verifies design rule compliance by completing an electrical rules check and provides statistical information such as utilization percentage and average fanout. After validation, the automatic place and route software (6) implements the engineer's design within the FPGA. The physical placement of the FPGA can be viewed using ChipView, a graphical placement viewer, which operates in the Microsoft Windows environment. After automatic placement and routing, the design can be optimized by either utilizing the incremental place and route tool or the optional ChipEdit manual placement tool. After placement and routing, the Timer (7), a static timing analysis tool, verifies circuit timing and delays. Alternately, post-route simulations (8 and 9) can be done with a backannotated delay file to the OrCAD Digital simulation tools simulator. Once the design is complete, the Activator or Data I/O programmer (10) programs the proper antifuses to configure the FPGA. The Actionprobe diagnostic hardware and software (11) provides 100 percent real-time observability of internal nodes while the FPGA is in the target system.

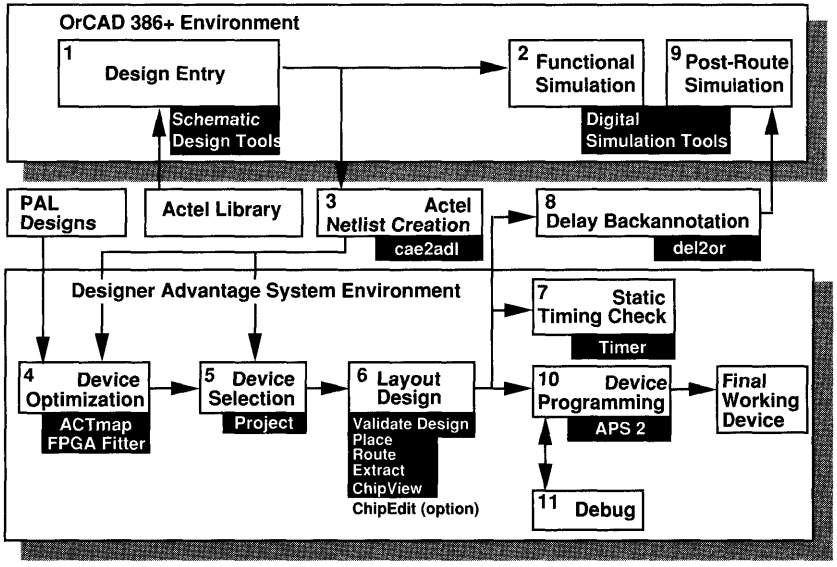


Figure 1. Designer Advantage with OrCAD

Software Requirements

- MS-DOS™ version 3.3 or later
- Windows version 3.1 or later
- OrCAD Schematic Design Tools 386+ version 1.1 or later
- OrCAD Digital Simulation Tools 386+ version 1.1 or later

Hardware Requirements

386/486 based PC-AT

- 8 MB RAM (Designer for Windows)
- 16 MB RAM (Designer Advantage for Windows)
- 30 MB Free Hard Disk Space
- One Parallel Port
- 1.2 or 1.44 MB Floppy Drive
- VGA, EGA, or Monochrome Graphics Card

Activator Programmers

DS-P2S-PC

The Activator 2S programmer is used for ACT 1, ACT 2, and ACT 3 FPGA families. It programs one device at a time, making it ideal for prototyping and low to medium volume production applications. Its modular approach allows different packages to be programmed by switching programming adapters. Each Activator 2 requires at least one programming adapter.

ALS-219

The Activator 2 programmer is used for ACT 1, ACT 2, and ACT 3 FPGA families. It can program up to four devices simultaneously, making it ideal for medium to high volume production applications. Its modular approach allows different packages to be programmed by switching programming adapters. Each Activator 2 requires at least one programming adapter.

Activator Programming Adapters

ACT 1 Programming Adapters	Package	ACT 2 Programming Adapters	Package	ACT 3 Programming Adapters	Package
ALS-280	100 QFP	ALS-286	132 PGA	MA3-PL84	84 PLCC
ALS-281	44 PLCC	ALS-287	176 PGA	MA3-QF100	100 QFP
ALS-282	68 PLCC	ALS-288	84 PLCC	MA3-QF160	160 QFP
ALS-283	84 PLCC	ALS-289	100 PGA	MA3-QF208	208 QFP
ALS-284	84 PGA	ALS-290	100 QFP	MA3-PG100	100 PGA
ALS-285	84 QFP	ALS-292	144 QFP	MA3-PG133	133 PGA
MA1-VQ80	80 VQFP	ALS-293	160 QFP	MA3-PG177	177 PGA
		ALS-294	172 QFP	MA3-PG207	207 PGA
				MA3-PG257	257 PGA

Actionprobe Diagnostic Tools

ALS-218

Actionprobe diagnostic tools provide 100 percent real-time observability of internal nodes while the FPGA is running in the target system. This observability is a unique feature that reduces the time required for design verification and debugging.

ChipEdit Placement Editor

DS-CE-PC or DA-CE-PC

The ChipEdit placement editor allows design optimization after automatic placement and routing. Designs are viewed in ChipView, a graphical placement viewer operating in Microsoft Windows on the PC and then optimized by manual placement of logic modules.

Designer and Designer Advantage System Selector Table

System Part Number	ACT 1, ACT 2, and ACT 3 design software up to 2500 gates	ACT 1, ACT 2, and ACT 3 design software up to 10,000 gates	Activator 2S Programmer	Activator 2 Programmer	Actionprobe Diagnostics	ChipEdit
DS-PC-OR	Included	—	Optional	Optional	Optional	Optional
DA-PC-OR	—	Included	Optional	Optional	Optional	Optional

Designer and Designer Advantage System with Viewlogic Design Kit

Development System Capabilities

Actel's Designer and Designer Advantage™ system for the Viewlogic® design environment is a low-cost field programmable gate array (FPGA) design, programming, and verification software system for the ACT™ 1, ACT 2, and ACT 3 families. The Designer Advantage system supports all ACT FPGA devices including the 10K gate A14100. The system consists of Actel's design software and design kit for the Viewlogic environment, which contains macro libraries and simulation models for all three families of Actel FPGAs. Included in the design software is the ACTmap™ FPGA Fitter that optimizes schematic or behavioral design descriptions for Actel FPGAs. Available options are the Activator® 2 and Activator 2S Programmer and Actionprobe® diagnostic tools.

Viewlogic Design Flow

Figure 1 shows how the ACT design kit is integrated with Viewlogic's ViewDraw® or PROcapture® schematic capture (1) and ViewSim® or PROsim simulator packages (2 and 9) to provide all the elements for a complete FPGA design, simulation, and programming environment. Design files can be exported (3) from Viewlogic directly into the Designer or Designer Advantage environment, which runs under Microsoft Windows™.

After importing the files into the Designer Advantage system, the design can be optimized with the ACTmap FPGA Fitter (4). PALs®, Boolean equation descriptions, or state machines can be merged into the design. A specific Actel device is selected (5), and the Design Validator then verifies design rule compliance by completing an electrical rules check and provides statistical information such as utilization percentage and average fanout. After validation, the automatic place and route software (6) implements the engineer's design within the FPGA. The physical placement of the FPGA can be viewed using ChipView, a graphical placement viewer, which operates in the Microsoft Windows environment or the X Window™ environment on Workstations. After automatic placement and routing, the design can be optimized by either utilizing the incremental place and route tool or the optional ChipEdit manual placement tool. After placement and routing, the Timer (7), a static timing analysis tool, verifies circuit timing and delays. Alternately, post-route simulations (8 and 9) can be done with a backannotated delay file to the Viewlogic ViewSim or PROsim simulators. Once the design is complete, the Activator programmer (10) programs the proper antifuses to configure the FPGA. The Actionprobe diagnostic hardware and software (11) provides 100 percent real-time observability of internal nodes while the FPGA is in the target system.

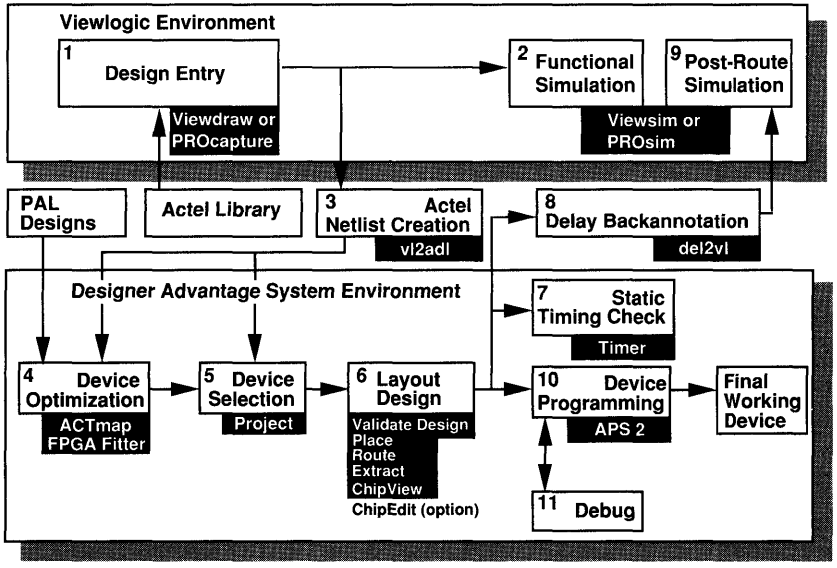


Figure 1. Designer Advantage with Viewlogic

Software Requirements

PC

- MS-DOS™ version 3.3 or later (required)
- Windows version 3.1 or later (optional)
- Viewlogic Workview® PLUS, or PROSeries
- Viewlogic Viewsim or PROsim

Sun Workstation

- Sun OS version 4.1 or 4.1.1
- Sun OpenWindows version 2.0 (or later)
- Viewlogic Powerview®

Hardware Requirements

386/486 based PC-AT

- 8 MB RAM (Designer for Windows)
- 16 MB RAM (Designer Advantage for Windows)
- 30 MB Free Hard Disk Space
- One Parallel Port
- 1.2 or 1.44 MB Floppy Drive
- VGA, EGA, or Monochrome Graphics Card

Sun Workstation

- Sun SPARC
- 16 MB RAM (minimum)
- High-density cartridge tape drive

Activator Programmers

DS-P2S-PC (PC)
DS-P2S-SN (Sun Workstation)

The Activator 2S programmer is used for ACT 1, ACT 2, and ACT 3 FPGA families. It programs one device at a time, making it ideal for prototyping and low to medium volume production applications. Its modular approach allows different packages to be programmed by switching programming adapters. Each Activator 2 requires at least one programming adapter.

ALS-219 (PC)
ALS-249 (Sun Workstation)

The Activator 2 programmer is used for ACT 1, ACT 2, and ACT 3 FPGA families. It can program up to four devices simultaneously, making it ideal for medium to high volume production applications. Its modular approach allows for different packages to be programmed by switching programming adapters. Each Activator 2 requires at least one programming adapter.

Activator Programming Adapters

ACT 1 Programming Adapters	Package	ACT 2 Programming Adapters	Package	ACT 3 Programming Adapters	Package
ALS-280	100 QFP	ALS-286	132 PGA	MA3-PL84	84 PLCC
ALS-281	44 PLCC	ALS-287	176 PGA	MA3-QF100	100 QFP
ALS-282	68 PLCC	ALS-288	84 PLCC	MA3-QF160	160 QFP
ALS-283	84 PLCC	ALS-289	100 PGA	MA3-QF208	208 QFP
ALS-284	84 PGA	ALS-290	100 QFP	MA3-PG100	100 PGA
ALS-285	84 QFP	ALS-292	144 QFP	MA3-PG133	133 PGA
MA1-VQ80	80 VQFP	ALS-293	160 QFP	MA3-PG177	177 PGA
		ALS-294	172 QFP	MA3-PG207	207 PGA
				MA3-PG257	257 PGA

Actionprobe Diagnostic Tools

ALS-218 (PC and Sun Workstation)

Actionprobe diagnostic tools provide 100 percent real-time observability of internal nodes while the FPGA is running in the target system. This observability is a unique feature that reduces the time required for design verification and debugging.

ChipEdit Placement Editor

DA-CE-PC (PC) or DS-CE-PC (PC) or
DA-CE-SN (Sun Workstation)

The ChipEdit placement editor allows design optimization after automatic placement and routing. Designs are viewed in ChipView, a graphical placement viewer operating in either Microsoft Windows on the PC or X Window on the Sun Workstation, and then optimized by the manual placement of logic modules.

Designer and Designer Advantage System Selector Table

System Part Number	ACT 1, ACT 2, and ACT 3 design software up to 2500 gates	ACT 1, ACT 2, and ACT 3 design software up to 10,000 gates	Activator 2S Programmer	Activator 2 Programmer	Actionprobe Diagnostics	ChipEdit
DS-PC-VL	Included	—	Optional	Optional	Optional	Optional
DA-PC-VL	—	Included	Optional	Optional	Optional	Optional
DA-SN-VL	—	Included	Optional	Optional	Optional	Optional

Logic Synthesis Libraries

Logic Synthesis

Actel field programmable gate array (FPGA) designs can be synthesized and optimized from hardware description language specifications such as VHDL or Verilog HDL. Popular logic synthesis tools such as Mentor Graphics'® Autologic, and Viewlogic's® VHDL Synthesis support Actel FPGAs. These technology libraries and tools are directly available from these vendors. Designs using synthesis methodology can be quickly converted from design concept to programmed FPGA.

Support for Synopsys is available from Actel. Listed below is a detailed description of the optional technology libraries available from Actel to support the Synopsys Design Compiler and FPGA Compiler environment.

Synopsys Library Capability

Actel's FPGA libraries for the Synopsys Design Compiler environment support a high-level design synthesis methodology. Designs are described in VHDL or Verilog HDL and mapped into an Actel library using the Synopsys Design Compiler or FPGA Compiler. Then they are transferred to Actel's Designer Advantage™ system for placement and routing, timing verification, and programming.

Design Flow

The Synopsys libraries support all ACT™ 1, ACT 2, and ACT 3 FPGA devices including the 10K gate A14100. These libraries interface with the Synopsys Design Compiler, FPGA Compiler, and Design Analyzer for synthesis and optimization into the Actel architecture. Each compiler maps the HDL behavioral description (1) into the Actel architecture. After satisfactory optimization, the compiler produces an EDIF 2 0 0 netlist. The Actel EDIF netlist program, cae2adl (3), converts the EDIF file to an Actel.adl netlist.

After importing the netlist files into the Designer Advantage system, the Design Validator verifies design rule compliance by completing an electrical rules check and provides statistical information such as utilization percentage and average fanout. After validation, the automatic place and route (6) software customizes the FPGA to the engineer's design. The actual placement of the FPGA can be viewed using ChipView, a graphical placement viewer, which operates in the X Window™ System. After automatic placement and routing, the design can be optimized by either utilizing the incremental place and route tool or the optional ChipEdit manual placement tool. After placement and routing, the Timer (7), a static timing analysis tool, verifies circuit timing and delays. Alternately, post-route simulations can be done with a backannotated delay file to a CAE simulator. Once the design is complete, the Activator® programmer (10) programs the proper antifuses to configure the FPGA. The Actionprobe® diagnostic hardware (11) and software provide 100 percent real-time observability of internal nodes while the FPGA is in the target system.

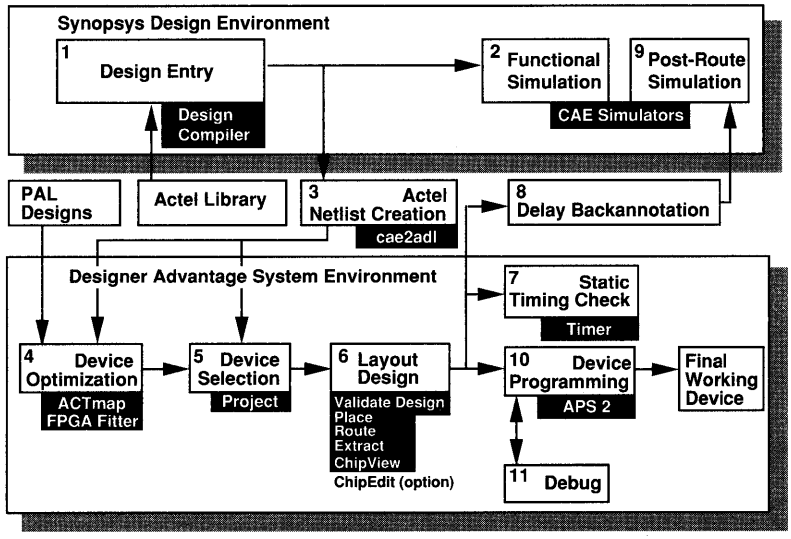


Figure 1. Designer Advantage with Synopsys

Note: For a more detailed description of how to design with the Synopsys libraries, please refer to the application brief titled "High-Level FPGA Design in the Synopsys Environment."

Software Requirements

- Actel version 2.2 libraries for Synopsys Design Compiler

Choose one:

- Synopsys VHDL Compiler version 2.2 or 3.0
- HDL Compiler for Verilog version 2.2 or 3.0

Choose one:

- Synopsys Design Compiler version 2.2 or 3.0
- FPGA Compiler version 3.0, or Design Analyzer version 2.2 or 3.0

Choose one:

- DA-SN (Designer Advantage system for Sun™)
- DA-SN-CD (Designer Advantage system for Sun with Cadence design kit)
- DA-SN-MG (Designer Advantage system for Sun with Mentor Graphics design kit)
- DA-SN-VL (Designer Advantage system for Sun with Viewlogic design kit)
- DA-HP7-MG (Designer Advantage system for HP700® with Mentor Graphics design kit)

Note: For detailed descriptions of the Designer Advantage systems, please refer to the specific data sheets listed earlier in the data book.

Table 1. Synopsys Library Selector Table

Synopsys Platform	Part Number
Sun	ALS-SYN-S4
HP700	ALS-SYN-HP7
HP400	ALS-SYN-HP4

Actel's Industry Alliance Program

Purpose

The Actel Industry Alliance program establishes a practical Technology Transparent Design (TTD) environment for Actel field programmable gate array (FPGA) users. Actel and Alliance members cooperate to provide Actel FPGA users complete design support using the EDA system of their choice. Actel design libraries available from Alliance members allow users to describe and simulate Actel designs using their existing EDA tools. After satisfactory design entry and simulation, these designs can be transferred to Actel's Designer Series development system to complete placement and routing, timing analysis, and programming.

Technical Cooperation

Actel provides technical cooperation and support to Alliance members, including:

- Access to Actel's FPGA development environment, the Designer Series, for placement and routing, timing analysis, and programming.
- CAE connection package, which provides the information necessary to connect the CAE system to Actel's development system.
- Standard verification circuits and test vectors for self certification of the design libraries and interfaces by the Alliance member.

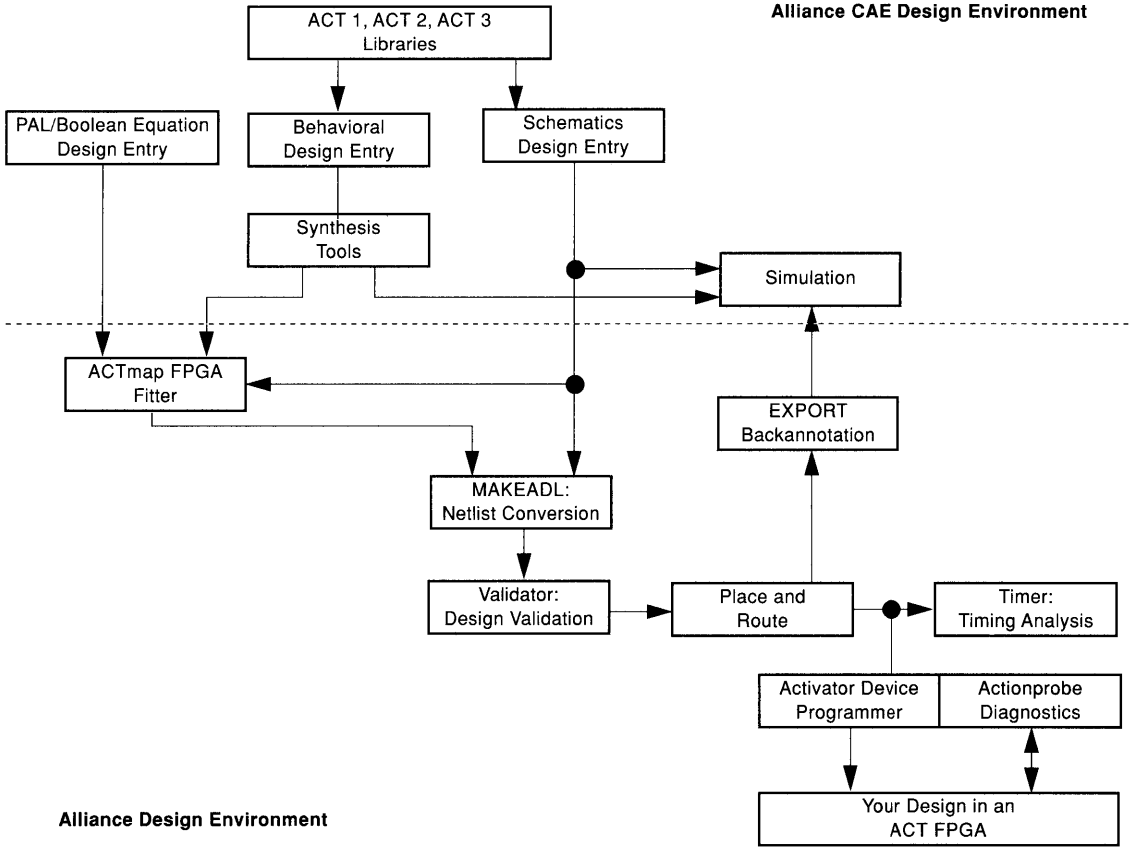


Figure 1. Shown above are typical ways Alliance member CAE environments interface with the Actel design environment.

Alliance Members

Customers can obtain the certified design libraries and interfaces to Actel's development system and FPGA devices by contacting their respective CAE vendor. CAE Alliance members and contacts are listed below.

Current Alliance members

Company Name	Contact Name	Address	Telephone
Aldec	Stanley Hyduke	3525 Old Conejo Road, Suite 111 Newbury Park, CA 91320	(805) 499-6867
Cadence Design System	Jim Watts	2655 Seely Road, Bldg. 6 San Jose, CA 95134	(408) 428-5817
Compass Design Automation	Mahendra Jain	1865 Lundy Avenue San Jose, CA 95131	(408) 434-7950
Data I/O	Linda Burgess	10525 Willows Road N.E. Redmond, WA 98073-9746	(206) 881-6444
Exemplar	Nanette Collins	2550 Ninth Street, Suite 102 Berkeley, CA 94710	(510) 849-0937
GenRad	Jim Redditt	510 Cottonwood Drive Milpitas, CA 95035-7499	(408) 432-1000
Gould/AMI	Robert Kirk	18711 Tiffeni Dr., Suite A Twain Harte, CA 95383	(209) 586-7422
Intergraph Company	Vincent Mazur	6101 Lookout Road, Suite A Boulder, CO 80301	(303) 581-2301
Logic Modeling Corporation	Laura Nichols	19500 N.W. Gibbs Drive Beaverton, OR 97075	(503) 531-2271
Mentor Graphics	Sam Picken	8005 SW Boeckman Road Wilsonville, OR 97070-7777	(503) 685-1298
Minc	Bill Schulze	6755 Earl Drive Colorado Springs, CO 80918	(719) 590-1155
OrCAD	Patrick Karger	3175 NW Aloclek Drive Hillsboro, OR 97124-7135	(503) 690-9881
Recal-Redac	Joe Cerruto	1000 Wyckoff Avenue Mahwah, NJ 07430	(201) 848-8000
Synopsys	Lynn Fiance	700 East Middlefield Road Mountain View, CA 94043-4033	(415) 694-4289
Viewlogic	Paul Granese	293 Boston Post Road West Marlboro, MA 01752	(508) 480-0881
UTMC	Ron Lake	575 Garden of the Gods Road Colorado Springs, CO 80907	(719) 594-8491
Zuken	Shyamal Roy	3945 Freedom Circle, Suite 1100 Santa Clara, CA 95054	(408) 562-0177

Activator® 2 and Activator 2S Programmers

Features

- Supports ACT™ 1, ACT 2, and ACT 3 device families and packages
- Versions available for 386™ PC, 486™ PC, Sun™, and HP® workstations
- Supports functional verification with Actionprobe® diagnostics
- Activator 2 simultaneously programs up to four identical devices
- Activator 2S is a low-cost, single-site version of the Activator 2
- Supports checksum and design name verification

Product Description

Activator 2 and Activator 2S are Actel's state-of-the-art desktop programmers. They utilize Actel's Designer and Designer Advantage™ system software and PLICE® antifuse technology to program Actel's ACT 1, ACT 2, and ACT 3 field programmable gate arrays (FPGAs). Customized programming adapters for each device type allow different packages to be programmed by switching adapters. Programming, verification, and debugging are executed on these programmers. The optional Actionprobe diagnostic hardware and software support observation of all internal signals. Individual versions of these programmers are available to support industry standard platforms such as 386 PC, 486 PC, Sun, and HP workstations.

The programmers are software-driven, providing flexibility of application and a built-in barrier to obsolescence. Independently powered, the units provide desktop device programming, functional testing, and in-circuit debugging. By using a SCSI interface, the programmers support different hardware platforms.

For easy device identification and verification, the blank check (blankchk) command in the programming software allows users to read back checksums and silicon signatures after programming. The checksum data is automatically generated by the Designer software, and the silicon signature is defined by the user during fuse file generation.

Activator 2 Base Unit

The base unit contains the control board and the analog board. The LED display on the top of the programmer shows when the unit receives power. Four adapter ports located on the top of the base unit accept different programming adapters for each device type. The adapter ports are identical, which allows programming adapters to be interchanged. SCSI connectors are located on the back of the unit.

Activator 2S Base Unit

Activator 2S is a low-cost, single-site version of the Activator 2 and supports all the Activator 2 features. A single adapter port is located on the top of the base unit; it accepts different programming adapters for each device type. The SCSI connectors are located on the back of the unit.

The Programming Adapters

There are unique programming adapters available to support each package type within a product family. The user only needs to switch programming adapters to program a different device type or package. For the Activator 2, one to four ports may be used simultaneously when programming a design. Any adapter may be used on any available port.

The Diagnostic Pod

The diagnostic pod supports Actionprobe diagnostics and connects to the programmer by a cable. The Activator 2 or Activator 2S permits the user to view any internal circuit activity in an Actel FPGA mounted on the user's PC board through Actel's on-chip diagnostic pins. These pins are connected to the programmer by the diagnostic pod; by using the Actionprobe diagnostics software to specify a circuit point, each pod can access two signals simultaneously. A single pod may be connected to the Activator 2S or Activator 2. A six-foot cord connects the pod to the programmer and provides diagnostic flexibility.

ACT 1 Hard Macro Library Overview

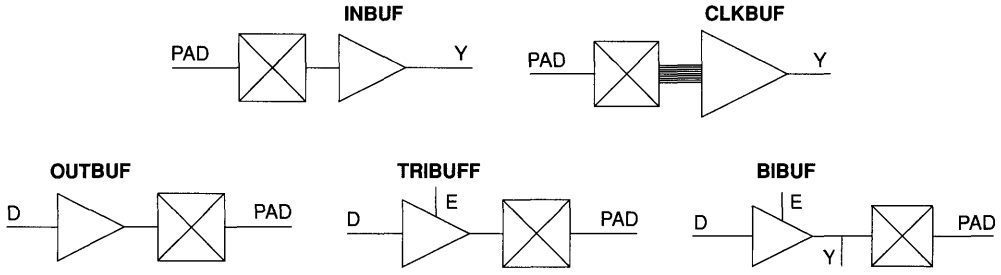
The following illustrations show all the available hard macros. The *ACT Family Macro Library Guide* contains module count, combinability, and pin loading information of each hard macro. It also includes a complete truth table for each macro.

Most ACT 1 hard macros are implemented by a single logic module. The following ACT 1 hard macros require two logic modules to implement:

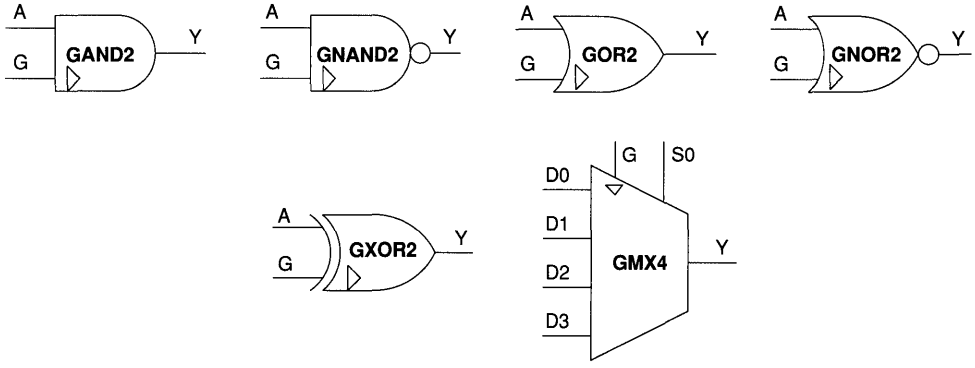
- All flip-flops
- All adders
- Two-module hard macro gates: AND4, AND4A, AND4D, AOI1, AOI4, NAND3, NAND4, NAND4A, NAND4B, NOR4, NOR4C, NOR4D, OAI3, OR3C, OR4B, OR4C, OR4D

Two-module hard macro gates have a "2" displayed on some input pins. This indicates that the input to output path has two levels of logic delay for these input pins only. Also, the full adders have a "2" on the "S" output pins. This indicates that there are two levels of logic delay from the input pins to the "S" output pin. Refer to the ACT 1 Timing Characteristics for detailed timing information of all ACT 1 macros.

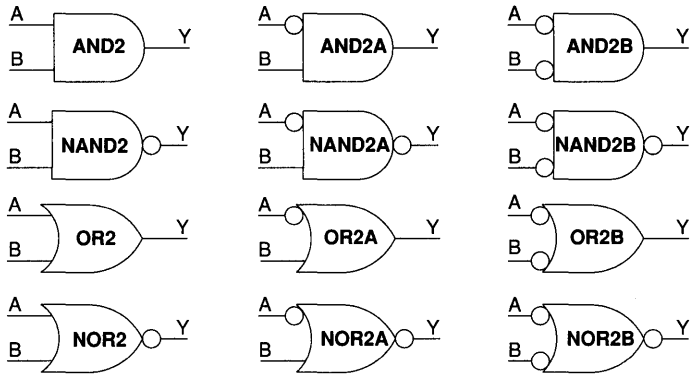
Input and Output Buffers



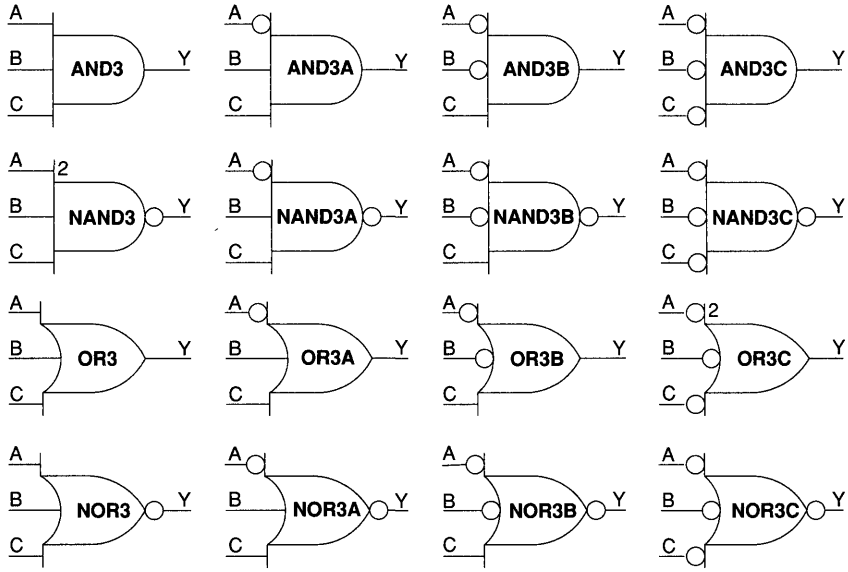
CLKBUF Interface Macros



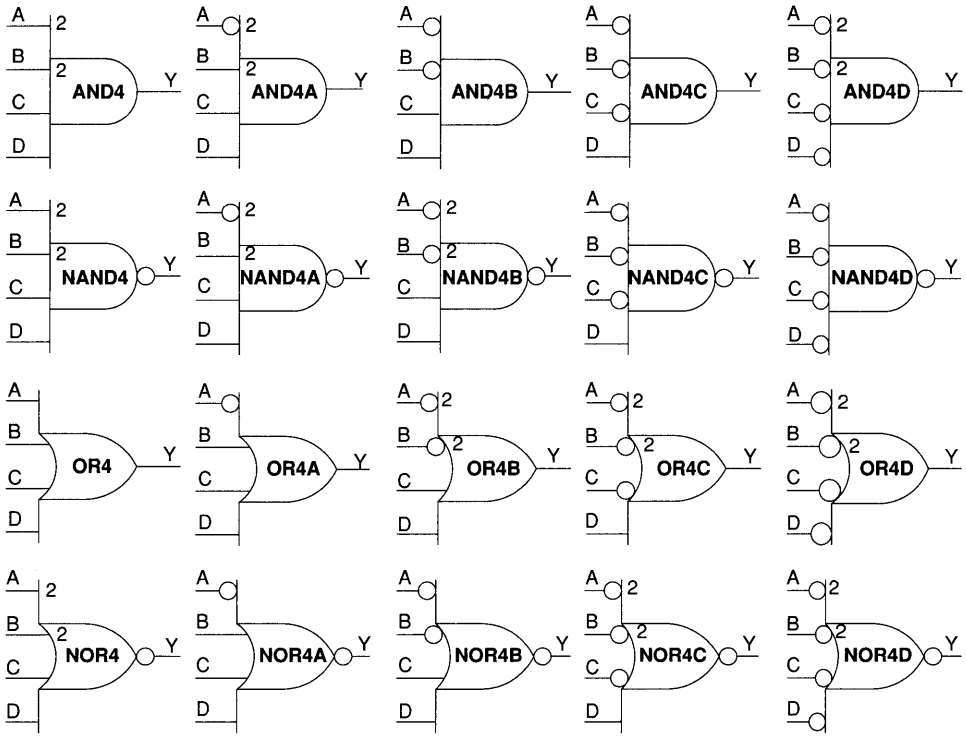
2-Input Gates



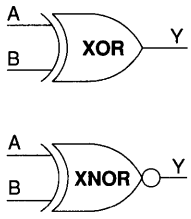
3-Input Gates



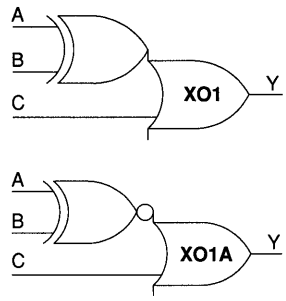
4-Input Gates



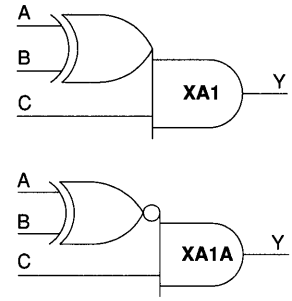
XOR Gates



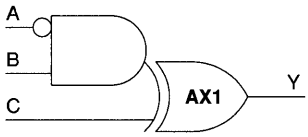
XOR-OR Gates

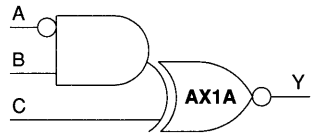


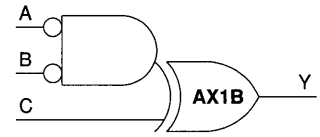
XOR-AND Gates



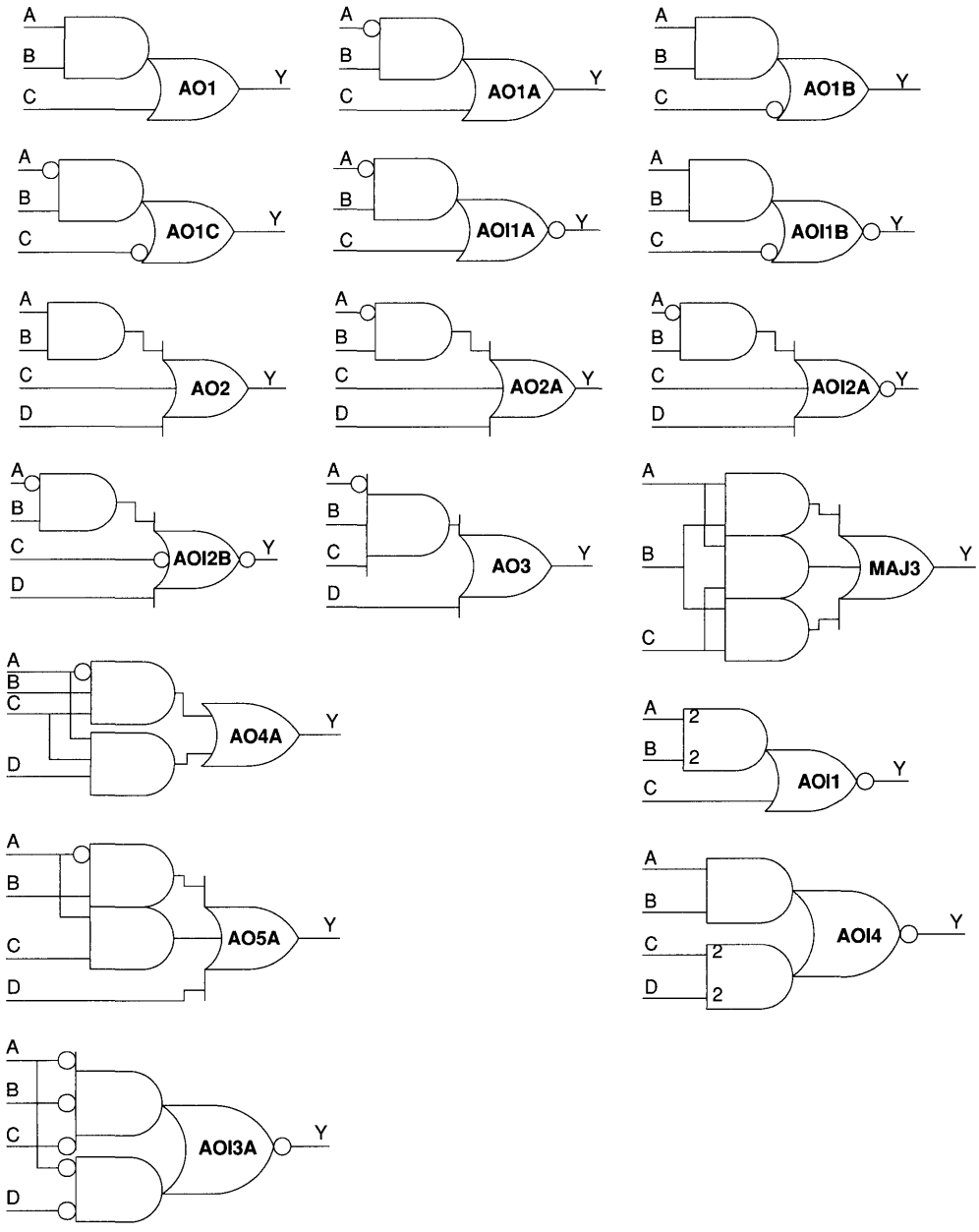
AND-XOR Gates



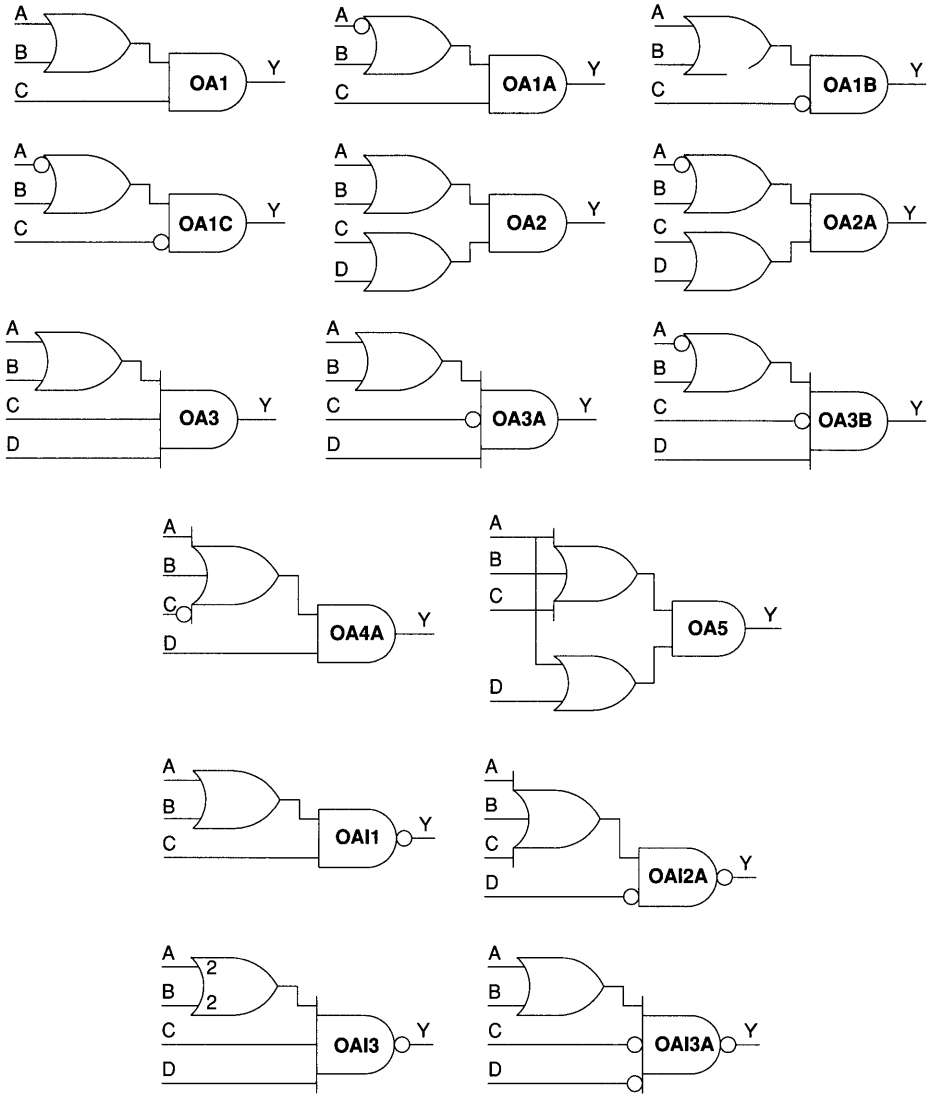




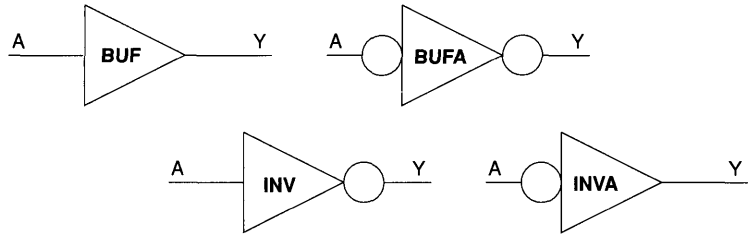
AND-OR Gates


A
B
C
AO1
Y
A
B
C
AO1A
Y
A
B
C
AO1B
Y
A
B
C
AO1C
Y
A
B
C
AOI1A
Y
A
B
C
AOI1B
Y
A
B
C
D
AO2
Y
A
B
C
D
AO2A
Y
A
B
C
D
AOI2A
Y
A
B
C
D
AOI2B
Y
A
B
C
D
AO3
Y
A
B
C
MAJ3
Y
A
B
C
D
AO4A
Y
A
B
C
D
AO5A
Y
A
B
C
D
AOI3A
Y
A
2
B
2
C
AOI1
Y
A
B
C
2
D
2
AOI4
Y

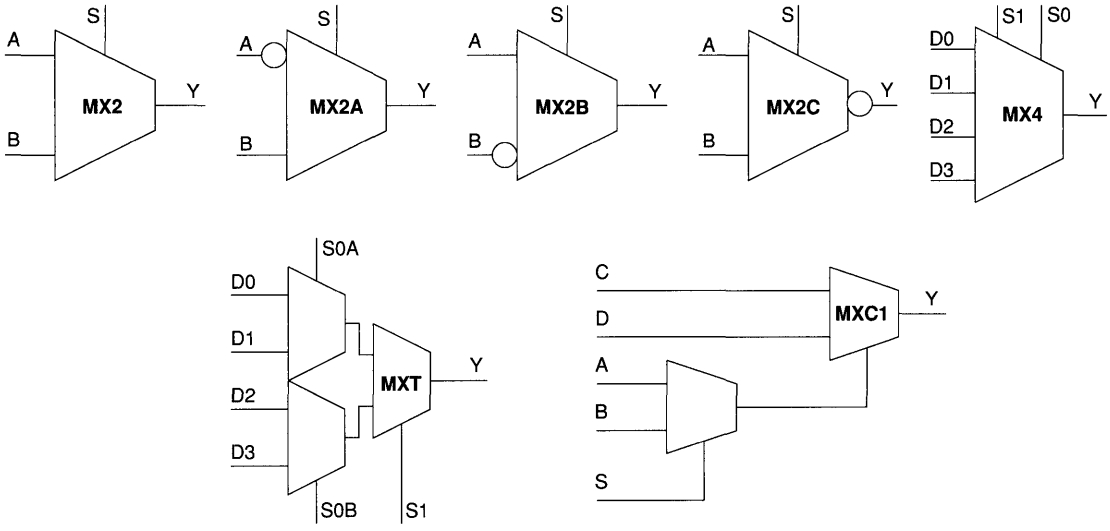
OR-AND Gates


A
B
C
OA1
Y
A
B
C
OA1A
Y
A
B
C
OA1B
Y
A
B
C
OA1C
Y
A
B
C
D
OA2
Y
A
B
C
D
OA2A
Y
A
B
C
D
OA3
Y
A
B
C
D
OA3A
Y
A
B
C
D
OA3B
Y
A
B
C
D
OA4A
Y
A
B
C
D
OA5
Y
A
B
C
OAI1
Y
A
B
C
D
OAI2A
Y
A
B
2
C
2
D
OAI3
Y
A
B
C
D
OAI3A
Y

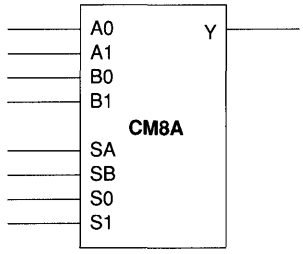
Buffers



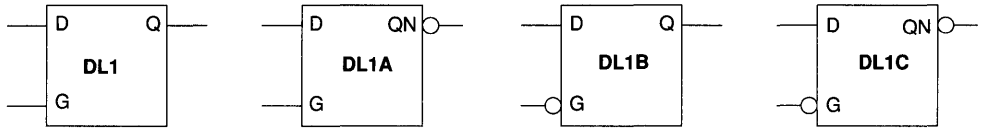
Multiplexors



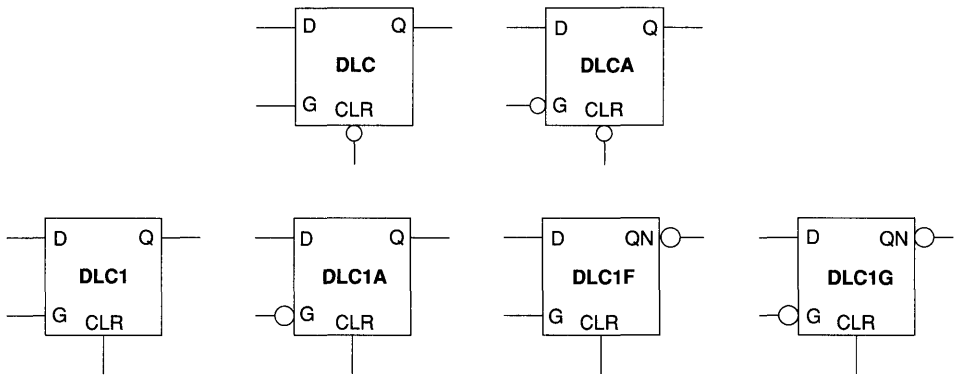
Combinatorial



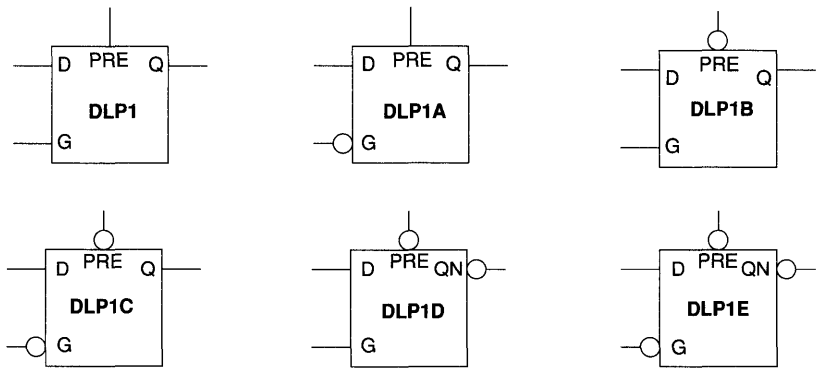
D-Latches



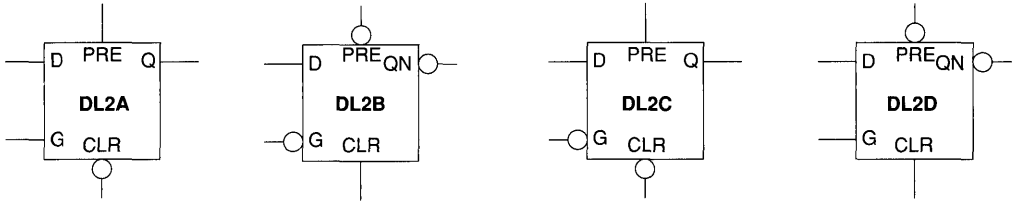
D-Latches with Clear



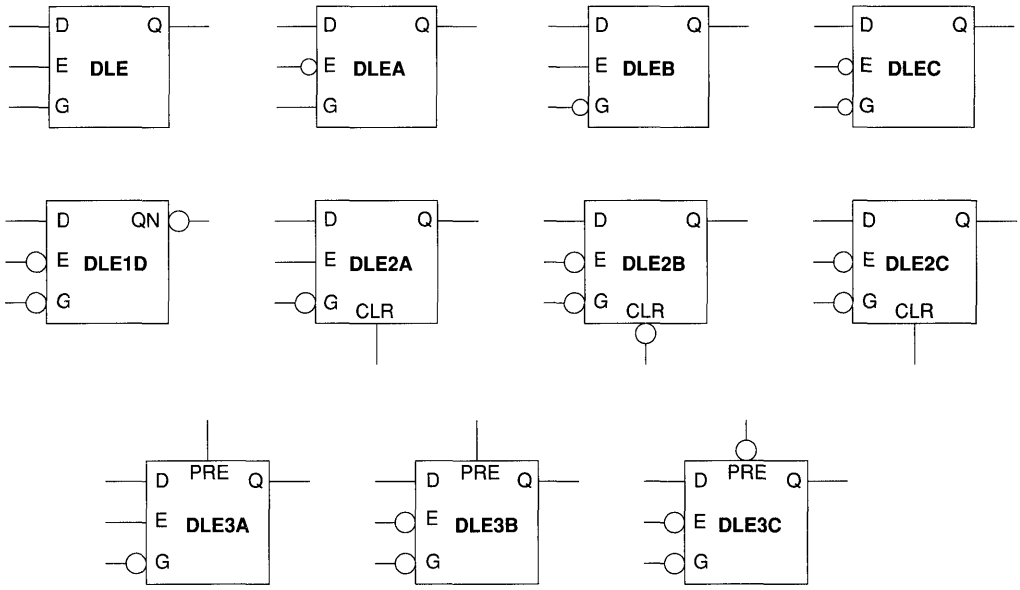
D-Latches wilth Preset



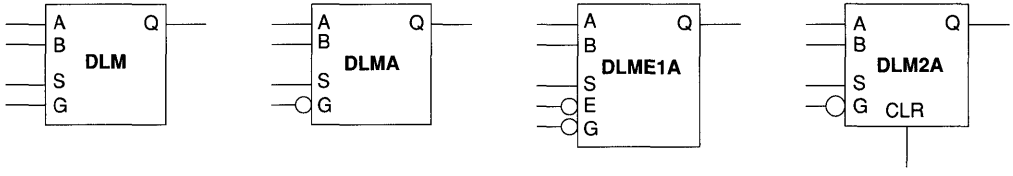
D-Latches with Preset and Clear



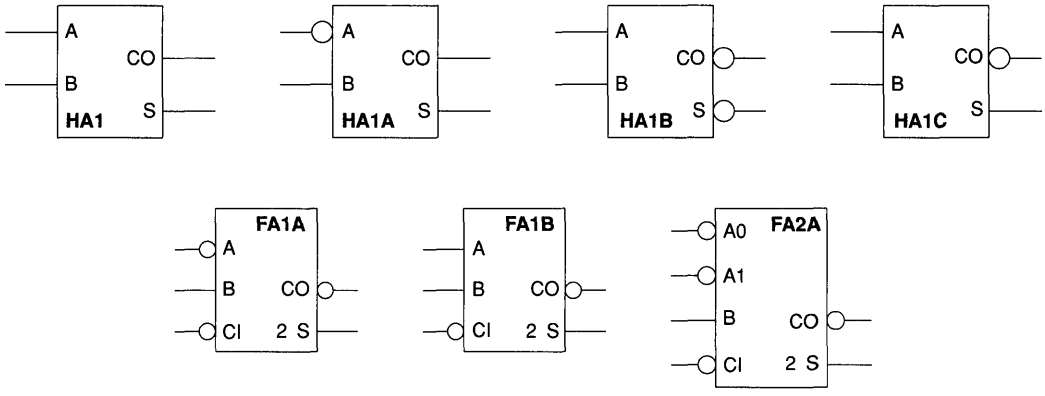
D-Latches with Enable



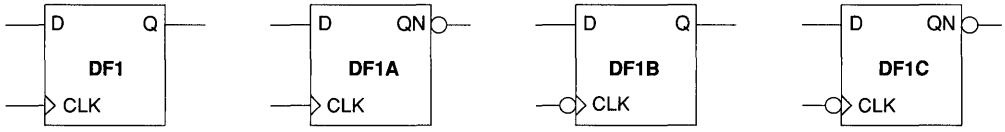
Mux Latches



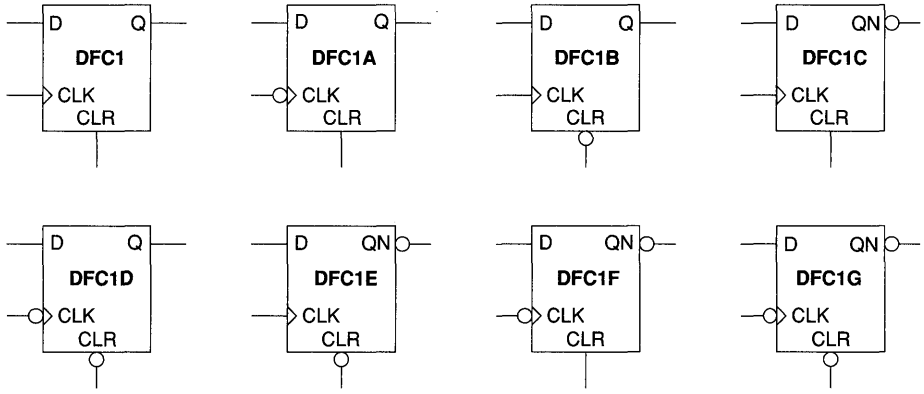
Adders



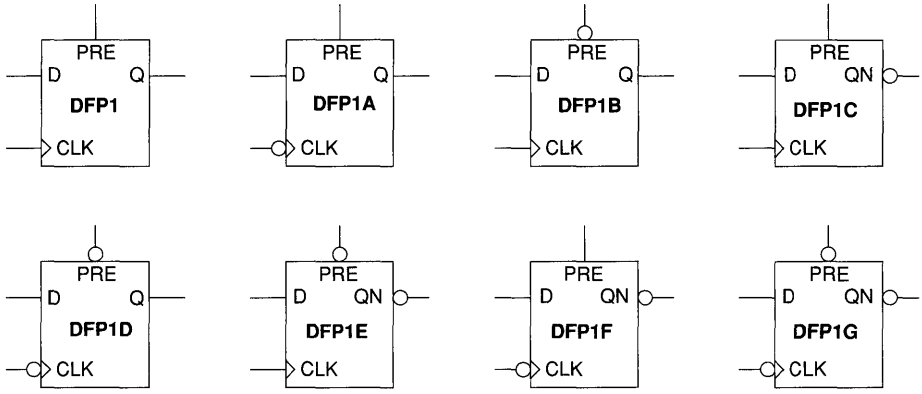
D-Type Flip-Flops



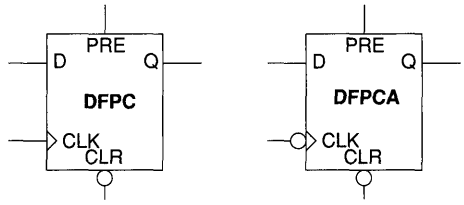
D-Type Flip-Flops with Clear



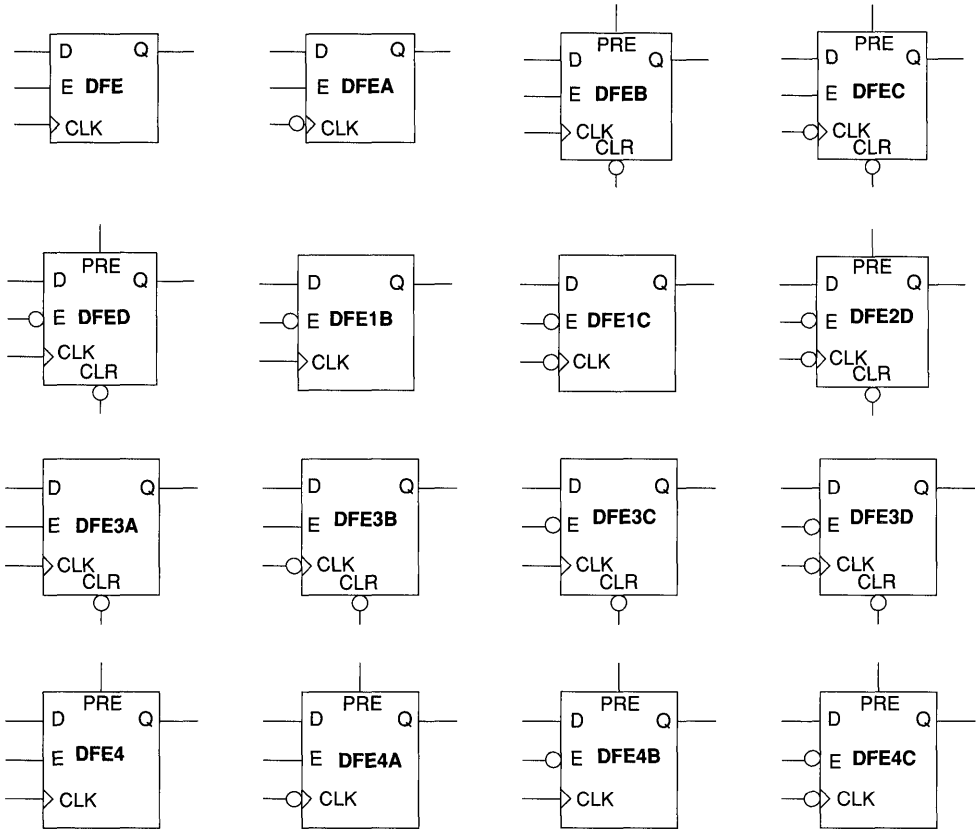
D-Type Flip-Flops with Preset



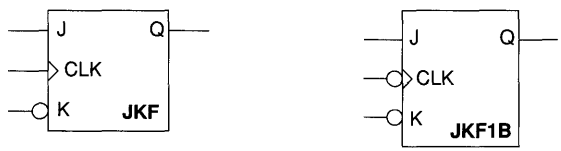
D-Type Flip-Flops with Preset and Clear



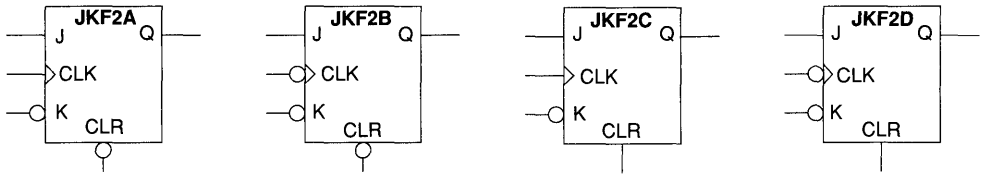
D-Type Flip-Flops with Enable



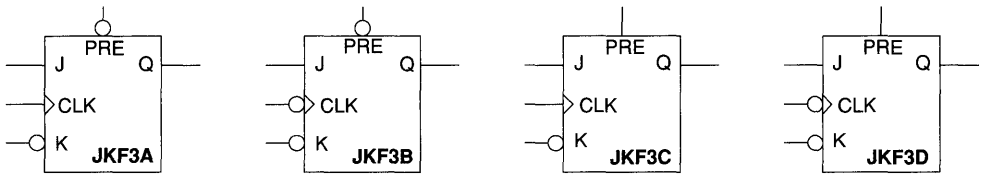
JK Flip-Flops



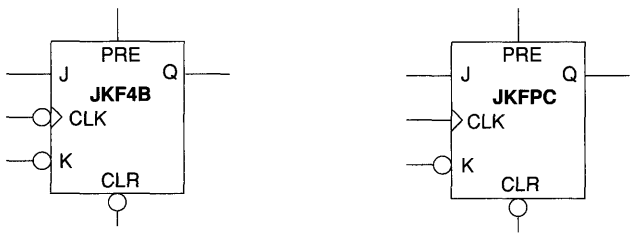
JK Flip-Flops with Clear



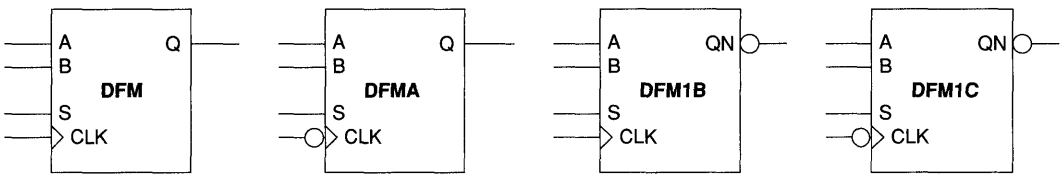
JK Flip-Flops with Preset



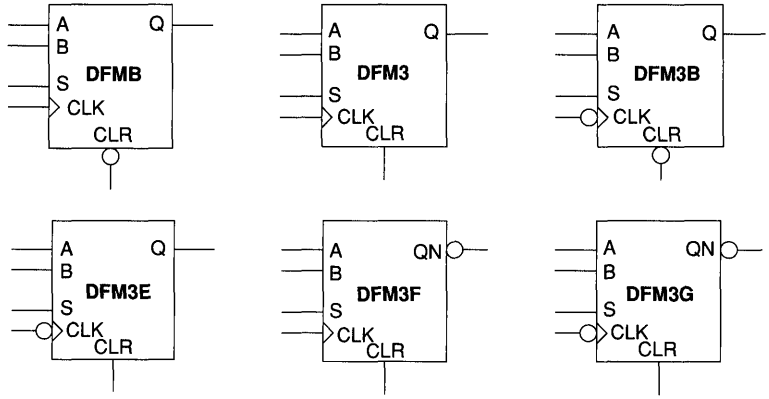
JK Flip-Flops with Preset and Clear



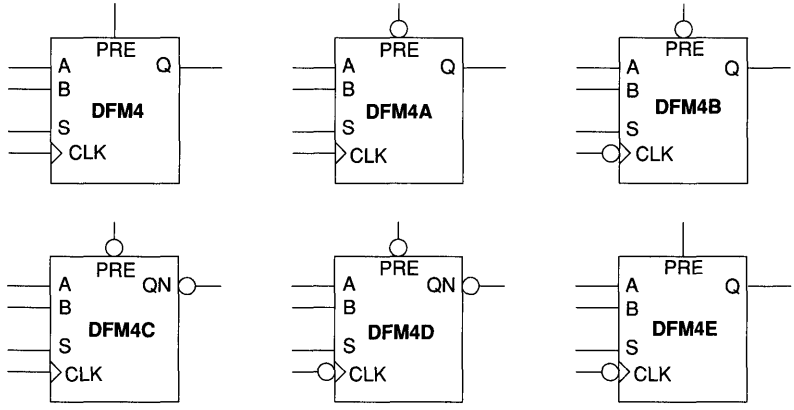
Mux Flip-Flops



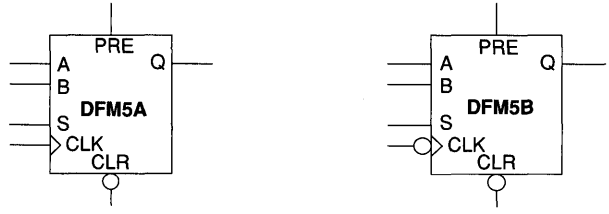
Mux Flip-Flops with Clear



Mux Flip-Flops with Preset



Mux Flip-Flops with Preset and Clear



ACT 2 and ACT 3
Hard Macro Library Overview

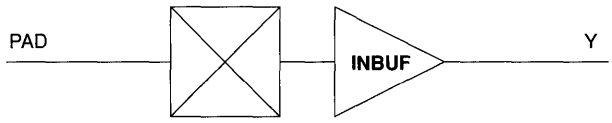
The following illustrations show all the available hard macros. The *ACT Family Macro Library Guide* contains module count, combinability, and pin loading information of each hard macro. It also includes a complete truth table for each macro.

Most ACT 2 and ACT 3 hard macros are implemented by a single logic module. The following ACT 2 and ACT 3 hard macros require more than one logic module to implement:

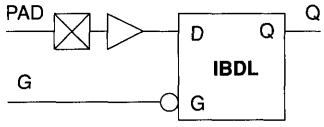
- Two-module hard macro gates: AND4D, AOI4, AX1A, MXC1, MXT, NAND4, NOR4, OAI3, OR4D
- Two-module adders: FA1A, FA1B, FA2A, HA1, HA1A, HA1B, HA1C
- Two-module latches: DL2A, DL2B, DL2C, DL2D, DLE2A, DLE3A, DLM2A
- Two-module flip-flops: DFC1, DFC1A, DFC1E, DFC1G, DFM3, DFM3E, DFP1, DFP1A, DFP1B, DFP1C, DFP1D, DFP1F, DFPC, DFPCA, JKF2C, JKF2D, JKF3A, JKF3B, JKF3C, JKF3D, JKF4B, JKFPC

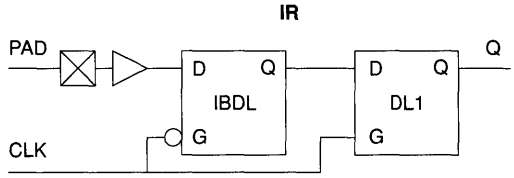
Two-module hard macro gates have a "2" displayed on some input pins. This indicates that the input to output path has two levels of logic delay for these input pins only. Also, the full adders have a "2" on the "S" output pins. This indicates that there are two levels of logic delay from the input pins to the "S" output pin. Refer to the ACT 2 and ACT 3 Timing Characteristics for detailed timing information of all ACT 2 and ACT 3 macros.

Input Buffers

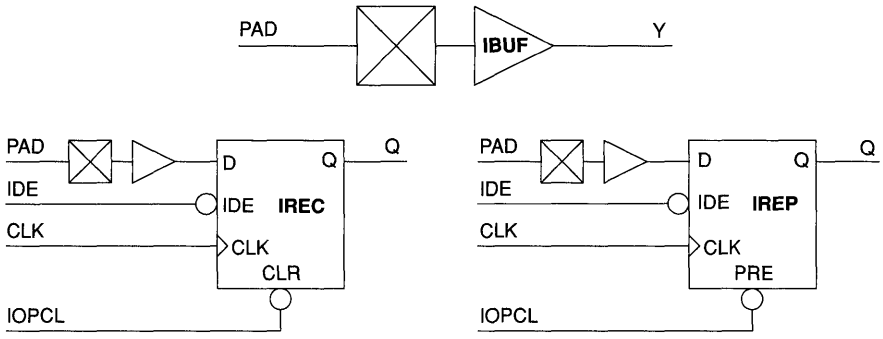


Input Buffers (ACT 2 only)

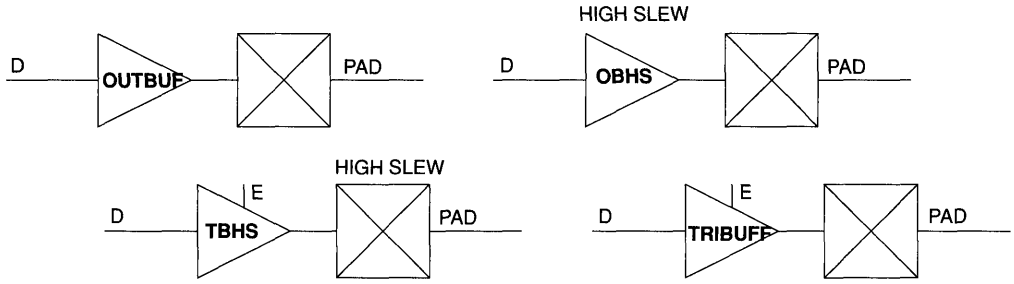




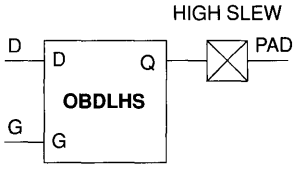
Input Buffers (ACT 3 only)

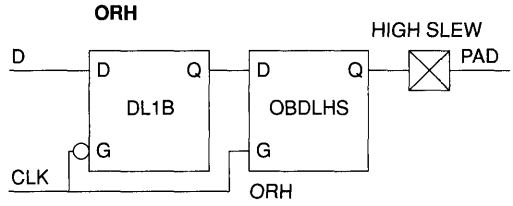


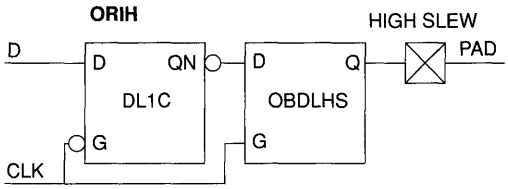
Output Buffers

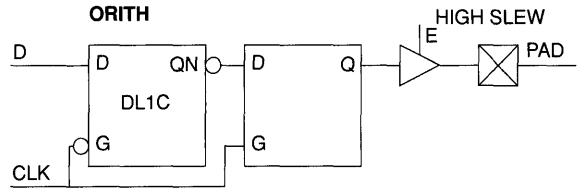


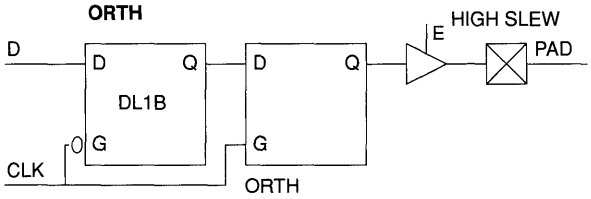
Output Buffers (ACT 2 only)

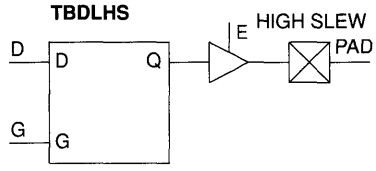






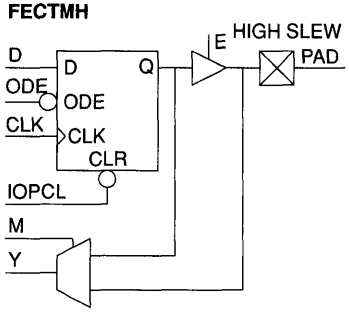




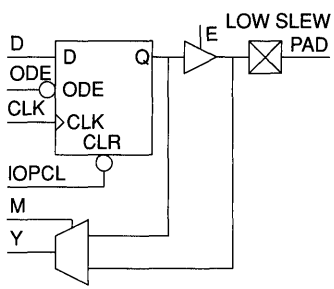


Output Buffers (ACT 3 only)

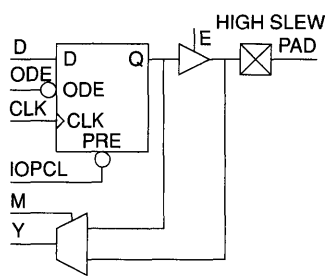
FECTMH



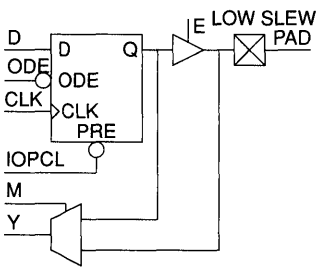
FECTML



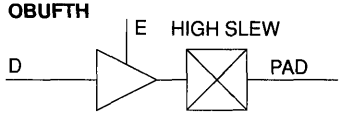
FEPTMH



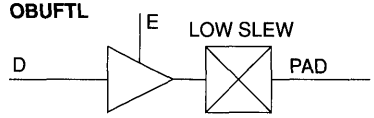
FEPTML



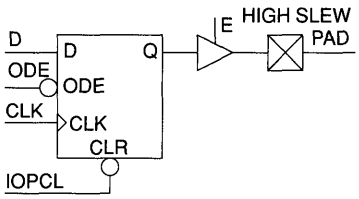
OBUFTH



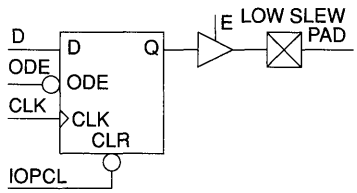
OBUFTL



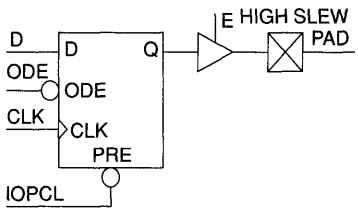
ORECTH



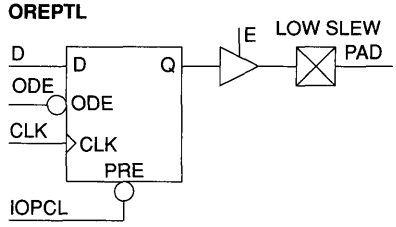
ORECTL



OREPTH

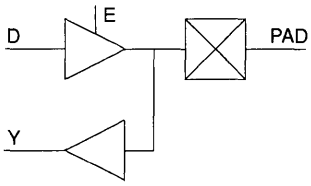


OREPTL

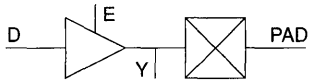


Bidirectional Buffers

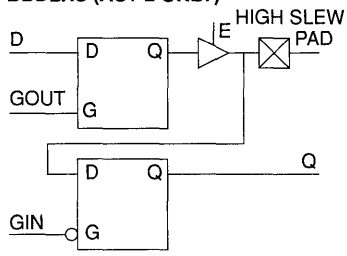
BBHS



BIBUF

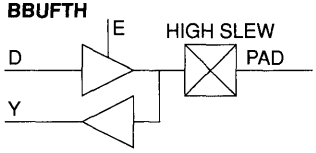


BBDLHS (ACT 2 ONLY)

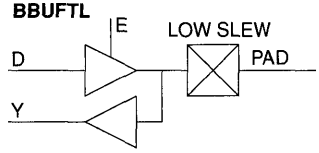


Bidirectional Buffers (ACT 3 only)

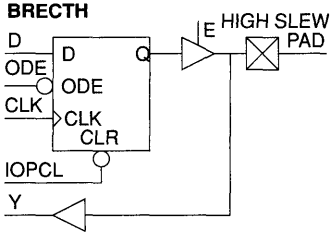
BBUFTH



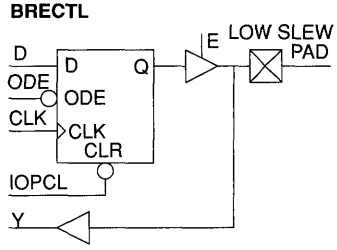
BBUFTL



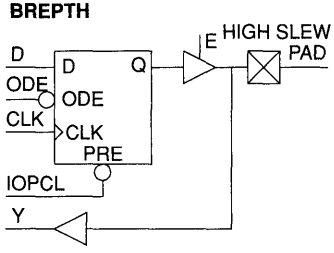
BRECTH



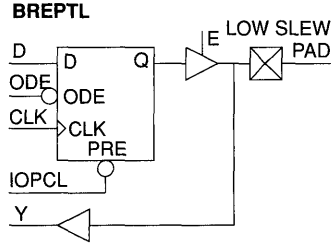
BRECTL



BREPTH

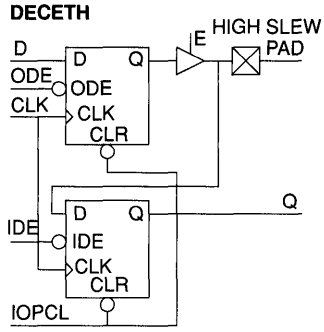


BREPTL

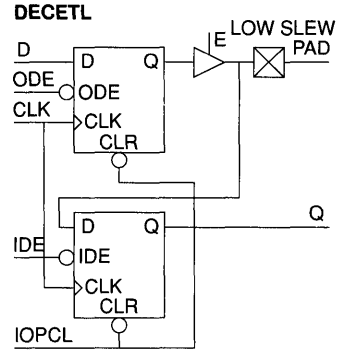


Bidirectional Buffers (ACT 3 only) (continued)

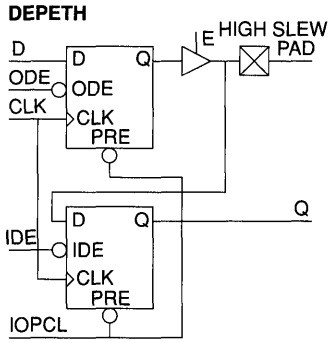
DECETH



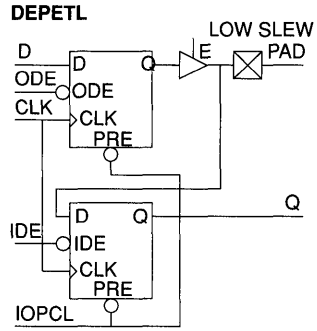
DECETL



DEPETH

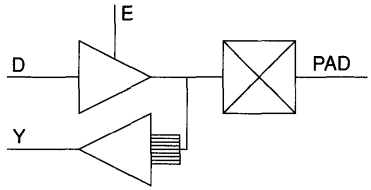


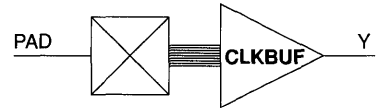
DEPETL

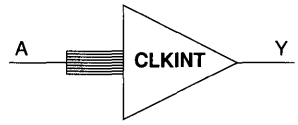


Clock (Dedicated Network) Buffers

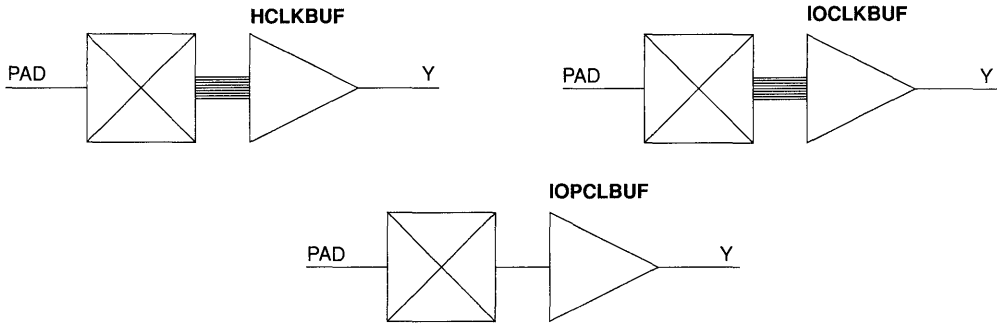
CLKBIBUF



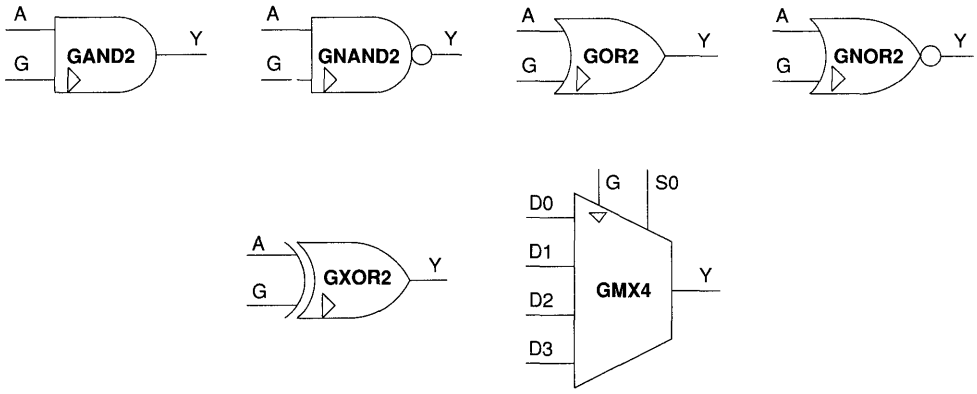




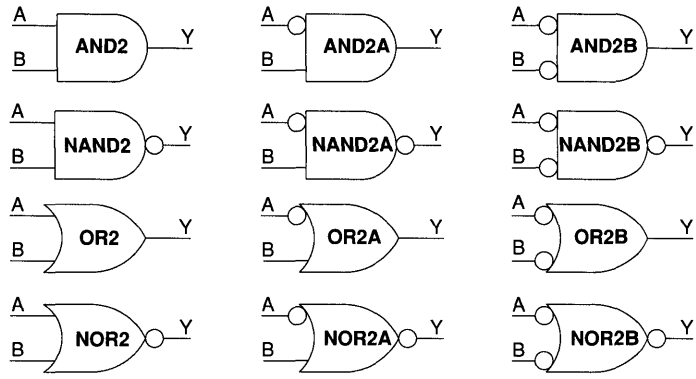
Clock (Dedicated Network) Buffers (ACT 3 only)



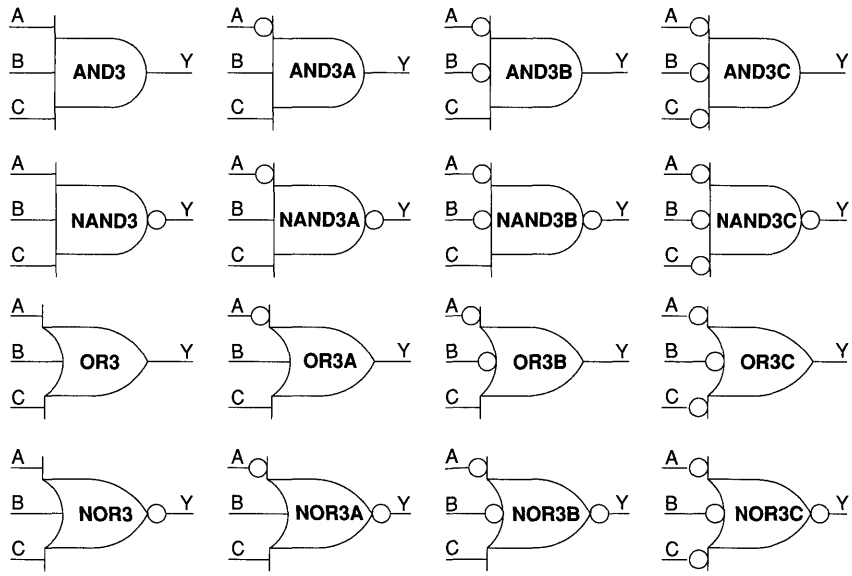
CLKBUF Interface Macros



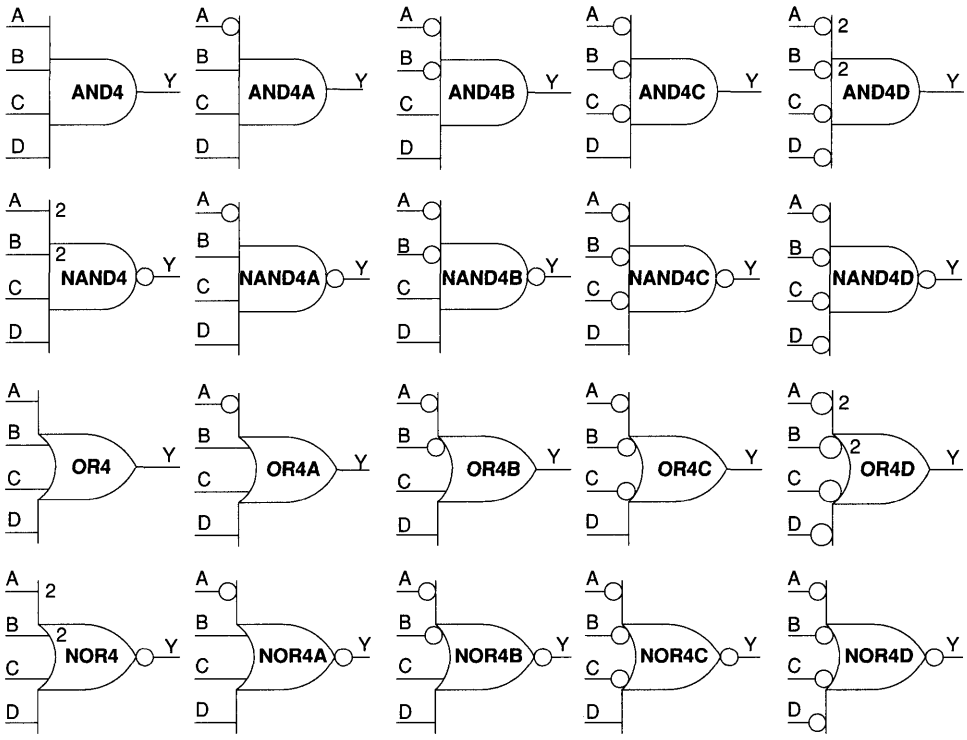
2-Input Gates



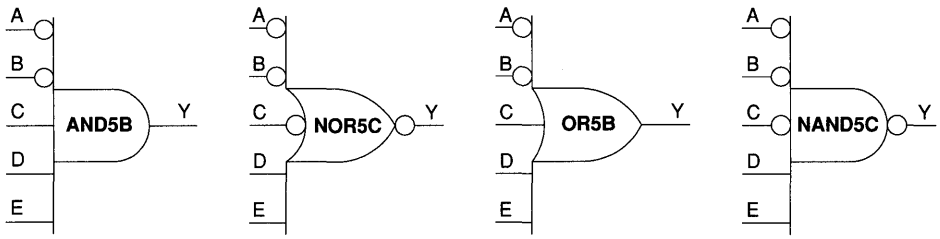
3-Input Gates



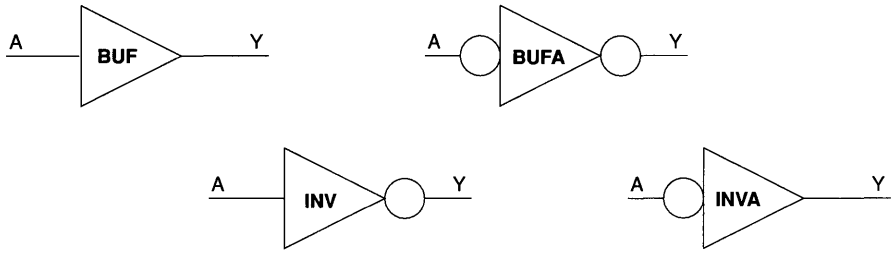
4-Input Gates



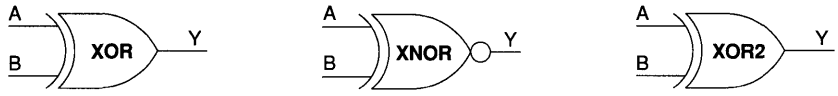
5-Input Gates



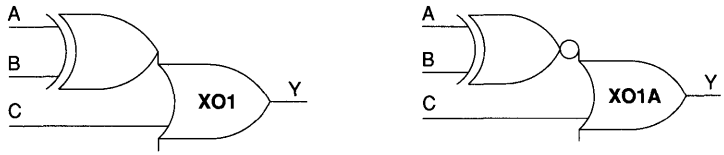
Buffers



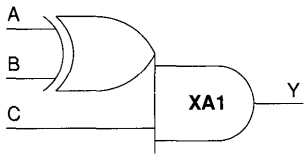
XOR Gates

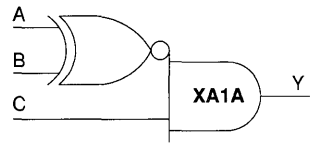


XOR-OR Gates

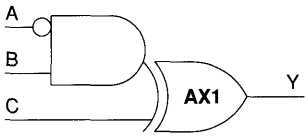


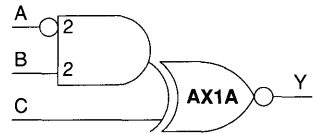
XOR-AND Gates

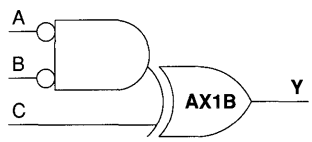


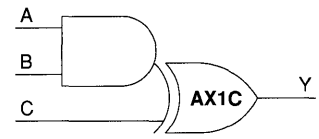


AND-XOR Gates



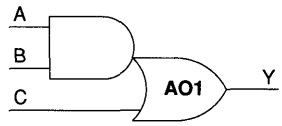


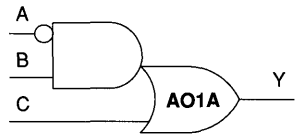


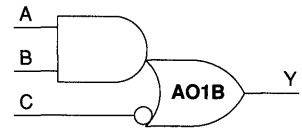


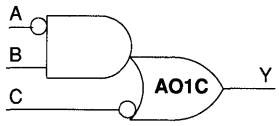
5

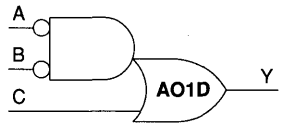
AND-OR Gates

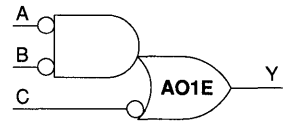

A
B
C
AO1
Y

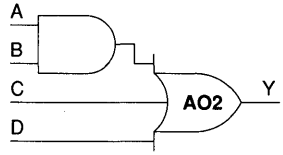

A
B
C
AO1A
Y

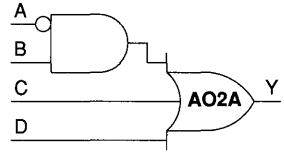

A
B
C
AO1B
Y

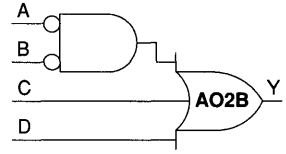

A
B
C
AO1C
Y

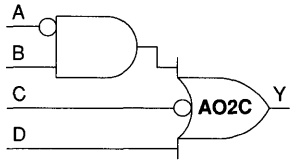

A
B
C
AO1D
Y

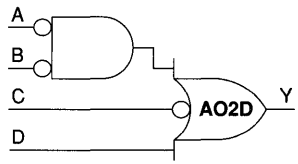

A
B
C
AO1E
Y

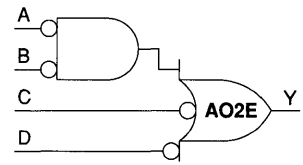

A
B
C
D
AO2
Y

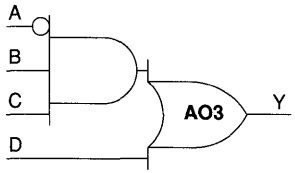

A
B
C
D
AO2A
Y

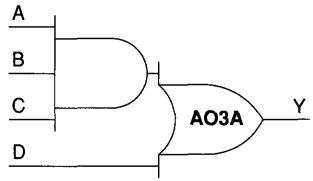

A
B
C
D
AO2B
Y

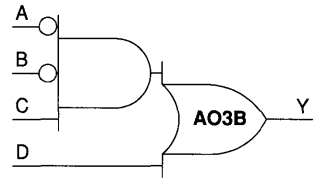

A
B
C
D
AO2C
Y

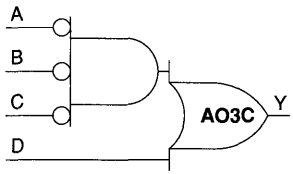

A
B
C
D
AO2D
Y

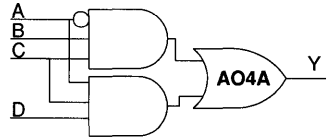

A
B
C
D
AO2E
Y

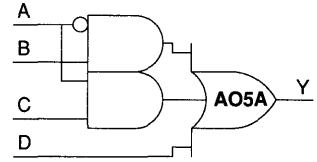

A
B
C
D
AO3
Y

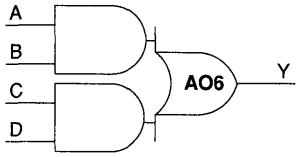

A
B
C
D
AO3A
Y

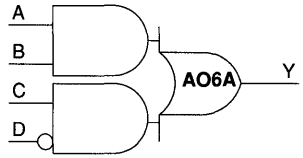

A
B
C
D
AO3B
Y

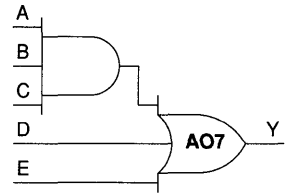

A
B
C
D
AO3C
Y


A
B
C
D
AO4A
Y

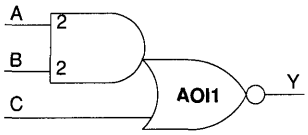

A
B
C
D
AO5A
Y

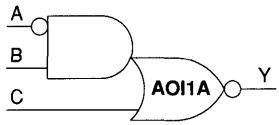

A
B
C
D
AO6
Y

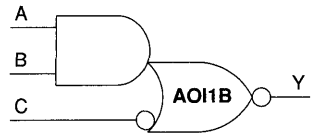

A
B
C
D
AO6A
Y

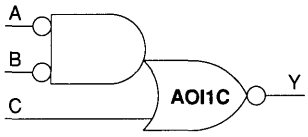

A
B
C
D
E
AO7
Y

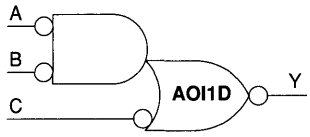
AND-OR Gates (continued)

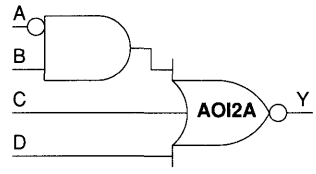


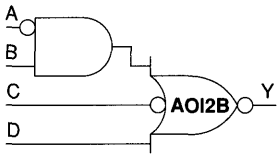


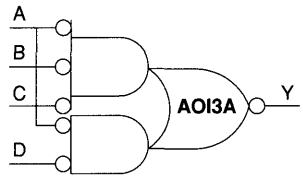


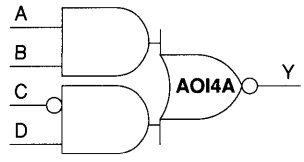


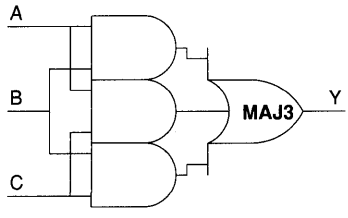


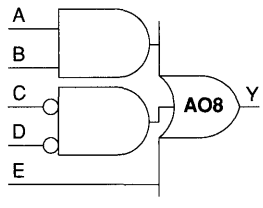


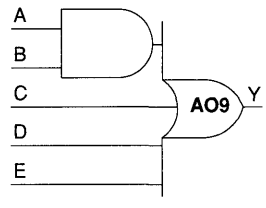


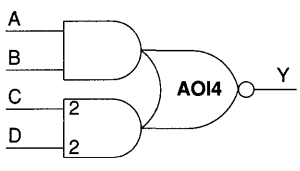


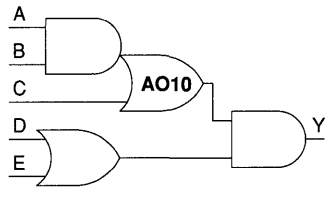


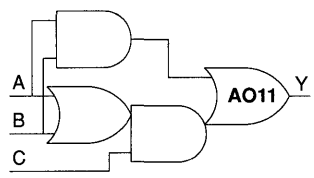




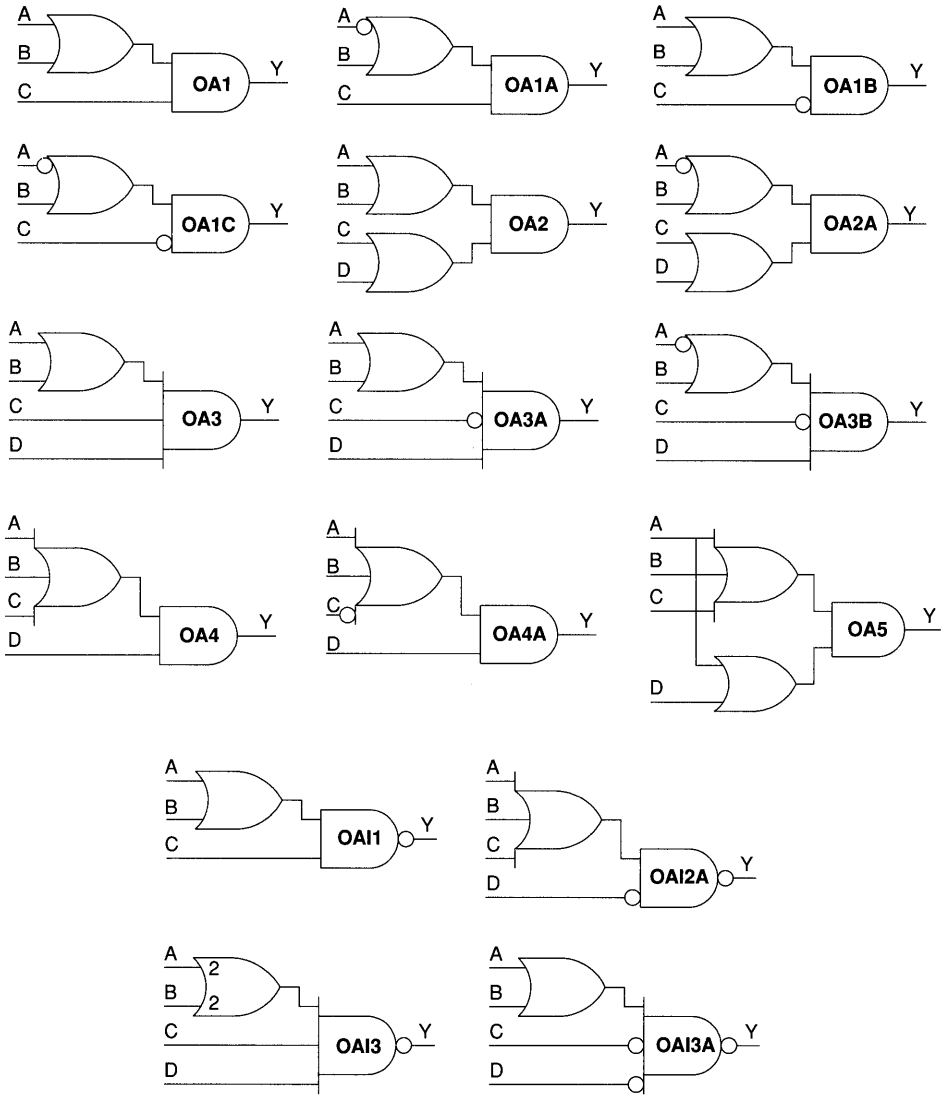




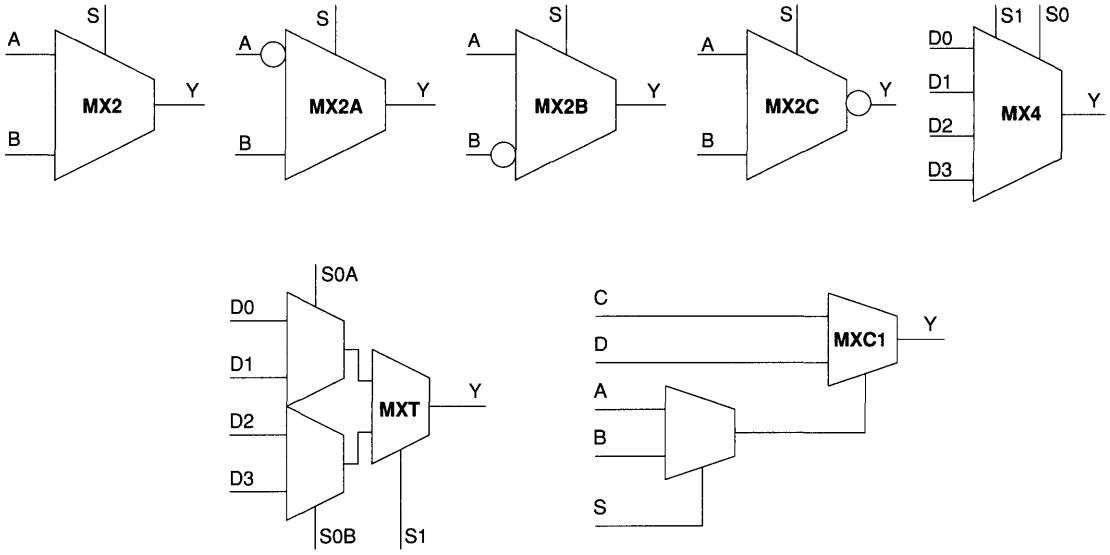




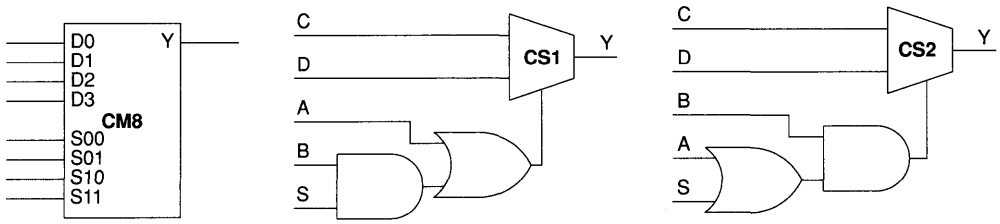
OR-AND Gates


A
B
C
OA1
Y
A
B
C
OA1A
Y
A
B
C
OA1B
Y
A
B
C
OA1C
Y
A
B
C
D
OA2
Y
A
B
C
D
OA2A
Y
A
B
C
D
OA3
Y
A
B
C
D
OA3A
Y
A
B
C
D
OA3B
Y
A
B
C
D
OA4
Y
A
B
C
D
OA4A
Y
A
B
C
D
OA5
Y
A
B
C
OAI1
Y
A
B
C
D
OAI2A
Y
A
B
2
2
C
D
OAI3
Y
A
B
C
D
OAI3A
Y

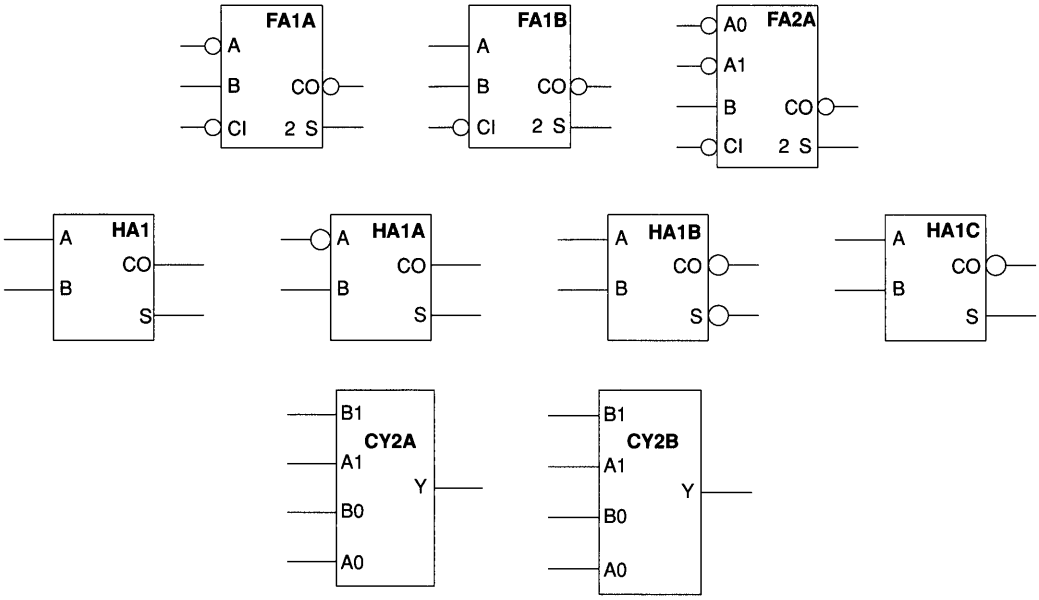
Multiplexors



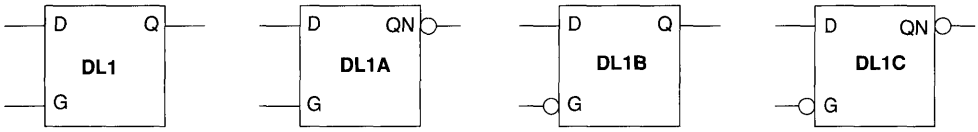
Combinatorial



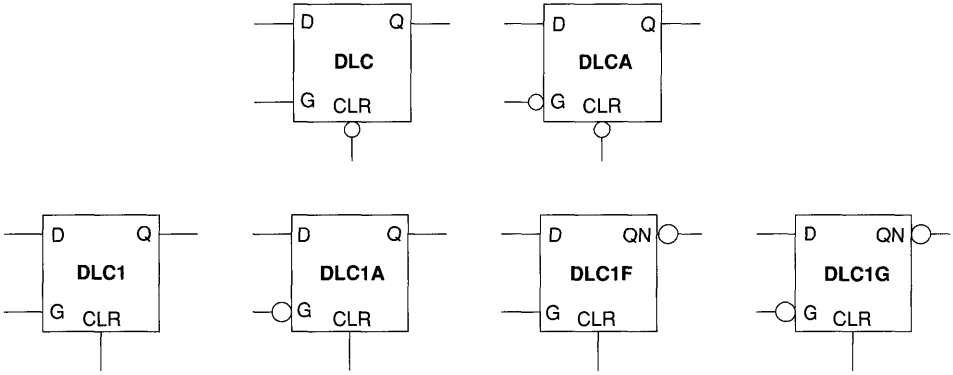
Adders


FA1A
A
B
CI
CO
2 S
FA1B
A
B
CI
CO
2 S
FA2A
A0
A1
B
CI
CO
2 S
HA1
A
B
CO
S
HA1A
A
B
CO
S
HA1B
A
B
CO
S
HA1C
A
B
CO
S
CY2A
B1
A1
B0
A0
Y
CY2B
B1
A1
B0
A0
Y

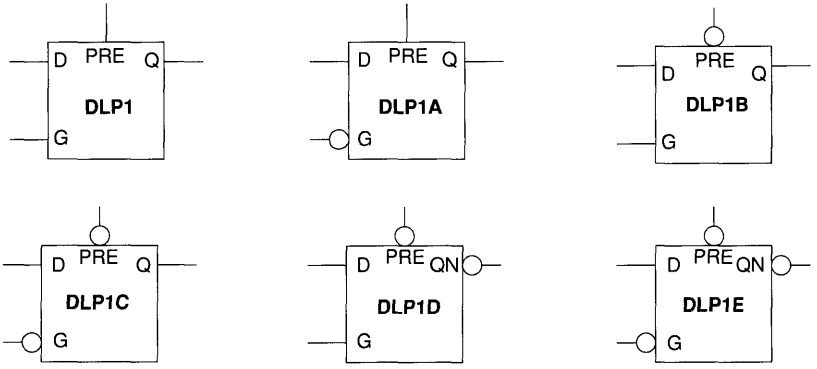
D-Latches



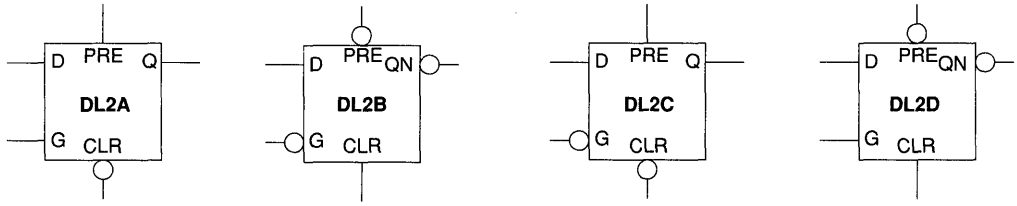
D-Latches with Clear



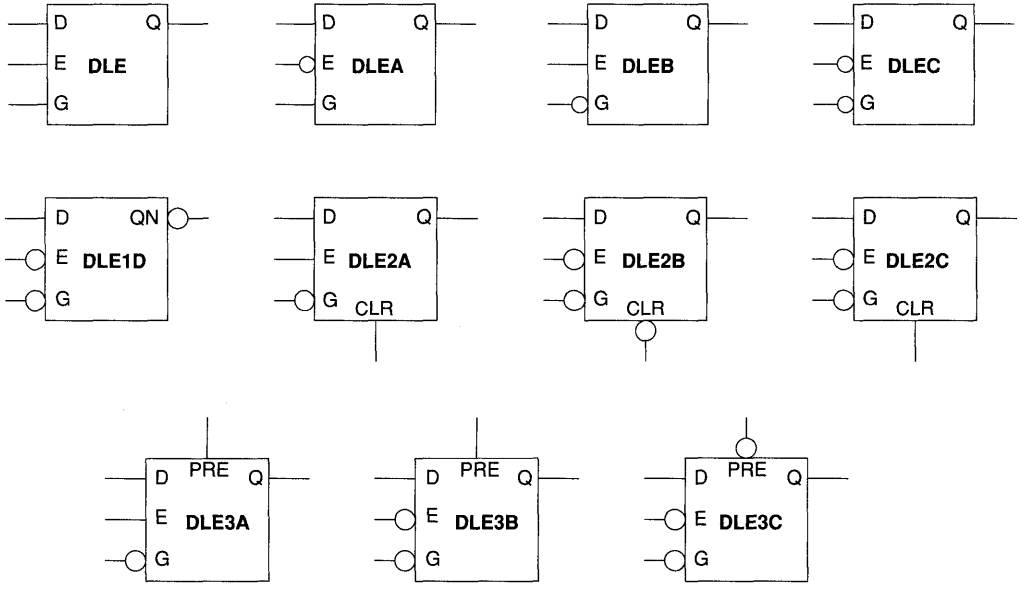
D-Latches with Preset



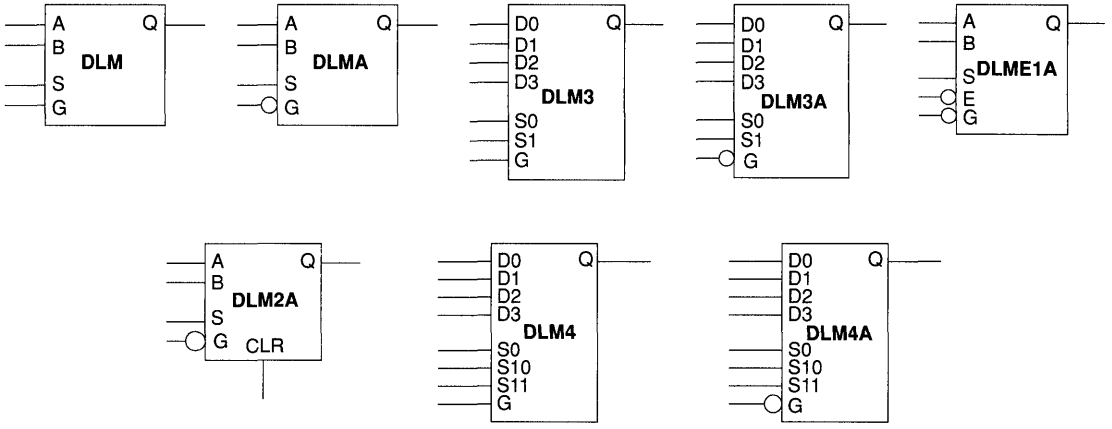
D-Latches with Preset and Clear



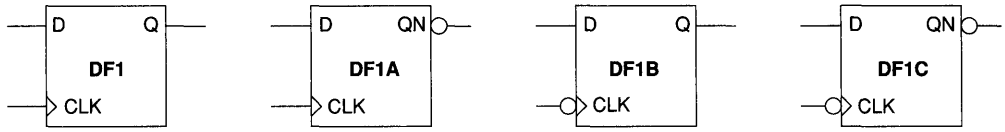
D-Latches with Enable



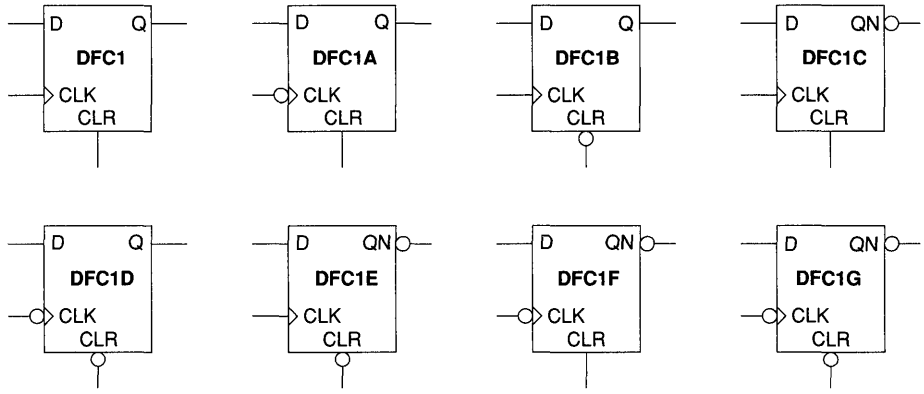
Mux Latches


A
B
S
G
Q
DLM
A
B
S
G
Q
DLMA
D0
D1
D2
D3
S0
S1
G
Q
DLM3
D0
D1
D2
D3
S0
S1
G
Q
DLM3A
A
B
S
E
G
Q
DLME1A
A
B
S
G
CLR
Q
DLM2A
D0
D1
D2
D3
S0
S10
S11
G
Q
DLM4
D0
D1
D2
D3
S0
S10
S11
G
Q
DLM4A

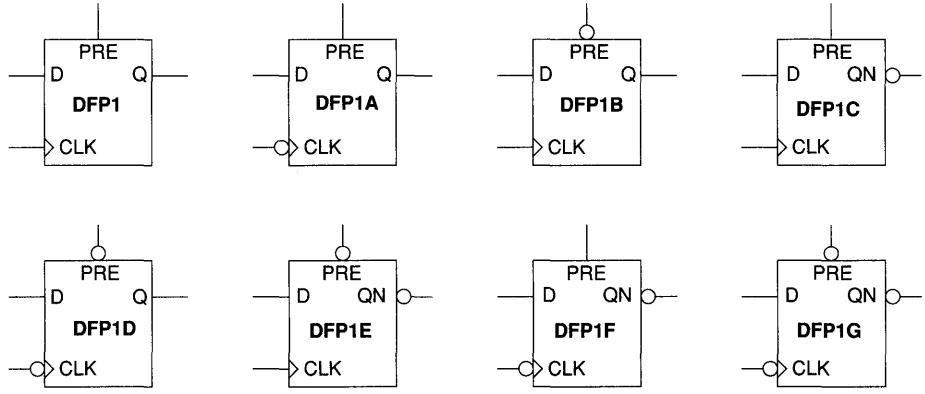
D-Type Flip-Flops



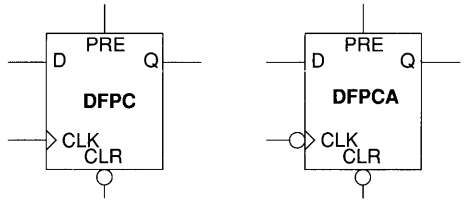
D-Type Flip-Flops with Clear



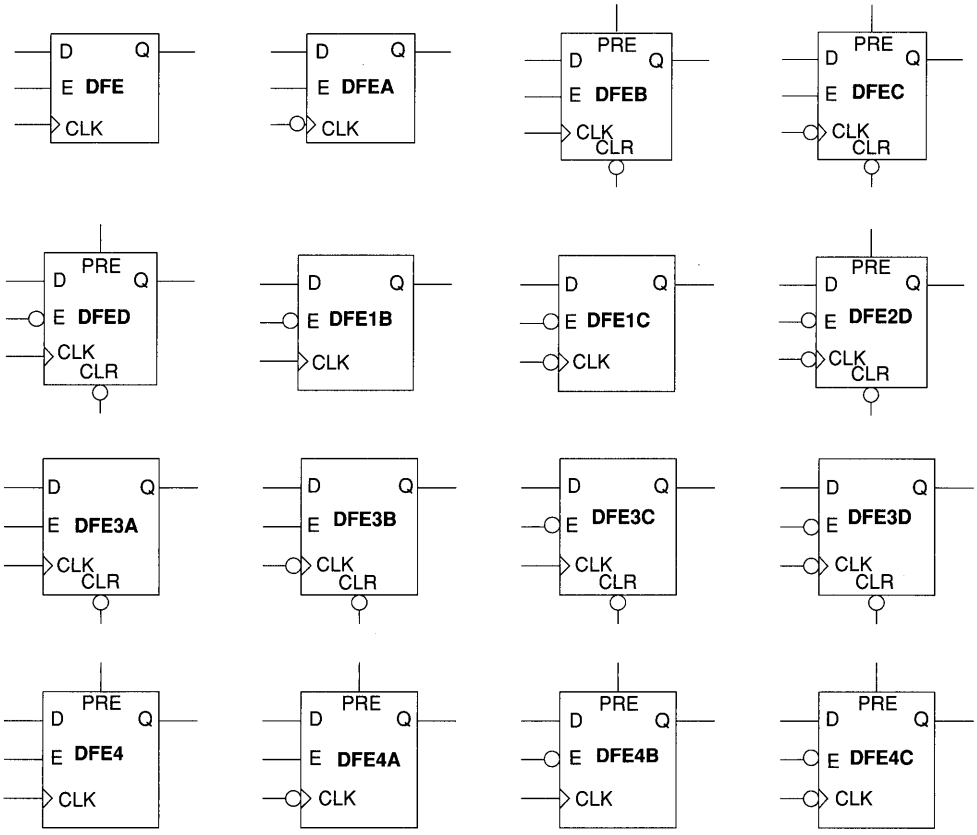
D-Type Flip-Flops with Preset



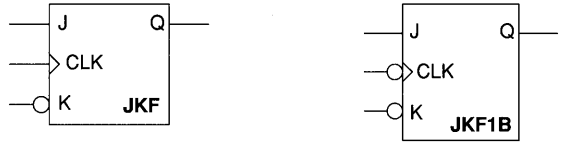
D-Type Flip-Flops with Preset and Clear



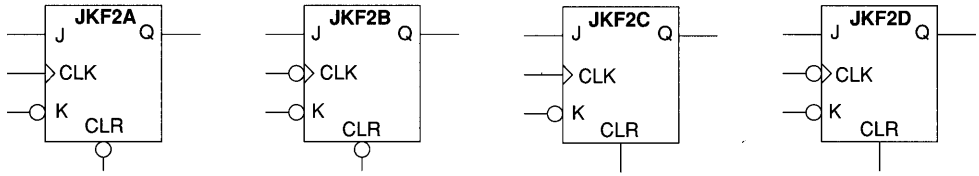
D-Type Flip-Flops with Enable



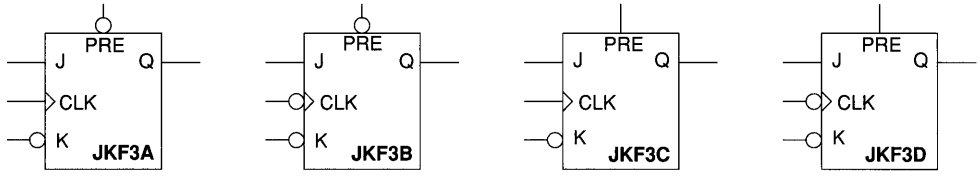
JK Flip-Flops



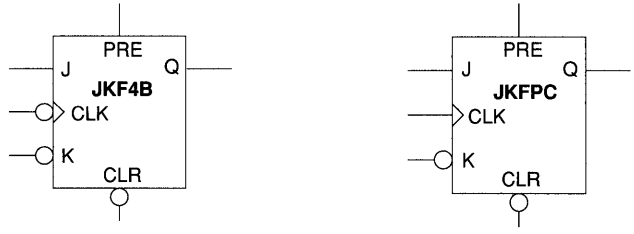
JK Flip-Flops with Clear



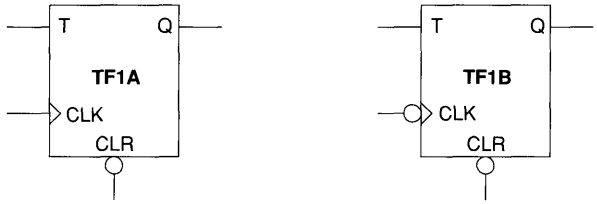
JK Flip-Flops with Preset



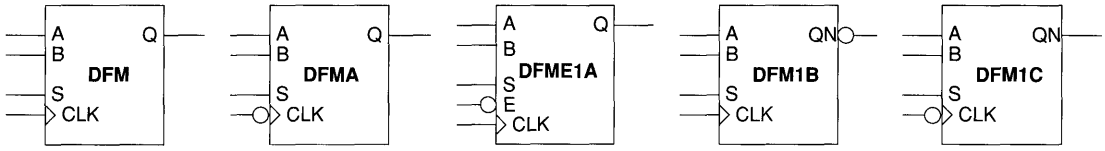
JK Flip-Flops with Preset and Clear



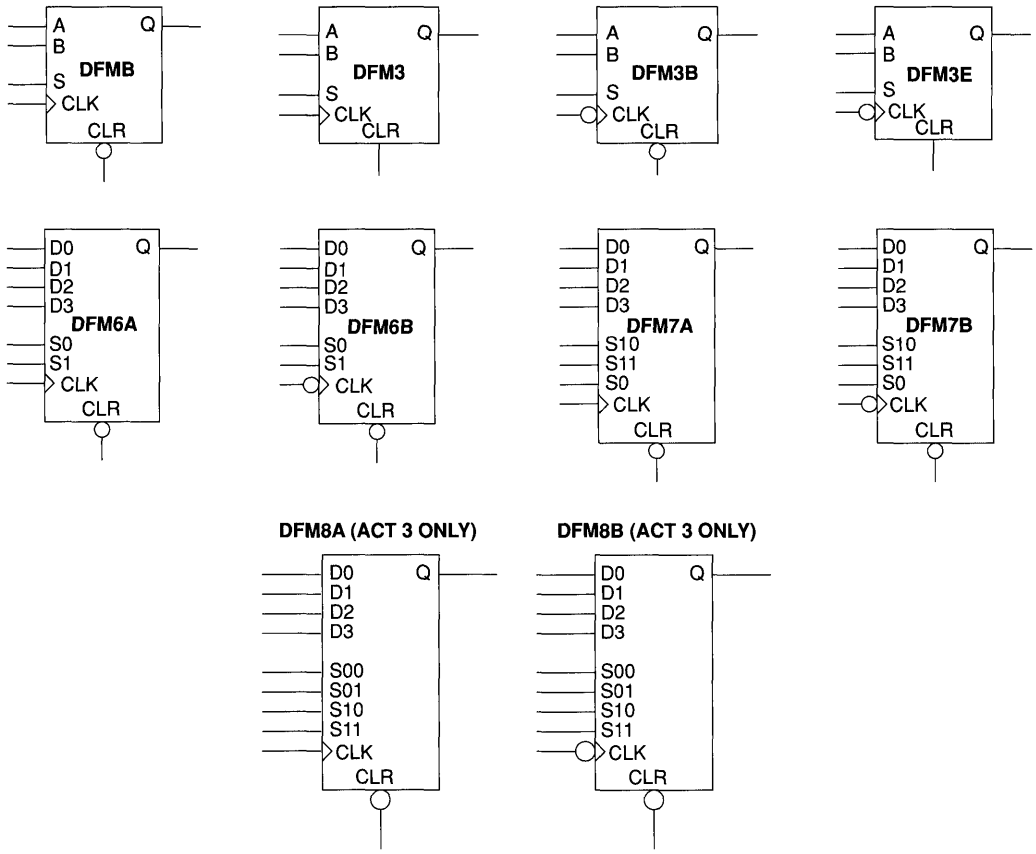
Toggle Flip-Flops



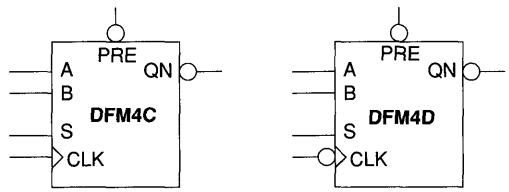
Mux Flip-Flops



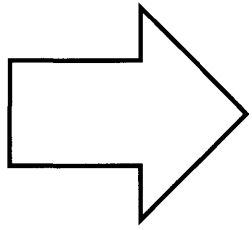
Mux Flip-Flops with Clear



Mux Flip-Flops with Preset



EDN-Special Report: Hands-on FPGA Project



EDN

ELECTRONIC TECHNOLOGY FOR ENGINEERS AND ENGINEERING MANAGERS WORLDWIDE

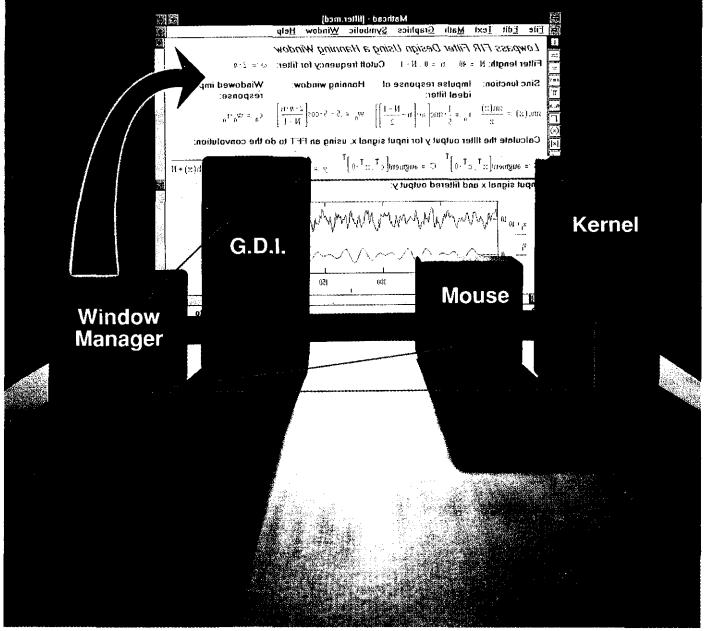
A CAHNERS PUBLICATION

April 9, 1992

Special Report:

An inside look at Windows for engineering software

pg 122



SPECIAL PROJECT

Hands-on FPGA design project Part 1
pg 98

SPECIAL REPORT

Windows and engineering software
pg 122

DESIGN FEATURE

Improve reliability by rigging pc boards for in-circuit programming
pg 135

TECHNOLOGY UPDATES

Design software links active-filter performance with real devices
pg 45

FDDI routers and bridges create niche for content-addressable memories
pg 61

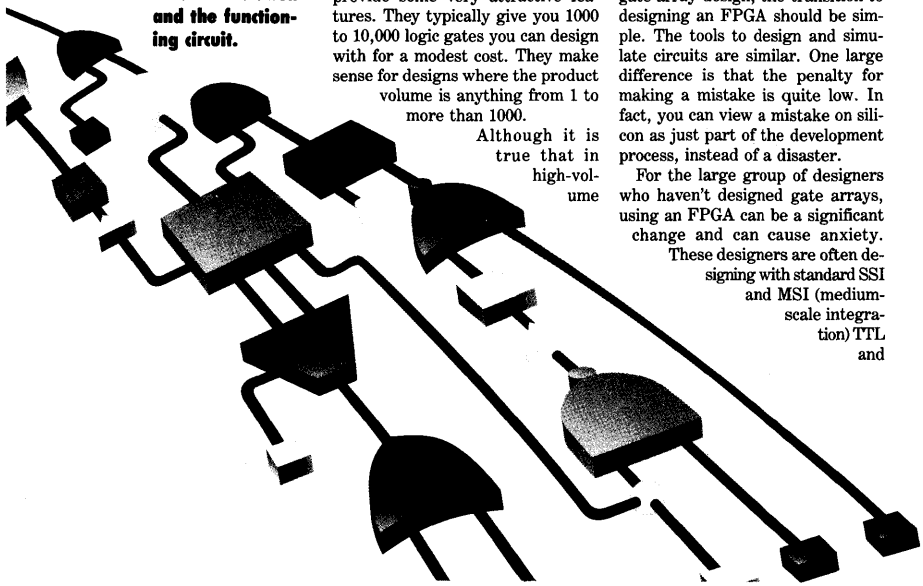
Product Updates
pg 73

EDN·SPECIAL PROJECT

FPGA—Taking the first steps

If you're considering designing with FPGAs, this 2-part hands-on design project will show you exactly what is involved. Part 1 covers the design and schematic entry, and part 2 covers simulation and the functioning circuit.

DOUG CONNER, Technical Editor



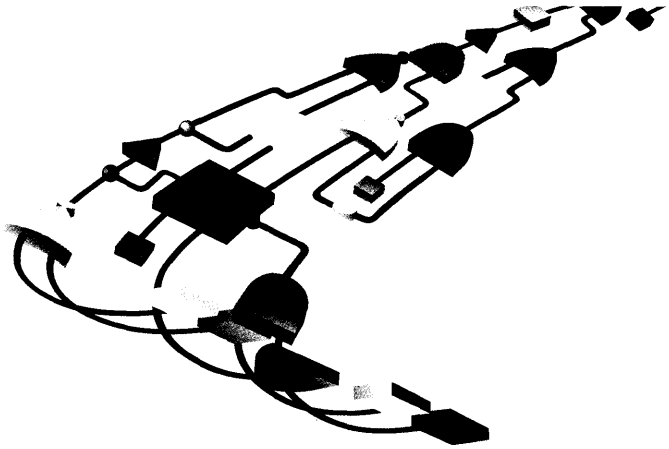
The fear and uncertainty of making a major shift in your design and development methodology is always compounded by tight schedules. As a result, you may be putting off designing with field-programmable gate arrays (FPGAs) because you don't know what to expect from them and you don't have the time to find out.

FPGAs and high-density PLDs provide some very attractive features. They typically give you 1000 to 10,000 logic gates you can design with for a modest cost. They make sense for designs where the product volume is anything from 1 to more than 1000.

Although it is true that in high-volume production a masked gate array can offer substantial savings, it is also true that they offer a much larger financial commitment up front. Penalties for an inexperienced designer who makes a design mistake or a system-definition mistake is high, both financially and in time lost in making another design turn.

For a designer experienced with gate-array design, the transition to designing an FPGA should be simple. The tools to design and simulate circuits are similar. One large difference is that the penalty for making a mistake is quite low. In fact, you can view a mistake on silicon as just part of the development process, instead of a disaster.

For the large group of designers who haven't designed gate arrays, using an FPGA can be a significant change and can cause anxiety. These designers are often designing with standard SSI and MSI (medium-scale integration) TTL and

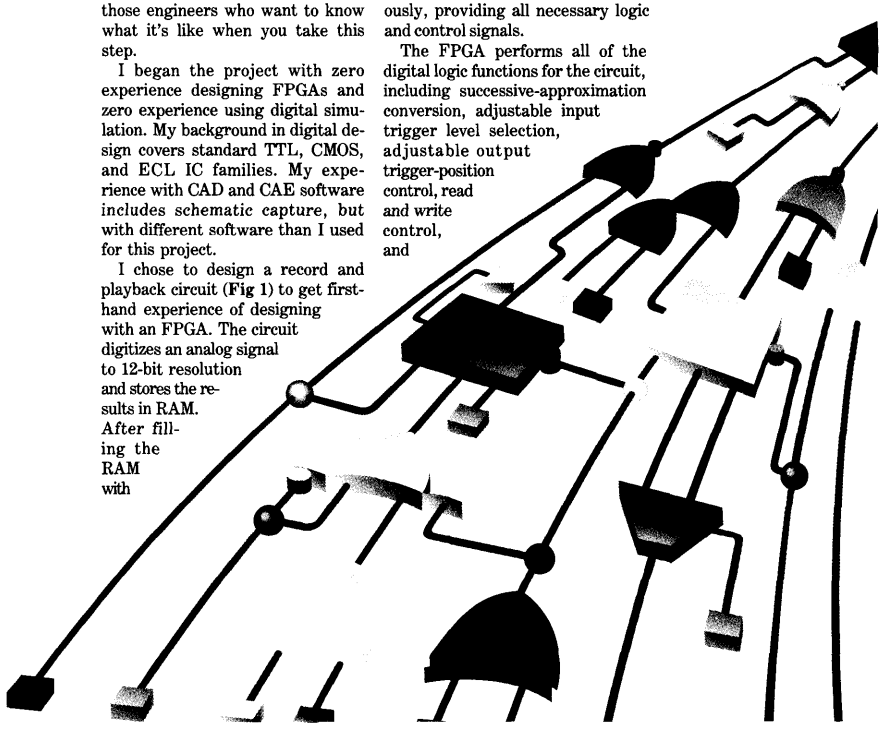


CMOS devices that interface to microprocessors, analog circuits, or both. Many have never used digital simulation. Moving to FPGAs is a step up for them. This project is for those engineers who want to know what it's like when you take this step.

I began the project with zero experience designing FPGAs and zero experience using digital simulation. My background in digital design covers standard TTL, CMOS, and ECL IC families. My experience with CAD and CAE software includes schematic capture, but with different software than I used for this project.

I chose to design a record and playback circuit (**Fig 1**) to get first-hand experience of designing with an FPGA. The circuit digitizes an analog signal to 12-bit resolution and stores the results in RAM. After filling the RAM with 32k words of data, it plays back the data, reconverting it to analog. The circuit is designed to work with an analog oscilloscope to capture a one-time event and play it back continuously, providing all necessary logic and control signals.

The FPGA performs all of the digital logic functions for the circuit, including successive-approximation conversion, adjustable input trigger level selection, adjustable output trigger-position control, read and write control, and



addressing the RAM. The design incorporates more than 1500 true logic gates and includes large regular structures, such as counters and compare circuits, plus plenty of gate-and register-level logic. (For a detailed circuit description and schematics for the full circuit, see **box**, "Pack the digital logic into one FPGA.")

Selecting the FPGA

I decided to use the Actel Act 1 FPGA family for my design. The choice of Actel was an arbitrary one—there are perhaps a dozen companies with products that fall into the FPGA and complex-PLD category that are appropriate for my design (**Ref 1**).

I chose the Act 1 family over Actel's higher performance and higher density Act 2 family because I didn't need the extra features. And, the Act 1 family costs less—the A1020A FPGA costs $36.25 (100).

To begin the project, I took Actel's 2-day training class. The class is included in the price of a system ($2950), or you can purchase it separately for $495. The class takes you through the process of designing an FPGA with Viewlogic schematic capture and simulation tools, and Actel's ALS software tools for all other functions. The basic design flow is shown in **Fig 2**.

The class uses canned files that you modify. For example, you'll add some components to a partially completed schematic to finish it. The class runs at a reasonably fast pace, but you won't fall behind even if you're unable to complete a step in the time allotted, because finished files are available. For example, if you haven't finished the schematic when it's time to move on to simulation, you can use a file that contains the completed schematic.

The class also covers some tools I didn't use in the project. A synthesis tool (ALES) lets you convert Boolean equations directly into logic. You can use the synthesized logic blocks in your schematic as you would use other macro symbols. Another tool, called the Timer, is a static timing tool that lets you look at path delays, both before layout and after place and route. At the end of the class you program an FPGA that contains a timing circuit and drives a 7-segment display.

Because Viewlogic CAE tools were used in the class, I elected to use them on the project, although Actel provides libraries and support for a variety of other workstation and PC-based tools.

Illustration by Gregg Dinderman

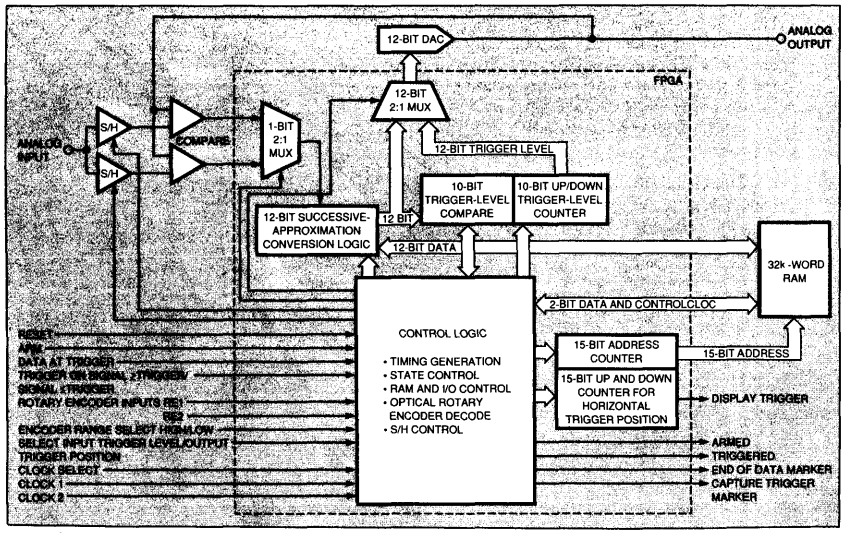


Fig 1—The FPGA contains all the digital logic of this record and playback circuit. The circuit converts a ±5V signal to 12-bit resolution at 167 ksamples/sec and plays it back continuously for viewing on an analog oscilloscope.

EDN-SPECIAL PROJECT

The building block on an Actel Act 1 FPGA is a logic module. What you actually design with is a logic module or group of logic modules configured as a hard or soft macro. A logic module starts as a flexible uncommitted block of logic; it can perform many different logic functions depending on how its connections are programmed. Actel provides hard macros, which define the logic-module connections to perform specific functions.

The hard-macro building blocks for designing an Actel FPGA are gates, gate combinations, latches, flip-flops, multiplexers, adders, and buffers. You can also configure every I/O pin as an input buffer, an output buffer, a bidirectional buffer, or a 3-state buffer. One input pin is designated as a clock buffer. You can see many of the basic building blocks and variations on pages of the circuit schematic (see **Figs 4** to **15**, which begin on pg 107).

Designing with the FPGA building blocks is similar to designing with 7400 series SSI devices, except in most cases the FPGAs are more flexible. For example, 2- and 3-input AND gates are available with any or all of their inputs inverted. You can select D flip-flops with positive clear, negative clear, and so on. Every gate macro I used requires a single module. Even a relatively complex gate combination, such as the 4-input AND/OR gate shown in **Fig 15**, is a single module. Although there are a few combinations that require two modules, I was able to avoid using them.

Latches also require only one module, even with a clear, an enable, or multiplexed inputs. Flip-flops, however, require two modules. In cases where a latch will work as well as a flip-flop, the module savings makes the latch a better choice. For example, the circuit needed to generate the DLY shown at the bottom of **Fig 6** uses two latches instead of flip-flops.

Another gate-saving consideration is to use multiplexed data inputs on both latches and flip-flops to bring 2-input gates inside them. The result saves a module. For example, the latch generating DISP_TRIG in **Fig 6** effectively ANDs together DISP_TM and PLYBK.

Part way through the design, I learned that the ALS software automatically combines 2-input gates with flip-flops and latches wherever possible. Therefore, you can see cases where I've left the gate separate, such as the latch and AND gate in **Fig 15**. The schematic is easier to read with the AND gate separate, so I'd recommend letting the software do its job. The end result on the FPGA is the same.

When your design calls for larger blocks (such as counters, adders, multipliers, decoders, and large registers), you've got several choices. You can use a soft macro if one exists, alter one if it's close but not quite what you need, or build what you want from scratch. The soft-macro library includes a wide selection of functions.

For example, you can select an adder with 8-, 12-, 16-, 24-, or 32-bit capacity. The soft-macro library also includes macros that are equivalent to some MSI TTL circuits. For example, the 8-bit up and down synchronous counter with rip-

Text continued on pg 104

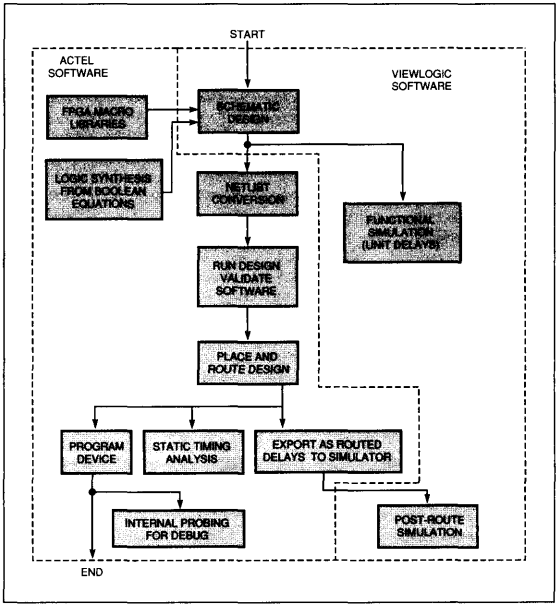


Fig 2—The total time for running netlist conversion, design validation, place and route, and exporting as routed delays took about half an hour for the design. This relatively fast turnaround lets you make quite a few design iterations in one day. The logic synthesis and internal probing for debug were not used on the project.

Pack the digital logic into one FPGA

When I decided to design an FPGA (field-programmable gate array) and write about it, I wanted to use it in a circuit with a minimum of other parts, yet I wanted the circuit to be moderately complex so that it would be a true test of designing with an FPGA. The record and playback circuit I chose packs all the digital logic into the FPGA, and the only other parts it requires are RAM and a few analog ICs (see **Figs 4** to **15** beginning on pg 107).

The top-level schematic for the overall circuit is shown in **Fig A**. The circuit uses the same 12-bit DAC and op amp for successive-approximation conversion during record and for generating the analog output during playback. Because conversion and playback use the same DAC, the gain and offset errors of the DAC and op amp do not add to the system error.

During conversion, the circuit compares the DAC's current output, converted to voltage by a high-speed op amp, with the sampled input voltage. The comparator output drives the successive-approximation logic. Two parallel paths alternately sample and compare the input against the DAC output. The alternating approach saves both the sampling time and the hold-settling time. Each bit decision takes 500 nsec, providing a complete 12-bit conversion every 6 µsec.

The design depends on closely matched offsets in each of the two S/H and comparator paths. You can expect close matching because both comparators are on the same monolithic IC. The same is true for the S/H channels.

Gain accuracy of the circuit depends on the gain accuracy of the S/H circuit and on the comparator's CMRR. The AD684 provides a worst-case gain error of ±5 mV over the ±5V input range. The LT119A used in the circuit has a minimum CMRR of 90 dB, contributing less than a 0.4-mV error over the ±5V input range. Although the LT119A used in the circuit has a minimum CMRR of 90 dB at dc, the CMRR is not specified at the 2-MHz frequency of the design. In fact, depending on high CMRR at frequency is risky, and generally frowned upon by knowledgeable analog designers. In this design I felt the risk was justified by being able to use one DAC for both record and playback.

The digital part of the circuit has four basic states (**Fig 5**): clear memory, armed, triggered, and playback. Playback is the default state when the circuit is reset. The other three states are also ORed together in the circuit to form the recording state (RECD).

To start recording, you depress the momentary arm switch to initiate the clear memory state. The clear memory state starts writing A/D conversions from the successive-approximation conversion into RAM, but disables the trigger until you fill the entire memory with new data, writing zeros to D13 and ones to D14. After you overwrite the entire memory, the state changes to armed, and the circuit continues to record data until the trigger logic is satisfied. Once triggered, the state changes to triggered (TRIGD) and the circuit converts 24,000 more samples, stores them in memory, and returns to the playback state.

You set the trigger level using a rotary encoder to adjust a 10-bit up-and-down counter (**Fig 14**). The logic performs a 10-bit magnitude compare (**Fig 15**) of the successive-approximation converter output with the trigger level to determine when to trigger the circuit.

The trigger-level compare is a full-magnitude compare that tests whether the digitized input signal is greater than or equal to the trigger-level setting or less than or equal to it, depending on the input (TRIG_GE). The 10-bit range provides a trigger-level resolution of 10 mV and gives time for the magnitude-compare results to become valid while the successive approximation is finishing the last two bits.

Control logic (**Fig 10**) also generates the RAM write enable (N_WE), the RAM output enable (N_OE), and the FPGA's output enable (F_OUT). **Fig B** diagrams the basic record and playback timing.

A 12-bit shift register (**Fig 4**) generates the 12 timing states needed for the successive-approximation conversion. These timing signals are also used to control all timing-related logic in the FPGA.

A clock-select circuit lets you select between two clocks. The circuit can play back the data at a much higher rate than it can during recording, because the DAC only changes state once every 12 clock cycles during playback.

Successive-approximation conversion

The A/D conversion starts with sampling and then holding the input. The timing generator uses a 12-bit shift register to control the 12 states of the successive-approximation conversion. I created a macro, called SAR, for the conversion and used one for each bit (**Figs 7** and **8**). The details of the macro are shown in **Fig 3**.

The conversion starts at the beginning of the T1 cycle, DAC data inputs are reset to a low state, except the MSB, which is set high. The correct analog-comparator input is multiplexed to the successive-approximation logic, and near the end of the T1 cycle, the global clock signal (GCLK) clocks in the comparator's output state. At the beginning of cycle T2, the next bit, DAC2, is set high, and driving the MSB remains in the state latched in at the end of T1. The conversion process continues in a similar manner through T11 and the 11th bit. The LSB is slightly different. Near the end of T12, the FPGA will write all 12 bits to the RAM. For this reason the data for the LSB comes straight from the comparator without being clocked into the flip-flop.

When in the playback state, the DAC receives data

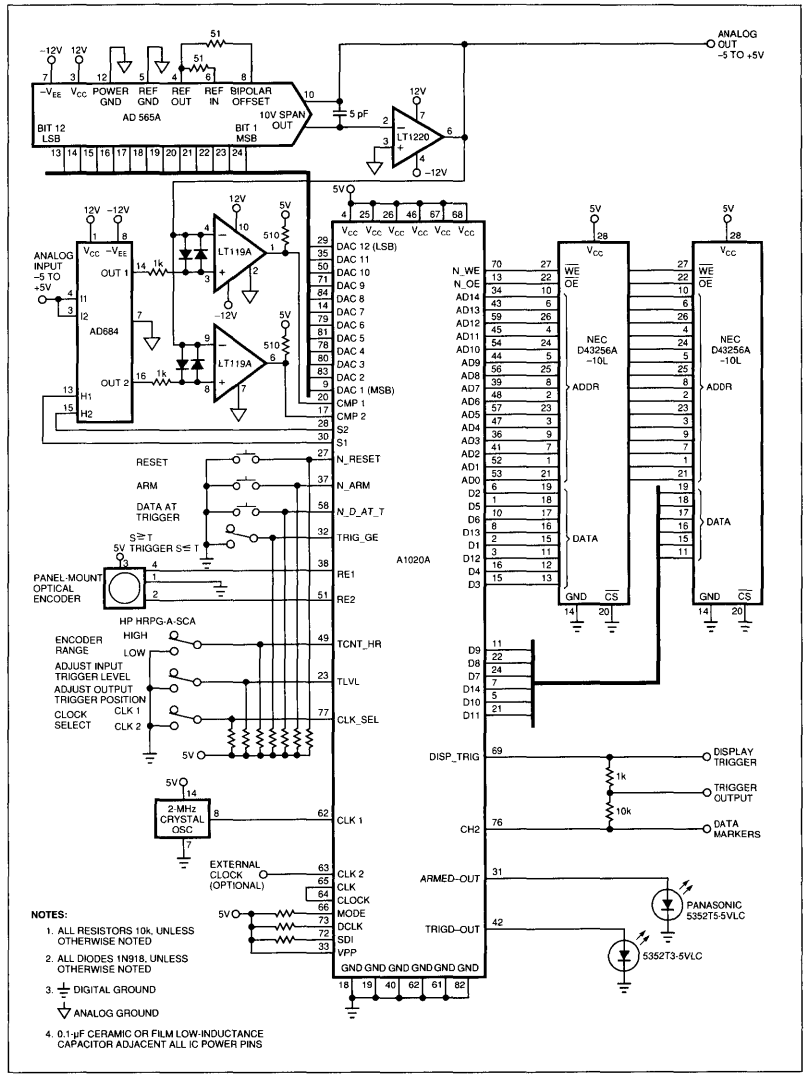


Fig A—The hands-on project was a record playback circuit.

ple carry used in **Fig 12** performs the function of a 74269.

When you need something a little different from the stock parts, the flexibility of a soft macro really shines. Unlike hard macros, which you cannot alter, you can copy and then alter soft macros to perform exactly the function you want. In fact, any time during the design that you want to see what is schematically in the guts of any soft macro, you just select the device and push down into the next level of the hierarchy.

The device labeled CNT 128 in **Fig 13** is a 7-bit version of the TA269 in **Fig 12**. I created CNT 128, my first soft-macro conversion, in approximately 10 minutes. Now that I know how, it should take less than 5 minutes. It really is that simple. All you do is copy and rename the macro's schematic and symbol, then make the modifications to the new schematic and symbol. When you want to use the new function, you call up the symbol and put it on your schematic. You can find other customized soft-macro examples in the schematic, such as 3-bit counters (**Fig 10**) and 7-bit latches (**Fig 12**).

Making a custom macro takes a little longer than merely modifying an existing macro because you need to create the full schematic and symbol. However, it isn't really any more difficult. SAR, used in **Figs 7** and **8**, is a custom macro I created to save a few pages on the schematic. The schematic for the macro is shown in **Fig 3**.

You don't necessarily have to modify a standard soft macro if you don't need all of it. The rule is that

Pack the digital logic into one FPGA *(continued)*

from the RAM and clocks it into the flip-flops at the end of cycle T12. A multiplexer switches the trigger-level setting (TL1-TL10) into the DAC input when the adjust trigger-level signal (TLVL) is asserted.

The 32k-word RAM stores conversion data from the successive-approximation conversion, plus two control signals (D13 and D14) (**Fig 6**). A 15-bit counter generates addressing for the RAM. While in record mode, the address counter is free running. The FPGA continuously writes the A/D results into RAM. When the trigger-level compare condition is satisfied by the incoming signal, the current value of the address counter is latched, the 15-bit up-and-down horizontal trigger-position counter is loaded, and the memory-trigger

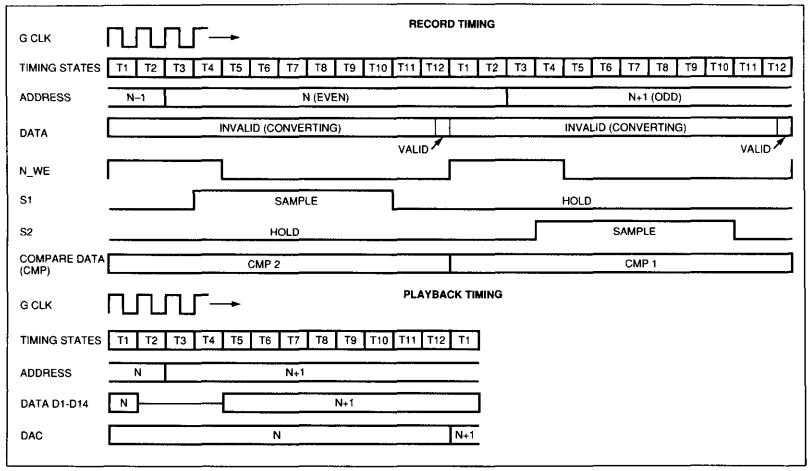


Fig B—At the end of each A/D conversion, the FPGA writes the data to RAM. During playback the FPGA latches data from RAM at the end of T12 to drive the DAC.

EDN·SPECIAL PROJECT

you can't leave any unused inputs—all inputs must be tied to a signal, V_{CC}, or ground. You may leave outputs unused; the software should remove any unnecessary logic associated with the unused outputs. The software will issue a warning whenever an output is unused, giving you a chance to verify that the omission is intentional.

You shouldn't tie unused inputs to V_{CC} or ground if it's possible to eliminate them. The flip-flops in **Fig 10** should be changed to macros without the preset. *CNT4B on* **Fig 6** loads all zeros. A more efficient design would just use a Clear and eliminate the load function on the counter. Even if the change doesn't result in a module savings, unnecessary inputs tied to power and ground restrict routing flexibility, which might affect the overall performance of the circuit.

Fan-out limits are perhaps the most noticeable change from standard TTL design. The software gives you a warning for more than 10 loads, and an error for more than 24. For the special cases of nets you designate as "fast criticality" (I'll discuss criticality in part 2), the fan-out limit drops to six loads. The only exception is the global clock signal. There is only one global clock signal on ACT 1 devices, and it can drive any number of loads.

On the surface, these fan-out limits may not seem too stringent, but you have to remember that macros are just a graphic convenience, no signal buffering occurs unless you put it inside the macro.

For example, the latch-control input of the 8-bit latch shown in **Fig 12** is eight loads, not one. You'll note a buffer in front of it. In fact, you'll see quite a few buffers scattered throughout the pages of the schematic.

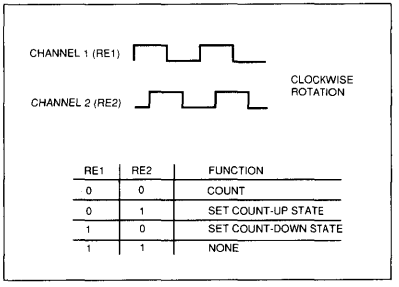
Buffers are easy to add, and the software errors and warnings tell

match condition (N_MEM_TM) is set up to stop acquisition after recording 24,000 more words of data, providing 8k words of pretrigger data. The FPGA writes the display-trigger match (DISP_TM) bit to RAM (D13) to mark the capture-trigger location during playback. N_MEM_TM is also recorded in RAM (D14) to mark the beginning and end of data.

During playback, the circuit compares the address counter with the 15-bit horizontal-trigger-position, up-and-down counter. When the two 15-bit words match the display, trigger-match condition (DISP_TM) is satisfied, and the display-trigger signal (DISP_TRIG) goes high for 12 clock cycles to drive an oscilloscope trigger. Initially, the circuit sets the horizontal-trigger position counter to the capture-trigger address. Subsequently, you may change it with the rotary encoder to trigger the oscilloscope at any address.

The address counter runs continuously during playback except when interrupted by one of two events:

- When the data at trigger signal (N_D_AT_T) goes low, the address counter is disabled the next time it reaches the display-trigger match and stays disabled until N_D_AT_T goes high. With the address counter disabled, the data going to the DAC is frozen so that the DAC continually outputs the voltage at the display trigger address. You can read the dc voltage on the analog output with a voltmeter.
- When D14 from the RAM goes low, indicating the end of the data, the address counter is disabled for 16 periods of 12 clock cycles each, before repeating the data. A high signal on the CH2 output provides a marker to use on an oscilloscope to indicate the beginning and end of data.



RE1	RE2	FUNCTION
0	0	COUNT
0	1	SET COUNT-UP STATE
1	0	SET COUNT-DOWN STATE
1	1	NONE

Fig C—The counter either counts up or down one cycle each time both encoder outputs are low. The count direction is determined by the previous state of the encoder.

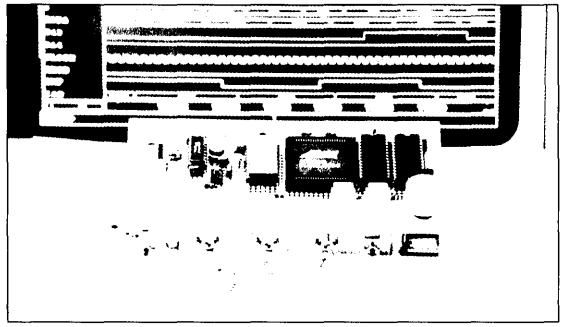
Logic shown in **Fig 9** decodes the quadrature signals (**Fig C**) from the panel-mount, rotary, optical encoder (RE1 and RE2) into count-up and -down signals and count-enable signals. A panel-mount switch lets you select between adjusting the input-trigger level and adjusting the output horizontal-trigger position.

Because the horizontal-trigger position covers a 15-bit range (32k-word address) a high- and low-range select lets you count in increments of 1 or increments of 256 addresses. The rotary encoder provides 120 quadrature cycles per revolution, so using the low range you'd need to turn the knob 273 revolutions to scroll the full address range. Using the high range, you can scroll the whole range in just over one revolution.

you where they are needed. Nonetheless, they are a minor nuisance and one of the few blemishes to what I consider a nearly ideal design environment. Of course, the addition of buffers should remain under the designer's control and not be made automatic because buffering is more than just a cosmetic change to the schematic.

Buffers require a module and add a module delay to the signal (In part 2, I'll discuss timing in detail). Letting module fan-out increase above the warning limit can cause large time delays too. For my particular design, the timing was not too tight, so I just added buffers as needed to eliminate errors and warnings. I probably could have left the warnings and still been okay. If your design has tight timing and you can't afford extra module delays, you can regenerate the signal.

For example, in **Fig 10** you'll find F_OUT and a buffered version F_OUT_A. Had this signal been timing critical, I could have cloned the preceding flip-flop to generate two identical versions of the signal without any additional module delays. The cost in this case would be an extra module because the flip-flop hard macro requires two modules, compared with the single module for the buffer. Also it means doubling the load on the flip-flop's input signals because they'll be driving two flip-flops instead of one.



Getting from a concept to a finished circuit using an FPGA requires learning new software. Even though all the software was new to me, I learned to use it effectively for this project in less than a week.

On the overall schematics (**Figs 4** to **15**) I've only labeled nets where I needed to for design reasons, with very few exceptions. One exception is IA0 in **Fig 11**. It's labeled for simulation reasons I'll discuss in part 2. In future designs, however, I plan to label every net and every module. Although labeling takes time, it pays off when simulating, using the static timing analysis software, and reading error reports. I have to note that several people recommended labeling everything, and I ignored the advice. In the end, I didn't save any time by omitting the labels. You can take my advice or learn the way I did.

After reading this far you may have come to the conclusion that designing an FPGA is not much different from designing with SSI and MSI ICs. That's my conclusion too. I spent my time during schematic design battling with system design issues and how to improve the design, not fighting with tools or wondering if the clever use of a different MSI device would make a cleaner design. In part II, I'll show you some of the bugs I caught in simulation and two that I didn't catch until I tested the circuit. I'll also present you with the chronological account of the project so you can see how much time I spent in each step. EDN

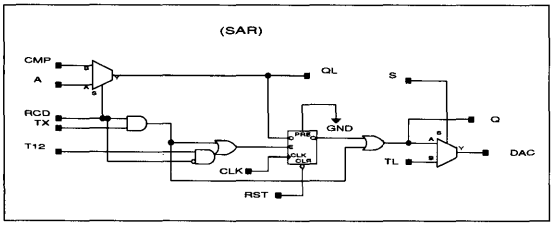


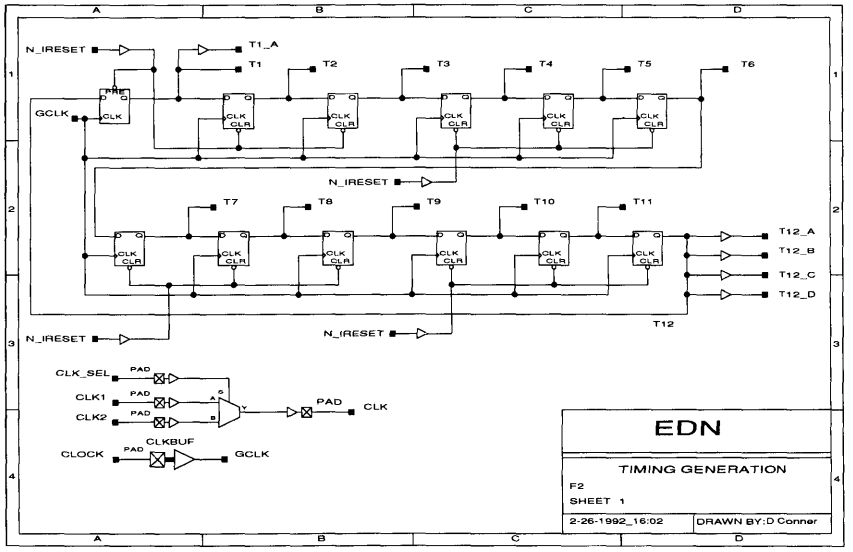
Fig 3—Creating custom macros is very simple and simplifies the schematic of the overall design. SAR is used in Figs 7 and 8 of the FPGA schematic. If you need to make changes to the design later, you need to change only the macro.

Reference

1. Conner, Doug, "High-Density PLDs," *EDN*, January 2, 1992, pg 76.

Doug Conner is a technical editor for EDN based in California. You can reach him at (805) 461-9669.

EDN·SPECIAL PROJECT



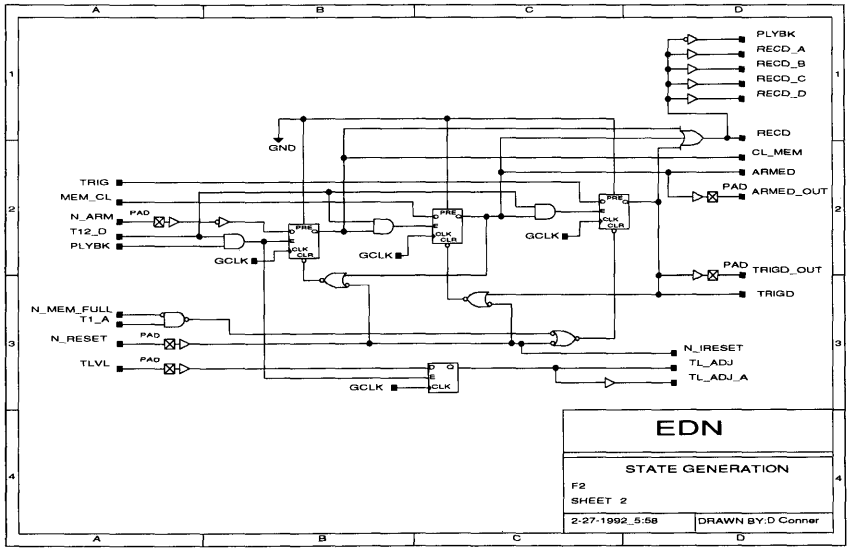


Fig 4 (top)—The schematic shows the logic for generating the 12 timing states used for all record and playback operations; Fig 5 (bottom) shows the logic for generating the playback state and the 3 record states: clear memory, armed, and triggered.

6

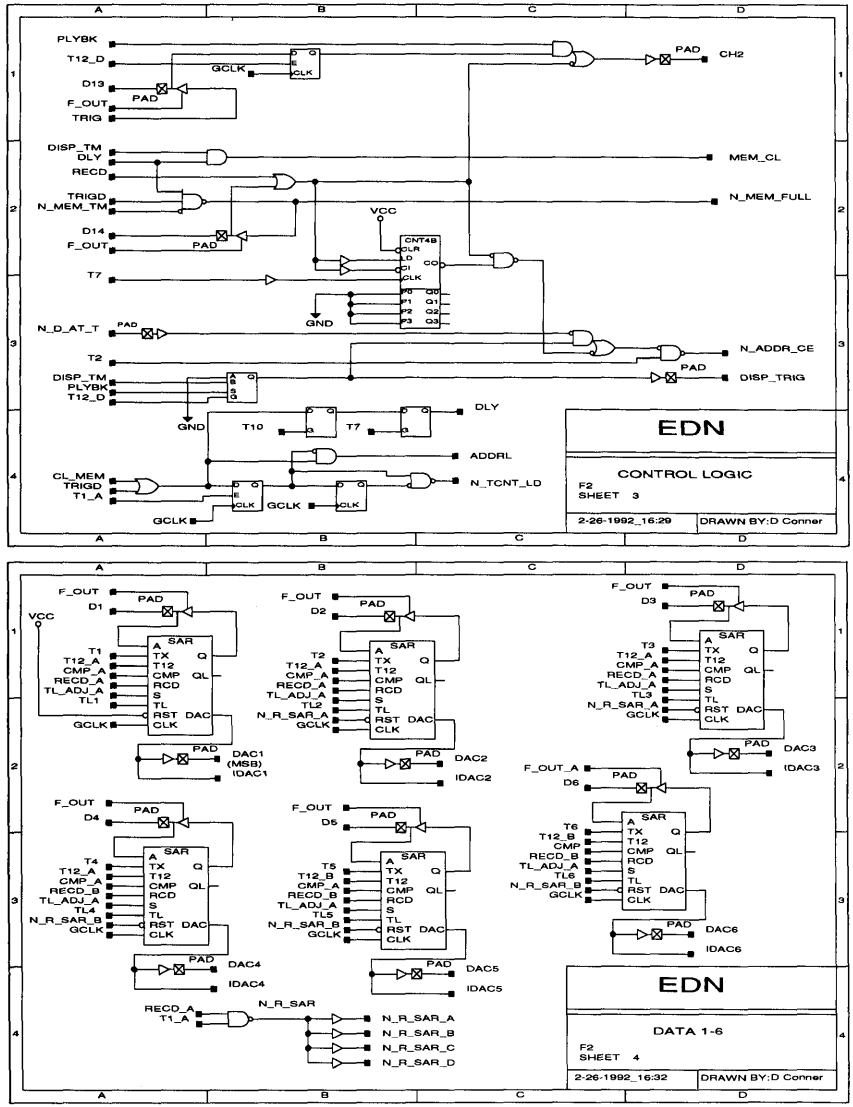


Fig 6 (top)—Logic for miscellaneous control functions is illustrated in this diagram; Fig 7 (bottom) shows the logic for the lower six data bits (see Fig 3 for SAR macro schematic).

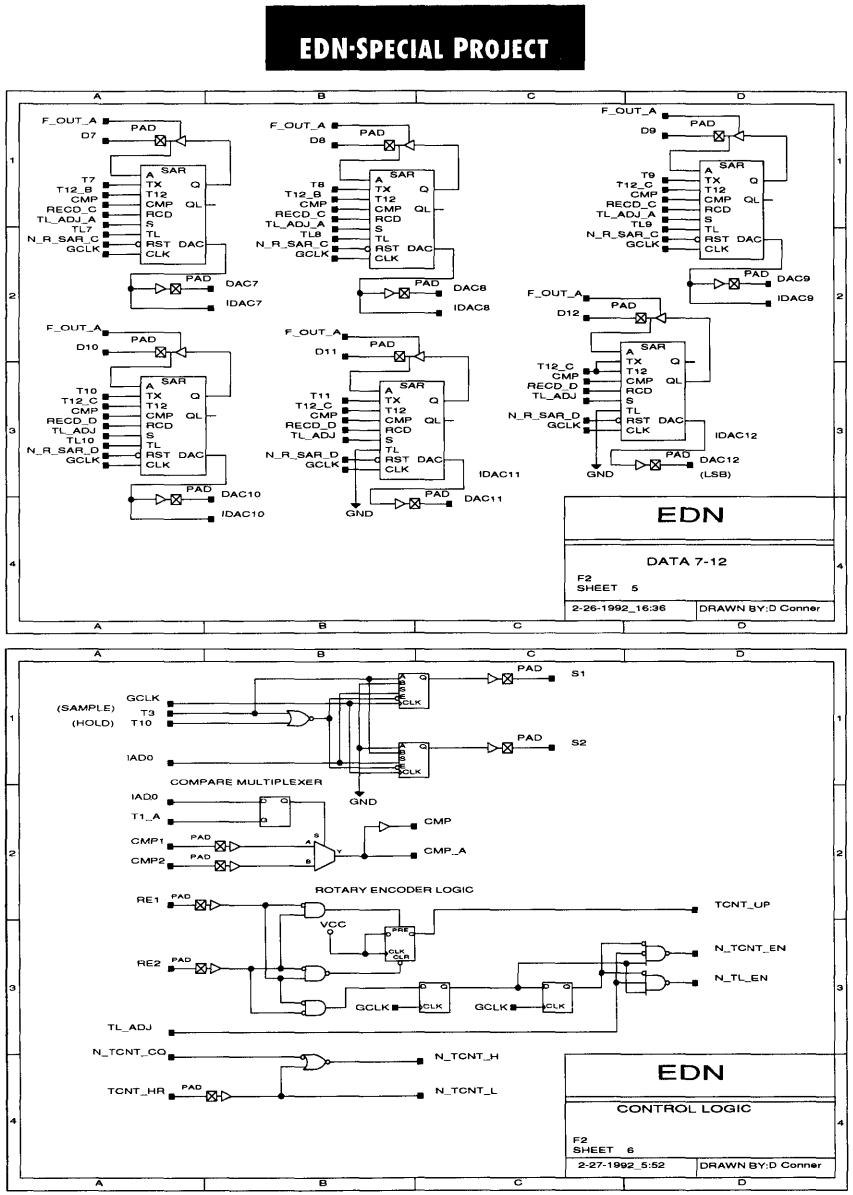
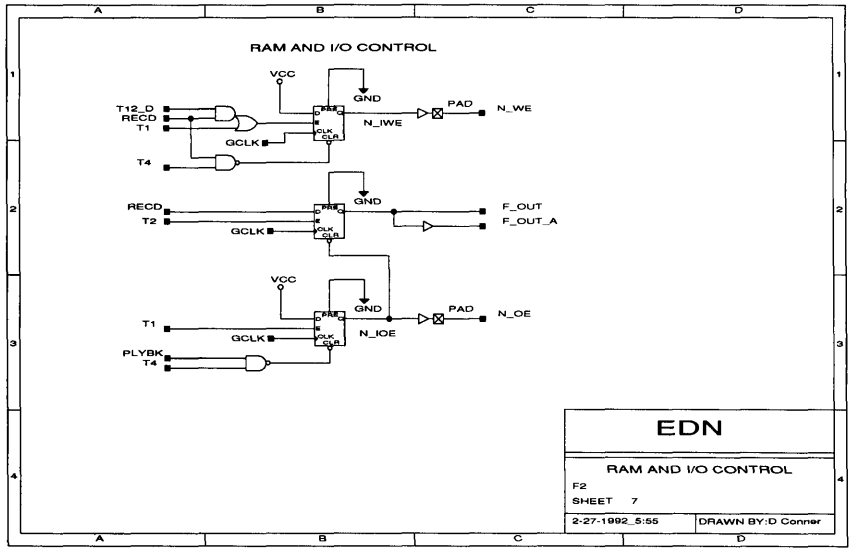


Fig 8 (top)—This schematic details the logic for the upper six data bits (see Fig 3 for SAR macro schematic); in Fig 9 (bottom), you can see the control logic for the S/H circuit, compare multiplexer, and rotary-encoder decode logic.



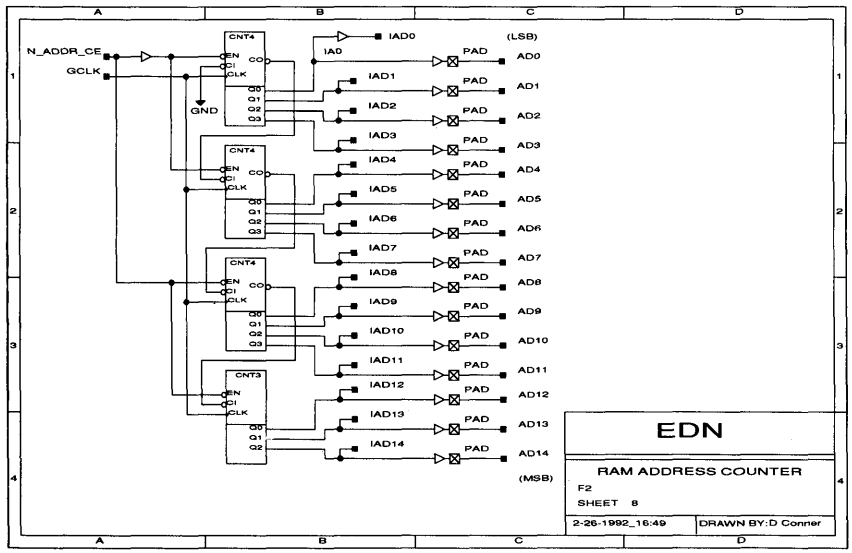
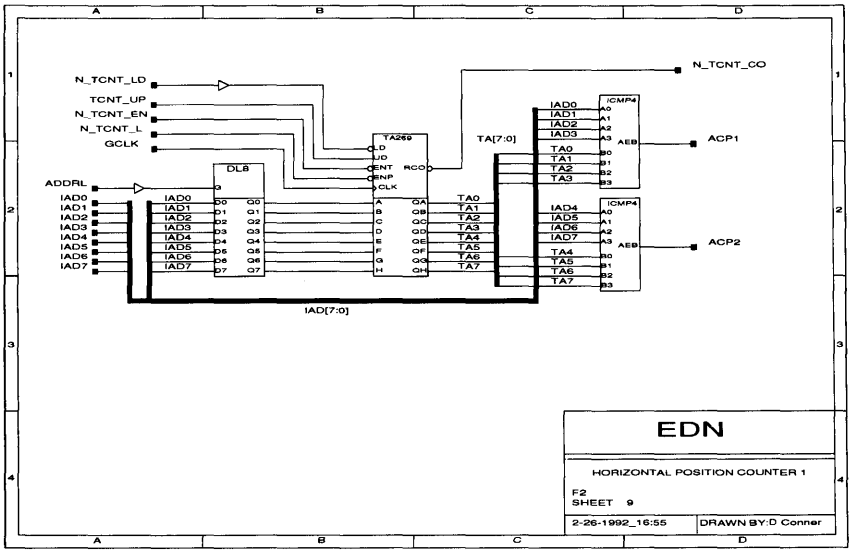


Fig 10 (top)—The schematic illustrates the write-enable and output-enable logic for the RAM and the output control for bidirectional data lines; Fig 11 (bottom) highlights the 15-bit counter for RAM address lines.

EDN·SPECIAL PROJECT



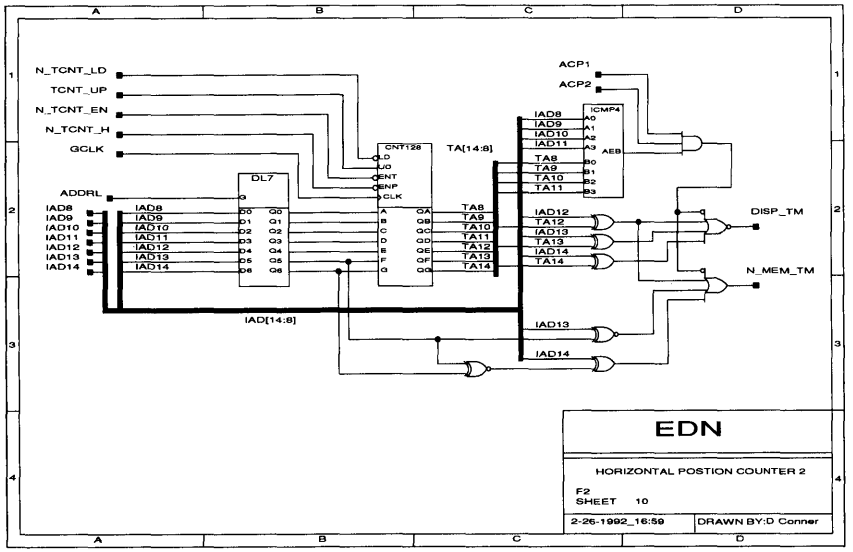
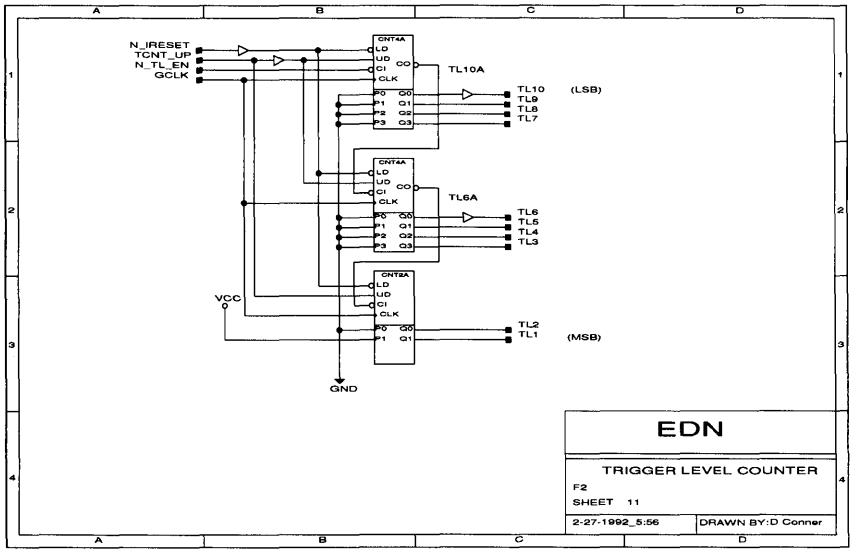


Fig 12 (top)—The lower eight bits for horizontal-output position control are illustrated here; Fig 13 (bottom) shows the upper seven bits for horizontal-output position control.

6



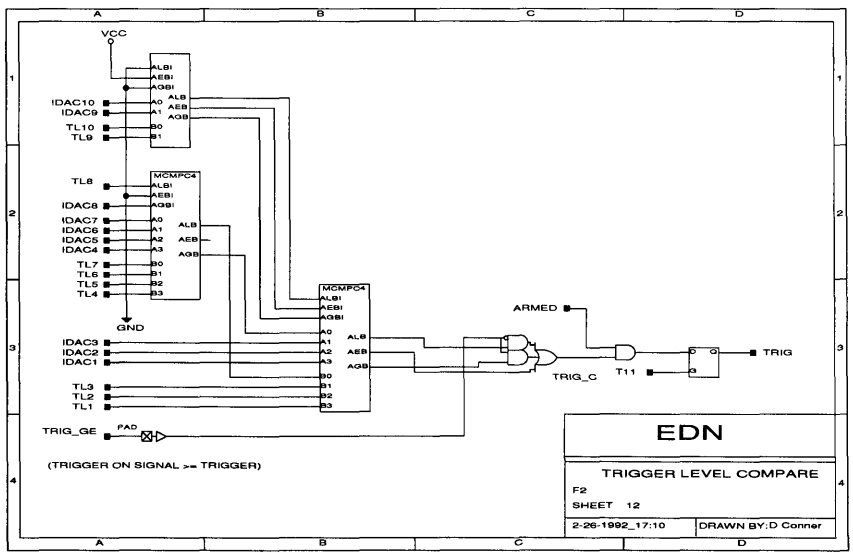
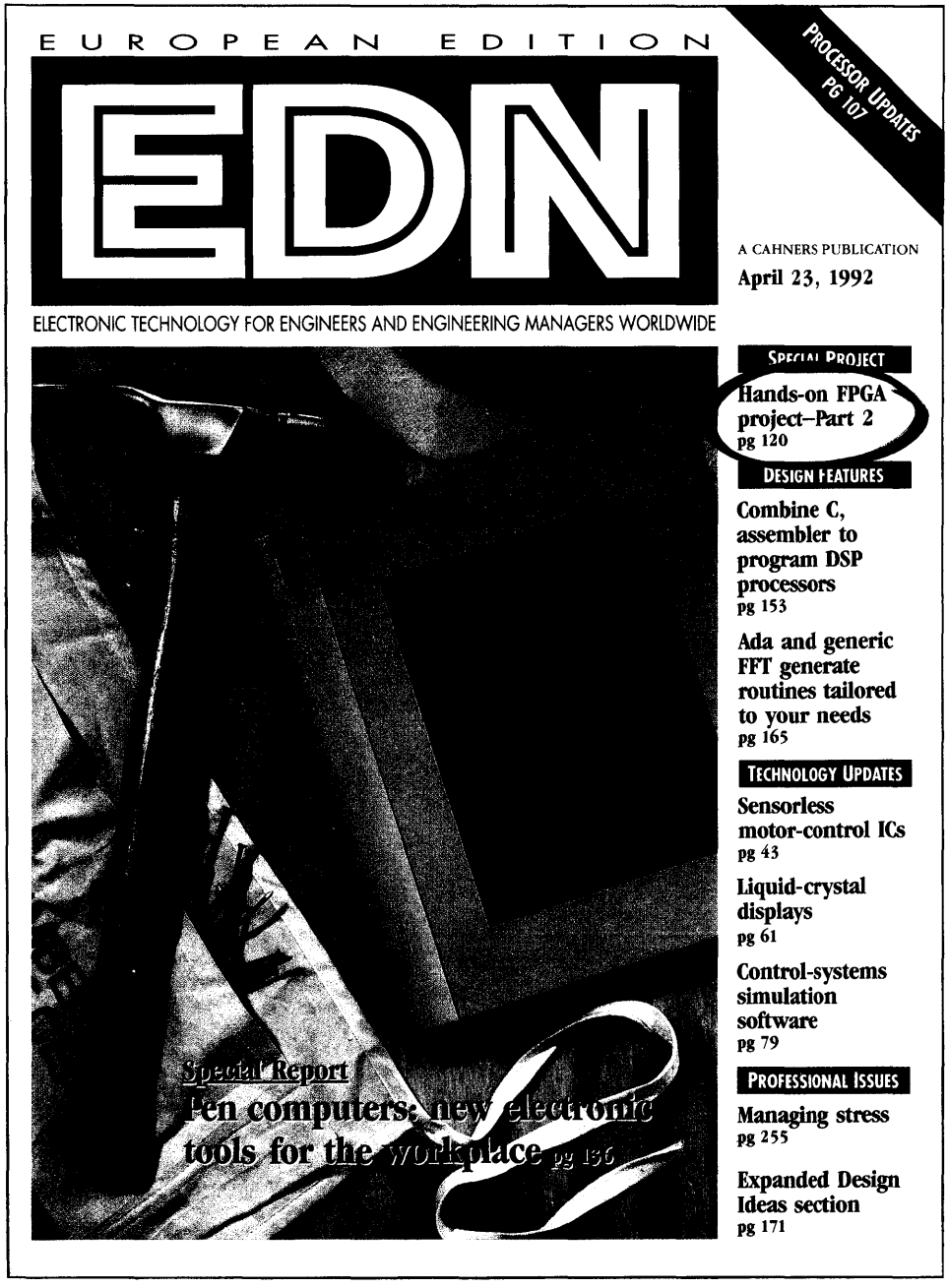


Fig 14 (top)—This schematic shows the 10-bit counter for setting the input trigger level; Fig 15 (bottom) illustrates the 10-bit magnitude-compare circuit for detecting the input-trigger event.

Migrating to FPGAs: Any Designer Can Do It



6

EDN·SPECIAL PROJECT

Migrating to FPGAs: *any designer can do it*

Part 1 of this 2-part hands-on design project (in the April 9 EDN) discussed the overall circuit and the schematic entry of this FPGA (field-programmable gate array) design project. Part 2 concentrates on the steps from simulation to the final functioning circuit. The project took 29 working days from start to finish.

DOUG CONNER, Technical Editor

In the April 9 issue I discussed the first part of my journey into FPGA design. The schematic part of the design wasn't much different from ordinary SSI or MSI TTL design. Because logic simulation is seldom used in SSI or MSI TTL design, it was a new experience for me and was the greatest worry when I started the project. If you haven't been using logic simulation, you'll find this phase of verifying your FPGA design a big change.

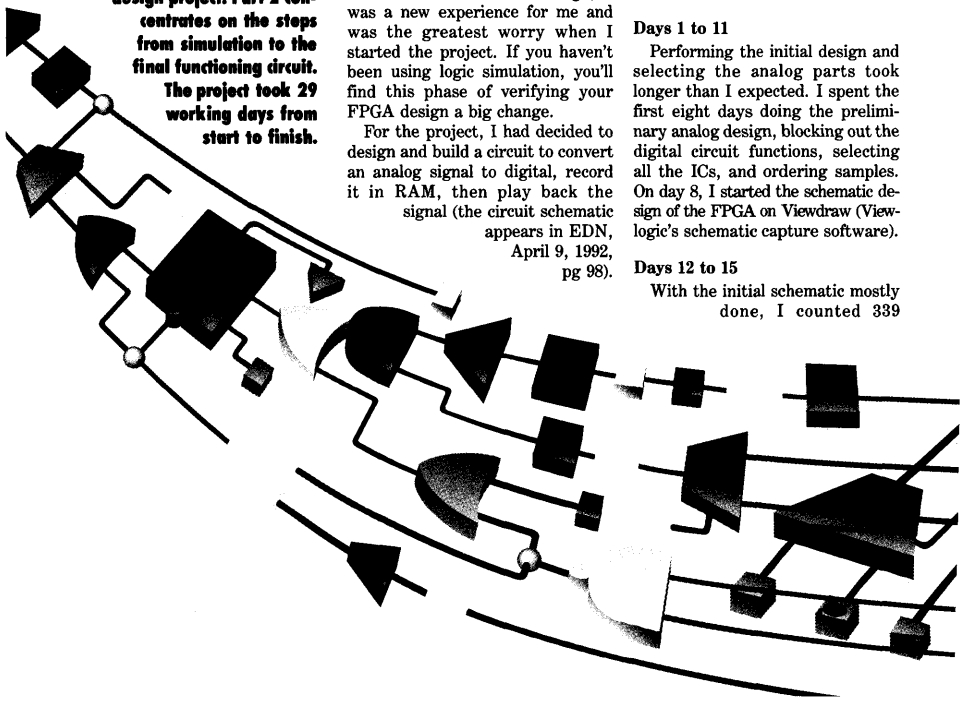
For the project, I had decided to design and build a circuit to convert an analog signal to digital, record it in RAM, then play back the signal (the circuit schematic appears in EDN, April 9, 1992, pg 98). All digital functions would be performed in the FPGA. I wanted to keep the analog portion of the circuit simple, yet be as fast as possible. I began the project by blocking out the design and figuring out what the overall circuit should do.

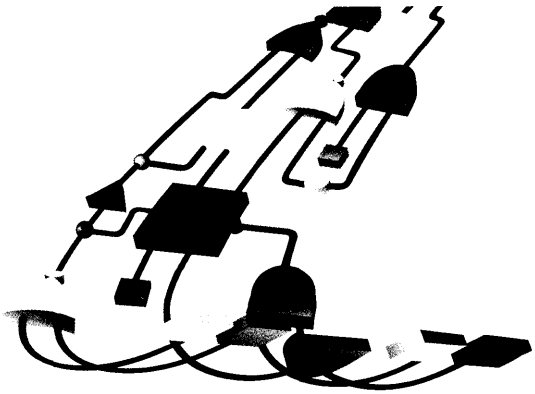
Days 1 to 11

Performing the initial design and selecting the analog parts took longer than I expected. I spent the first eight days doing the preliminary analog design, blocking out the digital circuit functions, selecting all the ICs, and ordering samples. On day 8, I started the schematic design of the FPGA on Viewdraw (Viewlogic's schematic capture software).

Days 12 to 15

With the initial schematic mostly done, I counted 339





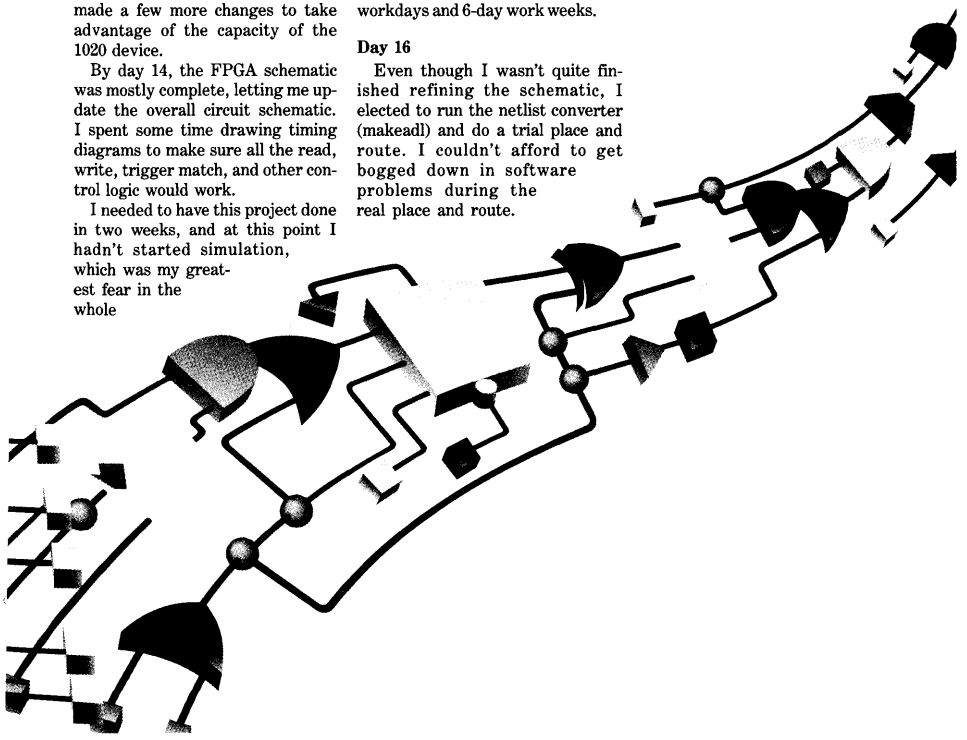
used modules. That's too big for the 295 modules on an Actel 1010 device and only 60% of the 547 modules on a 1020 device. I decided to bring triggering inside the FPGA and made a few more changes to take advantage of the capacity of the 1020 device.

By day 14, the FPGA schematic was mostly complete, letting me update the overall circuit schematic. I spent some time drawing timing diagrams to make sure all the read, write, trigger match, and other control logic would work.

I needed to have this project done in two weeks, and at this point I hadn't started simulation, which was my greatest fear in the whole project. The schematic-entry tools were working fine, but I kept finding design problems that had to be solved or improvements I considered necessary. I initiated 12-hour workdays and 6-day work weeks.

Day 16

Even though I wasn't quite finished refining the schematic, I elected to run the netlist converter (makeadl) and do a trial place and route. I couldn't afford to get bogged down in software problems during the real place and route.



I wanted to find them immediately and get them out of the way.

The netlist conversion is a single command and takes about five minutes to run. The Validator software checked the design and the computer returned the errors and warnings: I had one incorrectly named net, four fan-out errors, and ten fan-out warnings. I corrected the errors and went on to try an auto-pin placement.

The autopin software under the Pin Edit menu wouldn't run because it needed a file named *design*.pin, which I hadn't created yet. I put in a call to Actel, but before they returned my call, I tried some experiments. Under the configure selection on the menu I found an automatic I/O assignment selection. Reviewing the documentation, I learned that this appears to be the right menu selection to use to do the automatic I/O pin assignment. I tried it, and it worked. After you run the automatic I/O pin assignment once, the *design*.pin file is created. After that I could edit the pin assignments if I wanted to.

I left the automatic I/O pin assignments because they should let the automatic place-and-route software do its best job. The first time I ran the place-and-route software, it produced a fully placed-and-routed design in 16 minutes. I had used 486 modules and 57 I/O pins.

Day 17

Day 17 marked the beginning of simulation, or more accurately, the preparations. I was a complete novice at simulation, so I alternated between reading the Viewsim (Viewlogic's digital simulator) reference manual and setting up the files to provide stimulus for the circuit. I ran a few simulations to see if the file would do what I wanted.

I found learning to use the simulator similar to learning a relatively simple computer language. I used fewer than 20 of the roughly 60 commands available.

The Viewsim simulation environment (**Fig 1**) lets you work in three different modes. A text mode lets you input data and commands and output results. A graphical-waveform mode gives a logic-analyzer-type display showing the states of signals. You can bus signals together, such as data and address lines, so you can view many signals at once. You can also create stimulus in the graphical-waveform mode, although I didn't use that method on the project. The third method is to drop down into the schematic and see the actual data values on every node. You can even push down through the levels of hierarchy into macros so you can see what is happening inside a counter or the SAR macro (a soft macro I created).

I created my input commands and data in text files, viewed results with the waveform graphical display and occasionally dropped down into the schematic display to verify exactly where a problem identified in simulation was occurring. I found this method similar to debugging and testing real hardware.

I did not use any expect-data files to make automatic comparisons with simulation results. I visually examined the waveform graphical displays to get the most information. Although I reduced the number of signals displayed in the photographs to make them easier to read, in practice I crammed as many signals as I could onto the graphical waveform display for maximum information.

For those who haven't used digital simulation before, I can offer some comments on my experience. You have to set the initial conditions for all inputs (high, low, high-impedance, or unknown) for every input or I/O pin on the FPGA. Setting the inputs for control lines and changing them during simulation is straightforward. For bidirectional data lines such as the ones connecting to the RAM in this design, you need to drive them with

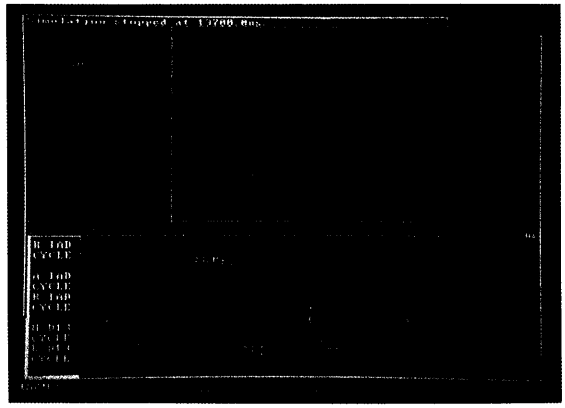


Fig 1—The FPGA software lets you observe simulation results in three different formats: text, graphical waveform, and on the schematic.

Illustration by Gregg Dinderman

EDN-SPECIAL PROJECT

the correct data during RAM reads, and release them during RAM writes. Even after you've set up the proper conditions on all external inputs and I/O lines, you're still not done. You may have to initialize internal conditions. Some of the initializing may be covered by reset logic you've designed into your circuit, and some of it may not.

For example, the address counter (**Fig 11** in Part 1, pg 98) is a free-running counter. When the simulation starts (and in the real hardware) the address is in an unknown state. After each count cycle, the hardware progresses from some initial state n to $n+1$, then $n+2$, and so on.

When you start simulation, the counter is in an unknown state and will stay there. To get useful simulation results, you have to force the counter into a known state, and then it will begin counting. I forced the counter's outputs to an initial state, then released the outputs, and the counter ran properly. You should label all counter output nets so that you can force them to known states during simulation.

Simulation is slow compared with real hardware, so you need to keep the clock-cycle count low if you want to make many simulation runs. Most of my simulation runs were a few hundred clock cycles or fewer. I don't believe a simulation ever took more than five minutes to run; a typical simulation run took about two minutes. For reference, the design is 1514 gates by standard gate-array counting measures. I ran the software on a 33-MHz 486 PC with 8 Mbytes of RAM.

To run the 15-bit counter through all 32,768 cycles would have taken nearly 400,000 clock cycles. Rather than having the simulator run for a day and generate reams of data, I forced counters to states near where some event should happen (or shouldn't happen, but might), released the counter, and let it count through the cycles I wanted to see. Then I forced the counter to the next event of interest (**Figs 2** and **3**).

My approach to simulating the design was twofold. First I tried to identify every part of the circuit where I had concerns, list them, and then make simulation test cases to verify that the function per-

Table 1—FPGA and other products used on this project

Manufacturer	Product	Price	Description
Analog Devices	AD565AJD	$17.60 (100)	12-bit DAC.
	AD684JQ	$23.50 (100)	4-channel, 1-μsec S/H amplifier.
Linear Technology Corp	LT1220CN8	$3.85 (100)	Very-high-speed op amp.
	LT119AH	$6.35 (100)	Dual comparator (commercial version LT319A is $1.95 (100)).

Editor's analysis

All things considered, my opinion based on this project is that designing with an FPGA is actually easier than designing with SSI and MSI logic. You can use the same schematic design approach you are familiar with. You don't have to use simulation to design with FPGAs, but I'd highly recommend it. My first attempt at simulation wasn't perfect, but it got me close. Finding the last few bugs in hardware and correcting them was not a problem. Correcting the bugs did not even necessitate pc-board changes.

I no longer wonder what applications FPGAs are useful for, but rather what applications still make sense for small- and medium-scale integration TTL and CMOS logic. High-current bus drivers are one; extremely simple logic circuits are another. Some high-speed logic will favor SSI and MSI devices, but a number of FPGAs are available with the capability of loading and operating 16-bit counters in the 50- to 100-MHz speed range.

Price is perhaps the biggest barrier to FPGAs' taking over the low- to medium-volume logic market. At $36.25 (100) for the device I used, it's competitive with SSI and MSI devices, especially if you factor in the cost of pc-board space and the flexibility to make design changes easily. As you move to higher-density and higher-speed FPGAs, you'll pay a premium over SSI and MSI parts.

formed as expected. These cases include counters, magnitude compares, optical-encoder signal decode, and others. Second, I simulated the FPGA as a whole, performing entire sequences of clearing, arming, recording, and playing back the data. Other simulation sequences tested the trigger-level and trigger-position-adjust operations.

My concerns on this design were mostly functional. Counters have to count, so I simulated all major transitions. In fact, it's a good idea to test all macros you create or alter separately. And because macros contain relatively few gates, they simulate quickly. Even though I tested the counter macros that I modified, I still tested the entire counter in the full schematic.

Because I had already run a place and route on the design, I could export the as-routed delays to the simulator and have a more accurate simulation. But I didn't want to use the as-routed delays at that time because I was still fixing a few bugs, and the unit-delay simulation provided a faster turnaround. I could change a schematic, recompile the simulation, and simulate in about five minutes. The turnaround time with as-routed delays requires a netlist transfer, design validation, place and route, and exporting the delays to simulation. The total time for doing all those things is about half an hour.

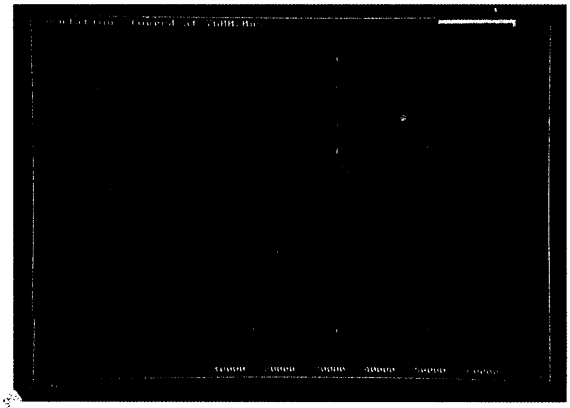


Fig 2—This sequence takes the design through reset, clearing memory, arming, triggering, completing data acquisition, playback with capture trigger marker, and end of data marker. To see more details, you need to zoom in as shown in Figs 3 and 4.

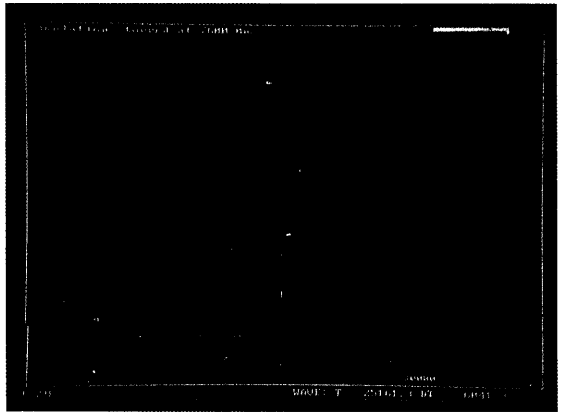


Fig 3—I've zoomed in to show a portion of the display in Fig 2 where the Memory Cleared (MEM_CL) signal indicates the memory has been cleared, and the FPGA changes its state to Armed. A short time later I force the trigger signal (TRIG) high, which causes the FPGA to change its state to Triggered (TRIGD).

Day 18

I spent lots of time on day 17 reading the simulation reference manual to understand the simulation commands and trying to figure out how to test my circuit with the available commands. By day 18 I was writing command files containing series of commands that initialize the FPGA and start taking it through its paces.

It seems incredible, but in two days I was able to learn enough to make simulation a useful tool. One of the attractive aspects of simulating an FPGA is that I didn't have to worry about simulation models. Anything I can design into the FPGA is covered by the simulation library. For this design, all the digi-

EDN·SPECIAL PROJECT

tal logic was in the FPGA, so I didn't see any need for board-level simulation.

Here are some of the bugs I found in one day (I corrected the bugs on the schematics in Part 1):

- D-14 needed to be gated so it only inhibited ADDR_CE (address count enable) on playback, not during record.
- &DISP_TM needed more delay so it wouldn't reset immediately when initiating CL_MEM. I had put in a delay, but simulation indicated it needed to be longer than I thought.
- The circuit was changing the compare multiplexer from compare 1 to compare 2 when the address changed during T2, instead of at the beginning of T1 (when it should happen).

I had been dreading simulation. I assumed I'd be bogged down in learning to use difficult software. Instead, I find myself a simulation convert—learning the software is reasonably easy. The bugs jump right out once you start exercising the circuit functions. It's just as much fun as debugging hardware, and you don't have to put probes on difficult-to-reach pins.

Day 19

I made a few changes to the successive-approximation logic and decided to create a soft macro (called SAR). I ran more simulation, and then I ran the place-and-route software and generated as-routed delay information for simulation. I spent a couple of hours making a final check of what remained to be simulated before I was ready to call simulation done.

The list of what was left to simulate seemed long, but because I was more familiar with the simulation commands and had simulation command files to perform most of the major functions on the FPGA, it went faster. I put simulations together quickly by calling initialization routines I'd already written, adding some commands, or modifying an existing command file.

I made some changes to the schematic and compressed the design onto fewer sheets. I then wanted to delete the excess pages, but couldn't find a utility for deleting schematic sheets. Instead, I deleted them from DOS.

A few hours later while simulating the design, things weren't quite right. I traced the problem to a 2-input multiplexer with the correct data going in, the correct data on the select pin, and unknown data on the output. I expected to find another output driving the net, but a double-check of the schematic indicated that that wasn't the problem.

This was my first, and only, serious problem with the simulation tools. It lasted for about two hours. Finally, I made the connection that perhaps the schematic sheets I deleted were not completely gone. It turns out that when you save a schematic sheet, the software creates a wirelist description file in the WIR directory. I deleted the files in the WIR directory for the schematic sheets I had deleted earlier, recompiled the simulation file, and was back on track. Of course, the simulation tools weren't really at fault, but I never did find anything in the documentation about how to delete schematic sheets properly.

I continued on to simulate the as-routed delays, exporting the delay information after a place and route into the simulator. The relatively short place-and-route time (approximately 15 minutes) is really useful when you make a design change and want to get back into a simulation with accurate timing.

Day 20

On day 20, my schedule called for having the design done and a functioning prototype board in one week, leaving me a week to tie up any loose ends before the article was due. However, I still hadn't got the design to the point where I wanted to freeze the FPGA pinouts. After that, I needed to lay out the prototype board, build the

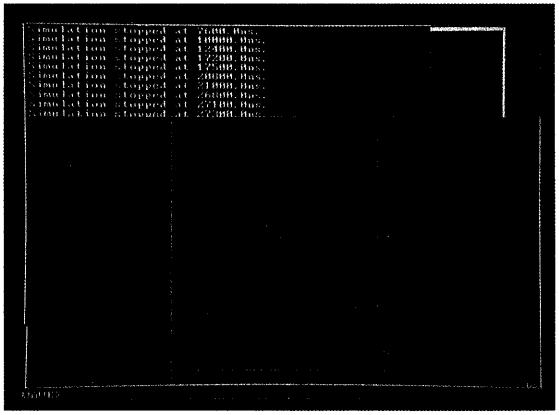


Fig 4—Zooming in still further on Fig 2, you can see the data written to RAM when the negative Write-Enable line (N_WE) goes high. Here I'm looking at the hold time on the Data bus.

board, program the FPGA, and get the circuit working.

Later that day, while running through my simulation test cases, I discovered a serious error: The FPGA would never write the memory-full signal to D14 because of a setup timing error. The problem was very easy to fix, but it made me wonder how many other errors remained.

I made a few minor changes and tried to run a unit-delay simulation, which didn't work. All the timing is in even clock cycles, but unit delay simulation should give each logic module a 1-nsec delay. I'm reasonably sure what the problem is, but don't know how to fix it.

Once you run the place and route software and export as-routed delays, the simulation software shifts from unit delay to zero delay per interconnect. It then looks up the real delay for each interconnect in a file. Because I've changed the schematic, the software apparently knows it can no longer use the as-routed delay file, but it isn't resetting the simulation to unit delays. I could have called Actel and found out how to fix the problem, but I elected just to run the place-and-route software, export the delays, and get on with simulation. I was at the point where I needed the timing accuracy anyway. (After I finished the project, I called Actel and got the answer to my problem. After you export the as-routed time delays, the software creates a file named *design*.VAR in your work-view directory. You need to delete that file and run *export wirelist* in the schematic window to return to the unit-delay simulation.)

My particular design had very few cases of critical timing because I was running at low clock rates. My requirement was 2 MHz, and 10 MHz was my goal for the digital. The analog part of the design could play back at more than 2 MHz, but the record mode probably couldn't go beyond 2 MHz and still settle properly during conversion. With this extremely loose timing, I didn't have to make any changes for speed in the design; I was more concerned about saving gates.

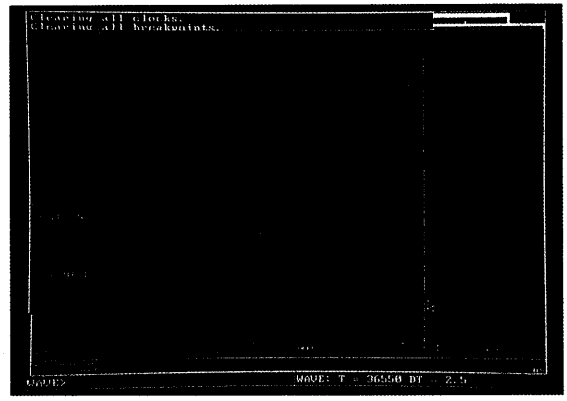
When designing faster circuits, you need to be sure critical networks don't end up with long interconnect delays. These long delays happen when two interconnecting modules are spaced far apart on the FPGA. The automatic place and route software attempts to place the design into the modules with a minimum of long interconnects. For this FPGA, long-vertical tracks are the worst, and long-horizontal tracks are the next worst. My design ended up having 16 long-vertical and 54 long-horizontal tracks.

The way you protect critical networks from long delays is with a network-criticality assignment. You can assign networks a criticality value of fast, medium, or uncritical. Fast or medium criticality keeps nets off long interconnects. You can designate as many as 5% of the nets fast and 15% medium. Assigning nets uncritical when they can tolerate long delays lets the routing software connect them with long tracks when necessary.

In my design, I designated fast criticality for the write-enable circuit because the tightest timing is at the end of a write cycle to the RAM. The RAM requires a zero hold time on the data when write-enable goes inactive. Initially, the design had a <10-nsec hold time in simulation, which should be okay. The fast criticality assignment widens the margin. Eventually, I added extra gating just to be sure my timing margin stayed on the proper side of zero.

Day 21

By day 21 I was still simulating. But since the circuit was in reasonably good shape, I started to push



Here I pushed the clock up to see where the circuit fails. At 20 MHz the signal enabling the upper 7-bit counter for the horizontal trigger position (N_TCNT_H) has only 2.5-nsec setup time before the rising edge of the clock, insufficient for the counter. You can see the Trigger Address bus (TA) does not make the transition from 0FFF to 1000 but goes to 0F00.

EDN·SPECIAL PROJECT

the speed. I had been working with a 2.5-MHz clock because that is all the speed I needed to have. I pushed the circuit up to 10 MHz, and then to 20 MHz.

At higher clock speeds, the simulator often puts out a warning that the circuit is not yet stable. What this means is that the results of the last data or clock transition are still propagating through the circuit.

An example of a relatively long path where I would get a circuit-not-yet-stable warning is in the trigger-level compare circuit. The 4-bit magnitude-compare soft macro was listed in the data book as having four module delays. The 2-bit compare had three module delays. The last bit to change in the compare will always be IDAC10, so this part of the circuit showed seven module delays through the comparator, plus three more modules to the TRIG signal for a total of ten module delays. Actually, the AND gate was combined with the latch when I compiled the design, so there were nine module delays.

Typical module delays, including interconnect, range from <6 nsec to approximately 11 nsec for a fan-out of eight. Long tracks can push the delay to approximately 35 nsec. Using 10 nsec as a round number, the delay from a magnitude-compare input change to the trigger-signal output was 90 nsec. Because

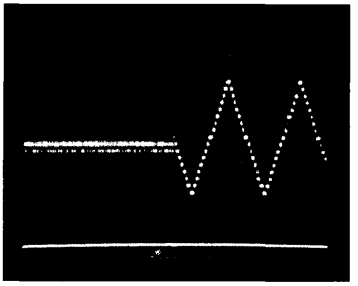


Fig 5—The waveform illustrates the record and playback circuit capturing a signal that changes from ground to a 1-kHz, 40-mV p-p triangle wave. The waveform was photographed during playback at twice the record speed using 0.2 msec/div and 10 mV/div. You can clearly see the 2.5-mV quantization levels. The pulse on the lower trace marks the capture-trigger location.

I hadn't specified any of the nets in the trigger-level compare circuit as being critical, long tracks could show up anywhere, even inside the soft macros. Because I had two clock cycles of 500 nsec each before I needed valid data for the nominal design condition, I wasn't concerned. Even if every module had a long track connection, the delay would be about 315 nsec. Of course, the as-routed simulation or the static timer would show just how long it takes to get through a given path.

When I simulated the circuit at 20 MHz and stopped it to view the data one 50-nsec clock cycle after a data bit had changed going into the trigger-level compare circuit, I would get a circuit-not-yet-stable warning. The simulator was still giving me the correct results at that instant, but it was also warning me that even if all clocks and external inputs freeze in their present state, some outputs have yet to reach their final state.

The simulation indicated that my FPGA would work at 12.5 MHz. If I needed the circuit to work at 20 MHz, I'd need to go back and start assigning fast and medium criticality to the appropriate nets or change the design.

With simulation complete, I spent the second half of the day laying out the circuit for the prototype board.

Days 22 to 24

Finally I got to the point where I was ready to freeze the pinouts. I had left them floating so that the place-and-route tools would have maximum flexibility to place and

The analog-circuit performance

The dc offset of the circuit from analog input to analog output is −5 to −7.5 mV. The dc gain error is within ±1 bit (2.5 mV) over the ±5V range, although the component specifications indicate you shouldn't expect better than ±2 bits. For better dc accuracy, you could trim the offset with an op amp on the input. Transition noise is approximately ±0.75 bit. I haven't been able to characterize the ac accuracy of the system to 12-bit accuracy.

The circuit is useful for examining signals to about 20 kHz with its 167-kHz sample rate. Filtering to avoid aliasing is a necessity if the circuit is used to examine signals with frequencies beyond 80 kHz. The 32k-word RAM provides 0.197 sec of storage with a 2-MHz clock speed. By slowing the clock speed to 12 kHz the circuit can sample at 1 kHz for more than 32 secs. During playback, you can increase the clock speed to 12 MHz for a flicker-free display on an analog oscilloscope.

EDN·SPECIAL PROJECT

route the design efficiently with a minimum of long tracks.

I had no more time for improvements. The only changes I could allow now were to fix bugs if I found them. As I transferred the FPGA pinout list to the full-circuit schematic, I discovered I hadn't brought out two signals—ARMED and TRIGD (triggered). I added the output pads, reran the place-and-route software, and got the signals. I used up 92% of the logic modules and 63 of the 69 I/O pins available.

On day 24, I assembled the prototype circuit. The prototype board was complete, except for an empty socket where the FPGA belongs. I created the fuse file for programming a chip, which took only a few minutes. Normally, you'd have the unit that you use to program the chip connected to your computer. When you're ready to program a device, you just put it in the programmer and run the software. I didn't have a device programmer, so I went to Actel to program my first chip. I didn't measure the time required to program a part, but I estimate it takes about 10 minutes.

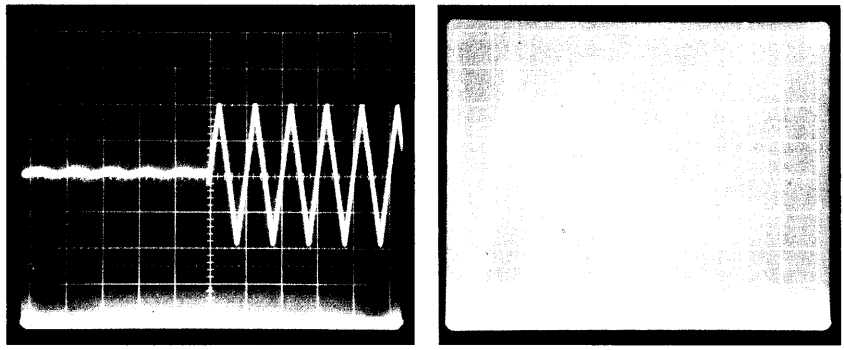
I plugged the part in the socket and powered up the prototype. I brought my power supplies and function generator with me to Actel and borrowed a scope. The circuit showed signs of life, but I couldn't get a good trigger from the display-trigger signal (DISP_TRIG). Gradually I came to the conclusion that the problem was the scope and not DSIP_TRIG. I asked for another scope and found that the circuit could perform all the basic operations. I don't know whether I was more surprised by my first FPGA design working, or that I handwired the prototype correctly.

I went back to my office for further testing, where I discovered a bug. As I turned the optical rotary encoder to adjust horizontal trigger position or trigger level, it occasionally jumped, rather than scrolling smoothly. The problem happened perhaps once in a hundred increments. The cause was a simple mistake: I had not synchronized the signal coming from the rotary encoder before using it in the circuit.

The rotary-encoder decode logic sets the count-up or -down signal correctly and then enables the counter for one clock cycle each time both rotary-encoder inputs are low. As the initial circuit was designed, the count-enable signal was set up for a random length of time before the clock. Most of the time the counter counted, sometimes it didn't, and other times it jumped because part of the count logic had sufficient setup time, and part of it didn't.

I ran the problem in simulation and it behaved just like the real thing. As I reduced the setup time below 5 nsec, jumps occurred on some count transitions. As setup time dropped below 2.5 nsec, the circuit just didn't count.

Adding a flip-flop to synchronize the signal solved the problem. If I were concerned about metastability, I would have added a second flip-flop. The effects of metastability in this application were not catastrophic and should be very infrequent with the long clock cycles.



(a) (b)

Fig 6—The record-playback circuit provides 2.5-mV resolution on a ±5V signal. The two scope photos here show the voltage range and resolution. In (a) you can see the circuit has captured an 8V p-p, 500-Hz triangle waveform when I switched out an attenuator on the function generator. The waveform is being played back at twice the record speed (1 msec/div and 2V/div). The same waveform in (b) is being played back with the oscilloscope settings changed to 20 mV/div and 0.5 msec/div.

EDN·SPECIAL PROJECT

Incidentally, I did not incorporate any debounce circuits for the switches because I previously concluded they were not necessary for this design. When switching the clock frequency, I assumed a reset was necessary.

Days 25 to 28

I added the flip-flop required to synchronize the encoder inputs to the schematic, then placed and routed the design. On this place-and-route run, all I/O pins were fixed. I resimulated all FPGA functions with special attention to make sure the bug was fixed. I also verified all other functions to make sure the place-and-route changes did not adversely affect other functions. Simulation indicated the FPGA should fully function at 12.5 MHz.

I spent the remainder of the day working over the analog portion of the circuit to get the best performance I could.

I went back to Actel to program a new chip. The problem was solved; the circuit now appears to work properly.

I spent more time on the analog. Digital is either right or it's wrong—analog can always get better. By the end of the day I decided I had done all I could for the analog and decided to take the weekend off.

By Sunday afternoon I couldn't stay away, so I spent an hour using the circuit to capture signals. I found a bug. The circuit was supposed to capture signals with 25% pretrigger data and 75% post trigger data. About half the time it worked correctly, and the other half of the time it captured signals with 75% pretrigger data. I couldn't believe I didn't notice this problem earlier.

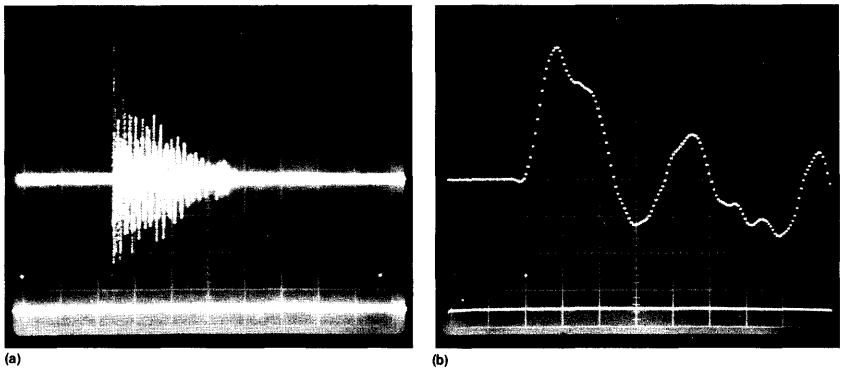
I was sure the error must be in how I computed the memory-trigger match signal (&MEM_TM), but it took me a while to see the problem. A simple logic error. The cases I tested earlier in simulation all worked properly. I added new test cases, and the bug showed up in simulation. I fixed the error by adding an XOR gate and reverifying the circuit in simulation.

I carefully tested the hardware one more time to make sure I couldn't find any more bugs. I went through the steps of placing, routing, and verifying that all simulations ran correctly. This time I sent my fuse files for programming the FPGA to Actel by modem. They programmed the part and mailed it back.

Day 29

The FPGA arrived. I plugged it in, and the circuit was fully operational. The project was finished. **Figs 5, 6,** and **7** show the circuit in operation.

The circuit was designed for recording with a 2-MHz clock rate. The clock-speed limit was set by the analog circuitry performing conversion. During playback, the circuit could run much faster. At 16 MHz, all playback functions appeared to work properly. Simulation indicated that the circuit was operating on the ragged edge at 16 MHz. At 20 MHz, some of the counters were



(a) (b)

Fig 7—The 32k-word record length provides 197 msec of data at 6-μsec intervals. The signal in (a) is an acoustic noise amplified from a microphone. The upper trace is the full record length. The lower trace shows the data beginning and end markers. The circuit captures the signal with 25% pretrigger data. Photo (b), 20 msec/div, 0.5V/div, shows the same waveform played back at 100 μsec/div. The capture trigger location is visible on the lower trace.

EDN·SPECIAL PROJECT

occasionally causing errors because the ripple carry had insufficient setup time, just as simulation indicated.

Hindsight is 20/20

The first step in simulation, and perhaps the most difficult and important one, is to make a list of all the cases that require testing. Simulation can only find problems when you look for them. My faulty logic for determining when to stop the counter to acquire 25% pretrigger data was tested using two different start addresses. Both of them worked correctly. The counter was free running and could start on any of its 32,768 values, so I didn't think it practical to test them all. Had I given more thought to the problem, I might have tested a few more critical cases to verify the logic and find the problem in simulation.

My error in not synchronizing the optical encoder inputs was a careless design oversight. Although the problem can be found with simulation—I verified it with simulation after I found it—this type of problem could probably have been avoided by taking more care in the design process. Anytime you have asynchronous signals coming into a synchronous system, they demand plenty of careful consideration to make sure they won't have undesirable effects.

I did not make the same kind of careful schematic check I normally do before having a circuit built. I thought simulation would provide a more thorough job finding errors than my going over the schematic a few more times. I also felt the time pressure to finish the project.

I think simulation did help me make a more thorough check of the logic than I could have done without it. Simulation however, should not be a substitute for carefully checking your schematic to identify potential sources of trouble. Once you've identified potential problem areas, simulation can help you test them.

Although I had hoped to be able to report a fully functioning circuit on my first silicon, reality turned out different. In retrospect, my experience on the project probably points out the strong points of FPGAs. I don't know how many days or weeks I would have had to spend on simulation to find the two bugs that slipped through. The problems were easy to find in real silicon and didn't take much longer to fix than when you find them in simulation.

I wouldn't want to push the approach of finding your mistakes in silicon too far. Simulation provides a better way to test a design over the full operating temperature and voltage ranges plus manufacturing process variations. I think of finding mistakes in silicon as a fall-back position after you've done the best you can in simulation.

The realities of schedules that don't allow weeks to simulate a design as completely as you'd like may force you into a corner if it is vital that first silicon be final silicon. I used as much time as I had for simulation—about four and a half days, which included learning to use the software, and then went on to try the real device. It would not have been worth another week of simulation to find the two problems I found in silicon. I'd have a different perspective if I'd designed a mistake into a mask-programmed gate array and spent $10,000 and lost a few weeks before I found my mistake.

Had I made a design mistake that left the FPGA with serious functional problems, I could have used the diagnostic probe capability on the FPGA. The diagnostic probes let you look at any two nodes in the FPGA with an oscilloscope or logic analyzer.

I started this project with no experience designing FPGAs and none in digital simulation. I wanted to see if designing a circuit using an FPGA was really simple enough for a designer familiar with 7400-series TTL design to jump into expecting to produce the first cir-

cuit in a reasonable amount of time.

My conclusion based on the products I used on this project is that migrating to FPGAs is a step any design engineer should be able to make. You always have to stretch yourself when learning to use new tools, but the jump shouldn't consume great quantities of time while you come up to speed. In the course of this project, I've covered all the problems I had that were worth mentioning. There weren't many. In the end, my biggest problems were the normal system design issues of deciding what the circuit should do. Once I knew what I wanted to do, designing the circuit was relatively easy.

My biggest surprise was how well the software worked. I had a few problems, but frankly I expected more. My dread of simulation turned out to be unfounded. I actually look forward to using simulation on my next project. It's more enjoyable to find mistakes in simulation than in hardware.

For this project I chose a circuit that would not require high clock rates. As a result, I didn't spend any time refining the design to make it run faster. Had I needed the circuit to run faster, I'd have needed more time to refine the schematic and criticality file, and I'd perform more simulation runs.

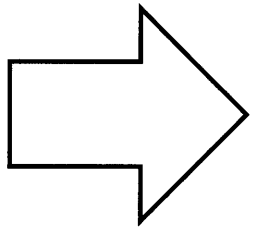
EDN

Acknowledgment

I'd like to thank the following companies for providing products for this project: Actel, Analog Devices, Hewlett-Packard, Linear Technology Corp, and Viewlogic.

Technical Editor Doug Conner is based in California. You can reach him at (805) 461-9669.

Using Actel Tools



High-Level FPGA Design in the Synopsys Environment

Application Note

Summary

Many system designers have abandoned traditional design methodologies in favor of a new high-level, top-down approach. Instead of designing a system chip by chip by capturing logic gates with schematic libraries, designers can describe the entire system at the behavioral level. This method of system design capture goes beyond functional specification. It can also include descriptions of external drive requirements, internal clock rates, electrical loading, and required signal arrival times. Today's new design methodology consists of mapping an HDL functional description into a technology library via synthesis.

What follows describes how to synthesize an HDL design description into an Actel FPGA using the Synopsys Design Compiler, FPGA Compiler, or Design Analyzer. Each compiler optimizes and maps the HDL behavioral description into an Actel ACT™ 1, ACT 2, or ACT 3 device family. After satisfactory optimization, the compiler can produce an EDIF 2 0 0 file containing Actel components. The Actel EDIF netlist program, cae2adl, converts the EDIF file to an Actel .adl netlist. After successful .adl generation, the Actel Action Logic™ System (ALS) software places and routes the design. After place and route, the ALS Timer, a static timing analysis tool, verifies the design's critical paths. Backannotation of actual routing delays to various simulators is also possible but not covered here. Figure 1 describes the basic design flow.

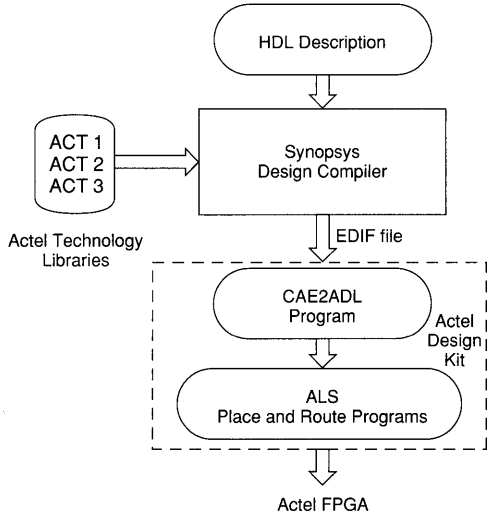


Figure 1. Actel/Synopsys Design Flow

Software Requirements

This design flow requires the following software tools from Synopsys and Actel to generate an Actel FPGA:

- Synopsys VHDL Compiler or HDL Compiler for Verilog. Depending on the design, one of these tools is required to compile VHDL or Verilog into internal Synopsys database format.
- Synopsys Design Compiler or FPGA Compiler or Design Analyzer. One of these tools is necessary to optimize the input from either the VHDL or HDL compiler. All three tools provide the same basic optimization and synthesis functions.
- Actel Synopsys Library. Actel provides libraries for ACT 1, ACT 2, and ACT 3 device families.
- Designer or Designer Advantage, Actel's FPGA software. Designer and Designer Advantage accept EDIF input from Synopsys. The software lays out the design into an Actel FPGA and generates a fuse file for programming a device.

Design Flow

The HDL Description

The design process begins with an HDL description of the design. Partitioning the design into several blocks of HDL and optimizing them individually achieves the best results. Each block of the design should have its own timing constraints and attributes. A top level of hierarchy contains instantiations of each partition and each I/O and clock buffer. I/O and clock buffers may be omitted from the top-level hierarchy when using the Synopsys 3.0 design compiler command "insert_pads". Insert_pads will automatically insert I/O and clock buffers in the design.

Compile Design in Synopsys

The Synopsys compiler reads the HDL description(s) of each design. Figure 2 depicts the steps that are performed in the Synopsys environment. As mentioned previously, you should set constraints and attributes individually for each partition. Some constraints and attributes to consider are max_delay, clock_period, max_fanout, and set_arrival. Compile each block and verify the critical paths with the Synopsys Timing Analysis Tool. The delays in the Actel Library are based on the estimated post-route delays listed in the *Actel FPGA Data Book and Design Guide*. These timing checks are estimates only. The actual delay information is derived from place and route with ALS. Hence, you should verify the HDL code only with the Synopsys timing analysis tool. After each block is compiled, issue a dont_touch on each partition. The dont_touch will stop the compiler from optimizing across hierarchical boundaries. Nevertheless, the top level must be compiled before an EDIF file can be written. As mentioned previously, the Synopsys v3.0 "insert_pads" command will insert I/O and clock buffers automatically into the top level.

This command is issued prior to optimizing the top-level HDL. If the version of Synopsys is not v3.0, I/Os and clocks must be manually instantiated in the HDL code.

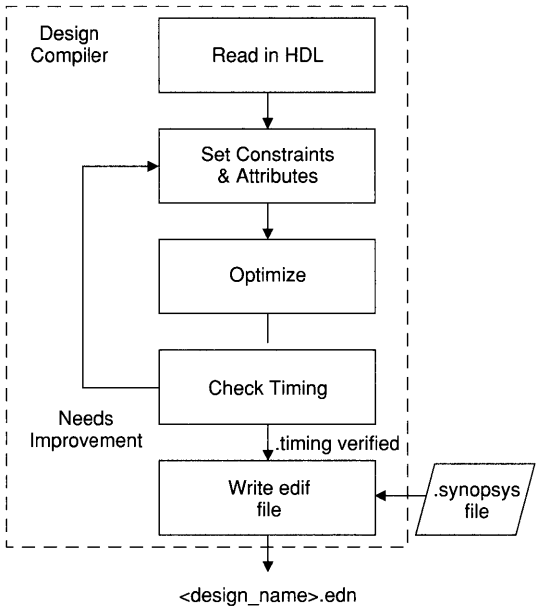


Figure 2. The Synopsys Design Compiler

CAUTION

The Actel libraries have fanout limits attached to each cell. However, the Synopsys Design Compiler does not maintain these limits across hierarchical boundaries. Hence, be aware of how each partition interacts with other blocks. For example, a signal in partition A may be confined to the ACT 2 fanout limit of 16, but it may also be connected to several more macros outside of its partition. The ALS software will produce an error message if the fanout exceeds the limit of 24.

The compiler writes an EDIF file for Actel translation once the entire design is compiled. Because EDIF has many "flavors," Actel requires specific switches in the Synopsys environment to produce Actel-compatible EDIF. Figure 3 lists the EDIF switches Actel requires. These switches are placed in the .synopsys file.

```
edifout_netlist_only = true
edifout_no_array = true
edifout_power_amd _ground_representation = cell
edifout_ground_name = GND
edifout_power_name = VCC
edifout_ground_pin_name = Y
edifout_power_pin_name = Y
edifout_pertty_print = true
```

Figure 3. EDIF Switches in .Synopsys File

EDIF to Actel Translation

Synopsys produces an EDIF file named <design_name>.edn. The Actel program, cae2adl, included in the Actel Design Kit, *Designer Advantage or Designer*, converts the Synopsys EDIF into an Actel ADL netlist. The command is invoked from the command line in the directory where the EDIF file resides. The syntax for invoking cae2adl is

```
cae2adl -edn2adl fam:<value> ednin:< edif file
name> <design_name>
```

where

fam	is family, which will be ACT 1, ACT 2, or ACT 3
edin	is the file to be converted, which must have the name <design_name>.edn
<design_name>	is the name of the design, which must be the same name as the top level of hierarchy that was synthesized.

Layout of an Actel Device

Once an Actel .adl netlist is generated, the design is placed and routed into an Actel ACT 1, ACT 2, or ACT 3 device via the Actel ALS tools, Designer Advantage or Designer. For more information regarding these tools, refer to individual Actel datasheets, Application notes, and User Guides.

Actel's Cadence Interface

Application Note

Actel provides a comprehensive package supporting the Cadence™ CAE development tools. This package is the Actel's Cadence Design Kit and is available for Sun™ Workstation users. This kit provides a complete design environment from design capture to post-layout simulation.

The design kit's flexibility allows for different design entry methods. One method is to describe the design with a Verilog behavioral or gate-level description and to use Synopsys synthesis tools to compile the Verilog HDL code and optimize it into an Actel FPGA. Another method is to use the Cadence Design FrameWork schematic capture tool, Composer, to draw the schematic for the design. A synthesized design may also be imported to Composer through the VERILOG IN translator from the Design FrameWork.

Tool Requirements:

- Sun 4 Workstation
- Cadence Verilog XL simulator
- Synopsys Design Analyzer, Design Compiler, or FPGA Compiler to provide the synthesis and mapping functions
- Synopsys HDL Compiler
- Cadence Design FrameWork (only if schematic capture is used)
- Actel Designer Advantage tools

Verilog/Synopsys Design Flow

The basic steps to create a design using the Verilog/Synopsys/ALS design flow, as illustrated in Figure 1, are as follows:

1. The starting point for the design process is to describe the design in behavioral or gate level Verilog HDL language or both. Describing the design behaviorally allows the design to be technology independent. The behavioral description can be mapped into an Actel library using the Synopsys synthesis tools.
2. The Verilog design file generated in the previous step is used to synthesize the design through the Design Analyzer provided by Synopsys. The appropriate library (ACT™ 1, ACT 2, or ACT 3) must be specified in the **.synopsys** file before reading the design into the Synopsys tools. After all constraints and optimization considerations have been obtained, a Verilog and EDIF 2 0 0 netlist must be generated from the Synopsys tools.
3. The Verilog netlist generated from the Synopsys tools contains a gate-level description of the design and may be used to perform functional simulation using the Verilog simulator.
4. The EDIF netlist generated by the Synopsys tools is used by the Actel EDIF reader program to translate the design from EDIF format to an Actel netlist. This program is called **edn2adl** and is included in the Designer Advantage software provided by Actel.
5. After an Actel netlist is generated, the design is mapped into an Actel device. Designer Advantage software performs the necessary functions. These functions include automatic pin assignment, placement and routing, and fuse file generation for programming.
6. The timing of the design may be checked using the Actel static timing Analyzer—the TIMER. This tool serves as a complement to the simulation process.
7. In addition to timing analysis, post-layout simulation is performed to verify the design. The backannotation of delays may be done for specific environments: Commercial, Industrial, or Military. The Verilog simulator is consequently invoked using the intrinsic module and routing delays. This allows the post-layout simulation to verify the design functionality within the device application environment.
8. The device is ready to be programmed.

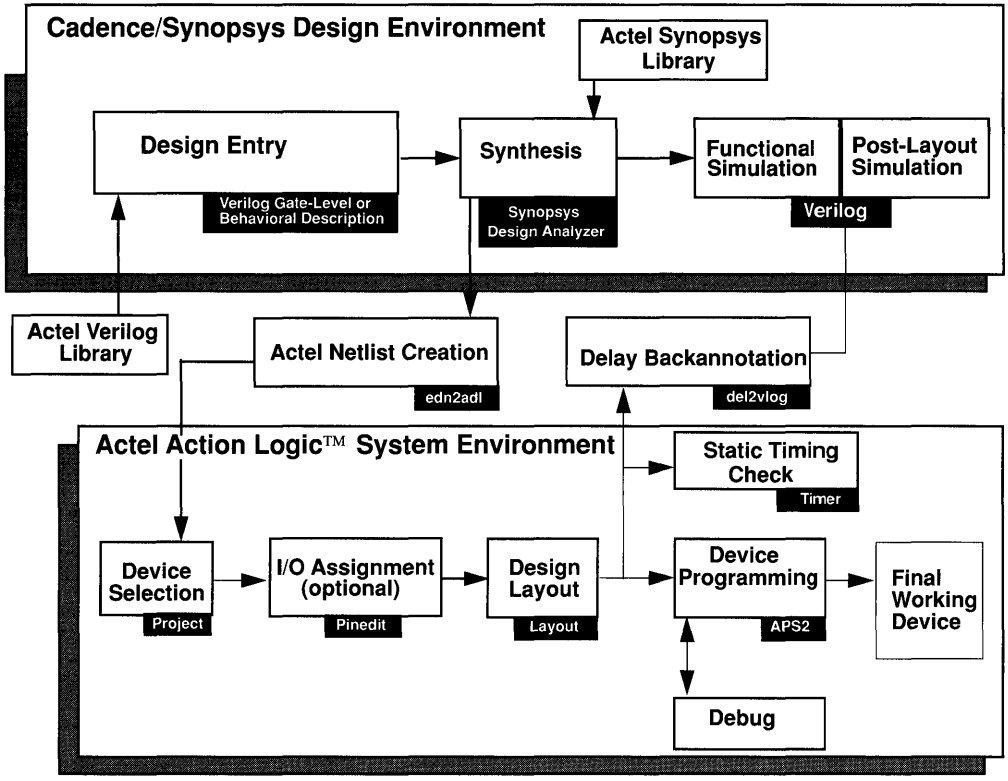


Figure 1. Verilog/Synopsys Design Flow

Composer Design Flow

In the design flow shown in Figure 2, the Composer schematic capture tool within the Design FrameWork is used as the design entry tool. Also, the Verilog simulator in this flow is invoked through the Design FrameWork.

The basic steps to create a design using Composer/ALS design flow, as illustrated in Figure 2, are as follows:

1. This design path begins by capturing the design with Composer from the Design FrameWork environment and either ACT 1, ACT 2, or ACT 3 library. (Also, a synthesized design may be imported to Composer using the VERILOG IN translator from the Design FrameWork. In this case, a gate-level description of the design needs to be input to the VERILOG IN function.) Once the design is captured with Composer, a Verilog and an EDIF netlist must be generated from the Design FrameWork.
2. The Verilog netlist generated from the previous step contains a gate-level description of the design and may be used to perform simulation using the Verilog simulator.
3. The EDIF netlist generated from the Design FrameWork is used by the Actel EDIF reader program to translate the design from EDIF format to an Actel netlist. This program is called **edn2adl** and is included in the Designer Advantage software provided by Actel.
4. After an Actel netlist is generated, the design is mapped into an Actel device. Designer Advantage software performs the necessary functions. These functions include automatic pin assignment, placement and routing, and fuse file generation for programming.
5. The timing of the design may be checked using the Actel static timing Analyzer—the TIMER. This tool serves as a complement to the simulation process.

6. In addition to timing analysis, post-layout simulation is performed to verify the design. The backannotation of delays may be done for specific environments: Commercial, Industrial, or Military. The Verilog simulator is consequently invoked using the intrinsic module and routing delays. This allows the post-layout simulation to verify the design functionality within the device application environment.
7. The device is ready to be programmed.

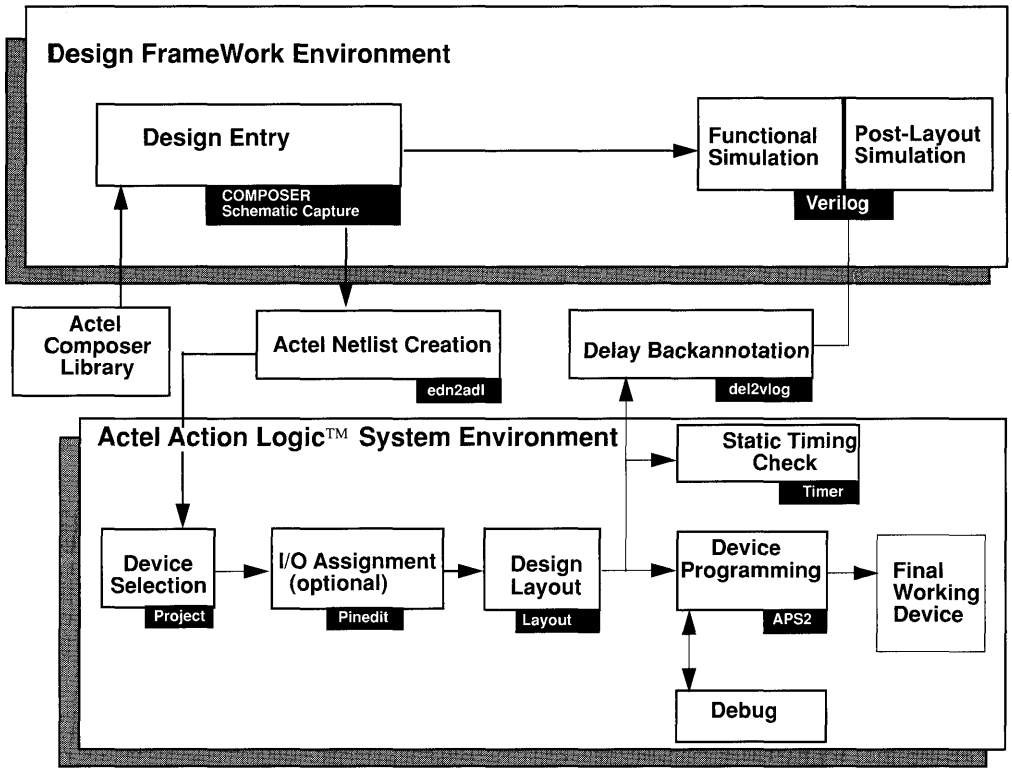


Figure 2. Composer Design Flow

Instantiating Actel's I/O Buffers in Verilog HDL

Application Note

Introduction

Actel supports design environments where Verilog HDL language is used to describe the design. This method of design entry allows for a high-level behavioral description of the design without any gate-level specifications. This HDL description, once compiled and synthesized with Synopsys and the Actel library, is mapped into an Actel field programmable gate array (FPGA). During the synthesis process, designs are mapped into Actel library macros, including the I/O buffers. Synopsys 3.0 provides this I/O mapping capability. Previous versions of Synopsys did not perform automatic I/O buffer instantiation. What follows briefly describes methods of Actel I/O buffer instantiation in a Verilog HDL design.

Instantiating I/O Buffers

With Synopsys 3.0, there is no need to instantiate I/O buffers in a gate-level format in the HDL design file. The exceptions to this rule are the tristate and bidirectional I/O buffers, which must be instantiated in a gate-level format in the HDL design file. To perform automatic I/O buffer mapping with the design_analyzer or the dc_shell command line, the following switches must be set prior to optimization:

```
set_port_is_pad all_inputs()
set_port_is_pad all_outputs()
set_pad_type all_inputs()
set_pad_type all_outputs()
set_pad_type -clock CLK
insert_pads
```

Note: CLK is an example signal name.

These switches will cause the Synopsys tools to map each port specified to an Actel I/O input buffer. The output ports are mapped to I/O output buffers. If there is a clock function in the HDL design file, it will be mapped to a CLKBUF. After each switch is set, messages appear showing the corresponding switch function being implemented for the applicable I/O ports, as shown in the example below:

```
Performing set_port_is_pad on port 'in1'.
Performing set_port_is_pad on port 'out1'.
Performing set_pad_type on port 'in1'.
Performing set_pad_type on port 'out1'.
```

If a version of Synopsys released before version 3.0 is being used or if a tristate or bidirectional I/O function needs to be implemented, the I/O buffers must be instantiated in a gate-level format in a Verilog HDL design file. The syntax for gate-level Actel I/O buffer instantiation in a Verilog HDL design file is shown in the example below:

```
module top(clear,clock,enable,outx);
input clear, clock, enable;
output [4:0] outx;

reg [4:0] outx_o;
reg clear_i,clock_i,enable_i;

count U0 (clear_i,clock_i,enable_i,outx_o);

CLKBUF U1 (.PAD(clock),.Y(clock_i));
INBUF U2 (.PAD(clear),.Y(clear_i));
INBUF U3 (.PAD(enable),.Y(enable_i));
OBHS U4 (.D(outx_o[4]),.PAD(outx[4]));
OBHS U5 (.D(outx_o[3]),.PAD(outx[3]));
OBHS U6 (.D(outx_o[2]),.PAD(outx[2]));
OBHS U7 (.D(outx_o[1]),.PAD(outx[1]));
OBHS U8 (.D(outx_o[0]),.PAD(outx[0]));
endmodule
```

In this example, the module "top" represents the highest level of hierarchy that instantiates the main behavioral Verilog HDL design module "count". In addition, the I/O buffers are described in a gate-level format.

The list of ports specified in the module "top" represent the Actel I/O pin names. These I/O pin names (clear, clock, enable, outx) are connected to the PADs on the I/O buffers. The items on the list of ports specified in the count module have a one-to-one correspondence to the PAD pins. The syntax of the Verilog HDL code instantiating the Actel I/O buffers is shown in Figure 1.

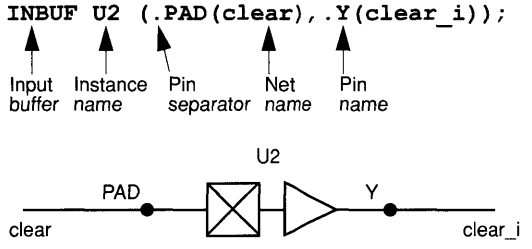


Figure 1. Actel I/O Buffer with Verilog HDL Code Instantiation

ALS EDIF Reader and Writer

Application Note

The diversity of CAE systems and tools without common standards has created many difficulties in transferring data among dissimilar design systems. The search for a solution to these problems led to the development of the Electronic Design Interchange Format (EDIF), a standard interchange format for electronic design data. The benefits of adopting EDIF are wide ranging. Among its many benefits, EDIF enables a wide choice of ASIC vendors by maintaining compatibility with a variety of device and CAE tools. The Actel system offers EDIF read and write capabilities to ensure compatibility with EDIF-based design tools.

ALS EDIF Reader

The Actel edn2adl EDIF reader translates an EDIF 2 0 0 netlist file into an ALS-compatible netlist file. The EDIF reader translates the EDIF source into an ADL netlist file. The program also creates an External Name Map (ENM) file that contains all original names renamed during translation from the original schematic. Properties in the EDIF source may be selectively imported into the ADL file. The usage statement for the **edn2adl** program follows:

```
edn2adl  [ FAM : {ACT1 ACT2 ACT3} ]
         [ EDNIN : <EdifFile> ]
         [ EDNINflavor : {generic wv mentor or} ]
         [ ADL : <AdlFile> ]
         [ AAL : <AalFile> ]
         [ ENM : <EnmFile> ]
         [ EDNINcfg : <ConfigFile> ]
         [ LIBoverRide : T ]
         [ NObadOrigName : T ]
         [ GNDnetname : <GlobalGroundNetName> ]
         [ VCCnetname : <GlobalPowerNetName> ]
         [ GNDnetProp : <GroundNetProperty> [ : <value> ] ]
         [ VCCnetProp : <PowerNetProperty> [ : <value> ] ]
         [ EDNINprops : <prop1> [ + <prop2> ... ] ]
         [ BUSformat : %s<%d> ]
         <design name>
```

The program switches control the treatment and characteristics of the input and output files. What follows are detailed descriptions of some of these switches.

EDNINflavor : {generic wv mentor or}

Some CAE vendors create EDIF output files with unique "flavors"—nonstandard formats that specify some of the properties or characteristics of the netlist. This switch selects one of several different flavors supported by Actel. **edn2adl** reads generic EDIF files as well as EDIF files with Viewlogic®, Mentor Graphics®, and OrCAD™ flavors.

ENM : <EnmFile>

The edn2adl reader changes the names of certain objects such as cells, instances, nets, and ports. The ENM file records a mapping of the original names of such objects. These objects have the ENM property with the original name as its value.

NObadOrigName : T

If the EDIF source contains a *rename* clause, the default action will be to import the original name into ALS. For example,

(rename SLASH_X "/X")

will result in "/X" being used in the ADL file as long as the netlist program permits it, perhaps with warnings. To ensure that all names imported into ALS are legal, set NObadOrigName to the value "T". The EDIF name will be used if the original name is not legal in ALS. In this case, "SLASH_X" will be generated and an entry will be recorded in the ENM file that can be used to map this back to "/X". But if the rename clause was

(rename BANG_X "!X")

the original name "!X" would get imported since it is legal in ALS, and there will be no entry made in the ENM file. EDIF names beginning with a nonalphabetic character are escaped with a '&'. For instance,

(rename &1234 "1234")

will generate "1234" with or without any setting of the NObadOrigName variable.

Since the name domains are mixed when this option is used, collisions may occur. Assuming the correctness of the EDIF netlist in such situations, a new unique name is generated and an entry made in the ENM file. So if there were two netlist objects with the following names in the same scope,

(rename SLASH_X "/X")

(rename SX "SLASH_X")

the first object would get named "SLASH_X" since its original name is not legal in ALS. The original name of the second object is legal, but its name was used for the first one. So the second object is assigned a unique name (say "SLASH_X_") and this mapping is recorded in the ENM file.

GNDnetname : <GlobalGroundNetName>
VCCnetname : <GlobalPowerNetName>

The value of these variables, if specified, indicates the global name of ground and power nets. For example in the case of Viewlogic, use

GNDnetName:GND

VCCnetName:VDD

When a net with these names is encountered, both GLOBAL and POWER properties will be attached. The net names used to match the given values are controlled by the NObadOrigName variable in the presence of *rename* clauses.

ALS EDIF Writer

The **adl2edn** EDIF writer translates Actel netlists into EDIF 2 0 0 format. The usage statement of **adl2edn** is shown below. The various options in this command allow for different traversal algorithms, power and ground representations, and use of a set of preferences from a configuration file.

```
als -adl2edn [ TYPE : {hier flat lib} ]
             [ POWERstyle : {cell net port portVerbose} ]
             [ EDNOUTadl : <adl_file> { + <adl_file2> ...} ]
             [ EDNOUT : <edif_file> ]
             [ EDNOUTflavor : {generic wv synps} ]
             [ EDNOUTcfg : <config_file> ]
             [ EDNOUTprops : <prop1> { + <prop2> ....} ]
             [ REPLACEname : T ]
             [ GNDnetname : <newName> ]
             [ VCCnetname : <newName> ]
             [ GNDnetProps : <name1> = <val1> { +
             <name2> = <val2> ...} ]
             [ VCCnetProps : <name1> = <val1> { + <name2>
             = <val2> ...} ]
             [ EDIFOUTcriticality : T ]
             [ NOportDirection : T ]
             <Design>
```

As shown above, specify the name of design as the last argument following any of the optional switches. The program switches control the treatment and characteristics of the input and output files. What follows are detailed descriptions of some of these switches.

TYPE : {hier flat lib}

This switch determines the netlist type for the output EDIF file. The netlist can either be hierarchical or flattened. The **lib** type output file is a hierarchical netlist including definitions for Actel ADLIB library components.

EDNOUTflavor : {generic wv synps}

The **adl2edn** writer generates generic EDIF files as well as EDIF files that can be read by Viewlogic and Synopsys systems. By default, this switch selects the generic EDIF file.

EDIFOUTcriticality: T

Nets with the CRT property are translated to the EDIF *criticality* construct using the relations in Table 1.

Table 1. Criticality Mapping

CRT Value	EDIF Criticality
U or L	–10
M or H	10
F	20

Mentor Graphics V7 to V8.2 Design Conversion

Application Brief

Introduction

Converting Mentor Graphics® designs is a topic of increasing concern as more V7 ASIC customers move to V8.2. To make the conversion, one uses the V8.2 conversion tools to migrate V7 designs into the V8.2 world.

What follows is a summary of the conversion of an Actel version 2.11 (or earlier) design compatible with Mentor Graphics V7 to an Actel version 2.2 design compatible with Mentor Graphics V8.2.

However, be aware that Actel's support for Mentor Graphic's V7 was for the DN3000, DN3500, DN4000, and DN4500 platforms, and that Actel currently supports only HP700 and Sun 4 (or fully compatible) Workstations for Mentor Graphics V8.2.

The conversion program from Mentor Graphics is compatible with DN workstations only. It reads the ALS 2.2 libraries and map files, which are first installed on the Sun workstation and copied to the DN workstation. After the conversion is completed on the DN workstation, the design must be transferred back to the Sun environment to recompile with ALS 2.2.

Follow these steps to complete the process:

1. Prepare the design for conversion.
2. Remove the "dollar sign" from the V7 design.
3. Convert the design.
4. Update symbols and nets.

Preparation

Use the following procedure to prepare each design for conversion to Mentor Graphics V8:

1. Install ALS 2.2 software on the Sun Workstation and add $ALSDIR, the Actel environment variable, to the path. More information on ALSDIR is found in the *CAE Guide: Sun/Mentor Graphics for ALS 2.2*, Chapter 3.
2. Transfer the Actel Libraries and map files from the Sun to DN workstation and set $ALSDIR. The library files are in the $ALSDIR/lib/me/parts directory, and the map files are in the $ALSDIR/des_arch directory.
3. Install Mentor Graphics V8 software on the DN and Sun workstations. Make sure that gen_lib is included in the software.
4. On the DN workstation, set all paths and environment variables for V7 and V8. Refer to the *CAE Guide: Sun/Mentor for ALS 2.2*, Chapter 3.
5. On the HP700 or Sun workstation, set all paths and environment variables to V8. Refer to the *CAE Guide: Sun/Mentor Graphics for ALS 2.2*, Chapter 3.
6. Create the MGC_LOC_MAP file (Mentor Graphics Location Map file), which should include the correct soft and hard links to the working directories for both ALS 2.2 and Mentor Graphics V8. A sample copy of the MGC_LOC_MAP file is included in the *CAE Guide: Sun/Mentor Graphics for ALS 2.2*, Chapter 3.
7. Verify that the V7 design is free of errors and that every block is expanded. Then delete extra versions of the design using "dlv ... 1" unless otherwise desired.
8. Verify that $MGC_WD is set to 'pwd' (project working directory).
9. Check to see if the V7 design has references to the old Actel library.

Remove Dollar Sign

The "remove_dollar" program from Mentor Graphics removes dollar signs ($) from the Mentor V7 design files. The dollar sign ($) has a special meaning for Location Map Variables in Mentor Graphics V8. The dollar sign references must be removed before converting from V7 to V8. the "remove_dollar -help" command gives more information on this, as does the Mentor Graphics document, "Transition Guide for V8 Capture Products."

If the "remove_dollar" command is not available on your system, follow these steps:

1. From one level above the design directory (not in the design directory), type the V7 command:

   $MGC_HOME/unrlsd/listref <design_name>.

   References should be pointing to the old ALS library path.
2. *cd* to the design directory and list references by typing: *ls *.ref*
3. Remove files that have the following extension: **.erel_**. These files refer to the flattened design.
4. To remove directories associated with the flattened designs, type: *rm -r *.erel_*\$*.bak*. At this stage, the directory should contain all ALS files in addition to the sheet and symbol files generated with Mentor Graphics V7.
5. Make sure that $MGC_WD is still set to 'pwd', otherwise type: *setenv MGC_WD 'pwd'*
6. To change references from the old Actel library to the new Actel library, type:

   chref <design_name> '/user/als/lib/x' '/user/als/lib/me/parts'

   where x = a1000 for ACT 1 library or x = 1200 for ACT 2 library.
7. To verify whether the design is referencing to the new library, type: *listref <design_name>*

7

Conversion

After preparing the design and removing dollar signs from the design files, the conversion program can be executed. The conversion program is called cvt_comp. This V8 program is located in $MGC_HOME/bin on the DN workstations only. To run the conversion program, type:

cvt_comp <design_name> -preview -convert -nodelete -nomodel -map <map_path>

Where:

- *preview* checks the conversions needed and produces a list of parts within the design that are to be converted.
- *convert* selects both symbols and schematics to be converted.
- *nodelete* keeps V7 intact during the conversion process until conversion is complete.
- *nomodel* means no behavioral models exist (i.e., VHDL).
- *map <map_path>* specifies an ACT 1, ACT 2, or ACT 3 family map file for referencing new V8 symbols, and *<map_path> =$ALSDIR/des_arch/actx_map_file.*

After a successful conversion, the program issues the following message:

conversion complete, 0 errors.

Now the conversion is complete. During the conversion process, the convert command creates the V8 schematic and symbol files with updated references. If the conversion fails, all files referring to the V8 design must be deleted before running convert a second time.

To verify that all references are updated, type the V8 command:

$MGC_HOME/bin/listref <design_name>

Updating Symbols and Nets (V7.X to V8.X and V8.1 to V8.2)

There is some cleanup necessary of the V8 schematics due to the changes in the basepoints of the symbols from V7 to V8. All Actel symbols must be selected individually or in groups and then updated to properly route the schematic nets. To update the symbols, invoke Design Architect (DA) with the design and from the popup menu, select:

edit > select > area > instance
edit > update > instance

Now the nets are connected to the symbols properly but are not straight. The nets must be moved manually to make them orthogonal. The basepoint is not the same for all parts.

One of the advantages of Actel's FPGA design approach is the automation of each development step. This "push button" approach used by Actel's development software tools is best exemplified by the automatic Layout program. The Layout program performs automatic pin assignment and placement and routing. This approach saves hours of development time, shortens the learning periods, and avoids tedious manual placement and routing sessions.

However, in applications where extracting maximum performance from the device for certain critical paths is important, manual pin assignment and module placement may be the better approach. Actel provides ChipEdit, a graphical tool for manual pin assignment and module placement, to provide a complete solution for these applications.

Using the graphical user interface provided by ChipEdit, you can assign or move pins or logic modules quickly and safely from the Designer Advantage environment. Figure 1 shows the ChipEdit graphical user interface.

The user interface includes a Graphical Display Area, a Message Bar, a Status Bar, an Object List Box, and a Context Window. Each of these features is now described briefly.

Graphical Display Area

The graphical display area shows the physical layout of the design in an Actel device (Figure 2). All modules are identified by both color and symbol. Different Actel families have different modules available.

A module that contains a macro is filled with the color that identifies the module type, while a centered black symbol identifies the macro type. When a module is empty and available for use, it is unfilled and its symbol appears in the color that identifies the module type. If a macro takes up more than one module, a solid line appears between the two modules containing the macro.

Message Bar

The message bar displays one-line help messages.

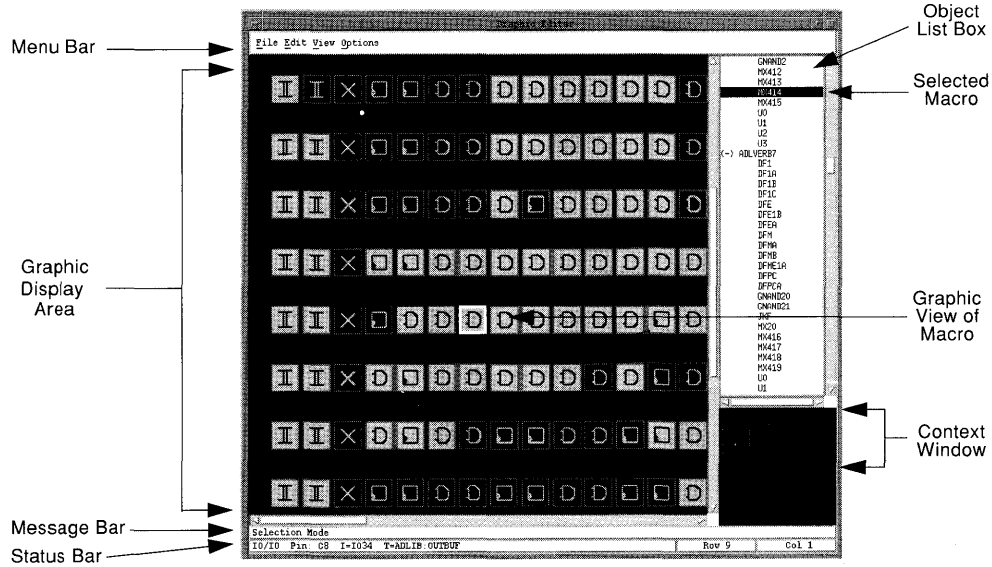


Figure 1. ChipEdit Window

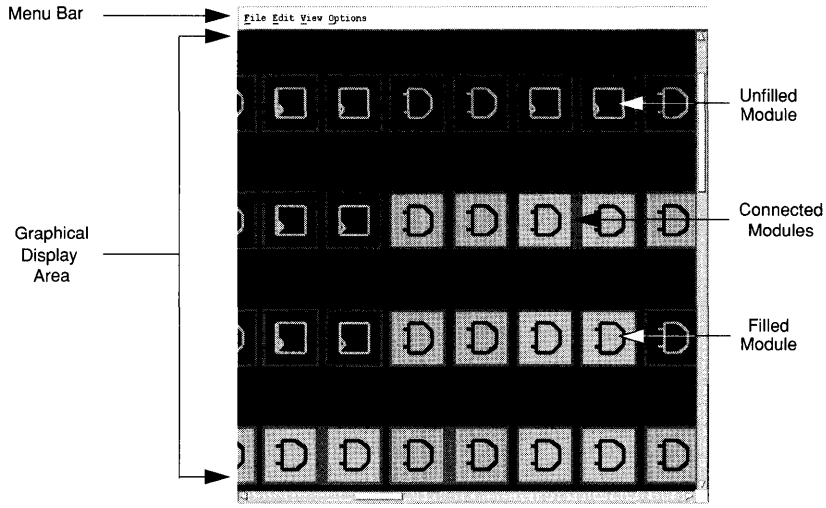


Figure 2. Graphical Display Area

Figure 3. Status Bar

Status Bar

If the pointer is located over a module, the status bar displays information about the module, such as its row and column. If the module is not filled, only the module's type will be displayed in the status bar. If the module is filled, the status bar provides information about the module type and macro placed inside the module. Figure 3 displays the status bar.

Object List Box

The Object List Box displays objects by their text names. Macros may be displayed by their instance names, I/O macros by their net names, and I/O modules by their pin names. The display may also be filtered.

While there is an Object List Box inside the ChipEdit window, multiple Object List windows can be opened outside the ChipEdit window environment.

Context Window

Located in the bottom right corner of the ChipEdit window, the Context window shows the location of the current zoom area within the chip design (see Figure 4).

In the Context window, the entire chip design is represented by a large rectangle, while the current position within the entire design is represented by a smaller rectangle called the "you are here" box. The view can be zoomed into a specific area of the design to view and edit macros, yet the position within the entire chip design can be located by referring to the Context window.

When the ChipEdit window is first opened, the Context window shows a one-to-one relationship. After zooming into a specific area of the chip and manually placing modules, the position within the overall design may be located by referring to the "you are here" box in the Context window. The position on the chip may also be altered by manipulating this "you are here" box.

Moving and Placing Macros

Macros can be moved or placed, individually or as a group, from the Graphical Display Area or the Object List Box. From the Graphical Display Area, simply by using the mouse, you can point to any macro and drag the macro to its new position. From the Object List Box, you can select any macro and move it to its destination by using the Edit menu. You can also have multiple ChipEdit windows open to view different perspectives of the

same design and to move a macro, or group of macros, from one ChipEdit window to another.

ChipEdit is a complement to the ALS software and its automatic Layout programs. It not only provides a graphical view of the Actel chip but also allows the editing of the pins and placement of logic modules. For applications where certain critical paths must be optimized for speed, ChipEdit is the ideal solution.

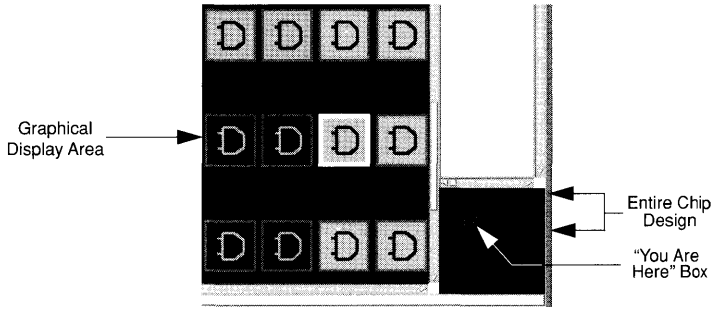


Figure 4. Context Window

ACTmap™ Design Flow

Application Note

Introduction

Since the introduction of the Programmable Logic Device (PLD) in the late 1970s, logic designers have been presented with a series of different design methodologies. Designs may be described in Boolean equation or state machine format, then compiled to generate fuse files for device programming. Boolean equations, written in the form of sum of products, map readily into the low-complexity, AND/OR array of PLDs. PLD synthesis tools are readily available to support these low-density devices.

Field Programmable Gate Arrays (FPGAs) are high-density programmable logic devices with a gate array-like architecture. Actel FPGAs have a flexible architecture that allows high gate utilization with automatic place and route for fast time-to-market. Experienced PLD designers who wish to use FPGAs while maintaining a PLD design methodology are supported by FPGA synthesis tools. The most important requirement of the FPGA synthesis tools is the efficient selection of architecture-specific macros from a general logic description. The need for such synthesis tools is critical to device performance and utilization. This application brief describes the ACTmap logic synthesis tools available from Actel.

Scope

ACTmap software tools allow the quick and easy implementation of logic in Actel FPGAs using popular PLD design tools. They accept behavioral inputs such as PALASM® and ABEL™ PLD description languages. It also accepts EDIF format descriptions generated from synthesis tools. Any combination of these previously mentioned input formats as well as library-based schematics may be used to describe designs. ACTmap optimizes designs for speed or area based on the design constraints. Once the design is optimized, its netlist is processed by the Action Logic® System (ALS) for layout. The design may be verified with post-layout delays before device programming. Figure 1 shows how ACTmap fits into the design flow.

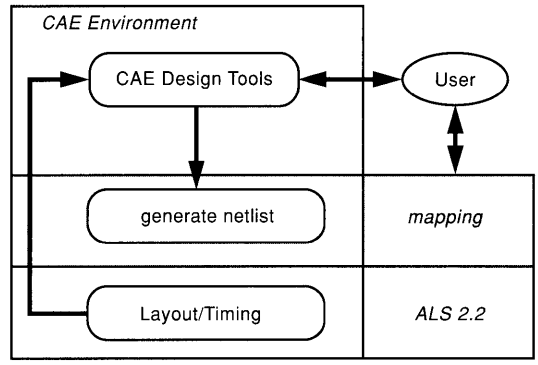


Figure 1. Scope of ACTmap

Design Flow

Actel FPGAs may be derived directly from PALASM-format source files or from ABEL™, CUPL™, or LOG/IC™-based descriptions. The design can be a combination of textual descriptions and schematics. Figure 2 shows a top-level design that combines PALASM-format descriptions with schematics. ACTmap transforms the equations or netlists or both into an optimized Actel-formatted design file. ACTmap uses an Actel-specific algorithm to find the best mapping for the particular ACT family device. The design can be mapped as individual blocks or as a complete design. ACTmap also accepts EDIF-format files generated from synthesis tools. Designs may be optimized to implement the highest performance or smallest area. Actel-specific, multiplexer-based algorithms map the initial design to fit the logic module architecture for optimum results. The final implementation is brought into ALS for layout and device programming. Figure 3 shows the design flow using ACTmap and ALS to configure Actel FPGAs.

7

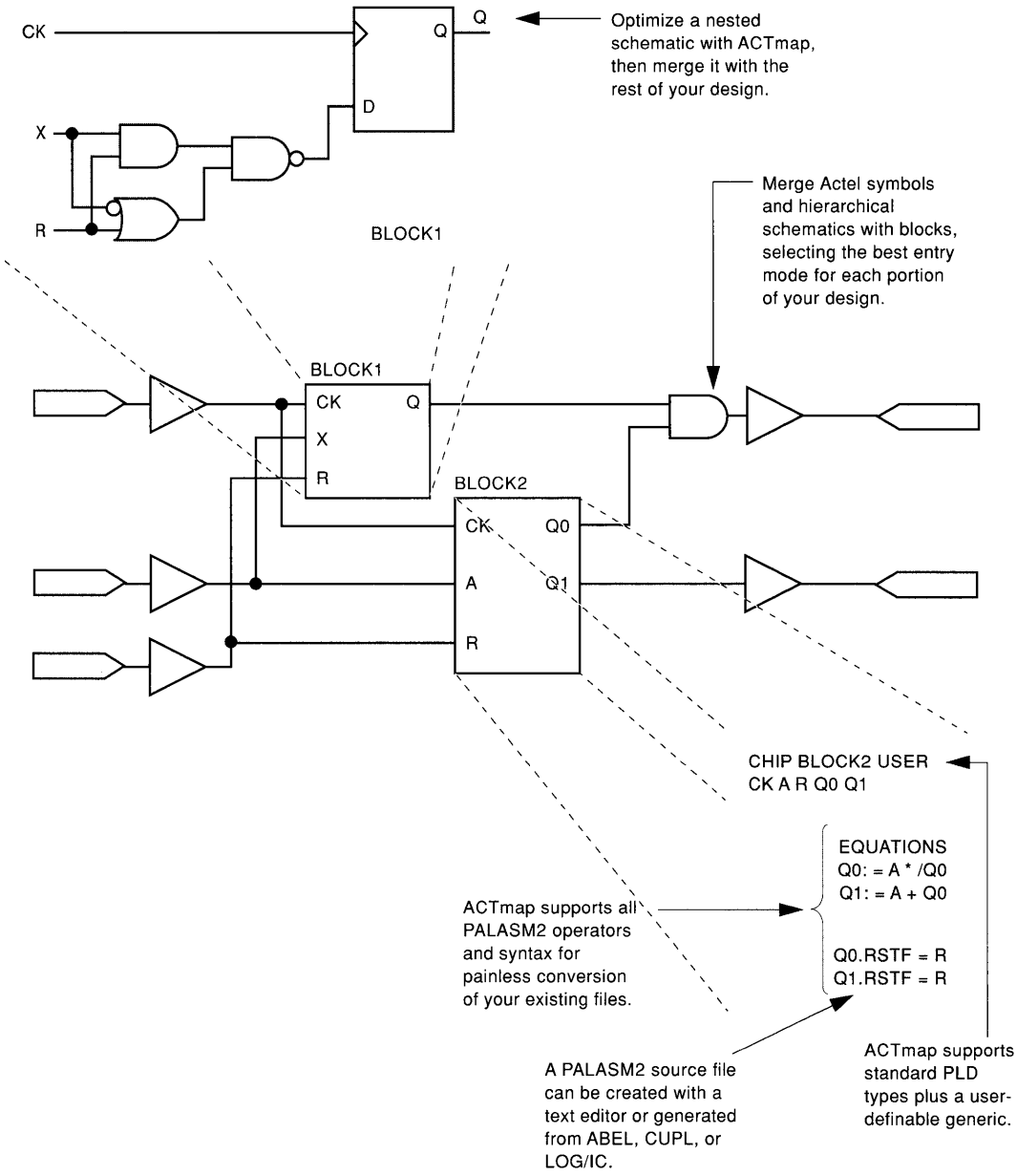


Figure 2. Top-Level Design

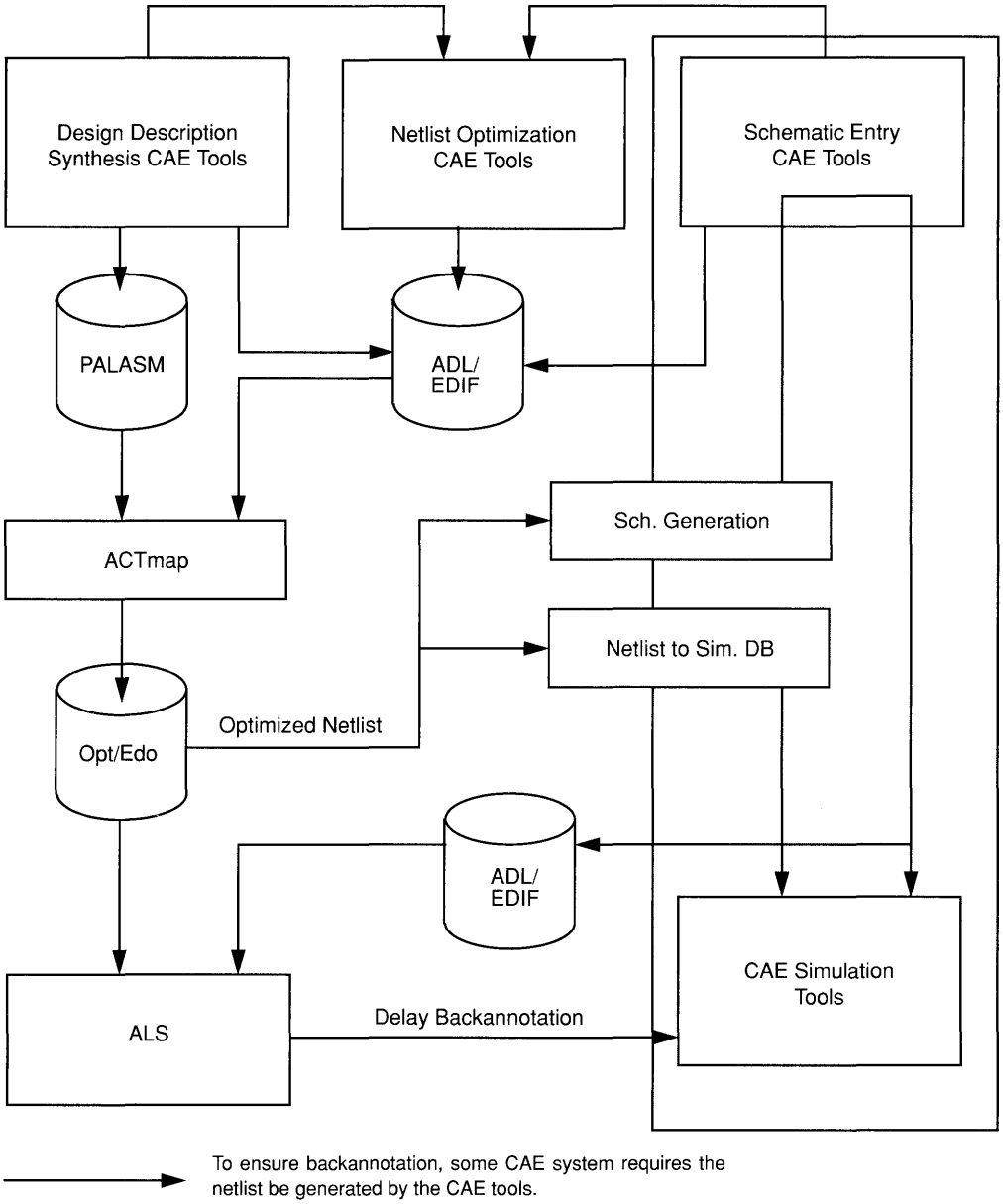


Figure 3. Design Flow

Selecting and Modifying I/O Assignments

Application Note

Introduction

Effectively selecting I/O assignments is critical to the success of an Actel field programmable gate array (FPGA) design. Although manual I/O assignment is optional in the design flow for the Action Logic® System (ALS), the importance of obtaining an effective assignment to the overall flow should not be underestimated. The placement and routing of a design may fail if the pin placements are not optimized for the design and ACT family device. Its overall performance is largely determined by the I/O selections as well. Poorly placed designs often require longer routing tracks, which slow the maximum operational frequency.

System designers know that I/O assignments have a great impact on schedules and deadlines as well. The I/O assignments for all ASICs must be chosen before printed circuit board (PCB) layouts can be completed. Aggressive schedules may require that the PCBs are designed and ordered before the electronic circuits have been tested and debugged. Completing the I/O assignments and the PCB layout becomes one of the gating items for on-time system delivery.

The ACT FPGA design flow includes several different methods of selecting I/O placements. The flexibility of ALS enables many design methods to lead to successfully completed designs. One of its benefits is that there is no exact "right" or "wrong" way to use the system. However, there are recommended methods that will increase the probability of getting optimal I/O assignments and fast, routable designs.

I/O Assignment Methods

ALS allows I/O assignments to be chosen after a design has been converted into an Actel netlist and its device type and family have been selected. There are three options available for assigning I/Os at this point. Each offers benefits that may apply to different design methodologies and system requirements. The I/O assignment options and their benefits are:

Automatic Assignment

No manual assignment is made for an I/O signal. ALS automatically selects the best pin location depending on characteristics of the design such as criticality assignments, device type and family, and overall design topology.

Manual, Fixed Pin Assignment

Pin numbers are manually assigned to I/O signals and are fixed in place. ALS will not move I/O assignments that are fixed. Manual, fixed assignments allow the maximum control over the design flow since the I/O assignments remain fixed regardless of design changes.

Manual, Unfixed Pin Assignment

Pin numbers are manually assigned to I/O signals but are not fixed to the I/O location. Since they are not fixed, the I/O assignments are used by ALS as suggested locations—they may be moved to other locations by the software. This method allows some input to the desired I/O configuration but gives ALS the flexibility to optimize the configuration further.

Any combination of these three methods may be used for the I/O signals of a design. Designs may have all pins manually assigned or automatically assigned, or have a percentage of each type. How these methods are used depends on the requirements of the each project.

Suggested Design Flow

Since the three methods of determining I/O assignments for an ACT family FPGA may be used in any combination, many methods are commonly used to create quality pin placements. The best method for a design depends on many factors such as performance specifications, design changes, and scheduling constraints.

Generally, the most effective method for pin assignment is to use automatic assignment for 100% of the I/O signals. ALS has been designed to optimize the placement and routing especially if it is given the flexibility to automatically assign all of the I/O placements. ALS is able to select, evaluate, and optimize many different I/O configurations specifically to the device architecture. If as few as 10% of the pin assignments are inefficiently assigned manually, the quality of the placement and routing may be compromised. Manual assignments should be minimized as much as possible. To obtain a 100% automatic assignment, skip the Pin Edit program in ALS and go directly to the Layout programs. The final pin assignments are stored in the ASCII file, <design name>.pin, located in the design directory.

If some of the I/Os require manual selection, use unfixed assignments as much as possible. This allows ALS the flexibility to modify some of the assignments as needed. In general, maintain the percentage of manual assignments, fixed and unfixed, to as low a level as possible.

The Most Effective Fast Design Flow

I/O assignment does not have to be a gating item to the overall system schedule. A near-optimal automatic assignment may be obtained from ALS before the circuit design is completely tested and debugged. As long as the number and function of the I/O signals are finalized, ALS can be used to assign all the pins before the details of the FPGA design are complete. Use the following

method to get a quick start on the PCB layout and speed schedules for fast turnarounds:

1. Begin by entering the design as completely as possible, ignoring small details that may need to be modified later. It is important to include all the major functions that will be in the FPGA circuit. The objective is to obtain a netlist that approximates the final design topology. The circuit does not need to be functionally correct at this point but must have the same major functional blocks (adders, counters, and so on) and approximately the same number of logic modules as the final design.
2. Determine the exact number of input, output, and bidirectional pins for the FPGA and include them in the netlist. Select the appropriate ACT™ family device for performance level, logic module capacity, and I/O count.
3. Skip ALS's Pin Edit step.
4. Run the Layout programs in ALS. Ignore any warnings from the Validate program that are due to the unfinished state of the design. Run the Layout programs with no clock balancing and incremental placement turned off.

The Layout programs will create a 100% automatic I/O assignment specific to the design topology. This is done before all the functional bugs have been discovered and resolved. Minor functional changes that may be made later will have little effect on the quality and effectiveness of the pin placements. The PCB layout can be completed with the confidence that these pin assignments will require no modifications in the future.

After the design is complete, use the Pin Edit program to read the automatic I/O assignments and fix them permanently. Once they have been fixed, ALS will always use the pin placements initially created by the Layout programs. The Layout programs will not modify any placements that have been fixed.

This method is of value even if the I/Os will be manually assigned. The results of 100% automatic assignment by the Layout programs can be used as a template for an effective manual I/O assignment. It is easy to use the existing placements as a guide and make small modifications as required.

Manual Assignments

If any or all of the I/O assignments must be determined manually, a broader knowledge of Actel FPGA architecture is required. This databook contains pertinent information regarding different architectures of the ACT families. The structure of the logic modules and I/O modules, routing tracks, antifuses, and other architectural details are covered in detail. With this source of information, it is possible to manually assign I/O signals with the specific details of the device in mind.

During the process of assigning I/Os for the device, keep these guidelines in mind to increase routability and performance:

1. Try to force signal paths, especially large data buses, to flow horizontally across the die since there are more horizontal than vertical routing resources. In most cases, a large data bus requires many interconnections as it traverses the circuit. The greater number of horizontal routing tracks handles these interconnections to reduce routing congestion. The horizontal and vertical orientation of the die is the same as shown in the package pin assignment and mechanical detail drawings in the *Actel FPGA Data Book and Design Guide*.
2. Count the number of levels of logic between I/Os to determine placement. For example, I/Os separated by one gate should be placed closer than I/Os separated by a long shift register. In general, the architecture of ACT devices allows signals to be routed across two vertical rows and over one-third of the columns of the device without using a long routing track. For an A1225 device with 13 rows and 46 columns, two I/O signals should not be placed on opposite sides of the device unless there are at least two horizontal or three vertical levels of logic between them.
3. Use the top and bottom I/O pins for slow or local signals or both. The fact that there are fewer vertical routing resources will not harm the performance of these signals.
4. Use the global clock buffers. Each of these pins is connected to a dedicated, low-slew distribution network optimized for high fanout signals.
5. Refer to the information in the PLI (PLacement Information) and RTI (RouTing Information) files as feedback for I/O assignment iterations. Each of these files indicates potential problem areas of the design that may be due to the I/O assignment. The PLI and RTI files report specific problems with the placement and routing of the device respectively. Determine from this information whether reassigning some I/Os will remove these problems.

Modifying Existing I/O Assignments

Unfortunately, it is not always possible to allow ALS to assign most or all of the I/O placements. In addition, the placements may have to be finalized before optimization by ALS for design routability and performance considerations. In any case, a variety of factors may result in a nonoptimal I/O assignment being created and used for a design. This may cause the initial placement and routing to fail or degrade the performance level unacceptably. If this occurs, there are several actions that can improve the placement, routing, and performance of the design.

Recognizing Poor I/O placements

It is important to recognize when placement, routing, or timing problems of an Actel FPGA design are caused by poor I/O assignments. ALS will indicate potential problem areas in several different ways. For example, the <design_name>.VLD file, created by the validate program, contains several messages and statistics that may indicate I/O placement problems. If more than 33% of the I/O pin assignments are fixed, a warning message is issued during the routability check. The same routability check will fail if a critical path (M or F) must use a long routing track.

The outputs of the place and route programs are the <design_name>.PLI and <design_name>.RTI files respectively. Both files contain information to help understand the cause of

possible placement and routing problems. The PLI file lists every net that requires the use of a long track for routing. There are a limited number of long vertical tracks (LVTs) and long horizontal tracks (LHTs) in each device. If an input or output pin is directly connected to a long track, the I/O placement can probably be improved.

Improving Routability and Performance

If the I/O placements for a design are fixed and causing placement, routing, or performance problems, there are several ways to modify the design and its implementation into an Actel FPGA. Try the following steps to increase the routability and performance of the design:

- Use criticality assignments but don't exceed the maximum number of critical nets for the device. Determine the speed critical and uncritical paths in the design and specify them in the criticality file, <design_name>.CRT. The criticality assignments help the place and route software to make critical nets faster and improve the overall performance. Be careful not to exceed the maximum number of critical nets for each device as specified in the *ALS User's Guide*.
- Do not use clock balancing if possible. If there are place and route problems, clock balancing may reduce routability and actually increase delay times. Reduce the clock balancing strength to the lowest level that will satisfy the clock skew requirements of the design.
- Reduce high fanout nets between I/O modules. Routing congestion between I/Os is especially critical to the placement and routing success. For low critical nets especially, use buffers to reduce fanout to a few loads.
- Watch the overall fanout statistics of the design. The validate program reports the average fanout for the design in the VLD file. A high average fanout (greater than 3.5 loads) will usually be harder to place and route. If possible, reduce the overall fanout by buffering high fanout nets. In addition, reduce the use of high fan-in macros to less than 25% of the total logic module count. High fan-in macros such as the DFM8A and CM8 have up to eight inputs per module. Using a large percentage of these macros may cause unsolvable routing congestion problems.
- Be careful of designs with high I/O utilization (greater than 80%) and low logic module utilization (less than 50%). Since most of the I/O modules but a low percentage of the logic modules are used, many long routing tracks may be required. This will deteriorate the routability and performance of the design. Remember that high logic module utilizations (greater than 90% and up to 100%) are common, so increase the utilization percentage accordingly or change to a smaller device.
- Check the PLI file for the list of long routing tracks and buffer them appropriately if they are high fanout nets.

Socketing Option

In a few cases, the previous suggestions may not solve problems caused by a poor I/O placement. A hardware solution may be necessary. There are commercially available IC sockets that re-map the pinout of the device to any chosen configuration. If such a socket is available, ALS may be used to reselect the I/O placements for the design without any constraints. The probability of a successful place and route with higher performance can be greatly increased this way.

Summary

The most effective design methodology for Actel FPGAs includes 100% automatic I/O assignment by ALS. The software is designed to optimize I/O assignments especially if it is given a large degree of freedom from manually fixed placements. If manual assignments are necessary, a broader background of the FPGA architecture is necessary to achieve similar results. If manual selections are necessary, use the suggestions from this application note to select the placements as well as possible. If the placements of the I/Os are fixed and they appear to be hampering the routability and performance of the design, there are still several modifications that may be made that can improve the probability for success.

Critical Path Analysis Using the Timer

Application Note

Introduction

The critical paths of Actel designs determine the maximum performance of the device in a system. These critical paths must meet system specifications to function properly. A worst-case timing analysis of the critical paths ensures that the device will work reliably in production. Critical paths are design dependent; they can be combinational paths from input pads to output pads, sequential paths from flip-flops to other flip-flops, clock to output pad paths, setup times and hold times of the field programmable gate array (FPGA) relative to other devices on board, or any combination of these paths.

The Actel Timer is a static timing analysis tool used to verify device timing. The Timer is used interactively or with command files to automate the timing analysis process. The Timer generates delay reports, including derating factors, for different operating conditions due to variations in temperature, voltage, and device process. Voltage and temperature ranges for commercial, industrial, and military devices are included in the menu selection.

This applications brief describes how to analyze critical path delays of Actel's FPGAs using the Actel Timer. It illustrates strategies to determine the propagation delays of combinational and sequential paths. It also explains the concept of net delays and how the Actel Timer reports them.

Net Delays in the Actel Timer

The Timer reports propagation delays between a Startset and Endset of pins in the design. The Startset contains the start pins and the Endset contains the end pins of the paths under investigation. The Timer lists propagation delays from the input pins of a macro in the Startset to the input pins of a macro in the Endset, including the net delays. These net delays are calculated from layout and fanout information. Keep in mind that the Timer reports timing from input pins to input pins so that all net delays are automatically included. Figure 1 shows that delay DEL1 includes the delays of the gate U1 as well as the net delay arriving at gate U2. Also note that the Timer can reference the output (PAD) pin of output buffers to include the propagation delay to the actual package pin. The output pins of output buffers are the only non-input pins that the Timer can reference for timing analysis. DEL3 in Figure 1 shows the propagation delay from the D pin to the pad pin of the output buffer

Combinational Paths

As mentioned earlier, critical paths are design dependent. They can consist strictly of combinational logic. In many cases, the combinational paths are paths from input pads to output pads. The cumulative propagation delays of all elements in the critical paths can be determined using the Timer. The following examples of combinational paths do not involve asynchronous feedback.

Internal Combinational Paths

To analyze this type of critical path, first set up a Startset and an Endset. The Startset contains all input pins that will initiate the changes through the critical paths. The Endset contains all input pins of macros at the end of the combinational paths. Based on the specified Startset and Endset, the Timer will report delays in all paths between these two sets.

Figure 2 shows examples of internal combinational paths. In this circuit, A and B pins of gate U0 as well as A and B pins of gate U1 are specified to be start pins. The D pins of U5 and U6 are specified to be end pins. The Timer reports delays in all paths between the start pins and end pins. The delay breakdown of each path and the fanout of each element in the path are also reported.

Input Pad to Output Pad Combinational Paths

Combinational paths from input pads to output pads are special paths where the start pins are input pads and end pins are output pads (see Figure 3). As previously stated, the output pins of output buffers are the only non-input pins that may be specified in the Timer. The Timer references the PAD pins of the input buffers as start pins and the PAD pins of the output buffers as end pins.

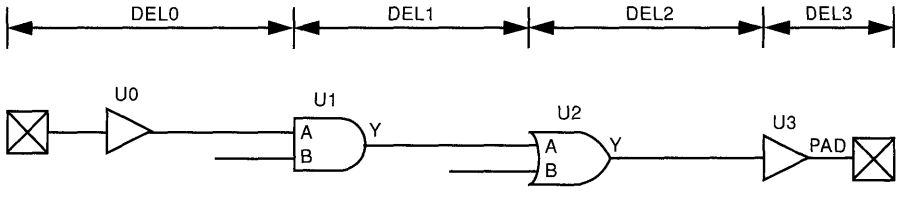


Figure 1. Net Delays

An example of this type of path is a decoder, where inputs of the decoder come on chip via input pads and outputs of the decoder exit via output pads. The Timer automatically puts all input pads in a set named Inpad and it also puts all output pads in a set named Outpad. Use Inpad and Outpad as the current working sets to analyze the timing of these paths. Use the breakdown in delay elements that make up the path for delay optimization. The fanout of each element and the type of macro is listed.

Sequential Paths

The design's critical paths can also consist of sequential paths. Sequential paths are paths that include sequential elements such as flip-flops and transparent latches. Sequential paths must be broken into segments for timing analysis. They are input pads to clocked macros, clocked macros to clocked macros, and clock pads to output pads. In most synchronous designs, one of these three segments will determine the maximum operating frequency of the device.

Input Pads to Clocked Macros Paths

The input pads to clocked macros paths are data signals that come onchip via the Actel FPGA input pads, propagate through the logic, and arrive at the gated inputs of other clocked macros. These data signals have to be stable some period of time before the clock edge arrives at the clocked macros. This period of time is the intrinsic setup time of the clocked macros, which is specified in the datasheet. The Timer presents these data paths, including the setup time of the clocked macros, in the longest to shortest order. Figure 4 shows several typical data paths. As previously stated, the Timer includes all input pads in a set named Inpad and all gated inputs of clock macros (all inputs of flip-flops or transparent latches except clock inputs) are included in a set named Gated. Simply change the current working Startset and Endset to be Inpad and Gated respectively and list the propagation delays for these paths. The clocked macros' setup times are automatically included in these paths.

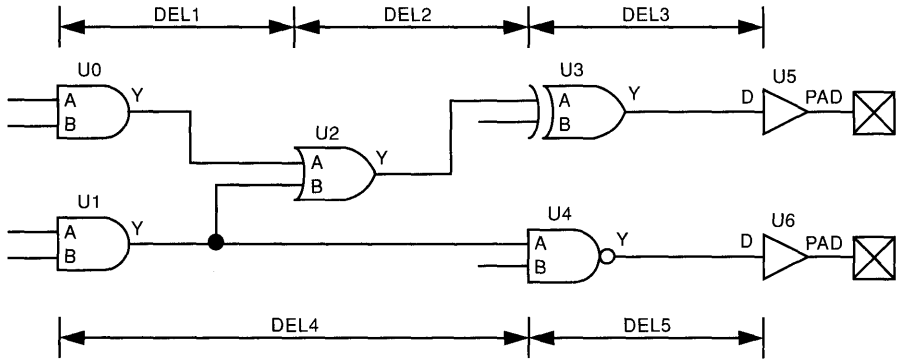


Figure 2. Internal Combinational Paths

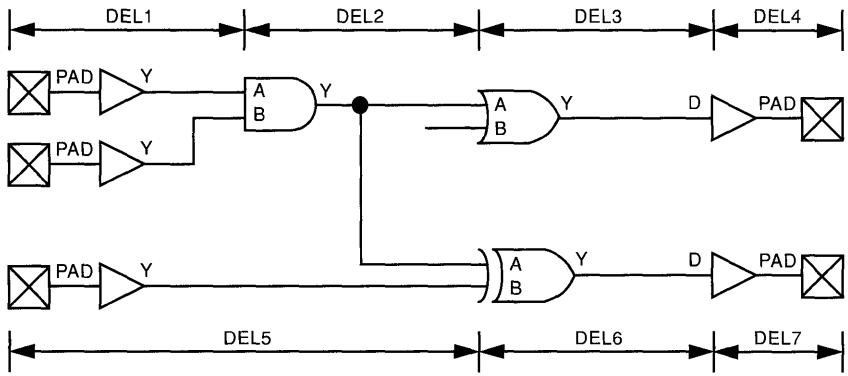


Figure 3. Pad to Pad Combinational Paths

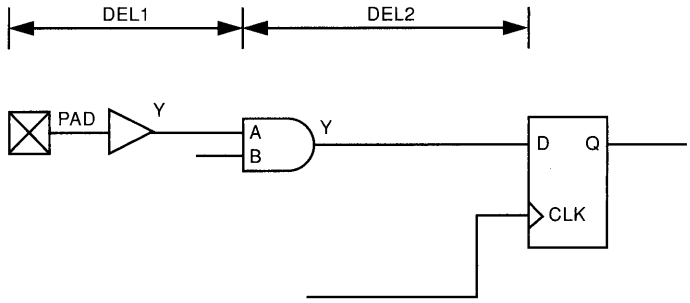


Figure 4. Input Pads to Clocked Macros Paths

Clocked Macro to Clocked Macro Paths

The paths between clocked macros and clocked macros are sometimes the design's critical paths (see Figure 5). These paths begin at the clocked inputs (CLK or G) of the clocked macros and terminate at the gated inputs of the clocked macros. In cases like counters or state machines, these paths actually feed from the outputs back to the inputs after propagating through several levels of logic. Similarly, the Timer includes the setup times of the clocked macros so that the maximum frequency can be easily determined.

The Timer facilitates the timing analysis of these paths by saving all clock pins in a set named Clock and saving all gated pins in a set named Gated. In this example, all CLK pins are saved in the Startset Clock and all data and enable pins of sequential-type macros (flip-flops and latches) are saved in the Endset Gated. Use these sets to investigate all paths between clocked macros easily. Again, the clocked macro's setup times are included in the report.

Clock Pad to Output Pads Paths

The clock pad to output pads paths are slightly different in the source and destinations. These paths begin at the external clock pin and terminate at the output pads. These paths go through the clock pad, pass through the clock macros, and exit the device via the output pads. In many cases where the output pads are parts of a bus, there will be a small output skew between these output pads. Therefore, use the slowest output pad as the limiting path. To analyze the clock pad to output pads paths, create a Startset containing the clock pad and use the default set Gated as the Endset. Figure 6 shows some paths starting at the clock pad and ending at the output pads. The Gated set is automatically loaded by the Timer as the working Endset. For these paths, simply list all paths between the sets and take the slowest path into consideration. In many pipelined designs, the clock pad to output pads paths turn out to be the longest paths and are the design's critical paths.

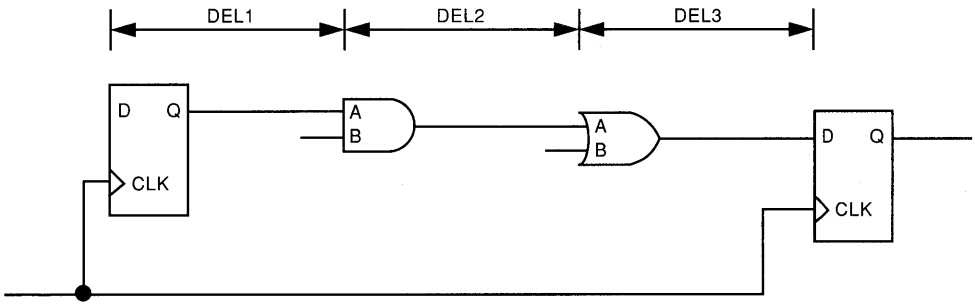


Figure 5. Clocked Macros to Clocked Macros Paths

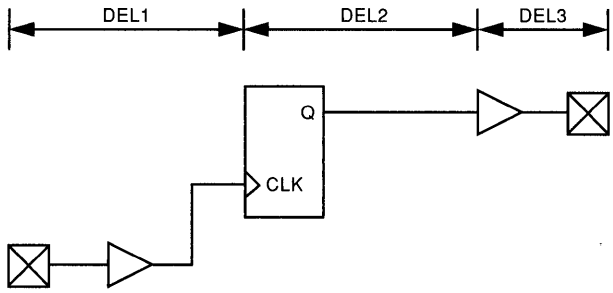


Figure 6. Clock Pad to Output Pads Paths

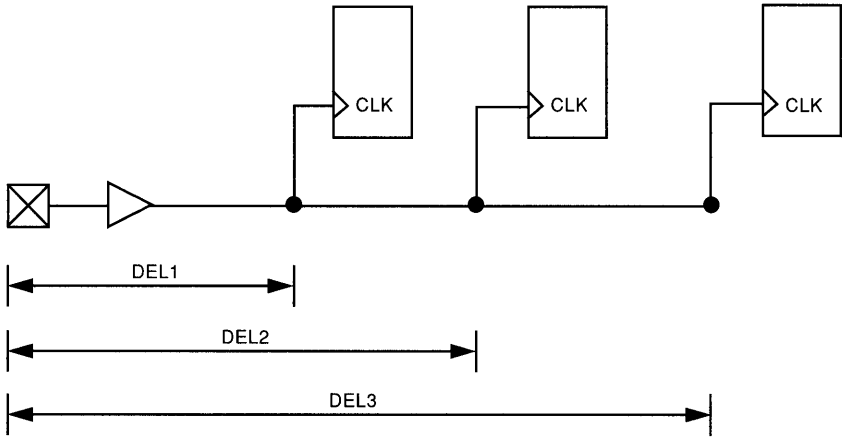


Figure 7. Skew Time

Setup and hold times and skew time are also important. Be aware of these characteristics to ensure that the device works reliably in production. The device setup time is the time that data signals must be stable before the clock transition. The device hold time is the time that data signals must be valid until after the clock transition. It is important to understand that the device setup and hold times are not the flip-flop setup and hold times given in the datasheet. The device setup and hold times are design dependent, and they must be characterized to guarantee that the Actel FPGA will work in sync with other devices on board.

Device Setup and Hold Time

The device setup time can be easily determined by comparing the input pads to gated macros data paths with the clock pad delay. The difference in propagation delays between these two paths is the device setup time. It is possible that this delay is a negative number. The setup time is the minimum required time that data have to be stable before the transition of the clock signal; in this way, data can be stable long before the transition of the clock signal. The hold time can be determined similarly by comparing the data paths with the clock pad delay.

Skew Time

Skew time is important in synchronous designs. The skew time is the difference in arrival time of clock signals at the clock inputs of commonly clocked macros. Ideally, the skew time must be as close to 0 ns as possible. A good example is a serial shift register. If the clock skew is such that the clock edge arrives late, wrong data will be shifted in the register. The skew time is especially important in designs using a normal input pad for clock purposes. A normal input pad is not buffered internally like a clock pad. The input pad used for a clock signal may introduce additional delays in the paths and result in a larger skew time. Using the input pad for clock signals may result in a skew time of more than 2.5 ns, which is the maximum skew time of the clock network. This skew time must be added to the critical path timing to ensure that the design will work reliably.

The skew time may be determined by comparing the longest clock path to the shortest clock path. The difference in delay of these two paths is the skew time. List all paths between the clock pad and all clock inputs of clocked macros and calculate the difference between the delays to the clock inputs of adjacent clocked macros. This skew time should be added to the overall design's critical path for the result to be valid.

Multichip Post-Layout Simulation Using ALS and Viewsim

Application Note

System designs are typically divided into functional modules that are implemented by several Actel devices. To check the functionality of the system, it is very important to simulate all of the Actel devices together. The Actel Action Logic® System (ALS) is capable of multichip simulation with many common simulators. Here is an example of multichip post-layout simulation using Workview®/Viewsim® 4.1 with ALS 2.2 on a PC.

This example requires the use of the Workview-Viewfile utility to set the project directory and to switch to different projects. The top-level system design schematic is **mltchip**, which contains two components, **chip1** and **chip2**. These components represent two different Actel devices in the system. There are three subdirectories, chip1, chip2, and mltchip, in the c:\designs directory. Each of these subdirectories has **sch**, **sym**, and **wir** subdirectories. The chip1 and chip2 directories must contain all ALS generated design files required for post-layout, backannotated timing simulation. Figure 1 represents the directory structure for this design. Note that the names written in normal text represent file names and those in bold text represent directory names. This example contains only single-sheet schematics for each design. Similar procedures apply to multisheet schematic designs as well.

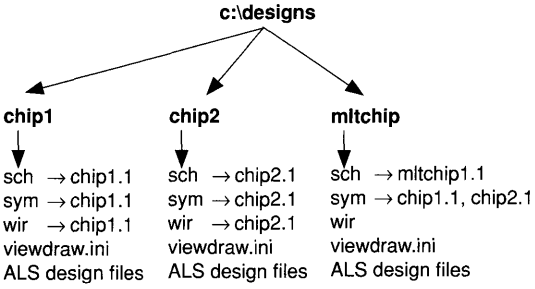


Figure 1. Required Directory Structure for Multichip Simulation Using Viewsim

Use the following procedure for multichip simulation of Actel designs:

1. Go to the c:\designs\chip1 directory. Its sch and wir subdirectories must have all schematic and wir files for chip1. Run the following two commands from this directory:

   del2vl chip1

   vsm chip1 -w -d chip1.dtb

   The del2vl program creates chip1.vsm and chip1.dtb files in the c:\designs\chip1 directory. The vsm program creates a flattened wir file chip1.1 in the same directory. At this point, there is a nonflattened wir file in c:\designs\chip1\wir and a flattened wir file in c:\designs\chip1.
2. Repeat step 1 for chip2 by running the following commands from the c:\designs\chip2 directory:

   del2vl chip2

   vsm chip2 -w -d chip2.dtb
3. Copy each **flattened** wir file (chip1.1 from c:\designs\chip1 and chip2.1 from c:\designs\chip2) to c:\designs\mltchip\wir directory.
4. Select Export->Wirelist->Viewsim from the menu of the mltchip Viewdraw schematic window. This will place mltchip.vsm and mltchip.1 files into the c:\designs\mltchip and c:\designs\mltchip\wir directories. Both of these files contain backannotated delay information. At this point, open a viewwave window to observe signal waveforms for the design.
5. (Optional) Copy the schematic files chip1.1 and chip2.1 from c:\designs\chip1\sch and c:\designs\chip2\sch to c:\designs\mltchip\sch. Then each schematic can be observed with backannotated simulation values from mltchip schematic.

WARNING

To run multichip simulation again for the same design, first remove the design schematic files (chip1.1 and chip2.1) from c:\designs\mltchip\sch directory. Then follow the procedure from step 4. Otherwise, Export->Wirelist->Viewsim (step 4) will create nonflattened vsm and wir files for chip1 and chip2, which will not contain backannotated delays.

Board Level Post-Layout Simulation Using ALS and QuickSim II

Application Note

Introduction

System designs are typically divided into functional modules, which are implemented by several Actel devices. To check the functionality of the system, it is very important to simulate all of the Actel devices together. The Actel Action Logic® System (ALS) includes board-level, multichip simulation capability using Mentor Graphics® software for Actel devices.

The software requirements are identical for chip-level and board-level simulations. The requirements are Mentor Graphics' Design Architect (DA), Design Viewpoint Editor (DVE), QuickSim II, and Actel's ALS software. Note that ALS version 2.2 supports Mentor Graphics V8.2 on Sun and HP700 platforms.

Create the Design

Consider the board-level design, *board*, which contains the chip-level components *chip1* and *chip2*. Each design is composed of a symbol and a schematic. The board schematic includes both *chip1* and *chip2* symbols, which are ACT™ 1 and ACT 2 family designs respectively. The following procedure is shown in Figure 1.

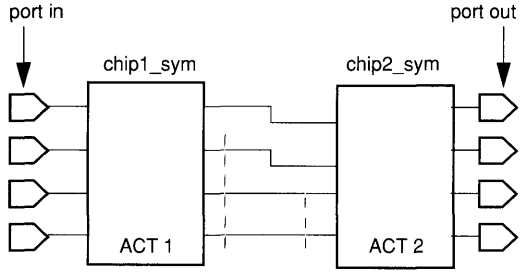


Figure 1. Board Schematic Sheet

1. Create the *chip1* design, including all I/O pads and ports. Execute check sheet and save the design.

Open a symbol sheet to create a symbol for *chip1* with its I/O pins. Add a model property with the name, *als_technology*, and a property value, *act1* or *act2*. Execute check sheet and save as chip1. If the message

WARNING

Property "als_technology" on the symbol is not on the interface.

appears, make no modifications to the design. This warning message does not indicate any problems with the symbol and will no longer appear after the symbol is saved. Verify by rerunning check sheet after saving the symbol.

2. Run mgc2adl on the design to generate the Actel netlist by typing the ALS command

   mgc2adl fam:<*family name*> chip1

   where <*family name*> = *act1* or *act2*.

3. Use ALS to layout and extract post-layout delays for the device. Backannotate these delays to QuickSim II by typing the ALS command

   del2mgc chip1

   This step creates the backannotation file, *chip1.bao*.

4. Repeat steps 1 through 4 for the chip2 design.

5. In Design Architect, create the schematic sheet, *board*. From the popup menu, select

   Instance > Choose Symbol

   Use the Navigator to select and place *chip1* on the schematic sheet. Repeat this procedure for chip2. Connect the symbols and add Portins and Portouts to the schematic. Execute check sheet and save changes. Open a new symbol sheet and create the board symbol. Execute check sheet and save the changes to disk.

DVE Configuration

Next, configure the Design Viewpoint Editor (DVE) to setup the design with backannotated post-layout delays.

1. From the *board* directory, invoke DVE for the design as follows:

   dve <*viewpoint file name*>

   If <*viewpoint file name*> is not specified, DVE uses the default viewpoint file, *default*.

2. Select File > Open > Sheet from the menus. Enter the board level schematic name, *board*.

3. Select the *chip1* and *chip2* symbols and execute Report > Object > Long. An Object window opens and lists the path contexts for the symbols, including references and property information. Verify that the path contexts are of the form

   .../I$1 , .../I$2, etc.

   Remember the path contexts for future reference. Close the Object window.

4. Verify that the Design Configuration window is open. If it is not active, select File > Open > Viewpoint. Verify that the *board* viewpoint is selected.

5. From the pop-up menu, select Add > Primitive to open an ADD PRIMITIVE window. The primitive name is *model* with values of ACT 1 and ACT 2. The primative type is *string*.
6. Select File > Back Annotation > Import from the pull-down menu to open an IMPORT BACK ANNOTATION window. In the ASCII BA field, enter the path to the *chip1.bao* file. In the import context field, add the path context of *chip1*. Save the results. Repeat these steps for *chip2*.
7. Use the Design Viewpoint window to find a numbered list of connected, backannotated objects. Objects are listed by number from highest to lowest priority.
8. To activate the backannotation objects, use the pull-down menu to select

   File > Back Annotation > Connect
9. Save the viewpoint information and exit DVE.
10. Invoke QuickSim II with backannotated delays by entering the command

    quicksim. -tim max <*viewpoint file name*>

    The board-level design is ready for simulation with post-layout, backannotated delays from ALS.

Board-Level Simulation Using ALS/OrCAD

Application Note

Introduction

Board-level designs typically include Actel devices generated with Actel library modules along with other vendors' devices. It is important to simulate and verify the function of the entire board-level design before it is assembled. What follows is a simple example of board-level simulation using OrCAD™ 386+ with the Actel Action Logic® System (ALS) version 2.2 on a PC.

Using Board-Level Simulation

This example uses TTL 74xx library cells and Actel ACT™ 1 library cells. It includes two Actel ACT 1 family devices, chip1 and chip2, that have been created in the \ORCAD\CHIP1 and \ORCAD\CHIP2 directories. Refer to Actel's *OrCAD CAE Guide* for specific configuration and setup information. The board-level design is called "board". It contains the chip1 and chip2 devices and several TTL 74xx components as shown in Figure 1.

In the chip1 and chip2 designs, both user-created sheetpath parts and sheet symbol parts can be used as building blocks of the top level design as shown in Figure 2.

Use the following procedure to create and simulate the board level schematic board:

1. Use the Design Management Tool to create a design called "board". Use the ACT 1 template as a source to copy onto the board using the Design Management Tool. The new directory is \ORCAD\BOARD. The directory tree is shown in Figure 3.
2. Copy all the schematic files (typically *.sch) in \ORCAD\CHIP1 and \ORCAD\CHIP2 to \ORCAD\BOARD.
3. Copy \ORCADESP\SDT\LIBRARY\TTL.LIB to directory \ALS\LIB\OR\ACT1. The TTL.LIB library is provided by OrCAD as an installation option. To use other libraries, copy the corresponding .LIB files to the \ALS\LIB\OR\ACT1 directory.
4. Create the user library part symbols for chip1 and chip2 using the Edit Library menu and include these parts in USER.LIB (or any other unique name). Make sure the sheetpath points to the schematic of the Actel device. In the example, the sheetpaths are CHIP1.SCH and CHIP2.SCH. Refer to the *Action Logic System CAE Guide: PC OrCAD Interface* for a procedure to create new user library parts.

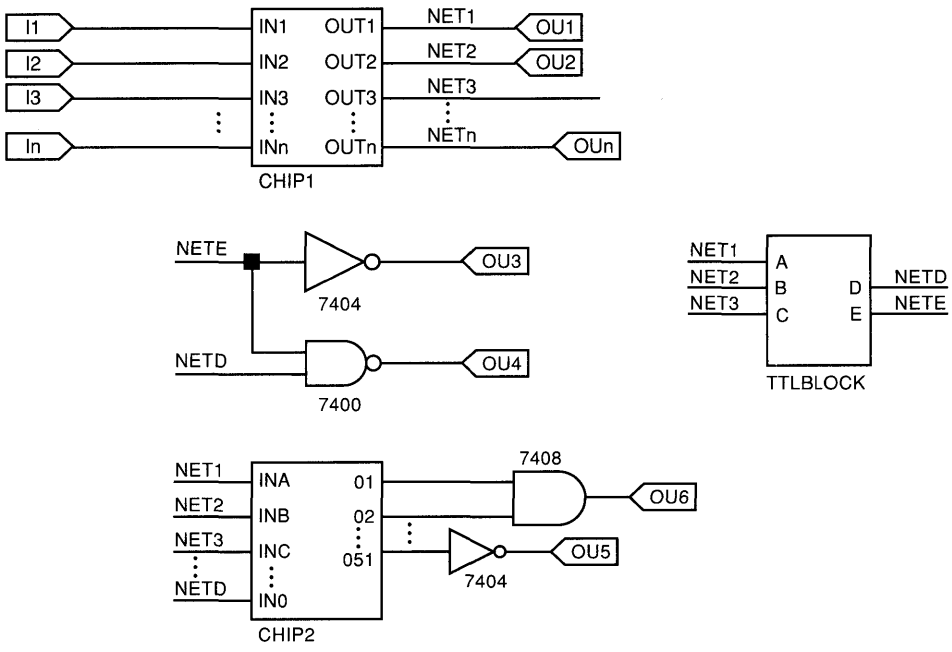


Figure 1. Schematic of Board

5. Verify that the configured library list in Schematic Design Tools includes the appropriate Actel library, any user-created libraries, and any other libraries (ex. TTL.LIB) to be used in the board design.
6. Create the schematics for the design. After the schematics are created, run the Cleanup Schematic, Annotate Schematic, and Check Electrical Rules programs.
7. Next, run To Digital Simulation to enter the simulation window. Verify that the local configuration of To Digital Simulation has the correct configurations for INET as specified in the *CAE Guide*.
8. To simulate the board, it is necessary to add the appropriate simulation models to the OrCAD simulation database. For this example, the Add Device Model program adds simulation models for Actel ACT 1 family and TTL 74xx devices. The TTL.DSF file must be copied to the \ALS\LIB\OR\ACT1 directory.

Please note:

- For each board design, verify that the correct Actel device family library has been configured for netlisting and simulation. For example, substitute "ACT 2" for "ACT 1" in the procedure for board level designs using ACT 2 devices.
- The local configuration for INET must include all component libraries represented in the board-level schematic. Otherwise, the simulation window issues the error message.

  <component> in <library>.LIB not found. Delete Device or EXIT Simulator?
- For ALS version 2.2, only functional simulation is available for board-level simulations and only chip-level designs may have post-layout delays backannotated for simulation.

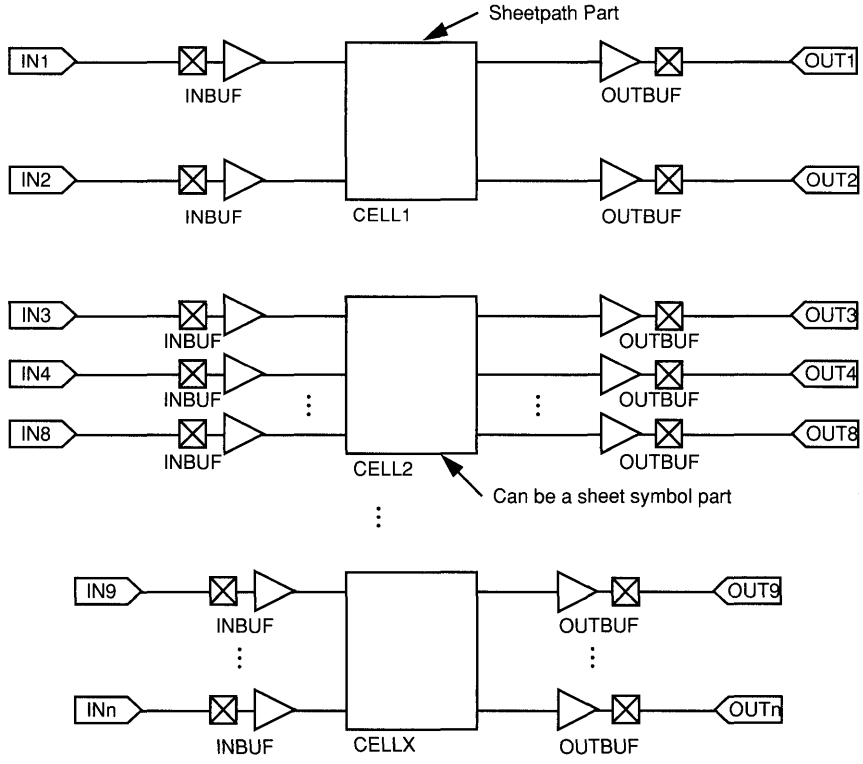


Figure 2. Schematic of Board

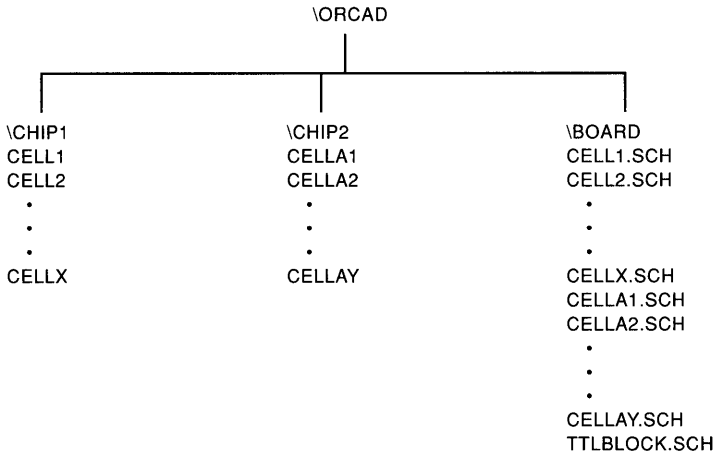


Figure 3. Directory Structure

External Probe Pin Control for Actel FPGAs

Application Note

Introduction

Each ACT™ device includes built-in probe circuits to observe and analyze any internal signals in a design. The probe circuits temporarily route signals from any two internal nodes to two special function pins on the device, PRA and PRB. These circuits are accessed and controlled in three ways: with the Actel Debugger software, the Actionprobes®, and user-created external control. The Actel Debugger and Actionprobe require an Activator programmer to enable the probe circuits. To use the probes without an Activator, you must create external probe control signals that emulate the Activator.

ACT Device Probe Circuits

The ACT devices have three special function input pins to control the probe circuits: serial data in (SDI), data clock (DCLK), and mode selection (MODE). Internally, the devices are arranged into arrays of logic modules of various numbers of rows and columns as shown in Table 1.

Table 1. ACT Device Cell Arrays

Device	Rows	Columns
A1280	18 (R0–R17)	82 (C0–C81)
A1240	14 (R0–R13)	62 (C0–C61)
A1225	13 (R0–R12)	46 (C0–C45)
A1020	14 (R0–R13)	44 (C0–C43)
A1010	8 (R0–R7)	44 (C0–C43)

The logic module arrays are surrounded by two sets of shift registers for the two probes plus a MODE register to determine the operating state of the device. Note that the MODE shift register is independent of and not related to the MODE pin. Figure 1 is a simplified representation of an ACT logic array.

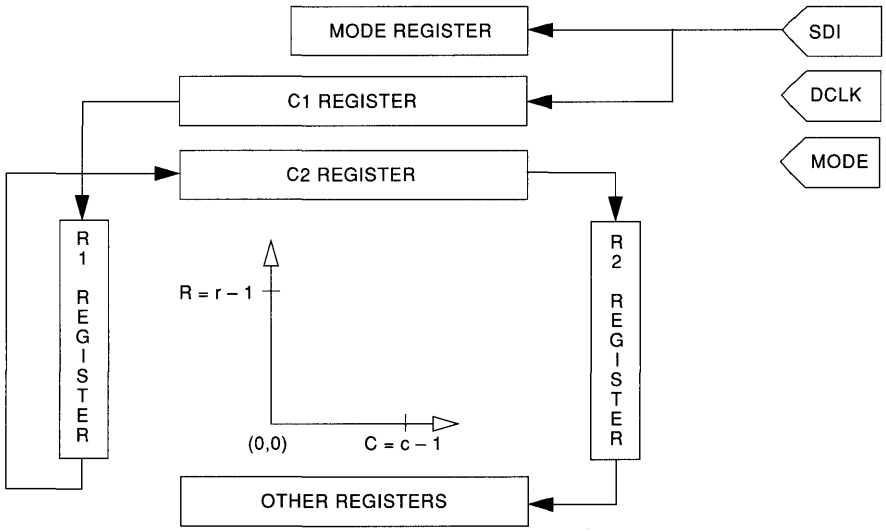


Figure 1. Shift Registers for ACT Devices

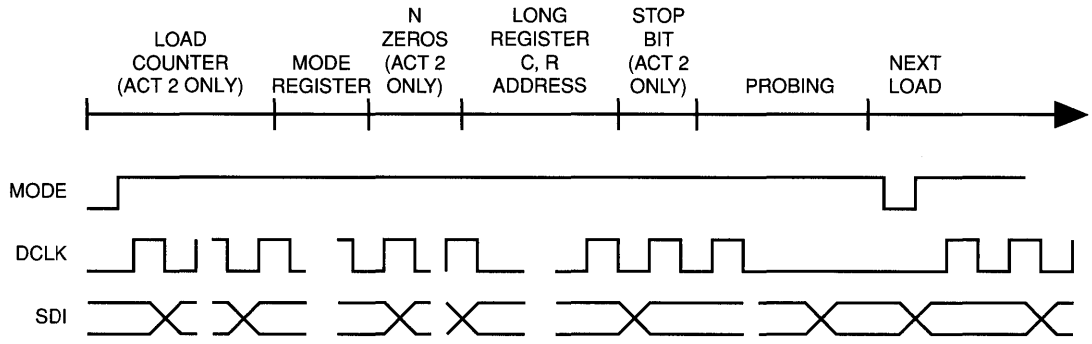


Figure 2. Timing Sequence

If the probes are to be used, avoid assigning input or bidirectional I/O macros to the PRA and PRB pins. Do not assign any user I/O signals to the SDI and DCLK pins. In addition, do not program the security fuses because they will disable the probe circuits.

External Probe Circuit Control

To activate the probe circuits, set the MODE pin to logic high. Then enter the address of the desired logic module at the SDI pin along with the appropriate mode control pattern. The desired signals will be temporarily routed to the PRA or PRB pins. Use the following procedure to externally control the internal probe circuits:

1. Connect V_{PP} and V_{SV} (ACT 2 only) to V_{CC}.
2. Connect V_{KS} to GND (ACT 2 only).
3. Set the MODE pin to logic 0 upon power up.
4. Find the X,Y array location of the desired signals
5. Set the MODE pin to logic 1.
6. Load the pattern in Figure 2 with the SDI and DCLK pins.
7. Observe the waveform(s) at the probe pin(s).
8. Repeat steps 3 through 7 for other signals to be probed.

MODE should be pulsed to logic 0 before other signal locations are loaded into the probe circuits. Pulsing MODE also initializes the test/programming circuits. The chip returns to normal operation when MODE returns to zero. DCLK is a falling edge triggered clock for ACT 1 devices, rising edge triggered for ACT 2 devices.

Table 2 describes the shift patterns for the probe circuits addresses. The waveforms in Figure 2 show the timing sequence for the probe control signals, MODE, SDI, and DCLK. Addressing for the row and column locations is active high; unselected rows and columns have logic 0 input addresses. Specify timing at a maximum of 10 MHz for DCLK, with 10 ns for setup and hold times for the SDI signal. Allow a minimum of 30 ns for the desired signal to arrive at the PRA or PRB pin, valid after the last address clock edge. The row and column addresses (X,Y coordinates) are located in the <design name>.loc design file.

Table 2. Probe Circuit Shift Patterns

Register	Bits	Comments
Counter	10 Bits	ACT 2 only
Mode Register	21 Bits (ACT 2) 7 Bits (ACT 1)	See Table 3 See Table 4
Filler Zeros	n Bits	See Table 4, ACT 2 only
R2	r Bits	Bit_0 ... Bit_r-1
C2	c Bits	See C2 order for chip
R1	r Bits	Bit_0 ... Bit_r-1
C1	c Bits	See C1 order for chip

R1, C1 = Row/Column coordinates for Probe_A
R2, C2 = Row/Column coordinates for Probe_B

Table 3. ACT 1 Mode Register Shift Patterns

Probe Mode	Mode Register Pattern (M6...M0)
Select probe A (PRA)	1001001
Select probe B (PRB)	1001010
Select probe A and probe B (PRA and PRB)	1001011

Table 4. ACT 2 Mode Register Shift Patterns

Probe Mode	Mode Register Pattern (M20...M0)
Probe A	00000011000111110000 0
Probe B	00000010100111110000 0
Probe A & B	00000011100111110000 0

Table 5. ACT 2 Device Counter Pattern

Device	Filler (n)	Counter Pattern	Clocks/Load
A1280	443	0011011111	675
A1240	361	1111000001	541
A1225	308	1101011010	458

Note that the row and column addresses for ACT 2 devices must be entered in the order specified by Tables 6, 7, and 8. ACT 1 device addresses are entered in sequential order

Table 6. A1280 C Register Order

Register	Bit Order
C2	< 80, 81, 78, 79, 77, 76, 74, 75, 73, 72, 70, 71, 69, 68, 66, 67, 65, 64, 62, 63, 61, 60, 58, 59, 57, 56, 54, 55, 53, 52, 50, 51, 49, 48, 46, 47, 45, 44, 42, 43, 41, 40, 38, 39, 37, 36, 34, 35, 33, 32, 30, 31, 29, 28, 26, 27, 25, 24, 22, 23, 21, 20, 18, 19, 17, 16, 14, 15, 13, 12, 10, 11, 9, 8, 6, 7, 5, 4, 2, 3, 1, 0>
C1	< 1, 0, 2, 3, 5, 4, 6, 7, 9, 8, 10, 11, 13, 12, 14, 15, 17, 16, 18, 19, 21, 20, 22, 23, 25, 24, 26, 27, 29, 28, 30, 31, 33, 32, 34, 35, 37, 36, 38, 39, 41, 40, 42, 43, 45, 44, 46, 47, 49, 48, 50, 51, 53, 52, 54, 55, 57, 56, 58, 59, 61, 60, 62, 63, 65, 64, 66, 67, 69, 68, 70, 71, 73, 72, 74, 75, 77, 76, 78, 79, 80, 81>

Table 7. A1240 C Register Order

Register	Bit Order
C2	< 60, 61, 58, 59, 57, 56, 54, 55, 53, 52, 50, 51, 49, 48, 46, 47, 45, 44, 42, 43, 41, 40, 38, 39, 37, 36, 34, 35, 33, 32, 30, 31, 29, 28, 26, 27, 25, 24, 22, 23, 21, 20, 18, 19, 17, 16, 14, 15, 13, 12, 10, 11, 9, 8, 6, 7, 5, 4, 2, 3, 1, 0>
C1	< 1, 0, 2, 3, 5, 4, 6, 7, 9, 8, 10, 11, 13, 12, 14, 15, 17, 16, 18, 19, 21, 20, 22, 23, 25, 24, 26, 27, 29, 28, 30, 31, 33, 32, 34, 35, 37, 36, 38, 39, 41, 40, 42, 43, 45, 44, 46, 47, 49, 48, 50, 51, 53, 52, 54, 55, 57, 56, 58, 59, 60, 61>

Table 8. A1225 C Register Order

Register	Bit Order
C2	< 44, 45, 42, 43, 41, 40, 38, 39, 37, 36, 34, 35, 33, 32, 30, 31, 29, 28, 26, 27, 25, 24, 22, 23, 21, 20, 18, 19, 17, 16, 14, 15, 13, 12, 10, 11, 9, 8, 6, 7, 5, 4, 2, 3, 1, 0>
C1	< 1, 0, 2, 3, 5, 4, 6, 7, 9, 8, 10, 11, 13, 12, 14, 15, 17, 16, 18, 19, 21, 20, 22, 23, 25, 24, 26, 27, 29, 28, 30, 31, 33, 32, 34, 35, 37, 36, 38, 39, 41, 40, 42, 43, 44, 45>

Example: The bit stream shown in Figure 3 probes the outputs of an AND gate with instance name U1 and an OR gate with instance name U2 in an A1240 device.

First, the coordinates must be determined. The following data is from the *<design_name>*.loc file. Note that X corresponds to column address, and Y to row address.

USE ; U1;

XY: 59%5.

USE ; U2;

XY: 58%10.

Since two locations will be probed, PRA and PRB are both used. PRA probes the output of the AND gate (U1), and PRB probes the output of the OR gate (U2). The R and C registers are assigned as follows:

R1 = 5, C1 = 59

R2 = 10, C2 = 58

Use Table 6 to determine the data for the C1 and C2 registers. For example, when C2 is 58, a *one* is placed in the third location of the C2 register, and the remaining locations are zeros.

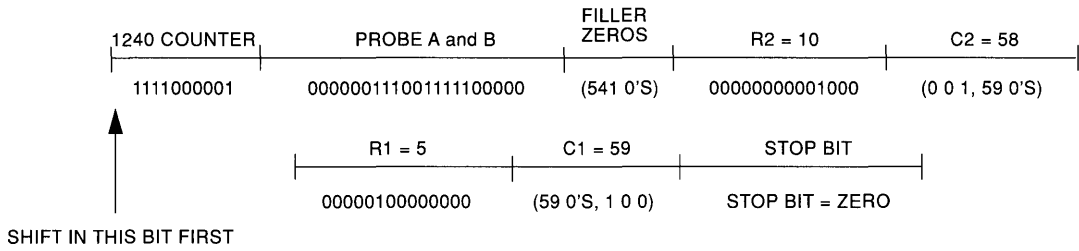


Figure 3. Example Shift Pattern

Using Actionprobe® Diagnostic Tools

Application Note

Introduction

Actel's probe pin circuitry permits external monitoring of ACT™ device internal signals after device programming. This unique diagnostic feature allows 100 percent observability of a device. Observability reduces the time required for design verification and test vector generation; it also facilitates system troubleshooting. One hundred percent observability of all internal device signals is unique to Actel field programmable gate arrays (FPGAs); this feature is not available in conventional masked gate arrays or programmable logic devices.

Two dedicated probe pins, PRA and PRB, provide this observability on ACT family devices. Actel's Actionprobe® software and Actionprobe diagnostics hardware permit the connection of any two signal nodes on the device to the probe pins. Signal node assignments may be changed freely under software control.

Setup

The Actionprobe hardware for the Activator® 2 and 2S programmer consists of a diagnostic pod that connects to the programmer by cable. The pod is connected to dedicated test points in the target system using several flying lead connections. The Activator 2 programmer supports up to four Actionprobe diagnostic pods. The device is verified and debugged in the target board as it receives real-time stimuli from the system.

The Activator Programmer drives the control signals SDI, DCLK, and MODE by a flat ribbon cable. The MODE pin determines whether the device is in debug mode. SDI receives the serial addresses of the internal nodes from the Activator. DCLK clocks the serial address into the device. When the device is being debugged in-circuit, SDI, DCLK, and MODE may be terminated to ground through a resistor greater than 10kΩ. Probe pins may be connected directly to a logic analyzer or oscilloscope.

In-Circuit Probing

Changing the signal nodes is done simply by changing node names with the Debugger software. The newly assigned signals are connected automatically to the probe pins. The internal signal passes through an inverting buffer before reaching the probe pin. Use the Debug/ICP command from the APS 2 menu to initialize this function.

Probe Calibration

The probe circuitry does not introduce any additional loading to the design, so the AC characteristic of the observed internal nodes remains unchanged. And, because probe propagation delay is independent of layout, probe delay remains unchanged for all points in the device.

The skew of the probe pins can be measured, then used to calibrate the accurate measurement of propagation delay. When both probe pins are assigned to the same point on the device, the measured delay difference is the skew of the probe pins. This skew is subtracted from subsequent delay measurements in the circuit. In Figure 1, for example, both PRA and PRB are connected electrically to node Net0. The delay difference is the skew, calculated as

$$t_{SK} = t_{PRA} - t_{PRB}$$

Using the Debugger software, the slower probe (PRA) is assigned to node Net3. Figure 2 shows this configuration. In this example, actual propagation delay is the measured delay between the output of G0 and the output of G3, minus the probe skew time. Actual delay is calculated as:

$$t_{PD} = t_{PRA} - t_{PRB} - t_{SK}$$

Note: Because of the difference in propagation delay between rising and falling signals, both probe pin signals must be either rising or falling when they are used to calibrate delay measurements.

Pin Assignment for Dedicated I/O Pins

During device verification, dedicated pins SDI, DCLK, PRA, and PRB are used as special I/O pins. These pins should be assigned to uncritical user-defined I/O signals so that the design is functional without them. After device verification, programming the security fuses disables the probe pins to prevent unauthorized device probing.

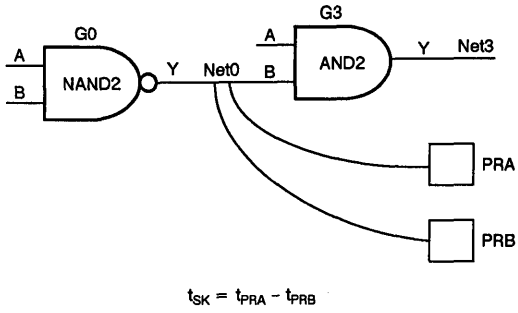


Figure 1. Measuring Skew of Probe Pins

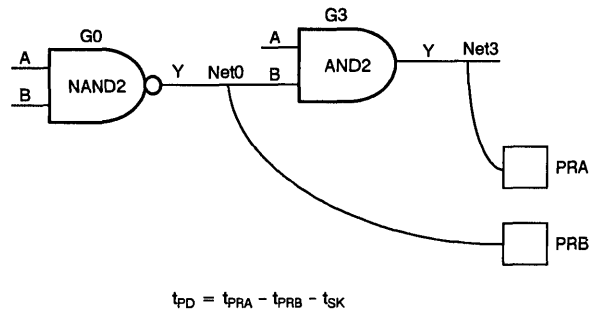


Figure 2. Calibrating for Accurate Propagation Delay Measurement

Using the Actel Debugger as a Functional Tester

Application Note

Introduction

Actel's Activator® programming and diagnostics unit, together with the Debugger software, provide powerful tools to functionally test an ACT™ device. Device debugging begins after design configuration and device programming. Debugging is performed with the device inserted in the Activator programming module.

A Debugger functional test allows the observation of any internal node or external pin of the device. Device inputs are defined with Debugger menu commands, with a command file, or with a combination of the two. Command files and test vectors are created with an ASCII text editor, then loaded into the Debugger. User-defined macros may be created in a command file, then executed in the Debugger.

This application note shows Debugger commands for a sample design. The sample design is Actel's TA269 (TTL 74269), an eight-bit binary loadable up/down counter with count enable. Figure 1 shows the sample design; Table 1 shows the truth table for the part; and Figure 2 shows a typical Debug command sequence. P0 through P7 are parallel load inputs; Q0 through Q7 are counter inputs; CLK is the counter clock; and UD is the up/down counter selector. Internal nodes (nets) should be labeled during design capture for easy reference during debugging.

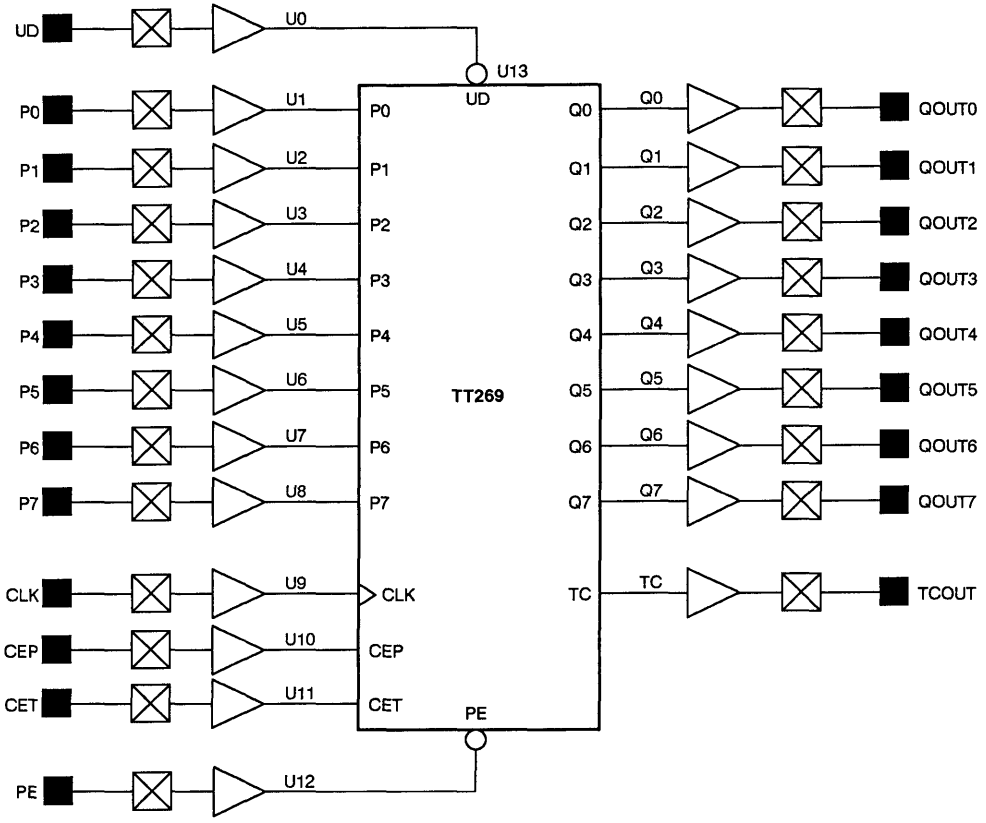


Figure 1. Debugger Sample Design with TA269 Soft Macro

Table 1. TA269 Truth Table

Inputs						Outputs	
PE	CEP	CET	UD	P[7:0]	CLK	Q[7:0]	TC
0	X	X	X	0	↑	0	1
0	X	X	X	FF	↑	FF	0
1	X	1	X	X	↑	Hold	
1	1	X	X	1	↑	Hold	
1	0	0	1	X	↑	Increment	
1	0	0	0	X	↑	Decrement	

Assigning Test Vectors

The default input radix for all text vectors is decimal. To specify a binary, hex, or octal radix, add a 0b, 0h, or 0o prefix, respectively, to the vector (for example, 0b1010 or 0h7e). Outputs are in binary format. To interactively define input test vectors, use the Debugger menu. As an alternative, use input command files to define test vectors. To view outputs and internal nodes, print them to the screen display or to an output file.

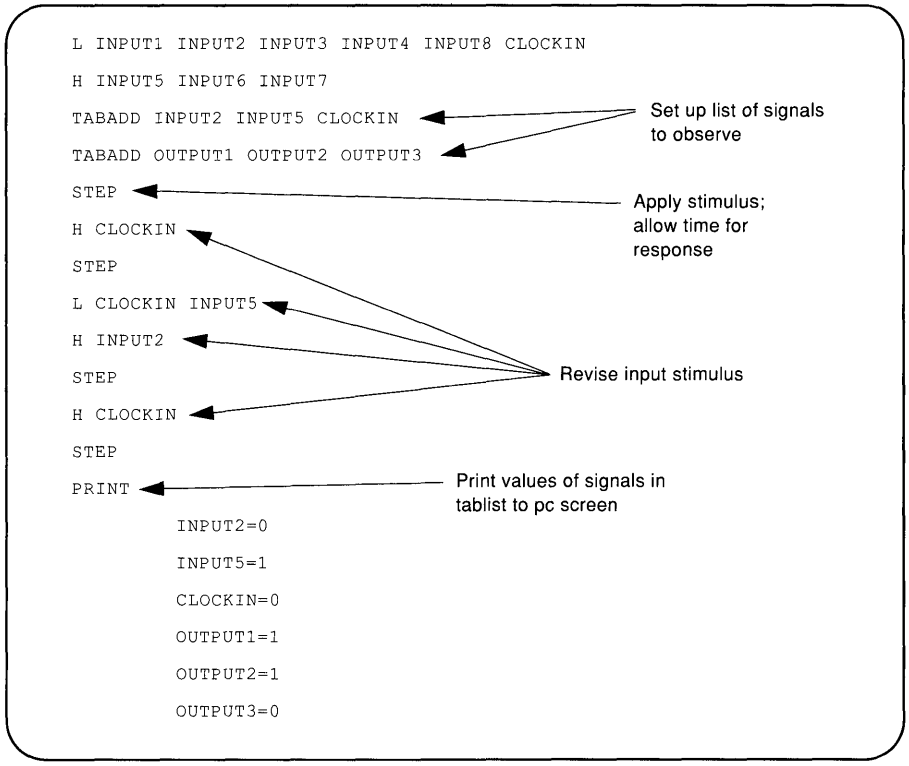


Figure 2. Typical Debug Command Sequence

Defining User Macros

You may save time by creating user-defined macros for the Debugger. These macros may contain a series of basic Debug commands or may nest any combination of basic commands and user-defined macros.

The sample macro in part A of Figure 3, `clk10`, provides 10 clock pulses to the pin CLK and prints the value of internal vector Q to a specified output file after each clock pulse. The `outfile` command specifies the output file.

Part B of Figure 3 shows a nested macro, `clk100`, that executes the `clk10` macro 10 times, providing 100 clocks to the CLK pin.

A

```
define (clk10) (repeat 10 (1 CLK) (step) (h CLK) (step) (fprint Q))
```

B

```
define (clk100) (repeat 10 (clk10))
```

Figure 3. Sample Command Macros

Creating a Command File (`TA269.cmd`)

The command file (see Figure 4) applies test vectors to the `TA269` counter. It redirects results to an output file, `TA269.out`, and compares the output vector Q of the counter to an existing results file, `TA269.cmp`. The following notes correspond to each line in the command file.

Lines 1 and 2: The `vector` command defines eight parallel load input bits as vector P and counter output as vector Q.

Line 3: The `tabadd` command defines the internal or external nodes to be displayed or printed when the `print` or `fprint` command is executed.

Lines 4, 5, and 6: The `infile`, `outfile`, and `compfile` commands define input and output files. The `infile` command opens a file containing input test vectors. The `outfile` command contains the output results. The `compfile` command contains the data to be compared against the current device status. Use the full path name of the file, and enclose it in quotation marks.

Line 7: The `define` commands create user-defined macros. In this example, the `clk10` macro provides 10 clock pulses to the CLK input, `fprint` prints all nodes in the `tabadd` command to a file defined by `outfile`, and fcomp compares the status of vector Q to the file specified by `compfile`.

Lines 8, 9, and 10: Defines three user macros, `up`, `down`, and `load`.

Loading a Command File

The `loadfile` command loads a defined command file into the Debugger. Select `setup | loadfile` from the Debugger menu, then enter the full path name of the command file. For example:

```
/design/TS269/TA269.cmd
```

Executing User-Defined Macros

To execute any previously defined macros, type the macro at the command line while in the Debugger.

```
Line
  1       (vector P P0 P1 P2 P3 P4 P5 P6 P7)
  2       (vector Q Q0 Q1 Q2 Q3 Q4 Q5 Q6 Q7)
  3       (tabadd PE CEP CET US P CLK Q TC)
  4       (infile "/designs/ta269/ta269.pat")
  5       (outfile "/designs/ta269/ta269.out")
  6       (compfile "/designs/ta269/ta269.cmp")
  7       (define (clk10) (repeat 10 (1 CLK) (step) (h CLK) (step) (fprint) (fcomp Q)))
  8       (define (up) (1 CEP CET) (h PE UD) (step) (clk10))
  9       (define (down) (h PE) (1 CEP CET UD) (step) (clk10))
 10       (define (load) (1 PE CLK) (step) (fassign P) (h CLK) (step))
```

Figure 4. Debug Command File (`TA269.cmd`)

Running the Sample Command File

In the following example, APS 2 assigns input test vectors from an input pattern file named `TA269.pat` and writes output results to a file named `TA269.out`. The command file (`TA269.cmd`) compares the counter's output vector Q to an existing results file (`TA269.cmp`).

Output Results File (`TA269.out`)

```
  S     P C C U          P C          Q T
  T     E E E D            L            C
  E       P T              K
  P
00005: 1 0 0 1 00000000 1 10000000 0
00007: 1 0 0 1 00000000 1 01000000 0
00009: 1 0 0 1 00000000 1 11000000 0
00011: 1 0 0 1 00000000 1 00100000 0
00013: 1 0 0 1 00000000 1 10100000 0
00015: 1 0 0 1 00000000 1 01100000 0
00017: 1 0 0 1 00000000 1 11100000 0
00019: 1 0 0 1 00000000 1 00010000 0
00021: 1 0 0 1 00000000 1 10010000 0
00023: 1 0 0 1 00000000 1 01010000 0
00028: 1 0 0 0 10000000 1 00000000 1
00030: 1 0 0 0 10000000 1 11111111 0
00032: 1 0 0 0 10000000 1 01111111 0
00034: 1 0 0 0 10000000 1 10111111 0
00036: 1 0 0 0 10000000 1 00111111 0
00038: 1 0 0 0 10000000 1 11011111 0
00040: 1 0 0 0 10000000 1 01011111 0
00042: 1 0 0 0 10000000 1 10011111 0
00044: 1 0 0 0 10000000 1 00011111 0
00046: 1 0 0 0 10000000 1 11101111 0
00051: 1 0 0 1 01000000 1 11000000 0
00053: 1 0 0 1 01000000 1 00100000 0
00055: 1 0 0 1 01000000 1 10100000 0
00057: 1 0 0 1 01000000 1 01100000 0
00059: 1 0 0 1 01000000 1 11100000 0
00061: 1 0 0 1 01000000 1 00010000 0
00063: 1 0 0 1 01000000 1 10010000 0
00065: 1 0 0 1 01000000 1 01010000 0
00067: 1 0 0 1 01000000 1 11010000 0
00074: 1 0 0 0 01000000 1 01000000 0
00076: 1 0 0 0 11000000 1 10000000 0
00078: 1 0 0 0 11000000 1 00000000 1
00080: 1 0 0 0 11000000 1 11111111 0
00082: 1 0 0 0 11000000 1 01111111 0
00084: 1 0 0 0 11000000 1 10111111 0
00086: 1 0 0 0 11000000 1 00111111 0
00088: 1 0 0 0 11000000 1 11011111 0
00090: 1 0 0 0 11000000 1 01011111 0
00092: 1 0 0 0 11000000 1 10011111 0
```

Input Pattern File (`TA269.pat`)

```
0b00000000
0b10000000
0b01000000
0b11000000
```

Output Compare File (`TA269.cmp`)

```
0b10000000
0b01000000
0b11000000
0b00100000
0b10100000
0b01100000
0b11100000
0b00010000
0b10010000
0b01010000
0b00000000
0b11111111
0b01111111
0b10111111
0b00111111
0b11011111
0b00011111
0b11101111
0b11000000
0b00100000
0b10100000
0b01100000
0b11100000
0b00010000
0b10010000
0b01010000
0b11010000
0b00110000
0b01000000
0b10000000
0b00000000
0b11111111
0b01111111
0b10111111
0b00111111
0b11011111
0b01011111
0b10011111
```

Moving Actel FPGA Designs from Prototype to Production

Application Note

Introduction

Requirements for system designs often change as a project moves from concept to prototype to production. Prototype designs and systems may not be required to conform to specifications that are necessary in the final production release. A system is usually developed faster and less expensively when the constraints for the prototype are relaxed. For example, prototype designs frequently employ "cut and jumper" techniques to quickly modify the connections between devices on a printed circuit board (PCB). While these modifications are effective and useful for debugging and verifying designs, they are unacceptable for the final product. The final product is only produced after incorporating the required changes discovered during the prototyping stages.

Several features are built into ACT™ devices and the Action Logic® System (ALS) software to provide the flexibility and ease of use that enable quick prototyping and transition to production. With these features, early prototypes can be quickly developed and modified to speed engineering schedules. Last minute changes can be made without excessive delay. Then, final production versions of ACT designs are produced without requiring additional, "schedule killing" development cycles.

Prototype Development

Ideally, system designs can be tested and debugged before PCBs are completed for production release. However, PCBs are often finalized and released to production well before the rest of a project is finished. This requires the pin assignments for programmable devices with user-defined I/O pins to be complete before the rest of the design. Since the pin assignments for all field programmable gat arrays (FPGAs) largely determine their performance and routability, it is critical that the pin assignments be as close to ideal as possible even if the rest of the design is not complete.

Fortunately, ALS and ACT devices accommodate this type of scheduling requirement. To finalize the pin assignments early in the development cycle, first complete as much of the internal design (schematic) as possible. Then, let ALS automatically assign the locations of the I/Os. ALS considers the topology of each design to place I/Os in optimum locations. Therefore, approximate the design description (schematics and equations) as close to its final form for the best results. Refer to "How to Select I/O Assignments" in this databook for more information.

After a design has been fully placed and routed, timing analysis with the ALS Timer and backannotated simulation verifies the functionality and performance of the device. At this point, the pin assignments and PCB layout are complete. Small, last-minute design changes could lead to major problems if they resulted in reduced performance or required time-consuming modifications. ALS addresses this issue by including a feature that preserves the timing and performance of a design if small changes are required. If a design has satisfactory performance but less than 5 percent of its netlist must be modified, then the Incremental Placement option of ALS saves the timing of the rest of the design while replacing the modified portions. When this is done, another timing analysis of the design is often unnecessary, saving many hours of verifying and debugging.

Moving to Production

ACT devices are available in a wide variety of die sizes, package types, and performance grades. For maximum flexibility, prototype designs may be developed in device configurations different from the production device. All the "hooks" are incorporated into the design flow seamlessly to enable convenient changes from prototype to production. For example, designs may be prototyped in one package, then put into production in another package effortlessly. Suppose a military project requires ceramic pin grid array (CPGA) packages in the final product. In this case, prototypes may be initially developed in cheaper, plastic quad flat pack (PQFP) packages. When the design is complete, the ALS repackager can automatically transfer a design to another package type. The timing characteristics for the design are completely transferred to the new package without needing further timing analysis.

If a design is transferred to a package type with fewer I/O pins, there may not be a one-to-one correspondence between the I/Os of the two packages. The Action Facts service provides I/O cross-reference guides for equivalent die locations to help solve this problem. For example, if an A1280 design is transferred from a PG176 package (140 I/Os) to a PQ160 package (125 I/Os), some of the I/O pins may not map to bonded die locations of the new device. In this case, ALS cannot automatically repackage the design. However, the cross-reference guides can be used to accurately transfer most of the I/O pins to equivalent locations. Also, use the cross reference guides to prevent any repackaging problems before layout by choosing commonly bonded die locations for all possible packages.

Another feature of the ACT devices involves the use of different speed grades. Each family of ACT devices is available in different speed grades to meet various performance requirements. Initial designs may incorporate cheaper, standard speed grade devices. Development may proceed even if the FPGA speed is inadequate for the particular application. In this case, simply slow the system clock speed until the device performance is adequate. If the standard grade performance is within 25% of the system requirement, then choose the appropriate speed grade to satisfy the needs for increased performance.

Summary

These are only some of the ways that ALS and ACT devices enable a smooth transition from prototype to production. With these tools, PCB procurement becomes less of a scheduling concern. Last minute design changes don't require complete timing analyses to be redone. Package and speed grade options allow a cheaper, faster development cycle without the usual hassles. Throughout the ALS design flow, many other convenient features enable similar time savings and ease of development. Refer to the *ALS User's Guide* for a complete description of ALS programs and capabilities.

Production Programming for Actel FPGAs

Application Note

Introduction

Actel offers several programming options for the ACT™ families of field programmable gate arrays (FPGAs). The Activator® 2 desktop programmer supports all ACT device families. Up to four devices may be programmed simultaneously with the Activator 2. The Activator 2S is a single device desktop programmer with capabilities similar to the Activator 2. These Actel programmers support functional testing and in-circuit debugging with the APS 2 programming software and Actionprobe® diagnostic pod. The Actel programming hardware and software are compatible with 386/486 PC, Sun™, and HP700® workstations. In addition, Actel supports third-party programmers such as the UniSite from Data I/O.

In the early development stages of an Actel FPGA design, single devices are typically programmed at an engineer's desk or in the prototyping laboratory. The Activator 2S is ideally suited for this application with its single module adapter, compact size, and reduced cost. In most cases, the same computer workstation is used to develop, simulate, program, and debug the prototype devices. As the project moves from the engineering development to production phase, the device programming quantities may increase dramatically. The Activator 2 programmers are better suited for this application. The change in programming quantity requires the development of independent programming stations to meet this demand. To build reliable, stand-alone programming stations, several hardware and software issues must be adequately addressed.

Production Programming Work Area

Creating and using a dedicated programming work area is the most effective way to program Actel FPGAs in production-level quantities. Since Actel FPGAs are CMOS devices, appropriate measures are necessary to ensure that programming problems are minimized. Even though all Actel devices have built-in ESD protection, proper handling procedures are still required. All personnel in the programming work area should be properly equipped with antistatic clothing and wrist straps. In general, use standard CMOS device handling procedures for Actel devices.

The Activator programmers should be placed in a clean, well-ventilated location. Since each Activator contains an internal power supply, avoid exposure to heat sources such as direct sunlight or heating vents. Allow adequate ventilation on all sides of the unit including the bottom.

Hardware Requirements

The Activator 2 programmers may be connected to the SCSI port of 386/486 PC, Sun, and HP700 workstations. For Sun and HP700 workstations, use the built-in SCSI port; a SCSI card is included with Activator 2 programmers connected to PCs and Apollo workstations. For these two types of computers, the Activator 2 will only operate with the supplied SCSI card. Other types of SCSI cards cannot be used to program Actel devices. Refer to the appropriate *APS 2 User's Guide* for more information regarding RAM and hard disk requirements.

Activator 2 programmers may program up to four identical devices simultaneously. Each programmer has four adapter ports, which accept adapter modules for all available ACT device packages. There are different adapter modules for each package of each ACT family. Thus, there are different programming modules for ACT 1, ACT 2, and ACT 3 devices in the 84PLCC package.

Software and Design Requirements

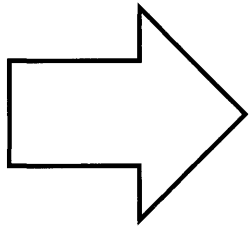
The **aps2** programming software facilitates communication between the Activator 2 and the host computer. This software is included with every Actel software system. The *APS 2 User's Guide* contains complete instructions for its installation and operation.

Many files are created for each design throughout the Action Logic® System design flow. Only two of these files are required for programming devices with the Activator 2 programmer and **aps2** programming software, <design name>.def and <design name>.fus. To archive or transfer the design to another host computer *for programming purposes only*, only the .def and .fus files are required. No other ALS programs may be executed without the balance of the design files. In a similar situation, design files updated from previous versions of ALS using the "**als -update**" command may only be used for programming devices. All other ALS functions are disabled by this program. To update design files for other ALS functions, use the "**als -convert**" program.

Daisy Chained Activator 2 Programming

In general, it is most convenient and effective to use a dedicated host computer to control each Activator 2 programmer. For 386/486 PCs, this is currently the only possible setup: one PC is dedicated to each programmer. However, it is possible to daisy chain more than one Activator 2 programmer to the SCSI bus of a single workstation. Only workstations that have operating systems with independent, multi-tasking windows (for example, X-Windows) can control more than one programmer on the SCSI bus. By default, the Activator 2 is configured for the "one computer/one programmer" setup. If it is necessary to connect more than one programmer to the SCSI bus, an internal resistor pack must be removed and the **aps2** software installation must be modified. For more information regarding this procedure, refer to the Action Facts document, "Daisy Chained Activator 2 Programming."

Designing with Actel Devices



Introduction to FPGA System Design

Introduction

Field programmable gate arrays (FPGAs) are powerful devices for implementing complex digital systems. FPGAs are best used with an understanding of the key differences between FPGAs and previous logic technologies (like PLDs or SSI/MSI). Understanding these differences and using design techniques appropriate for FPGAs result in 50 percent to 100 percent improvement in speed and density compared to design styles that treat FPGAs and PLDs or SSI/MSI equally. This application note identifies the key architectural differences between FPGAs, PLDs, and SSI/MSI, explain design methodologies that result from understanding these differences, and give some simple examples illustrating these techniques in real applications.

FPGAs Compared to PLDs

PLDs are array-oriented devices that typically have an AND-OR structure with wide input AND gates feeding a narrower OR gate. A register is typically available at the output of each OR. This architecture is termed "logic rich" because there are typically many more logic gates than register gates. The ratio can be as much as 5 to 1 logic to register type. PLDs pay a significant speed penalty when multiple levels of logic are required because of the large delay through the wider logic module. Speeds tend to be more predictable in PLDs because of the larger "speed quanta."

FPGAs, on the other hand, are register rich, with a logic to register ratio closer to 2 to 1. (This ratio is equivalent to the traditional gate array usage ratio and tends to be related to the fact that high density designs [more than 1K gates] need more registers than the traditional "glue" oriented low density [less than 1K gates] applications.) FPGA logic structures are optimized for functions narrower than PLDs. FPGAs have a smaller speed quanta than PLDs, so logic functions can be incremented in complexity while incrementing the delay only a little each time. Additionally, signals that need to be fast can be sourced near the bottom of the logic tree, minimizing the number of logic levels required, while slow signals can be sourced at the top of the logic tree, where more logic levels are required.

FPGAs Compared to SSI/MSI

SSI/MSI building blocks are created by optimizing the number of pins on popular functions to fit in the small packages available. Logic functions are typically constructed of a few hundred popular building blocks like counters, multiplexers, shift registers, and comparators. The typical design is optimized to reduce package count, and "tricks" have evolved to make the most use of a device. For example, simple state machines are constructed from counters and decoders with appropriate pins tied to one or zero. This technique minimizes package count compared to a package-intensive gate-for-gate design. The interconnections in these designs are done on the PC board, causing insignificant timing delays.

FPGAs have logic building blocks closer in function to SSI than MSI, so SSI-oriented FPGA designs are generally more efficient. MSI designs, which utilize common tricks (like the counter decoder based state machine mentioned above), do not make efficient use of the FPGA architecture, since MSI building blocks were not developed with FPGA architectures in mind. Even though most FPGA soft macro libraries contain popular MSI functions (like decoders and counters), you should refrain from using the popular MSI-oriented design tricks, since they will result in inefficient FPGA designs.

FPGA Design Techniques

Understanding the main differences between FPGAs, PLDs, and SSI/MSI devices is the first step towards creating efficient FPGA designs. The next step is to understand efficient FPGA design techniques. Techniques will be described by grouping them as state machine, data path, and random logic functions.

State Machine Oriented Techniques

The traditional PLD design techniques for implementing state machines are geared toward the logic rich and register lean architecture of the standard PLD. A small number of state registers are used (usually the theoretical minimum), since registers are scarce. This requires a larger amount of combinatorial logic to decode the state, but PLDs usually are able to provide enough combinatorial logic to do this effectively. Using this technique for FPGAs would not be an efficient use of FPGA strengths—numerous registers and fast narrow logic gates. A bit-per-state approach to state machine design, where each state uses a separate register instead of encoding states in multiple registers, results in faster and more efficient state machines in FPGAs. In many cases, speed improves by 50 percent to 100 percent compared to the PLD-oriented methodology of an encoded state machine.

In PLD-oriented designs, logic is typically used to develop outputs from state machines. Usually this requires an additional level of logic after the state register and adds delay. In FPGAs, this level of logic can be eliminated in many cases by combining the logic in front of the state bits in which an output is active. For example, if the CE output from a state machine needs to be active in states 3 and 5, the logic feeding state bits 3 and 5 can be ORed together and registered to create the CE output without incurring a logic delay after the register. Since the logic in front of state bits is simple, usually no additional delay or logic resources are required in front of the new register.

Another popular state machine design technique for PLDs uses counters to generate a sequence of wait states. For example, a state machine may need to wait for 16 cycles until a data transfer

can begin. A four-bit counter can be used to generate the required state sequence. This is fairly efficient in PLD architectures because of the logic rich and register lean characteristics of the count function. It is not as good a fit for FPGAs, however. In FPGAs, registers are rich and a shift register is more efficient and faster than a counter. A normal shift register will require one register per wait state. If very large delays are required, a feedback shift register can be used that implements only one state less than a counter, but requires much less logic and is significantly faster.

Another useful state machine design technique for FPGAs is state splitting. Sometimes the overall performance of a state machine is limited by a few complex states that require additional levels of logic from all the other levels. If these states could be simplified, the overall speed of the machine could be significantly increased. These states can be simplified by "splitting" the complex state into two or more simple states. This may require an additional clock period to complete the function associated with the original state, but this time may be insignificant compared to the time gained by speeding up the entire machine.

A common state machine design technique with MSI uses a loadable counter to implement a state machine. Load inputs are tied to a jump address (sometimes logic is used if more than one jump address is needed). The counter either counts (to advance to the next state) or loads (to jump to a different state). This is efficient in MSI since it requires only a couple of packages to implement a simple state machine. While this design technique reduces MSI package count, it results in inefficient logic usage for FPGAs. The bit-per-state technique is much more efficient and easier to design when using FPGAs.

Another common inefficient FPGA design technique uses a single large state machine instead of multiple communicating small state machines. In MSI, sometimes a single microcoded state machine controls a complex data path. This works well since large registered PROMs are available to implement a design in a small number of packages. These designs are complicated, however, because each state could have several activities occurring simultaneously, and the interactions between each activity need to be checked in every state. In FPGAs, multiple communicating state machines are easier to design, since most of the communication is local, and only a few activities need to be communicated between different state machines. The distributed machines tend to have much simpler logic requirements that also fit better with the FPGAs register rich, small logic building block characteristics. This approach is also better for FPGA routing because the routing resource requirements are more spread out. Thus, less congestion will occur, making routing more efficient.

Data Path-Oriented Techniques

One of the big strengths of FPGAs is in implementing data path functions efficiently and at very high speed. The register rich architecture combined with the ability to implement multiplexers efficiently make FPGAs ideal devices for implementing data path functions.

FPGAs' register rich architecture allows complex functions to be pipelined easily to improve performance. Pipeline adders, multipliers, and other complex data functions can improve performance considerably over non-pipelined versions. Since registers are readily available at the output of logic, this important technique requires virtually no extra resources. Even functions that wouldn't normally be considered candidates for pipelining should be considered. For example, pipelining can be used in counting, comparing, and code translation when latency isn't an issue. Typically, any portion of the data path where latency can be introduced is a candidate for pipelining.

Some data path functions are less efficient to implement in FPGAs than others, and you should use the more efficient functions as much as possible. As discussed previously, shift registers and feedback shift registers are easily implemented in FPGAs and should be considered for untraditional applications such as address generation for FIFO or buffer memories, positioning in waveform generation, or high-speed event timing or counting.

If counters that are only loaded occasionally are required, prescaling techniques can be used to improve operating frequency. However, these techniques also result in slower load capability. Applications that need to generate long address sequences—for example, memory access—can use this load latency counter very effectively and operate at a higher speed than a nonlatency version.

Nonlatency counters have better performance than MSI equivalents when they are designed using look ahead techniques. These techniques do not impose any additional constraints on the application like the load latency counter, but they take advantage of the register rich nature of FPGAs in implementing counter functions. For example, in a 16-bit down counter, each register should "roll over" after the counter reaches all zeros. Instead of detecting the all zero case by placing combinatorial logic after the counter registers, logic can be placed in front of a register to detect the case when the counter contains a one and is counting down. The register will then be active on the same cycle in which the counter contains all zeros, saving the combinatorial delay associated with the all zero detection. Vendor-supplied soft macros should use this technique to provide users with the fastest possible nonlatency counters.

Adders are other common data path elements commonly used in FPGAs. When pipelining isn't possible, the carry select technique is fastest for implementing combinatorial adders. This technique uses additional logic to produce the two possible results of an addition operation. One result assumes the carry in to a particular bit is active and the other assumes that the carry in to a particular bit is inactive. The actual carry is developed in parallel and is used as the input to a multiplexer that selects the actual result. This utilizes the multiplexer capability of FPGAs to its fullest and implements adders in the smallest number of levels of delay possible. This technique can be generalized to other complex data path functions and shows that logic can be paralleled to effectively increase performance. More logic modules are required over serial approaches, but for speed critical paths it is an excellent technique.

FPGAs can be very efficient at implementing small multiport memories, for example, in algorithms that require scratch pad data storage. Since each register input and output is simultaneously accessible, unlike accessible memories where only single values can be accessed, algorithms that need to access several variables simultaneously can be implemented in a single cycle. If this is the critical portion of a complex algorithm, performance can be increased dramatically over more serial approaches.

Random Logic Oriented Techniques

Many of the techniques mentioned so far also apply to random logic oriented functions. Using parallel logic, feeding the critical path near the end of a logic tree to use the minimum number of logic levels, using registers to predecode and pipeline, and using shift registers instead of counters are all useful techniques for optimizing random logic designs.

Management of fanout is perhaps the most important aspect of implementing high-speed random logic designs. As fanout increases, interconnect delays also increase, slowing performance. Keeping fanout low will almost always result in better performance. The two best techniques for managing fanout are buffering and duplicating logic.

Buffering a design is simply the process of adding buffers to reduce the fanout of a large net. Typically, the additional delay of a buffer is less than the additional delay associated with a heavily loaded net. Thus, buffering results in a faster signal overall. Buffering is also useful when a logic signal is needed at two different levels of a logic tree. The signal closest to the output of the logic tree can arrive later than the signal going into the higher level of the tree. A buffer can be used to isolate the portion of the signal that can arrive later and ensure that the critical signal is as fast as possible. Buffering is also useful if a signal has a local component and a component that needs to travel to a more distant portion of the device. The buffer can isolate the local portion of the signal from the more distant portion so that the local portion does not suffer any speed degradation because of the possible long routing delay for the long interconnect.

Since a buffer uses the same logic module as any other logic function, in some cases it is more effective to duplicate logic instead of buffering. For example, a three-input OR gate that drives a fanout of eight could be duplicated so that each output only drives a fanout of four. (If a particular destination is critical, the load could be split unevenly with the critical signal given the lower fanout.) The inputs to the OR gate now drive one extra load, but the additional delay associated with a single extra load is in almost all cases less than the speedup associated with the logic duplication. This technique is particularly effective when high fanout registers are duplicated, since registers are an abundant resource in FPGAs.

Many of the SSI-oriented tricks designers use for random logic translate directly into FPGA devices because of the similarity of the basic building blocks. You must keep in mind that routing resources are limited inside FPGAs, whereas routing resources in SSI designs on PC boards are virtually inexhaustible. Sections of logic that use too many different clock sources and high fan-in may overly constrain routing. For example, it is usually more efficient to use a synchronous clock source with synchronous enables instead of a large number of individual clock signals to load individually selected data bits into registers because synchronous enable signals have more routing flexibility than clock signals.

Since FPGAs are most efficient at implementing logic at the input of registers, a good rule of thumb in implementing random logic is to use logic at the input of registers instead of the outputs wherever possible. For example, multiplexing a signal prior to a register is more efficient than multiplexing the signal after the register.

Conclusion

This section has shown several design techniques that should help you use FPGAs more efficiently. You should learn and use these techniques to improve the efficiency of your FPGA designs.

Reprinted from PROCEEDINGS OF THE IEEE 1990 CUSTOM INTEGRATED CIRCUITS CONFERENCE, Boston, Massachusetts, May 13-16, 1990

An FPGA Family Optimized for High Densities and Reduced Routing Delay

Mike Ahrens, Abbas El Gamal, Doug Galbraith, Jonathan Greene, Sinan Kaptanoglu
K.R. Dharmarajan, Lynn Hutchings, Sifuei Ku, Phil McGibney, John McGowan, Amer Samie
Kitty Shaw, Norma Stiawalt, Telle Whitney, Tom Wong, Wayne Wong, Bortay Wu

Actel Corporation
955 E. Arques Ave.
Sunnyvale, California 94086

ABSTRACT: The Act-2 family of CMOS Field-Programmable Gate Arrays uses an electrically programmable "antifuse" and new architectural and circuit features to obtain higher logic densities while increasing speed and routability. Improvements include: two new logic modules, novel IO and clock driver circuitry, and more flexible and faster routing paths. New addressing circuitry shortens programming time and speeds complete testing for shorts, opens and stuck-at faults. Fully automatic placement and complete routing are retained. Special software tools used for architectural exploration and layout generation are noted.

1. Introduction.

Previous papers described an architecture for field-programmable gate arrays (FPGAs) [1], and its implementation in the Act-1 FPGA circuits [2]. These demonstrate that user programmability can be obtained without sacrificing the application flexibility of a channeled gate array architecture.

This paper describes new architectural features, circuit techniques and software that approximately double system speeds, and are capable of extending the architecture to logic densities of 8,000 gates in 1.2 micron technology and to approximately 16,000 gates for 0.8 micron. (Note that these gate counts are based on the capacity of an equivalent mask-programmed gate array. Other measures would yield higher values.) The circuits employ a one-time electrically programmable "antifuse" offering small area and capacitance, and low resistance once programmed [3].

As before, the architecture consists of rows of logic modules separated by horizontal channels. This organization is similar to that of a channeled gate array, except that instead of an area for custom metallization the channels contain wiring segments of various lengths which can be connected by antifuses.

A key goal was to insure complete automatic placement and routing with acceptable routing delays. This is facilitated by the inherent flexibility of the channeled architecture and the integration of large numbers of antifuses (700,000 or more) on a single chip.

2. Logic Module

The choice of the logic module is critical to an FPGA architecture. The module must be simple enough to permit a compact and high-speed circuit layout. Yet it must also be flexible enough to accommodate the most frequently used logic functions (macros) with several choices of routing. Our approach is to evaluate many candidate modules against macro usage statistics from actual applications. (The philosophy is similar to that used to define the instruction set of a RISC microprocessor. It has also recently been applied to BiCMOS gate arrays [4].) To assist in this task, a program has been developed that can enumerate all macros accommodated by a given module in minutes [5].

The Act-1 family uses one general-purpose module, which implements all combinational functions of 2 inputs, many of 3 or 4 inputs, and others ranging up to 8 inputs [1]. Any sequential macro can be configured from one or more modules using appropriate feedback routings.

At higher logic densities, the law of averages makes designs begin to adhere more closely to typical macro usage statistics (see, e.g., [4]). This motivates the use of a mix of two new modules, each of which is most efficient for a different set of macros. The "C-module" is a modified version of the Act-1 module reoptimized to better accommodate high-fan-in combinational macros, e.g. wider AND gates, though with some loss in ability to accommodate sequential functions. The S-module, on the other hand, is optimized for configuring sequential macros. It can accommodate a latch or flip-flop and/or many combinational macros of one to seven inputs. Both transparent-high and -low latches and rising- and falling-edge-triggered flip-flops are possible.

The two-module scheme can reduce the number of modules required for a block of logic by up to a factor of 3. On average, logic density per module is increased by over 50%. Furthermore, because the density is increased, the number of routed nets in a typical critical path is reduced. This significantly improves speed. Fig. 1 shows how a typical critical path in a state machine can be implemented to take advantage of the wide fan-in of the C-module, and the capabilities of the S-module. The delay paths include only two routed nets. Performance data is summarized in Table 1.

Since the fan-in of each module is no larger than that of a typical gate-array macro, the two-module scheme maintains

IEEE 1990 CUSTOM INTEGRATED CIRCUITS CONFERENCE CH2860-5/90/0000-0166 © 1990 IEEE

the generality of a "fine-grained" architecture. Significantly larger and more specialized modules would risk a sharp loss of efficiency for applications that deviate from typical usage statistics. Using a larger module, or more types of modules, also adds constraints to the placement and routing problem, making automatic solution more difficult and ultimately increasing net delay.

3. Input/Output

Of particular importance to system performance is the delay between the time a clock signal changes at an input pad and when data appears on an output pad, referred to as T_{clk-Q}. (Memory bus interface applications are a good example). The goal is to gain maximum speed without sacrificing flexibility.

This is accomplished by providing a dedicated transparent-high latch in each output path. If desired, the dedicated latch can be combined with a transparent-low latch configured from a logic module to form a rising edge-triggered flip-flop. (Note that the net connecting the two latches is not in the critical path, so T_{clk-Q} is not increased relative to having a dedicated flip-flop in each IO.) If flow-through operation is desired, the output latch gate is simply tied off to make the latch transparent.

To limit set-up time requirements, a dedicated transparent-low latch is provided on each input path. The polarities of the input and output latches are chosen so they can be combined with each other, and possibly with other internal latches or flip-flops, to form a path that is functionally equivalent to a chain of rising-edge flip-flops. (See Fig. 2.).

Chips with many simultaneously switching outputs require some form of slew rate control to avoid noise problems; several alternatives are possible. Sequencing the operation of several parallel drivers limits the slope of the current ramp when driving a passive load, but large di/dt can occur in bus contention situations when the contending driver suddenly shuts off. Feedback remedies this problem, but can still allow large di/dt in asynchronous systems where the logic state changes before a transition is complete. Instead a current mirror circuit was used to limit the drive current.

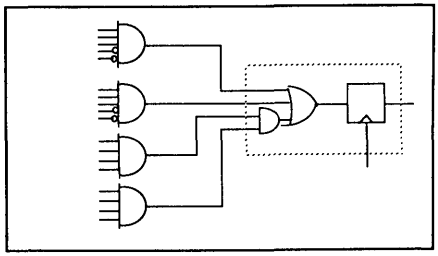


Figure 1: part of a state machine implemented in four C-modules and one S-module.

This results in lower di/dt noise in worst case situations, a simple way to implement programmable slew rate, and 90% power efficiency. The output buffer meets the 4mA HCT buss driver specification for AC, and the 6mA specification in steady state when the current limit shuts off. ESD protection is >2000V.

Connections between the array and the IO pads are made via special IO modules interspersed with the logic modules. The IO module has inputs for data, slew control, tristate, and separate gates for the input and output latches. The gate inputs are not restricted to a dedicated clock signal, but may each be driven from any pad or internal net.

4. Clock Distribution

Clock distribution is a problem in most large chips. In an FPGA, where the load capacitance may be changed or redistributed to suit each application, it is a greater challenge.

Special distribution networks are provided to deliver high-fanout clock signals to the inputs of any logic or IO module with minimal skew. Each network may be driven directly from an input pad for high speed, or from user-defined internal logic. High speed and low power are obtained by a distributed driver with 90% power efficiency.

Skew is further reduced by automatic placement algorithms that balance the loading on each branch of the distribution tree.

All clock inputs may also be routed in the normal way instead, allowing many local asynchronous clock signals if desired.

parameter	nsec
module input to module input (critical net):	7-8
setup+hold time (module used as flip-flop):	< 6
input pad to IO module output:	5-7
IO module input to output pad:	8
clock distribution net skew:	< 5
in-circuit probe delay (module to pad):	15-20

Table 1: Performance Estimates.
(1.2 micron CMOS, typical process, 5 volts, 25 C).

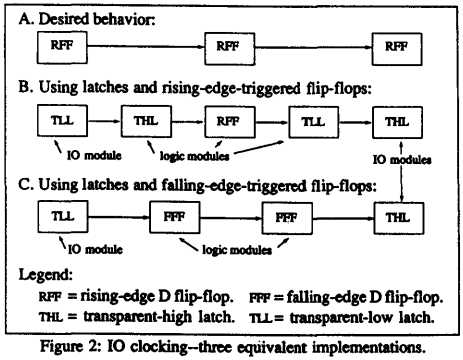


Figure 2: IO clocking--three equivalent implementations.

5. Routing Architecture

Each routing channel contains horizontal tracks divided into segments of various lengths [1]. Surprisingly, the restriction to segments of predefined length does not greatly increase the number of tracks beyond what would be required in the unrestricted case of mask-programmed channels [6].

In an efficient architecture it is inevitable that some nets' routings will be slower than others. The use of a low resistance switch, such as the antifuse, helps to narrow the resulting delay distribution. Further improvements have been obtained by a reduction in the maximum number of antifuses in the worst delay paths, as follows.

In the vertical direction, most nets are routed using a short dedicated segment connected to the module's output driver through an "isolation" transistor (Fig. 3). (The transistor isolates the module circuitry from programming voltages present on the segments). In this case, there are only two antifuses plus the isolation device in the path from the buffer to each input (input A in the figure).† Though this favorable routing can be assured for speed critical nets, generally some 5-10% of the other nets must be placed with an input in some channel beyond the span of the dedicated segment (input B, Fig. 3). In the past, this required use of an uncommitted vertical segment and 4 antifuses.

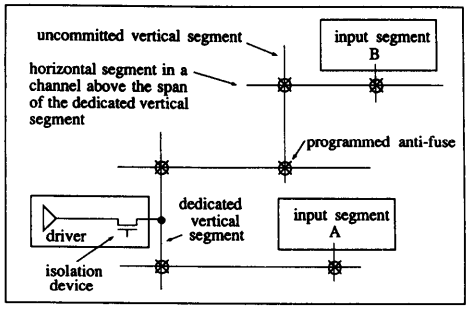


Figure 3: Act-1 Routing

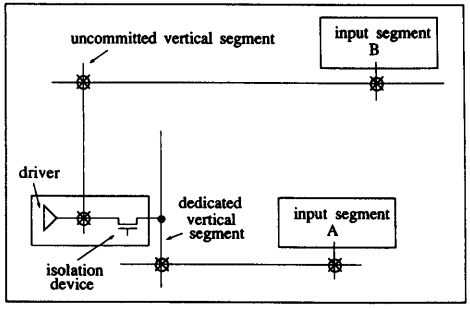


Figure 4: Act-2 Routing

Alteration of the order in which antifuses are programmed and a robust driver circuit permit limited programming of antifuses on the node connecting the driver to the isolation device, without risk of device breakdown [7]. This allows direct connection of the driver to any of several uncommitted vertical segments, as shown in Fig. 4. Since the additional antifuse presents little more resistance than that of the bypassed isolation device, the delay of these nets is not much greater than those using dedicated segments. Prediction of delays prior to placement (when it is not yet known which nets require uncommitted segments) becomes more accurate as well.

Segmented channels represent an unusual layout challenge. They are as dense and large as a memory array, yet not repetitive. (A carefully chosen but irregular mix of segment lengths is provided for good routability.) For this reason, a layout generation program was developed that assembles the channels and modules automatically from the same database used by the routing software. This permits rapid layout of a family of arrays of various sizes by simply rerunning the generator with the appropriate input files.

6. Placement and Routing Software

Several new complexities are added to the placement optimization problem. Macros must be placed in modules of the appropriate type (C or S). Macros hooked to a clock network should be distributed so as to balance the load on the network's branches. There should not be excess demand for uncommitted vertical segments within the same column. Speed critical nets should be routed using only short horizontal segments and dedicated vertical segments.

Nevertheless, new algorithms make it possible to satisfy all these constraints. Nearly all designs with module utilization under 85%, and most designs with utilization under 95%, route without manual intervention. Table 2 summarizes results for several applications. Time for complete placement and routing is about 45-60 minutes on a 68030-based workstation.

7. Programming and Testing

The time required to program an antifuse falls exponentially with the applied voltage. To keep programming time under 5-10 minutes for a chip with nearly a million antifuses, new circuit designs were developed that eliminate the threshold voltage drop along the path from the chip's supply pad to the antifuse being programmed.

Changes have also been made in the addressing circuits. The pass transistor scheme described in [1] is appropriate for cases where there are many short segments in a track.

† Two adjacent horizontal segments in the same track may be connected end-to-end by an antifuse to form a longer segment [1]. For good routability it is necessary to route some small percentage of the nets in this way [6], which adds an antifuse to the path. However, speed critical nets are routed without this additional antifuse.

However in larger chips the number of horizontal segments per unit area decreases to the point that it is possible to address each individual segment directly using only a small proportion of area for the addressing circuitry [7]. The reduction in the number of pass devices in the programming path improves the programming current and lowers the resistance of programmed antifuses, improving performance. The pass transistor scheme is still used in the vertical direction where tracks are highly segmented.

Direct addressing also reduces the time required to test for breakdown of defective unselected antifuses during programming. A complete test for unintended connections between any two segments can be done after the conclusion of programming (despite the fact that it is not possible to uniquely address each individual antifuse once programming commences). The number of vectors required is only logarithmic in the number of nets. Previously, the test for shorts required one or more vectors after each antifuse is programmed.

Proper closure of a programmed antifuse is confirmed by the passage of the programming current. Note that this complete testing for shorts and opens, combined with exhaustive testing of each logic and IO module prior to programming, is more thorough than even a so-called "100% stuck-at fault coverage" test done on a conventional gate array.

Once programming and testing are complete, no increase in resistance of a programmed antifuse or false programming of an unprogrammed antifuse have been observed in 1.8 million accelerated burn-in device-hours [8].

8. Other Circuit Improvements

The Act-1 and Act-2 architectures allow user selection of any internal logic signal for presentation at a "probe" pad. This allows real-time external observation of each net as the chip operates in a system (similar to an in-circuit emulator for a microprocessor). Use of a sense amp circuit greatly increases the speed of the in-circuit probe path.

Another challenge is to keep the gates of thousands of isolation devices pumped to a high voltage during normal operation. A rapid, high-power pump operates when the chip turns on. It is then shut down when the desired voltage is reached and a low-power sustainer pump takes over. The required standby current is under 300uA.

design	logic module utilization	pins per logic module
a) design done in an 8K gate TI gate array	99.5 %	4.28
b) 32 bit datapath, 16x16 mult, state machine	99.4	4.30
c) 2901 ALU (x4)	98.1	4.57
d) DMA controller (x3)	97.1	3.99
e) asynchronous serial ECC	97.0	4.78
f) pipelined fixed point mult, div, sqrt	94.5	3.37
g) state mach, mult/add, datapath, counter	92.7	4.34
h) color crt controller (x3)	87.3	3.83
i) 32 bit datapath w/ sum, compare (x3)	86.8	5.25
j) 40 bit floating point add/sub	86.7	4.33
k) 16 bit datapath, 16x16 mult, state machine	98.1	4.86
l) 2901 (x2)	93.2	4.68
m) DRAM, DMA &SCSI controllers, UART	92.6	4.73

Table 2: Place and Route Examples.

Examples routed in possible implementations of the architecture with 1232 (a-j) and 649 (k-m) logic modules. The notation "(xN)" means the block was replicated N times.

Acknowledgements

The authors acknowledge contributions by: Rick Wilkenson (layout); Sam Beal, Andy Haines, Dennis McCarty, Bob Osann (applications); Gregg Bakker, Steve Chiang, Shafy Eltoukhy, Esmat Hamdy, John McCollum, (technology); Sanko Lan, Justin Reyneri (logic) and Jeff Schlageter for his support. Sample designs were contributed by Texas Instruments, Data General, many Actel customers, and the EE 218 class at Stanford University.

References

[1] A. El Gamal, J. Greene, J. Reyneri, E. Rogoyski, K. El-Ayat, and A. Mohsen. "An Architecture for Electrically Configurable Gate Arrays." IEEE J. Solid-State Circuits, Vol. 24, No. 2, April, 1989, pp. 394-398.

[2] K. El Ayat, et. al. "A CMOS Electrically Configurable Gate Array." IEEE J. Solid-State Circuits, Vol. 24, No. 3, June, 1989, pp. 752-762.

[3] E. Hamdy, et. al. "Dielectric Based Antifuse for Logic and Memory ICs." IEDM Tech. Digest, San Francisco, CA, 1988, pp. 786-789.

[4] A. El Gamal, J. Kouloheris, D. How, M. Morf. "BiNMOS: A Basic Cell for BiCMOS Sea-of-Gates." Proc. IEEE 1989 Custom Integrated Circuits Conf., page 8.3.1.

[5] S. Lan, J. Reyneri, J. Greene, A. El Gamal. "An Automatic Function Generator for Field Programmable Gate Arrays." In preparation.

[6] J. Greene, V. Roychowdhury, S. Kaptanoglu, A. El Gamal. "Segmented Channel Routing." Submitted for publication.

[7] A. El Gamal, J. Greene, J. Reyneri. "Programmable Interconnect Architecture." US Patent 4,873,549.

[8] S. Chiang, R. Wang, J. Chen, K. Hayes, J. McCollum, E. Hamdy, C. Hu. "Oxide-Nitride-Oxide Antifuse Reliability." Int'l Reliability Physics Symp., March 1990.

Estimating Actel FPGA Device Capacity

Application Note

Introduction

Estimating the logic capacity of a field programmable gate array (FPGA) can be confusing and tedious for engineers because vendors count gates differently. The simple question is, "Will my design fit into this device?" For some vendors, the answer may be more complicated than might be expected. Fortunately, Actel FPGA device capacities are calculated using the most common method in the ASIC industry. This application note explains how to calculate Actel device capacities and how to determine logic resources for a particular application.

Capacity Estimation

Gate array capacities are commonly determined by the number of equivalent two-input NAND gates in a device and the percentage of usable gates rather than by the number of PLD equivalent gates. It should be noted that a gate array equivalent gate is a two-input NAND gate but a PLD equivalent gate is a more arbitrary value assigned to the device by the manufacturer.[1] The measure of usable gates is the percentage of total gates in a device that may be used in a typical design before exhausting its routing resources. Since Actel device sizes are based on typical gate array equivalents, gate array designers should already be familiar with the density estimation of an Actel FPGA.

Actel determines the gate capacities of its FPGAs by comparing actual designs and library macros with the same logic implemented on a masked gate array. The ACT™ family architecture consists of an array of logic modules with the rows and columns separated by routing tracks. It is more complex than a gate array, but the Actel logic module is much simpler than the basic element of most competing FPGA and PLD architectures. Furthermore, it offers a flexible interconnect technology and no fixed AND-OR planes. Because the gate counts for masked gate array designs can be determined easily, Actel uses the masked arrays as a convenient standard to characterize its devices in gate array equivalents. Based on comparisons with masked gate arrays and on analyses of actual designs, Actel logic module characteristics can be summarized by the following relationships. For the ACT 1 family,

$$\text{logic modules} = (\text{ff} * 2) + (\text{gates}/3.2) \quad (1)$$

and for the ACT 2 family,

$$\text{logic modules} = (\text{ff}) + (\text{gates}/3.4) \quad (2)$$

where ff equals the number of flip-flops. Equation 1 states that the number of ACT 1 modules needed to implement a design is equal to twice the number of flip-flops plus the number of gates divided by 3.2. Equation 2 states that the number of ACT 2 modules needed to implement a design is equal to the number of flip-flops plus the number of gates divided by 3.4. These relationships determine the capacity requirements for a specific design. The gate density factors were obtained by performing density tests on a number of randomly selected customer designs. Table 1 lists the capacities of different device types and their corresponding gate array equivalent gates, PLD equivalent gates, and logic modules. The ACT 1 family consists only of combinatorial modules, while ACT 2 and ACT 3 families consist of both sequential and combinatorial modules. Details of these modules are available in the *Actel FPGA Data Book and Design Guide*. Furthermore, using PREP benchmark measurements clearly shows that Actel capacity estimates are the most accurate of any manufacturer, and that the capacity of Actel devices varies the least among benchmarks, making it easy to determine the device required to implement a particular application.[2]

ACT 3 devices require fewer logic modules than do ACT 2 devices to implement designs because ACT 3 logic resources are enhanced. The ACT 3 S-module implements a more comprehensive combinatorial function than the ACT 2 version, and each ACT 3 I/O module contains input and output flip-flops. These features increase the gate density of the ACT 3 family.

Utilization

The utilization of any device depends on the connectivity of every logic module. For example, connecting a two-input NAND gate with an S-module will give high utilization since only three signals need to be successfully routed. In contrast, connecting a four-input multiplexer with an S-module might decrease the utilization since seven signals must be routed successfully.

Summary

The device sizes of the Actel FPGA are based on typical gate array equivalents. The tools and techniques used for Actel FPGAs are the same as those used in traditional ASIC gate array design. A conservative utilization factor for Actel FPGAs with automatic layout tools would be approximately 80% of total gates, although higher utilizations are frequently achieved. It is easy to determine the device requirements for implementing a specific design because the capacity of Actel devices varies the least among common benchmarks.

References

1. Dennis McCarty, "Interpreting FPLD Gate-Density Data High Performance Systems," *1990 Programmable Logic Design Guide*, pp 14-20.
2. "PREP PLD Benchmark Suite #1, Version 1.2," Programmable Electronics Performance Corporation, March 28, 1993.

8

Table 1. Device types with the corresponding capacities of logic modules.

Device	Capacity		Logic Modules	
	Gate Array Equivalent Gates	PLD/LCA Equivalent Gates	S-Modules	C-Modules
A1010A/A1010B	1200	300	N/A	295
A1020A/A1020B	2000	6000	N/A	547
A1225/A1225A	2500	6250	231	220
A1240/A1240A	4000	10,000	348	336
A1280/A1280A	8000	20,000	624	608
A1415	1500	3750	104	96
A1425	2500	6250	160	150
A1440	4000	10,000	288	276
A1460	6000	15,000	432	416
A14100	10,000	25,000	697	680

Estimating Capacity and Performance for ACT 2 FPGA Designs

Application Note

Introduction

If you are unfamiliar with field programmable gate arrays (FPGAs) you are initially faced with one of two tasks: integrating an existing discrete logic or PAL-based design or starting a new design. Before proceeding with a functional implementation, you must assess how much logic will fit into an FPGA and at what speed it will perform. In addition, you must make these determinations without having a detailed knowledge of the FPGA architecture. This section will address both of these issues specifically for Actel's ACT™ 2 family of FPGAs.

The high performance of ACT 2 FPGAs may be seen in macros from the soft macro library. An A1225A-2 runs a 16-bit loadable counter at 105 MHz, and the A1240A-2 implements eight of these counters at between 66 and 75 MHz (performance is stated at worst-case commercial operating conditions). A 16-bit Accumulator runs at 39 MHz.

Architecture Overview

Fortunately, it is easy to understand the basics of the ACT 2 architecture. The ACT 2 family of FPGA devices comprises rows of logic modules separated by horizontal routing tracks containing metal segments. Routing tracks (metal segments) also run vertically over the logic modules and the horizontal routing tracks. The vertical segments can connect to the inputs and outputs of the logic modules. Connections between logic modules are made by programming two antifuses: one at the intersection of a vertical segment connected to the output of a module and the other at the intersection of the horizontal segment and a vertical segment connected to the input of another module. Most connections are made in this fashion, although other special connections are possible. The ACT 2 data sheet contains a complete description of the routing architecture.

Logic Modules

There are two types of logic modules in the array: the combinatorial or C-module and the sequential or S-module. There are approximately an equal number of each type of module in an ACT 2 device.

Macro Library

Designs based on discrete logic (TTL or CMOS SSI/MSI) are easy to translate into Actel FPGAs. Most common digital functions are provided in the design library as either hard or soft macros. Hard macros utilize either one or two logic modules. Both C-module and S-module hard macros exist in the macro library in more than 100 logic variations. Soft macros are predesigned schematics containing many hard macros. A list of Actel hard and soft macros for the ACT 2 family can be found in the *ACT Family Macro Library Guide*.

Some soft macros, like the TA161, preserve the look and function of standard 74 series TTL logic. These macros retain the suffix of their SSI cousins, but have the TA prefix in place of the 74. Other macros, like the CNT4B, are standard digital functions needed in many designs. These macros frequently provide more functionality than what is offered in a standard TTL digital data book. Soft macros may be duplicated and modified, enabling you to create custom macros to suit individual needs. The Actel hard macro list contains Boolean functions of two, three, four, and five inputs, with most inputs available in either active high or active low versions. Similarly, active high output or active low output versions can be found.

Estimating Device Capacity

A few simple analysis steps determine the required FPGA device capacity when converting an existing circuit board logic into FPGAs.

Discrete Logic Replacement

An estimation of the size of an existing TTL or CMOS design may be determined by using the design's parts list. The data book lists the number of Actel logic modules needed to build each of the soft macros. Multiplying the number of logic modules needed by the quantity of any part used will give the total number of logic modules required for any one function. Consider the following parts list from a TTL design. First, select an Actel equivalent macro for each item on the parts list and note the number of S-modules and C-modules required. Then multiply by the quantity of macros used, and sum the products to get the total number of S-modules and C-modules required.

The total logic module count for this sample design is 150 C-modules and 134 S-modules. From the data book, we can see that the A1225 has 231 S-modules and 220 C-modules. Therefore, this design should occupy only 63 percent of the A1225, of leaving capacity for an additional 20 percent to 30 percent of logic. Extra logic modules may be needed to reduce internal fanout, so you may want to add one logic module for each signal with a fanout greater than eight. Determining fanout in soft macros is covered later.

PAL Logic Replacement

Estimating the number of logic modules needed to replace an existing PAL is also very straightforward. Figures 1 and 2 show an example of typical PAL files converted to Actel logic modules. These two equations are typical PAL functions: address decoding and state machine control. Timing information is included for later use. Figure 1 is a wide-registered AND function decoding 16 inputs and using only five logic modules, or one percent of the smallest ACT 2 part. Figure 2 shows this AND function enlarged to include the decoding of 24 input signals. Sixty-four signals may be decoded and registered in this way using only 21 logic modules and two levels of logic.

Table 1. Converting Sample Designs from TTL to Actel Macros

					Per Macro		Total	
No.	**Part no.**	**Description**	**Quantity**	**Actel Macro**	**S-module**	**C-module**	**S-module**	**C-module**
1	74LS161	4-bit counter	3	TA161	4	10	12	30
2	74F151	8:1 multiplexer	4	TA151	0	5	0	20
3	74LS684	8-bit mag comparator	2	MCMPC8	0	36	0	72
4	74LS377	8-bit register	6	TA377	8	0	48	0
5	74F166	8-bit shift register	8	SREG8A	8	0	64	0
6	74LS74	Dual D flip-flop	9	DFPC	1	0	18	0
7	74F113	Dual JK flip-flop	4	JKF1B	1	0	8	0
8	74F04	Inverter	2	n/a	0	0	0	0
9	74F32	Quad OR gate	2	OR2	0	1	0	8
10	74F08	Quad AND gate	1	AND2	0	1	0	4

Figure 3 is a familiar three-product term AND/OR array using only four logic modules to implement a typical state machine equation. Figure 4 shows the product and sum terms expanded further. In the PAL equation examples, note how easily the number of either product or sum terms can be expanded. A small incremental change in both delay and size is added each time the array is enlarged. This is in sharp contrast to the large step function increase in delay and size when a second PAL array must be used to implement a particular equation.

PAL equation conversion shows the flexibility of the Actel logic module, implementing a variety of different combinatorial and sequential functions in few modules without wasting dedicated resources. With ACT 2 devices, the cost of using a sequential function is equal to the cost of using a combinatorial function. Any logic module can be used for either type of function. In contrast, PALs are rich in AND/OR gating but lacking in registers. Encoding states in a PAL requires using the least number of flip-flops possible, while with Actel, you are free to use any combination of flip-flops and logic. Both examples in Figures 1 and 2 make use of the multiplexed inputs of the S-module. This four-input multiplexer feeds a dedicated flip-flop, allowing both combinatorial logic and registering to be done in a single module. This also improves speed by eliminating the routing delays to the flip-flop.

Gate Array Replacement

Gate array designers should have no trouble estimating density in an Actel FPGA, since device sizes are based on typical gate array equivalents. Each logic module in the ACT 2 family roughly equals six two-input NAND gate equivalents. The tools and techniques used for Actel are the same as those used in traditional ASIC gate array designs. A conservative utilization factor for ACT 2 parts would be 80 percent of total gates, although higher utilizations are frequently achieved.

Determining Intrachip Timing

Actel FPGA designers should keep these factors in mind: the target internal clock frequency and the operating temperature, voltage, and speed grade. These factors are used to determine average fanout and levels of delay allowed.

Scaling

The operating conditions are used to determine the proper delay scaling required. Data book delays are provided at worst-case commercial conditions (TA = 70 °C, V_{CC} = 4.75 V, and worst-case processing). Derating multipliers and interpolation graphs are provided to scale or derate the data book delays to worst-case industrial or military values. CMOS devices, such as those manufactured by Actel, are affected by both temperature and voltage. In addition, device speed will vary because of process variations. Actel devices are 100 percent tested to ensure that the process variations are within the maximum specification. The interpolation graphs are useful when the device will be operated at nonstandard conditions, such as V_{CC} = 4.6 V and TA = 65 °C.

Speed Grades

Actel offers different speed-grade versions of each part. The faster parts have a suffix of -1 or -2 added to the part number, before the package type. The -1 parts are 15 percent faster and the -2 parts are 25 percent faster than standard speed devices. Speed-grade selection is based on factory testing of delay chains programmed into the device during factory screening.

Understanding Routing Delays

Unlike most FPGAs, the timing and delay information in Actel's data book includes routing delays. Each delay parameter is listed for a fanout matrix. This allows accurate prediction of the performance of any circuit in an Actel FPGA.

Dividing the clock period by the logic module delay determines the average number of logic modules that can be placed between any two flip-flops. Repeating this for several different fanouts gives a good guide to use.

Consider the A1225A-2 running with a 30 MHz clock, over commercial operating conditions. Table 2 gives a guideline to use in creating the schematic. First a maximum number of six modules can be traversed in 33 ns. For a complicated combinatorial section that needs four levels of logic, the maximum fanout should be limited to eight loads. In data path portions of the design with only two levels of logic, fanouts greater than eight can be acceptable. High signal fanouts should be split by duplicating the logic rather than by buffering the output as shown in Figure 5. A single buffer and a four input AND/OR are both implemented using one logic module. Therefore, duplicating the source takes the same amount of resources as adding a buffer on the output, but it uses one less logic module delay and results in a faster design.

When determining fanout, it should be noted that many of the input pins on Actel's hard and soft macros have a load greater than one. An example of this would be a four-bit counter where the outputs are fed back to the inputs to determine the next count. These feedback paths load down the outputs. The *ACT Family Macro Library Guide* lists this pin loading information for macro input. Alternatively, the loading can be manually calculated from the schematics that are part of the hard and soft macro libraries.

Soft macro selection plays a big part in determining the speed of an Actel design. Just as in TTL design, building a large counter by using several smaller counters with ripple carry outputs will lead to a very slow design. Soft macros should be chosen based on speed and density requirements, not on pin-for-pin compatibility to familiar MSI packages.

Each soft macro listing includes the number of levels of logic, (equivalent to number of gate delays) and the number of logic modules required to build each of the macros. This is the best information to use when selecting a soft macro. For example, consider three four-bit counters: the CNT4A, CNT4B, and UDCNT4A. If the counter always counts UP, then the best choice based on speed and density is the CNT4B, which uses only 11 logic modules and four levels of logic.

Determining PAL and State Machine Performance

Figures 1 and 2 used earlier as sample PAL equations also show some typical timing calculations. Here, the external setup for registering a 16-input decoding function is only 1.8 ns. This could be expanded to 64 inputs, which only adds one more level of delay of about 4.9 ns. The external setup time for registering 64 inputs is 6.7 ns.

Figure 3 shows an internal state machine running at 92 MHz. This example can be used to show module delays based on fanout. Each of the state equation inputs is assumed to have a fanout of four, while the other signals have a fanout of one.

Figure 4 shows an expanded state machine. Only the added signals have an increased delay. If these are static inputs, the rest of the circuit will still run at 92 MHz. If these are outputs of clocked registers, then the new clock frequency is 66 MHz for three levels of logic.

Note that additional product terms may be added with either no increase or only an incremental increase in the delay. This allows more flexibility, and makes less impact on changes to existing designs. In contrast, once the product terms are used up on a PLD output, a change requires traveling through an adjacent array term, and the delay jumps to two times the original delay.

Determining Interchip Timing

While internal clock frequencies are determined by logic module delays, system clock frequencies are determined by how fast data may be moved onto and off of the chip. I/O signals are one of two types: combinatorial through the chip or clocked onto and off of the chip. In the following examples t_{DLH}, the worst-case time of driving a CMOS signal high with worst-case commercial conditions will be used.

For combinatorial I/Os, the simple addition of input, internal, and output delays will give the total through chip delay as shown in Figure 6. The data book gives output delays for driving the outputs to both TTL and CMOS levels. For latched I/Os, two options are available: using I/O latches or using the internal flip-flops and latches. Each has its own advantages. Varying the loading of the two global clock circuits may also affect the delays by a few nanoseconds. Figure 7 shows three different options for latching inputs.

Table 1. Module and Routing delay values for A1225A-2, Commercial Operating Conditions

	Fanout = 1	Fanout = 2	Fanout = 3	Fanout = 4	Fanout = 8	
Worst-Case Commercial	4.9	5.5	6.1	6.6	8.2	ns
Levels of logic allowed @30 MHz	6.7	6.0	5.4	5.0	4.0	

External setup and hold times for the input latches are given in the Sequential Timing Characteristics section of the data sheetdata sheets. The times listed, t_{INH} and t_{INSU}, relate the G input of the I/O latch to the data pad. The delay from the external clock to the G input must be included. Figure 7 shows how to calculate the setup and hold times for an external clock with one load. When one of the two global clocks are used with an input latch, the external setup and hold times may be found for a variety of clock fanouts. These externally measured times include t_{INH} and t_{INSU}. The setup time, t_{SUEXT}, is zero ns, and the hold time, t_{HEXT}, is 7 ns for a clock network fanout of 32 loads or less. The input latches offer zero setup time, which allows for very high system clock speeds but a longer hold time. An internal flip-flop or latch may be used for shorter hold times with an external input. The last example in Figure 7 shows this method.

Output latch clock-to-q, t_{GHL} and t_{GLH}, is shown under the Output Module Timing section of the data sheet. Adding the G input delay, whether it is the global clock or an independent clock, gives the total external clock-to-Q delay. Setup and hold times for the output latch are represented in the Sequential Timing Information as t_{OUTH} (= 0.0 ns) and t_{OUTSU} (= 0.4 ns). The first circuit in Figure 8 shows the timing for an output register.

If an output must be driven by a D flip-flop and not a transparent latch, two options are available. One uses the internal S-module and an output buffer. The second builds a master/slave D flip-flop from one internal latch and one of the output latches. This is included in the ACT 2 library as a firm macro, ORH. Timing for both cases may be seen in Figure 8, with the master/slave flip-flop giving the faster clock-to-out time.

When fast clock-to-out is needed off chip, one of the global clocks can be used exclusively for outputs. This will decrease the delay for the clock signal by reducing the fanout.

Estimating System-Level Performance

System-level performance can be estimated after chip-level timing parameters. System-level performance depends on the message passing constraints (FPGA to FPGA, FPGA to PAL, or TTL to FPGA) imposed by the FPGA. For example, in the Actel-to-Actel message transmission in Figure 9, you can expect the maximum FPGA-to-FPGA propagation times to be between 16 and 27 ns, depending on the internal device loading and speed grade.

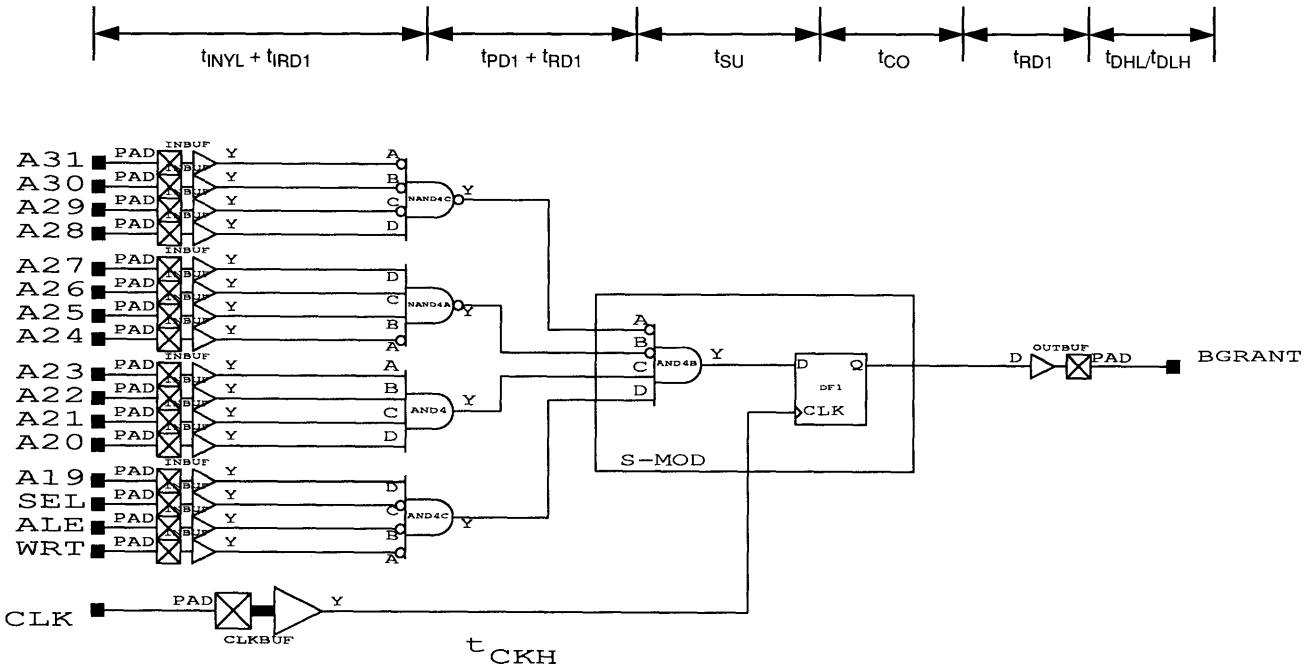
An Actel-to-PAL transmission is shown in Figure 10. The PALs in the circuit are either 15 ns or 25 ns devices. The use of 5 ns PALs will reduce Actel-to-PAL transmission times significantly.

Finally, the Actel-to-TTL transmission example in Figure 11 can be run at from 25 to 35 MHz. This high transmission rate is due in part to the short setup time of TTL F-series registers.

Summary

These techniques will enable you to make tradeoffs between density and speed. The same Actel parts may be used for either high-speed, medium-density designs, or for low-speed, high-density designs. These tradeoffs may also be made within an individual design, tailoring the architecture for your needs.

The information in this section should enable you to predict the size and speed of an ACT 2 design. The size of a design may be predicted using the soft macro library and by translating PAL equations. The speed of a design may be determined by controlling the levels of logic, fanout, and the type of I/O modules used.




```
BGRANT = /SEL * /ALE * /WRT * /A31 * /A30
       * A29 * A28 * /A27 * A26 * A25 * A24
       * A23 * A22 * /A21 * A20 * A19
```

Figure 1. PAL Implementation with Wide Fan-In

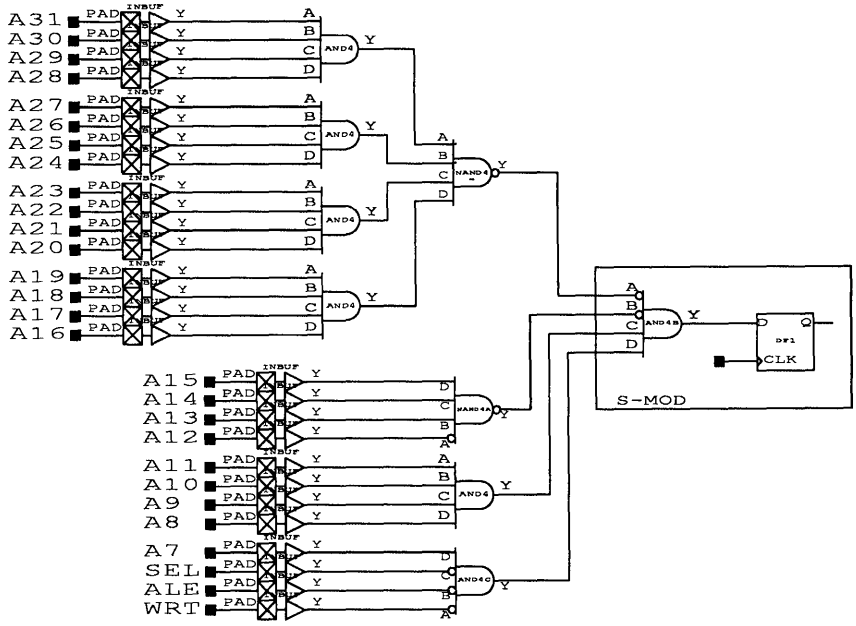


Figure 2. Expanding Fan-In to 28 for PAL Implementation

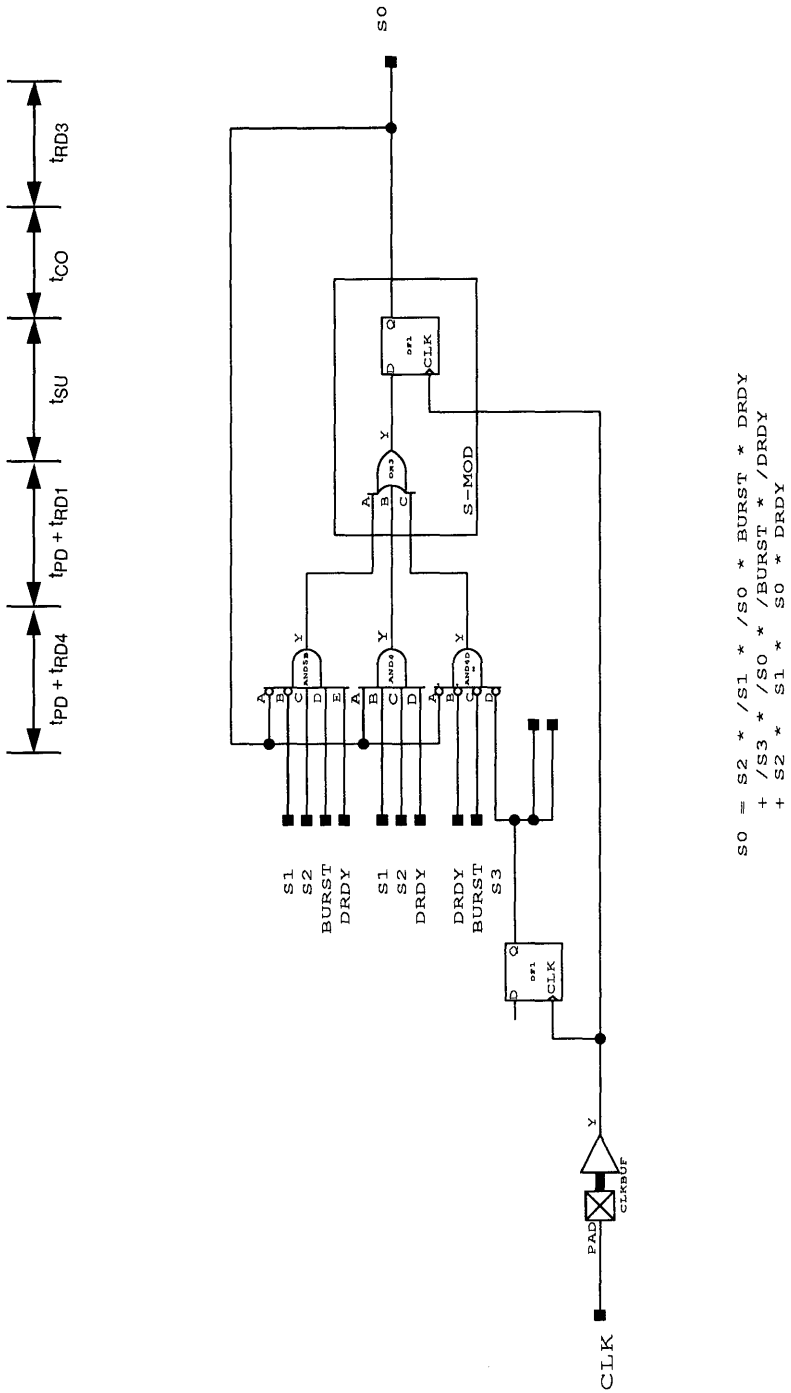


Figure 3. Implementing State Machine

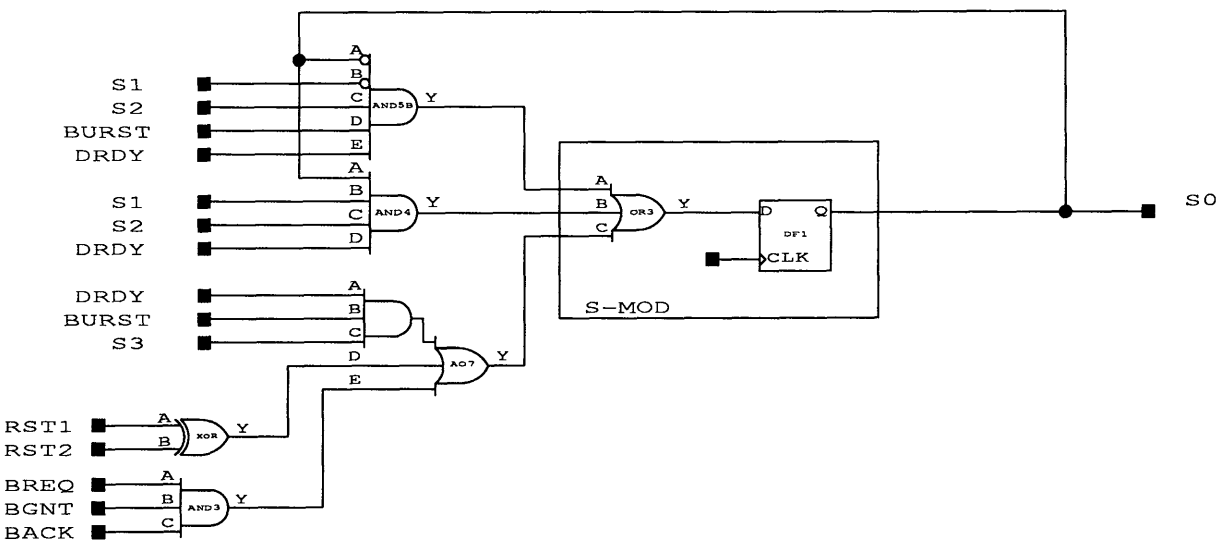
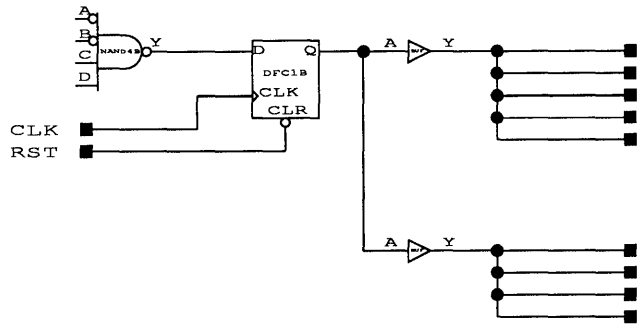
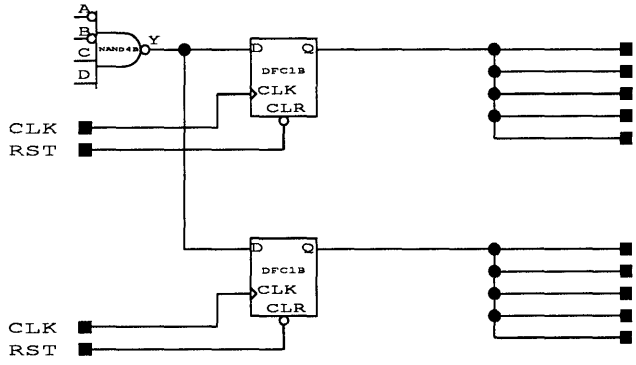


Figure 4. Expanded State Machine



Slower



faster

Figure 5. Redundant Source Versus Buffering

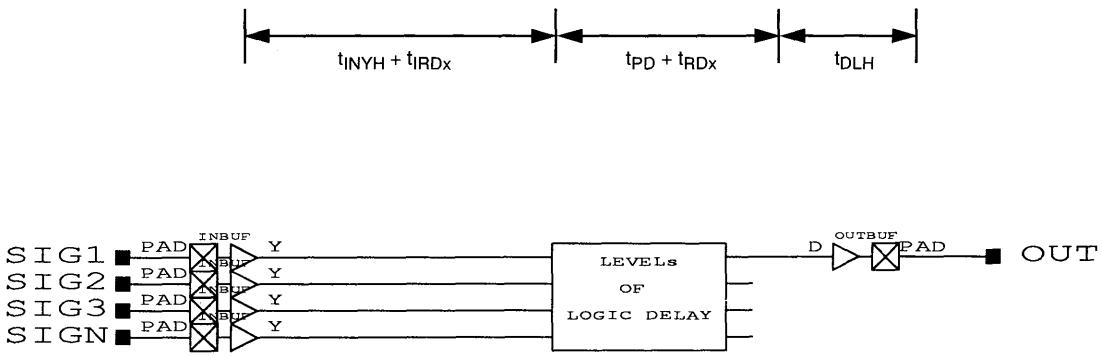
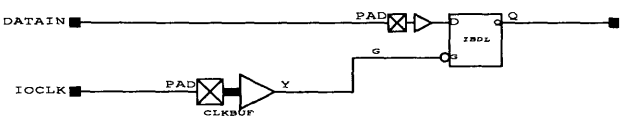
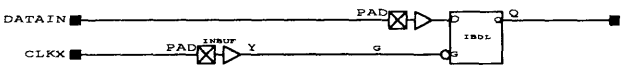


Figure 6. Simple Combinatorial I/O



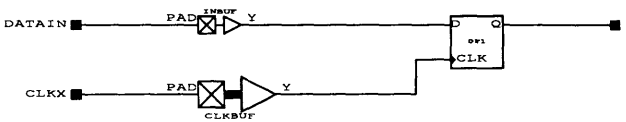
Setup Time = t_{SUEXT}

Hold Time = t_{HEXT}



Setup Time = $t_{INSU} - t_{INYH} - t_{IRD1}$

Hold Time = $t_{INH} + t_{INYH} + t_{IRD1}$



Setup Time = $t_{SUD} + t_{INYH} + t_{IRD1} - t_{CKH}$

Hold Time = $t_{HD} + t_{CKH} - t_{INYH} - t_{RD1}$

Figure 7. Latching Input Examples

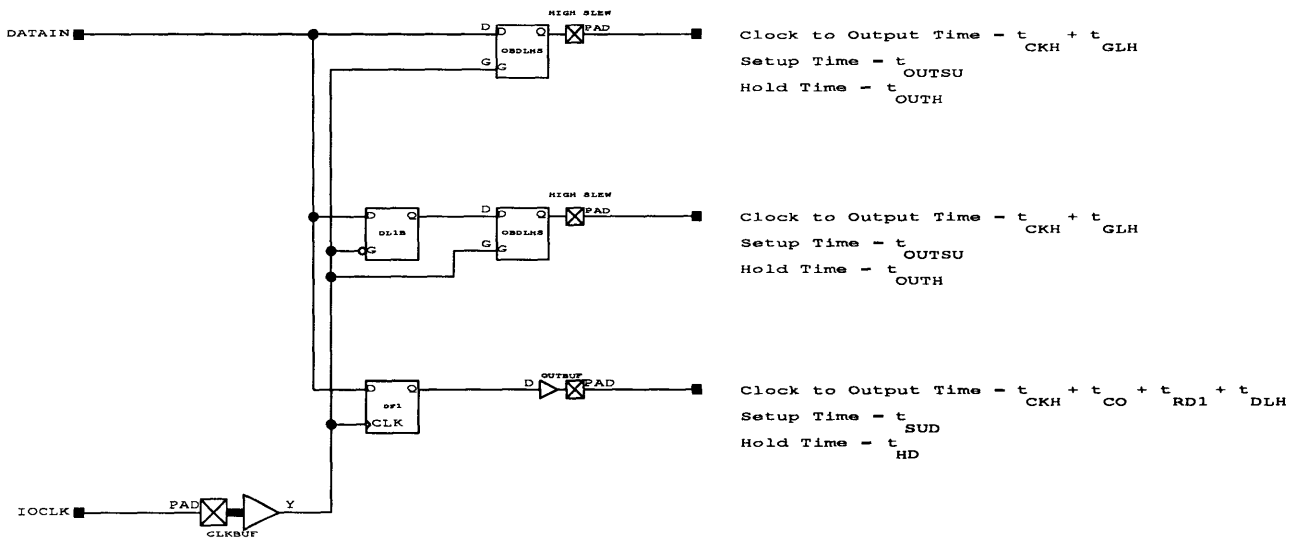
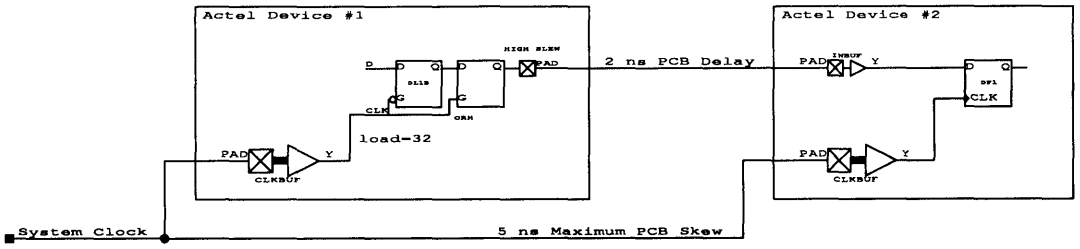


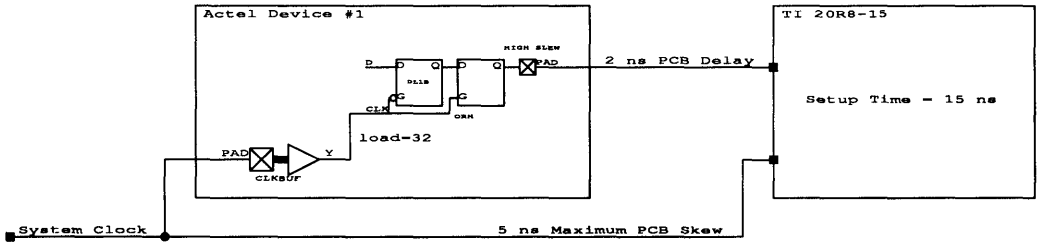
Figure 8. Registered Output Examples



$$F = 1/(T_{CK-Q} + PCB_{DELAY} + PCB_{SKEW} + T_{SU})$$

$$(F = 1/(19.1 + 2 + 5 + (-2.8)) = (1/23.3)\,ns = 42.9\,MHz)$$

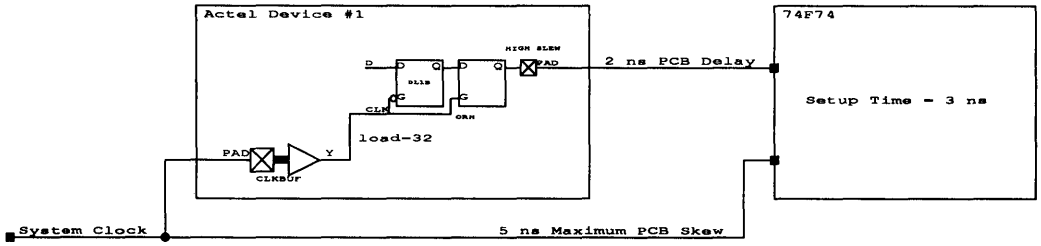
Figure 9. Actel to Actel Message Transmission



$$F = 1/(T_{CK-Q} + PCB_{DELAY} + PCB_{SKEW} + T_{SUPAL})$$

$$F = 1/(19.1 + 2 + 5 + 15) = (1/41.1)\,ns = 24.3\,MHz$$

Figure 10. Actel to PAL Message Transmission



$$F = 1/(T_{CK-Q} + PCB_{DELAY} + PCB_{SKEW} + T_{SUTTL})$$

$$F = 1/(19.1 + 2 + 5 + 3) = (1/29.1)\,ns = 34.4\ MHz$$

Figure 11. Actel to TTL Message Transmission

The Hidden Cost of Reprogrammability

Application Note

On the surface, reprogrammability may appear to be a no-lose feature in a field programmable fate array (FPGA) design environment. However, this issue is more complex in the real engineering world. Unless reprogrammability is required for the system-level design, its purported benefits can be substantially outweighed by hidden costs and pitfalls. This application note outlines the areas an engineer must examine before selecting a programmable digital logic technology.

Background

Programmable logic device (PLD) and enhanced programmable logic device (EPLD) designers have evolved a trial and error design philosophy—they enter a design, compile and program it, then "see if it works." If the design doesn't work, they are forced to analyze output states based on input states because internal signals are not available for analysis. Since the behavior of the internal signals cannot be observed, the solutions to design problems are difficult to uncover. Each possible solution must be entered, compiled, and programmed. Then, the engineer must "see if it works" again. For many designs, this cycle may be repeated several times.

For asynchronous designs, reprogramming is even more difficult and time consuming, especially because changes may affect the functionality of the overall design as well as portions that previously worked! An alternative to this approach is to utilize incremental logic design: implementing small functional blocks, debugging them, and then combining all the blocks. This approach also consumes many iterations and still requires much educated guessing from the engineer.

With a reprogrammable PLD/EPLD, an argument can be made that this trial and error approach is acceptable because one device can be utilized over and over again. However, the hidden cost is the time spent debugging, modifying, compiling, and repeating the process. Obviously, for a simple 22V10 design, little time is involved. But as the design increases in gate count, speed, and circuit complexity, the design time goes up exponentially. Specifically, a 2000 or 4000 gate design might have many errors that take weeks to find and debug with a trial and error design approach. Reprogrammability is worth very little if the errors in the design cannot be discovered easily. This same phenomenon was experienced by the early users of microprocessors and microcoded bit slice machines. This predicament drove the development of in-circuit emulators (ICE) by the microprocessor and development tool suppliers such as Intel and Motorola. Few engineers today design microprocessor-based systems without utilizing an ICE.

The Real Cost of a Design

The cost of design time has always affected the engineering budget, but in today's competitive marketplace, design time even more dramatically affects the product's success and profitability. Successful companies such as 3Com and Hewlett-Packard have determined that each week of delay translates into a loss of $30,000. Even worse, a delayed product might miss the market window entirely. Together these costs force engineering departments to consider the total cost of the design, including time and device costs, in deciding a design process.

Cost-Effective Design Alternative

To afford the engineer a more cost-effective total solution, Actel has developed an architecture and tools that enable an engineer to utilize not only the most cost-effective programmable devices but also the most cost-effective design methodology. At first glance, Actel does not appear to offer the most cost-effective design solution because the Actel devices are one-time programmable. In fact, given the design methodology described above, Actel's devices might appear to be the most expensive, potentially requiring an engineer to program dozens of devices before a system design is complete.

However, Actel's architecture enables engineers to approach logic design in an entirely different manner, saving days if not weeks of design time as well as reducing the number of devices needed to achieve a fully functional design. Historically, addressing the logic design problem meant using simulation. While simulation enables the observation of an unlimited number of internal functions and all timing aspects of a device, it usually requires that a new tool and design methodology be learned. To avoid the investment in the time and money learning new methods, Actel offers a tool that achieves practically all the advantages of simulation without a lengthy, expensive learning period.

Actionprobe® Diagnostic Tools

Actel's architecture offers a unique feature that supports the probing of any internal node in an Actel device running in system. Conceptually, Actel's Actionprobes work just like a microprocessor in-circuit emulator, enabling any internal points to be examined "on-the-fly" in real time. Architecturally, the Actionprobes are controlled and observed by four special probe I/O pins on each device. These pins function as user-defined I/Os during normal device operation. Note that probes can be utilized in conjunction with simulation to catch undefined or ill-defined environmental conditions because the chip is now running in system at real time.

When the Actionprobes are enabled, two probe pins are used to select two internal nodes for observation. These signals are sent to the two Actionprobe output pins. There they can be viewed with an oscilloscope or logic analyzer. The Actionprobe signals may be changed dynamically in circuit, in real time.

Because the Actionprobe network is overlaid across the entire chip, there is no incremental load applied to the point being tested. Thus, the use of the Actionprobes does not change the dynamic characteristics of the circuit. Also, since the choice of probe signals can be changed at any time, any piece of the design can be debugged in real time without any additional compiling or programming. A chip's inputs can be traced through the entire device up to the output pins. If desired, after the design is tested and released to manufacturing, the probes can be used in circuit for the in-system testing of pin continuity as part of the manufacturing test process.

The Actionprobes allow an improved design methodology that is quicker, more controllable, and substantially less frustrating than the trial and error process. Achieving a working design requires very few iterations, saving time and money. In practice, anywhere from a few days to weeks can be saved depending on the design complexity and problem/solution iteration cycle. Additionally, very few devices are actually used to achieve a working design. A survey of Actel's users have shown that on average fewer than five devices are needed to achieve a working design. Designers attain working designs sooner, with fewer iterations, and at the cost of less than five devices.

The total cost of designing with Actel FPGAs is substantially more cost-effective than with reprogrammable approaches. The Actel FPGA approach typically saves one week of design time per chip. The cost savings (one week of engineering design time minus the cost of a handful of devices) may be conservatively estimated to be several thousand dollars per design. The real cost savings can be much greater given considerations such as the complexity of the design. The Actel solution will still be more cost-effective in the long run even if the number of devices used in development becomes large. In production, Actel devices are substantially more cost-effective than reprogrammable devices.

The Real Cost of Reprogrammability

To understand some of the less obvious costs associated with reprogrammable parts, consider the fundamental technology used to program them. The programming elements in reprogrammable devices are much larger than the Actel antifuse. The antifuse is less than 1.0 micron in diameter, it is a passive device, and it lies within the metal routing tracks of the device. No overhead is required other than that for the programming and testing circuitry, which is roughly the same amount for all programmable devices. Other programming elements require substantially more room because they are active devices. The larger programming elements increase the area overhead required for sufficient routing resources and ultimately result in a larger die size, a less flexible architecture, or both. As can be seen in Figure 2, because of the antifuse advantage, the Actel die sizes are substantially smaller and as a result offer a more cost-effective production device.

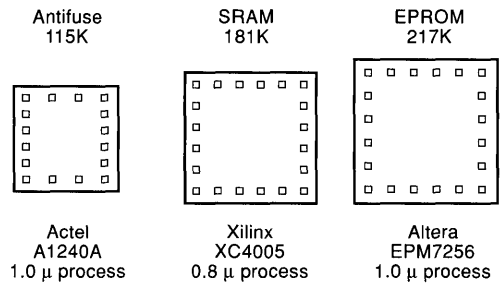


Figure 1. FPGA Die Sizes

While the unit cost tradeoffs are obvious, the architecture tradeoffs are not. The abundant, virtually free routing resources afforded by the antifuse switching element enable logic changes to be made with little or no impact on the performance or utilization of the device. No other programmable element-based architecture offers this flexibility. The smaller antifuse not only makes abundant internal routing resources available, but it also enables abundant routing resources between I/O and the internal device logic. This enables design changes to be made with little or no impact on the I/O locations. I/O locations can be finalized and the PCB artwork sent out for manufacturing without worrying about design changes affecting the pinout of the chip, which requires the PCB to be revised prior to manufacturing. Obviously, turning a PCB is not only very expensive, but it also requires a schedule delay.

The impact of routing resources on I/O locations is directly related to the design's gate count, complexity, and speed. The higher the performance, complexity, or gate count as a percentage of the device's available gates, the worse the problem can be. Actel's antifuse-based devices are the most flexible programmable devices in this area. The designer is virtually guaranteed that I/O locations won't change because of logic changes to the design. No other programmable technology can make this claim.

Conclusion

Reprogrammable architectures appear at first glance to offer the most cost-effective design solution, but in reality they are the most expensive. Not only are reprogrammable architectures more expensive on a per unit basis, they don't offer a probing capability that enables users to find and debug errors. Furthermore, they don't offer enough routing resources to make design changes without I/O changes that could require expensive PCB turns or extra nonbudgeted design time. When selecting a programmable architecture, design engineers should weigh all these factors and their associated costs, as well as the opportunity cost of being late to market. Given these, the logical choice is Actel FPGAs for the most cost-effective design solution.

Introduction

The ACT™ 1, ACT 2, and ACT 3 families of Actel FPGAs are based on channeled array architecture consisting of rows of modules interspersed with routing channels. There are two types of modules: logic modules and I/O modules. The logic modules are blocks that implement logic functions. What follows describes in detail the structure of logic modules for the ACT 1, ACT 2, and ACT 3 families.

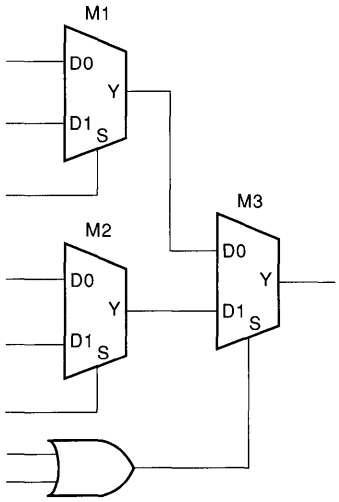


Figure 1. Logic Module of ACT 1 Devices

Discussion

ACT 1 Logic Modules

A logic module for ACT 1 devices is an eight-input, one-output logic circuit that can implement logic functions (NAND, AND, OR, NOR , and so on) in gates of two, three, or four inputs. One logic module has three 2-to-1 multiplexers with different select lines (Figure 1). The select lines of the two multiplexers (M1 and M2 in Figure 1) are individual inputs. The select line of the third multiplexer (M3 in Figure 1) is an OR function of the other two inputs. One logic module can implement all functions of two-input variables, most functions of three-input variables, and some functions of four-input variables. Figure 2 shows one of several ways to implement a three-input AND gate using one ACT 1 logic module. Actel's layout software automatically selects the best implementation for each instance of a logical function. All latches in the ACT 1 architecture are implemented with one logic module, while all flip-flops are implemented with two adjacent logic modules in a master/slave configuration. The full ACT 1 logic module is available for use as the CM8A hard macro.

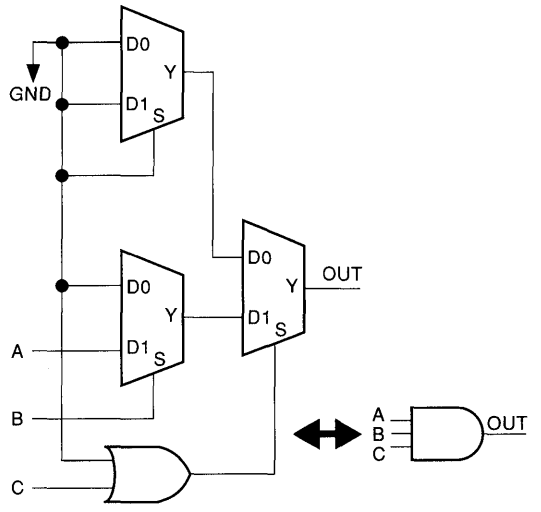


Figure 2. An Example of Implementing a Three-input AND Gate Using One ACT 1 Logic Module

ACT 2 Logic Modules

Logic modules for ACT 2 devices are classified into two types: combinatorial modules (C-modules) and sequential modules (S-modules) (Figure 3 and Figure 4). The C-module is an enhanced version of the ACT 1 family logic module optimized to implement high fan-in combinatorial logic functions. One C-module consists of a 4-to-1 multiplexer with one select line (S1 in Figure 3) acting as an OR function of the two inputs, A1 and B1, and the second select line (S0 in Figure 3) acting as an AND function of two other inputs, A0 and B0. All three-input gates, most four-input gates, and some five-input gates can be generated with one C-module. The S-module is designed to implement a high-speed flip-flop (or latch) with additional combinatorial logic in a single module without additional delay. For example, hard macro DFM6A fits into one ACT 2 S-module, but it does not fit into one ACT 1 logic module. The S-module includes a C-module and a sequential element. An S-module can be configured as a seven-input combinatorial function plus a D-type flip-flop with clear (Figure 4a) or as a seven-input combinatorial function plus a latch without clear (Figure 4b). However, one S-module implements only a four-input logic function plus a latch with clear (Figure 4c). Thus, the combinatorial function used in the S-module is not a full implementation of the ACT 2 C-module. The S-module can also be configured strictly as a C-module for maximum combinatorial logic utilization (Figure 4d).

8

Hard macro logic functions range from simple SSI gates such as AND, OR to more complex functions such as flip-flops with 4-to-1 multiplexed data inputs. Hard macros are implemented in the ACT 2 architecture by using one or more C-modules or S-modules. For ACT 2 and ACT 3 devices, hard macros are divided into two groups: combinable and non-combinable. If a combinable macro is driving a register, the Action Logic® System (ALS) will automatically "combine" the combinable hard macro and register into one sequential macro (Figure 5). However, some hard macros cannot be combined even if they are driving a sequential macro. The combinable hard macros are divided into two groups: those that can be combined with sequential macros DF1, DF1B, DFC1B, DFC1D, DL1, DL1B, DLC, and DLCA, and those that can be combined with DF1, DF1B, DFC1B, DFC1D, DL1, and DL1B. Please refer to the *ACT Family Macro Library Guide* for the combinability of each hard macro. The full ACT 2 logic module is available for use as the CM8 (combinatorial) and DFM7A/B sequential macros.

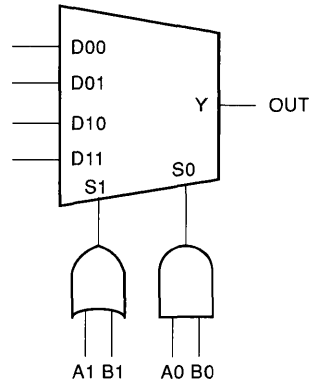
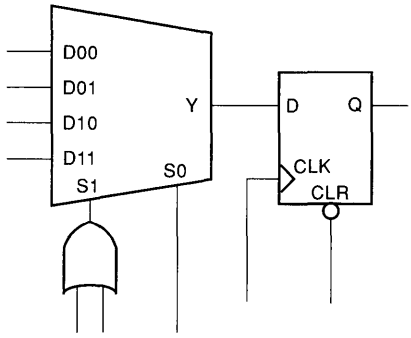
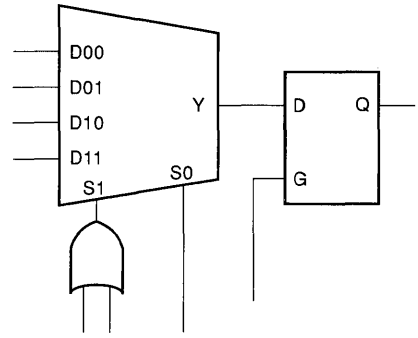


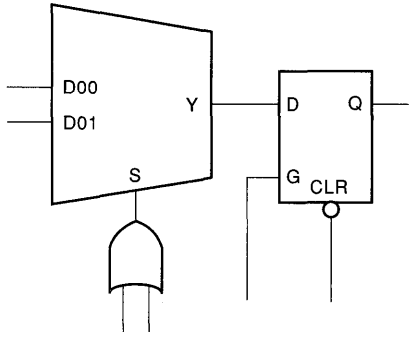
Figure 3. ACT 2 C-module Implementation



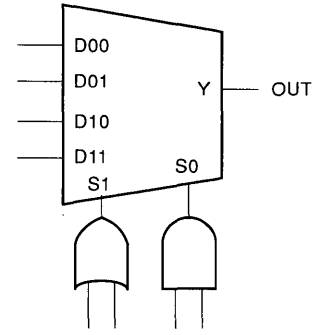
a. Up to Seven-input Function Plus D-type Flip-Flop with Clear (ACT 2)



b. Up to Seven-input Function Plus Latch (ACT 2)



c. Up to Four-input Function Plus Latch with Clear (ACT 2)



d. Up to Eight-input Function— Same as C-module (ACT 2)

Figure 4. ACT 2 S-Module Implementation

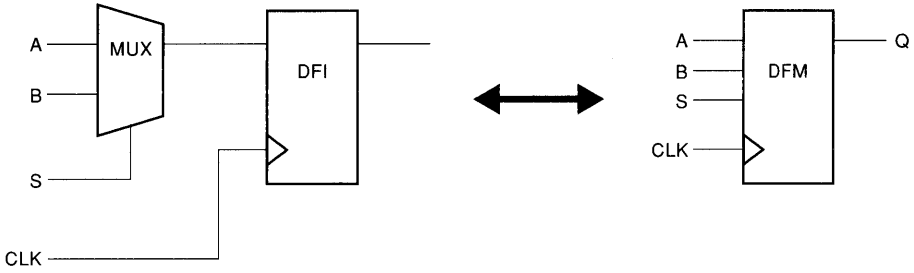


Figure 5. Logic Combining

ACT 3 Logic Modules

Logic modules of ACT 3 devices are enhanced versions of the ACT 2 family logic modules. The C-module is equivalent to the ACT 2 family C-module (Figure 6). The S-module contains a full implementation of the C-module with eight inputs plus a clearable element that can be configured as a latch or a D-type flip-flop (Figure 7). A single S-module can implement any function implemented by a C-module plus a flip-flop or latch with clear. This allows ALS to automatically combine any C-module macro driving an S-module macro into the S-module. The clock input CLOCK may be connected to one of three clock networks, CLKA, CLKB, HCLK or any other internal macro. As in the ACT 2 devices, ALS can configure an S-module as a C-module. The ACT 3 logic module is available for use as the CM8 and DFM8A/B hard macros.

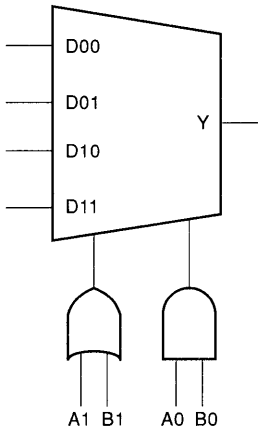


Figure 6. ACT 3 C-module Diagram

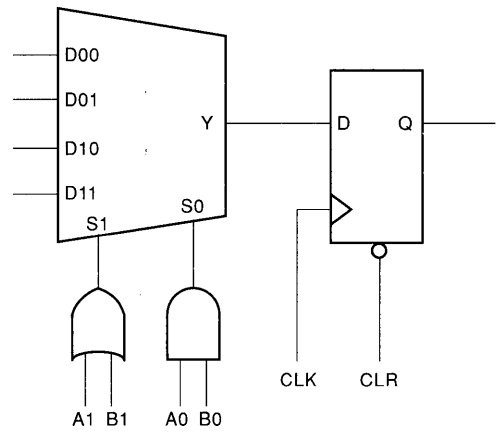


Figure 7. ACT 3 S-module Diagram

8

The following table lists the number of logic modules in all ACT 1, ACT 2, and ACT 3 devices.

	A1010	A1020	A1225	A1240	A1280	A1415	A1425	A1440	A1460	A14100
Logic Modules	295	547	451	684	1232	200	310	564	848	1377
S-modules			231	348	624	104	160	288	432	697
C-modules			220	336	608	96	150	276	416	680

Summary

Logic modules of ACT 1 devices are eight-input, one-output combinatorial modules. Logic modules of ACT 2 and ACT 3 devices are classified into S-modules and C-modules. Some of the flip-flop functions can be implemented within a single S-module without causing extra delay. Each logic function may have many versions with different combinations of active-low/active-high inputs and outputs.

Binning Circuit of Actel FPGAs

Application Note

Introduction

All Actel FPGA devices are available in different speed grades to meet varying system design requirements. As part of the production process, devices are separated, or "binned," into groups by their measured performance. The performance of each device is measured by a dedicated binning circuit that closely characterizes its speed. The Actel binning circuit measures the AC performance of a device prior to programming. Since the binning circuit shares the same process as the rest of the die, the binning circuit speed reflects the speed of the device after programming. Actel guarantees the performance of every device for all speed grades. Therefore, the following descriptions of ACT™ device binning circuits are mostly for informational purpose only.

ACT 1 Binning Circuit

The ACT 1 binning circuit consists of one input buffer, n logic modules (n = 16 for A1010/A1010A/A1010B and n = 28 for A1020/A1020A/A1020B), and one output buffer. The binning circuit delay is obtained by setting the device to the test mode (that is, MODE pin = HIGH), and setting up the appropriate mode register bits. Using the SDI pin as the mode register input and DCLK as the clock, seven mode register bits are clocked into the device in the sequence, 1011101. After this setup process, the propagation delay of the binning circuit is measured from the input pin BININ to the output pin PRA. Figure 1 shows the delay of binning circuit T_{PDB}. Table 1 shows the binning circuit pin assignments for ACT 1 family devices.

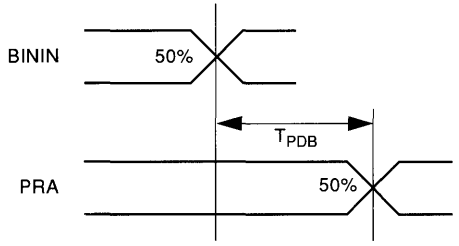


Figure 1. T_{PDB} Measurement Waveform

ACT 2 Binning Circuit

The delay path of the ACT 2 binning circuit includes an input pad through an input module, n combinatorial logic modules (n = 16 for A1280/A1280A, 12 for A1240/A1240A, and 10 for A1225/A1225A), an output module, and a pad driver. Binning circuit delays are obtained by setting the MODE pin high, then shifting data into the internal mode, control, and binning registers through the SDI pin for the appropriate mode of operation. To set up the binning circuit, several data patterns are shifted into SDI with each rising edge of DCLK. The first ten data bits are counter bits used to set up the internal counter length. The next 21 bits are Mode bits used to set up the mode of operation, and the remaining bits are used to fill the registers in the shift register chain. There is one stop bit at the end. Then, the binning circuit delay can be measured from input pin, BININ, to output pin, BINOUT. Tables 2 and 3 list the binning circuit pin assignments and the data patterns for ACT 2 binning circuits.

ACT 3 Binning Circuit

The ACT 2 and ACT 3 binning circuits are identical except for the data patterns. Tables 4 and 5 list the binning circuit pin assignments and the data patterns for ACT 3 binning circuits.

8

Table 1. ACT 1 Binning Circuit I/O Pins

		PLCC/JQCC			CPGA	CQFP	PQFP
		44-pin	68-pin	84-pin	84-pin	84-pin	100-pin
MODE		34	54	66	E11	55	92
SDI		36	56	72	B11	61	98
DCLK		37	57	73	C10	62	99
BININ	1010	9	12	NA	D2	NA	34
	1020	7	10	13	C2	2	32
PRA		38	58	74	A11	63	100

Table 2. ACT 2 Binning Circuit I/O Pins

	A1280/A1280A			A1240/A1240A		
	176 PGA	172 CQFP	160 PQFP	144 PQFP	132 PGA	84 PLCC
MODE	C3	1	159	2	A1	12
SDI	B14	131	38	110	B12	76
DCLK	B3	171	2	144	C3	10
BININ	N3	44	119	37	L3	34
BINOUT	P2	45	118	38	M2	35

	A1225/A1225A		
	100 PGA	100 PQFP	84 PLCC
MODE	C2	4	12
SDI	C8	79	76
DCLK	C3	2	10
BININ	K2	28	34
BINOUT	L2	29	35

Table 3. ACT 2 Binning Circuit Data Patterns

Device	#Bits	Counter Bits	Mode Bits	Shift Register Bits and Stop Bit
A1280/A1280A	276	0001101010	0000 0101 1100 0101 0000 0	one 1, followed by 244 0s
A1240/A1240A	218	0110010011	0000 0101 1100 0101 0000 0	one 1, followed by 186 0s
A1225/A1225A	176	1010101011	0000 0101 1100 0101 0000 0	one 1, followed by 144 0s

Table 4. ACT 3 Binning Circuit I/O Pins

	A1425/A1425A		A1460/A1460A	
	133 CPGA	160 PQFP	208 PQFP	207 CPGA
MODE	E3	9	11	D7
SDI	C2	2	2	C3
DCLK	D4	160	208	E4
BININ	K3	38	50	D13
BINOUT	L4	39	51	E14

Table 5. ACT 3 Binning Circuit Data Patterns

Device	#Bits	Counter Bits	Mode Bits	Shift Register Bits and Stop Bit
A1425/A1425A	410	1101100001	0000 0101 1100 1101 0000 0	one 1, followed by 378 0s
A1460/A1460A	553	0101101100	0000 0101 1100 1101 0000 0	one 1, followed by 521 0s

The Actel architecture provides global clock networks that allow high fanout drive for flip-flops and latches with minimal skew. Table 1 shows the available global networks and their characteristics, which are defined as follows.

- *Routed clocks* are clock networks which can be used by selecting CLKBUF/CLKBIBUF or CLKINT macros or both.
- *Dedicated clocks* are clock networks that are directly wired to sequential and I/O modules. They contain no programming elements in the path from the I/O pad driver to the input of S-modules or I/O modules; they provide sub-nanosecond skew and guaranteed performance.
- A *special* network refers to a special hard-wired input for I/O modules that can only drive preset/clear pins of I/O modules.

Note that the *Internal Drive Option* for ACT™ 2 and ACT 3 can only be utilized by selecting the CLKINT macro to drive an internal clock network.

ACT 1 Clock Network

A single clock distribution network is provided on ACT 1 arrays. The clock network provides unlimited fanout (the ability to drive all logic modules in the array) with minimal delay and skew time. Figure 1 illustrates the clock distribution network for an ACT 1010 array. It is arranged similarly in the ACT 1020 array.

Table 1. Global Clock Attributes

Input Pad Name	Type	Family	Number	Internal Drive Option	Macro	Note
CLK	routed	ACT 1	1	No	CLKBIBUF, CLKBUF	Can adjust skew with clock balancing
CLKA, CLKB	routed	ACT 2	2	Yes	CLKBIBUF, CLKBUF, CLKINT	Use CLKINT for Internal Drive Option
CLKA, CLKB	routed	ACT 3	2	Yes	CLKBIBUF, CLKBUF, CLKINT	Use CLKINT for Internal Drive Option
HCLK	dedicated	ACT 3	1	No	HCLKBUF	Connected to all S-modules
IOCLK	dedicated	ACT 3	1	No	IOCLKBUF	Connected to all I/O modules
IOPCL	special	ACT 3	1	No	IOPCLBUF	Connected to I/O module set and reset pins

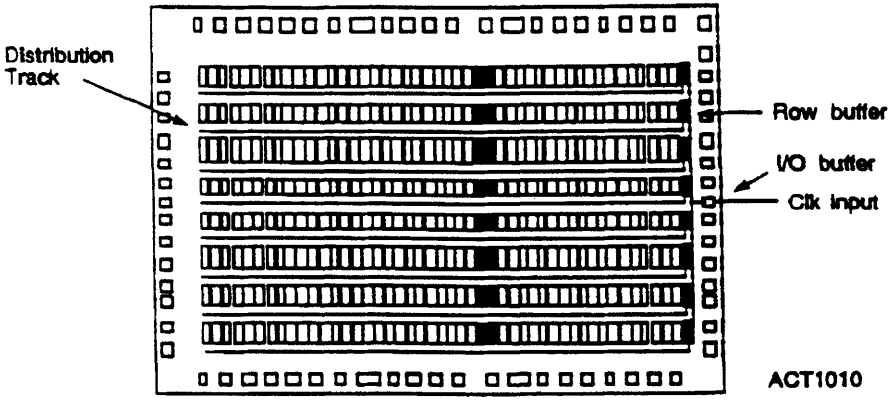


Figure 1. Clock Distribution Network for an ACT 1010

8

The network is driven by a specific I/O pin that drives a dedicated on-chip buffer tree. Each row of logic modules (8 on the A1010 and 14 on the A1020) has a dedicated buffered clock track. The clock distribution network is selected automatically when the CLKBUF macro is used in the schematic and assigned to its dedicated package pin. The CLK I/O pin can also be used for normal I/Os by assigning INBUF, OUTBUF, TRIBUFF, or BIBUF to the CLK pin location.

Clock Balancing Scheme

Clock balancing equalizes the clock loads on each branch of the global clock network, thereby minimizing clock skew. Clock skew can cause setup and hold time problems. The clock balancing strength sets the level of clock balancing for ACT 1 designs only. It is not used for ACT 2 and ACT 3 designs because the global clock networks for those families have been designed for minimal clock skew. The results are design dependent. If more than 50 percent of the logic modules are driven by the global clock, the effect is minimal. Also, strong clock balancing may result in increased delays and reduced routability.

ACT 2 Clock Networks

Two low-skew, high fanout clock distribution networks are provided in the ACT 2 architecture. Figure 2 illustrates the implementation of one of two ACT 2 clock networks using CLKBUF/CLKBIBUF or CLKINT macro or both. ACT 2 devices offer two identical clock networks. The clock modules are located in the top row of I/O modules. Clock drivers and a dedicated horizontal clock track are located in each horizontal routing channel. The clock input pads may also be used as normal I/Os, bypassing the clock networks. These networks are referred to as CLKA and CLKB. Each network has a clock module that selects the source of the clock signal and may be driven as follows:

- externally from the CLKBUF macro
- externally from the CLKBIBUF macro
- internally from the CLKINT macro

As mentioned above, the macro CLKBUF is used to connect one of the two external clock pins to a clock network, and the macro CLKINT is used to connect an internally generated clock signal to a clock network. Figure 3 illustrates the implementation of internal clock network using CLKINT macro. In the figure, the input signals are driven by regular I/O buffers, not CLKBUF or CLKBIBUF. These input signals drive a user-created Clock Conditioning Module, which generates the internal clock signal. This signal is subsequently driven by the CLKINT macro.

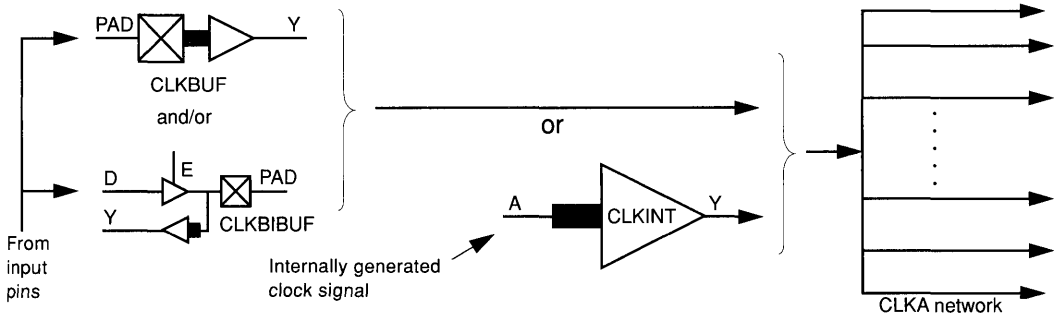


Figure 2. One of Two ACT 2 Clock Networks (CLK0)

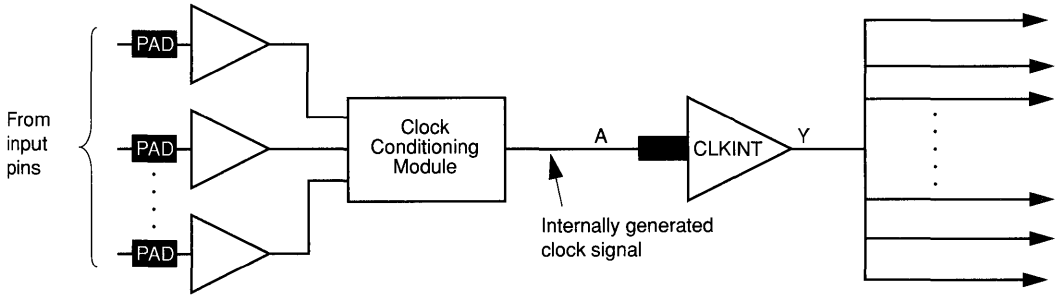


Figure 3. Implementation of Internal Clock Network Using CLKINT Macro

ACT 3 Clock Networks

The ACT 3 architecture contains four clock networks: two high performance dedicated clock networks and two general purpose routed networks. The high performance networks function at up to 150 MHz, while the general purpose routed networks function at up to 75 MHz. Figure 4 illustrates the ACT 3 clock networks.

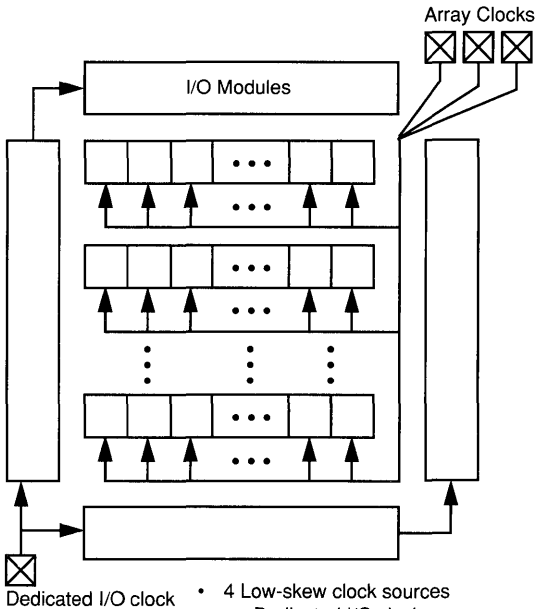


Figure 4. ACT 3 Clock Networks

Dedicated Clocks

There are two dedicated clock networks: one for the array registers known as Dedicated Array Clock (HCLK) and one for the I/O registers known as Dedicated I/O Clock (IOCLK). The clock networks are accessed by special I/Os. Figure 5 shows the macros that drive the Dedicated Array and I/O clock networks in ACT 3.

HCLK is a dedicated hard-wired clock input for sequential modules. HCLK is directly wired to each S-module and offers guaranteed clock speeds independent of the number of S-modules being driven. IOCLK is a dedicated hard-wired clock input for I/O modules. IOCLK is directly wired to each I/O module and offers guaranteed clock speeds independent of the number of I/O

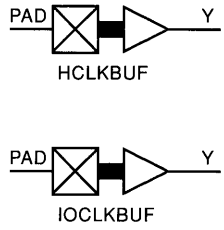


Figure 5. Dedicated Array and I/O Clock Buffers

modules being driven. These dedicated clock networks support high performance by providing sub-nanosecond skew and guaranteed performance. Dedicated clock networks contain no programming elements in the path from the I/O pad driver to the input of S-modules or I/O modules.

Routed Clocks

The routed clock networks for ACT 3 have the same characteristics as the ACT 2 networks as shown in Figure 2 and Figure 3. The macros that drive routed clock networks in ACT 1, ACT 2, and ACT 3 are shown in Figure 6.

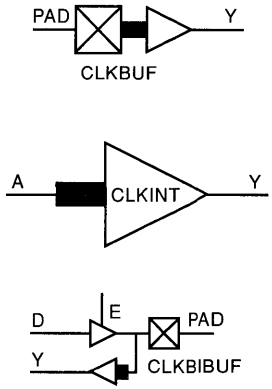


Figure 6. Dedicated Routed Clock Buffers

CLKA and CLKB are global signals with unlimited fanout. Refer to ACT 2 clock networks described above for more detail on ACT 3 routed clocks.

Special Hard-wired Preset/Clear Network

IOPCL is a dedicated special hard-wired input for I/O modules. It is directly wired to the Preset and Clear inputs of all I/O registers. IOPCL functions as an I/O when no I/O preset or clear macros are used. Figure 7 shows the IOPCLBUF macro that drives the dedicated hard-wired Preset/Clear network. IOPCLBUF can only be connected to the preset/clear pins of I/O macros.

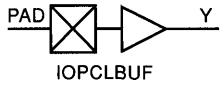


Figure 7. Dedicated Preset/Clear Network Buffer

The routed clocks CLKA and CLKB can also be used to drive high fanout nets like resets, output enables, or data enables. This saves logic modules and results in performance increases in some cases.

Clock Connections for ACT 2 and ACT 3

To minimize loading on the clock networks, only a subset of module inputs has antifuses on the clock tracks. Therefore, only a few of the C-module and S-module inputs can be connected to the clock networks. To further reduce loading on the clock network, only a subset of the horizontal routing tracks can connect to the clock inputs of the S-module. Figure 8 illustrates the connections to the clock networks.

Creating a User-Defined Clock Distribution Network

Some applications require many internal clock networks. For these type of designs or for clock networks with a small number of loads, it may be better to create a user-defined clock network. Because this clock network is not a built-in, dedicated circuit, the skews and delays cannot be guaranteed.

However, by intelligently placing the I/O pins and declaring the nets associated with the clock network as critical, the automatic placement of these macros create a network that controls clock skew and delay. The results can be verified by running a simulation or using the ALS Timer. It may be necessary to place and route again to meet timing requirements. To minimize the skew between paths, try to equalize the loading in each leg of the clock distribution network.

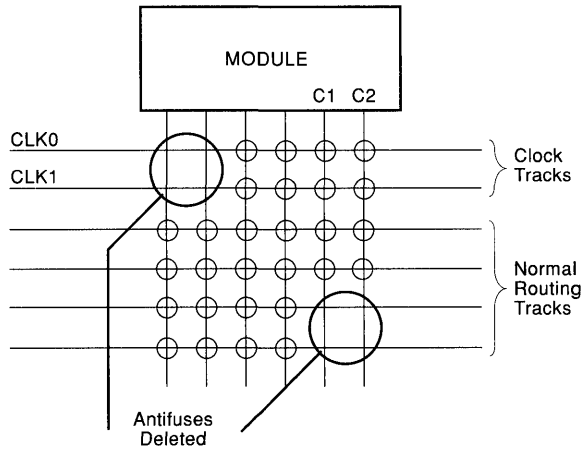


Figure 8. Fuse Deletion on Clock Networks

Using Dedicated Clock and Clear for ACT 3 Registered I/O Macros

Application Note

Introduction

The ACT™ 3 FPGA family includes several performance and capacity features that make fast, large, effective designs easier to develop. Among these new features, the architecture of the ACT 3 I/O modules enables fast on-chip and off-chip data transfer through the use of built-in register functions. These input/output macros include built-in registers with asynchronous preset or clear functions. The ACT 3 architecture also includes special circuits to drive the clock and preset/clear input pins. These dedicated circuits are used as global signals for all I/O macros in an ACT 3 design.

Using the Clock and Asynchronous Control Inputs

Every registered ACT 3 I/O macro (for example, DECETH, IREC, and OREPTL) includes an asynchronous preset or clear input IOPCL. This input asynchronously sets the output of I/O flip-flops to 0 or 1. The IOPCL pin of a registered I/O macro must be driven by the dedicated I/O macro, IOPCLBUF. The IOPCLBUF is externally driven in turn by the dedicated IOPCL package pin as shown in Figure 1.

Similarly, the clock (CLK) inputs of registered I/O macros are driven by a dedicated circuit. The dedicated I/O clock buffer, IOCLKBUF, is used to drive the clock pin of each I/O macro. The IOCLKBUF is driven in turn by the dedicated external package pin, IOCLK, as shown in Figure 2. See the package pin assignment diagrams in the ACT 3 datasheet for specific locations of the IOPCL and IOCLK package pins.

If no registered I/O macros are used in an ACT 3 design, then the dedicated IOCLK and IOPCL package pins can be used as normal I/O pins. In that case, these pins can be connected to any other type of I/O buffer except IOCLKBUF, IOPCLBUF, and HCLKBUF. For example, undedicated normal I/O buffers such as INBUF and OUTBUF may be connected to these pins. If any registered I/O macros are used in an ACT 3 design, then the IOCLKBUF and IOPCLBUF macros must be included in the design, and the pin assignment must include the IOPCL and

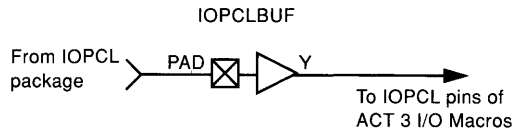


Figure 1. An IOPCLBUF Driven by a Dedicated IOPCL Package Pin

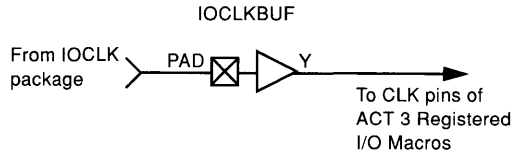


Figure 2. An IOCLKBUF Driven by a Dedicated IOCLK Package Pin

IOCLK pins. The IOPCL and IOCLK pins must be explicitly specified because ALS will not automatically create pin assignments for IOCLKBUF and IOPCLBUF.

In an ACT 3 design, all types of I/O macros can be used in a single design. For example, I/O macros without registers, I/O macros with preset, and I/O macros with clear can be used together in one design. However, IOCLKBUF must drive all the clock inputs of the registered I/O macros and the IOPCL must drive all the clear or preset inputs of the registered I/O macros. CLK and IOPCL pins of registered I/O macros can not be connected to GND or V_{CC}. All ACT 3 I/O registered buffers share the same dedicated clock and asynchronous clear or preset network. Once the built-in dedicated IOCLK and IOPCL networks are used, all registered I/O macros will be cleared and preset at the same time by the same signal.

8

Designing for Combinability with the ACT 2 and ACT 3 Architectures

Application Note

Introduction

The Action Logic® System (ALS) and ACT™ devices are both designed to easily optimize the performance and gate density of Actel field programmable gate array (FPGA) designs. With the Actel system, projects are easier to finish within schedule and with satisfactory performance specifications. The ACT 2 and ACT 3 architectures include the "combinability" feature, which enables the devices to be used efficiently with little or no extra effort. A basic understanding of the ACT 2 and ACT 3 architectures is necessary before taking advantage of the combinability feature. With this background, it is easy to *design for combinability.*

The ACT 2 and ACT 3 Architectures

The ACT 2 and ACT 3 architectures have two types of modules: a combinatorial module (C-module), similar to the ACT 1 module, and a sequential module (S-module), containing a combinatorial module whose output drives a dedicated flip-flop. The combinatorial module is identical for ACT 2 and ACT 3 devices (Figure 1). The ACT 2 sequential module implements the functions shown in Figure 3 on the next page. The ACT 3 sequential module (Figure 2) implements the entire function of the C-module plus a sequential element (flip-flop or latch) with asynchronous clear.

Each library contains many different types of flip-flops with different features to offer the most flexibility for the device. Some flip-flops are implemented very efficiently, requiring only a single S-module. More complex flip-flops may require an S-module and a C-module.

It is important to be aware of the resources used to implement different types of logic, including flip-flops. ALS includes information that classifies flip-flops according to the number and type of logic module used to implement them. Flip-flops made from a single S-module are further divided into combinable and noncombinable flip-flops. Only those flip-flops in the former category may be used to combine.

Designing for Combinability

Combinable flip-flops and latches are so called because the combinatorial resource is free to be used to implement the combinatorial functions in the library. The resource is not used by the flip-flop itself, leaving it available for combinatorial functions. If a combinatorial function in the schematic is driving a single data input to a combinable flip-flop or latch, ALS will put the combinatorial function and the flip-flop into the same S-module. When the two functions are combined into one S-module, there is only one module delay for both macros. Use this technique of connecting combinable combinatorial macros to combinable sequential elements for the fastest, most compact designs.

The ACT Family Macro Library Guide provides information about the combinability of each combinatorial function for ACT devices. Of the 111 combinatorial macros available in the ACT 2 library, 86 are combinable. Some examples of combining are shown in Figure 4. *For ACT 3 devices, all combinatorial macros that use a single C-module are combinable.*

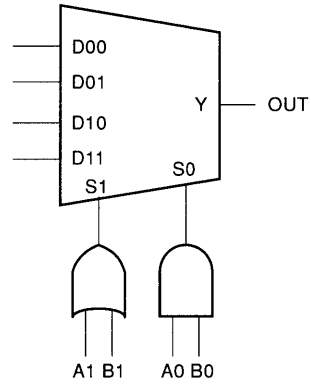


Figure 1. ACT 2 C-Module Implementation

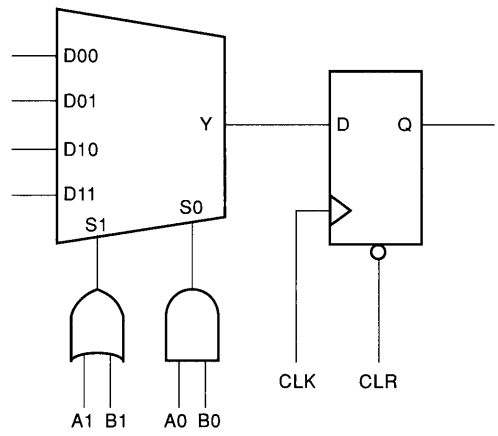
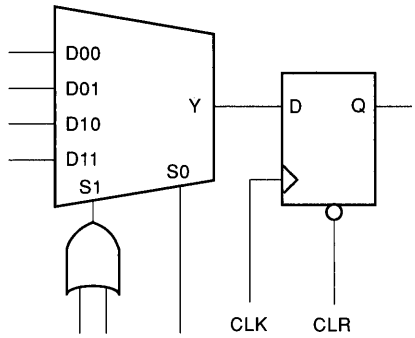
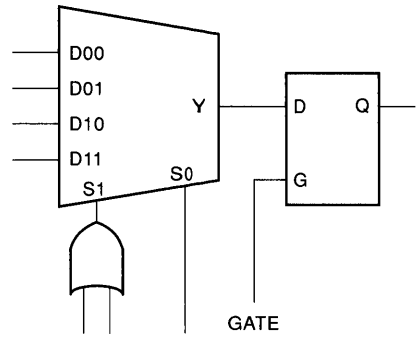


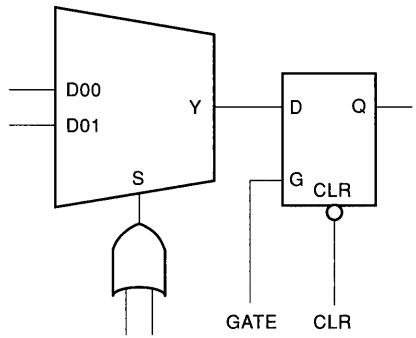
Figure 2. ACT 3 S-Module Diagram



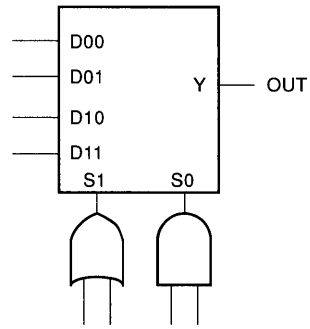
a. Up to Seven-input Function Plus D-type Flip-Flop with Clear (ACT 2)



b. Up to Seven-input Function Plus Latch (ACT 2)



c. Up to Four-input Function Plus Latch with Clear (ACT 2)



d. Up to Eight-input Function—Same as C-module (ACT 2)

Figure 3. ACT 2 S-Module Implementation

Following these three rules will ensure that the architecture will be used effectively most of the time for either flip-flops or latches:

1. Use combinable flip-flop (or latch) whenever possible if it is being driven by a combinatorial function.
2. Combinatorial functions driving flip-flops (or latches) should be combinable whenever possible.
3. If the first two conditions are met, there should be only one connection between the flip-flop (or latch) and the combinatorial function.

Remember that all ACT 3 single C-module macros are combinable. If there is a single load on the outputs.

Benefits of Combining

Combining a sequential and combinatorial function into a single function increases the gate capacity of the device, since it makes more efficient use of the available gates in the S-module. It also improves the performance of the design because the delay through the combinatorial function is included with the setup time for the flip-flop data input. In effect, the combined logic level has zero propagation delay.

For designs that require pipelining, the pipeline stages do not incur additional cost if they can be done using combinable logic. An example can be found in the conversion of an adder to an accumulator. Adders may be designed using the ACT 2 and ACT 3 libraries so that every output is combinable. Converting

such a design to an accumulator can be done without cost simply by adding a register to the adder output bus in the schematic as shown in Figure 5. ALS will automatically combine each output function with its respective flip-flop in the register into one S-module. The ALS timing analyzer and backannotated simulations will list the propagation delay of combined functions as zero.

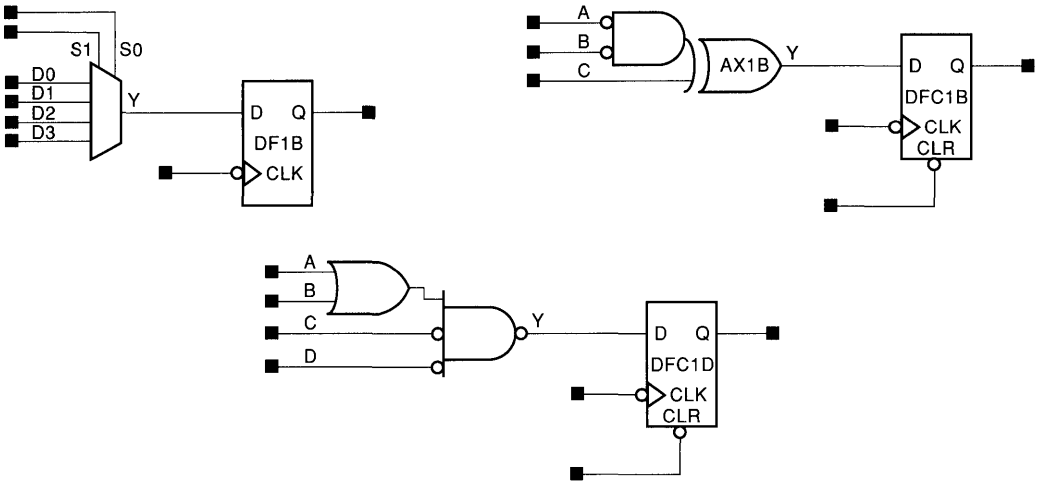


Figure 4. Examples of Combinable Functions

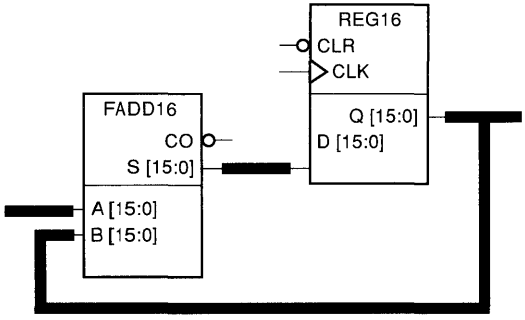


Figure 5. 16-Bit Accumulator

Fast On and Off Chip Delays with ACT 2 I/O Latches

Application Note

Introduction

Using ACT™ 2 devices, latched I/O buffer macros can improve clock input-to-registered-output performance. Latched I/O buffer macros also increase the speed of latching signals into the FPGA. Flip-flops created from these I/O latch macros improve performance by up to 22 percent compared with traditional approaches. This application note compares the use of traditional approaches with the use of I/O latch macros in ACT 2 designs.

Master/Slave Flip-Flops

As shown in Figure 1, two level-sensitive latches can be combined to create a positive edge-sensitive flip-flop.

Where:

$T_{CO} = t_{CO2}$

$T_{SU} = t_{CKL} - t_{CO1} - t_{RD1} > t_{SU2}$

t_{SU2} = minimum setup time for slave

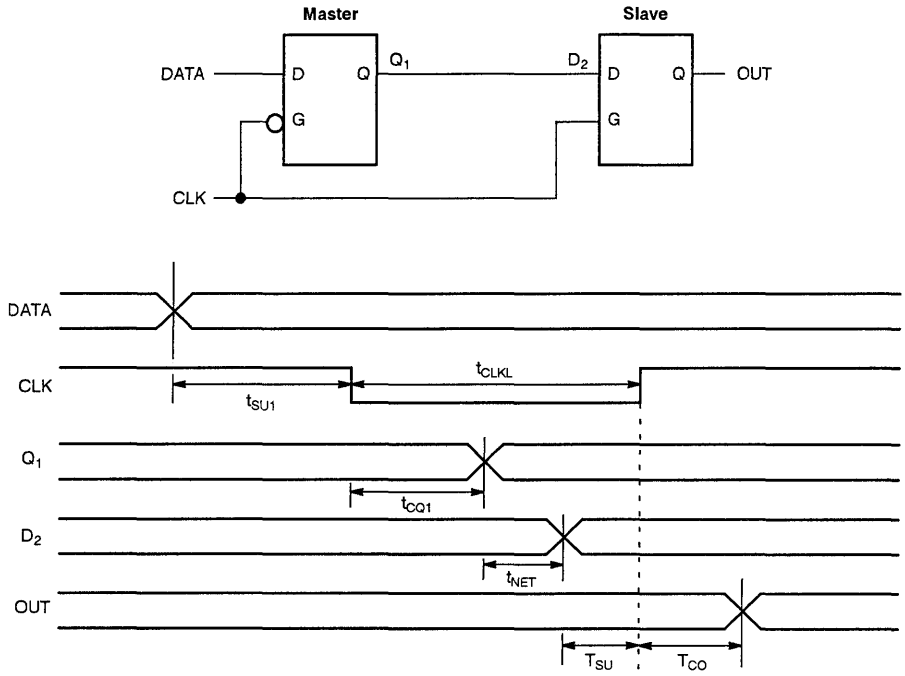
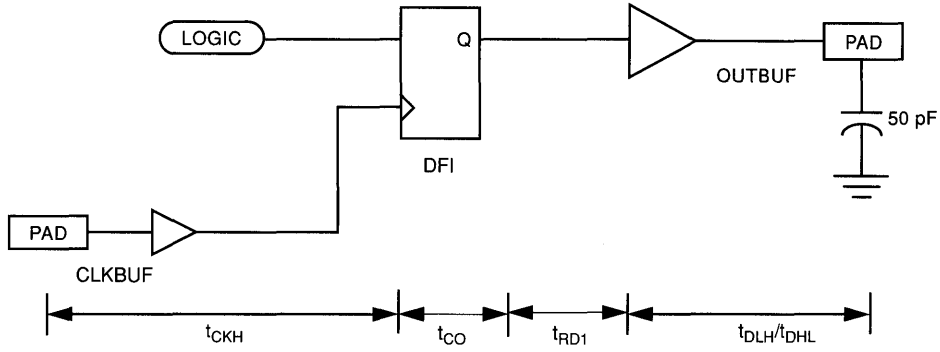


Figure 1. Master/Slave Flip-Flop

The clock- to-output delay of the resulting flip-flop is determined by the clock-to-output delay of the slave latch. The clock period low time (t_{CKL}) must be greater than the clock-to-output delay of the master latch (t_{CO1}) plus the net delay from the master latch output (t_{RD1}) plus the setup time of the slave latch (t_{SU2}).

Constructing Registered Outputs

You can construct a registered output by combining a flip-flop macro with an output buffer as depicted in Figure 2. The clock-to-out delay for an A1225A-2 under worst-case commercial conditions is 23.1 ns rising or 25.2 ns falling.



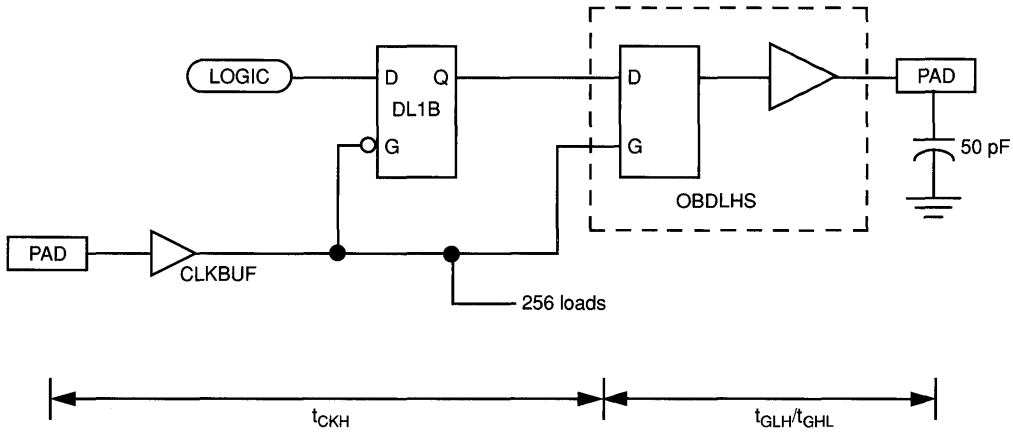
$T_{CO} = t_{CKL} + t_{CO} + t_{RD1} + t_{DLH}/t_{DHL} = 10.2 + 3.8 + 1.1 + 8.0/10.1$ ns = 23.1/25.2 ns (rising/falling)

Figure 2. Conventional Registered Output

I/O Latch Flip-Flops

You can also constuct a registered output as a master/slave flip-flop using a logic module latch (DL1B) and a latched output buffer (OBDLHS) as shown in Figure 3.

In this case, the clock-to-out delay for an A1225A-2 device under commercial worst-case conditions is 20.7 or 23.0 ns. In this case, the loading on the global clock network is assumed to be 256 loads.



$T_{CO} = t_{CKH} + t_{GLH}/t_{GHL} = 11.8 + 8.9/11.2$ ns = 20.7/23.0 ns (rising/falling)

Figure 3. Master/Slave Registered Output

Registered inputs can also be constructed using a logic module latch (DL1) with a latched input buffer macro (IBDL) as shown in Figure 4. This is equivalent to the I/O macro, IR. Data can be latched into the FPGA most efficiently in this configuration.

Because of the unique architecture of the I/O buffer latches in ACT 2, the input latch has zero external setup time. This means that the data may change just before the rising edge of the clock signal. Data must be held at the input of the latch for the time t_{HEXT}.

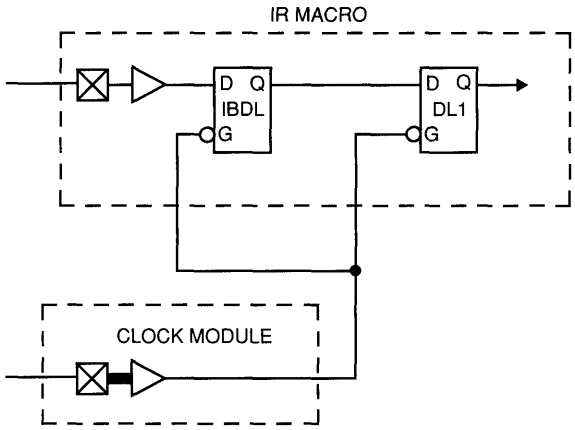
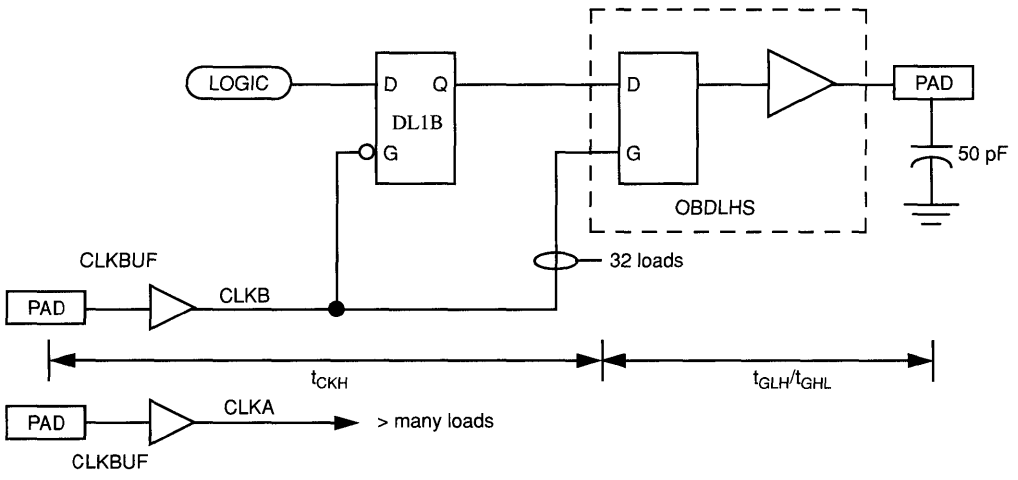


Figure 4. Master/Slave Registered Input

Dual Clock Approach

Faster clock-to-out can be achieved by utilizing the second global clock network as an I/O clock. In this configuration, shown in Figure 5, CLKA drives the synchronous circuitry on the device and CLKB drives the I/O master/slave latches. Both clock networks are operating at the same frequency and must be connected together outside of the devices. In this case, a 19.1 or 21.4 ns clock-to-out delay can be achieved.

Alternately, an INBUF can be used to drive the I/O latch, if fanout on the INBUF is kept less than four to achieve similar performance.



$T_{CO} = t_{CKH} + t_{GLH}/t_{GHL} = 10.2 + 8.9/11.2 \text{ ns} = 19.1/21.4 \text{ ns (rising/falling)}$

Figure 5. Dual Clock Master/Slave Registered Output

Implementation Rules

Actel strongly suggests following these rules when constructing flip-flops with the I/O latches:

1. Do not put combinatorial macros in the data path between master and slave latches. Added delay may prevent the flip-flop from operating properly. For inputs, combinatorial macros are allowed in the data path between the master and slave latches *only* if the logic module latch and the combinatorial latch are both combinable hard macros.
2. For outputs, do not connect the master latch output to any loads except the I/O latch D input.
3. Use a latch made from sequential logic modules for the master stage for outputs on the slave stage for inputs. Sequential module latches have better timing characteristics. They also allow combining to take place, which can improve the performance of the data being registered. The transparent-low sequential module latches are DL1B and DL1C.
4. Use net criticality to ensure that the net delay does not violate the setup requirements of the slave latch (as defined in Figure 1). Verify the timing conditions after place and route is complete. Note that an asymmetrical duty cycle on the clock signal (less than 50% low time) will provide more tolerance on the allowable net delay between latches.
5. Design to combine. The Action Logic® System (ALS) will automatically combine combinatorial logic into the D input of the DL1B latch if the combiner rules are met.

Three-Stating ACT Device I/O Pins for Board Level Testing

Application Note

Introduction

During board testing and debugging, it is frequently desired or necessary to place all device I/O pins into a three-state, high-impedance condition. This isolates the device from other devices that have common signal paths on a printed circuit board. The three-state condition also allows board testing for trace integrity or insertion damage to pins. It is usually more convenient to simultaneously three-state all I/O pins of a device with a built-in procedure rather than to create test vectors to force this condition by circuit function.

All Actel field programmable gate arrays (FPGAs) include a special operating mode to place all I/O pins in a temporary, three-state condition. Thus, each ACT™ device may be easily isolated from I/O signal paths of other devices on a circuit board, enabling a convenient board testing procedure. This capability, combined with Actionprobe® diagnostic tools, allows both single device and system board testing with a power and ease previously unavailable in programmable devices.

ACT 1 Three-State Procedure

Three-stating an ACT 1 device is easy using the unique debugging features of the ACT architecture. Three special pins on ACT devices are used for this function: MODE, SDI, and DCLK. To maintain the three-stating feature on ACT 1 devices, no user-defined output or bidirectional pins may be assigned to the SDI and DCLK locations. These pins should remain unassigned or should be defined as input-only locations. All other user-defined pins have no restrictions for their function; they may be assigned as input-only, output-only, or bidirectional pins. These pins can be temporarily three-stated for testing and debugging.

Figure 1 shows the sequence for three-stating an ACT 1 device. Seven data bits are clocked into the device using the SDI pin as data input and DCLK as the clock signal. The MODE pin distinguishes "test" mode from "normal" mode. The data sequence is {0001011}. After clocking the seventh bit, all user-defined pins are placed in a three-state condition until MODE is returned to logic low.

ACT 2 and ACT 3 Three-State Procedure

The same special purpose pins are used to three-state ACT 2 and ACT 3 devices: MODE, SDI, and DCLK. In contrast to the ACT 1 family, there are no restrictions on the assignment of user-defined functions to these pins. All user-defined pins have no restrictions for their function; they may be assigned as input-only, output-only, or bidirectional pins.

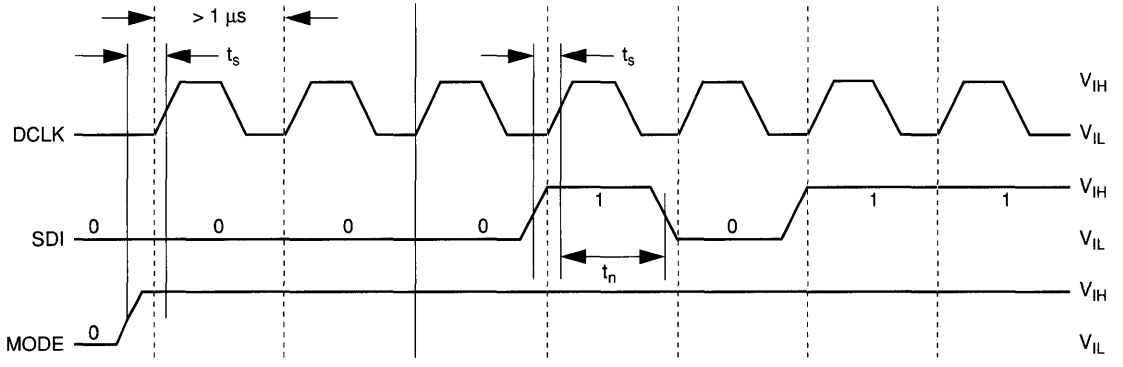
Thirty-two data bits are clocked into the device using the SDI pin as data input and DCLK as the clock signal. The MODE pin distinguishes "test" mode from "normal" mode. The data sequence for ACT 2 and ACT 3 devices is determined by the programmed state of the device. For unprogrammed, blank devices, the data sequence is

(LSB) 0000000000 0000000000 0000000111 10 (MSB)

For programmed devices, the data sequence is

(LSB) 0000000000 0000000000 0000000110 00 (MSB)

Load the data beginning with the LSB. After clocking the 32nd bit, all user-defined pins are placed in a three-state condition until MODE is returned to logic low. Use the same relative timing for the MODE, SDI, and DCLK inputs as shown in Figure 1.



Notes:
1. $0\ V \le V_{IL} \le 0.5\ V$; $3.0\ V \le V_{IH} \le VCC$
2. Test mode configuration is a low frequency (< 1.0 MHz) operation.
3. All setup and hold conditions (t_n, t^S) ≥ 250 ns.

Figure 1. ACT 1 Device Three-State Timing Diagram

Predicting the Power Dissipation of Actel FPGAs

Application Note

Introduction

Calculating the power dissipation of field programmable gate arrays (FPGAs) is similar to using the calculations for other CMOS ASIC devices, such as gate arrays and standard cells. The power dissipation depends on such factors as utilization, average operating frequency, and load conditions. In contrast, most PALs and PLDs have a fixed power consumption.

This application note discusses power dissipation and the concept of equivalent power capacitance. The general approach to calculating power in an ACT™ device will be described using equivalent power capacitance values for ACT devices. This general equation is useful if internal switching frequencies can be accurately determined. Since this is often difficult to do, a set of approximation curves based on average frequency rules of thumb are provided. The graphs provide an upper limit estimate for active power sufficient for most designs.

General Power Equation

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

Where:

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

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

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

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

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

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

Determining N and M depends on the design and the system I/O. An accurate determination of power dissipation comes from two components, static and active, which are considered separately.

Static Power Component

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

The power due to standby current is typically a small component of the overall power. For an ACT 3 device, the standby power is specified as 5 mWatts, worst case.

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

Active Power Component

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

Equivalent Capacitance

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

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

Where:

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

V_{CC} is the power supply in volts.

F is the switching frequency in MHz.

Equivalent capacitance is calculated by measuring I_{CC}active at a specified frequency and voltage for each circuit component of interest. Measurements have been made over a range of frequencies at a fixed value of V_{CC}. Equivalent capacitance is frequency independent so that the results may be used over a wide range of operating conditions. The results for ACT 1, ACT 2, and ACT 3 devices are given in Table 1.

Table 1. CEQ Values for ACT FPGAs

	ACT 1	ACT 2	ACT 3
Modules	6.3	7.7	8.2
Input Buffers	16.0	18.0	1.5
Output Buffers	25.0	25.0	2.3
Clock Buffer Loads	5.3	2.5	N/A
I/O Clock Buffer Loads	N/A	N/A	0.4
Dedicated Array Clock Buffer Loads	N/A	N/A	0.5
Routed Array Clock Buffer Loads	N/A	N/A	0.5 + fixed/ device

Finding the active power dissipated from the complete design requires solving Equation 2 for each component type. This requires the switching frequency of each part of the logic. The exact equation is a piecewise linear summation over all components as shown in Equation 3. For ACT 1 and ACT 2 devices:

$$\text{Power} = [(m * C_{EQ} * f_m)_{modules} + (n * C_{EQ} * f_n)_{Inputs} + (p * (C_{EQ} + C_L) * f_p)_{Outputs} + (q * C_{EQ} * f_q)_{clk\_loads}] * V_{CC}^2 \quad (3)$$

Where:

m = Number of logic modules switching at frequency f_m

n = Number of input buffers switching at frequency f_n

p = Number of output buffers switching at frequency f_p

q = Number of clock loads on the global clock network

f_m = Average logic module switching rate in MHz

f_n = Average input buffer switching rate in MHz

f_p = Average output buffer switching rate in MHz

f_q = Frequency of global clock

C_L = Output load capacitance

For ACT 3 devices:

$$\text{Power } (\mu W) = [(m \times 8.2 \times f_1) + (n \times 1.5 \times f_2) + (p \times (2.3 + C_L) \times f_3) + (q \times 0.5 \times f_4) + ((r_1 + 0.5\, r_2) \times f_5) + (s \times 0.4 \times f_6)] \times V_{CC}^2 \quad (2)$$

Where:

m = Number of logic modules switching at f_1

n = Number of input buffers switching at f_2

p = Number of output buffers switching at f_3

q = Number of clock loads on the dedicated array clock network

A1415:	q = 104
A1425:	q = 160
A1440:	q = 288
A1460:	q = 432
A14100:	q = 697

r_1 = Fixed capacitance due to routed array clock network

A1415:	r_1 = 60
A1425:	r_1 = 75
A1440:	r_1 = 105
A1460:	r_1 = 145
A14100:	r_1 = 195

r_2 = Number of clock loads on the routed array clock network

s = Number of clock loads on the dedicated I/O clock network

A1415:	s = 80
A1425:	s = 100
A1440:	s = 140
A1460:	s = 168
A14100:	s = 228

f_1 = Average logic module switching rate in MHz

f_2 = Average input buffer switching rate in MHz

f_3 = Average output buffer switching rate in MHz

f_4 = Average dedicated array clock rate in MHz

f_5 = Average routed array clock rate in MHz

f_6 = Average dedicated I/O clock rate in MHz

C_L = Output load capacitance in pF

Since all of the modules or inputs or outputs do not switch at the same frequency, a weighted average can be used. For example, a design consisting of 100 modules switching at 10 MHz and 200 modules switching at 5 MHz would have a weighted average frequency of:

$$f_{ave} = [(100 * 10) + (200 * 5)]/(100 + 200) = 6.67 \text{ MHz}$$

Determining Average Frequency

Determining the exact average frequency for a design requires a detailed understanding of the data input values to the circuit. Logic simulation can provide insight into average frequency, although simulation is limited by the percentage of real-time stimulus that can be applied. Fortunately, studies based on large numbers of ASIC designs have been made to determine rules of thumb for average switching frequency in logic circuits. These rules are meant to represent worst-case scenarios, hence their use for predicting the upper limits of power dissipation is generally acceptable. The rules given in Tables 2 and 3 for ACT 1 and ACT 2 devices. Table 3 gives the rules for ACT 3 devices. Using these rules, we can develop power estimates.

Table 2. Rules for Determining Average Frequency for ACT 1

1.	Module Utilization = 90%
2.	Average Module Frequency = F/10
3.	1/3 of I/Os are Inputs
4.	Average Input Frequency = F/5
5.	2/3 of I/Os are Outputs
6.	Average Output Frequency = F/10
7.	Clock Net Loading = 45%
8.	Clock Net Frequency = F

Table 3. Rules for Determining Average Frequency for ACT 2

1.	Module Utilization = 80% of combinatorial modules
2.	Average Module Frequency = F/10
3.	1/3 of I/Os are Inputs
4.	Average Input Frequency = F/5
5.	2/3 of I/Os are Outputs
6.	Average Output Frequency = F/10
7.	Clock Net 1 Loading = 40% of sequential modules
8.	Clock Net 1 Frequency = F
9.	Clock Net 2 Loading = 40% of sequential modules
10.	Clock Net 2 Frequency = F/2

Table 4. Rules for Determining Average Frequency for ACT 3

1.	Logic Modules (m)	= 80% of modules
2.	Average module switching rate (f_1)	= F/10
3.	Inputs switching (n)	= # I/Os used/12
4.	Average input switching rate (f_2)	= F
5.	Outputs switching (p)	= # I/Os used/15
6.	Output loading (C_L)	= 35
7.	Average output switching rate (f_3)	= F/2
8.	Dedicated array clock loads (q)	= fixed by device
9.	Average dedicated array switching rate (f_4)	= F
10.	Routed array fixed capacitance (r_1)	= fixed by device
11.	Routed array clock loads (r_2)	= 40% of sequential modules
12.	Average routed array switching rate (f_5)	= F/2
13.	I/O clock loads (s)	= # I/Os used
14.	Average I/O switching rate (f_6)	= F

Average Frequency Example

While some portions of a logic design switch at the system frequency, F, most of the logic switches at a reduced (or divided) frequency. Consider a 16-bit synchronous counter with a system input clock equal to F as shown in Figure 1.

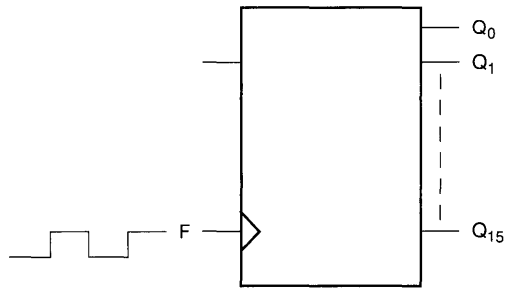


Figure 1. 16-Bit Synchronous Counter

Where:

The Q0 output is switching at F/2 (or $1/2^1$),

The Q1 output is switching at F/4 (or $1/2^2$),

The Q15 output is switching at F/65536 (or $1/2^{16}$).

The average frequency is:

$F_{ave} = 1/16 * (1/2^1 + 1/2^2 + ...1/2^{16}) \cong F/16$

Thus, the average frequency of an n-bit synchronous counter switching at F MHz is F/n.

Estimated Power

The rules in Tables 2 and 3 are applied to ACT 1 and ACT 2 devices. The resulting power components are detailed in Tables 5 and 6, and the total device power is shown in Figures 2 and 3. The graphs provide a simple guideline for estimating power. The tables may be interpolated when your application has different resource utilizations or frequencies. Table 4 details the rules applied to ACT 3 devices.

Table 5. Power Components for ACT 1 (Watts)

F (MHz)	Module Power	Input Power	Output Power	Clock Power	Total Power (watts)
A1010					
1	0.005	0.002	0.009	0.019	0.035
2	0.010	0.004	0.017	0.038	0.069
5	0.025	0.009	0.043	0.096	0.173
10	0.051	0.018	0.085	0.192	0.347
15	0.076	0.027	0.128	0.289	0.520
20	0.101	0.036	0.171	0.385	0.693
25	0.126	0.046	0.213	0.481	0.866
A1020					
1	0.009	0.002	0.010	0.035	0.057
2	0.019	0.004	0.021	0.071	0.114
5	0.047	0.011	0.052	0.176	0.286
10	0.094	0.022	0.103	0.353	0.572
15	0.141	0.033	0.155	0.529	0.858
20	0.188	0.044	0.207	0.705	1.144
25	0.235	0.055	0.258	0.882	1.430

Table 6. Power Components for ACT 2 (Watts)

F (MHz)	Module Power	Input Power	Output Power	Clock Power	Total Power (watts)
A1280					
1	0.011	0.013	0.021	0.027	0.072
2	0.023	0.025	0.042	0.054	0.144
5	0.057	0.063	0.105	0.136	0.360
10	0.113	0.125	0.210	0.272	0.720
20	0.227	0.250	0.419	0.544	1.440
30	0.340	0.375	0.629	0.815	2.159
40	0.453	0.500	0.839	1.087	2.879
A1240					
1	0.006	0.009	0.016	0.015	0.046
2	0.013	0.019	0.031	0.030	0.092
5	0.032	0.046	0.078	0.074	0.231
10	0.064	0.093	0156	0.149	0.461
20	0.127	0.186	0.311	0.298	0.923
30	0.191	0.279	0.467	0.447	1.384
40	0.255	0.372	0.623	0.596	1.845
A1225					
1	0.004	0.007	0.012	0.009	0.033
2	0.008	0.015	0.025	0.019	0.066
5	0.020	0.037	0.062	0.047	0.166
10	0.040	0.074	0.124	0.094	0.332
20	0.080	0.148	0.249	0.187	0.664
30	0.120	0.222	0.373	0.281	0.996
40	0.160	0.297	0.497	0.375	1.329
50	0.200	0.371	0.621	0.468	1.661

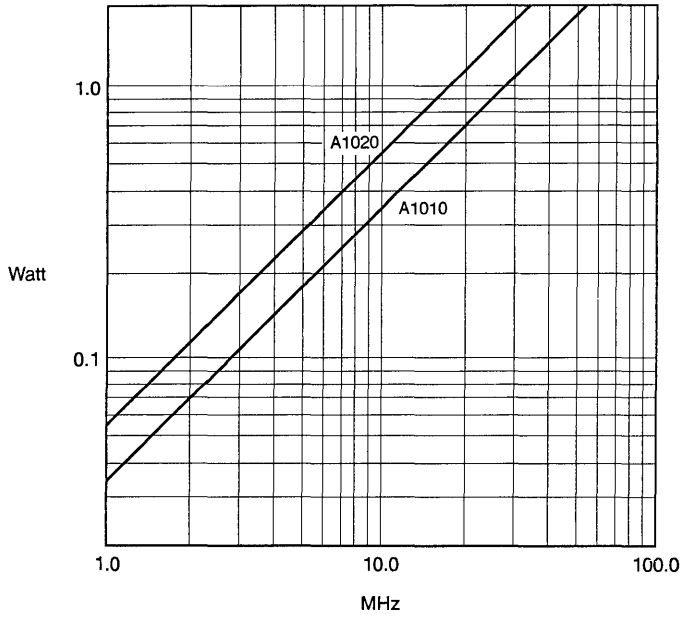


Figure 2. ACT 1 Power Estimates

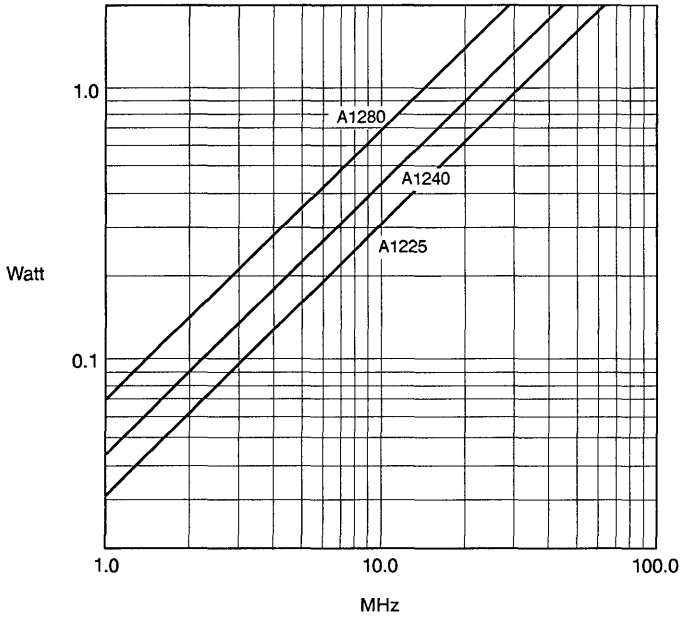


Figure 3. ACT 2 Power Estimates of Total Power (watts)

Board Level Considerations for Actel FPGAs

Application Note

Introduction

Simulating and debugging individual components is the first step towards verifying a system design. In some cases, the devices no longer behave as expected after they are integrated. Many factors, such as power, airflow, and transmission lines can introduce undesirable results in a system and ultimately impair system performance. This application note will explain how many of these factors should be treated when integrating ACT™ family field programmable gate arrays (FPGAs) in board-level designs.

Power Up

Actel FPGAs are nonvolatile and therefore require no external configuration circuitry on power up. However, at power up it does take a finite amount of time for the device to become stable and operate normally. For a V_{CC} slew rate of ~30 ns/V, it takes approximately 250 µs for the device to become fully operational. Power up time varies with temperature, where cold is worst case. At power up, the state of all flip-flops is undefined. Refer to *A Power On Reset Circuit for Actel Devices* for more information.

Operating Environment

ACT family FPGAs must not be operated outside of the recommended operating conditions as described in the Absolute Maximum Ratings section of the ACT data sheets. Exposure to maximum rated conditions for prolonged periods may result in irreparable damage to the device.

As can be seen in the timing derating section of the ACT datasheets, variations in voltage, temperature, and process will affect device performance. You must take care to design systems so that the effects of these variables, from best to worst case, will not cause timing problems during interchip communications.

Thermal Considerations

An often overlooked test case examines ambient temperature effects on FPGA performance. The Action Logic® System reports timing as derated by thermal junction temperature. This is the temperature found at the junction of the die and the package casing. Additional derating must be applied for ambient to junction temperature effects (θja). The value for θja will vary from package to package. The units for θja are °C/W. This implies that the power dissipation must first be calculated to determine the value for θja. (Please refer to the Power Dissipation section of the ACT datasheets.) Once θja is calculated, this temperature should be added at the junction to case temperature (θjc). This is the temperature value to be used for the temperature derating portion of the timing analysis. Because θja is a function of ambient temperature, the use of a fan can decrease the θja value. This can be seen by observing the difference between the θja value in still air and at 300 ft./min., as shown in the Package Thermal Characteristic section of the datasheets.

Ground Bounce and Switching Characteristics

Actel defines simultaneously switching outputs (SSO) as any outputs that transition in phase within a 10 ns window. The resultant ground bounce produced is a function of the number of outputs switching simultaneously and the capacitive loading of the outputs.

Table 1 shows the recommended SSO limit for ACT 1 and ACT 2 FPGAs. To measure worst-case ground bounce, the I/Os are placed adjacently and are driving a 50 pF load. The observed ground bounce is less than 1 V, with a pulse width of less than 2 ns.

Exceeding the recommended limits results in a larger ground bounce. However, the width of the ground bounce pulse prevents it from being recognized by most discrete digital receivers. Refer to the specifications of the external receivers for verification.[1]

Should you have the need to switch more signals than recommended, the SSOs should be distributed evenly around device. This will reduce the amount of mutual inductance produced between adjacent I/Os and thus reduce the ground bounce.

Table 1. Recommended SSO Limits for Actel FPGAs

Device	Package	Maximum Recommended SSOs (50 pF load)
A1010A/A1020A	44 PLCC	16
A1010A/A1020A	68 PLCC	24
A1020A	84 PLCC	32
A1010A/A1020A	84 PGA	32
A1010A/A1020A	100 PQFP	32
A1280	PG176, PQ160	64
A1240	PG132, PQ144	48
A1240/A1225	84 PLCC	32
A1225	100 PGA, PQFP	32

Buffers may also be inserted in the path before the output buffer. This will reduce the possibility of adjacent signals switching within 10 ns of each other. The ALS Timer should be used to verify the delays.

Power and Ground Pins

Actel FPGAs are designed with ample power and ground pins on the device resulting in minimal ground bounce characteristics during I/O switching. A ground pin is provided for approximately every eight I/Os (please refer to package drawings found in the *Actel FPGA Data Book and Design Guide* for further detail). Unused I/Os cannot be programmed to act as ground pins.

Table 2 describes the function of each of the power and ground pins found on the devices.

Table 2. Power and Ground Pin Functions

V_{CC}	Device operating power.
V_{PP}	Device programming voltage. Should be tied to V_{CC} during normal operation.
V_{SV}	Super voltage supply. $V_{SV} = V_{PP} + 3$. V_{SV} provides a precise lower limit on the voltage that is applied to a fuse during programming. During normal operation $V_{SV} = V_{CC}$.
V_{KS}	V_{KS} provides voltage supply for keepers (keepers maintain floating tracks at $V_{PP}/2$ during programming and prevent them from leaking down to GND). During programming, $V_{KS} = V_{PP}/2$. During normal operation, V_{KS} = GND.
GND	Supplies GND to the array (all channels).

Slew Rate

ACT 1 and ACT 2 outputs operate at high slew rate only. Figures 1, 2, and 3 illustrate the slew rate characteristics for high slew rate outputs, both loaded and unloaded, for an A1280 PG176 device.

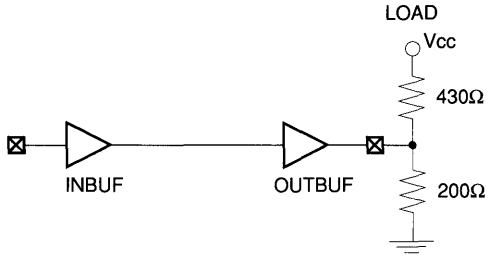


Figure 1. The Test Circuit

Device I/O

The I/Os on each Actel FPGA are nonrestrictive. Any pin shown as an I/O in the pin description list can be assigned to be either input, output, bidirectional, or tri-state. ACT 2 FPGAs add latch capabilities to each of these options. Actel FPGAs do not incorporate any type of internal pullup on the I/Os. Inputs must not be left floating, since this may cause irreparable damage to the device. Any unused I/Os will be configured as active low outputs by the Action Logic System. The sink and source capabilities of individual outputs for ACT 1 and ACT 2 FPGAs have been extrapolated from V-I curves and are shown in Table 3. The V-I curves for ACT 1 and ACT 2 are shown in Figures 4 through 9.

Table 3. Worst-Case Sink and Source Current for Actel FPGAs

	ACT 1	ACT 2
TTL	8 mA	10 mA
CMOS	4 mA	6 mA

I/Os may be tied together externally to increase drive capability. The outputs must switch within 4 ns of each other, thus they should be placed as close to each other as possible. (Please refer to device die bonding diagrams for pad locations.) The switching times can be verified with the ALS Timer.

Decoupling Capacitors

Actel recommends the use of decoupling capacitors with all FPGA devices. We suggest a minimum of one capacitor per side, located next to power and ground. A 0.1 µF monolithic ceramic type is sufficient.

References

1. David Shear, "EDN's Advanced CMOS Logic Ground-Bounce Tests," *EDN*, March 2, 1989, p. 88.

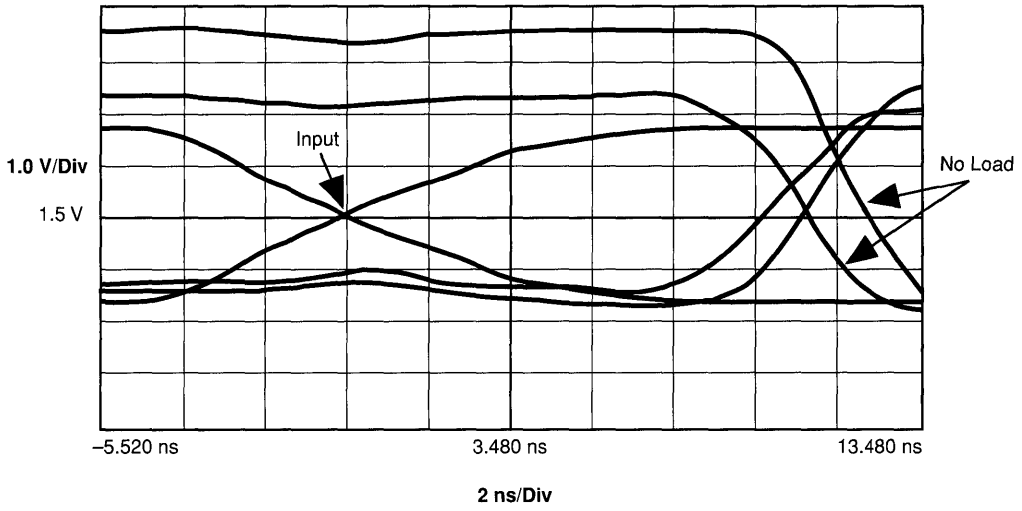
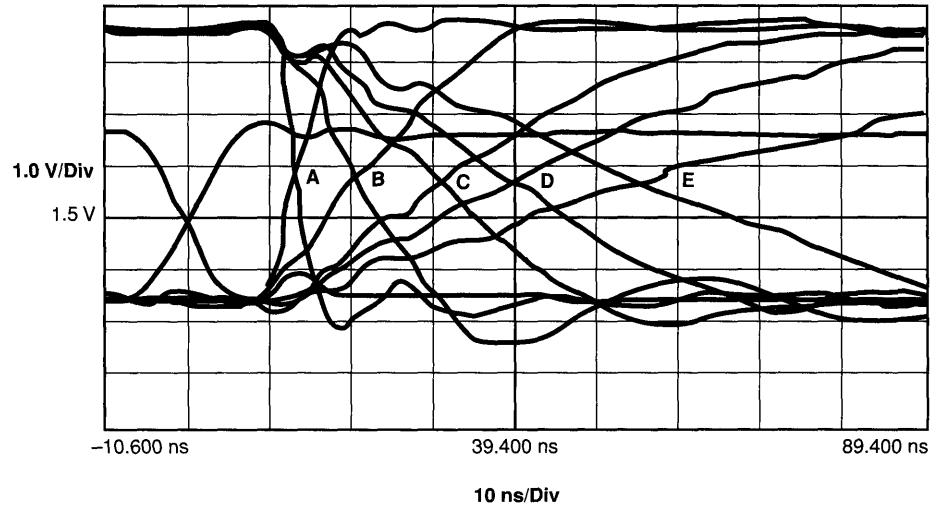


Figure 2. No Load Versus Load—Fast Slew



A = "No Capacitive Load" (~25 pF jig + 9 pF scope)
B = 82 pF
C = 200 pF
D = 300 pF
E = 560 pF

Figure 3. Capacitive Loading Versus Slew

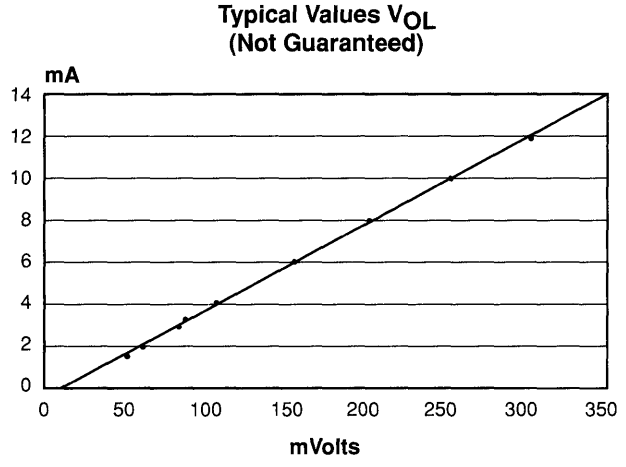


Figure 4. A1020A Typical Sink Current

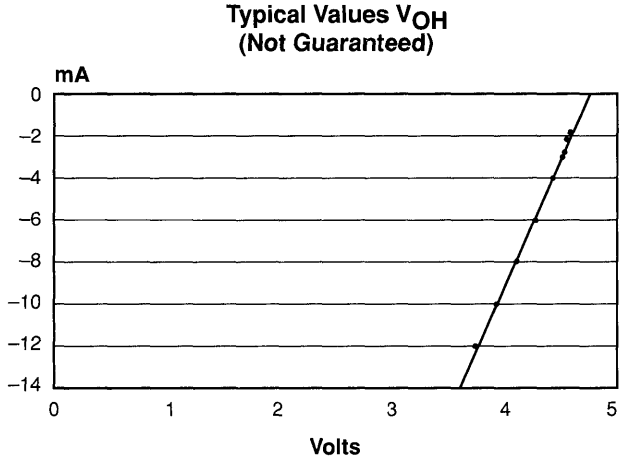


Figure 5. A1020A Typical Source Current

8

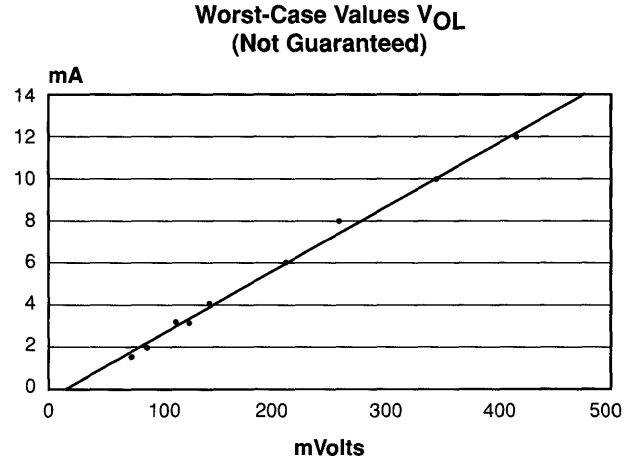


Figure 6. A1020A Worst-Case Sink Current

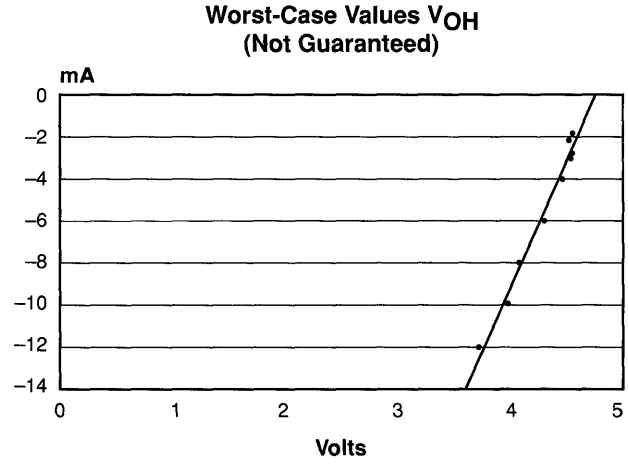


Figure 7. A1020A Worst-Case Source Current

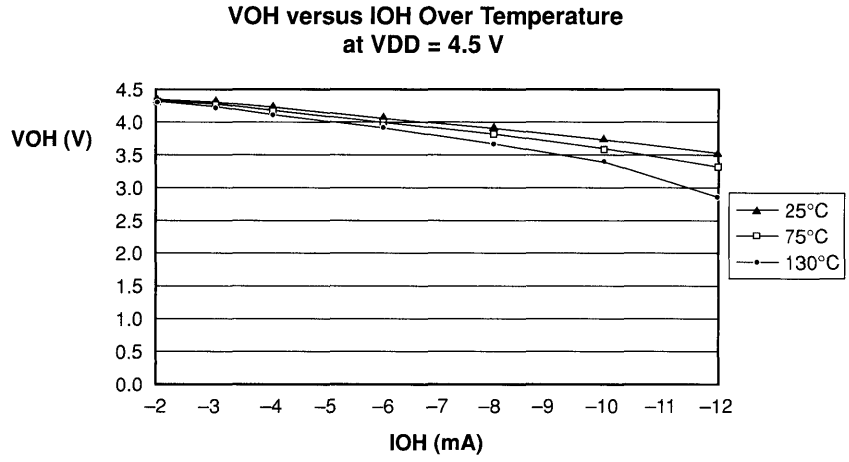


Figure 8. A1280 Typical Source Current

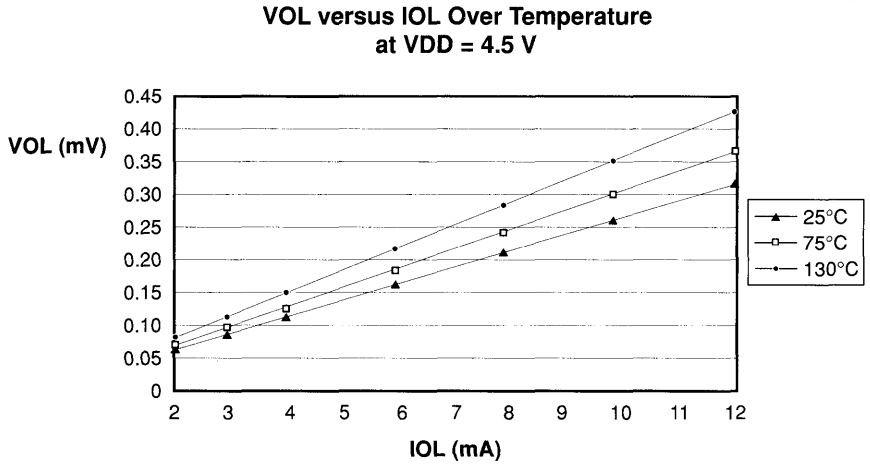


Figure 9. A1280 Typical Sink Current

A Power-On Reset (POR) Circuit for Actel Devices

Application Brief

The state of a system at startup is an important consideration in designing a circuit. It is usually desirable to provide an input signal at startup to reset synchronous circuitry. Otherwise, the system may initially operate in an unpredictable fashion because flip-flops are not designed to power-on in any particular state. Figure 1 shows a typical power-on reset (POR) circuit where the series resistor, R1, is omitted for TTL circuits.

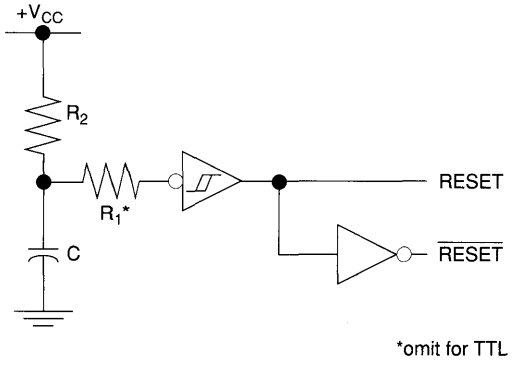


Figure 1. Power-On Reset Circuit

This resistor is necessary with CMOS implementations to prevent damage to the device when power is removed from the circuit. Otherwise, the capacitor will try to power the system via the CMOS input gate protection circuit. A Schmitt trigger (40106, 74LS14) may have its advantages in making the RESET signal switch off cleanly. The hysteresis symbol shown in Figure 1 indicates an inverter with a Schmitt trigger input such as the CMOS 40106 hex inverter. The following sections describe the power-up conditions of an Actel device and a recommended POR circuit.

Behavior of ACT™ 1 FPGA Inputs

During power-on, the +5 V logic supply rail of a system typically rises from 0 to +5 V in 50 ms or less. Because regulator outputs are usually current limited during this transition, the rise time is more or less linear, with a slope in the range of 0.1 V/ms to 5 V/ms. Each ACT 1 FPGA has a universal pad driver design that may be configured as an input, output, three-state output, or bidirectional input/output. This configuration of the pad driver is accomplished by programming antifuses in the pad driver circuitry.

As the +5 V logic supply rail passes through the region from approximately +2.2 V through +2.5 V, pad drivers that have been programmed as inputs may behave temporarily as outputs that are in the logical '1' state. Thus, these input pins will temporarily source current (approximately 8 to 10 mA, if not otherwise limited) into whatever driver is connected to them. They will be sourcing this current from the +5 V logic rail, which at this time is at +2.2 to +2.5 V. This duration is a function of the power rail rise time. For +5 V rails that come up quickly, at 5 V/ms, the duration of the behavior will be approximately 60 µs. For supply rails that rise slowly, at 0.1 V/ms, the duration of the current sourcing behavior will be 3 ms. In the former case, the Actel input can deliver as much as 0.6 µC to the circuit that drives it; in the latter case, the charge is as much as 30 µC. For many driver circuits, this amount of charge is insignificant; however, for others it may be unacceptable.

Inserting a series resistance of sufficient size in the Actel input line can limit the effect of this behavior. In the case of the POR circuit in Figure 2, the series resistance must be chosen to keep $\Delta V \leq 1V$. This guarantees that the POR remains at a logic '0' following the irregularity when the logic supply rail is at approximately 2.5 V, where $\Delta V/\Delta t = i/C$ is the voltage rise time of the capacitor. The capacitor is charged through a resistor to the 5 V logic supply rail, and the diode across the resistor is used to discharge the capacitor at power-off. For a power rail rise time of 0.1 V/ms, the duration of this behavior will be approximately 3 ms. This means that for a POR capacitance of 0.1 µF, the current out of the Actel device input must be limited to

$$i = C\Delta v / \Delta t = (0.1\ \mu F * 1\ V) / 3\ ms = 33\ \mu A$$

which can be achieved using a resistance of 2.24 V/33 mA = 68 kΩ.

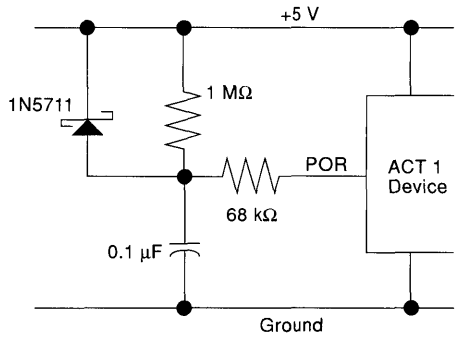


Figure 2. Power-On Reset (POR) Circuit with Current-Limiting Resistor

Furthermore, for drivers that cannot accept a source of current at their outputs or for a multiple-source data bus, it is strongly recommended that the bus driver(s) be three-stated during POR.

Summary

Use these methods for avoiding POR problems. The transistors for these devices are turned on at approximately 0.7 V, their threshold voltage, while the circuit is functionally operational at a voltage level of approximately 3.3 V. The global routed clocks in Actel devices can also be used as resets for synchronous circuits when connected to either CLEAR or PRESET inputs of synchronous macros for the ACT 1 family, and the CLEAR input for ACT 2 and ACT 3 families.

Simultaneously Switching Output Limits for Actel FPGAs

Application Note

Introduction

For high performance field programmable gate arrays (FPGAs) with many I/Os, the allowable number of Simultaneously Switching Outputs (SSOs) for each device is an important issue for system designers. The limits for SSOs depend on factors such as package type and die size. Use the following guidelines to help ensure reliable system designs.

System Noise and Transients

Noise generated by off-chip drivers is a major concern in FPGA design for high-performance systems. Noise is closely related to interconnections and increases as a function of fast rise times, large total chip currents and die dimensions, and small spacing between components on-chip and onboard. As clock frequencies and die dimensions increase, signal wavelengths become comparable to wire lengths, thereby making better antennas. Reduction in the spacing between circuits leads to an increase in the capacitive, inductive, and resistive couplings. The voltage (IR) drop across the power lines becomes more significant as the current densities increase and the feature size decreases. With larger amounts of current switching, the inductive noise associated with the power lines also increases. These current transients may generate large potential drops because of the inductance of the power distribution network and is referred to as simultaneous switching (ΔI) noise.

When several circuits switch simultaneously, the current supplied by the power lines can change at a very fast rate and the inductive voltage drop along the line can cause the power supply level to fall. This voltage drop is proportional to the switching speed, the number of drivers switching simultaneously, and the effective inductance of the power lines. This effect can be summarized by Faraday's Law, which states that any change in the magnetic flux will be confronted by an opposition, a self induced EMF, determined by the rate of change in the total magnetic flux, represented by this equation:

$$EMF = d\Phi/dt = LdI/dt$$

This reduction in the supply level diminishes the current drive of a circuit, increases the delay, and may lead to the spurious switching of the receiver.

SSO Recommendations

Actel defines SSOs as any outputs that transition in phase within a 10-ns window. The output current of these drivers is shown in the *Actel FPGA Data Book and Design Guide*. The amplitude and duration of the ground bounce is a function of the number of outputs switching simultaneously and the capacitive loading of the outputs.

Table 1 shows the recommended SSO limits for the Actel FPGA family. These may vary because the amount of ground bounce that an application can tolerate is difficult to determine. Worst-case conditions are simulated by placing the I/Os adjacent to each other while driving 20 pF, 35 pF, and 50 pF loads. The observed ground bounce is less than 1.5 V, with a pulse width of less than 2.0 ns. This is within the acceptable limits of the dynamic threshold of 74F, 74LS, and ACT families. Results from the EDN Special Report on Ground Bounce Tests by David Shear[1] show a plot of the ground bounce pulse amplitude versus pulse width duration (ns) for different logic families. This article shows that as the input pulse width gets shorter, the voltage must be higher to affect the output of the device. Exceeding the recommended limits may result in larger ground bounce. The output drivers should be placed separately if more outputs must be switched simultaneously. This arrangement reduces the mutual inductance produced between adjacent I/Os resulting in a lower ground bounce. If necessary, buffers may also be inserted in the path before the output buffer. This will reduce the probability of adjacent drivers switching within 10 ns of each other.

References

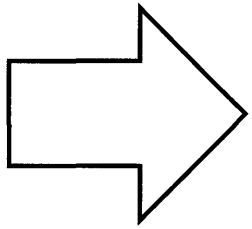
1. David Shear, "EDN Special Report on Ground Bounce Tests," *EDN*, April 15, 1993, p. 120–134,.

8

Table 1. Recommended SSO Limits for Actel FPGAs

Device	Package	Maximum Recommended SSOs for Loads		
		20 pf	35 pf	50 pf
A1010A/A1020A	44 PLCC	40	22	16
A1010A/1020A	68 PLCC	60	34	24
A1020A	84 PLCC	80	45	32
A1010A/1020A	84 PGA	80	45	32
A1010A/A1020A	100 PQFP	80	45	32
A1280	PG 176, PQ 160	160	90	64
A1240	PG 132, PQ 144	120	68	48
A1240/A1225	84 PLCC	80	45	32
A1225	100 PGA, PQFP	80	45	32
A1425	84 PLCC			

Application Examples and Design Techniques



Hints and Tips for Better Actel Designs I

Application Note

Actel field programmable gate array (FPGA) designs benefit from the same digital design techniques that are effective for other types of devices in a digital system. PLD, TTL, and FPGA-based designs use many of the same methods to implement common functions. However, there are many architectural differences between these device types that require distinct techniques for optimum results. Use the following techniques to improve the performance of Actel FPGA designs. These hints and tips provide a wide range of ideas to be considered during the design flow. In most cases, more information is available in the *ALS User's Guide*.

Reducing Unnecessary Logic

The macro library for each ACT™ family contains a wide variety of hard combinatorial macros with many combinations of inverted (bubbled) input and output pins. For example, there are five different four-input NAND gates, having zero to four inverted inputs. Use each of these hard macro types as necessary to eliminate the use of inverters. This practice reduces the logic module count and increases speed and efficiency by removing unnecessary logic levels.

Using De Morgan's Law

Apply De Morgan's law to reduce combinatorial logic expressions to their least complex form. Reducing the logic equations reduces logic module use and complexity. In many cases, lower fan-in macros may be used as a result of simpler logic equations. The overall routability usually increases as a result of simplifying the design's implementation.

Pipelined Design

For highest performance throughput, use pipelined design techniques if possible. Pipelined designs may use more logic modules to implement a function, but they will have higher performance. If high performance is required and the device capacity is sufficient, use pipelining instead of serial data processing.

Balancing Logic Levels

Balance the number of logic levels between registers in a register-to-register design. For a data bus that traverses ten logic levels and three registers, separate the number of logic levels between registers as equally as possible. For example, put three logic levels between the first and second registers, three between the second and third registers, and four between the third and fourth registers. Balancing the logic levels enables the highest overall system clock frequency.

Using Synchronous Enable Signals

The hard macro library contains many flip-flops and latches with synchronous clock enable inputs. The input of these macros enables and disables the clock or gate pin. Using this input is more reliable and efficient than "gating" the clock signal. The enable input has definite setup and hold times that are easy to calculate and control. In contrast, determining and managing the relative arrival times of the clock signal and a gated enable signal can be difficult and tedious. The use of the enable inputs removes the costly need for extra "clock gate" logic modules.

Designing with the DFM8 and CM8 Macros

For ACT 1 designs, the CM8A macro represents the complete function of a logic module. For ACT 2 and ACT 3 designs, the CM8 macro represents the complete function of the combinatorial module (C-module). The DFM7A and DFM7B represent the complete sequential module (S-module) for ACT 2 designs and the DFM8A and DFM8B are the S-module macros for ACT 3. Use these macros to implement functions not available in the existing hard macro libraries. See the *Actel Family Macro Library Guide* for a complete description of the macros.

Using Look Ahead Techniques

Actel FPGAs benefit from design techniques such as look ahead generation. Good examples of the use of this technique to improve a design are the adders in the soft macro library. Carry look ahead generators are used to greatly increase the maximum operating throughput of these functions.

Using S- and C-Modules for Sequential Macros

For ACT 2 and ACT 3 devices, both S- and C-modules are utilized to implement flip-flops and latches. These devices contain approximately equal numbers of each type of module. Designs that are flip-flop intensive may use a disproportionately large number of S-modules, which may decrease the chance for success of the placement and routing tools.

Hints and Tips for Better Actel Designs II

Application Note

Generally, Actel Field Programmable Gate Array (FPGA) designs benefit from the same digital design techniques effective for other types of devices in a digital system. PLD, TTL, and FPGA-based designs share many methods of implementing common functions. However, there are many architectural differences between these device types that require distinct techniques for optimum results. Use the following techniques to improve the performance of Actel FPGA designs. These hints and tips provide a wide range of ideas to consider during the design flow. In most cases, more information is available in the *ALS User's Guide*.

1. **I/O Pin Assignment**
   The overall performance of the design is largely influenced by the I/O pin placement. Well-placed I/O pins usually lead to a successful layout of the logic modules. If possible, use automatic assignment for all I/O pins. Usually, designs with automatically assigned I/O pins have higher performance than those with manual assignments. For more information on I/O assignment, refer to Chapter 7 in the *Actel User's Guide, Release 2.2*.
2. **Criticality**
   A common mistake made in assigning criticality to the nets in a design is to layout the device with no criticality assignment initially and then to iteratively assign criticality to nets that do not meet timing requirements. A better approach is to determine the critical paths in the design initially. The Timer may be used with pre-layout delays to assist in finding the critical paths. Finally, add the critical nets to the criticality file.
3. **Incremental Placement**
   The Incremental Placement option in Layout allows minor modifications to a design with minimal or no impact on the timing of the device. Depending on the selected incremental placement strength, place will attempt to preserve the original placement of common portions of a design.
4. **Global Clock Network**
   Low-skew, high fanout clock networks are provided in the Actel architecture. These dedicated networks exist in all Actel devices. The clock networks are used by selecting the clock macros, such as CLKBUF or CLKINT, from the library. Global clocks guarantee the minimal skew possible for high fanout signals. Global clock networks use independent short routing channels that prevent ALS from placing clock signals on long vertical or horizontal tracks, which may cause excessive delay and skew. In general, it is highly recommended to use global clock networks to drive high fanout signals such as enable and reset lines, also.
5. **Delay Verification**
   Actel provides two efficient techniques to measure timing delays—the ALS Timer and backannotation to a simulator. The Timer and backannotation provide a simple way of finding the expected delays on the chip, the maximum operating frequency of the chip, and the internal device delays. Both have the capability of operating with worst-case conditions as well as typical and best-case conditions. The Timer identifies point-to-point delays within the device. Simulation with backannotated delays verifies the functionality of a circuit in addition to its speed.
6. **Combinable ACT 2 and ACT 3 Macros**
   The ACT™ 2 and ACT 3 device architectures are based on two types of logic modules, the C-module and the S-module. These architectures allow the combination of gates with certain flip-flops and latches, which allow the implementation of a logic function with fewer modules and shorter propagation delay. If there is a combinatorial gate solely connected to one of these flip-flops or latches, the ALS software will merge the two macros in a single S-module. Consult Table 3-2 in the *ALS User's Guide* for a list of combinable macros.
7. **90% Full Chip Versus 90% Empty Chip**
   Ordinarily, it might be hard to understand that the operating frequency of a 90% utilized device can be better than a 90% empty chip. Placement and routing in a 90% empty chip might give better results than a 90% utilized chip due to the extra logic resources and routing tracks available. However, even though a 90% empty chip underutilizes the device, signals might be routed onto long horizontal or vertical tracks. Some signals may even be unroutable. In a 90% empty chip, the placement software may disperse the macros throughout the chip depending on the I/O placement, creating long distances in between them. In this case, the software will be forced to use long vertical or horizontal tracks or may fail to complete. Therefore, it is recommended that test circuits occupying less than 10 to 15% of the chip should not be used to predict the operating frequency of the chip. Instead, the test circuit should use at least 50% of the chip.
8. **Fanout Management**
   ALS automatically calculates the fanout for every net in a design. The built-in fanout limitations guarantee more efficient design routability and better timing performance. ALS reports warning and error fanout messages as follows.
   - Fanout errors are reported when any net (except global clock networks) exceeds the limit of 24 loads or when a critical net exceeds its limit. The fanout limitations and net loading restrictions are listed in Table 1.
   - Critical net fanout limits are more restrictive in order to achieve the speed requirements for the nets. As fanout increases, the average delay on the net increases.

– Fanout warnings are reported when an ordinary net exceeds the recommended limit of 10 or when a critical net exceeds its recommended limit. The recommended limit of 10 is a warning only. If many nets in a design have fanout of 10 or greater, routability is negatively affected. If a large number of nets in the design have a high overall fanout, such as 10, many long routing tracks will be used, which may preclude other net connections. Fanout warnings need only be corrected if a large number of nets exceed a fanout of 10. If nets are noncritical, a fanout of 10 or greater is acceptable.

Table 1. Net Loading Restrictions

Net Criticality	Fanout Warning Limit	Fanout Error Limit
F (Fast)	net loads > 3	net loads > 6
M (Medium)	net loads > 8	net loads > 12
default	net loads > 10	net loads > 24

In general for best routability performance, the average fanout on a net should be about three loads. This is not a required value but it is recommended to keep the fanout low for best results.

A TTL Designer's Guide to FPGA Design

Application Note

Actel field programmable gate arrays (FPGAs) offer many advantages over traditional technologies such as TTL. FPGAs integrate large amounts of logic into one device, increasing reliability and reducing board space and power. For example, a single A1280 FPGA holds the equivalent of 165 MSI TTL devices (assuming 70 gates per MSI device). Designs require smaller, simpler printed circuit boards (PCBs), since most of the design's connections are made inside the FPGA by the 100 percent automatic place and route software. This beats costly PCB design and fabrication.

It's Easy to Start

You may be familier with TTL components and see some of the advantages of Actel FPGAs, yet not realize how easy it is to begin using them. You don't have to know anything about the internal workings of the FPGA. In fact, the schematic entry and simulation process is the same as it is with TTL.

Actel Library

Actel provides a library with the system for popular schematic design tools. The library contains both hard macros and soft macros. Hard macros are similar to SSI components. They form the basic functional building blocks, such as gates and flip-flops. Many Actel hard macros are identical in function to TTL parts though they have different names. The *Actel FPGA Data Book and Design Guide* contains a cross-reference guide showing the names of hard macro components that match functionally to TTL components.

Soft macros are more complex functions built from a number of hard macros. They include counters, decoders, and adders of various sizes. Some of the soft macros in the library have identical functions to MSI TTL parts. These may be identified by names that begin with TA instead of 74. The rest of the name matches that of the TTL name. Other soft macros offer generic logic functions.

Creating Custom Soft Macros

All soft macros are easily copied and modified, and there is no limit to the number of custom soft macros you can add to the library. Should you need a TTL function for which there is no near equivalent in the library, it is easy to build it from scratch. Simply copy the schematic as shown in the TTL databook using components such as gates and flip-flops from the Actel hard macro library.

Also, make use of logic enhancers such as ACT'x'press™ and ABEL™ FPGA. Logic enhancers may be used to optimize individual soft macros for area or speed. As you gain familiarity with the Actel hard macros, you will find instances where you can create a more efficient design than that found in the TTL databook. For example, if the book shows an AND gate driving an OR gate, you may find a single hard macro containing both the AND and OR functions. Using such multifunction macros is efficient because you get better, faster logic from a macro. Compare the 74AS161 counter schematic from a TTL databook with the TA161 from the Actel library in Figure 1.

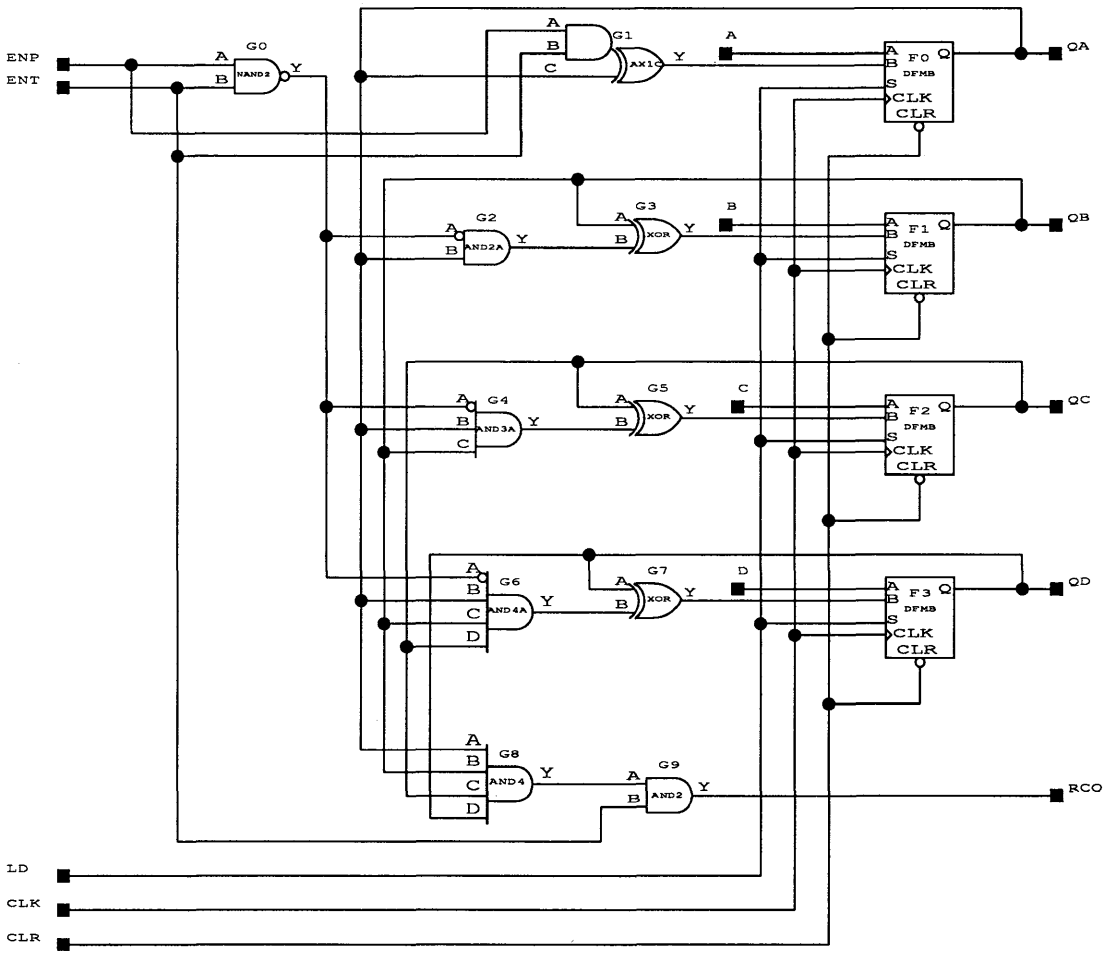


Figure 1. TA161 Counter

Three-State Design

Many TTL parts have three-state outputs allowing them to be connected to a common bus. Three-state functions don't work well with ASICs or FPGAs because they tend to be slow and inefficient. You don't have to give up using buses in your designs. Simply implement them using multiplexers as shown in Figure 2. Multiplexers are efficient on Actel parts. For example, you can create an eight-bit bus with four possible drivers using less than 3 percent of an A1010, Actel's smallest part.

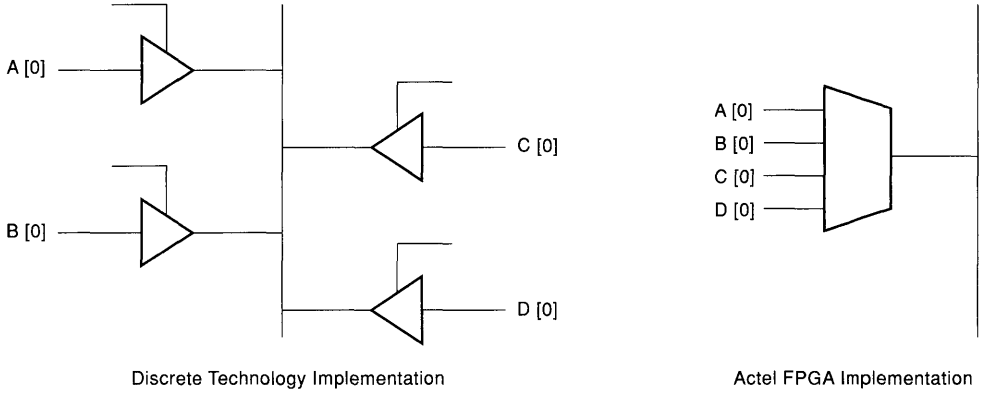


Figure 2. Least Significant Bit of a Bus with Four Possible Drivers

Design Tips

If you use a soft macro, but don't need all the outputs, you don't need to modify the macro. The Actel software contains a program that will automatically eliminate any unused modules before the design is placed and routed.

If you use a soft macro or a hard macro that has inputs you don't need, the situation is different. The software won't allow inputs to be left unconnected, so simply tie unused inputs to V_{CC} or GND. A better solution is to select a hard macro that only has inputs you need or to modify the soft macro to eliminate the unused function.

For example, the TA161 counter has a load function and four data inputs. Rather than tying off these pins, a better solution is to make a copy of the counter and modify it as shown in Figure 3. This will allow the wiring resources on the chip to be used for things other than power and ground connections.

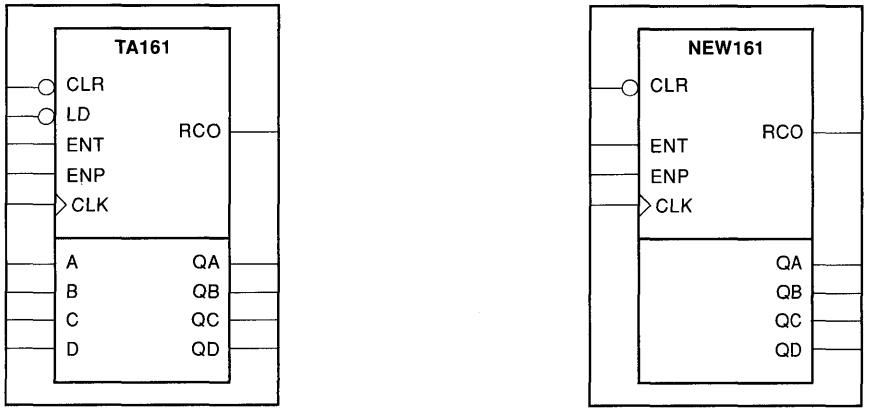


Figure 3. A Library Symbol and a Modified Symbol

Simple State Machine Design Techniques

A common TTL design technique is the use of a Jump/Reset counter (74161) to generate a delay sequence. The counter is loaded with a fixed value, then counted until the terminal value is reached. This results in a fixed delay depending on the loaded value. Figure 4 illustrates this technique with a loaded value of seven, which results in a delay of eight. This is a popular technique in TTL because the function is implemented in a single package and the value is easy to change by modifying the load value.

Most FPGA libraries include a 74161 soft macro, which designers often use the same way they would when designing with TTL. This technique, however, results in an inefficient function when using FPGAs. For example, the 74161 requires 22 logic modules and three levels of logic delay in the ACT™ 2 family. Alternately, you could use a simple shift register to generate a delay that requires only N modules (for a delay of N states) and requires only a single level of logic. The amount of interconnect is much reduced, also requiring about two connections in the 74161 and only about 2N for the shift register. The FPGA implementation of this delay function is shown in Figure 5.

The approach to more complex state machine designs with TTL also rely on simple functions like counters, decoders, and random logic. One common example is a counter with a decoder on the output. The counter is used to hold the state information and is sequenced by the control inputs. Jumps can be made to different values based on input conditions, and counting can be activated to advance the state machine or make it hold in the existing state. An example of a state machine implemented using this technique is given in Figure 6 along with the state diagram. The state machine starts in the reset state and advances when S is high. When it reaches state 3 it makes a transition to state 5 and 7 depending on the input signal L. When it reaches state 10, the state machine waits until the G signal is active and then advances to state 12, when it is reset. Notice the use of the load inputs to determine the transition to state 5 or 7, and the use of the decoder outputs to detect states 10 and 12. This ad hoc approach is not easily mechanized and requires experience and trial and error to get correct results. The TTL implementation is a good one since it uses common functions and a small number of packages. If implemented in the Actel ACT 2 FPGA, the module count is over 30 and five levels of delay are required.

A better technique for implementing state machines in FPGAs uses a shift register instead of a counter and decoder. Each state is represented by a register and the active state register will contain a logic one with the other states containing a logic zero. A decoder is not needed, since the states are distinct and thus already decoded. Figure 7 shows the resulting FPGA design. It requires only 13 logic modules and one level of logic delay.

Conclusion

These examples illustrate that, to take the best advantage of the capabilities of FPGAs, designers need to rethink some of the common design techniques used with TTL devices.

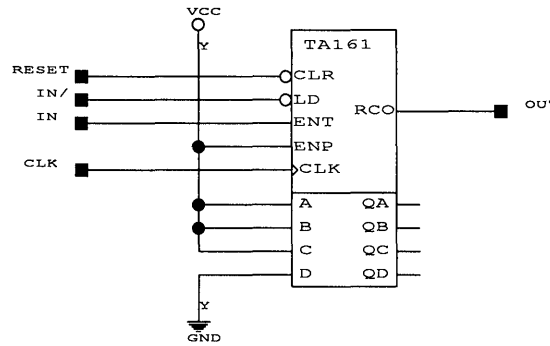


Figure 4. Eight Clock Delay Using 74161

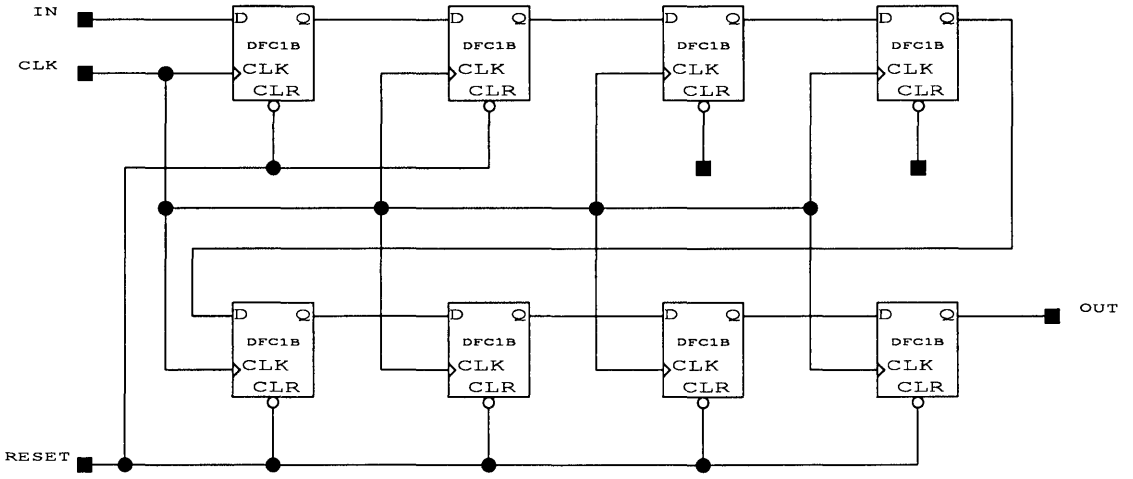


Figure 5. Eight Clock Delay Using FPGA

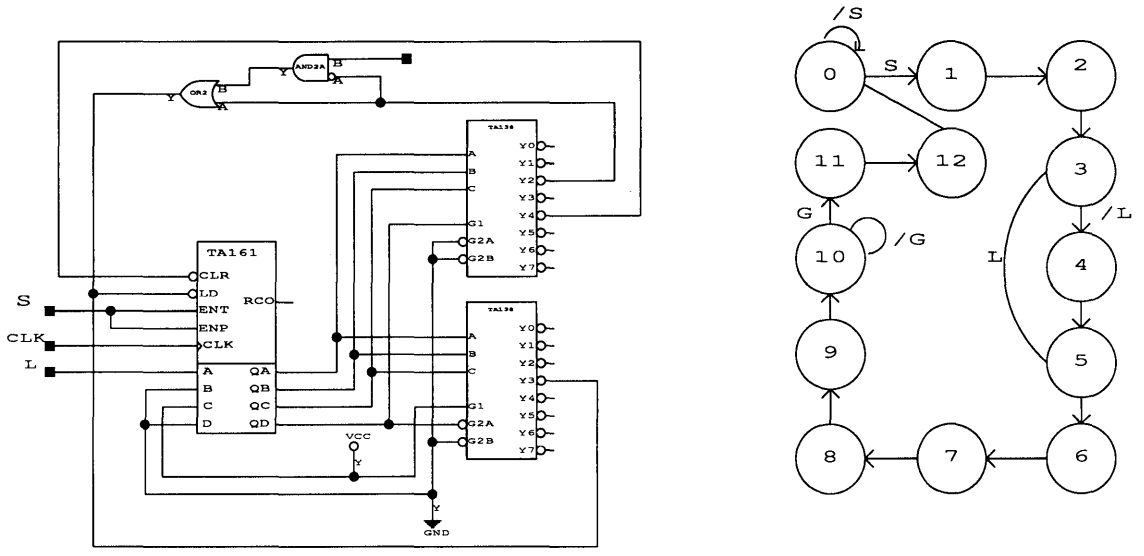


Figure 6. State Machine Using TTL Technique

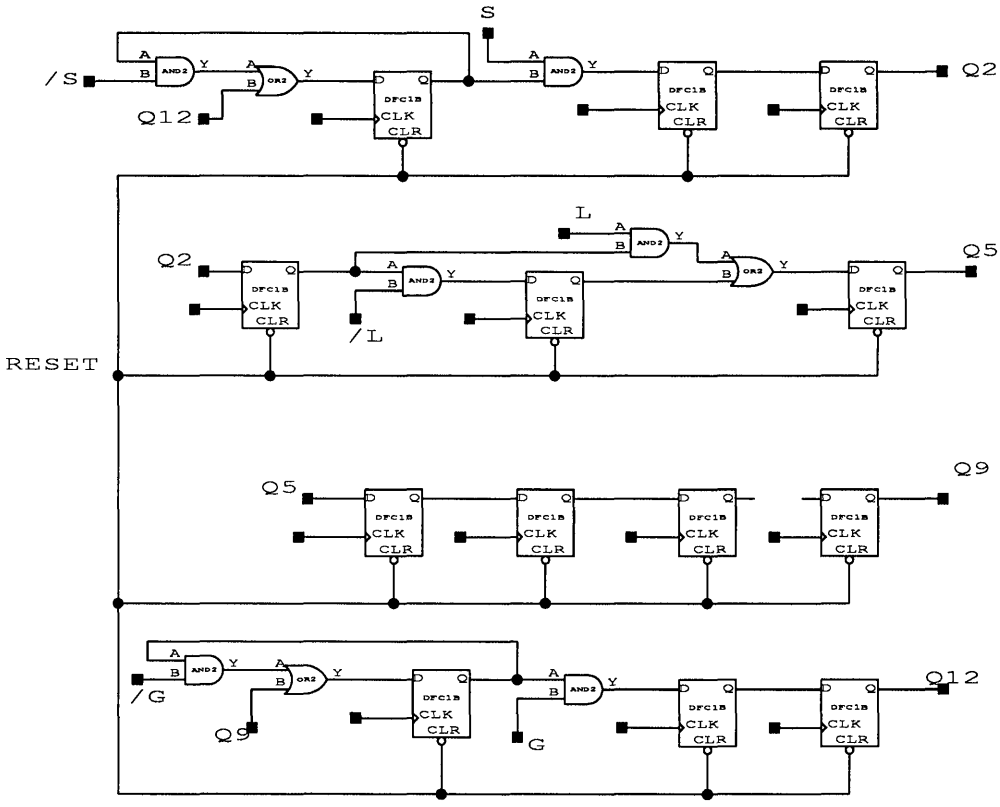


Figure 7. FPGA State Machine

JTAG Implementation in ACT 2 Devices

Application Note

Introduction

JTAG (Joint Test Action Group) is a test approach used to debug various problems caused during manufacturing. It requires special functions built into a device. This paper defines the steps in implementing JTAG in Actel's ACT™ 2 devices and includes the cost in logic resources and performance. This application note describes the basic information needed to implement JTAG; however, the designer should refer to the IEEE Std. 1149.1 to ensure compliance. This note also considers alternate test approaches significantly less costly than JTAG.

JTAG Implementation

JTAG consists of two major functions, the scan cells and the TAP (Test Access Port) controller that controls the shifting and loading of the scan cells. There are three types of scan cells: the basic I/O scan register, the instruction register, and the bypass register. The block diagram for the complete JTAG structure is shown in Figure 1. The TAP controller drives the control signals for the shifting and loading data in the bypass register, instruction registers, and I/O scan registers. The instruction register contains information that defines the current test type for the I/O scan registers, and both the instruction register and TAP controller supply information to the MUX SELECT DECODE to determine the source for TDO. The different types of tests supported by JTAG are shown in Table 1. Implementation for the EXTEST, SAMPLE/PRELOAD, INTEST, BYPASS, and RUNBIST will be considered in the following sections. For a complete description of optional JTAG functions and test types, please refer to the IEEE Std. 1149.1.

Table 1. JTAG Test Types

BYPASS	Mandatory
SAMPLE/PRELOAD	Mandatory
EXTEST	Mandatory
INTEST	Optional
RUNBIST	Optional
IDCODE	Optional
USERCODE	Optional

Implementing the Basic I/O Scan Register

The basic scan structures are shown in Figures 2, 3, 4, and 5. The signal SHIFTDR is used to select between system data and scan data. The signal CLOCKDR is used to capture the data selected by the SHIFTDR signal. The UPDATEDR signal controls the clock to a flip-flop used to isolate the output pin from data being shifted through the device. The configurations shown in Figure 2 will support only the SAMPLE/PRELOAD and the EXTEST routines. To support the INTEST and RUNBIST, the input scan cell must be configured as shown in Figure 6. Also, the input scan portion in the bidirectional scan cell of Figure 5 must be replaced with the structure in Figure 6. Note that when this is done the decoded MODE values will no longer be common.

Implementing Instruction Register Cells

Instruction registers are used to encode the current instruction. These values are decoded and used to create the MODE signal that defines the functions in the JTAG scan macros. The number of instruction registers required is defined by several factors. The first is the number of test types that must be performed and are used to encode the MODE signals. The second is to control the multiplexer select signals used to define the source for the TDO signal shown in Figure 1. The implementation for an instruction register is shown in Figure 7. The value for the MODE signal is shown in Figures 3, 4, 5, and 6 to support the various test types.

The DATA input is used to load optional test information into the instruction scan register. JTAG requires that the DATA input on the least significant bit of the instruction register (that is, closest to TDO) be set to a logical one and that the second least significant bit be set to a logical zero. Optional status data can be input to any additional instruction registers.

The TRST_ and RESET_ lines are used to force the value of the instruction registers to the default test type. These signals can be used to SET or RESET the instruction bits. The RESET only circuit as shown in Figure 7 is merely for convenience. The instruction IDCODE is the default instruction if it is implemented. If the IDCODE test is not implemented, then the BYPASS test is the default test type. The instruction bits to define the BYPASS test should all be logical ones. Therefore, the flip-flops should all be SET for this test case.

Information from the instruction register is passed to the TDO output whenever the TAP controller signal SHIFTIR is active (high).

Bypass Register

The bypass register is a required element for JTAG implementation. This register is used to bypass the normal I/O scan cells to shorten test time. Implementation for the bypass register is shown in Figure 8. The long scan chains are bypassed by storing the current information present on TDI and passing it to TDO selected by the multiplexer values as shown in Figure 1. Once again, the instruction register must be set to all ones to define the BYPASS function.

Implementing the TAP Controller

The TAP controller is the state machine that creates the various signals controlling the activity of the scan registers. The Actel implementation for the TAP controller is shown in Figures 9 and 10. The TAP controller implements the state diagram described in the IEEE Std. 1149.1. The state machine is updated on the rising

edge of TCK and the registered control signals are updated on the falling edge of TCK. The TMS signal is used to define branching in the state machine.

Architectural Considerations

When implementing JTAG in an Actel ACT 2 device, the following should be considered:

- Since the scan chains are basically shift registers, the designer must guarantee that there are no hold time violations caused by clock skew. The easiest way to prevent this is to use a global clock buffer (CLKINT) to drive the CLOCKDR signal. Otherwise, a buffer tree must be constructed and careful timing analysis performed.
- All I/O scan chains should be tied together after pin assignment has been made and preliminary place and routes have been done. Tying together I/Os that are physically close together reduces routing requirements. The same is true for the UPDATEDR, MODE, and SHIFTDR signals. Typically these signals will be buffered before the actual I/O scan cells. Any given buffer should only drive the UPDATEDR and SHIFTDR signals on I/Os close together (in groups of 4 to 8 pins). To determine which I/Os are close together, please refer to bonding diagrams and pin cross-reference charts available from Actel Technical Support.

Actel's tool set contains a function that eliminates all modules that do not drive a sink. These recommended steps will cause this to happen and will skew the total required logic resources. To prevent this from occurring, the PRESERVE option described in the *ALS User's Guide* should be used on all nets without sinks (this can be done in the JTAG soft macros).

Estimating Cost of JTAG Implementation

While there are benefits to using JTAG to implement system-level tests, there are also significant costs in both logic resources and performance. The following sections define the cost.

Impact on Density of Device

Internal resources are required to implement JTAG. Some of the required resources, like the TAP controller and the bypass register, are fixed. The other elements required for implementing JTAG are on a per bit basis. Table 2 describes the resources required for both sequential and combinatorial modules to implement JTAG. Since significant loading can occur on the I/O scan chains that will require buffering, Table 3 shows the number of loads on a per bit basis for the TAP controller signals.

An example is given in Appendix 1 with an overhead of about 33 percent of total device resources for a heavily utilized 1280. The actual amount of resources required will vary depending on the number and type of I/O in a given application.

Table 2. Logic Module Resources Required to Implement JTAG

	seq	comb	total
TAP Controller	8	19	27
Bypass Register	1	0	1
Instructions Register	2	1	3 per bit
Inputs	1	0	1 per input**
Outputs	2	1	3 per output
Tristates	4	2	6 per tristate
Bidirectional	5	2	7 per bidirectional**

** Supports EXTEST, BYPASS, and SAMPLE/PRELOAD only.

Table 3. Input Loading on JTAG Scan Macros

Input	SHIFT DR	CLOCK DR	UPDATE DR	MODE
Bypass Register	1	1	0	0
Instructions Register	0	0	0	0
Inputs	1	1	0	0
Outputs	1	1	1	1
Tristates	2	2	2	2
Bidirectional	3	3	2	

Impact on Performance

JTAG implementations can also affect performance. As shown in Figure 2, the input scan cell puts an additional load on the normal system circuitry. The additional load will typically add some delay (less than one nanosecond) to the system signal. Outputs are much more heavily impacted because of the need to place a 2 to 1 multiplexer between system output data and the output pin shown in Figures 3, 4, and 5. The multiplexer can add delay of 5 to 10 ns depending on routing and speed grades. The additional multiplexer of Figure 6 will also affect performance on inputs similar to its effect on outputs.

Impact on I/O and Clock Resources

JTAG implementation also requires I/O dedication. The TAP controller requires a minimum of five I/O resources. The I/Os are as follows:

- TCK – Test Clock Input
- TMS – Test Mode Select
- TDI – Test Data Input
- TDO – Test Data Output
- TRST – Optional Test Reset Input

A global clock resource is also used for the CLOCKDR signal, leaving only one for system level logic.

Alternatives to JTAG

The purpose of JTAG is to ease the testing and debugging of manufacturing problems at the board level. Different test approaches exist that use significantly fewer logic and I/O resources than JTAG and could be used when either is a problem. The following sections describe various test structures that aid board-level tests that have significantly lower logic requirements.

Using Actel Built-in Test Structures

One of the functions built into Actel devices is a feature called the Actionprobe® diagnostics. This function provides 100 percent observability of internal nets by serially addressing various points in the array. Internal nets are addressed by driving address information and a qualifying clock on the SDI and DCLK pins. Internal activity can then be observed on the PRA and PRB pins. This function can be used to verify device connection to a board by first addressing an internal net associated with the I/O (for example, NETA in Figure 11), then stimulating an external pad, and finally observing the effect on an external probe pin. Equivalently, NETD could be selected, and then activity on inputs IN1, IN2, and IN3 could be observed. This test method requires no dedicated logic to implement. Control of the Actionprobes is described in the *ALS User's Guide*.

Use of the Actionprobes can be extended to all pins, including outputs, by changing the selection of the output cells. If bidirectional outputs are chosen rather than standard outputs, a new path is created back into the array that can be observed with the Actionprobes. Since the outputs will continue to drive during the test (unless disabled), the tester must temporarily back-drive the pin to a specified state for observation on the probe pins. Back-driving can be avoided by adding tristate circuitry to the outputs under test and disabling the outputs for the test. Since the input portion is not tied to any internal resources, the PRESERVE option (described in the *ALS User's Manual*) must be used to prevent the software from eliminating the nonfunctional input net. Once again, no additional logic is required to implement this test method.

In addition to the Actionprobe diagnostics, a special function to tristate all I/Os is also built into all Act 2 devices. For a complete description of this test feature, please contact Actel Technical Support.

Non-JTAG Test Structures

The following sections describe test structures that can be designed into the part to aid in board-level tests. The test structures described use significantly fewer logic resources than JTAG and also can reduce the impact on performance.

I/O Mapping

This built-in test method simply maps inputs directly to outputs with a multiplexer at the output selecting between normal system data or input test data. An example is shown in Figure 12. The test is done by first asserting the TEST pin. The inputs can then be toggled and their function observed on the outputs. If the number of inputs exceeds the number of outputs, then simple combinatorial functions can be added to map many inputs to a single output. The OR gates in Figure 12 are examples of this. If the number of outputs exceeds the number of inputs, then a single input can be tied to multiple outputs. Bidirectional pins can be mapped as inputs or outputs, but not both.

I/O mapping also requires architectural considerations in mapping inputs to outputs and should only be done once pin assignment is complete. Inputs should be mapped to outputs that are close together to minimize routing and performance problems. The impact of this test method is basically one pin for test and one multiplexer for each output in addition to the delay associated with the multiplexer.

Input Only Test Structure

Typically, a functional test to toggle outputs can be generated with a subset of functional patterns. The difficulty is creating test vectors that can trace faults to specific inputs. The input only test method adds logic to inputs apart from normal chip function. This test method requires all of the inputs and bidirectional pins to be tied together through an OR/AND structure as shown in Figure 13. The test is performed by asserting the test pin and driving all inputs low. Each input is then toggled and observed on the output pin labeled O7 in Figure 13. When bidirectional pins are used as inputs, the test must guarantee that the bidirectional pins are in the tristate condition during the test. If this cannot be guaranteed, additional test logic must be built in as shown in Figure 14 for the bidirectional pins. The cost of this test method is

summation k

k = 1,2,3

$$\frac{n+m}{4^k}$$

n = inputs

m = bidirectional pins

This test can be further extended by making every output bidirectional and using the TEST pin to disable the driver during the test as shown in Figure 14. No functional test patterns are then required. To estimate costs, the total I/O should be considered in the summation. This test method would require about 4 percent of a 1280 with all pins utilized.

When implementing this test method, pins should be tied into the OR/AND test array only after pin assignment. All pins that drive a unique OR/AND module should be closely associated on the chip.

Conclusion

JTAG is an effective test method to aid in verifying and debugging board-level problems. When implementing JTAG in Actel ACT 2 parts, the resources required as well as the impact on performance should be considered. Architectural considerations should be made to minimize the impact on performance and routing. Where the impact to logical resources or pins is unacceptable, the designer should consider optional test methods that can also help in the testing and debugging process.

Appendix 1
Example of JTAG Resource Requirements

1) Device and Test Description

Device	1280
Tests	EXTEST, BYPASS, SAMPLE/PRELOAD
Inputs	45
Outputs	35
Tristate	10
Bidirectional	32

Use a CLKBUF to drive the ClockDR signal.

2) Loads on additional signals

	LOADS	NUMBER OF BUFFERS REQUIRED
SHIFTDR	197	22
UPDATEDR	119	12
MODE	119	12

3) Total logic module resources required

	seq	comb		seq	comb
TAP Controller	8	19	x 1	8	19
Bypass Register	1	0	x 1	1	0
Instructions Register	2	1	x 2	4	2
Inputs	1	0	x 45	45	0
Outputs	2	1	x 35	70	35
Tristates	4	2	x 10	40	20
Bidirectional	5	2	x 32	160	64
Miscellaneous Buffers	0	46			
Total				228	186

Cost of example is 414/1232 logic modules = 33 percent of available logic resources.

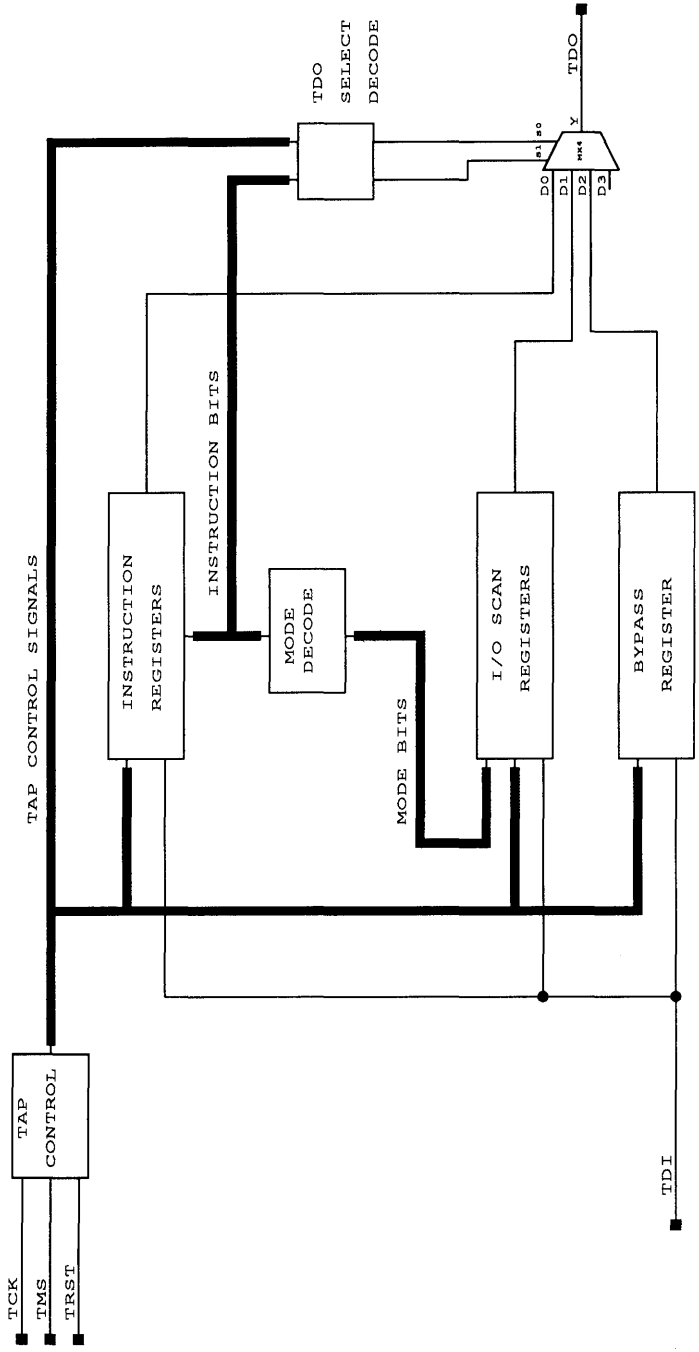


Figure 1. JTAG Block Diagram

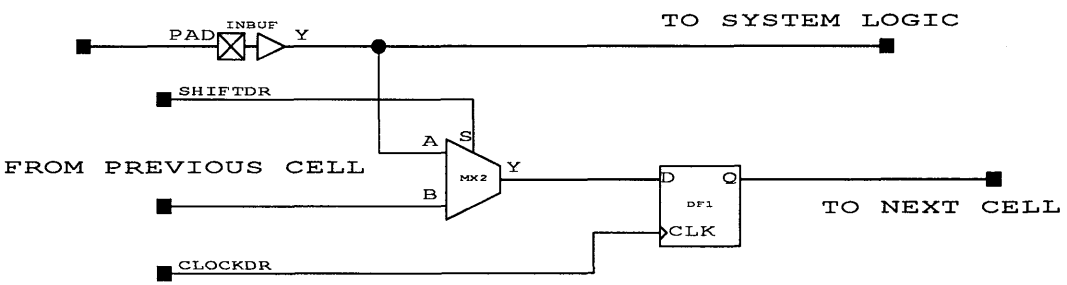


Figure 2. Basic Boundary Scan Cell for an Input Pin

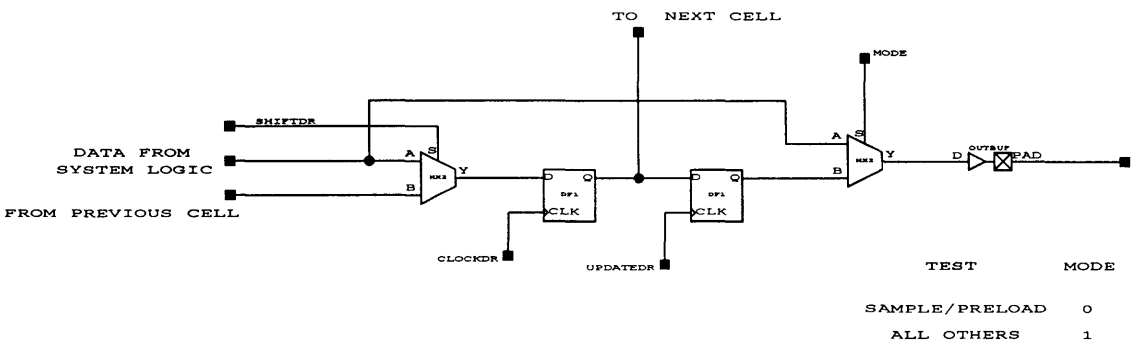


Figure 3. Basic Boundary Scan Cell for an Output Pin

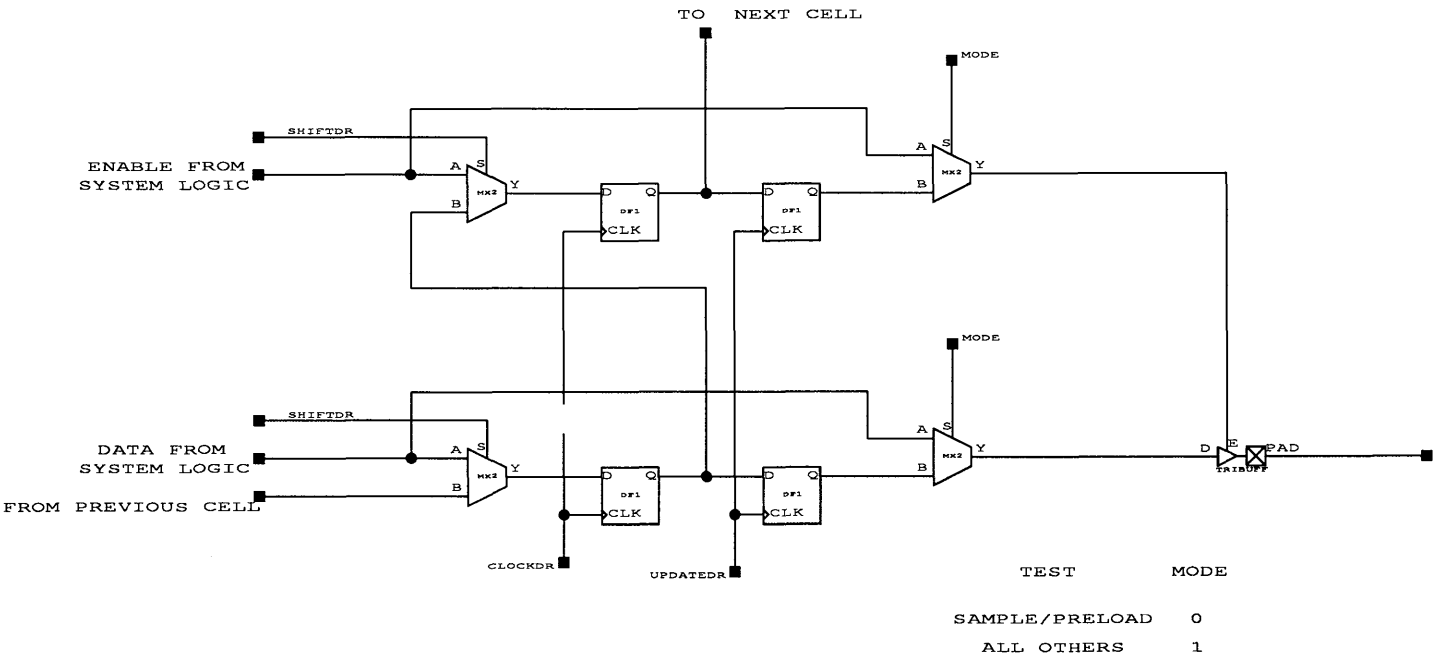


Figure 4. Boundary Scan Cell at a Three-State Output

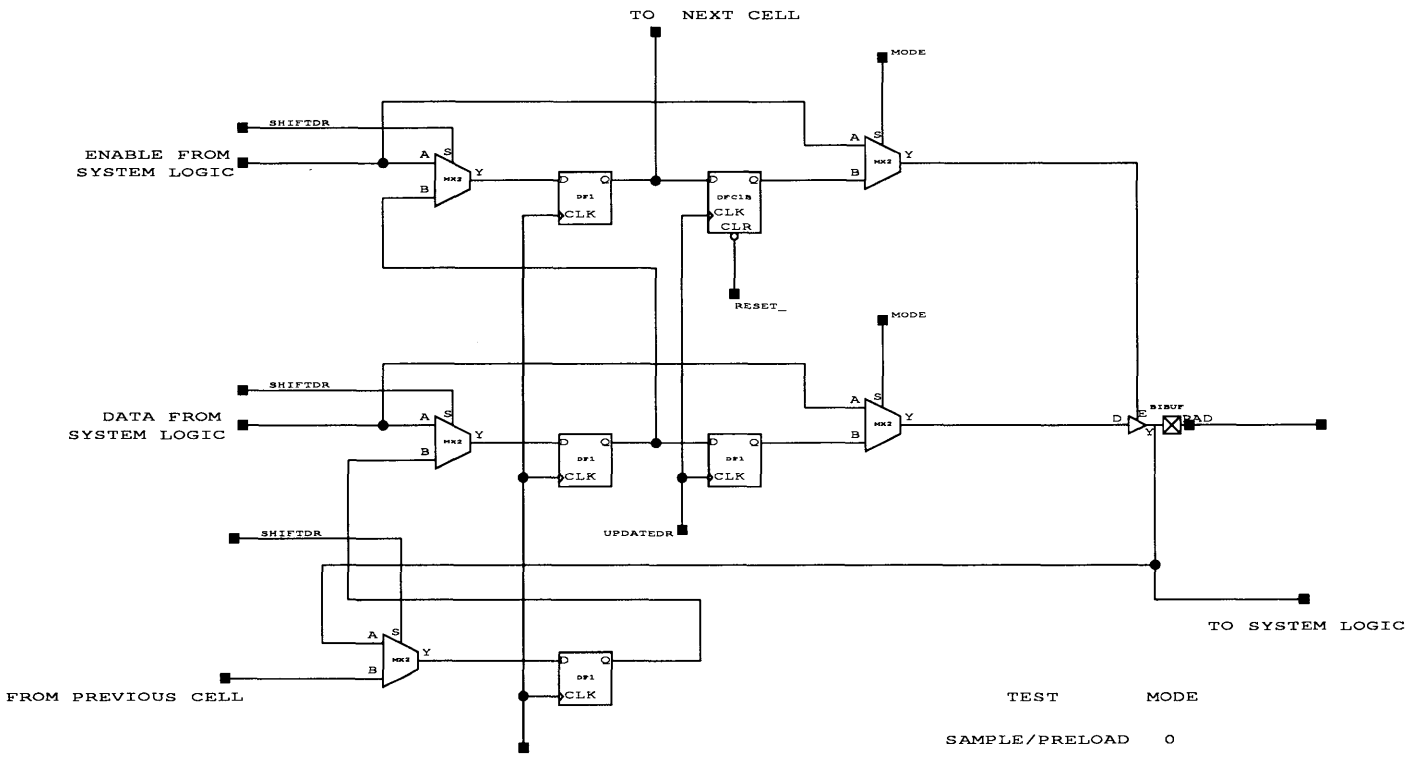
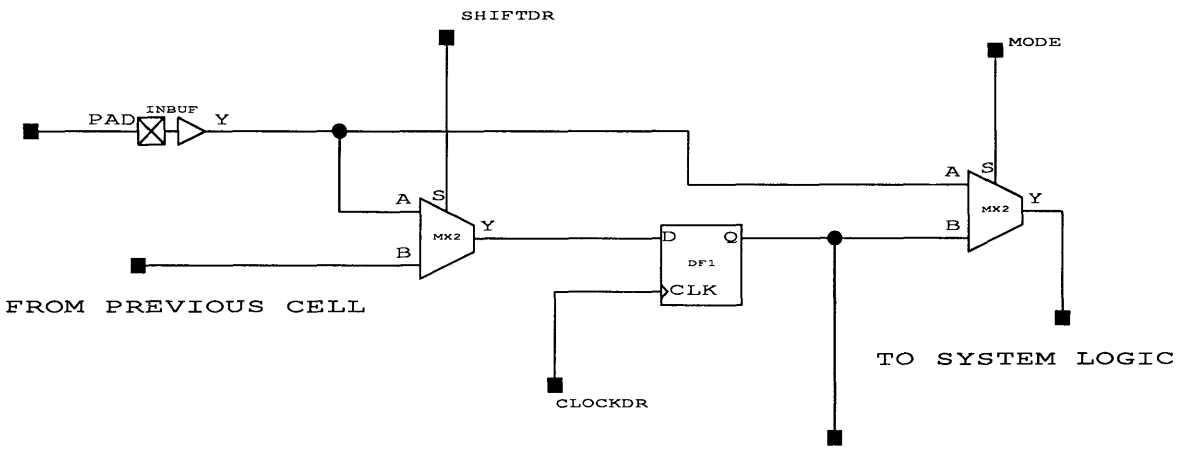


Figure 5. Boundary Scan Cell at Bidirectional Pin



TEST	MODE
EXTEST	0
SAMPLE/PRELOAD	0
INTEST	1
RUNBIST	1

Figure 6. Modified Input Cell That Supports All Test Types

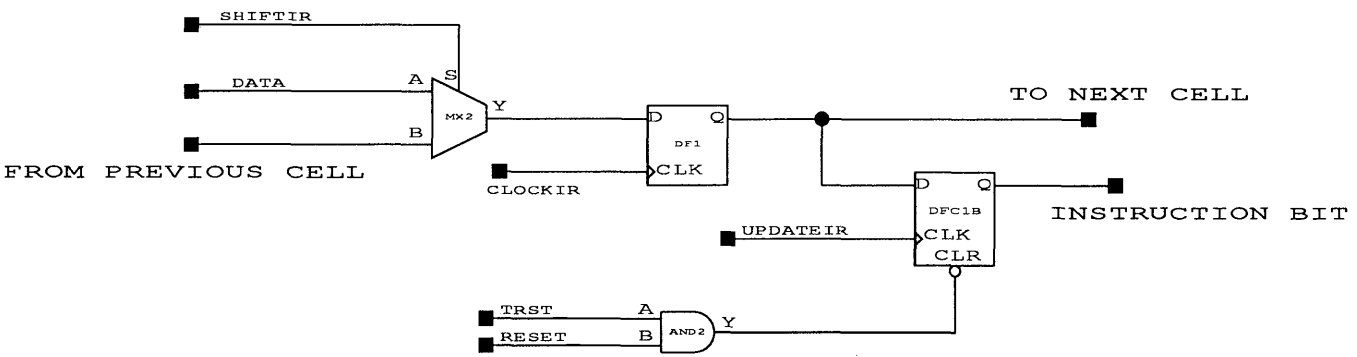


Figure 7. Instruction Register Implementation

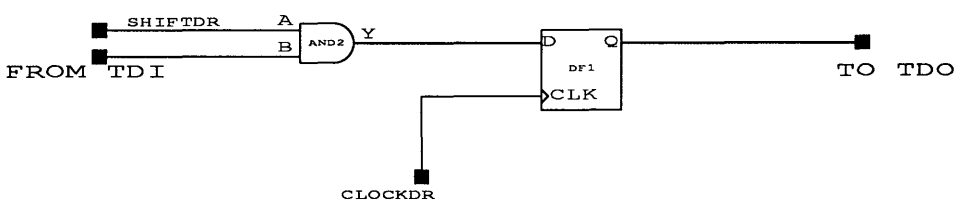


Figure 8. Bypass Register Implementation

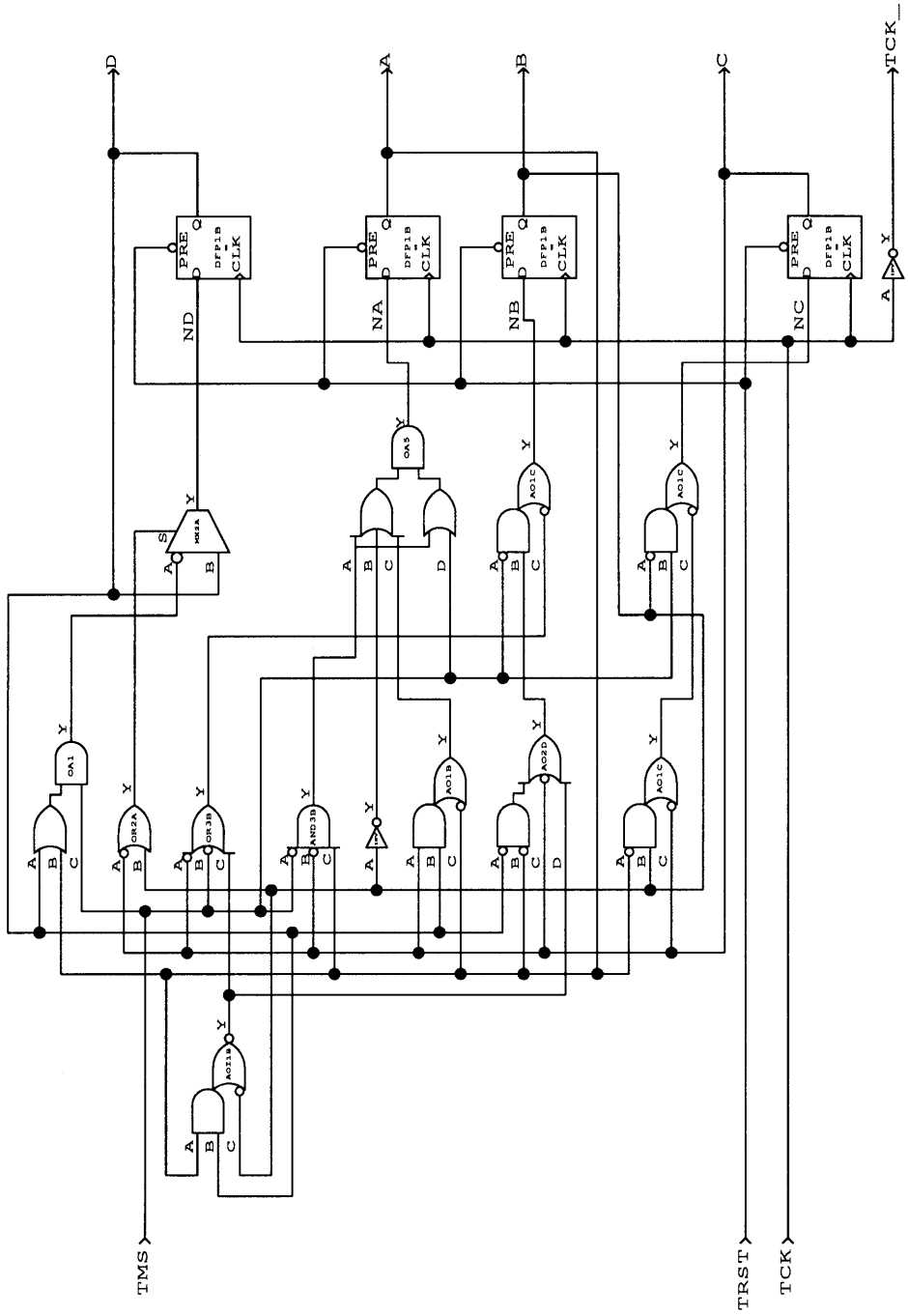


Figure 9. TAP Controller State Machine

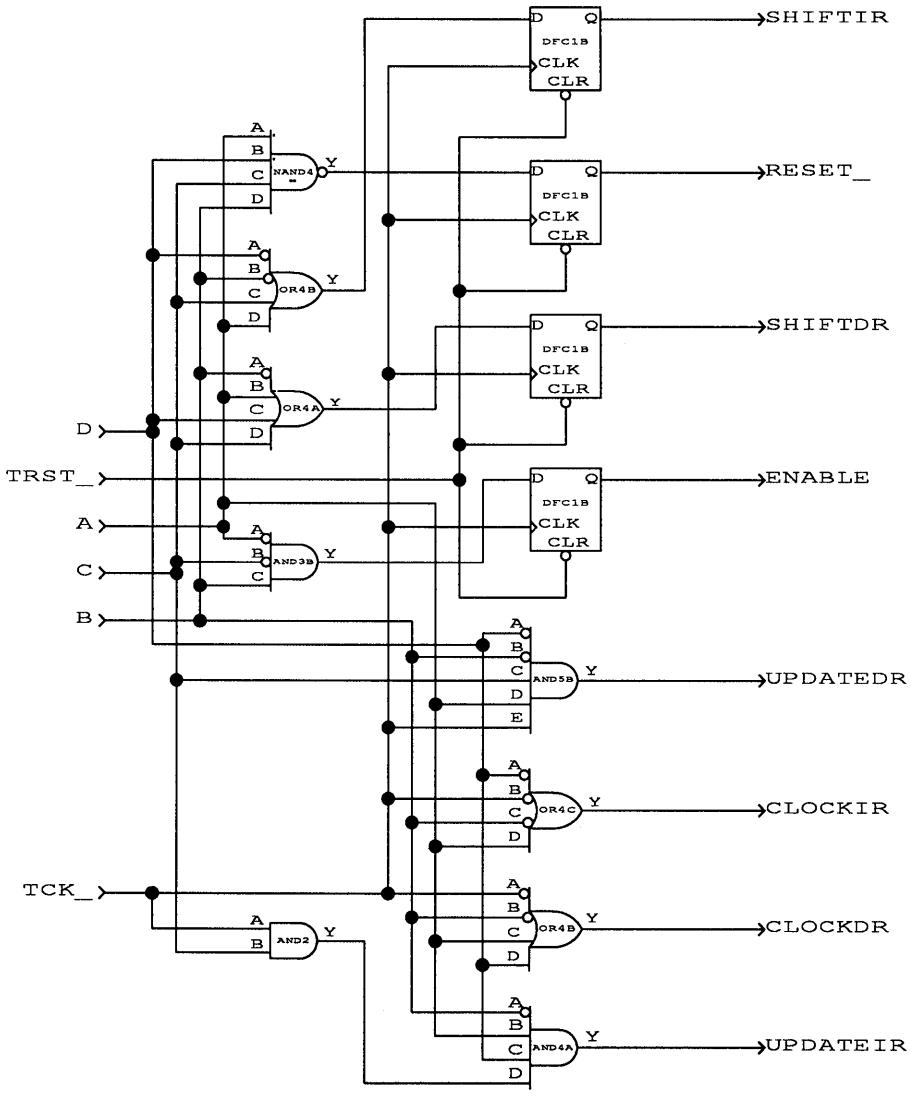


Figure 10. Implementation of TAP Control Signals

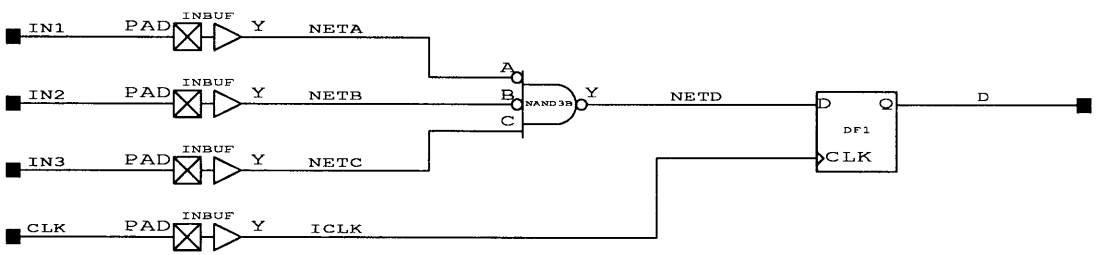


Figure 11. Example Circuit for Use of Actionprobes

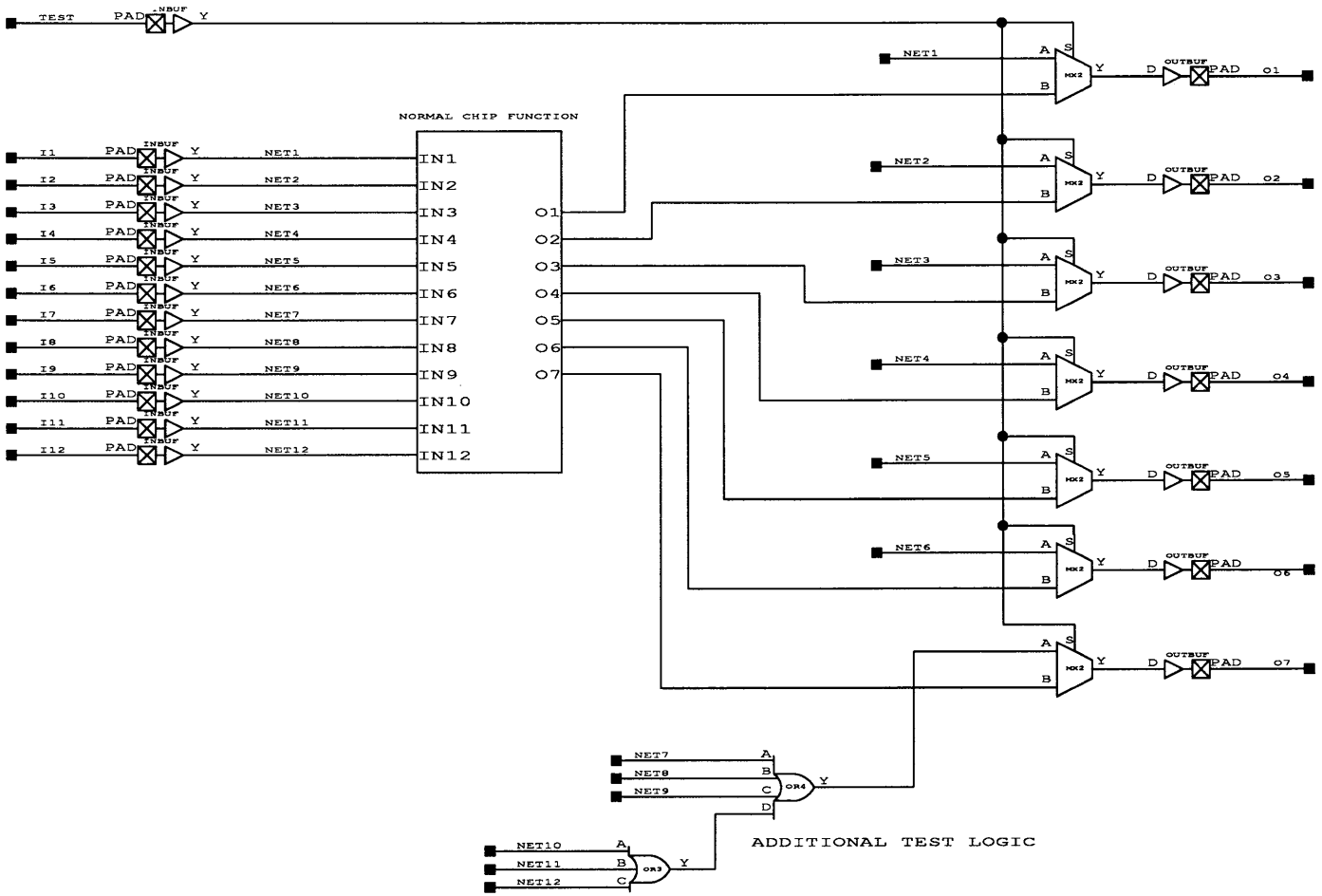


Figure 12. Example of the I/O Mapping Test Method

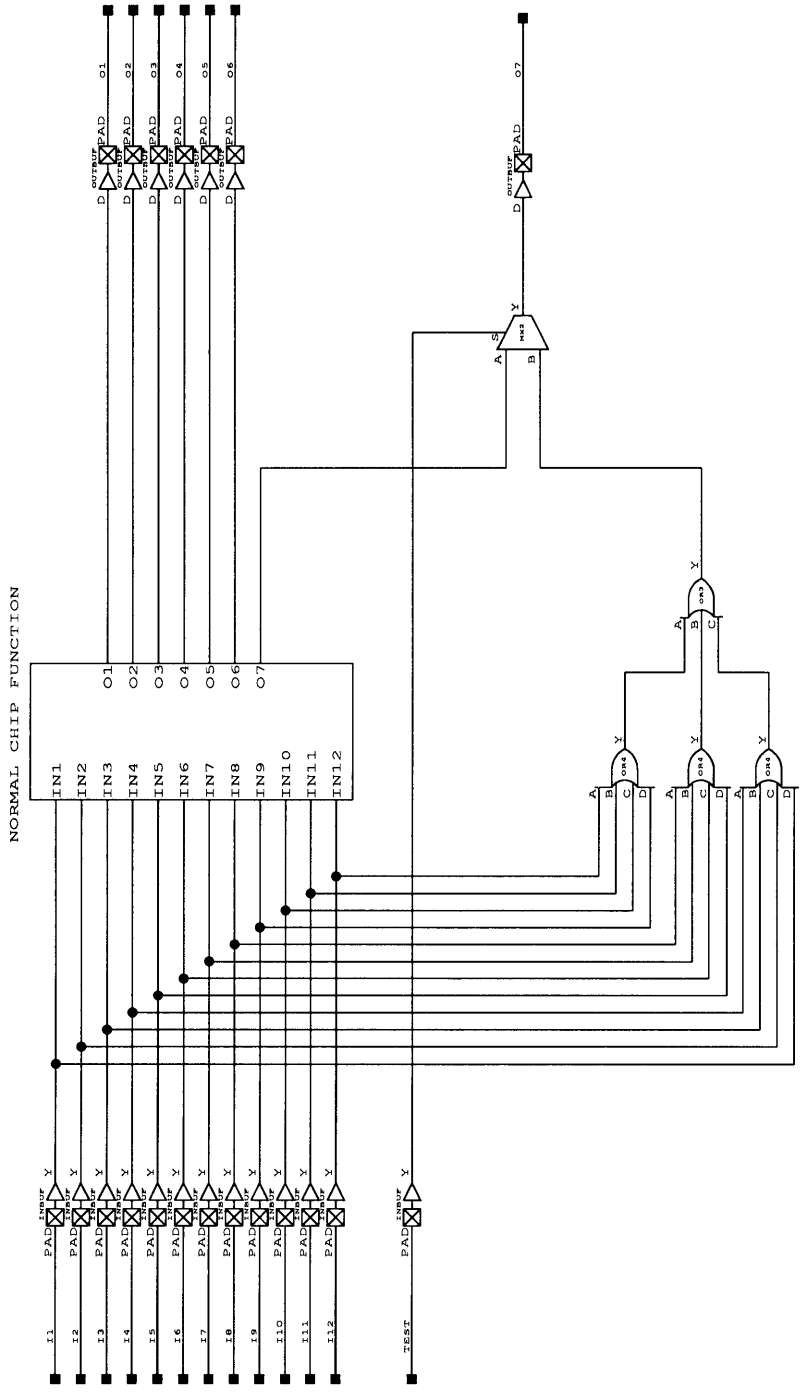


Figure 13. Input-Only Test Implementation

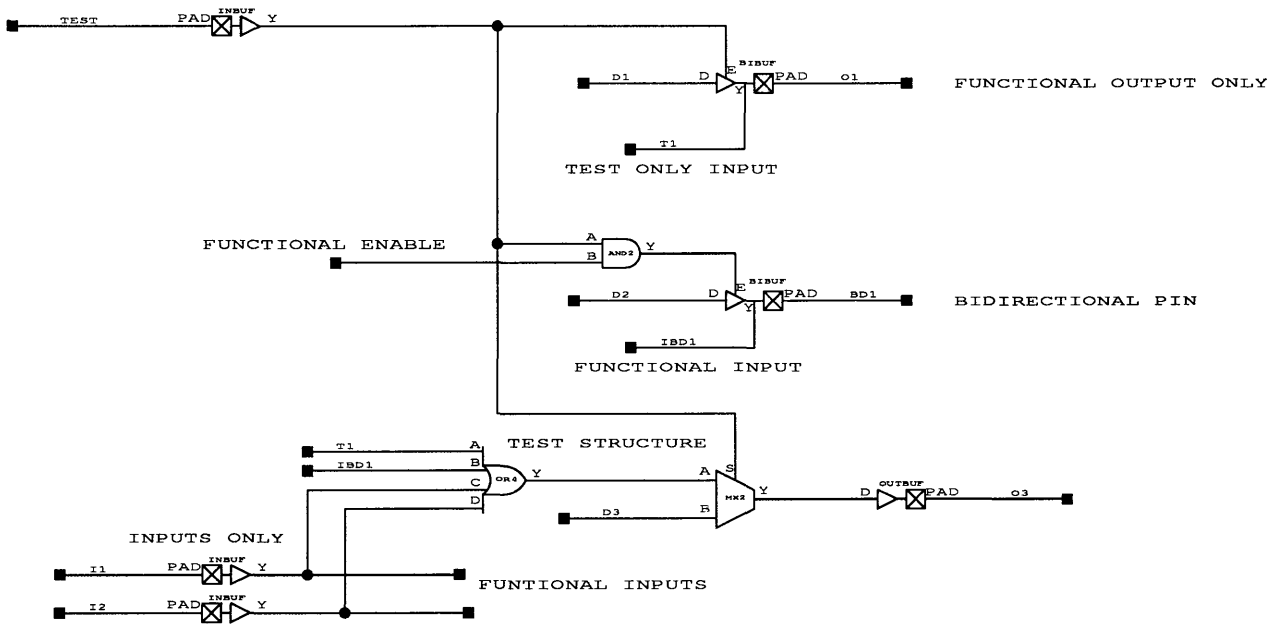


Figure 14. Handling of Outputs and Bidirectional Pins for Input-Only Test

Designing Adders and Accumulators with the ACT 2 Architecture

Introduction

Many designers implement adders using carry-propagation techniques. The multiplexer-based ACT™ 2 combinatorial module (C-module) allows you to use the more efficient carry-select design. This method partitions the add functions into blocks that perform two additions simultaneously on a number of bits of the two operands.

The two additions are the same except that one assumes a carry-in and one assumes no carry-in. The two sums are input to 2 to 1 multiplexers, one for each bit pair. The carry line from the low bits to the high bits is used to select the appropriate sum for each block.

The ACT 2 architecture lends itself well to implementing adders of various sizes using the carry-select technique. A sample design for a 16-bit adder, as shown in Figure 1, will be used to illustrate adder design.

Balancing Sum and Carry Levels

To obtain optimal performance from a carry-select adder, design it so that the number of levels of logic required for the carry chain equals as closely as possible the number for the largest sum block. When they have the same number of levels, the sum bits arrive simultaneously at the data pins and the select pins of the output multiplexing stage.

To balance the levels of logic modules for the sum blocks with the carry, is to partition the sum blocks by considering the logic levels required for the sums and the levels for the carry between sums. The size of the partitions varies with width of the data. The ACT 2 library contains some powerful hard macros used to shorten the levels of logic required for generating sums and carries. The description of the sample design will illustrate the use of the macros.

Sample Design

For the 16-bit adder, the optimal organization is to perform two 2-bit additions on the four least significant bits with the remaining higher order bits broken into four sections of three bits each.

In the top-level schematic the addition logic of the two least significant bits is visible. The other additions are performed in lower levels of the design hierarchy described in the next section.

Carry Logic

The ACT 2 library includes two 2-level carry hard macros. One macro generates a carry for the two-bit pairs assuming the carry-in is true; the other assumes it is false. The latter macro may be seen at the bottom of Figure 1 making the carry for the two least significant bits.

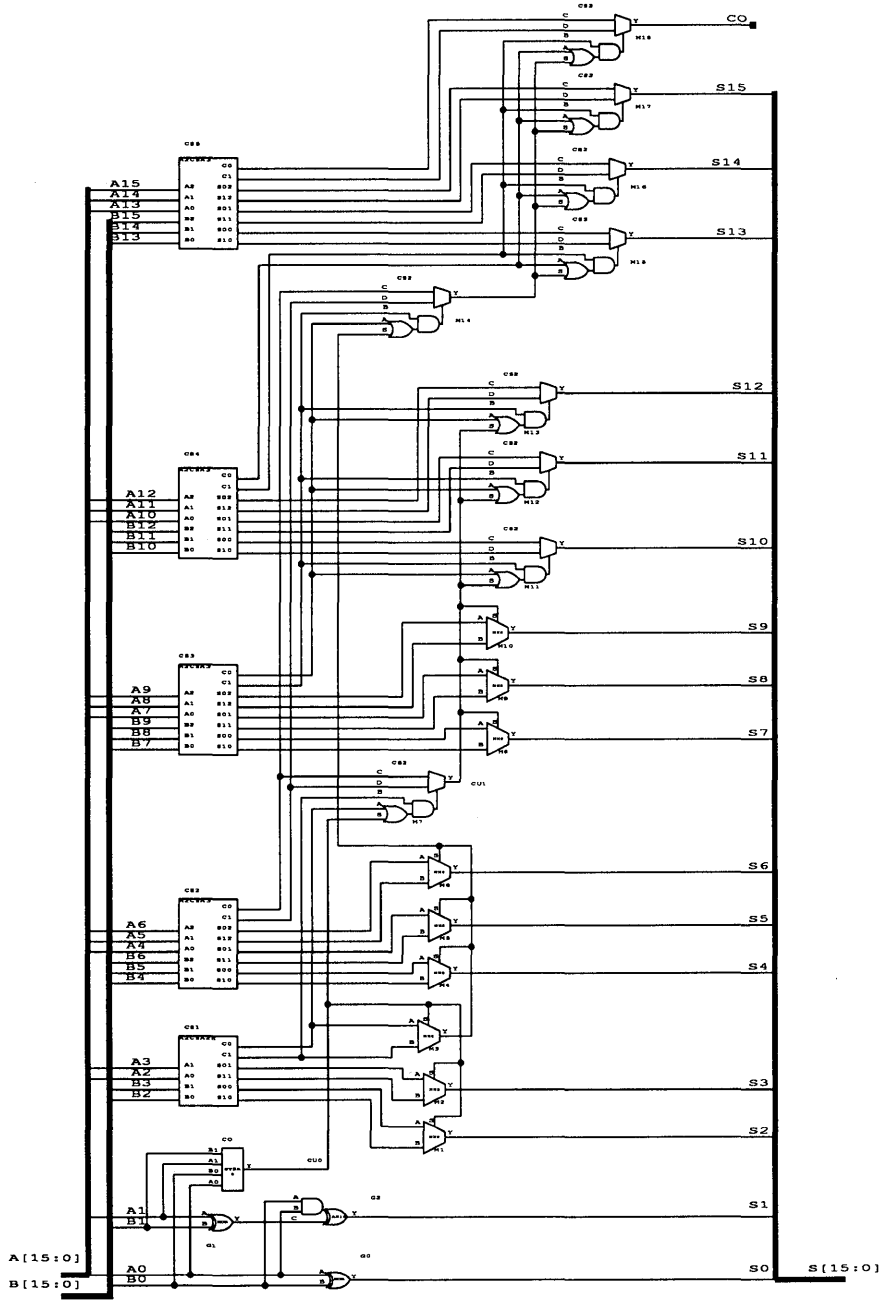


Figure 1. 16-Bit Adder Schematic

The carry macro output drives the select line for the 2 to 1 multiplexers for sum bits two and three. It also drives the select line on the cascade multiplexer. The cascade multiplexer is a special ACT 2 hard macro that can propagate two levels of carry. The macro is depicted in Figure 2 and has five inputs. The top multiplexer inputs select the most significant sum or carry. The three lower inputs drive logic that implements a simplified form of a 2 to 1 multiplexer.

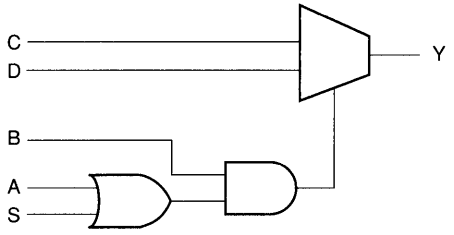


Figure 2. Cascade Multiplexer Macro

A fully implemented 2 to 1 cascade multiplexer does not map into the ACT 2 module efficiently, but the full functionality is not required in a carry-select adder. There is a simplified version of the cascade multiplexer that maps into a C-module or combines with a flip-flop in a sequential module (S-module).

The simple version has logic driving the select for the upper level multiplexer consisting of only a two-input OR driving one input of a two-input AND. The two OR gate inputs are driven by the carry output from the next lower sum block assuming no carry-in and by the carry-in from the rest of the lower bits of the adder. The remaining AND input is the carry from the sum block, which assumes a carry-in.

The logic is correct for a carry-select adder because, if the assume-no-carry-in input is true (meaning that a carry was generated within that sum block), then the assume-carry-in is always true (since it equals the false plus one). This completes the AND function.

If the carry from the lower bits is true (meaning a carry is propagated to the sum block), then we complete the AND if the assume-carry is true.

Three-Bit Carry Select Adder

The schematic for the three-bit adder block appears in Figure 3. The adder requires thirteen logic modules to generate the three sum and carry pairs. All the output paths are two levels of logic or less.

The two carries for the three bits come from two-level carry hard macros driving a three-bit majority macro.

All the sums are generated from exclusive OR or NOR gates. The Actel library contains several two-module adder macros with both a sum and a carry output. These macros may be used as part of a sum depending on how well they fit into the overall adder soft macro structure.

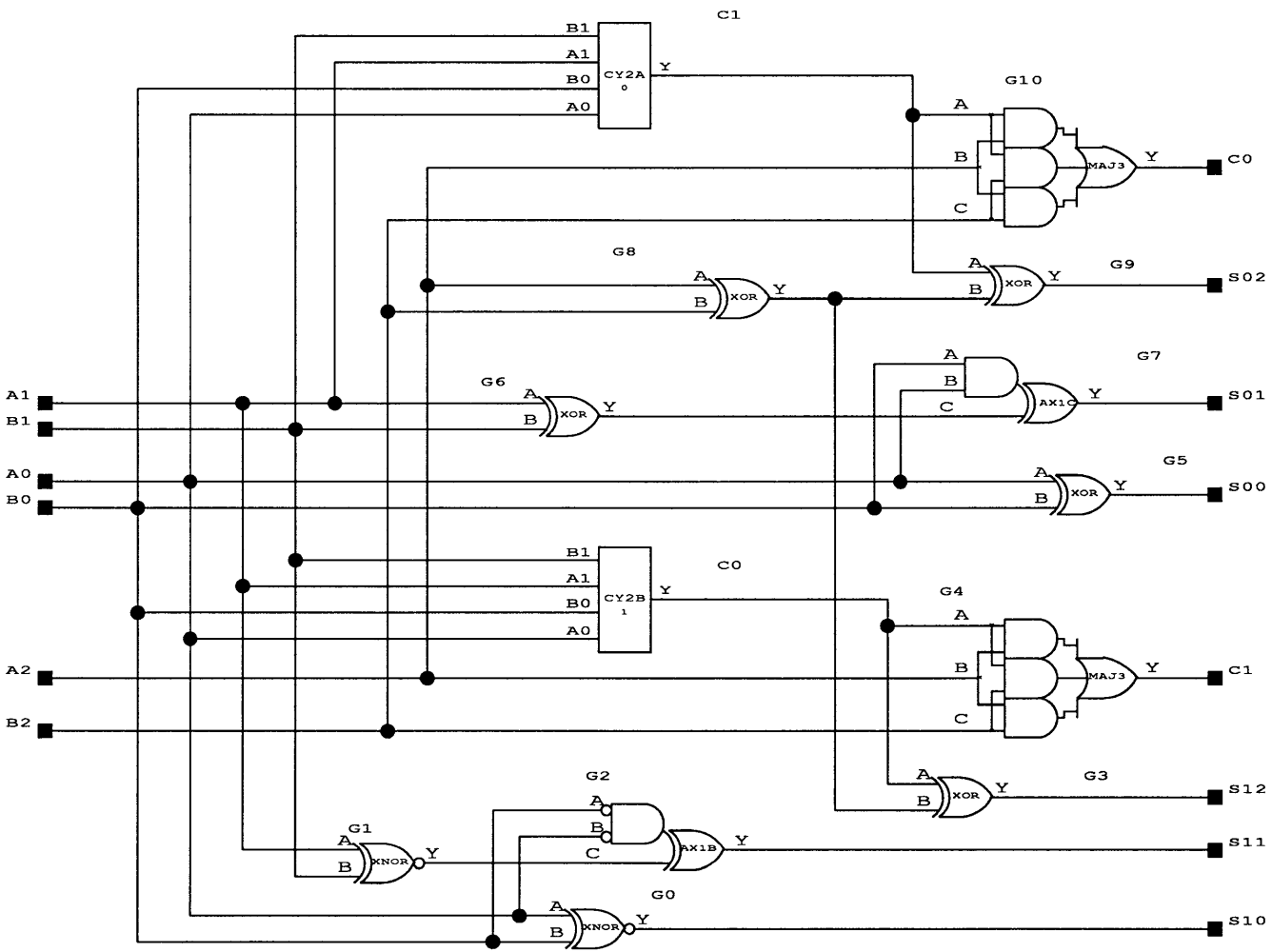


Figure 3. Three-Bit Adder Block

Making an Adder into an Accumulator

All the sum and carry outputs of the adder macro can be combined into a single ACT 2 S-module. This feature means that if the data inputs of a 16-bit register are schematically connected to an adder's outputs, the ALS software will automatically put the adder output macros (2 to 1 multiplexers or cascade multiplexers) into their respective flip-flop in the register.

The registered-output adder will suffer no degradation in performance from combining because the delay through the combinatorial part of the S-module is less than that of an uncombined macro. Tying the register output back into the inputs will make the circuit into an accumulator. A sample design for an accumulator made from an adder and a register may be seen in Figure 4.

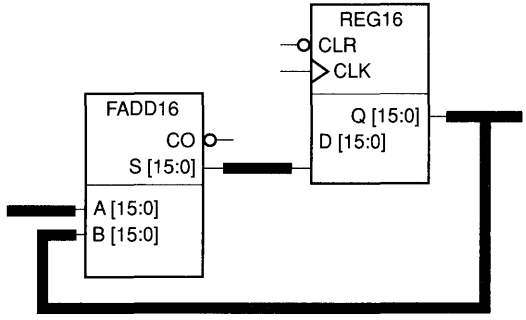


Figure 4. 16-Bit Accumulator

Sample Design Results

The sample design uses 82 ACT 2 modules. The slowest path in a function is usually the one with the most levels of logic. In this case, it is the carry chain, which has four levels of logic. As mentioned previously, all other paths have fanouts of three or less.

The modules in the chain have fanouts of three, four, seven, and four. Criticality may be used to optimize the path performance. Criticality works best when fanout is low. When the fanout of a speed-sensitive net exceeds seven, performance can usually be improved most by adding redundant logic. For fanouts of less than seven, adding a redundant module may bring no improvement. Using redundant logic for fanouts of seven should be considered on an individual basis. Adding a redundant module to the carry path would change its fanouts to three, five, three, and four. The expense of one module may be justified by the performance improvement from lowering the fanout.

It is also possible to improve performance by pipelining an adder. Since all of the combinatorial functions used in the adder can be combined (if the function's output drives a flip-flop, ALS will put both in a single S-module), designers may pipe the adder at the points that provide the best performance at no cost in additional modules.

Other Adder Macros

The carry select architecture can be extended to adders of any size. Adders of 8 to 15 bits may be designed using the technique in three levels of logic. Adders from 16 to 24 bits can be done in four levels.

When adapting the adder design to other operand sizes, remember to repartition the sum block sizes to match the logic levels of sums and carries.

Fast Adder Design Techniques

Application Note

Introduction

The VAD series of very fast adders in the Actel soft macro library uses several techniques to create optimally designed macros for the Actel FPGA architecture. The VADC32C very fast 32-bit adder macro uses variations of the VADC16C very fast 16-bit adder macro as a building block. This configuration yields very high performance for 32-bit system designs. What follows illustrates these adder design techniques as applied to the VADC32C macro. Refer to the application note regarding general adder and accumulator design, "Designing Adders and Accumulators with the ACT™ 2 Architecture", for more information. These techniques apply to ACT 2 and ACT 3 designs.

The carry propagation technique (ripple carry addition) for use in serial addition networks is defined as,

$$S_i = A_i \oplus B_i \oplus C_i$$

where S_i is the sum bit, A_i and B_i are the ith bits of each operand, and C_i is the carry to the ith stage; the carry to the next stage is defined as,

$$C_{i+1} = A_iB_i + C_i(A_i + B_i)$$

Therefore, adding two n-bit operands takes at most n-1 carry delays and one sum delay. This delay becomes large for adders with n > 4 and significantly reduces operating speeds.

A much faster algorithm for adding n-bit operands uses the combinational sum technique. This is a multiplexer-based design that partitions the add functions into blocks that perform two additions simultaneously. The two additions are the same except that one assumes a carry in while the other assumes no carry in. The two sums are input to 2:1 multiplexers and the carry in of this addition controls the output of the multiplexer. The adders provide high performance, requiring only four logic module delays from input to output.

Design

The VADC16C macro uses the carry look ahead technique to generate the carry out of the 16-bit addition while performing the summation of the operand twice. One addition assumes a carry in and the other assumes no carry in. Both additions are performed simultaneously. Both sums are input to a multiplexer, and the true carry in controls the output of the multiplexer—the sum of the operands. This initial addition is performed with a total of three logic module delays.

Cascading two 16-bit very fast adders is the simplest way to design a 32-bit adder. But this technique involves three additional logic delays, which is unacceptable for a very fast adder. A novel implementation of the VADC16C soft macro extends the width of the inputs to 32 bits with only one additional logic delay.

The design is split into two blocks, as shown in Figure 1. The first block is macro VADC16CN, which performs addition on the first 16 bits and generates a carry out. Actually, two carry outs are generated. One assumes a carry in, and the other assumes no carry in; the true carry in controls the multiplexer so that the true carry out is at the output of the carry out multiplexer. Similarly, two sums are generated for the 16-bit addition. One assumes a carry in, and the other assumes no carry in. The control input of the multiplexer is connected to the true carry in, thereby steering the correct sum to the output of the sum multiplexer. This macro is the same as soft macro VADC16C, except that it has several carry outs to allow for fanouts. These additional logic modules reduce possible loading effects to maintain high operating speeds.

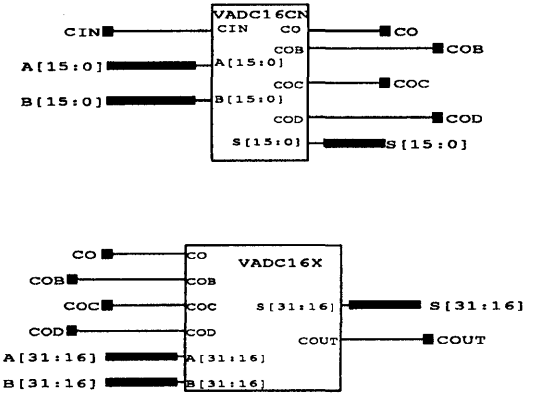


Figure 1. Block Diagram of Soft Macro VADC32C (Very Fast 32-bit Adder with Carryin)

The second block is macro VADC16X, is further split into five lower levels of logic blocks: VADC16XC, VADC16XL, VADC16XM, VADC16XU, and VADC16MX. VADC16XC generates the carry look ahead for the upper 16 bits of the 32-bit adder. Two carry outs are generated. One assumes that there is a carry in, and the other assumes no carry in from the addition of the first 16 bits. This result is input to a 2:1 multiplexer with the actual carry from the lower 16 bits controlling the output of this multiplexer—the carry out of the 32-bit addition. VADC16XL performs two additions for bits 16 to 24. One assumes a carry in and the other assumes no carry in. Similarly, VADC16XM performs two additions for bits 25 to 28, and VADC16XU performs two additions for bits 29 to 31. The sums generated by these blocks are input to the 2:1 multiplexer block VADC16MX. The carry out of the 16th bit is applied to the control of this

9

multiplexer, yielding the true sum at the output of the multiplexer block. This technique has a total of four logic delays, which is eight times less than the traditional ripple carry addition technique, which results in 32 logic module delays.

This adder design results in a 32-bit addition requiring a total of only four logic module delays and approximately 250 logic modules. It is useful for applications that require very fast additions in Actel FPGA devices.

Designing Counters with the ACT 2 Architecture

Perhaps the most common digital logic function is the synchronous binary counter. Regardless of the technology employed to implement counters, they are found in every type of application. Designers familiar with counter design using discrete logic can predict the performance of the counter by reading a datasheet. When using field programmable gate arrays (FPGAs), there is more to consider. The following describes some of the considerations for designing or modifying a counter in the ACT™ 2 soft macro library.

Performance Factors

The advantages of using FPGAs instead of discrete devices are well known. To maximize the benefits of high integration and low power, it is important for the designer to understand how to best implement the counter. The performance of the counter is variable, so it is important to use the optimal design. Four criteria influence performance of logic in FPGA designs, in descending order of importance:

- *Levels of logic.* The fewer the number of combinatorial logic levels between flip-flops, the faster the counter frequency.
- *Fanout.* Propagation delays in FPGAs are sensitive to fanout. Limiting fanout on individual nets improves performance.
- *Fan-in.* Fan-in measures the number of nets connected to a logic module's inputs. Library functions with heavy fan-in efficiently utilize the logic of the module; they aren't a problem when used sparingly. Too many high fan-in macros, though, can congest routing and reduce performance.
- *Number of modules.* Fewer logic modules allow them to be placed closer to each other. Shorter distances between modules speed the connection paths.

While each of these considerations is important in itself, it must be remembered that they are interrelated; an improvement in one may cause a degradation in another, for example, limiting the number of logic levels tends to increase fan-in. A balance must be found among them so that none becomes a drag on performance.

Before proceeding to a detailed description of a sample design, it would be useful to review some fundamentals of the ACT 2 architecture. In the design of a counter or any soft macro, device architecture should always be considered to obtain the best results.

Two Types of Modules

The ACT 2 architecture features two types of modules. Combinatorial modules (C-modules) are used to implement any combinatorial function in the ACT 2 library. Sequential modules (S-modules) can be used for either sequential functions (for example, flip-flops) or combinatorial functions or both. When the S-module is used to implement both a sequential and a combinatorial function (for example, a gate followed by a flip-flop), it is being used in the most efficient way.

The module types exist in roughly equal numbers on ACT 2 devices. The Place and Route software will automatically select the appropriate module for each library component in the schematic. It is up to the designer to understand how to select components from the library to take best advantage of the logic in the modules.

As the sample design will show, you can construct a 10-bit counter with only one level of combinatorial logic between flip-flops, and a 16-bit counter with only two combinatorial levels. If some counter outputs may be active-low or if additional modules are used for redundant logic (for example, some bits have both an active-high and an active-low output), then larger counters may be designed without additional logic levels using five-input gates. Such decisions should consider the implications for module count and fanout as detailed below.

Sample Design

The sample design for a 16-bit synchronous loadable binary counter with a count enable will illustrate some of the considerations for using counters in ACT 2 FPGA designs. The functional description for the counter appears in Table 1.

Table 1. Counter Function

RST	LD	CE	CLK	Q
0	X	X	X	0
1	1	X	↑	D
1	0	1	X	Q
1	0	0	↑	Q + 1

Using the Modules

The counter design makes extensive use (bits 0 through 5) of a 4 to 1 multiplexer driving a flip-flop as depicted in Figure 1. Both the multiplexer and the flip-flop will be combined into a single S-module by the ALS software. The select lines on the multiplexer are operated by the load control (S1) and by the counter enable and carry from the lower order bits (S0). The multiplexer data inputs are used for data to be loaded, held, or incremented.

C-modules are used as AND-EXORs and for ANDs to qualify the count function. The AND function is also used to bring the count enable (CE) to the multiplexer by means of the select line in bits Q3 and above. Using both the select inputs as well as the data inputs to AND lower order bits allows for more paralleling, which results in fewer levels of logic.

For the bits Q6 through Q15, an S-module macro with both a 4 to 1 multiplexer and an OR gate driving one select line is used to allow for more parallel propagation of the lower bits. Figure 2 shows the implementation of the most significant bit of the counter.

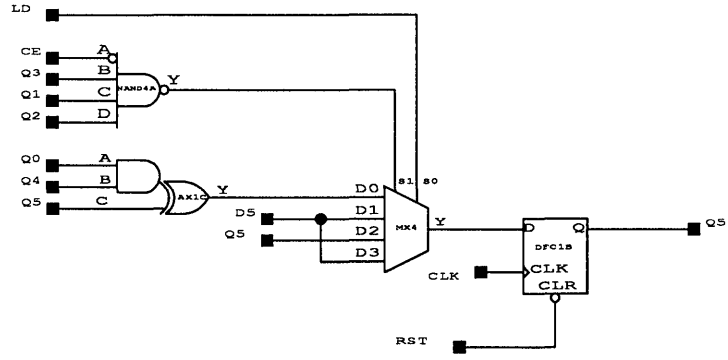


Figure 1. Counter Bit Q5

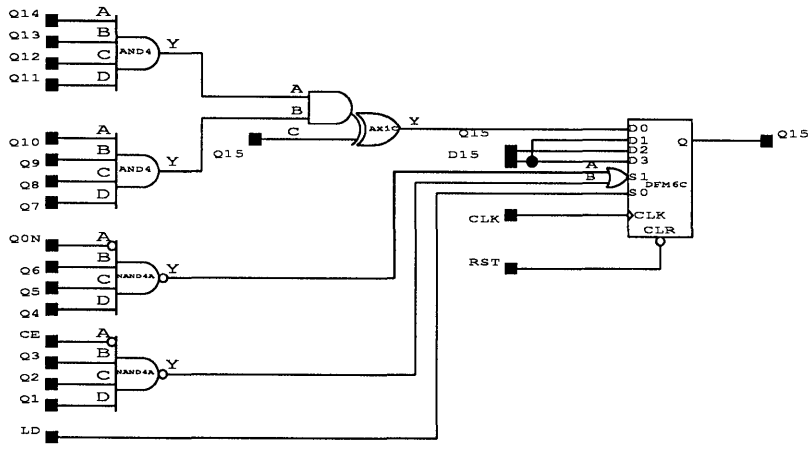


Figure 2. Counter Bit Q15

The S-module OR gate is used as a two-input NAND with active-low inputs that are, in turn, driven by NAND gates to propagate the lower bits and the count enable. The active-low output of the built-in two-input gate (OR used as a NAND) is adjusted for by shifting the position of the multiplexer data inputs.

Most four-input gates are implemented with a single C-module, but a four-input NAND with no bubbled inputs requires two modules. The limitation is avoided by using a NAND with a bubbled input. The count enable is active low, so it may be used to drive a bubbled input on a gate. Active-low counter bits are used to drive other bubbled gate inputs.

Levels of Logic

As may be seen in the complete counter schematic (Figure 3), two bits (Q0 and Q6) use inverters. The Q0 inversion toggles the flip-flop. A toggle flip-flop could have been used instead of a D flip-flop, but it could not have been combined with the multiplexer into a single S-module. Moreover, the inverter output is available as a resource to share the fanout load with the flip-flop and to allow the use of bubbled inputs on gates whenever it is desirable.

It could be argued that the use of the inverted output to drive gates causes the lower level bits to use two levels of combinatorial logic when it is not necessary. For a design of ten bits or less, the point would be valid because no path requires more than one combinatorial level. In the example design, however, two levels are already required by the upper bits and the improvement in fanout from the use of the inverter output at no additional cost in module count makes the practice worthwhile.

Limiting Fanout with Redundant Logic

The paths in the design most likely to limit performance are those with the largest number of logic levels and the highest fanout. In general, when fanout exceeds 9 on a critical path, redundant logic is often called for. For lower fanouts, the decision to use redundant logic might be a problem and must be balanced by considering both the cost in additional logic modules as well as the fanout to the outputs driving the redundant logic.

In the sample design, two redundant modules were added to illustrate the concept. One is the XNOR gate whose output is the inversion of Q1. The other is the four-input NAND gate that propagates flip-flop outputs. No fanout in the design exceeds seven, and the worst-case path is the redundant gate whose inputs are driven pins with fanouts of seven, six, six, and five, and whose output fanout is four. A total of 48 modules were used in the design.

Chip-Level Design Considerations

The trade-offs involved in soft macros such as the counter example cannot be evaluated outside the context of the overall design. Decisions such as whether to use redundant logic or high fan-in modules must consider the entire design. For example, if all the modules are already being used, redundant logic may not be an option. If the design has a significant number of high fan-in macros, additional high fan-in macros in a counter may cause routing congestion.

The ALS tools such as the Validator and Automatic Place and Route can rapidly provide answers to questions about design capacity and routability.

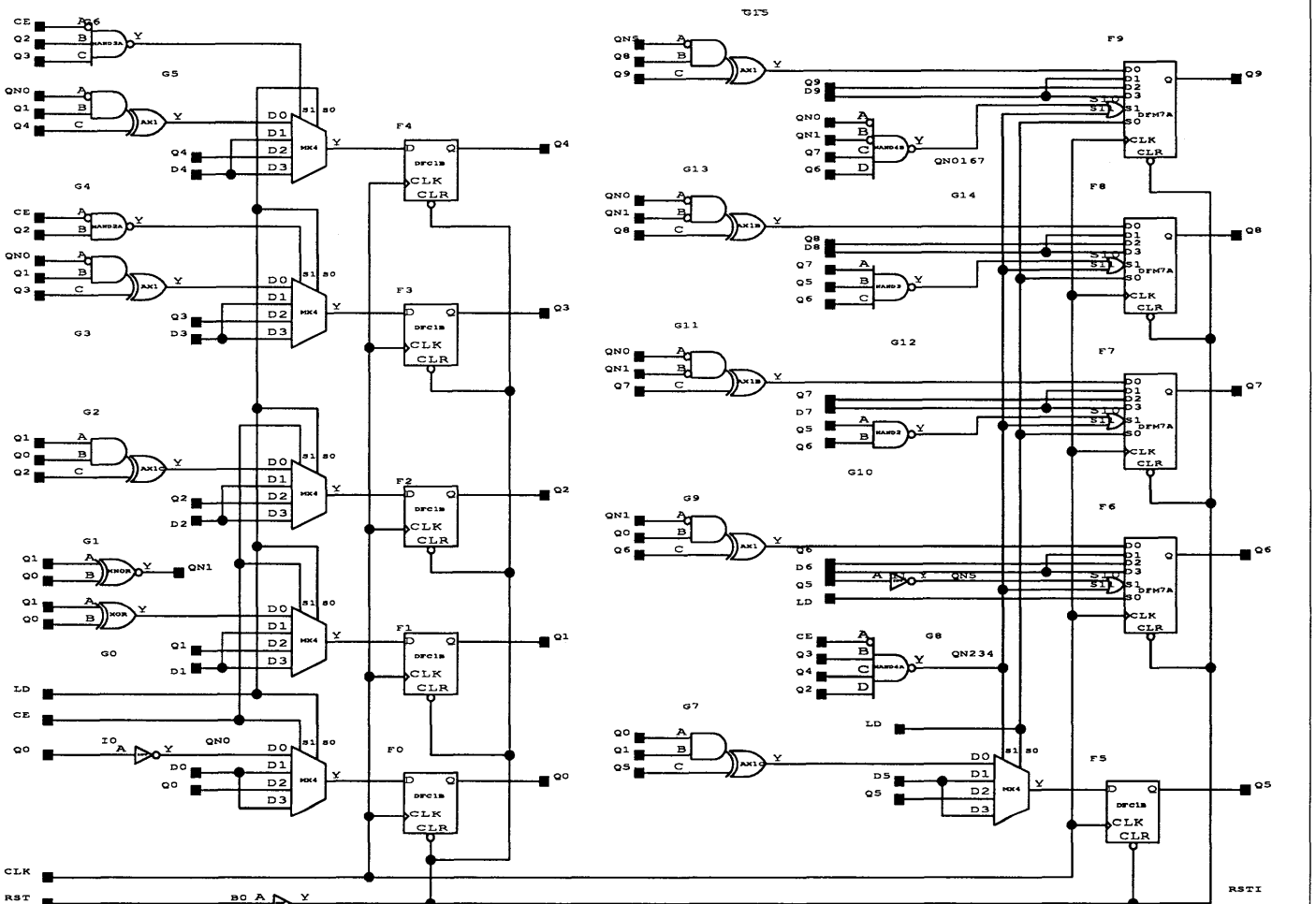


Figure 3. 16-Bit Counter Schematic

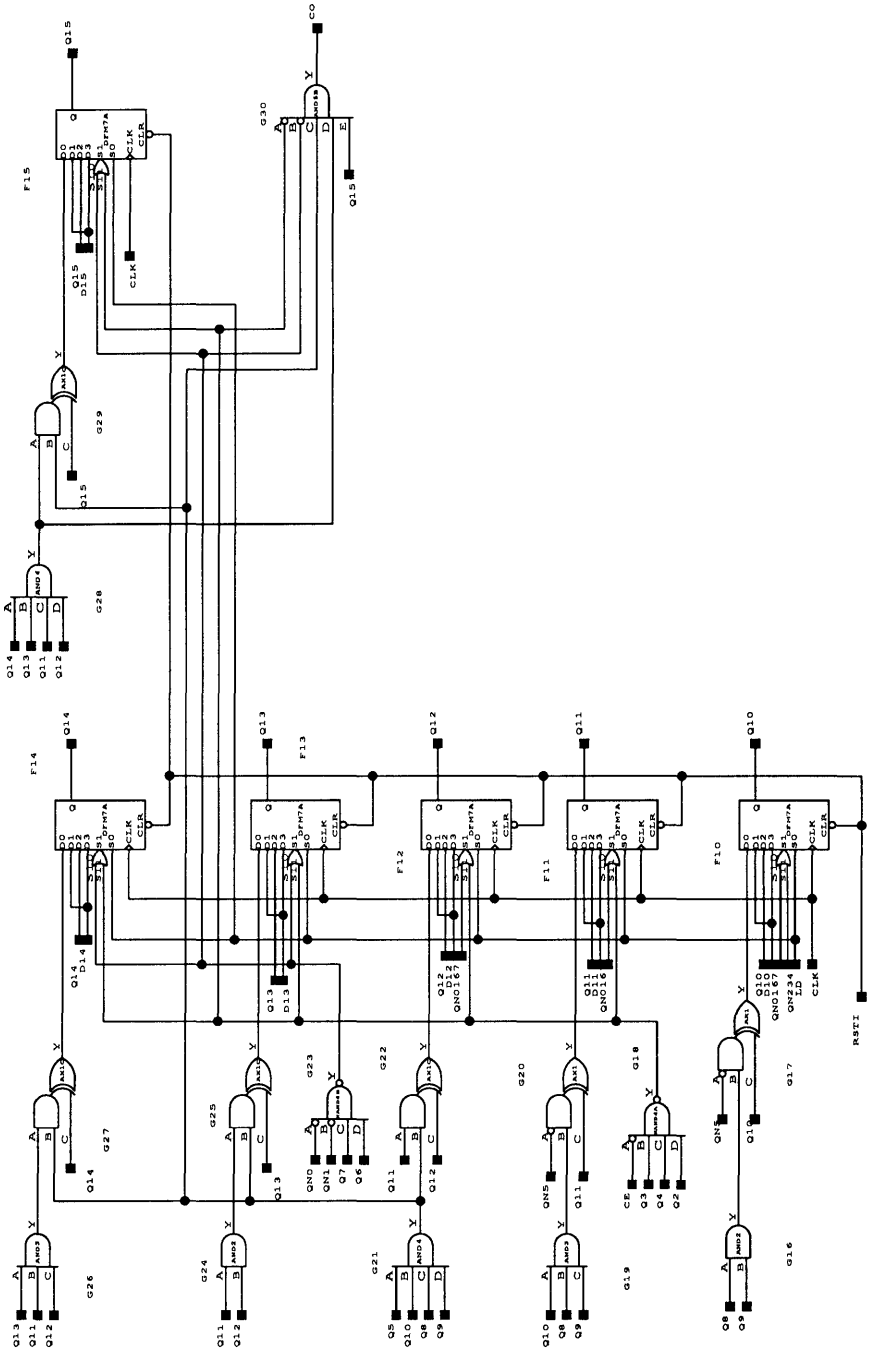


Figure 3. 16-Bit Counter Schematic (Continued)

Implementing Load Latency Fast Counters with ACT 2 FPGAs

Application Note

Introduction

Counters are one of the most important functions designed into field programmable gate arrays (FPGAs) and are frequently used as a benchmark to compare different technologies and products. The performance of a particular FPGA implementation is a function of the amount of logic used. Faster implementations use more logic and slower implementations use less. This application note will show a technique for making the highest possible performance counters in the ACT™ 2 family using a toggle prediction method to enhance performance.

An important realization in designing high performance counters is that the least significant bits (LSBs) of the counter change the most frequently and higher order bits change much less often. This fact can be used to optimize counter performance by ensuring that the least significant bits enter the logic trees at the lowest (fastest) level. Higher order bits can enter further up the tree since they have a longer time to propagate through the logic. Consider the design of a 6-bit counter using this technique. The counter will be loadable with asynchronous clear. It will also be a down counter, suitable for timing or address generation in a typical digital design. An up counter is an easy modification to the design, but conceptually they are similar.

Least Significant Bit

The counter must have a least significant bit that can toggle at the highest possible rate. In the ACT 2 family, the sequential logic module allows a 4-input multiplexer with gated select lines and a D-type flip-flop to be implemented in a single level of logic (see Figures 1 and 2). These logic modules can be used to construct a least significant bit with clear, load, and count enable as shown in Figure 3.

Data can be loaded into the register when the load enable (LD) signal is high, selecting the D1 or D3 input on the multiplexer. The count enable signal (CNT) is used to toggle Q0 when counting is enabled. If LD is disabled, the multiplexer input is either D0 or D2, depending on the state of Q0. If Q0 is a zero and CNT is a one, the register will be loaded with a zero (NOT CNT) from the D0 input. Thus Q0 toggles if CNT is a one. If CNT is a zero, Q0 will hold, not toggle, since the D0/D2 inputs will not be zero and one respectively. Thus, the least significant bit needs only a single level of logic to operate. Since the next most significant bit will only toggle when S0 is zero (implementing a down counter), there is one extra clock cycle to develop the signals for Q1's next state.

Figure 4 shows the implementation of Q1 using the DFM7A macro. It is similar to the LSB except that Q1 can be inverted to develop the toggle signal. Notice that this signal is selected only when Q0 and /CNT are low. This keeps the slower /Q1 signal from participating in the logic function until it has settled, ensuring fully synchronous operation. This technique will be the cornerstone of the most significant bits (MSBs), where slower signals are gated until there is sufficient time to settle and they are needed to compute the toggle of the associated counter bit.

Figure 5 shows the entire 2-bit prescaler (CNR2P) for fast counters. Added to Q0 and Q1 are three registers to source CNT, /CNT, and LD. These will be used to reduce fanout in the faster counter implementation.

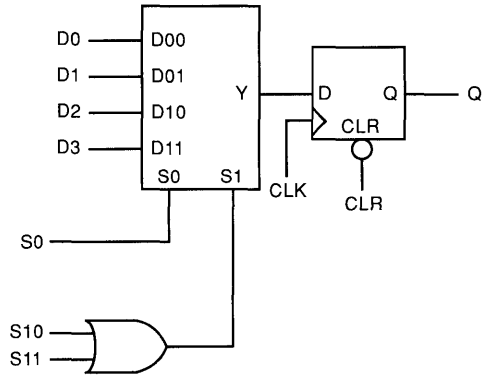


Figure 1. ACT 2 Family Sequential Logic Module

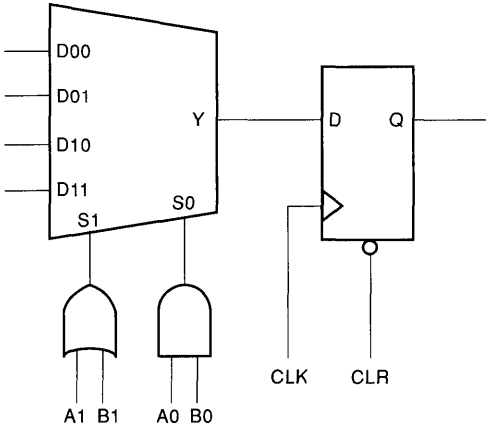


Figure 2. ACT 3 Sequential Logic Module

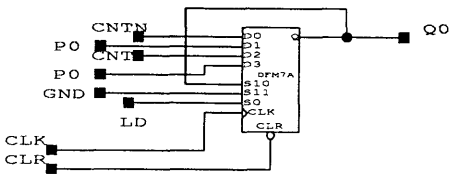


Figure 3. Counter Bit Q0

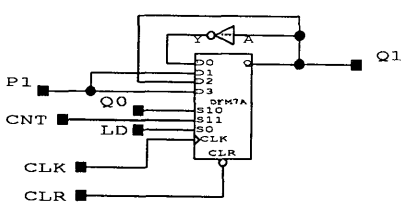


Figure 4. Counter Bit Q1

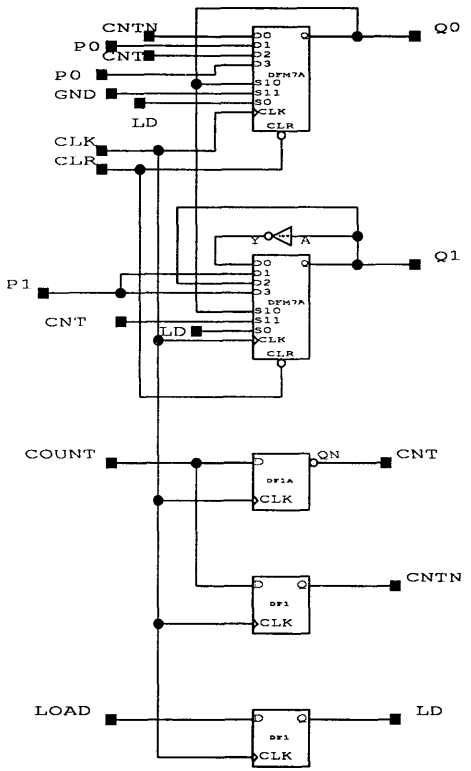


Figure 5. CNT2P Macro

Most Significant Bit

A 4-bit macro for the MSBs is shown in Figure 6. It uses the LSB from the CNT2P macro connected to CT0 (on S0A) and CNT (on S0B) to enable the multiplexer inputs used for toggling similar to the CNT2P macro. In addition, the next LSB is connected to CT1 (on B of AXB1) and the counter bits in the macro (Q0–Q2) are used to gate the input to the XOR (see Q3 for example), which determines whether the register bit holds or inverts. A ripple carry input (RCI) is also used to allow results from previous stages to participate. This technique allows MSBs at least four clock cycles (since the LSBs transition from 00 to 00 in four clock cycles worst case) to develop the input to the associated bits in the counter. A 6-bit counter is shown in Figure 7.

The limiting frequency for the 6-bit counter is based on the longest clock-to-clock delay in the design. There are four possibly limiting delays in the design: the minimum clock input period of the device, a single-level delay from Q0 to Q0–5, a two-level delay from Q1 to Q1–5, and a three-level delay from Q2 to Q2–5. Estimates for each delay are given below for an A1225A-2, over worst-case commercial conditions.

An 18-bit counter implemented with these macros is given in Figure 8. The limiting frequency for the counter is determined similarly to the 6-bit example, and the delays are given below for an A1225A-2.

Delay	Datasheet Parameter(s)	Value	Requirement
Minimum Clock Input Period	t_A =	9.4 ns	Must be < = 1 Clock Cycle
Q0 Delay	$t_{PD1} + t_{RD8} + t_{SUD}$ = 3.8 + 4.4 + 0.4 =	8.6 ns	Must be < = 1 Clock Cycle
Q1 Delay	$2[t_{PD1} + t_{RD8}] + t_{SUD}$ = 2[3.8 + 4.4] + 0.4 =	16.8 ns	Must be < = 2 Clock Cycles
Q2–5 Delay	$4[t_{PD1} + t_{RD8}] + t_{SUD}$ = 4[3.8 + 4.4] + 0.4 =	33.2 ns	Must be < = 4 Clock Cycles

Thus, maximum frequency is limited by the internal clock rate of 105 MHz.

Delay	Datasheet Parameter(s)	Value	Requirement
Minimum Clock Input Period	t_A =	9.4 ns	Must be < = 1 Clock Cycle
Q0 Delay	$t_{PD1} + t_{RD8} + t_{SUD}$ = 3.8 + 4.4 + 0.4	8.6 ns	Must be < = 1 Clock Cycle
Q1 Delay	$2[t_{PD1} + t_{RD8}] + t_{SUD}$ = 2[3.8 + 4.4] + 0.4 =	16.8 ns	Must be < = 2 Clock Cycles
Q2–5 Delay	$5t_{PD1} + 2t_{RD1} + 3t_{RD8} + t_{SUD}$ = 5(3.8) + 2(1.1) + 3(4.4) + 0.4 =	34.8 ns	Must be < = 4 Clock Cycles

Thus, maximum frequency is limited by the internal clock rate of 105 MHz.

The Maximum operating frequency of the 18-bit counter for other ACT 2 devices is shown below.

Use the corrensponding parameters from the ACT 3 datasheet to calculate the maximum frequency for ACT 3 devices.

	A1280A-2	A1240A-2
Maximum Clock Delay	11.7 ns	10.5 ns
Q0 Delay	10.9 ns	8.9 ns
Q1 Delay	21.4 ns	17.4 ns
Q2–5 Delay	42.9 ns	36.3 ns
Maximum Frequency	85 MHz	95 MHz

Conclusion

This application note explains a technique for creating very fast counters in the ACT 2 family showing operating frequencies as high as 105 MHz. These techniques will allow FPGAs to be used in new applications, which will further increase the popularity of these key devices.

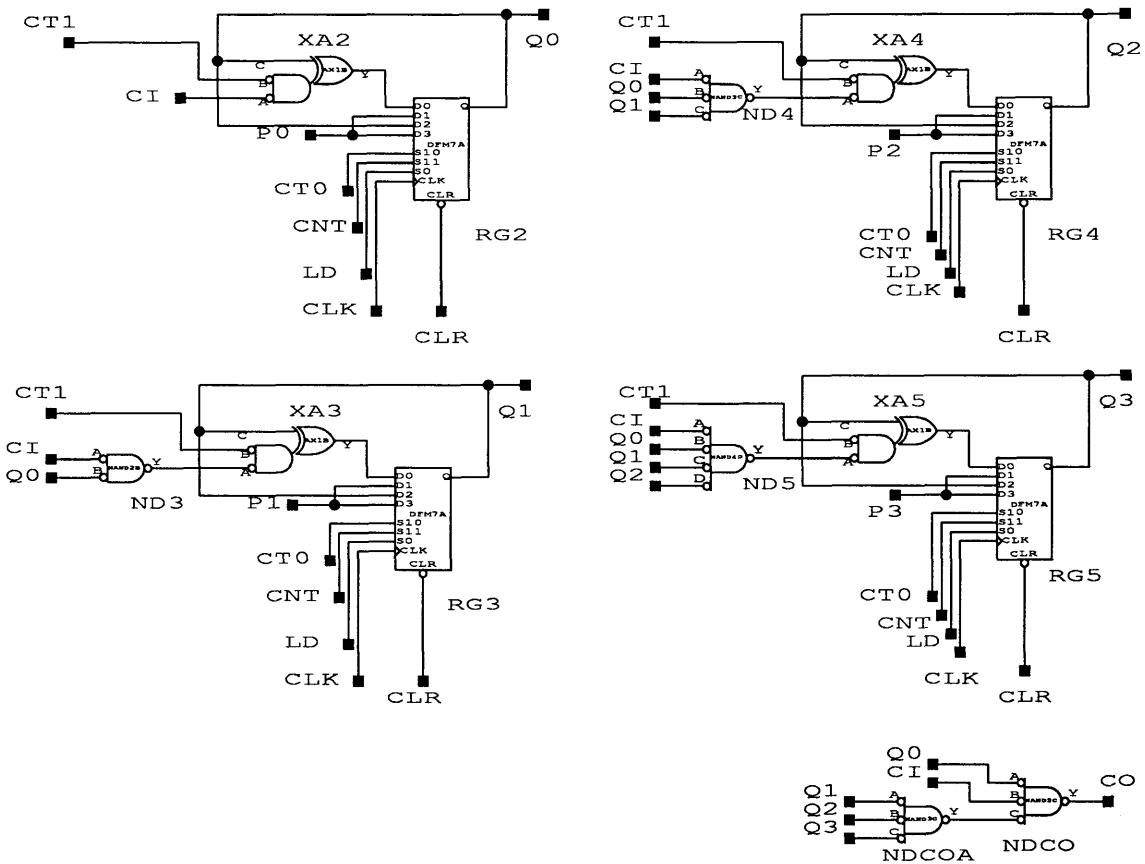


Figure 6. 4-Bit Counter Macro

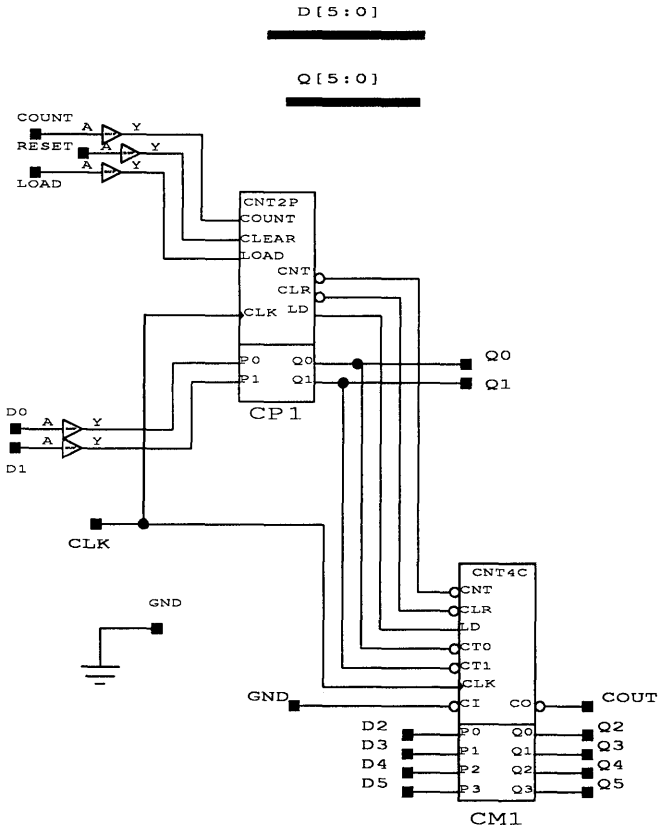


Figure 7. 6-Bit Counter

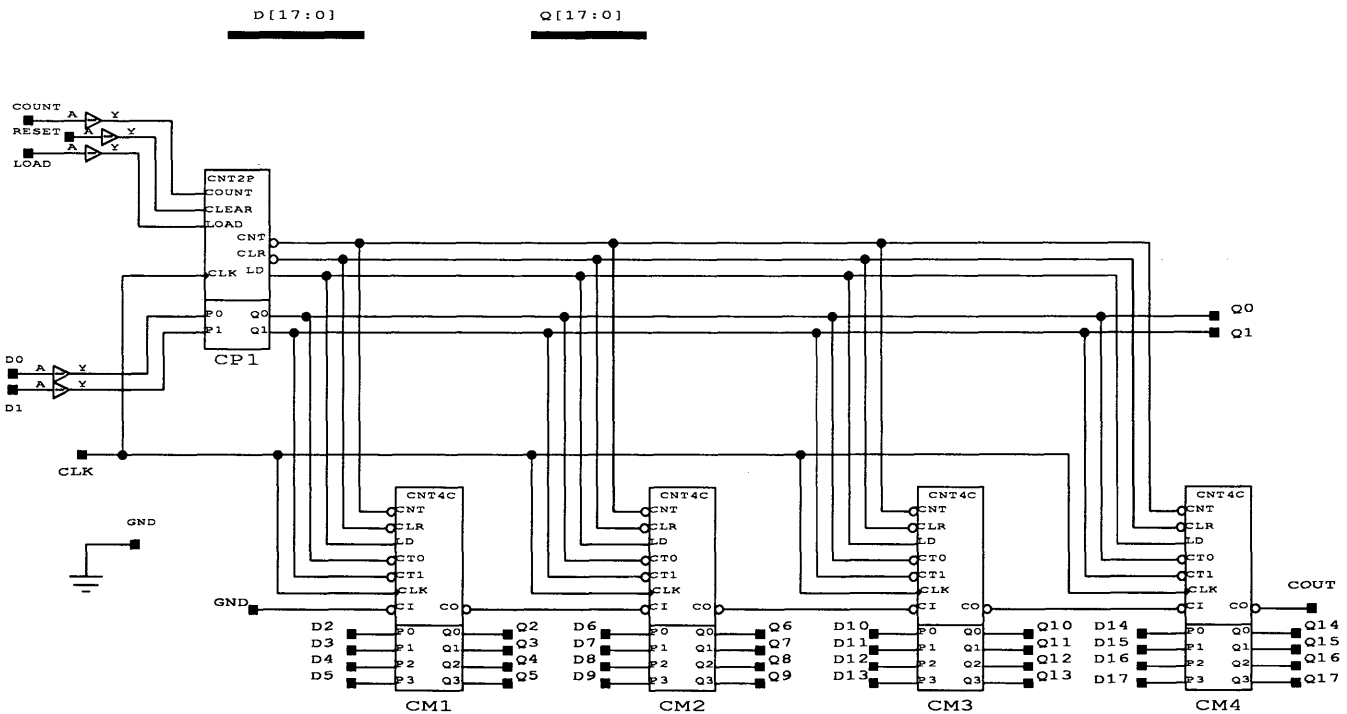


Figure 8. 18-Bit Counter

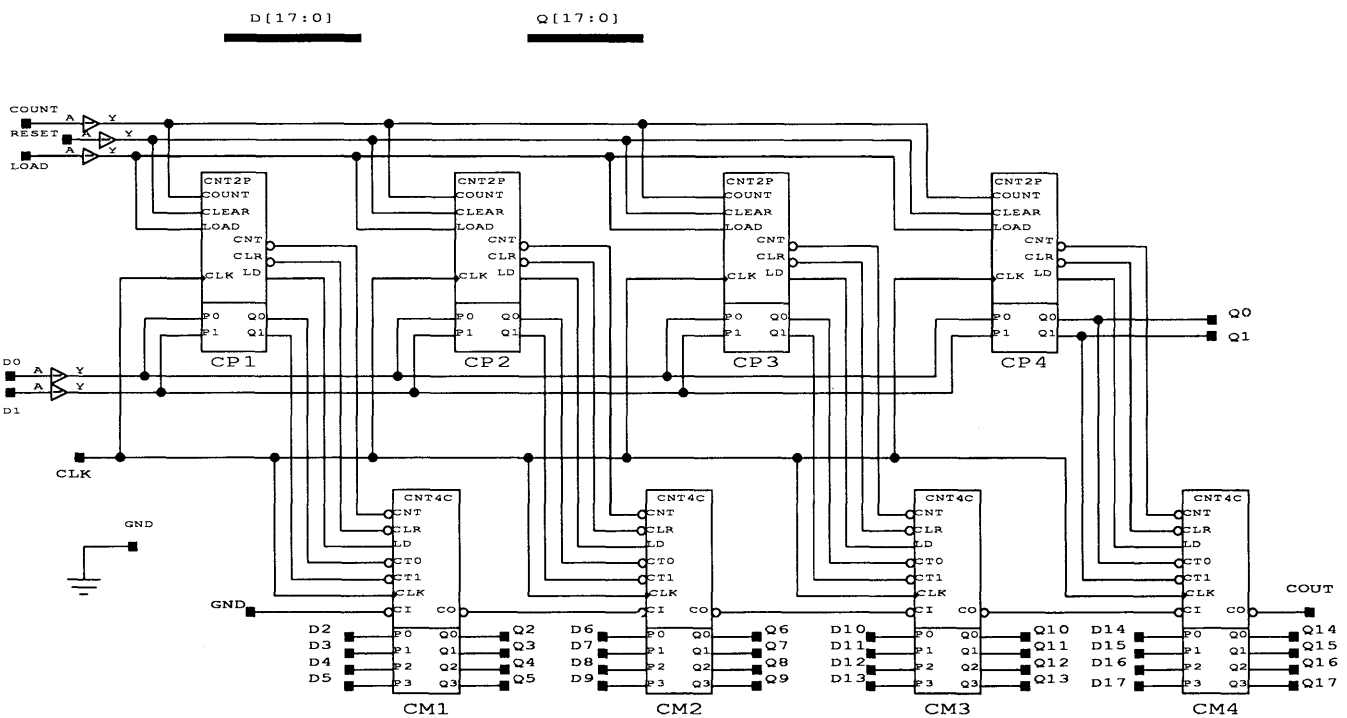


Figure 9. 18-Bit Counter with Fanout Reduced Design

Bit-Per-State Decoded State Machine for FPGAs

Application Note

Introduction

Signal controllers are one of the most important applications for Actel field programmable gate arrays (FPGAs). The controller accepts and interprets digital input signals and generates output signals in a sequential manner. Applications such as these can be designed with state machines that generate control output signals based on a function of the present and previous states of the input signals.

The traditional methodology for designing state machines has been to draw a state diagram, map the states into the minimum number of register bits, and determine the next state function for each register bit. The minimum number of register bits needed can be determined by rounding up the natural log of the number of states. This methodology results in a minimum number of registers but usually requires wide gating and complicated logic to encode the next state bit. This scheme is necessary to implement state machines using programmable logic devices (PLDs) because of their inherent lack of registers. Because registers are plentiful in FPGAs, state assignment is more efficient using bit-per-state methodology.

This application note will discuss techniques for efficiently implementing state machine using the bit-per-state for Actel FPGAs.

Sample State Machine

The state diagram of a sample state machine is illustrated in Figure 1. This state machine is the control section of a four-channel DMA controller supermacro for Actel FPGAs. The state machine contains six states, seven inputs and five outputs. Each circle represents a different state. Each arrow represents a transition between states. Inputs that cause state transitions are listed adjacent to the state transition arrows. Control outputs are labeled along with the states inside the circles. For example, in state 2, the state machine asserts the /CNTD and /CMREQ outputs. If the MACK input is low, the state machine remains at state 2. If the MACK input is high, the state machine goes to state 3 and asserts the /CE output in the process.

State Transition Equations

The next step is determining the logic to generate the state sequence. For bit-per-state implementation, assign each state to a separate register and write a state transition equation for each register. The state diagram in Figure 1 shows that state 0 (S0) can be realized when state 4 (S4) is asserted and the input CONT is low or it remains at state 0 if all four inputs A, B, C, and D are low. Therefore the transition equation of state 0 can be written as

S0 := /A*/B*/C*/D*S0 + /CONT*S4.

Similarly, state 1 (S1) can be achieved when state 0 (S0) is asserted and any one of the four inputs A, B, C, or D is high or it remains at the same state (S1) if the input PBGNT is low. The transition equation of state 1 can be derived as

S1 := (A+B+C+D)*S0 + /PBGNT*S1.

The complete state transition equations for the state machine are listed in Table 1. Note that state 3 (S3) will go to state 4 (S4) unconditionally; therefore, the transition equation for state 3 can be written as S3 := S4.

Table 1. State Machine Transition Equations

```
S0 := /A*/B*/C*/D*S0 + /CONT*S4;
S1 := (A+B+C+D)*S0 + /PBGNT*S1;
S2 := PBGNT*S1 + /MACK*S2;
S3 := MACK*S2 + MACK*S3;
S4 := S3;
S5 := CONT*S4 + /MACK*S5;
```

Output Equations

Once the state transition equations are determined, the output equations can be written by encoding the states. In some cases, the output is only active in one state. Therefore, the output is simply a function of that state. For example, CE is only active in state 3 (S3), so the output equation for CE is simply CE = S3. Other output may be active in more than one state. The output equations can be written simply as a function of those states. For example, CMREQ is active in state 2 (S2) as well as state 5 (S5). Therefore, the output equation is

CMREQ = S2 + S5.

Table 2 lists the output equations of the state machine.

Table 2. Output equations

PBREQ = S1;
CLD = S4;
CNTLD = S2;
CMREQ = S2 + S5;
CE = S3;

The state machine can be captured using schematic entry or automatically mapped using synthesis tools such as ACT'x'press™.

Figure 2 shows the schematic of the state machine implementation using the bit-per-state approach in the ACT™ 2 family. The circuit requires seven logic modules and three levels of logic modules.

Summary

A summary of the bit-per-state methodology is given below:

1. Draw a state diagram.
2. Assign each state to a separate register.
3. Write a state transition equation for each register.
4. Derive output equations based on active states.

Larger state machines can be implemented using this technique by distributing control to several smaller state machines and using a single master machine to coordinate activities between the state machines. This usually results in higher performance designs. It is also easier to design and debug simpler and smaller state machines.

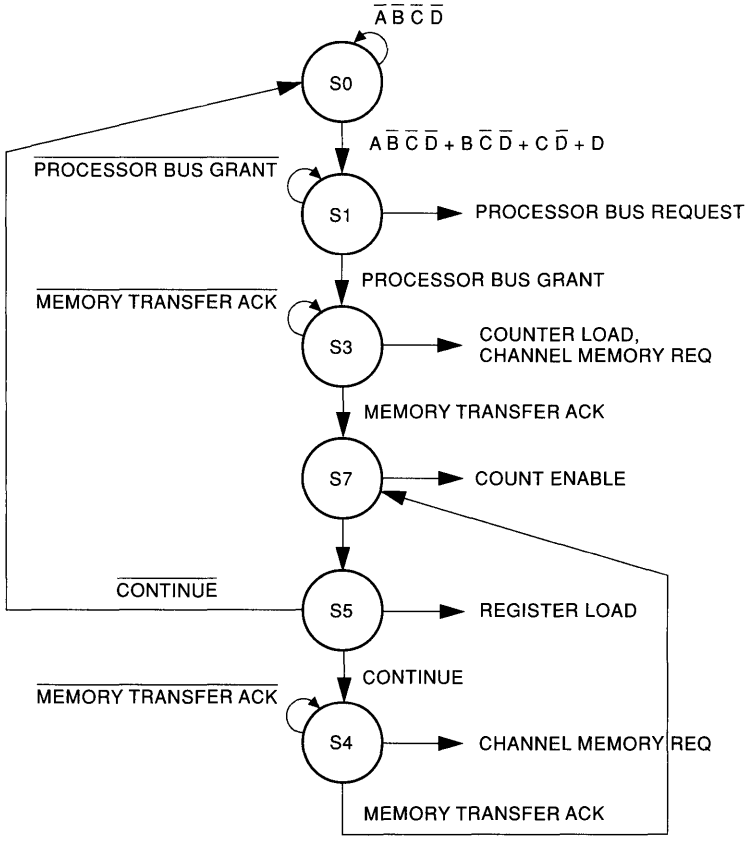


Figure 1. Four-Channel DMA Controller State Diagram (Control Section)

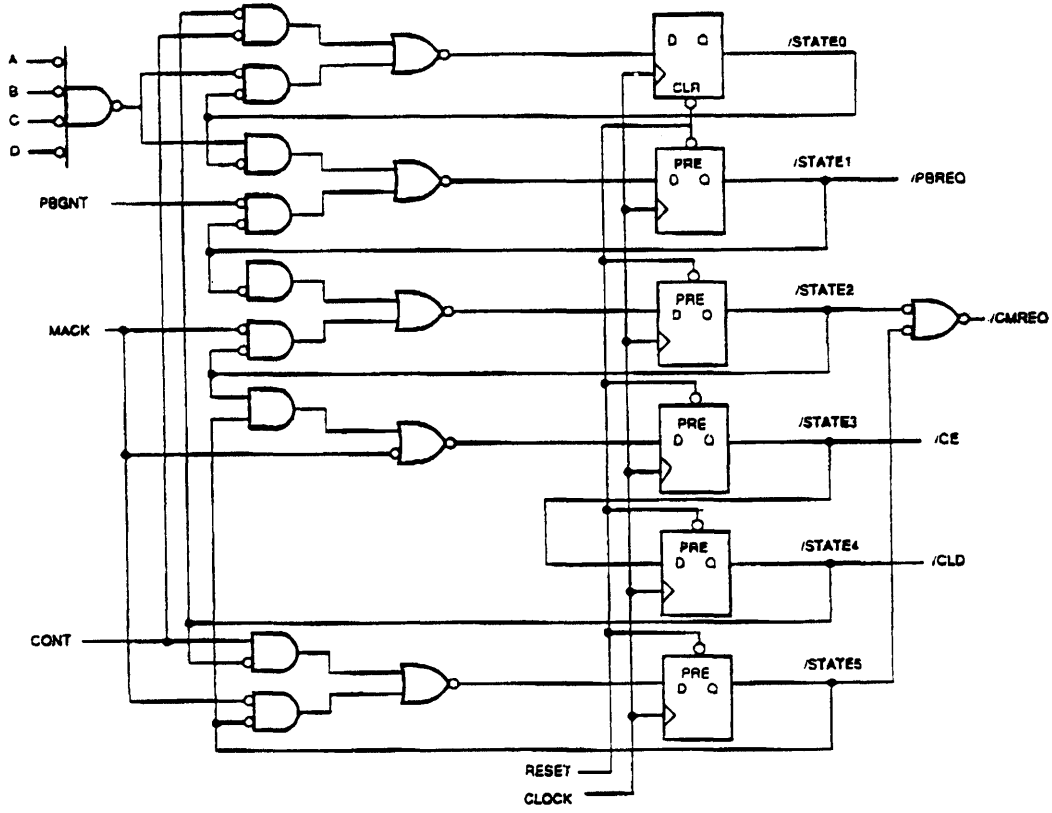


Figure 2. State Machine Schematic in AC1 2 Family

Implementing State Machines Using Shift Registers

Application Note

Introduction

Shift registers with serially controlled data inputs and parallel outputs may be used as powerful controlled sequence generators. As a result, a shift register can be used in high-speed state machine designs. This application brief shows a simple way to implement high performance state machines using serial shift registers.

Shift Register Design

A shift register is made of a series of flip-flops connected so that the data of one flip-flop is sequentially passed to the next flip-flop. It passes data from the beginning to the end of a "chain" of flip-flops. A clock signal synchronously controls this operation. For this application, a shift register sequentially shifts a single logic high signal while the rest of the output bits are low. This function can be implemented by simply connecting the output of the one flip-flop to the input of the next flip-flop and the output of the last flip-flop back to the input of the first flip-flop. Table 1 shows the function table of an eight-bit serial shift register. The width of the shift register is expandable by serially adding more flip-flops to the last stage. Figure 1, on the next page, shows the schematic of an eight-bit shift register. Note that the shift register is designed so that it can be set to a known state with a reset signal.

State Machine Implementation

The state machine implementation is best illustrated by an example. The sample state machine has six states with three output bits. The sequence is organized such that only one output bit changes state for every clock pulse. Figure 2 shows the state diagram of the state machine.

A six-state state machine requires a six-bit shift register with one register per state. Using this shift register determines the state sequence map of the state machine. The state machine creates the three output bits based on decoding the outputs of the shift register. The decoding logic is greatly minimized because there is only one output bit asserted in any state. Table 2 shows the state table of the state machine. When the outputs of the state machine are 001, the shift register outputs are 000010. In this state, Q5, Q4, Q3, Q2, and Q0 are all low and Q1 is high. Therefore, the state machine uses only Q1 for the decoding logic. The logic for the state machine outputs is based on the shift register outputs. Out0 is high when Q1 or Q2 or Q3 is high, Out1 is high when Q2 or Q3 or Q4 is high, and Out2 is high when Q3 or Q4 or Q5 is high. The complete decoding logic equations are the following:

Out0: Q1 + Q2 + Q3

Out1: Q2 + Q3 + Q4

Out2: Q3 + Q4 + Q5

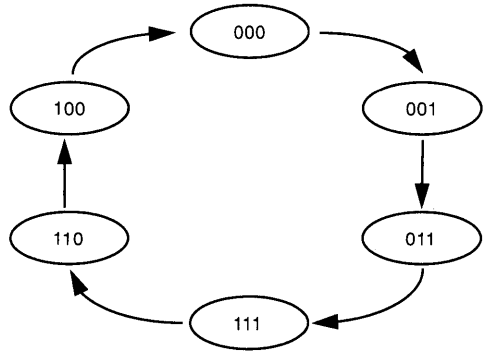


Figure 2. State Diagram of State Machine

Table 1. Eight-Bit Shift Register Function Table

RST	CLK	Q0	Q1	Q2	Q3	Q4	Q5	Q6	Q7
0	X	1	0	0	0	0	0	0	0
1	↑	0	1	0	0	0	0	0	0
1	↑	0	0	1	0	0	0	0	0
1	↑	0	0	0	1	0	0	0	0
1	↑	0	0	0	0	1	0	0	0
1	↑	0	0	0	0	0	1	0	0
1	↑	0	0	0	0	0	0	1	0
1	↑	0	0	0	0	0	0	0	1
1	↑	1	0	0	0	0	0	0	0

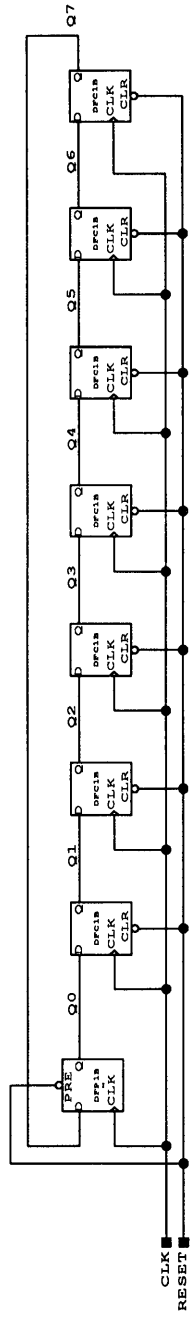


Figure 1. Eight-Bit Shift Register Schematic

Figure 3 shows the schematic diagram of the state machine using ACT™ 2 or ACT 3 macros. Note that the decoding logic only requires one level of logic to implement. Thus, using a shift register to implement state machines improves performance significantly.

Table 2. State Table for Example State Machine

Shift Register Outputs						State Machine Outputs		
Q5	Q4	Q3	Q2	Q1	Q0	Out2	Out1	Out0
0	0	0	0	0	1	0	0	0
0	0	0	0	1	0	0	0	1
0	0	0	1	0	0	0	1	1
0	0	1	0	0	0	1	1	1
0	1	0	0	0	0	1	1	0
1	0	0	0	0	0	1	0	0

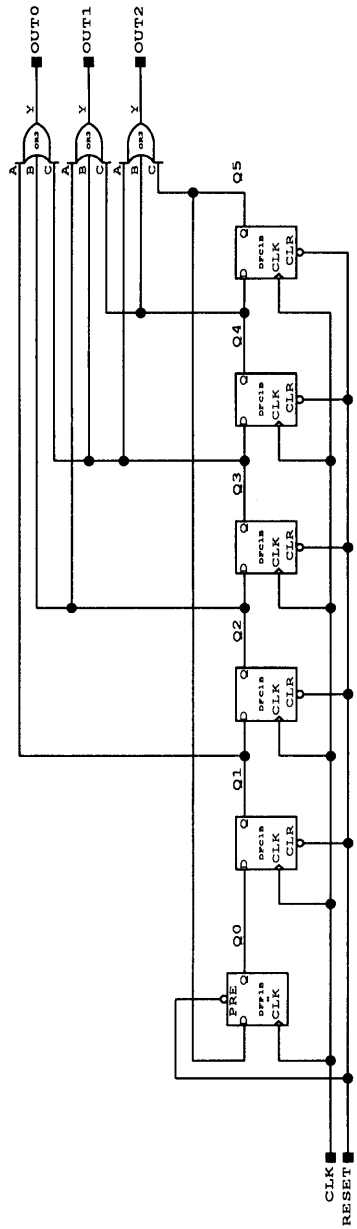


Figure 3. State Machine Schematic

Designing with Pseudo-Random Number Generators

Application Note

Introduction

As counter sizes increase, the amount and complexity of support logic also increase. With this increase, the maximum operating speed of the counter decreases. This application note describes an easy way to build pseudo-random number (PRN) counters using very few logic resources. These counters produce nonlinear sequences that can be used in many applications.

In many cases, only a simple modulo counter is required to generate a stream of clock pulses. Nonlinear counters can also be used to generate memory addresses. For example, in a FIFO, the order in which the memory is accessed is irrelevant as long as the data is stored and retrieved in the same order. This memory addressing technique could also be used for a ROM look-up table (LUT). In this case, the LUT data would be stored in ROM in the PRN sequence using a simple software routine to precalculate the addresses.

Pseudo-Random Counters

The most popular (and the simplest) PRN counter is the feedback shift register. Figure 1 shows a shift register of length m bits with the exclusive-OR (XOR) of the nth bit and the last (mth) bit fed back to the first bit location.

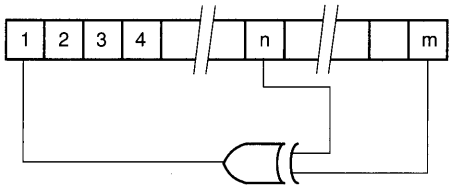


Figure 1. Feedback Shift Register

The PRN counter goes through a set of states (defined by the set of bits in the registers after each clock), eventually repeating itself after K clock pulses. The maximum number of conceivable states of an m-bit register is $K = 2^m$, that is, the number of binary combinations of m bits. However, the state of all zeros would get "stuck" in this circuit, since the XOR would generate a zero at the input. Thus, the maximum number of unique states is $2^m - 1$. It turns out that you can make "maximal-length shift register sequences" if m and n are chosen correctly. The resultant state sequence is pseudo-random. As an example, consider the 4-bit feedback shift register in Figure 2. Beginning with the state 1111 for the XOR implementation type and 0000 for the exclusive-NOR (XNOR) type, there are 15 distinct states ($2^4 - 1$), after which states repeat.

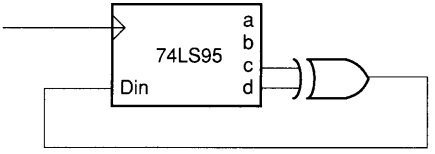


Figure 2. A 4-Bit Feedback Shift Register

Feedback Taps

Maximal-length shift registers can be made with XOR/XNOR feedback from more than two taps (in these cases, you use several XOR/XNOR gates in the standard parity tree configuration, that is, the modulo 2 addition of several bits). In fact, for some values for m, a maximal-length register can only be made with more than two taps. Table 1 shows a listing of all values of m up to 33 for which maximal-length registers can be made with just two taps, that is, feedback from the nth bit and mth (last) bit. A value is given for n and for cycle length K, in clock cycles. In some cases, there is more than one possibility for n, and in every case the value of $m - n$ can be used instead of n; thus the earlier 4-bit example could have used taps at $n = 1$ and $m = 4$.

Since shift register lengths of multiples of eight are common, you may want to use one of those lengths. In that case, more than two taps are necessary. There are other cases of shift register lengths that require more than two taps. Table 2 shows the taps points for the multitap point counters. Because of the greater complexity of the feedback path, the multitap counters tend to operate slower than the single tap version. It is sometimes more prudent to use a counter that is larger than needed just to reduce the number of taps. For example, if a modulo 250 counter is required, an 8-bit counter with three tap points could be chosen. But the wiser choice would be to use a 9-bit (max modulo 511) counter. The 9-bit counter actually uses two less logic modules (see next section) and has a higher operating speed.

Table 1. Values for Maximal-Length Registers Made with Two Taps

Length (m)	Tap (n)	Maximum Count (K)
2	1	3
3	2	7
4	3	16
5	3	31
6	5	63
7	6	127
9	5	511
10	7	1,023
11	9	2,047
15	14	32,767
17	14	131,071
18	11	262,143
20	17	1,048,575
21	19	2,097,151
22	21	4,194,304
23	18	8,388,607
25	22	33,554,431
28	25	268,435,455
29	27	536,870,911
31	28	2,147,483,647
33	20	8,589,934,591

Table 2. Tap Points for Multitap Counters

Length (m)	Tap (n)	Maximum Count (K)
8	3,4,5,7	255
12	1,9,10,11	4,095
13	6,10,11,12	8,191
14	1,11,12,13	16,383
16	10,12,13,15	65,535
24	16,21,22,23	16777215

PRN Counter Construction

Constructing a PRN counter requires only two elements—a loadable shift register bit and a fast XOR/XNOR gate. The Actel family of field programmable gate arrays (FPGAs) is ideal for building these types of elements. To construct the loadable shift register bit, use the DFM (D-type flip-flop with a 2 to 1 multiplexer) shown in Figure 3. Use the DFM6A hard macro with enable inputs to build the shift registers for cascadable PRN counters. The fast XOR/XNOR gate is implemented by one combinatorial logic module. A PRN counter can be constructed with these two basic circuits elements. Finally the END pattern can be chosen to minimize the complexity of the end-of-sequence (EOS) detection circuit.

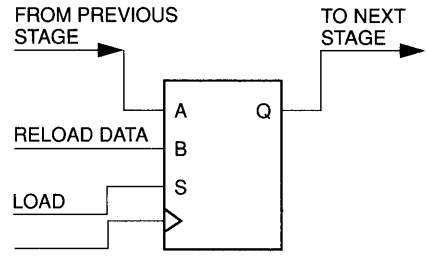


Figure 3. Loadable Shift Register Bit

One key feature of the PRN counter is that, as the counter length increases, the extra bits only add one extra shift register bit without increasing the feedback delay. For example, a 10-bit continuous PRN counter will take only 10 shift register bits (one DFM each) plus a fast XOR (one module). Increasing the length of the counter to 20 bits requires only an additional 10 shift register modules. The important point is that the feedback path still has only one level of logic delay. In other words, this 20-bit counter has the same feedback delay (maximum operating frequency) as the 10-bit counter.

PRN Sequence Software

Actel has developed a software utility called PRN_GEN2 for constructing PRN sequence counters. The software prompts you for the counter size (in bits), the end-of-sequence (EOS) pattern, the type of counter (XOR/XNOR), the desired modulo (cycles), and the pipeline delay of the EOS detect. Then, it provides you with the proper number to load to achieve the desired modulo and the tap points needed.

Note: You can have multiple modulos and use a multiplexer to select them.

Conclusion

PRN counters are easy to design and implement. They are useful for many types of applications and designs in Actel FPGAs. For more information about PRN counters and their implementation in Actel FPGAs, contact Actel Technical Support at 1-800-262-1060.

References

Paul Horowitz and Winfield Hill, *The Art of Electronics*, Cambridge University Press.

Implementing Three-State and Bidirectional Buses with Multiplexers in Actel FPGAs

Application Note

Three-state logic is used in conventional MSI logic devices to allow buses where multiple drivers are directly connected to one or more loads. Figure 1 shows a typical bus configuration with TTL three-state bus drivers and registers. Each driving device has a control input that places all outputs in a high impedance state when asserted. (For the register, the control pin is OC; for the bus driver, there are two control pins, 1G and 2G). To prevent data collisions, only one driver can be active at a time; the other drivers must be in a high impedance state. The four NAND gates perform a logic decoding function to ensure that only one driver is active at a time. If the first bus driver is selected to be active via SELA and SELB, then the data bits W0 to W7 will drive the bus (BUS0 to BUS7). Similarly, the other data bits will be selected when the respective register is active. The loads on the bus are not shown in Figure 1.

To make effective use of routing resources for many different applications, the Actel FPGA implements internal multiple drivers on a net with multiplexers instead of three-state logic. Figure 2 depicts the Actel implementation of the three-state bus discussed above. In this case, a 4 to 1 multiplexer is used for each bit of the bus to redirect the desired signal from one of four sources (W, X, Y, or Z). In addition to replacing the three-state nature of the MSI devices, the multiplexer eliminates the need for the LS241 bus drivers (inputs W and X). The desired source is selected by the two multiplexer select lines, which eliminate the need for the decoding logic used in the MSI implementation. For greater than four sources, the MX8 8 to 1 multiplexer can be used in a similar fashion.

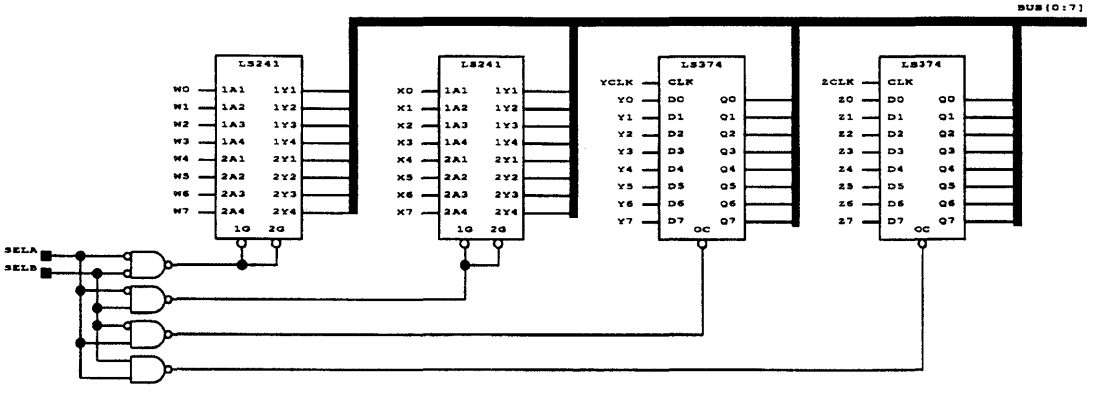


Figure 1. Three-State Bus Implementation with MSI Logic

9

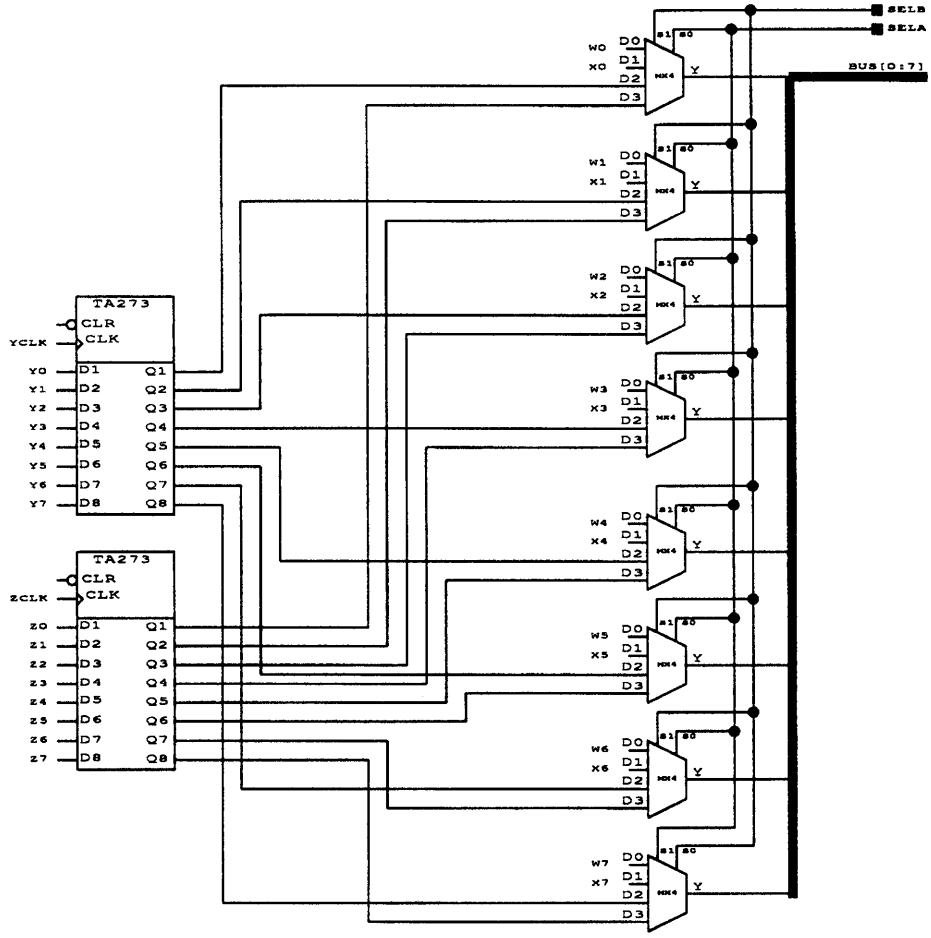


Figure 2. Three-State Bus with Actel FPGA Using Multiplexers

Bidirectional signals can also be replaced readily with multiplexers in the Actel architecture. Figure 3 shows the conversion of a popular transceiver to a 2 to 1 multiplexer.

Figure 4 illustrates the conversion to the multiplexer implementation of three bidirectional transceivers driving a bus.

The multiplexer circuit assumes that A's driver also drives any inputs on the A sub-bus, B drives B's inputs, and so on. Two additional sub-buses can be made available by controlling the select lines accordingly.

If the bidirectional element is located at the FPGA pad, the BIBUF macro can be used directly. Figure 5 shows a simple circuit using the LS245 transceiver with flip-flops driving and leading the bus. Figure 6 shows the Actel implementation of the BIBUF macro and the DFM multiplexed flip-flop. Note that when enable is low the pad drives the B flip-flop and that when enable is high, A drives both the PAD and B.

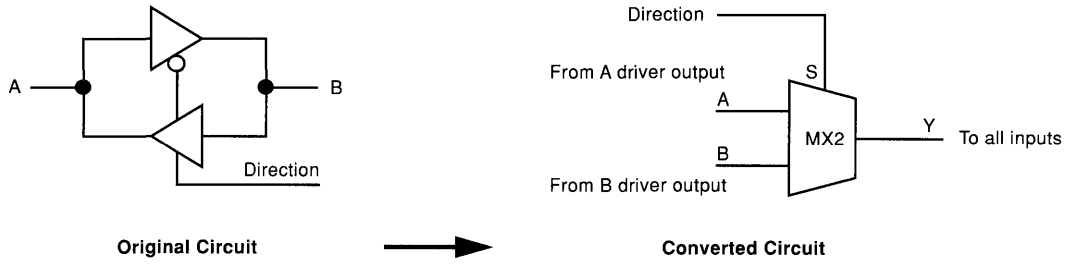


Figure 3. Transceiver Conversion to a 2 to 1 Multiplexer

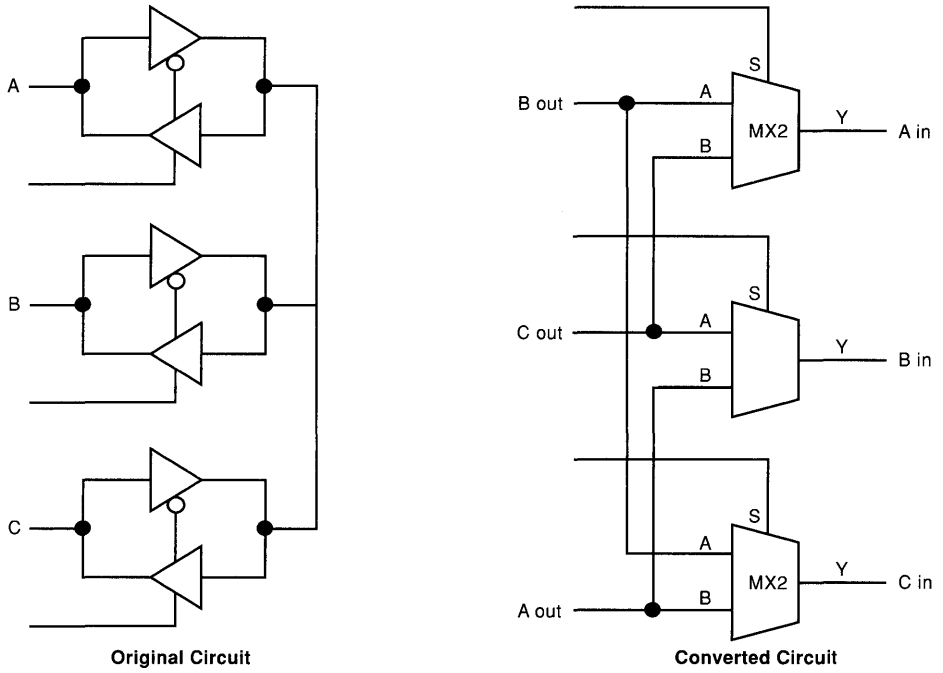


Figure 4. Three Bit Transceiver Conversion

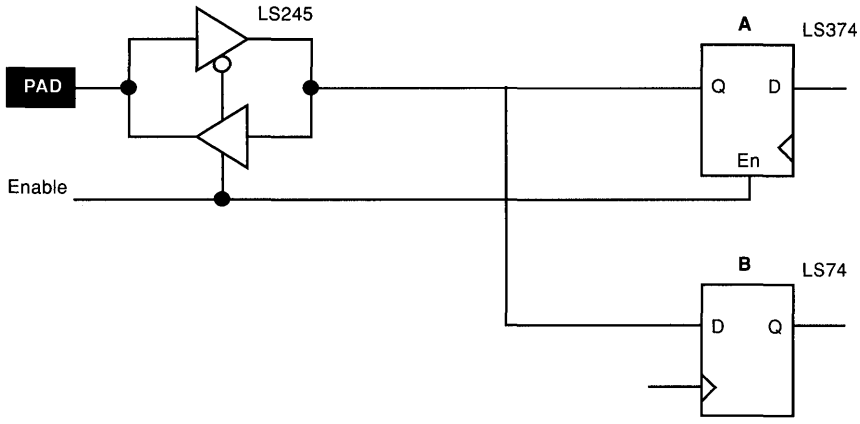


Figure 5. Bidirectional Bus with LS245

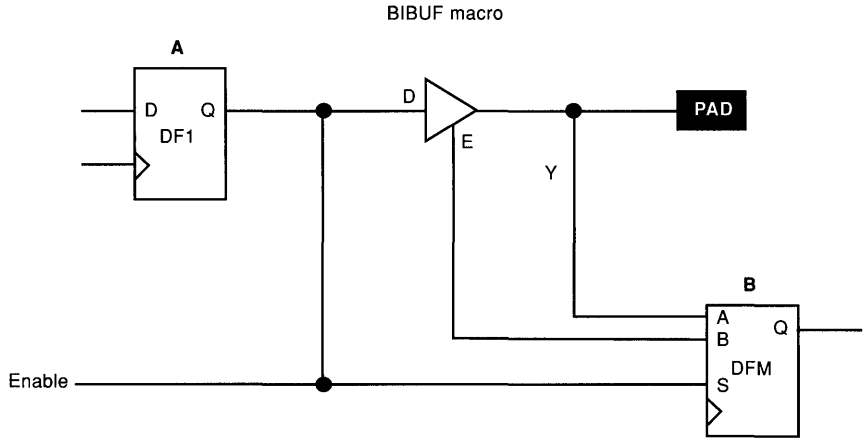


Figure 6. Bidirectional Bus with BIBUF Macro

Oscillators for Actel FPGAs

Application Note

Crystal Oscillator

Oscillators are fundamental design circuits used to provide a reference clock signal essential for digital designs. Crystal oscillators provide a simple solution for precise, stable, and calibration-free clocks. An on-chip crystal oscillator can be implemented with Actel devices using the traditional configuration shown in Figure 1. This oscillator has been tested up to 20 MHz and is used in Actel programmers. The 10 MΩ resistor provides a negative feedback path for the inverter, which makes it behave like a high-gain amplifier. The remaining passive elements, including the crystal, form a pi network that provides a 180 degree phase inversion. The inverter will lock on to the parallel resonant frequency of the crystal, thus providing a very stable output. The RC network also acts as a low-pass filter to ensure that the crystal operates at the fundamental frequency and not a harmonic frequency. The capacitor values range from 5 to 30 picofarads and depend on the crystal frequency. Some experimentation is suggested to get an optimal value for a specific design. Generally, the two capacitors will have the same value, although the capacitor connected to the input of the inverter can be varied independently to alter the output frequency by ± 0.1 percent.

A second OUTBUF or CLKBIBUF output buffer is used to provide a sharper clock signal at full amplitude when used outside the FPGA. Alternatively, the output buffer can be replaced with an internal buffer, which will allow a direct connection to internal macros. For ACT™ 2 and ACT 3 devices, the CLKINT macro can be used allowing high fanout drive capability of internal macros with minimal skew.

Fix the placement of the oscillator I/O macros to adjacent package pins to minimize internal delays. Consult the *ALS User Guide* for more information on fixing pins. Also, fixing the I/O macros near ground pins and far from other high-speed switching I/O pads minimizes noise effects.

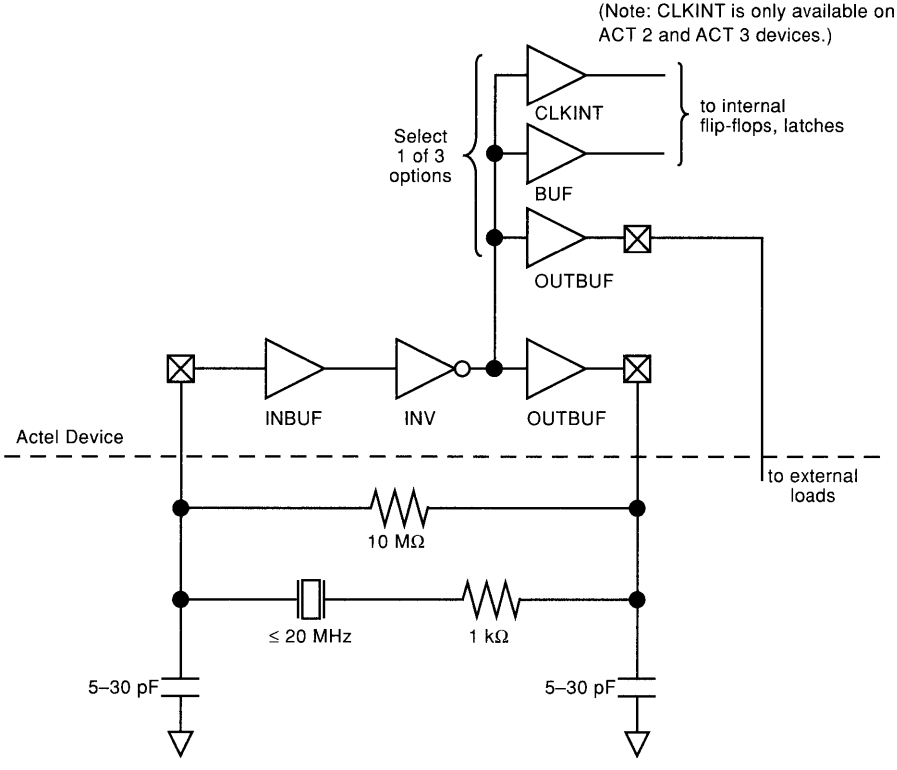


Figure 1. Crystal Oscillator Circuit

RC Oscillator

For applications not requiring the accuracy of a crystal, there is an RC oscillator that can be used in an Actel device as shown in Figure 2. As a strong word of caution, this circuit is not recommended for system clocks, since it is heavily dependent on resistor and capacitor tolerances, process variation, and temperature. Some applications for this lower cost oscillator include LCD backplane and debounce circuits.

The circuit reaches alternate switching thresholds by charging and discharging the capacitor with resistor R2. The R1 resistor provides a better square wave output by minimizing effects of input protection diodes of the input buffer. The approximate formula for output frequency is:

$$frequency \cong \frac{1}{2.2\ R2\ C}$$

The formula is most accurate if parameters have the following limitations:

R1 > 10R2

10K > R2 > 1 M

1000 pF > C > 10 µF

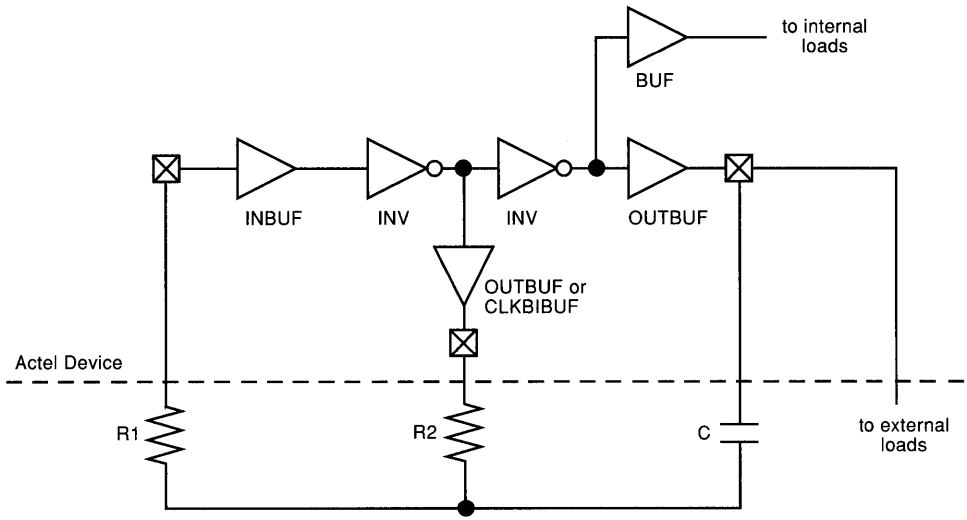


Figure 2. RC Oscillator

Page Mode DRAM Controller

Introduction

The ACT™ 2 DRAM controller supermacro allows you to access up to 16 MB of memory space from two different channels. Using automatic refresh circuitry and DRAM control logic, it can operate a 4 MB DRAM in page mode for up to 2,000 transfers starting from any point in a column. At the conclusion of a paged transfer, lock-out logic prevents any other access until all standard refreshes have been done. The supermacro uses just over 13 percent of an A1280. The schematic symbol and architecture of the DRAM controller are shown in Figure 1 and Figure 2.

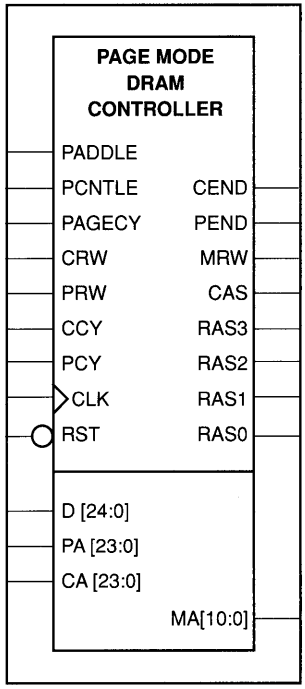


Figure 1. Schematic Symbol

Operational Overview

Address and Data Buses

Three buses in the controller bring the two channel addresses. A third bus with the processor data to load the counters for paged operation. The processor can request a single or paged memory transfer by means of address decodes. Peripheral channels can make memory transfer requests from a Direct Memory Access (DMA) interface.

Refresh Logic

The two counters (U0, U1) time refresh requests through an up/down counter (UDC). The UDC serves as an intermediate counter to pass the requests to the memory arbitration controller. When the memory is busy for long periods, (during page accesses, for example) the UDC stores refresh requests by counting up each time a request is made. When the memory is free, the UDC can request all the refreshes be made successively until the refreshes are caught up. Each time a refresh is complete, the UDC is counts down until it is cleared.

Arbitration State Machine

The arbitration state machine decides which channel has access to the memory and tells the memory timing control what type of access to begin. Its outputs are also used to select the source for the memory address and read/write source.

Memory access requests are prioritized from highest to lowest in the following order: refresh, processor cycle, page cycle, and channel cycle. When the memory is idle, the highest priority request begins a cycle. According to the type of request, the arbitration logic moves through a sequence of states until the timing control signals the end of the cycle.

Memory Timing Control

The memory timer issues the signals to control the operation of the DRAMs. It times the sequence and duration of the signals to conform to the requirements of the DRAMs and informs the arbitrator when the cycle is complete.

The timer contains a counter for paged accesses that will operate paged transfers up to 2,000.

Address Multiplexer

The address multiplexer selects the appropriate address source for memory accesses and, under the control of the memory timer, drives the DRAMs with the row or column address.

The multiplexer contains a register and a counter for paged accesses. Both are loaded by the processor. The register contains the address of the row for the page access and the counter has the beginning offset within the column. After each page access cycle, the column counts up by one to supply the address for the next cycle.

Bank and Read/Write

The two most significant bits of memory address are used for the bank select. They are selected by multiplexers from the sources for memory addresses. The multiplexer outputs are then decoded by the memory timer to drive one of the four RAS lines.

The processor loads a three-bit register with the page bank address and read/write line prior to initiating a page access.

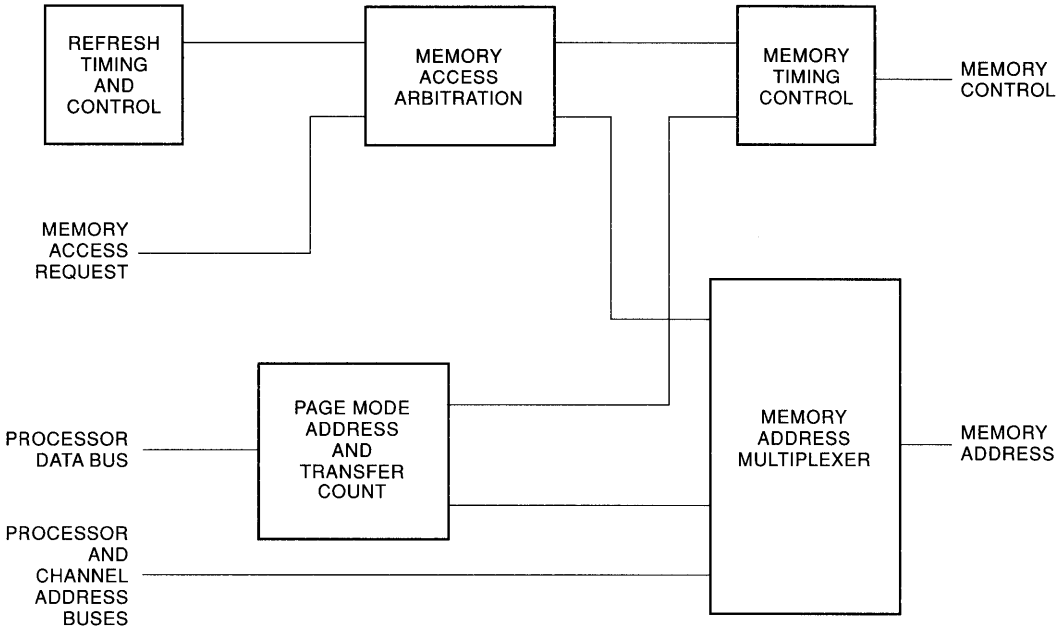


Figure 2. Page Mode DRAM Controller Block Diagram

Designing a DRAM Controller Using Language-Based Synthesis

Introduction

Why Use HDL?

A new level of abstraction for the design descriptions uses a combination of synthesis tools and HDL languages. As the trend toward more complex designs continues, the old ways of design entry do not accommodate everyone's needs. In the same way that higher level programming languages replaced assembly languages, traditional schematic capture and truth tables will be substituted by HDL design description.

A highly complex and detailed design can mislead you to overlook important details. Systematically partitioning the design into different subsystems and expressing the behavior of each subsystem in a higher level language helps to manage a big and detailed design. Such a specification is in a soft level, which can easily be changed, instead of in a hard level, which cannot be easily changed. This specification is unambiguous and complete and can hide the details of implementation.

Systems created with high-level constructs can be functionally simulated. Simulating the design at this level reveals problems before committing the design to hardware. An efficient HDL description accompanied by a sufficient test plan and host simulator that exercises the behavior of the design can minimize the number of design errors and reduce the time-to-market. Such a simulated behavioral description can later be synthesized for detailed gate-level implementations.

A behavioral description is also the best way to document a design. A well-prepared and documented HDL description can always describe a design better than a set of schematics with many gate-level details.

Why Use Verilog?

Some HDL languages, such as Verilog, resemble the C programming language very closely. Adopting the Verilog HDL-Synthesis approach can provide many benefits. Verilog HDL language is readable, elegant, and easy to learn. You only need to learn one language to create functional models, generate the design, create simulation vectors, simulate, and perform any other aspects of logic design. Verilog allows a design to be described in behavioral or structural terms or a mixture of both.

The detailed implementation of a Verilog code may be left to various synthesis tools.

Benefits of HDL Designs for the Actel Devices

Using a targeted vendor-specific library, most synthesis tools such as Synopsys, Cadence, Mentor Graphics®, and Auto Logic, can generate a gate-level representation of a design described in Verilog HDL.

Synthesis tools like the Synopsys Design Compiler are designed to optimize logic into regular conventional architectures. The Actel devices, with their highly regular channeled array architecture, can be used very efficiently with most existing synthesis tools. Synthesizing a design using ACT™ 1, ACT 2, or ACT 3 libraries can be optimized with respect to the speed, circuit size, or other cost functions where that gate utilization and performance are maximized.

Verilog/Synopsys Design Example

Complex designs are best suited for behavioral descriptions. Such a design can then be synthesized for the Actel devices using a synthesis tool such as Synopsys.

A good candidate for the HDL/Synthesis methodology is a DRAM controller. In such a design, many events, such as reset, refresh, and memory access, can happen at any time or simultaneously. Managing these events and arbitrating among them in a structural level can be mind boggling and error prone.

The following is an example of such a design.

Figure 1 demonstrates a typical design flow using Synopsys and the Cadence Actel kit. The following are the steps to create a design for an Actel device.

Step 1—Core Description

Most Verilog logic descriptions begin with a behavioral description of the core logic. The core logic description is found in the top level of hierarchy that contains the I/O and clock buffers.

Example 1 is the core logic captured in Verilog HDL for a DRAM controller. It controls the refresh, memory access, reset cycles, and address mapping for 4 MB of dynamic memory. The physical addressing space of this memory can be programmed to be mapped in any of 16 possible banks and designated with the top address bits (A22 to A25). Figure 2 is a block diagram description of this design.

A typical access from a processor is either I/O or memory. In this example, there are three control state machines that implement the I/O and the memory cycles. Figure 3 is their state diagram representation.

These state machines are directed with parallel_case Full _case Synopsys directives. These directives cause all the states to be evaluated in parallel and create all possible states that are not covered. The main state machine is gray-coded to achieve the minimal required logic decoding.

The initial I/O access will set the upper and the lower address boundaries of the memory bank. Any memory access to this DRAM should be in this addressing space.

The reset, memory, and refresh cycles are prioritized. A valid address sent to the memory will start a Read or Write cycle. The RAS, CAS, address switch, and other handshake lines will be

generated accordingly. If no other memory access is in progress, a refresh request will start a CAS cycle before a RAS cycle. If some other cycle is in progress, it will be latched to be granted at a later time. A reset always has the highest priority; it resets all the cycles to their initial states.

Step 2—I/O Buffers

I/O buffers must be either instantiated in the Verilog code or automatically inserted by Synopsys. Synopsys V3.0 contains a command, insert_pads, that automatically inserts INBUF, OUTBUF, and CLKBUF. If the design must contain BIBUF, TRIBUF, or any special ACT 3 I/O cell, they must be hand instantiated in the code. Example 2 shows how to hand instantiate I/O buffers in the Verilog code. A level of logic, or "top level," is created to merge the core logic with the I/O buffers.

Step 3—Test Circuit

A Verilog test file needs to be generated to Simulate the design and verify behavioral level functionality. Example 3 demonstrates this test circuit.

Step 4—Invoke the Synopsys Tool

The Synopsys setup file (.Synopsys) needs to be set for appropriate Actel devices. The EDIF options need to be set correctly to generate an EDIF netlist appropriate for ALS. Following is an example of a desirable .synopsys file.

Example of .Synopsys file

```
search_path = { . /user1/ashena/ /proj1/3.0synop/
admin/install/ /proj1/3.0synop/libraries/syn};
designer = "DESIGNER NAME"; company = "Actel";
target_library = /cad2/cad/release/cae/3.0/synop-
sys/2.2/lib/act2_30.db;
symbol_library = /cad2/cad/release/cae/3.0/synop-
sys/2.1001/lib/act2.sdb; link_library = /cad2/cad/-
release/cae/2.2/synopsys/2.2/lib/act2_30.db;
edifout_netlist_only = true
edifout_no_array = true
edifout_power_and_ground_representation = cell
edifout_ground_name = GND
edifout_power_name = VCC
edifout_ground_pin_name = y
edifout_power_pin_name = x
edifout_pertty_print = true
```

The Verilog code should be read in the Synopsys environment for it to synthesize and optimize the behavioral circuit. The optimization level can be selected to be high, medium, or low.

Synopsys can write out an EDIF file as well as a Verilog structural level output file. The ALS program EDN2ADL will translate the generated EDIF file to ADL (Actel Design Language). The ALS software places and routes the ADL netlist into an Actel field programmable gate array (FPGA).

Synopsys also generates a gate-level schematic for reference or debugging.

Step 5—Simulate the Structural Verilog File

Resimulate the Synopsys generated structural level design to verify the gate-level functionality. The same test file used in step 3 is also used to further simulate the gate-level DRAM Controller design. The two simulator results are compared.

Step 6—Generate an Actel Netlist

The Actel program, cae2adl, produces an Actel ADL netlist from the Synopsys generated EDIF. The program is executed from the command line with several input switches. The syntax for invoking cae2adl is:

cae2adl -edn2adl fam:<*value*> **ednin:**<*edif file name*> <*design name*>

Where:

fam	is family, ACT 1, ACT 2, or ACT 3
edin	is the file to be converted, which must have the name <design name>.eds
<design name>	is the name of the design, which must be the same name as the top level of hierarchy, that was synthesized

Step 7—Place and Route the Design

The ALS software verifies the integrity of the design, assigns I/O pin numbers, places, and routes the design, and generates a fuse file for programming. After place and route, the del2vlog program backannotates actual delays for simulation.

Step 8—Run Actel Timer

The Actel timer is a static timing verifier. Unlike simulators, the Timer does not require test vectors. The Timer tool is useful for verifying internal and external setup and hold time requirements, clock skew, and maximum frequency.

Step 9—Post-Route Simulation

To perform postlayout simulation, a PLI routine ($als_add_delays) must be added to the stimulus test file. The syntax for this routine is:

$als_add_delays ("<module>.<instance>", "<design_name>.del", "2.2", "<design_name>.def)

For more detailed information regarding the Actel/Synopsys/Verilog interface, refer to the Action Logic® System, Release 2.2, CAE Guide, Sun/Cadence manual part number 5029047-0.

The DRAM controller core logic is presented in the following pages.

Example 1. DRAM Controller

```
/*              DRAM CONTROLLER
This state machine controls the memory handshake, refresh, and address decoding for a DRAM */
`timescale 1 ns / 100 ps
module dram1 (ras_,cas_,rw_en,ready_,mux_add,sysclk,add,ads_,reset,rw_,ref,data,m_io);
output ras_,cas_, rw_en, ready_; output [11:2] mux_add;
input sysclk;
input [25:2] add;
input ads_, rw_, ref, reset, m_io;
input [7:0] data;
reg [3:0] low_add;          /* lower address bytes */
reg [3:0] up_add;           /* upper address bytes */
reg [1:0] state;            /* I/O states */
reg [6:0] cs;               /* Current State */
reg [6:0] ns;               /* Next State */
reg rw_en, refreq, lat_ads_; wire mux_sel, hit, ref_end, io_ready_, m_ready_;
parameter s1=7'b1111111, s2=7'b1011111, s3=7'b1011110, s4=7'b1011100,s5=7'b1011000,  s6=7'b1010000,  s7=7'b1101011,
s8=7'b1101010,s9=7'b1001010, s10=7'b0001111;
/*————————————————ASSIGNMENTS————————————————————————————————*/
/* SET the memory and refresh handshake signals */
assign ras_ = cs[0]; assign mux_sel = cs[1]; assign cas_ = cs[2]; assign m_ready_ = cs[3]; assign ref_end = !cs[6];
/* RAS address or CAS address */
assign mux_add = mux_sel ? add[11:2] : add[21:12];
/* Is the address in the boundary? */
assign hit = (add[25:22] >= low_add[3:0]) && (add[25:22] < up_add[3:0]); assign io_ready_ = !state[1]; /* End of I/O access */
assign ready_ = (io_ready_ & m_ready_);/     * End of I/O or Memory access */
/* I/O ACCESS TO SET THE UPPER AND LOWER ADDRESS BYTES*/
always @ (posedge sysclk)
if (reset) begin
/* Only a subset of Verilog Language is supported by Synopsys */
        state = 2'b00;
        // synopsys translate_off
                fork
        // synopsys translate_on
        up_add [3:0] = #2 4'h0;
        low_add [3:0] = #2 4'h0;
        // synopsys translate_off
                join
        // synopsys translate_on
                end
                else
        case (state)    //synopsys parallel_case full_case
        00 : if (!reset & !m_io & !rw_ & !ads_) state = 2'b01;
        01 : begin
```

Example 1. DRAM Controller (Continued)

```
                        // synopsys translate_off
                                fork
                        // synopsys translate_on
                                low_add [3:0] = #2 data[3:0];
                                up_add [3:0] = #2 data[7:4];
                        // synopsys translate_off
                                join
                        // synopsys translate_on
                                state = 2'b10;
        end
        10 : state = 00;
                        default : state = 2'b00;
                        endcase
/*      MEMORY and REFRESH ACCESS */
always @(posedge sysclk) cs = #3 ns;
always @(ads_ or reset or refreq or cs or rw_ or hit or m_io) begin if (reset) begin
        ns=s1;
        #2 rw_en = 1;
        end else
        case (cs)                       //synopsys parallel_case full_case
        s1: begin
                        rw_en = 1;
                        if (refreq) ns = s7;
        /* memory cycle starts */
                        else if ((!ads_ | !lat_ads_) & !refreq & m_io ) ns=s2;
                        else ns=s1;
                        end
        s2: begin
                        if (hit) ns=s3;  /* Address is in the boundary */
                        else ns=s1;
                        end
        s3: ns=s4;
        s4: begin
                        ns=s5;
                        if (!rw_) #2 rw_en = 0;
                        else #2 rw_en = 1;
                        end
        s5 : ns=s6;
        s6 : ns=s1;
        s7 : ns = s8;
        s8 : ns = s9;
        s9 : ns = s10;
        s10: ns = s1;
        default: ns = s1;
        endcase end
```

Example 1. DRAM Controller (Continued)

```
always @ (posedge sysclk) if (reset) refreq = 0;
                  else if (ref & !ref_end) refreq = 1;
                  else if (ref_end) refreq = 0;
always @ (posedge sysclk)
if (reset) lat_ads_ = 1;
                  else if (refreq & !ads_ & m_io) lat_ads_ = 0;
                  else if (!m_ready_) lat_ads_ = 1;
endmodule
```

Example 2. Top-Level Design

```
`timescale 1 ns / 100 ps
module topd (pras_,pcas_,prw_en,pready_,pmux_add,psysclk,padd, pads_,preset,prw_,pref,pdata,pm_io);
input psysclk;
input [25:2] padd;
input pads_, preset, prw_, pref, pm_io;
input [7:0] pdata; output pras_, pcas_, prw_en, pready_;
output [11:2] pmux_add;
wire sysclk;
wire ras_, cas_, rw_en, ready_;
wire [11:2] mux_add;
wire [25:2] add; wire ads_, reset, rw_, ref, m_io; wire [7:0] data;
/* Instantiate the Core logic */
dram1 u1 ( ras_, cas_, rw_en, ready_, mux_add, sysclk, add, ads_, reset, rw_,ref, data, m_io);
/* Connect the I/O and Clock buffers */
CLKBUF UC1 (.PAD(psysclk), .Y(sysclk));
OUTBUF UO0 (.PAD(pras_), .D(ras_));
OUTBUF UO1 (.PAD(pcas_), .D(cas_));
OUTBUF UO2 (.PAD(prw_en), .D(rw_en));
OUTBUF UO3 (.PAD(pready_), .D(ready_));
OUTBUF UO5 (.PAD(pmux_add[2]), .D(mux_add[2]));
OUTBUF UO6 (.PAD(pmux_add[3]), .D(mux_add[3]));
OUTBUF UO7 (.PAD(pmux_add[4]), .D(mux_add[4]));
OUTBUF UO8 (.PAD(pmux_add[5]), .D(mux_add[5]));
OUTBUF UO9 (.PAD(pmux_add[6]), .D(mux_add[6]));
OUTBUF UO11 (.PAD(pmux_add[7]), .D(mux_add[7]));
OUTBUF UO12 (.PAD(pmux_add[8]), .D(mux_add[8]));
OUTBUF UO13 (.PAD(pmux_add[9]), .D(mux_add[9]));
OUTBUF UO14 (.PAD(pmux_add[10]), .D(mux_add[10]));
OUTBUF UO15 (.PAD(pmux_add[11]), .D(mux_add[11]));
INBUF UI0 (.PAD(padd[2]), .Y(add[2]));
INBUF UI1 (.PAD(padd[3]), .Y(add[3]));
INBUF UI2 (.PAD(padd[4]), .Y(add[4]));
INBUF UI3 (.PAD(padd[5]), .Y(add[5]));
INBUF UI4 (.PAD(padd[6]), .Y(add[6]));
INBUF UI5 (.PAD(padd[7]), .Y(add[7]));
```

Example 2. Top-Level Design (Continued)

```
INBUF UI6 (.PAD(padd[8]), .Y(add[8]));
INBUF UI7 (.PAD(padd[9]), .Y(add[9]));
INBUF UI8 (.PAD(padd[10]), .Y(add[10]));
INBUF UI9 (.PAD(padd[11]), .Y(add[11]));
INBUF UI10 (.PAD(padd[12]), .Y(add[12]));
INBUF UI11 (.PAD(padd[13]), .Y(add[13]));
INBUF UI12 (.PAD(padd[14]), .Y(add[14]));
INBUF UI13 (.PAD(padd[15]), .Y(add[15]));
INBUF UI14 (.PAD(padd[16]), .Y(add[16]));
INBUF UI15 (.PAD(padd[17]), .Y(add[17]));
INBUF UI16 (.PAD(padd[18]), .Y(add[18]));
INBUF UI17 (.PAD(padd[19]), .Y(add[19]));
INBUF UI18 (.PAD(padd[20]), .Y(add[20]));
INBUF UI19 (.PAD(padd[21]), .Y(add[21]));
INBUF UI20 (.PAD(padd[22]), .Y(add[22]));
INBUF UI21 (.PAD(padd[23]), .Y(add[23]));
INBUF UI22 (.PAD(padd[24]), .Y(add[24]));
INBUF UI23 (.PAD(padd[25]), .Y(add[25]));
INBUF UI24 (.PAD(pads_), .Y(ads_));
INBUF UI25 (.PAD(preset), .Y(reset));
INBUF UI26 (.PAD(pref), .Y(ref));
INBUF UI27 (.PAD(prw_), .Y(rw_));
INBUF UI28 (.PAD(pdata[0]), .Y(data[0]));
INBUF UI29 (.PAD(pdata[1]), .Y(data[1]));
INBUF UI30 (.PAD(pdata[2]), .Y(data[2]));
INBUF UI31 (.PAD(pdata[3]), .Y(data[3]));
INBUF UI32 (.PAD(pdata[4]), .Y(data[4]));
INBUF UI33 (.PAD(pdata[5]), .Y(data[5]));
INBUF UI34 (.PAD(pdata[6]), .Y(data[6]));
INBUF UI35 (.PAD(pdata[7]), .Y(data[7]));
INBUF UI36 (.PAD(pm_io), .Y(m_io));
endmodule
```

Example 3. An Example of Test Design

```
module test_d;
reg [25:2] add; reg sysclk; reg ads_; reg reset; reg rw_; reg ref; reg [7:0] data; reg m_io;
wire ras_,cas_; wire rw_en; wire ready_; wire [11:2] mux_add; reg [4:0] count; reg mem_all;
integer i;
reg [2:0] state; parameter a0= 3'b000 , a1=3'b001 ,a2 = 3'b010 , a3=3'b011, a4= 3'b100 , a5=3'b101 ,a6 = 3'b110;
dram1 u0 (ras_,cas_,rw_en,ready_,mux_add,sysclk,add,ads_,reset,rw_,ref,dat a,m_io);
initial          begin
$gr_waves("clock%b",sysclk,"ads",ads_,"reset%b",reset,"m_io",m_io,"state%b",u0.state[1:0],"mem_all",mem_all,"rw_",rw_,"low_a
dd%h",u0.low_add[3:0],"up_add%h",u0.up_add[3:0],"io_ready_",u0.io_ready_,"hit",u0.hit,"ready_",ready_,"addup%h",u0.add[25:2
2] ,"ras%b",ras_,"rowadd%h",add[11:2],"coladd%h",add[21:12],"cas%b", cas_,"read_write%b",rw_en, "mux_sel", u0.mux_sel,
"cs", u0.cs, "ns", u0.ns, "state", state, "add", add, "ref", ref, "count", count, "refreq", u0.refreq, "ref_end", u0.ref_end);
                 end
initial          begin
#10;             sysclk = 0;     count = 0;     ref = 0;        rw_ = 1;        reset=0;
add [25:2]= 0; ads_ = 1;         mem_all = 0;   #10 m_io = 0;
#20 reset=1;     #60 reset = 0;
@ (posedge sysclk) if (!mem_all & !reset) begin
                 fork
                                  #2 ads_ = 0;   #2 m_io = 0;
                                  #2 rw_ = 0;    #3 data [7:0] = 8'b00010000;
                 join
                                  end
@ (posedge sysclk) #10 ads_ = 1;
wait (!ready_)  #2 m_io =1;
#4 ads_ =1;      #5 rw_ = 1;     #30 mem_all = 1; end
always @ (posedge sysclk)begin
if (reset) state = a0;
else
case (state)
a0: if (mem_all) state = a1; a1: begin
#2 ads_=0;
#2 add[25:22] = 4'b0000;
#2 rw_ = 0;
#2 state = a2;  end
a2 : begin
#2 ads_ = 1;
#2 state = a3;  end
a3 : if (!ready_) #2 state = a4;else #2 state = a3;
a4: begin
#2 ads_=0;
#2 rw_ = 1;
#2 state = a5;  end
a5 : begin
#2 ads_ = 1;
#2 state = a6;  end
```

Example 3. An Example of Test Design (Continued)

```
a6 : if (!ready_) begin
                #2 state = a0;
                #2 add[21:02] = add[21:02] + 1;
                end             else #2 state = a6;
default : #2 state = a0;
endcase
end
always @(posedge sysclk)
if (reset) #2 ref = 0;
else if (count == 5'h0f) #2 ref = 1;
else #2 ref = 0;
always @(ref | count==0 ) if (reset) count = 0;
else for (i= 0; i<20; i=i+1) count = #70 count + 1;
initial begin
$monitor ($time,"%b %b %b %h %h %b %b %b" ,sysclk,add[25:22],ras_,add[11:2],add[21:12],cas_,rw_en,ready_);
end
always forever #30 sysclk = ~sysclk;
endmodule
```

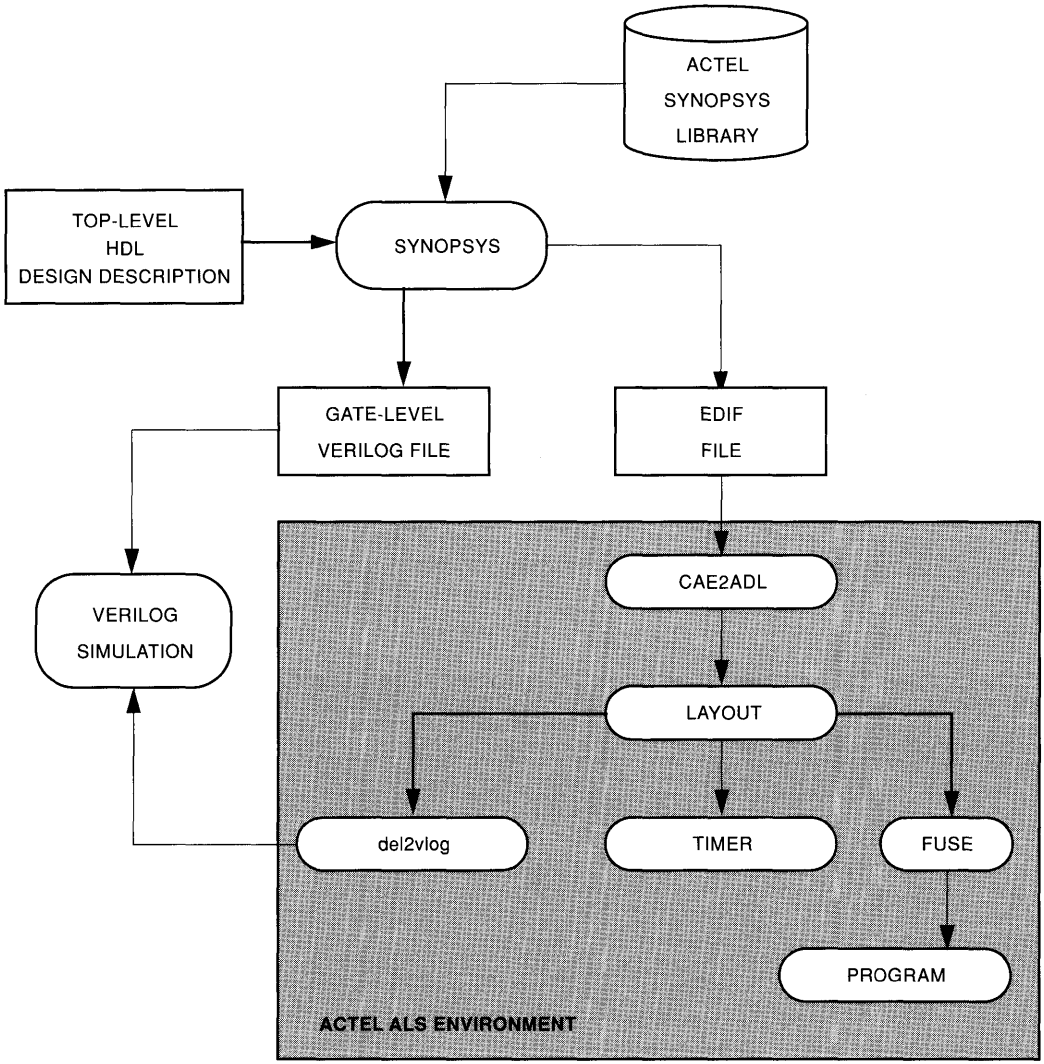


Figure 1. Typical Design Flow Using SynopsYs and Cadence

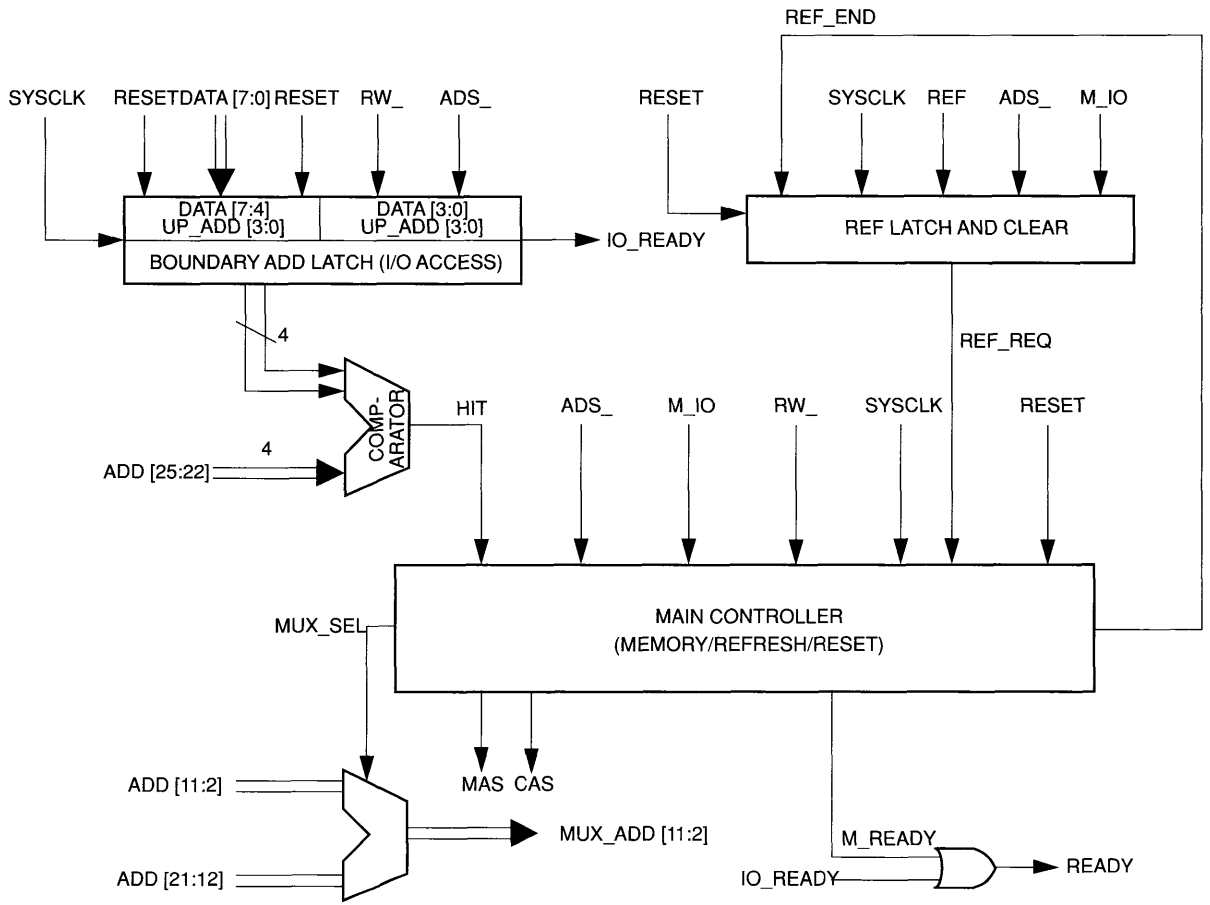


Figure 2. Block Diagram Description for a DRAM Controller

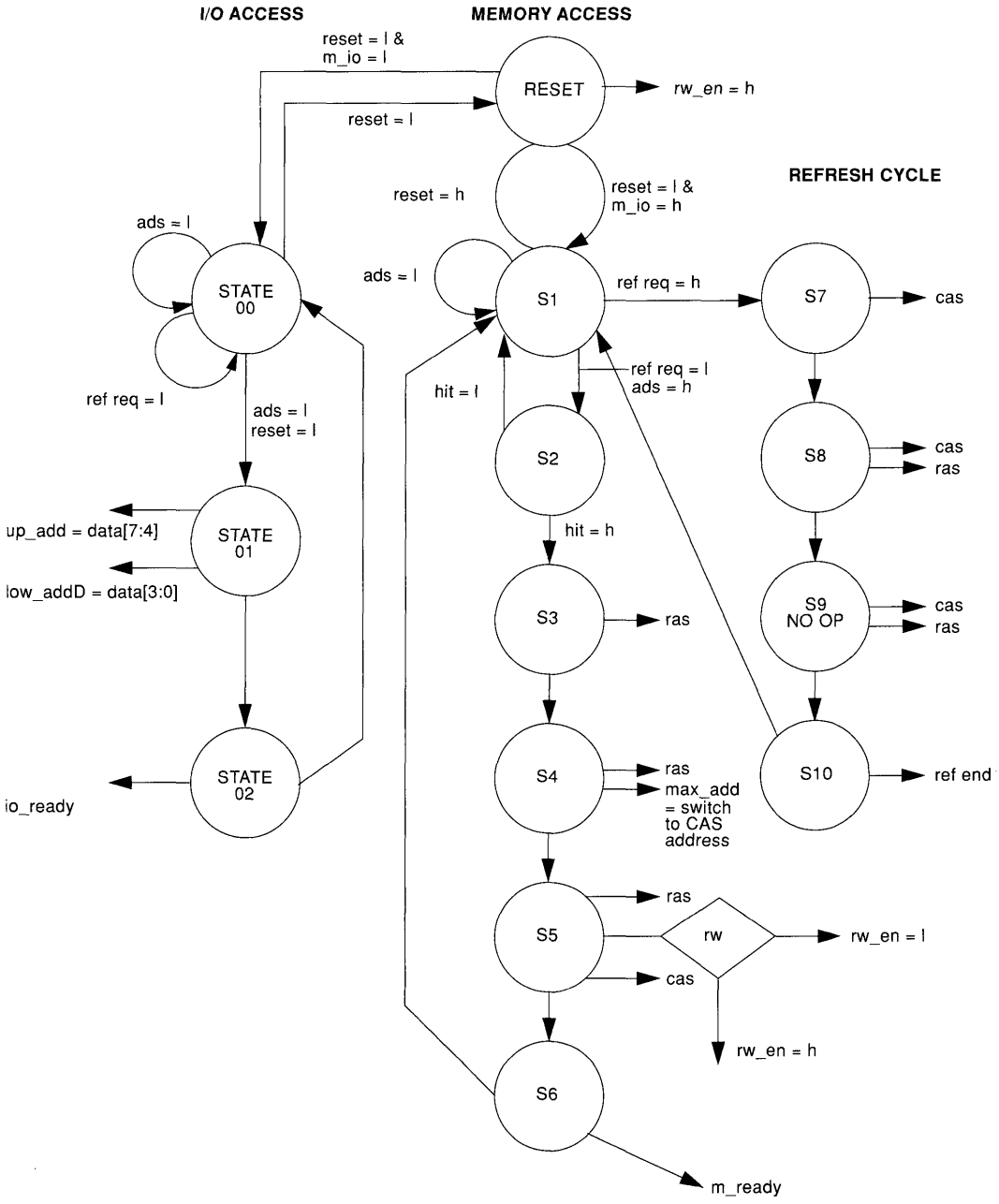


Figure 3. Memory Access

Four-Channel DMA Controller

Introduction

The ACT™ 2 DMA controller supermacro allows you to connect up to four devices to a common memory controller interface. The supermacro uses approximately 19 percent of an A1280. The schematic symbol and architecture of the DMA controller are shown in Figure 1 and Figure 2. Each channel has a 24-bit register loaded by the processor with the starting memory address.

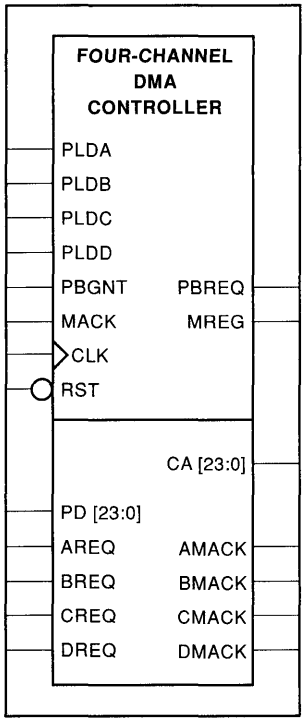


Figure 1. Schematic Symbol

Cooperation Overview

When a channel makes a request for a transfer, the controller makes a request to the processor to use the system bus. When the bus is granted, the address the channel is using (except for the two most significant bits, which are used for bank select) is loaded into a counter driving the memory controller by using a state machine. The state machine also issues a transfer request to the memory controller. The completed memory transfer is acknowledged to the requesting channel via the state machine, which also increments the address in the counter and writes it back to the requesting channel's register.

State Machine Description

The four-channel memory transfer request lines are prioritized so that only one request is recognized at a time. The control of the DMA is handled by a state machine submacro containing additional logic for prioritization. A state graph of the state machine may be seen in Figure 3.

In state S0, the state machine loops, awaiting a request. A request causes a transition to S1, which issues a processor bus request and waits for an acknowledgment. After the bus is granted, the state moves to S3 where the counter is loaded and a memory access is requested.

The state machine loops on S3 until the transfer is completed and moves to S7 to increment the counter. Going next to S5 to load the counter value back into the channel's register, the state changes to S0 if there are not further requests from the channel. If there is another request from it, the machine loops through S4, S7, and S5 following the memory transfer, counter increment, and register load sequence, as described above.

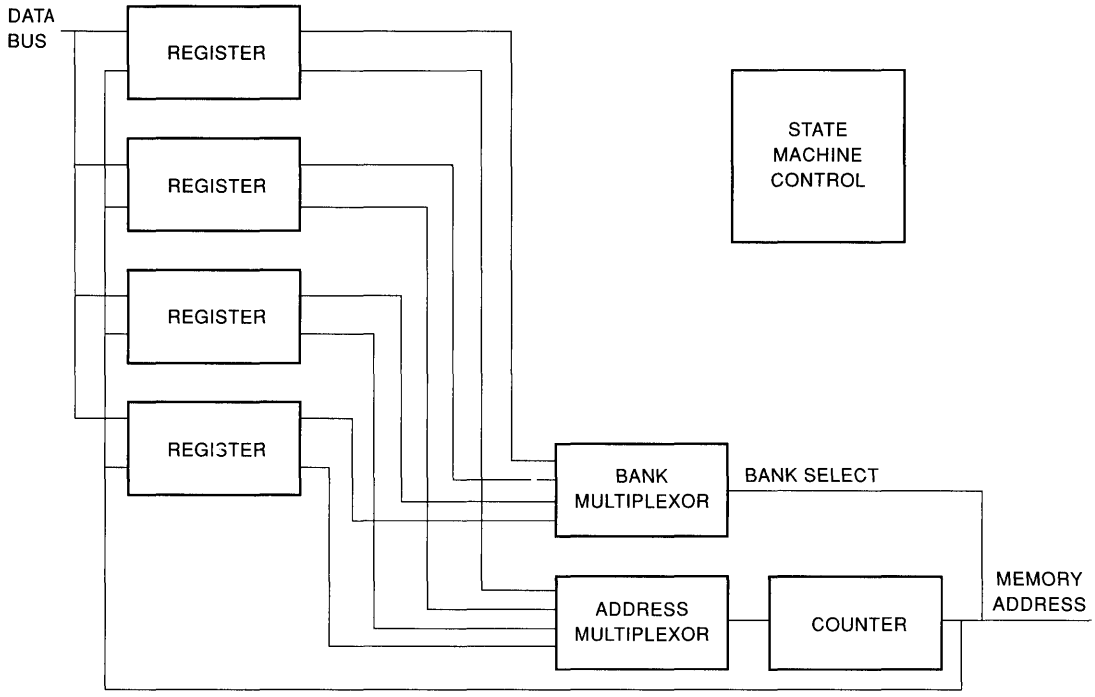


Figure 2. DMA Controller Block Design

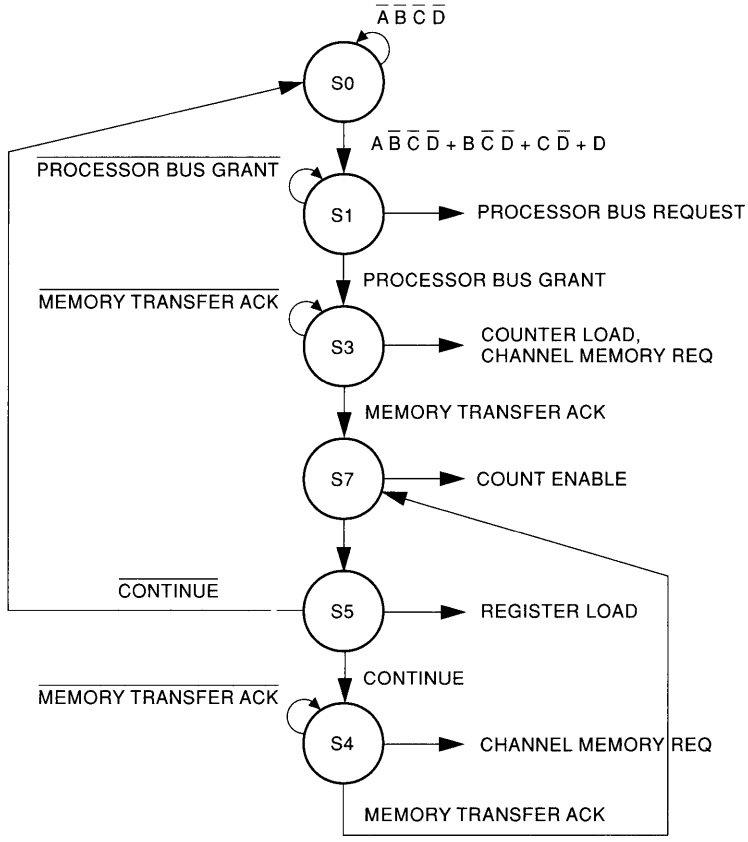


Figure 3. Four-Channel DMA Macro State Machine Control

A High-Performance Networking Interface Using Actel FPGAs

Introduction

High-speed serial data transmission is quickly becoming the most cost-effective method of connecting systems for a variety of applications. From workstation and PC connectivity to factory automation and automotives, networks have become a pervasive part of the modern day computer and communications fabric. The standard use for networks is to allow a variety of different equipment to effectively communicate but there are non-standard, proprietary applications as well, where high bandwidth communications is needed. This note will discuss the design of a proprietary high-speed network that connects distributed processing elements in a document retrieval system. It uses a single Actel 1425 field programmable gate array (FPGA) running at up to 100 MHz.

Network Architecture

Reliability and high performance were the goals in this system. It was important to have immediate access to a large amount of data, usually scanned images of documents. A 100 MHz fiber-optic token ring network was chosen to ensure high performance and reliability. Network management was key to ensuring reliable transmission and managing data flow effectively. This suggested a tree architecture with concentrators distributed throughout the tree that could manage data transmission and ensure reliable connections. The resulting network architecture is shown in Figure 1.

The top of the diagram shows the network concentrators; the document processing stations are at the bottom. Disk farms containing document archives are distributed throughout the tree. Data requested by the processing stations flow from the disks through the concentrators and arrive at the requesting station. Data transfers between disks are also possible if network traffic is determined to be better managed by transferring data between disks, thereby keeping data transfers local.

Although the network architecture is shown as a tree, the data actually travels around the network as if it were a ring going from station to station until the receiving station takes the data off the network. This is illustrated more clearly by looking at a single concentrator in the network, as shown in Figure 2. Data from the network enters at the top of the diagram and flows down to node 0. If the data is not destined for node 0, it is sent back up the link to the concentrator where it is routed to node 1. This process continues until the data arrives at the correct destination, where it is removed from the network. The header of the message is modified, showing that the data was received, so that when it arrives back at the sender the sender knows the data was received by the requesting station.

A node processor is located at the concentrator and is responsible for monitoring data flow through the local network. If a connection to a node fails because of a station malfunction or a line being removed, the node processor must detect the failure and re-route the data around the failing node. For example, if node 0 in Figure 2 experienced a network link failure, the concentrator would detect the failure and route the incoming data to node 1 instead. If node 1 also failed, the data would be routed to node 2, and so on. Thus, failing nodes do not effect the transmission of data to other nodes. This allows the system to continue operating even if some nodes are failing, being serviced, or nonexistent.

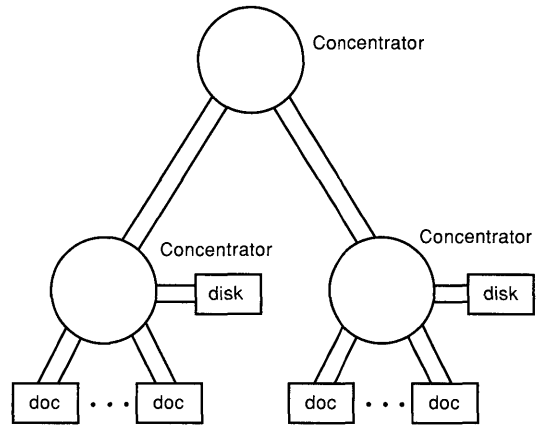


Figure 1. High-Speed Network Architecture

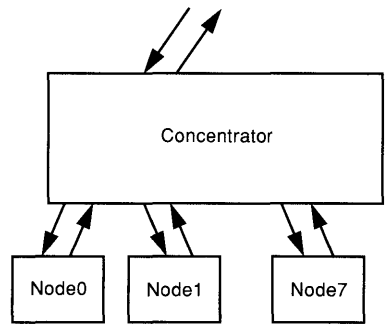


Figure 2. Network Concentrator Interface

Concentrator Architecture

The concentrator architecture is shown in Figure 3. The main components are the node processor, its associated memory, the optical data links, and the field programmable gate array (FPGA) containing the digital logic needed to connect everything. The node processor monitors data packets moving through the concentrator and detects failing links, traffic jams, or other unusual flow patterns that might require data to be transferred between disks, diagnostics, and other network management functions. It can receive data as a node in its own right, and it communicates with other concentrators regularly.

The optical data links translate the electrical signals from the concentrator to optical signals carried by the optical fiber and translate optical signals from the fiber into electrical signals used by the concentrator. The 100 MHz serial data clock is extracted by the main receiving data link and used to synchronize all other links, the node processor, and the FPGA.

The FPGA is used to connect the synchronized inputs and outputs of the data links and the node processor. The node processor can control data flow between data links and bypass failed links. The node processor can also monitor transmissions between links to accumulate statistics useful in detecting bad links, network traffic flow, and other important items.

Architecture of the Concentrator FPGA

The major tasks of the FPGA are to allow the node processor to control adding or removing links from the ring (bypassing), passing data from one active link to another, and capturing data packets for the node processor to use in gathering network statistics. The FPGA contains three major operational sections, as illustrated in Figure 4; the serial data path, the node processor interface, and the status/control section. The serial data path connects the serial data from the data links as determined by the path control register. A single bit is used with each serial data channel to determine if it is bypassed or not. The processor sets the desired bits in the path control register to determine interconnectivity.

The status/control section provides the node processor with the needed "hooks" to observe packets as they flow through the network and to read the contents of various status and control registers. A simple state machine manages the flow of data between the node processor and the serial network and synchronizes the transfer between these sections. A description of the major status and control registers follows.

Control Registers

The link control register determines which links (if any) are bypassed (one bit per serial channel).

The snoop address register determines which data links output is stored in the snoop data register for access by the node processor.

Status Registers

The link active register indicates which data links (if any) are inactive (no signal is present). This is used to determine if a link must be bypassed.

The snoop data register contains the data received from the addressed data link.

The node processor interface allows the node processor access to the interconnection control register, status registers, and the packet snoop register. It contains a bidirectional data bus, bus control logic, address decode logic, and the usual microprocessor glue.

Design of the Concentrator FPGA

The most challenging portion of the concentrator design is the serial data link section. It contains the 100 MHz data path, which is usually a speed out of the reach of most FPGAs. As shown in Figure 5, the serial data path for an individual bit consists of the bypass multiplexer, a shift register bit, and the output register. The output of the previous link can be selected as the data source or that link can be bypassed by selecting the other multiplexer input. The input/output pairs are cascaded together, with the main concentrator serial inputs and outputs appearing at the ends of the chain.

Given the register-intensive nature and high operating frequency requirement of the application, an FPGA was the natural choice. To hit the 100 MHz performance goal, a 10 ns clock to output is required on the output register, a 3 ns setup time is required on the input register, and an internal data path rate of 100 MHz is required. The Actel 1425 meets or exceeds all the requirements and has the capacity to implement the serial data path at only a fraction of a single device. The timing for the worst-case paths of the serial data section in a fully automatically placed and routed device is shown at the bottom of Figure 5. All numbers are worst-case commercial.

The status and control section of the design consists of the necessary registers and selection circuitry to allow access by the node processor. The registers are written into as determined by the address supplied by the node processor and read out in a similar manner. Of more interest is the snoop data register. It needs to be loaded from the selected data link at the 100 MHz network rate and read by the node processor before the next word is shifted in. These transfers are accomplished within the needed 100 ns by the node processor, using a block transfer capability. The FPGA ensures that the block transfer is accomplished and synchronizes the serial data stream from the network with the node processor data bus. Figure 6 shows the portion of the design where the serial data is stored by the FPGA, transferred to the snoop data register, and accessed by the node processor. A small state machine controls the transfer of data from the serial register to the snoop register and ensures that the processor doesn't miss any data. The transfer of data is stopped by the node processor after a complete block has been transferred. A new link can then be selected for snooping.

The node processor interface is the most straightforward portion of the design and is shown in Figure 7. It contains the processor interface, address decode, and control logic sections. The data bus is bidirectional and is synchronized with respect to the processor

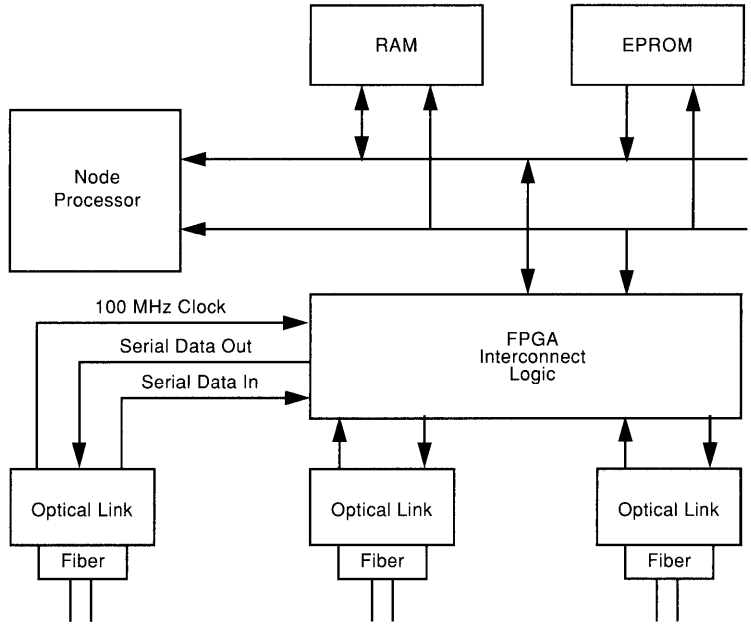


Figure 3. Network Concentrator Architecture

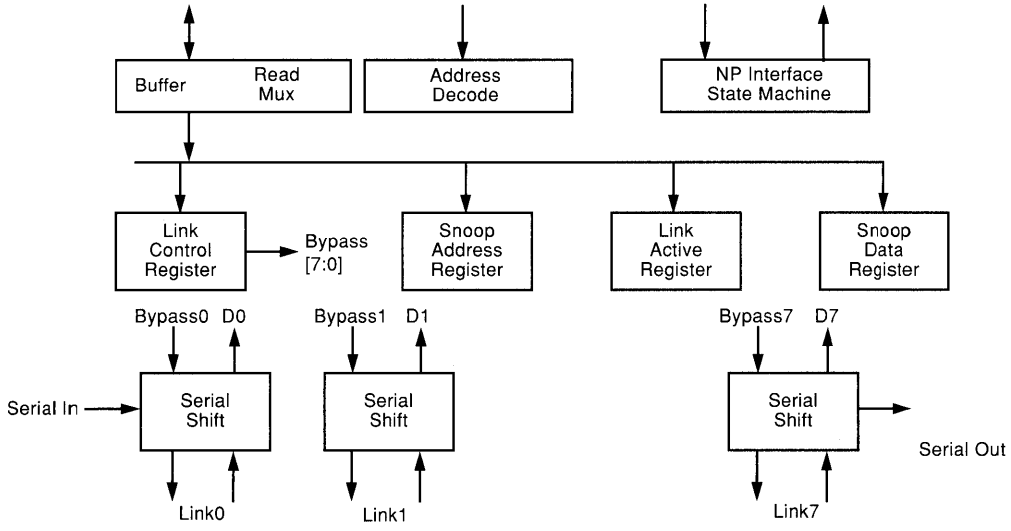


Figure 4. Concentrator FPGA Block Diagram

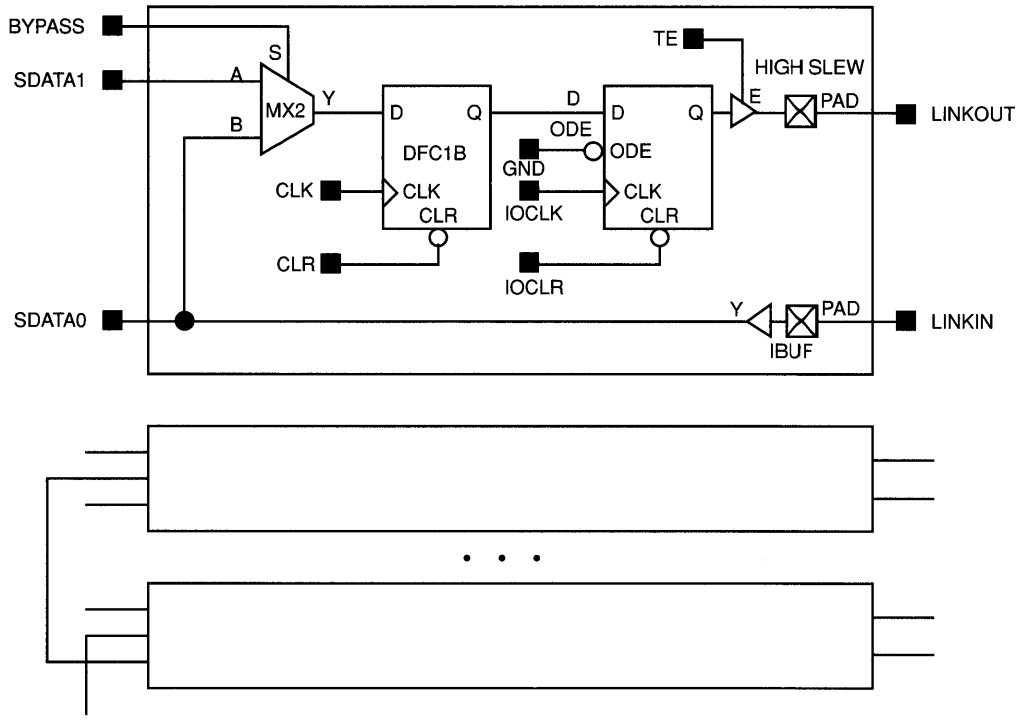


Figure 5. Concentrator Schematic and Timing Diagram

clock. The control logic determines which register is read from or written to according to the address inputs. Additionally, the interface to the snoop register is controlled by this section and manages the data transfer between the processor and the FPGA during burst mode transfers. When the processor has captured as much of the packet as desired, it simply stops reading data and the snoop register is allowed to be overwritten by the snoop shift register until another transfer is requested.

Conclusion

This application note discussed the detailed design of a high-speed data concentrator using the Actel A1425. Applications like this illustrate the types of designs this new generation of FPGAs can address, applications that were out of the reach of previous FPGA devices.

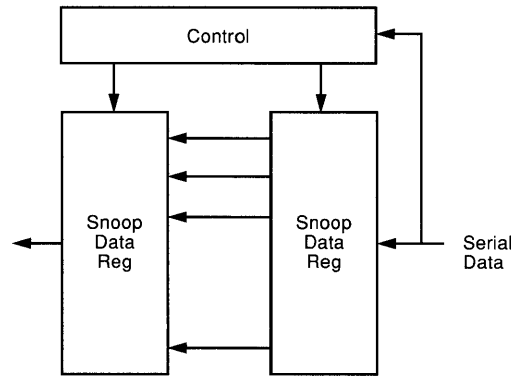


Figure 6. Serial Data Storage

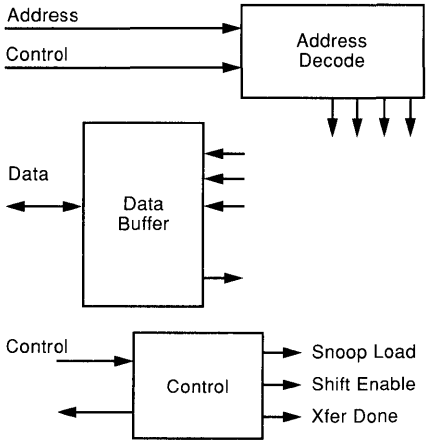


Figure 7. Node Processor Interface

A High-Performance Synchronous Memory Interface Using Actel FPGAs

Introduction

Synchronous memory interfaces are used in a broad number of applications that require high bandwidth access to large amounts of data. Graphics subsystems in particular can benefit from the use of synchronous memory architectures based on block transfers. What follows describes the design of the interface using a single Actel 1425 field programmable gate array (FPGA) running at up to 100 MHz.

Object Movement Subsystem

The graphics subsystem we discuss is the subsystem responsible for the rapid movement of a large number of objects within a window as illustrated in Figure 1. Any object may be translated by an (X,Y) coordinate to another location in the window. Each object is represented by an 18-bit word containing two tag bits, an 8-bit object number, and an 8-bit color value (objects may have the same color). The tag bits define the beginning and end of an object, as well as the existence of the object. This allows objects to be transferred in blocks even if they have irregular outlines. For example, the object in Figure 2 exists on 8 lines (only the first is shown) with the first line containing a start word, two object words, a non-object word, one more object word, and an end word.

The entire line can be moved in a contiguous transfer by beginning somewhere left of the start word, reading to the end word and writing the line out to final memory. During the line write the object will be copied beginning with the start word and ending with the end word. Only words with the tag bits set to an object will be written into. Notice that objects need not contain any words on a line if only a start and stop word are present.

Graphics Subsystem Architecture

Figure 3 depicts the hardware implementation of the subsystem. The processor loads the starting image memory independently from the FPGA by taking the FPGA off the bus during the load. After the image is loaded, the processor loads the FPGA with a series of transfer requests and grants the FPGA access to the memory. After the FPGA is done processing the transfer requests by copying the starting image memory, to the final image memory it interrupts the processor to inform it that the operations are completed. The processor can now access the final image memory and determine the results of the transfers. Notice that two separate physical memories are not needed to hold the starting and final images; the upper address bit can be used to select between starting and final images.

Architecture of the Subsystem FPGA

Figure 4 shows the detailed block diagram of the FPGA portion of the design. The blocks making up the design are the object memory interface, the object stack, the processor interface, and the master control section.

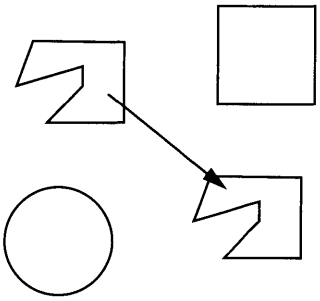


Figure 1. Graphics Subsystem Window

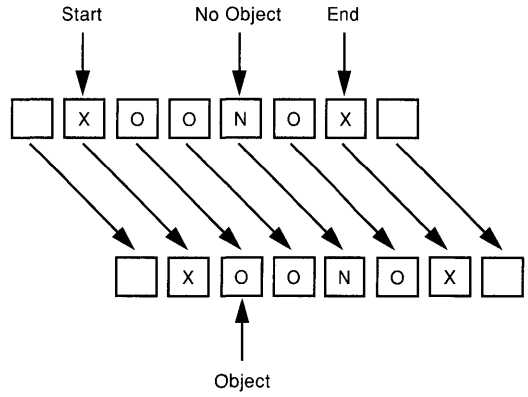


Figure 2. Object Translation

A major portion of the design is the memory interface that controls access to the object memory. The memory is organized as four banks, each of 16K words. This interleaved architecture allows four-word bursts at the faster output enable rate of the memory instead of the slower address access rate. This improves performance considerably in block transfer designs, but does require more memory devices (four times as many for four-way interleaved) than a noninterleaved scheme. The address for an object is thus divided into a 16K word address (14 address bits) and a 2-bit address within the block. Addresses are consecutive within the window on a line basis and wrap around after 256 words. Because both the starting and final image memories are contained in a single physical memory, the window size is 128 by 256. The more detailed diagram in Figure 5 shows the design of the interface.

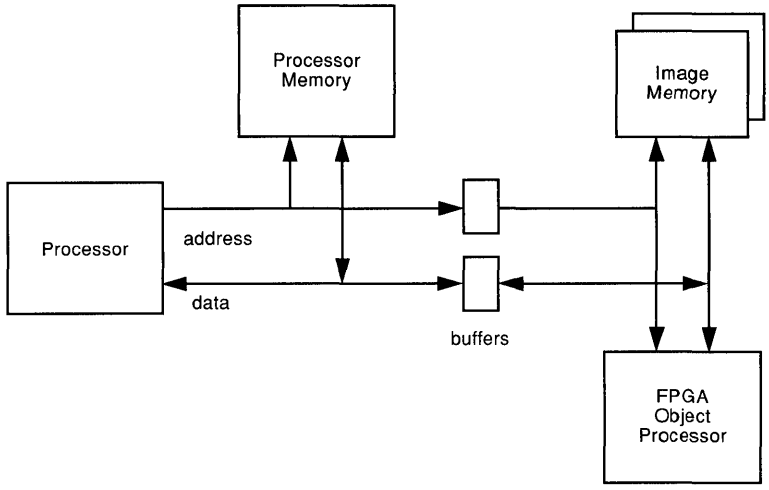


Figure 3. Graphics Subsystem Hardware Implementation

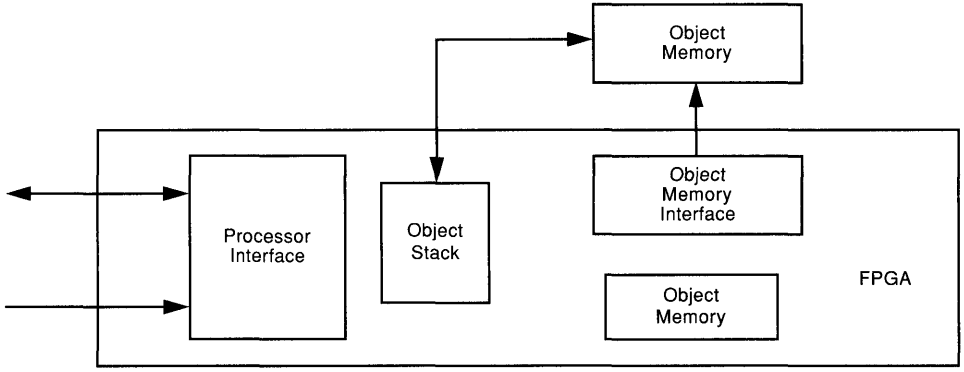


Figure 4. FPGA Block Diagram

The upper address bits to the memory are selected from the source and destination line counters as the object is moved from one location to another. A 6-bit address counter is used to count words on a line. A small state machine provides the chip selects and write-enables to the memory depending on the type of transfer (read or write) as well as the tag bits associated with the word. Address outputs are synchronous and are clocked on the 50 MHz system clock. Notice that the WE and CS outputs are clocked on the 100 MHz I/O clock to generate the needed 2x cycle control signals.

The 4-word, 18-bit object stack, shown in Figure 6, holds the data during read and write bursts. Data is shifted into the stack on a read and then shifted out on a write. Tag information is used by the address generation portion of the interface to ensure that only valid object words are written into memory. Notice that the stack is organized as a LIFO (last in, first out) stack. This allows the first and last access of the object line to be unaligned with the four word interleave of the memory. For example, if the first access of an object contains only two words, the two words can be shifted in and then shifted out without needing to randomly access the

words in the stack. This is a more efficient implementation in an FPGA architecture. Because of the LIFO nature of the stack, the data written to memory will be in the reverse order, and the write-enable signals are reversed to compensate.

The processor interface is shown in Figure 7. It contains a set of command registers that hold the move coordinates of a series of transfers. These registers are loaded by the processor on a bus separate from that of the image data so that object transfers can be processed concurrently with object translation operations. Status information to the processor is also available on this bus and is used by the processor to monitor progress on object transfers. The interrupt control block will inform the processor when the end transfer command is processed so that final image data can be accessed.

The master control section in Figure 8 contains the main state machine, which processes transfer commands, initializes the memory interface prior to a line transfer, and manages each object transfer. It holds starting and ending address pointers and determines when the last line of an object has been transferred by decrementing the object length counter until it reaches zero. The master control section is responsible for transferring objects, while the memory interface is responsible for transferring lines. The state machine is implemented in a high-level language using the "one-hot" methodology; it operates at 50 MHz.

Conclusion

This application note described the detailed design of a high-speed graphics memory interface using the Actel A1425. The interface operates at 100 MHz with a burst data transfer rate of 900 Mb/s (18-bits every 20 ns). The internal state machine controlling object transfers runs easily at 50 MHz and contains a variety of counters, registers, and random logic. The data stack stores a burst of four 18-bit words between transfers and can be clocked at over 100 MHz. Applications like this illustrate the types of designs this new generation of FPGAs can address, applications that were out of the reach of previous FPGA devices.

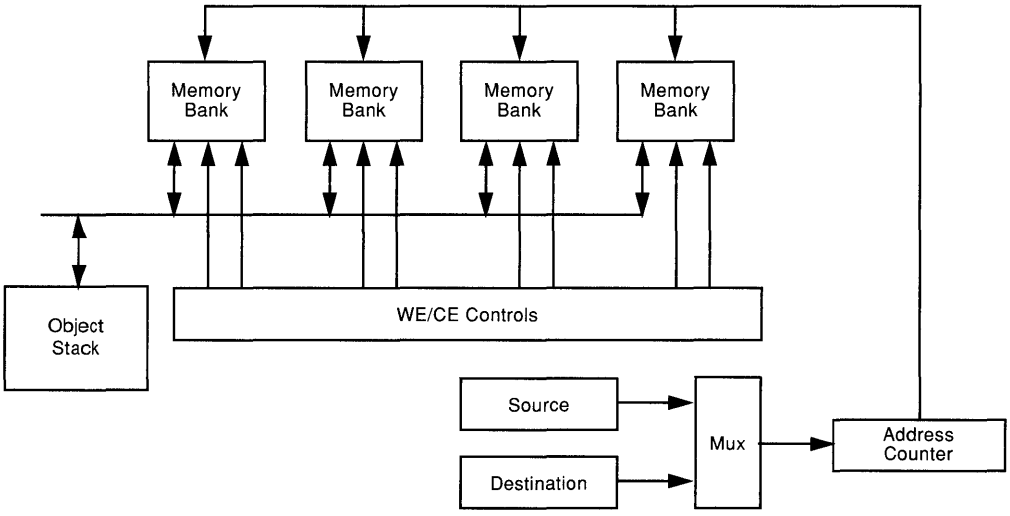


Figure 5. Interface Design

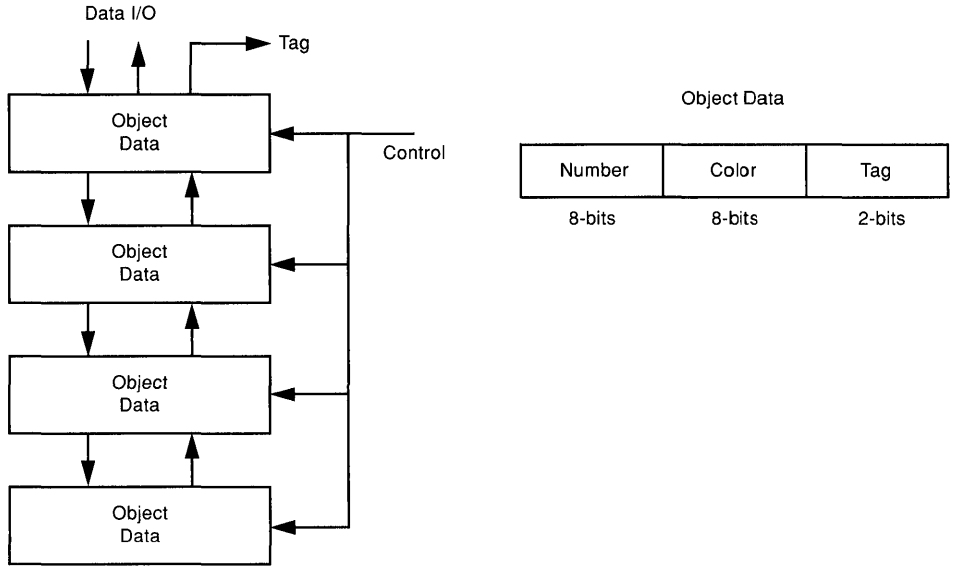


Figure 6. Object Stack

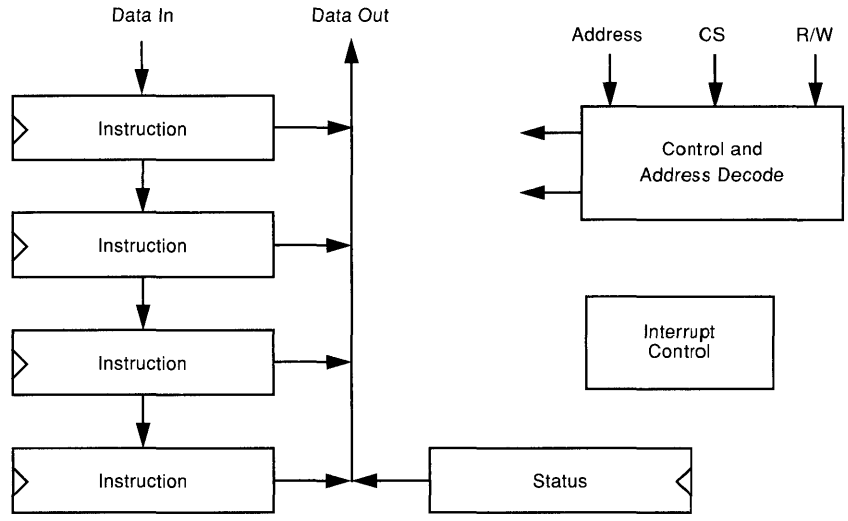


Figure 7. The Processor Interface

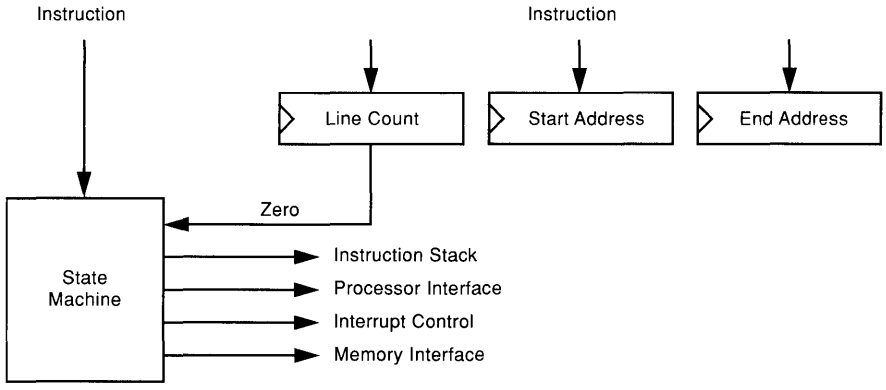


Figure 8. Section of the Master Control

Synchronous Dividers in Actel FPGAs

Application Note

Synchronous dividers are useful in numerous applications, such as prescalers, timing generators, and multi-phase clocks. Although ripple dividers use less logic, glitches and potential race conditions make them hazardous in all but the simplest applications. Synchronous dividers offer a better solution for reliable operation.

Figure 1 shows divide-by-two to divide-by-ten synchronous dividers using combinable Actel logic modules. Using the Actel ACT 2 and ACT 3 architectures, the combinatorial gates will be absorbed with the following flip-flop. The flip-flops used do not have reset pins as dividers and are generally used as astable devices. The OR-AND gates are used for some of the dividers to avoid nonconvergent illegal states. To initialize the dividers for simulation, the output can be held high and clocked once for each flip-flop, then released. For example, with the Viewsim™ simulator, the divide-by-three synchronous divider (with output node F3) would be initialized with the following command file sequence:

```
h F3
cycle 2
r F3
cycle 5
```

The resulting waveforms for each of the dividers are shown in Figure 2. Note that the duty cycle is 50 percent for all even dividers and as close to this as possible for the others.

9

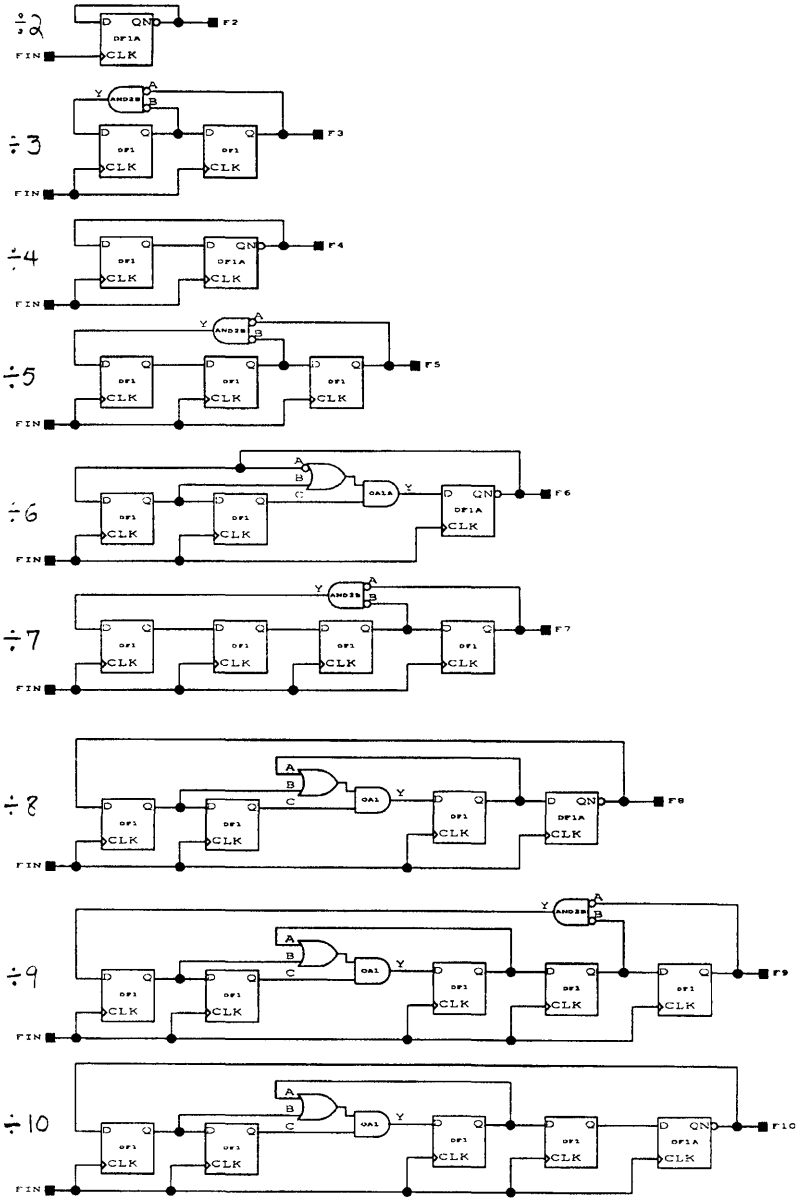


Figure 1. Actel Implementation for Synchronous Dividers

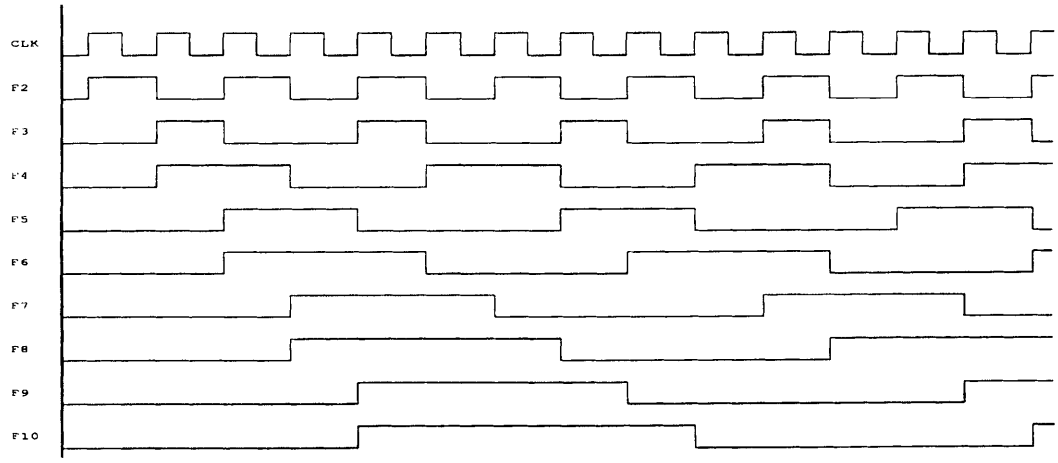


Figure 2. Synchronous Divider Waveforms

A Stepper Motor Controller in an Actel FPGA

Application Note

Introduction

Stepper motors are electromechanical devices that provide accurate incremental rotation. Printer paper feeders, floppy drives, and robotic manipulators are popular stepper motor applications. The most common stepper motor uses four windings for a four-phase operation. Rotation is effected by actuating the phases in a specific sequence. Field programmable gate arrays (FPGAs) are ideal for integrating the control logic for these motors with other system control logic to minimize device count and board size.

Four-Phase Motor Circuit

A typical four-phase motor driving circuit is shown in Figure 1 using an FPGA to generate the sequence logic. The four windings have a common connection to the motor supply voltage (V_S) which typically ranges from 5 to 30 Volts. Each of the four phases is driven by a high power NPN transistor, since the FPGA cannot drive the motor directly. Each motor phase current may range from 100 mA to as much as 10 A. The transistor selection depends on drive current, power dissipation, and gain. The series resistors should be selected to limit the FPGA current to 8 mA per output. Power MOSFET devices can also be used to drive the stepper motor.

Within the Actel FPGA, four inputs are required to fully control the stepper motor. The clock (CLK) input synchronizes the logic and determines the speed of rotation. The motor advances one step per clock period; the angle of rotation of the shaft will depend on the particular motor (the angle ranges from 1.8 to 90 degrees per step). To determine the clock period, consider that the stepper motor torque increases as frequency decreases. The direction (DIR) control input changes the sequence at the outputs (PH1 to PH4) to reverse the motor direction. The enable input (EN) determines whether the motor is rotating or holding. The active low reset input (RST) initializes the circuit to ensure that the correct starting sequence is provided to the outputs.

The basic control sequence of a four-phase motor is achieved by activating one phase at a time as shown in Figure 2. Figure 3 shows an enhanced sequence that uses an overlap technique with two phases active at any one time. The enhanced sequence provides increased torque but requires twice the current.

Boolean equations using PALASM®2 syntax for the basic control sequence are shown in Figure 4. The phase equations (PH1 to PH4) are written with a colon and equal sign (:=) to indicate a registered implementation of the combinatorial equation. Each phase equation is either enabled (EN), indicating that the motor is rotating, or disabled (/EN), indicating that the current active

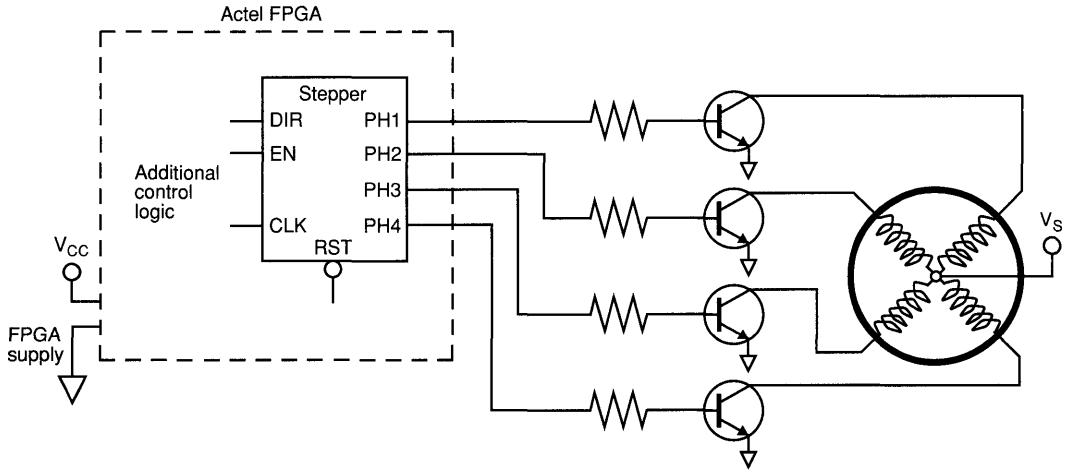


Figure 1. Four-Phase Stepper Motor Driving Circuit

Step Sequence ->					
PH1	1	0	0	0	1
PH2	0	1	0	0	0
PH3	0	0	1	0	0
PH4	0	0	0	1	0

Figure 2. Basic Stepper Motor Sequence

Step sequence ->					
PH1	1	1	0	0	1
PH2	0	0	1	1	0
PH3	1	0	0	1	1
PH4	0	1	1	0	0

Figure 3. Enhanced Stepper Motor Sequence

```
CHIP step1 ACT
clk en dir rst ph1 ph2 ph3 ph4

EQUATIONS

ph1 := /dir * en * (/ph1 * /ph2 * /ph3 * ph4)
     + dir * en * (/ph1 * ph2 * /ph3 * /ph4)
     + /en * ph1

ph2 := /dir * en * ( ph1 * /ph2 * /ph3 * /ph4)
     + dir * en * (/ph1 * /ph2 * ph3 * /ph4)
     + /en * ph2

ph3 := /dir * en * (/ph1 * ph2 * /ph3 * /ph4)
     + dir * en * (/ph1 * /ph2 * /ph3 * ph4)
     + /en * ph3

ph4 := /dir * en * (/ph1 * /ph2 * ph3 * /ph4)
     + dir * en * ( ph1 * /ph2 * /ph3 * /ph4)
     + /en * ph4

ph1.setf = /rst
ph2.rstf = /rst
ph3.rstf = /rst
ph4.rstf = /rst
```

Figure 4. Boolean Equations for Basic Stepper Motor Sequence

phase remains on and the motor is locked. The value of the direction input (DIR) determines which product term is used to sequence clock wise or counterclockwise. The asynchronous equations (for example, ph1.setf = /rst) initialize the circuit. The Boolean equations for the enhanced motor stepper sequence shown in Figure 5 are simpler than the basic sequence. By inspection of the sequence from Figure 3, phases one and three are mere inversions of phases two and four. Therefore, two phase equations can be drastically reduced (ph1 = /ph2, ph3 = /ph4). Also, the next sequence for each phase is only dependent on one other phase and not all four, thereby reducing the number of terms required for the remaining phase two and four equations. Note that inverting phases one and three provides the correct initial sequence values.

```
CHIP step2 ACT
clk en dir rst ph1 ph2 ph3 ph4
EQUATIONS
ph2 := en * /dir * ph4 + en * dir * ph3 + /en * ph2
ph4 := en * /dir * ph1 + en * dir * ph2 + /en * ph4
ph2.rstf = /rst
ph4.rstf = /rst
ph1 = /ph2
ph3 = /ph4
```

Figure 5. Boolean Equations for Enhanced Stepper Motor Sequence

The Boolean files were synthesized and optimized for the Actel ACT™ 2 family. Modules utilized for the basic and enhanced sequences were 14 and 6 modules respectively. Figures 6 and 7 are the schematic representations of the basic and enhanced sequencing circuits respectively. Note that these schematics do not show the output I/O macros (OUTBUF) that are needed to connect to the external circuit (that is, outside the FPGA).

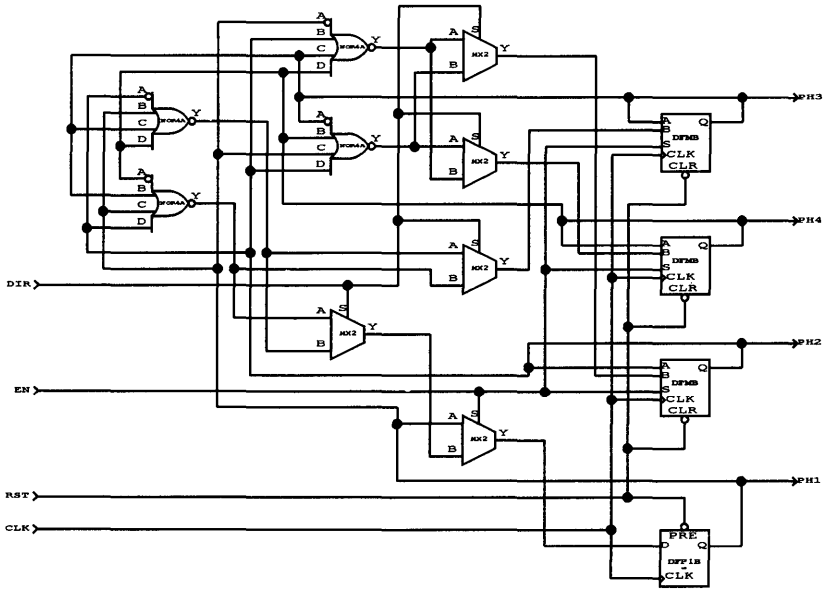


Figure 6. Basic Stepper Motor Sequence Schematic

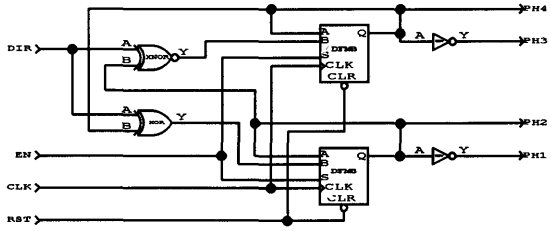


Figure 7. Enhanced Stepper Motor Sequence Schematic

The timing diagrams in Figures 8 and 9 show the operation of the two stepper motor sequences. Reset, enable, and direction inputs are exercised for a clear understanding of full circuit operation. Additional logic can be added to control the enable and direction inputs. For example, a pre-loadable down counter with a zero detect output connected to the stepper motor enable input can be used to rotate the motor a predetermined number of steps.

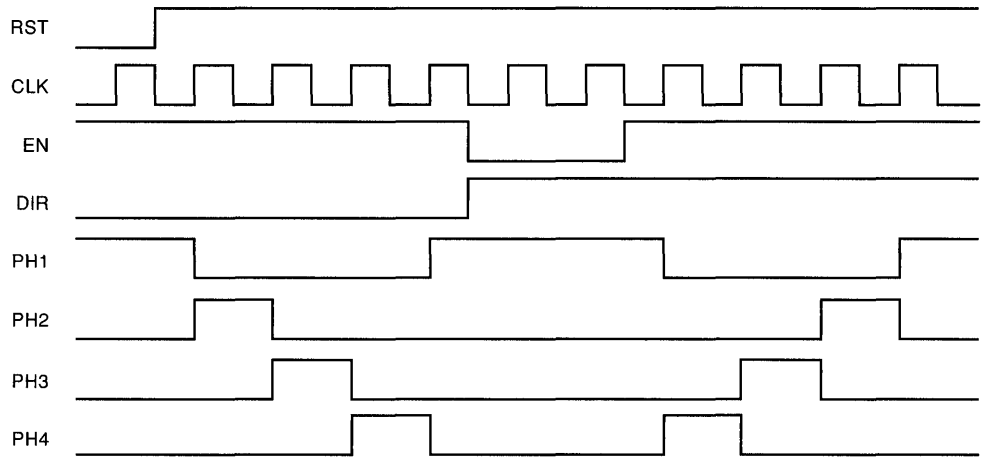


Figure 8. Basic Stepper Motor Sequence Timing Diagram

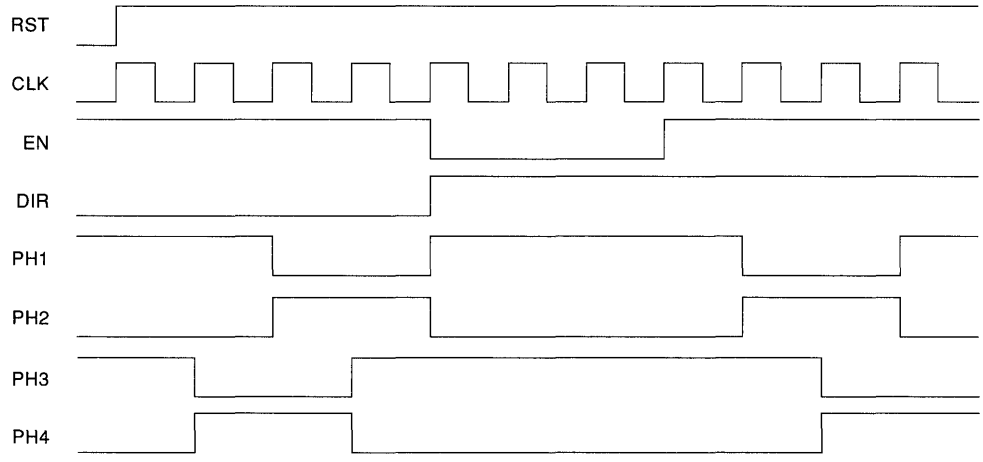


Figure 9. Enhanced Stepper Motor Sequence Timing Diagram

A Pulse Stretching Circuit for Actel FPGAs

Application Note

The ability to stretch a pulse is often useful in applications requiring specific timing. For example, synchronizing external short events to a processor may require pulse stretching to ensure signal recognition. One method to accomplish this function is to use a "one-shot" RC delay network. However, this technique yields wide variations in pulse accuracy due to its dependence on resistor and capacitor tolerances and temperature sensitivity. Another method is to put a large number of inverters in series to create a delay. This technique is most commonly used to meet the setup or hold time requirements of a critical circuit. This procedure is also not recommended due to the 30 percent variation in device processing between best- and worst-case specifications.

For reliable operation, a digital synchronous pulse stretching design is ideally suited for accurate timing and close tracking with the system clock. The block diagram in Figure 1 shows the circuit inputs and outputs. A downcounter provides a variable output pulse width (OUT) controlled by the binary preset inputs (D0, D1, D2, D3). The output is triggered by an input pulse (IN). The reset signal (RST) initializes the circuit.

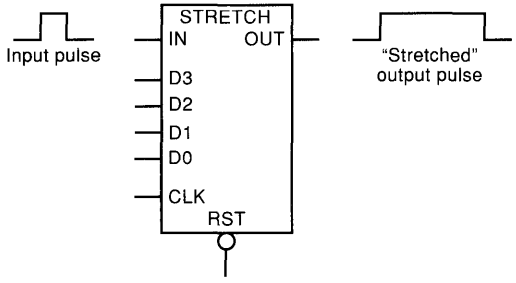


Figure 1. Pulse Stretcher Block Diagram

The pulse stretcher circuit is described in Boolean format using PALASM2 syntax as shown in Figure 2. There are four register equations in the design. The left hand side defines the logic preceding the D input of a flip-flop (:= indicates a register). Each of these equations has two components. The first component defines loading of the counter (preset inputs D0 through D3 ANDed with IN). The second component of these equations specifies the decrement logic for each bit when there is no input pulse present and the count is not zero (that is, ANDed with the inversion of IN and the inversion of ZERO). The zero flag equation is merely the combinatorial decoding of all registers equal to zero. The output equation is asserted whenever the count is not zero or the input pulse is present.

```
CHIP stretch act
clk rst in d0 d1 d2 d3 out
EQUATIONS
q0 := d0*in + (/q0*/zero)*/in
q1 := d1*in + (/q1*/q0 + q1*q0)*/zero*/in
q2 := d2*in + (/q2*/q1*/q0 + q2*q1 + q2*q0)*/zero*/in
q3 := d3*in + (/q3*/q2*/q1*/q0 + q3*q2 + q3*q1 + q3*q0)*/zero*/in
zero = (/q3*/q2*/q1*/q0)
out = /zero + in
q0.RSTF = /rst
q1.RSTF = /rst
q2.RSTF = /rst
q3.RSTF = /rst
```

Figure 2. Pulse Stretcher Boolean Description

The Boolean description of the circuit was optimized for an ACT™ 2 device. The schematic in Figure 3 was generated from the output netlist for a visual representation of the actual micro implementation. Note the use of the DFMB flip-flops, which require a single sequential module in ACT 2 (or ACT 3) and take advantage of a built-in multiplexer (inputs A, B, and S) to reduce logic. The complete circuit is shown implemented in 14 logic modules (the AX1B macros require two modules).

The output pulse is up to one clock cycle longer than preset inputs because it is ORed with the IN signal. If the OR gate is removed from the circuit, the output pulse will have a width exactly equal to n clock periods where n is the value of the data inputs. The input signal may only be active for one clock edge. If it is active for a longer period, the output will remain active for this duration in addition to the selected value. To have the output reflect only the selected delay if the input is longer than one clock period, change the output equation as follows: OUT = /ZERO * /IN. The timing diagram in Figure 4 illustrates the output pulse with a selected "stretch" value of four.

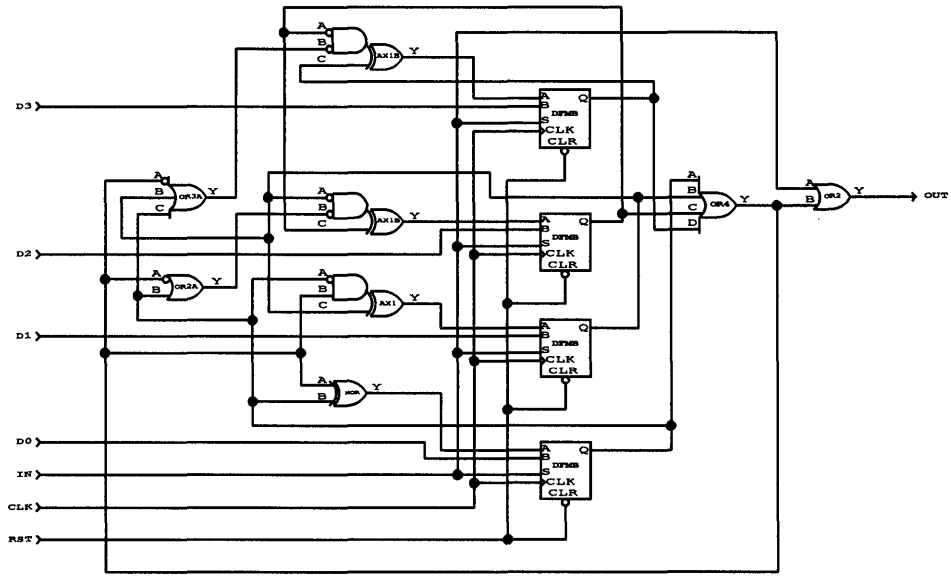


Figure 3. Pulse Stretcher Schematic

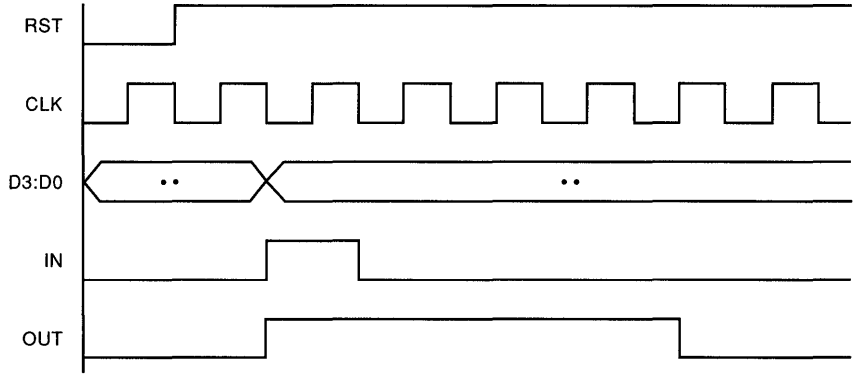


Figure 4. Timing Diagram for Pulse Stretcher

Using FPGAs for Digital PLL Applications

In addition to purely digital applications, many designs use Field Programmable Gate Arrays (FPGAs) for DSP. We'll examine one such application, digital PLLs, to show various ways of implementing PLL designs using FPGAs.

Pulse Steal PLL

In telecommunications applications, it is often desirable to generate a digital signal that is locked to an incoming signal and is some multiple of its frequency. A drawing of a pulse steal PLL, which is a simple way to generate such a signal, is shown in Figure 1. Note that the design contains an ordinary oscillator but no VCO. Except for the crystal, the entire design will operate in an FPGA.

Note the frequency relationship that holds at points A and B in the figure, where

$$OSC/(K*M) = Input/N = Comparison\ Frequency \quad (1)$$

The technique is based on selecting a reference oscillator frequency slightly higher than OSC. This frequency (OSC+) should be chosen so that

$$1/Comparison\ Freq. - (K*M)/(OSC+) = .5 * (1/OSC) \quad (2)$$

The right side of equation 2 equals one-half the period of the reference oscillator.

The reference oscillator frequency delta will cause point B (the detector flip-flop D input) to begin to precede point A (the detector flip-flop clock input) by half a period. When the edge of the D input is sufficient, the detector will clock true and begin a pulse train through the two deglitching flip-flops. The output of the second of these clears all three flip-flops and steals a pulse by disabling the divide by K output. Stealing the pulse puts point B behind A until the reference oscillator delta can move it ahead by one period, repeating the cycle. Points A and B are always within one-half a cycle of each other.

The circuit allows the frequency of the output signal to be selected simply by adjusting the values of the dividers K and M. The lock range of the loop is given by the following:

$$Lock\ Range = \pm (OSC+/osc)/Input \quad (3)$$

Jitter-Bounded Digital PLL

Another technique[1] for generating a wide variety of synchronized clock frequencies with low jitter employs an accumulator Digital Controlled Oscillator (DCO) and phase and frequency comparators. The system, shown in Figure 2, can lock to any division of a reference frequency (F ref.) as selected by the data loaded into frequency divider counters.

The Successive Approximation Register (SAR) and its controller serve as a low-pass filter supplying the DCO with frequency and phase correction data. Among the three inputs to the SAR controller is the F ref. divided by a factor Q to form F q.

The other two inputs come from the phase and frequency (zero) comparators. The frequency comparator output is the DCO frequency divided by P to form F p. When the system is in lock the following equation is true:

$$F\ dco = (P/Q) * F\ ref. \quad (4)$$

The heart of the system is the accumulator DCO, which determines the ability to lock to a frequency and the amount of jitter allowed. The DCO consists of a four-bit accumulator whose input is fed by the SAR. The DCO input value is determined from the phase and frequency comparison feedback loops. The most significant bit of the accumulator output is the DCO output signal. It is generated by successively adding the SAR value to itself at the high-frequency system clock rate. The frequency comparator uses the value of P to divide the DCO frequency. If the frequency is out of lock during a period of F ref., the comparator asserts greater-than-zero or less-than-zero to the SAR controller to modify the value of the register. If the P counter output is zero, the DCO has the correct frequency.

The DCO latch acts as a phase register indicating the phase of the DCO with respect to F ref. The DCO phase is calculated by the N most significant accumulator output bits. When the DCO is out of phase, the jitter, or phase difference, is detected by the phase comparator and accumulates with time until it equals one period. The feedback loops then cause the SAR register controller to load a correcting value into the register or to clear the accumulator with a synchronizing pulse.

The Jitter-Bounded DPLL may be implemented entirely on an ACT™ FPGA. The resource requirements vary with the relationships of the system input and output frequencies, but for any F ref., system clock, and desired output frequency, the design is easily accommodated on an ACT FPGA.

References

1. S. Walters and T. Troudet, "Digital Phase-Locked Loop with Jitter Bounded," *IEEE Transactions on Circuits and Systems*, Vol 36, No. 7, July 1989.

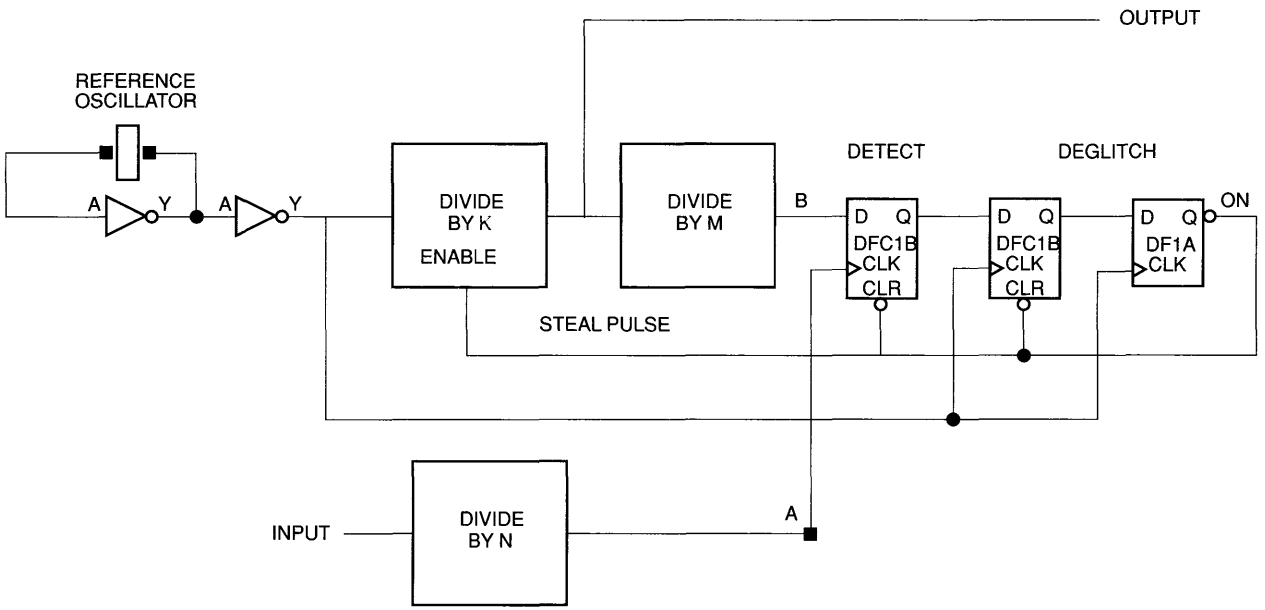


Figure 1. Pulse Steal PLL

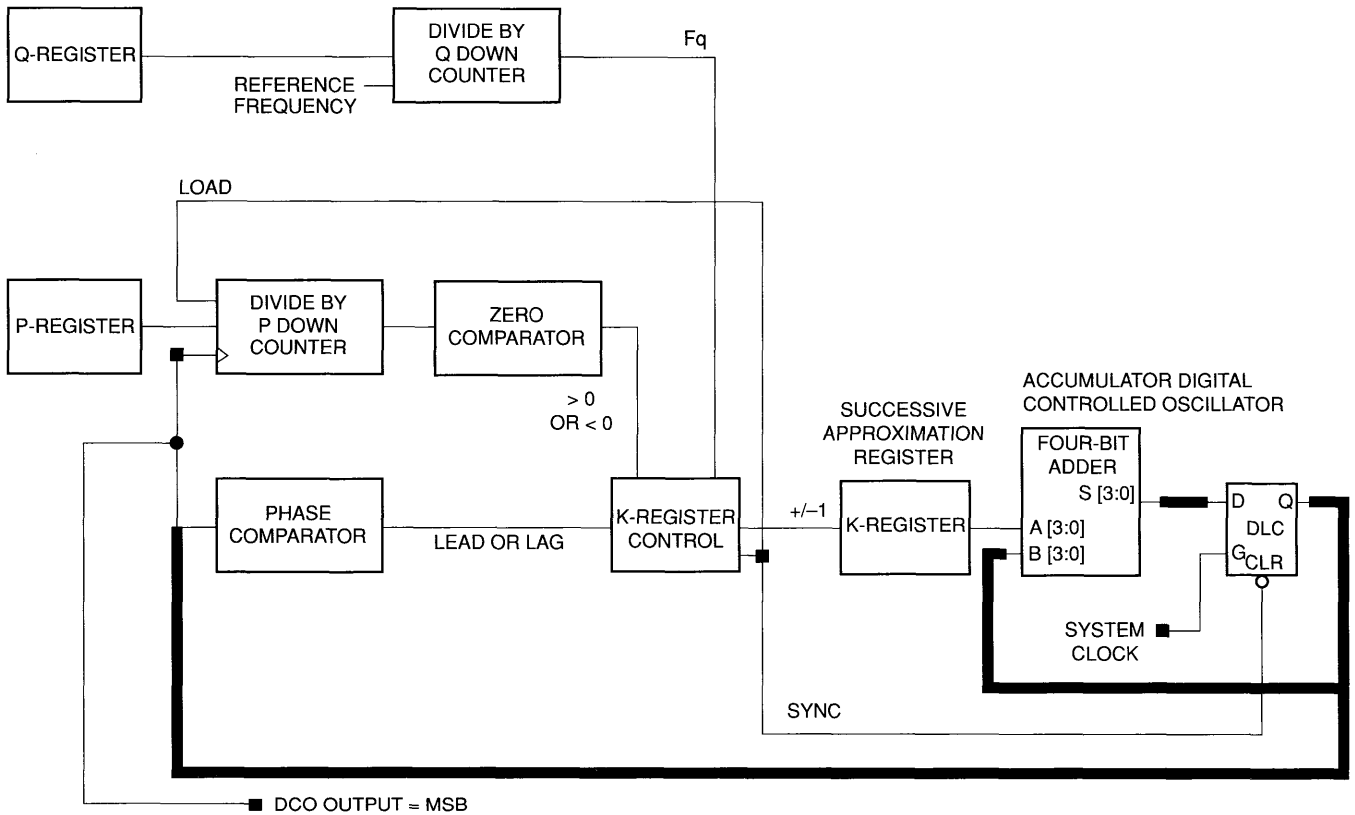
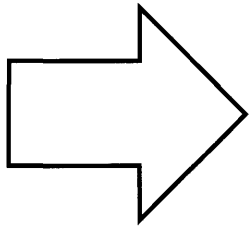


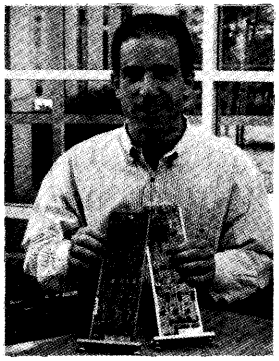
Figure 2. Jitter-Bounded Digital PLL

Customer Case Histories



CUSTOMER CASE HISTORY

3Com Selects Actel FPGAs for LAN Logic Integration, Gate Array Migration Path



David Kranzler
Hardware Design Engineer

3Com Corporation, an industry leader in local area network systems, faced several challenges when designing the TP module for its popular MultiConnect Modular Multiport Repeater platform. The MultiConnect TP Module was needed to enable Ethernet network users to communicate over previously installed telephone wiring using the IEEE 10BASE-T communications protocol standard. When installed within 3Com's MultiConnect Repeater, the TP Module would facilitate wiring hub connections for operation of 10-Mbps standard Ethernet on voice- and data-grade dual twisted pair wiring.

To design the TP Module efficiently for market use, 3Com's engineering department had to maximize the number of 10BASE-T ports that could be integrated onto a standard-sized 3Com module board, 15 of which could be accommodated by the MultiConnect Repeater chassis. The solution to this logic integration challenge was found in Actel's ACT 1 family of field programmable gate arrays (FPGAs) which ultimately helped lead to the module's design success.

The Design Task: A Need for Logic Integration and Migration Path

The design task fell onto the shoulders of David Kranzler, a hardware design engineer within 3Com's Network Adapter Division. Kranzler recalls that three factors affected the design of the product – high system density, low production cost and the need for a field-programmable solution early on in the design cycle.

"We faced various obstacles up front in designing the MultiConnect TP Module," stated Kranzler. The main problem was how to maximize the number of 10BASE-T connections on a four layer board that would fit into the MultiConnect chassis. System density played a critical role as 3Com aimed to maximize the number of ports on the module in order to optimize the number of users for the repeater. Kranzler also pointed out that since the 10BASE-T was quickly becoming a standard, the MultiConnect Module had to be designed to fully accommodate the emerging spec. Thus, 3Com needed a solution that could be reprogrammed if the standard changed. Because it was important for 3Com to be one of the first to market with its new module, it was critical that a means for field programming be implemented as quickly as possible. In addition, based on potential sales volume, it was determined that only the most cost-effective mode of high volume production could be employed to manufacture the new modules.

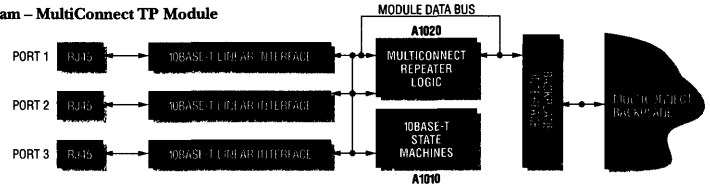
Under such constraints, Kranzler knew from the beginning that only a masked gate array could satisfy the density goals of the high-volume design. And because the market for the 3Com product was very cost competitive, a gate array would also be required to lower the module's production costs. However, the initial use of a masked gate array could not satisfy all the module's design requirements; the field programmability issue still remained.

Because the 10BASE-T standard had not been formalized at the time of the module's conceptualization, there were not a lot of off-the-shelf parts available to implement the standard, Kranzler recalls.

Actel's FPGAs: The Answer for Logic and Conversion

3Com initially looked at a number of different options to design the MultiConnect TP Module. First, Kranzler considered implementing the entire design discretely. Unfortunately, given the module's form factor, a discrete logic design would have yielded a board with only one 10BASE-T port – an unacceptable solution. Kranzler next considered designing a two-port module with high-density surface mount components, using a daughter board and the same discrete logic technology. But this alternative would have only allowed two ports and the cost would have been high. Next, the company also considered going to a full custom gate array from the onset. This solution would have provided the density needed within the cost targets but there would have been a great deal of risk involved. Finally, a field programmable gate array logic integration solution was proposed to avoid the risk element of gate arrays, the density problem of the discrete version and to eliminate the 10BASE-T specification-change

Block Diagram – MultiConnect TP Module



problem. According to Kranzler, Actel's FPGA, with an open migration path to an eventual masked array, was the clear choice.

"I was well aware of Actel's technology from the beginning and knew that its FPGAs could be used in 3Com's designs at some point," mentioned Kranzler. Actel was well suited for the design because of the generic gate-array like, granular architecture. Because Kranzler knew that these FPGAs would be utilized to almost maximum capacity, this granular architecture was a plus. In addition, with Actel FPGAs, an upgrade path to a gate array was available. Because it was proposed from the beginning to use FPGAs and then upgrade to gate arrays, this proven migration path was crucial to 3Com. Finally, because Kranzler needed maximum density within a minimum amount of space, the choice in density available to him through the ACT 1 family suited the specific needs. Ultimately, the A1010 with 1200 gates and the 2000-gate A1020 devices were chosen because they closely matched the amount of logic-integration capacity required.

According to Kranzler, Viewlogic's ViewDraw schematic software was used to capture the three-port 10BASE-T logic design and ViewSim to simulate its operation. After schematic capture, it was confirmed that an Actel A1010 and a A1020 would be needed. The choice of two Actel devices allowed the 3Com designer to cleanly partition the design of the MultiConnect TP module: The 1200-gate A1010 was used to integrate the module's 10BASE-T logic; the 2000-gate A1020 accommodated the portion of the module needed for the Ethernet repeater logic. And because Kranzler simulated the design extensively following capture, the design worked the first time it was programmed.

"The Actel devices worked out exceptionally well. I was not only happy that I could alter my design with the FPGAs in the event that the 10BASE-T specification changed, but I was pleasantly surprised that we were able to integrate as much logic as we could into the parts," explained Kranzler. "In fact," he added, "the spare gates on the devices allowed us to program logic that was used elsewhere on the board as well." According to Kranzler, the Actel devices replaced upwards of 60 TTL parts in the register-intensive 10BASE-T module design. Given the fact that the module had to accommodate three ports, a discrete-logic implementation would have been impossible.

Programmable Pinouts: An Unexpected Benefit

The programmable nature of Actel's FPGAs came in handy, Kranzler recalled, when he discovered that the pinout pattern on the board did not correspond to the FPGA pinout – the board pinout had been shifted over inadvertently by one pin during layout. "This situation would normally have been a nightmare," he said. "But with the Actel system, I was able to easily remedy the problem by reassigning the pinouts under Actel's software to accommodate the board layout, and then program another device." Thus, Kranzler was able to save the hours it would have taken to rework the board (not to mention the cost of board re-design) by simply programming the Actel device to the new pinout in a matter of minutes.

According to Kranzler, the Actel FPGAs were utilized to 98 percent of their capacity. With this utilization, 3Com was able to integrate the equivalent of three discrete TP module logic designs onto a single board. The Actel devices routed very intelligently, Kranzler remarked, and as a result, he was able to achieve his design goals and meet all of the system speed requirements. He was also pleased with the cost-effectiveness of the Actel solution. As a measure, Kranzler took his design to two other FPGA companies to see how well their products compared with the Actel two-chip solution. According to the designer, neither company could integrate the TP module with a similar dollar's worth of chips. Thus, Actel clearly was the lowest cost FPGA solution.

The FPGA-to-Gate Array Migration

The A1010 and A1020 parts were used on the initial TP Module boards that were shipped. But as the volume expanded to over nearly 1000 modules a month, it was decided, as initially intended, to convert to a 5000-usable-gate, gate array from a popular vendor for further cost reductions. According to Kranzler, the Actel FPGAs, combined with additional logic on the board, mapped into approximately 3600 gates in the masked gate array, more than the stated combined capacity of the Actel FPGAs. In fact, according to Kranzler, there was almost a 1:1 mapping of FPGA gates to masked array gates – a testament to the Actel architecture. The conversion process took approximately one month to implement and worked out extremely well, remarked the 3Com designer.

"3Com is in a high volume business and the anticipation for shipping many modules encouraged us to convert to a masked gate array," said Kranzler. There were no timing problems in migrating from FPGAs to the gate array and the teamwork between Actel and the gate array vendor made the process even smoother. 3Com currently is shipping thousands of modules a month with the gate array.

Actel Rates High with 3Com

3Com, as a first-time user, was extremely happy with Actel's FPGAs and their application into the design of the MultiConnect TP Module. Actel's FPGAs perfectly fit Kranzler's design needs. Kranzler stated that if he had a design in the future that required the use of FPGAs, he would strongly consider using Actel parts again. Stated Kranzler, "I still follow the FPGA market and I am well aware of the progress Actel has made in terms of FPGA technology. Not only are they offering denser, faster parts, but I have noticed that they are now aligning with other industry players to provide more open solutions for the designer. Indeed, the use of standard technology-transparent synthesis tools and high-level design languages will be important requirements in my next design."

Actel Corporation
955 East Arques Avenue
Sunnyvale, CA 94086
408.739.1010

 51921020

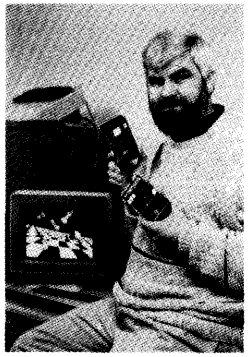
Actel

Beckworth Enterprises, Inc.

CUSTOMER CASE HISTORY

Actel

Actel FPGAs Provide High Capacity and Multiple Clocking Network for SBus-to-Printer Channel Controller Board



Billy Beckworth, *President*
Beckworth Enterprises Inc.

Beckworth Enterprises Inc. (BEI), a Mesquite, TX-based systems design firm had a fairly routine design task when it was asked by a major customer to provide an SBus interface card that would allow a mainframe laser printer to interface to Sun Microsystems' SPARCstations.® BEI's SBus channel driver was developed to allow a SPARCstation to drive a laser printer for those applications where customers did not already have a mainframe but did need the speed and performance of a mainframe printer or for printing in remote locations using lower cost SPARCstations.

With a BEI interface board integrated into Sun SPARCstations and with appropriate software, a mainframe printer would be accessible to a host of workstations – rather than a single mainframe – and, therefore, would become more versatile. But when that same customer came back a year later and asked the company to provide roughly twice the functionality in the same space, an otherwise straightforward design became a challenge.

A Solution that Reduced Board Space by 50 Percent Was Needed

Due to the board size limitations of a single slot board for the SPARCstation, Beckworth Enterprises' entire design needed to fit on 15 square inches of board space. This proved to be a challenge since Beckworth Enterprises had done a similar design with discrete logic and used over 30 square inches of board space. Although that earlier design involved an interface to a PC/AT bus, rather than an SBus, the design constraints were roughly the same, and the implementation would require the same number of devices.

With the size restrictions for the Sun workstation, Billy Beckworth, president, realized a higher integration solution was needed and began to explore higher capacity programmable logic that would allow him to cut the real estate in half. Additionally, Beckworth had design-security concerns for this project, as the SBus channel driver is proprietary and the company wanted to protect the design.

Although a custom gate array would have satisfied his real-estate concerns, cost restrictions prevented the consideration of masked devices. Moreover, Beckworth liked the flexibility of programmable logic. Therefore, the goal was to select a programmable device that was as granular and as close to a gate array architecture as possible with all the advantages of desktop programming. Having determined that programmable logic was the only solution, the requirements Beckworth had for this design were high capacity, high I/O, good speed performance and a solid security feature.

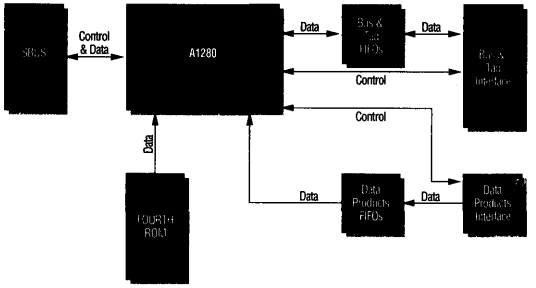
Beckworth researched the various programmable logic options and determined that the smaller programmable logic devices had the speed required for the design but did not provide the necessary density, I/O or security features. Beckworth's choice boiled down to a higher capacity solution – a Field-Programmable Gate Array (FPGA). And within that category, there were only two choices: Actel and Xilinx.

Actel Provides Security Features, Granularity Needed

With an external static-RAM memory to hold the device configuration program, as is the case with the Xilinx architecture, the risk of competitors mapping out the SRAM and determining the circuit design were too high. "There are a handful of companies that design and sell PC/AT interface cards, but as the only company that makes SBus channel drivers," says Beckworth, "we needed to protect our design. SRAMs, or any memory programming element, make it too easy for competitors to read the design and quickly benefit from our blood, sweat and tears."

Beckworth chose Actel because the chip is completely programmed internally with antifuses and provides security elements that can be blown to protect the design by preventing the map from being read. Actel's programming element, the PLICE® antifuse, is extremely small – 1.8 μm^2 – and the only way to read the antifuse map would be to view minute cross-sections of the chip under an electron microscope, a difficult, if not impossible, task.

Mega Board Data Flow Diagram



In addition to the security features, Beckworth felt that Actel's architecture was a higher grained pitch – more granular – and would allow a more flexible design approach. "Because the Actel design entry methodology closely reflects the techniques I use when designing with conventional standard logic," says Beckworth, "I could design the interface using the Actel device much the same way I would using a discrete logic implementation. Therefore, ease-of-use was high."

Board 1: Two Actel FPGAs Solve the Problem

Using Viewlogic for schematic entry, Beckworth implemented his initial design using two A1020s, Actel's 2000-gate FPGAs. The BEI interface board design was partitioned in such a way that the two devices represented distinct functions and shared no common circuitry other than power and ground. This partitioning proved useful as each device had to be driven with separate, asynchronous clocks.

Within one Actel A1020, Beckworth integrated the SBus interface and control and status registers for the board and the interface to the FIFO memories. The second A1020 handled all of the logic for the Bus and Tag channel interface. This latter device required a 5 MHz clock while the former was driven by a 25 MHz signal.

Beckworth maximized the high I/O provided in the A1020, using all but one pin on the first chip and all but six on the other. More important, real estate was also maximized. The Beckworth design implemented in 15 square inches what would normally have taken twice that amount using discrete logic.

But a year later, Beckworth's customer requested a modified version of the design. The modification called for a Data Products parallel input port plus two serial synchronous/asynchronous (USART) I/O ports to be combined with the original channel controller logic, with logic provisions to support all, onto the same, single SBus card formfactor. For this "MEGA" board design, it became clear to the designers that the approach used on the first channel controller with multiple A1020s would not fit in the same amount of board real estate. A higher density solution was needed.

A1280 Makes MEGA Board Design Possible

Due to the extremely high levels of logic integration that the MEGA board design would require to meet the equivalent board size specifications – 15 in.^{2} – Beckworth chose to use Actel's new A1280, the 8000-gate device, to replace the two A1020s and to implement the additional logic required by the modification. The A1280 provided the SBus interface and Control Section, 3211 Channel Interface and Data Products input port interface all in one 176-pin PGA package. This level of integration required the requisite drivers and receivers for the various interfaces to reside on an external interface board and a 76-pin cable connecting the two boards. Within the A1280, 133 of 140 I/O pins and 700 of 1200 logic modules were used for the MEGA design.

"The arrival of the A1280 with its 140 user I/Os and 8000 equivalent gates was the key to making the design possible at all," says Beckworth. In addition, Beckworth added that the A1280's dual clocking networks made the device the ideal choice for logic integration. This feature proved especially useful, particularly as the original design required two independent clocks. "The two independent clocking networks would not have been possible without the A1280. While I might have been able to piece a solution together using multiple subnets, that approach is not the most elegant solution."

Two Clocks are Better than One

Specifically, the two clocking networks allowed Beckworth to simultaneously clock the A1280 at 25 MHz for the SBus interface and 5 MHz for the Data Products and Bus and Tag Channel Interfaces. Separate clocks were needed because the complex state machine can't be driven as fast as the backplane due to the prop delays.

The wider gate inputs of the A1280 also proved useful to Beckworth. In fact, only one-half of the A1280's capacity is utilized because of the ability to map to the wider inputs and the combinatorial nature of the sequential and combinatorial blocks. "I tried to make it as efficient a state machine as possible," says Beckworth. "Basically, I've got two or three state machines that operate on the different inputs and outputs. That technique makes for a very clean design." As important, the larger density device offered the performance and I/O he needed to meet his size specifications.

The conversion to the A1280 from the A1020-based design was very easy and integral to the success of the new design. "The conversion process was a snap," says Beckworth. "All that was required was to take the basic [A1020] design and modify it for the additional logic functions and implement it with an A1280." Beckworth points out that although he could have recompiled the A1020 design for the A1280, it was not the most efficient approach to implement the modifications. Instead, Beckworth took the A1020 building blocks and enhanced them for the A1280 architecture. "The nice thing about the Actel architecture is that once you've designed with one device, the knowledge you've gained enables you to shorten your design time of the next project."

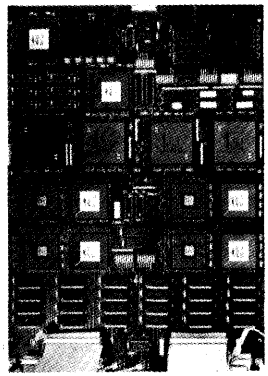
Both versions of the interface card using the Actel parts were very successful. In addition to the device benefits, Beckworth was pleased with the design support he received from Actel and felt the applications engineers were very helpful in solving his design problems. Beckworth plans to use the Actel devices in his next revision of the interface card in order to guarantee the same level of integration and performance.

Actel Corporation
955 East Arques Avenue
Sunnyvale, CA 94086
408.739.1010

 51921104-0

CUSTOMER CASE HISTORY

Interstate Electronics Opts for Actel FPGA's Speed, Density and Non-Volatile Technology



Interstate Electronics' high-speed parallel processing demodulator board

Delivering performance on time often requires getting a head start. So when a major customer tells you that the high-speed modem you've just designed and delivered needs to be 26 times faster to satisfy the requirements of the next generation product, the time to start the upgrade is now. Such was the case with Interstate Electronics, a defense systems contractor located in Anaheim, California. Interstate had just reached the final stages of development for a 25 megabit per second (Mbps) modem for NASA's second Tracking Data and Relay Satellite Subsystem (TDRSS) ground terminal. Destined to replace the present system in White Sands, New Mexico, which had been on-line since 1985, the new Interstate modem was in its production phase when the company received word that NASA was preparing a Request For Proposal (RFP) on an advanced version of the TDRSS Ground Terminal. The modem for this new system would require a 650 Mbps data rate – 26 times faster than the one Interstate had just completed. Although the RFP wasn't expected until the end of the year, Interstate wanted to step up to the challenge and prove the feasibility of meeting such a requirement with a modification to its existing digital modem design.

As far as the Interstate system designers knew, no one had ever built a digital modem that even approached these speeds. "The project presented an intriguing design opportunity for us," said Mike Casteloes, a senior engineer assigned to the project. "Moreover, because it is highly likely that NASA has plans to place a similar modem in space, that facet of the project held an even greater interest for us," he noted.

Design Objective: A Cost-Effective Approach to Higher Speed

Clearly, winning the project spurred interest at Interstate. But first, the company needed to solve the modem's speed problem, so it began working on a solution at once. Two choices presented themselves to Casteloes as he reviewed the problem of handling a 650 Mbps data rate: he could maintain the speed throughout the system with high-speed, gallium arsenide (GaAs) circuitry, or he could decrease the speed in successive stages by converting the serial input to a parallel data path.

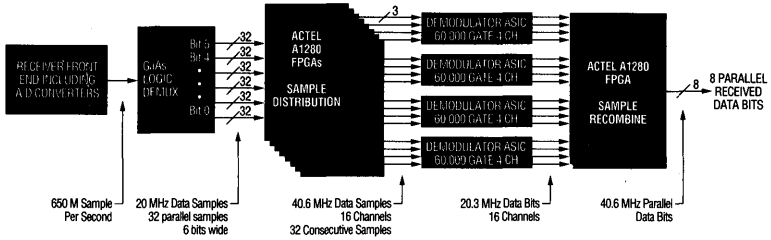
"A careful examination of the signal processing required to convert the satellite's microwave transmission to a digital output revealed that the complexity of the system would make the GaAs serial solution too expensive to implement," said Casteloes. "We opted for the parallel processing solution that employed a judicious application of GaAs, ASIC and some sort of lower cost parallel CMOS logic integration solution such as the new high-complexity PLDs (Programmable Logic Devices) or FPGA (Field Programmable Gate Array) technologies." According to the Interstate engineer, the combination offered a good performance-speed compromise to the more expensive all-GaAs approach.

The modem system design begins with a receiver front-end that converts the microwave input to an Intermediate Frequency (IF) range. At this point, the data exists as a digitally phase shifted IF waveform – a sine wave that has been modified by a stream of digital data so that every state transition causes a phase shift in the sine wave.

The receiver's IF output is then converted to baseband (mixed) and sampled using high-speed analog-to-digital (A/D) converters. The resulting serial stream of 6-bit samples is then fed to a GaAs demultiplexer circuit that separates the 650 megasample-per-second data into 32 parallel samples, each six bits wide. Splitting the data into parallel samples slows the data rate to 20.3 MHz, a rate that easily can be handled by less expensive CMOS circuitry.

The next stage of the system required a massive array of shift registers to store and distribute the sample data to a bank of four 60,000-gate custom gate arrays. "This was a critical section of the design," Casteloes recalled. "To convert the sampled analog data to logical ones and zeros, we needed to collect the data into blocks of 32 consecutive samples. Otherwise we would never have been

650 MHz Receiver Structure Using Parallel Processing



able to reassemble them accurately. To accomplish this we designed an array of 32 shift registers, each one 32 x 3 bits wide. By combining their outputs, we could produce 16 channels, each containing 32 consecutive samples, and each running at 40 MHz."

The Selection:
Non-Volatile Fuse Map is the Key

"Designing the circuit was straightforward enough, but implementing it posed a problem with board space," said Casteloes. "It simply wasn't feasible to implement that many shift registers in PALs; it would have taken up far too much room. And even if we used any one of the higher-complexity PLDs, it still would have required too many devices to implement the design." The designer recalls that creating another ASIC was also out of the question. "We had spent a considerable amount of non-recurring engineering (NRE) charges with the custom ASIC already. Spending even more on NRE was immediately discounted," he added.

FPGAs seemed the only logical choice, and Casteloes was already familiar with these devices – he had used Actel's FPGAs and design tools for an earlier project. "We calculated that our Sample Distribution circuit would only require eight A1280s – less than a quarter of the number of high-complexity PLDs that would have been required to implement the same function," said Casteloes. "As it is, the amount of I/O on our 10" x 18" board is substantial," he added.

Casteloes also looked at the A1280 because of the high speed requirements of the system. "Finding an FPGA that can operate with a 40 MHz system performance is no easy task," he said. "We believed we could get that kind of performance out of the A1280 devices because of our prior experience with the Actel architecture."

Actel's FPGA was not the only product considered. "Although we could have used a greater number of Xilinx FPGAs because they had adequate performance for this particular application, the A1280 device had one major advantage – its fuse map used a non-volatile technology," Casteloes explained, "which guarantees that the device will be mapped correctly on each power up." According to Casteloes, designing with the static-RAM-based Xilinx product would have required down-loading all fuse map information each time the system was powered up. "With all the high-speed digital switching on board," Casteloes observed, "I was afraid the slightest noise glitch could cause the Xilinx device to be configured incorrectly and malfunction, a situation that you wouldn't want to tolerate on the ground and which couldn't be tolerated in a satellite-based system."

Along with the potential vulnerability of the system, programming the fuse map each time would require additional circuitry, including a PROM to store the fuse map information for each unique FPGA design, as well as support for diagnostics to test each FPGA after power up. "The choice was obvious," said Casteloes, "Actel was the clear winner."

The next stage in the system reassembles the 6-bit samples into digital data. Four ASICs synchronize the sample data received from the FPGAs and reconstruct them into 16 channels of useable data. The ASICs also provide timing and phase error estimates. Finally, the 16-channel parallel output of the ASICs are recombined into eight parallel data bits at 40 MHz by an additional FPGA. Ultimately, the digital data is fed to a GaAs 8:1 multiplexer used to convert the parallel data back to a serial data stream for retransmission.

Actel FPGAs are the Right Choice

Implementing a complex design in a high-performance, high-density environment is a challenge under any circumstances. Developing a working design quickly can seem insurmountable without the aid of FPGAs. "Even though we are just now shipping the first of the 25 Mbps systems to NASA, we have already proven the 650 Mbps system works, and we were able to stay within the circuit board space and budget constraints," said Casteloes.

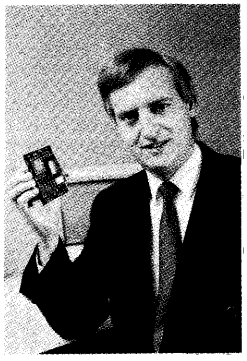
"If I had my way I would use Actel FPGAs for every high-density design," Casteloes said. "Actel's products are easy to work with, offer the highest available logic density and performance and can be relied upon to satisfy the most demanding design requirements. The way I look at it, once you have a good design environment, stay with it!"

Actel Corporation
955 East Arques Avenue
Sunnyvale, CA 94086
408.739.1010

5192105-0

CUSTOMER CASE HISTORY

Delphi Combines Actel FPGAs and DSP Technology for Superior Speech Compression



Peter Schwartz, *CEO*
Delphi Systems

With the increased use of conventional phone lines as a major communications medium, many large companies are relying on private telephone networks to reduce their communications costs. In such cases, higher-cost leased lines would primarily be used for long distance traffic. Although these leased lines often carry data, the predominant requirement is to carry voice traffic.

In the UK, British Telecomm (BT) and Mercury Communications Ltd. are the typical suppliers of long distance services. A BT KiloStream service, normally used to carry data with a capacity of 64kbits/sec, can carry only one PCM (Pulse Code Modulated) speech channel, while MegaStream at 2.048Mbits/sec was developed for higher traffic and is equivalent to 30 KiloStream channels. Although MegaStream is considerably more costly than the KiloStream service, users have been willing to pay a premium to gain the higher capacity channels. It's to no surprise, therefore, that there have been considerable efforts by several firms to improve the channel-handling capabilities of lower-cost long distance communications lines.

At least one company, Delphi Systems (Salisbury, Wilts), has pioneered a solution. Delphi is using speech compression techniques to enable multiple-channel communications over the single-channel KiloStream service. In fact, the company's newly developed Delphi Speech Compression CODEC (DSC) provides speech compression sufficient to allow KiloStream to carry up to eight toll-quality voice channels plus control information, or twelve at near toll-quality, potentially reducing costs and offering the communications manager more flexibility in planning his network. Savings vary according to the amount of traffic. As an example, for between five and 20 lines, DSC speech compression on a KiloStream service would cost around half the MegaStream equivalent in the first year, reducing to around 20 percent of the cost thereafter.

The DSC exploits a technique called Code Book Excited Linear Prediction (CELP) to achieve this compression. A prediction based approach, CELP was first detailed by AT&T Bell Labs in a 1985 IEEE conference paper, where speeds 100 times slower than real time on a Cray 1 supercomputer were reported. By the late 1980s, the emergence of high performance digital signal processors made it possible to carry out the heavy computation needed for CELP in real time. Delphi was founded to exploit the market for CELP-based communication products, particularly the private telephone network market.

Different Product Phases Require Flexible Solutions

According to Peter Schwarz, chief executive of Delphi Systems, the product development for the DSC was segmented into three distinct phases: algorithm development and feasibility study, evaluation products and final implementation.

The first phase, algorithm development, had the twin objectives of creating and efficiently coding the CELP technique and confirming the feasibility of running CELP in real time. Schwarz recalls that at the time the project was initiated, only AT&T's 32 bit floating point processor, the DSP32, was available; the higher performance DSP32C was due within the next six months. Software was initially developed on a personal computer and coded in Turbo Pascal, with the intention of porting to the DSP32C when it became available.

Initially running 5,000 times slower than real time, a subsequent port to a higher-performance PC – a 386 machine with a 387 co-processor – produced a fifty-fold speed improvement. Porting to the DSP32 saw a further 80 times improvement, sufficient to both prove the technique and provide confidence that with the next move, to a DSP32C, real time speech compression was viable.

The second stage was to develop a real time system on a PC board to enable potential customers to evaluate the CELP technology, educating the potential user base on the technology and providing a revenue stream to help fund future developments. The evaluation systems used the first commercially available DSP32C parts available in the UK and

were sold to a number of telecommunications and related companies. One commercial product was based directly on the PC board: a multi-channel voice logging system for emergency services. Compressing speech and storing it on hard disks made it possible to instantly retrieve and replay random messages while continuing to record incoming calls, something impossible on previous tape based systems. Customer feedback from the evaluation systems was positive and provided valuable input to the final design stage.

Choosing the Logic: Density, Security and Flexibility Requirements

At the beginning of the third phase, the development of the final product, there was a technology evaluation of the most appropriate form of logic. The board was to be a compact daughter board, placing a severe limit on the chip count after the DSP, EPROM and SRAM circuitry were added. The technology evaluation criteria included the potential chip count, security (both in use and to protect the proprietary elements of the product), ease of design and cost. Products evaluated included a microcontroller, masked gate arrays and field programmable gate arrays (FPGAs). The microcontroller was ruled out on both chip count and lack of security. The masked gate array was inflexible, had long turn-around times, was expensive with large up-front NRE charges and did not suit Delphi's preferred incremental development route. FPGAs, therefore, became the technology of choice.

Two FPGA families were evaluated, one from Actel Corporation and the other from Xilinx Inc. According to Schwarz, "although both FPGA approaches had no NRE and offered fast system implementation, we quickly concluded that the Actel part was the logical choice. Indeed, because it used an antifuse technology for programming rather than a static-RAM plus external EPROM-based technique, it was nonvolatile and provided a much higher level of security – a strong requirement for the DSC." The Actel gate array-like architecture, Schwarz continued, also offered a straightforward migration path to a full masked gate array as production volumes rose. The design tools were available on a PC platform, further minimizing the up-front costs.

Using Actel FPGAs For Integration

Once the decision to go to Actel was made, an Action Logic™ development system was purchased from Gothic Crellon, a UK-based sales representative. Schematic capture and simulation from Viewlogic® and place and route device programming from Actel were used.

The challenge to Delphi's design team was to create a standard set of modules, integrating these to create larger functional blocks. "As an area was integrated," said Schwarz, "a device was programmed and tested in a prototype board. At the early stages of development, both the EPROM and the DSP could be run as emulation systems via the PC with connections to the prototype board." The first A1010 (1200-gate) device contained only the clock circuitry and the circuits to load code from the EPROM to the DSP32C. There was no I/O, so the PC tools were used to ensure that this part was functioning correctly.

The step-wise approach allowed the final development of the software to take place in parallel to the hardware, an approach impossible with a standard gate array. This approach provided very tight project control and was a major factor in the ultimate success of the project.

Timing Mystery Solved with Actionprobe

Schwarz points out that at the final iteration of the design, an unexpected problem arose. "After adding the final security circuitry to the tested and fully functional design," he remarked, "some programmed devices inexplicably failed to work correctly." He remembers that a period of careful analysis identified the problem as a timing error that had not been present previously. Following a design change, the Actel system reruns the place and route, resulting in a different layout on the FPGA. In this case, a timing alteration had occurred with a circuit that had not been identified as a critical net. Further analysis localized the problem. However, even the post layout simulation failed to locate the problem – the circuit simply did not operate in accordance with the simulation.

To isolate the problem, Schwarz decided to take advantage of a special diagnostic feature that was developed for Actel FPGAs. Called Actionprobe,™ the feature exploits the antifuse programming circuitry to probe any pair of internal nodes in an in-circuit functioning device without loading the internal nodes or modifying the circuit operation. Coupling Actionprobe to an oscilloscope allowed signals from the faulty circuit to be displayed. These were directly compared with the post layout simulation waveforms, displayed on the PC screen. This direct comparison reduced the actual time spent in locating and probing the timing problem to minutes.

With the timing problem identified, place and route was instructed that this was a critical net and the re-layout proved to be 100 percent successful. The Actionprobe was used again with the relayed chips to measure the timing tolerance in the critical nodes.

Actel FPGAs Are the Right Choice

The DSC is now in production with six major variants. These include versions that operate at either 6,400 bps or 4,800 bps and a dual speed switchable version; each of these is available with or without an echo canceller. In each case, the PCB and assembly is identical, with the actual variant being produced by selecting one of three different FPGAs and one of two different EPROMs.

Manufacturing is contracted out. When an order is placed, parts are ordered from Gothic Crellon, programmed by them to meet the order requirements and passed to a subcontractor for board build. Capital tied up in stock is kept to a minimum with this just-in-time approach and the programmed FPGAs are produced as needed. The latest version of the CODEC now uses double sided surface mount technology (SMT) to shrink the board from 4" by 3.75" (15 square inches) to 2" by 3" (6 square inches), but still uses the same Actel FPGA.

It is fair to say that without the Actel FPGA approach not only would Delphi not have the same range of CODECs as are offered today, but the product itself might never have reached the market.

Actel Corporation
955 East Arques Avenue
Sunnyvale, CA 94086
408.739.1010

 5192105-0

GE Medical Systems

CUSTOMER CASE HISTORY

Actel Corporation

February, 1993

GE Medical Chooses Actel FPGAs for their Design Flexibility and Ease-of-Use

GE Medical Systems, one of the leading suppliers of medical imaging equipment, continues to push the limits of technology through on-going product development efforts in order to provide customers with the most advanced real-time imaging systems available. One of the recent systems from the Waukesha, Wisconsin-based organization, required the design of a complex image acquisition and display board to facilitate the processing of images. The image acquisition and display board, a VME-based board residing in a Sun SPARCServer®, would need to take the image data from a proprietary data detection device, store it into the board's image memory, process the image and drive a high-line (1024 x 1024) display monitor. However, this was not an easy design task because of the significant amount of logic to be incorporated on a board strictly limited by the fixed VME form factor.

Additionally, in the fast paced world of medical imaging, where systems are constantly being enhanced and new technologies replace old, limiting the system design time is of the essence. Therefore, the engineers were working to meet a six month design time frame for a fully implemented imaging board.

Clearly, choosing the logic integration vehicle would be key in maximizing the engineering team's productivity. Moreover, the solution would need to meet the dense logic requirement of the application in the limited space of the 14" x 14", 9U VME board and all in the short six month time frame. Mike Juhl, senior design engineer with GE Medical Systems, initially evaluated several options before deciding on the one that best met his needs.

One option was to use a completely discrete solution. However, Juhl immediately rejected this because he knew the application was heavily logic intensive and would require a large number of discrete devices, which would exceed the limited board space. In fact, if implemented using discretes, the solution would need at least 600 MSI devices, translating into 3 VME boards, for implementation.

GATE ARRAY USE DISCOUNTED

A second alternative was to use masked gate arrays, which could integrate the logic required to accommodate the limited board space. However, the design, implementation and fabrication of a single gate array can itself be a six month project, and Juhl needed a completed board in six months. To use the gate array solution, it would be absolutely necessary to have first-time-design success. Even a single design flaw would require a new gate array. Juhl determined he could not take the risk.

"The decision seemed obvious," Juhl recalled. "The only viable solution that would meet all of our requirements would be the flexibility of field programmable gate arrays (FPGAs). FPGAs would provide the high capacity required for the application in the board space available, and, because of its field programmability, the design could be implemented quickly."

The next step was to determine which FPGA supplier to choose. Juhl knew approximately how much I/O and flip-flop capacity he would need for the design implementation. Because he wanted to partition the design by functionality, he knew that he would need to use multiple FPGAs, each in the 6,000 to 8,000 gate arena.

After comparing the available solutions, Juhl concluded that he would be able to get the densest solution available by using Actel's A1280, an 8,000 gate device from the ACT™ 2 family. The A1280 had a usable gate equivalent of 6,400 gates, with

955 E. Arques Avenue Sunnyvale, CA 94086 408 739-1010

120 user I/Os and 998 flip-flops. Additionally, he had implemented a design using Actel's ACT 1 family in a prior system and was very happy with their performance.

AN EXTREMELY FLEXIBLE SOLUTION

Another factor that influenced Juhl's selection of Actel was his previous experience with the ACT 1 family and Action Logic™ System for design and programming. "The migration from the ACT 1 tools to the ACT 2 tools was very easy. In fact, the design process didn't change at all," stated Juhl.

Juhl designed the board using Mentor Graphic's schematic capture tool, NetEd, and digital simulator, QuickSim, which support Actel's library. These tools provided support for both funtional and post-layout timing simulation. "This is a key requirement for designers today. Not only does the FPGA need to work, but eventually the entire board needs to work. With Actel's devices and tools, in conjunction with QuickSim, the FPGA can be simulated by itself and then re-simulated in the complete board environment," stated Juhl.

FPGA INTENSIVE: USING 10 ACTEL DEVICES

When Juhl completed the design and implementation, within his six month window, the image acquisition and display board required approximately 75,000 gates of logic. To accommodate the logic requirement, Juhl used 10 Actel devices: nine A1280s and one A1020, a 2,000 gate device from Actel's ACT 1 family. Each of the Actel devices is partitioned by its functionality in the system. Functions include VME interface, receiver data decode and format, image memory data interface, memory controller, transmitter data format and video data format.

Using multiple FPGA devices from any other supplier would have made the placing and routing of the individual device an extremely difficult and time-consuming task. Thus, Actel's ability to perform 100% automatic place and route was a key advantage for multiple-FPGA designs. "Once the partitioning of the 10 devices was completed, programming them was as easy as pressing 10 buttons, walking away and coming back to find they were 100% placed and routed. This saves engineering time that can then be applied to other activities, rather than spending it manually placing and routing," commented Juhl.

Operation of the board occurs by the image data coming in from a detector source, via a fiber optic channel. Working with the Sun host CPU, the VME interface consisting of one A1280 device, receives commands that tell it what format that the data is coming in. For example, the commands inform the board of whether the data is encoded and whether it is coming in at a 768 x 768 or a 1024 x 1024 line rate format. The receiver data decode and format devices then process the incoming data, with two A1280s to process two pixels in parallel, and send the formatted data to the image memory data interface. The four A1280s used for the image memory data interface connect the on-board image pipe-line to the 64 MByte DRAM buffer, which stores up to 32 images in a 1024 x 1024 format.

The memory controller is the main control function on the board arbitrating the storage of received images with the real-time display of images and the transfer of images over VME to the host CPU. In the video display mode, the controller can be configured to display any sequential set of images in the buffer in a variable rate video loop. Additionally, the memory controller can be configured to operate as a simulator for the detector source and transfer images out over a fiber optic channel at a 30 frame per second rate.

The memory controller was the most challenging design on the board. However, Juhl hoped that with the A1280 and careful design he would be able to integrate the solution into one device. Due to the efficiency and flexibility of the Actel PLICE® antifuse-based architecture, Juhl was able to implement the memory controller using only one A1280, utilized at the extremely high rate of 95%. "This was the most difficult design in the system and was efficiently implemented in the Actel A1280 device," stated Juhl.

Because the system performance requirement for the design was specified at 28 MHz, Juhl did not have a performance concern with the Actel parts,

which offered operation to 55 MHz. Juhl added, "I had no problem obtaining the performance I needed from the Actel devices."

UNEXPECTED BENEFITS FOR ADDITIONAL FUNCTIONALITY

In addition to the density, flexibility and performance Juhl expected from the Actel devices, he gained additional functionality not originally anticipated. "One benefit of the Actel approach was the ability to put boundaries around the logic by partitioning functions to each of the FPGAs —I felt confident that the FPGAs would be able to handle those functions. I was then able to start placing and routing the board, even before I had all the detailed design completed on a couple of the FPGAs."

Juhl also discovered that the Actel devices allowed him to add functionality late in the design cycle, typically a monumental task with other programmable logic solutions. "I was pleasantly surprised that I could add functionality to the Actel devices late in the design cycle with basically no extra cost. This happened a couple of times during the design cycle when we put additional features on the device and once when we decided to change functionality after we had fabricated the board."

NEXT GENERATION ALREADY IN DEVELOPMENT

Juhl was very satisfied with his choice of Actel FPGAs and continues to use them in his next generation design. Currently, GE Medical Systems' next generation of the image acquisition and display board is in the R & D process. "In general, the new design is a bit more complicated, but Actel FPGAs are easily handling the increased logic complexity. We will also be able to save even more time in this second generation by leveraging the design experience and the macros from the initial design."

Indeed, GE Medical has chosen to stay with the Actel FPGAs but has changed the mix of the Actel parts to accommodate the new partitioning of functions on the board.

Juhl commented, "Again, we are under a tight time frame for the development of the second generation board and believe that the Actel devices offer the functionality that we will need. The devices provide high available logic density, offer an easy-to-use device and design tools package that is well integrated with third party board level design and simulation tools, and provide the performance necessary for this logic intensive system."

955 E. Arques Avenue Sunnyvale, CA 94086 408 739-1010

Chipcom Co.

Article published in the 1993 PLD Conference Proceedings

1.1.3.C

Real World State Machine Design with FPGA's

Ira D. Hart, Section Manager
Chipcom Co., Southborough, MA

Introduction

Designing state machine control circuitry for implementation in an FPGA is easier today than ever before. A designer no longer needs to spend a great deal of time optimizing state machines with Karnaugh maps and state tables. With the advent of synthesis tools such as those made by Exemplar or Synopsys, a state machine can be described in an HDL such as ABEL or VHDL and quickly converted to FPGA gates. The state machines can be synthesized with the constraint of gate count or speed. They can be represented in your schematic as small behavioral "black boxes" which can be linked together and simulated. The FPGA and it's associated tools allow a designer to quickly capture the design and burn it at his or her desk to prove the design functionally in the lab.

This paper will describe a design example and the trade-offs made for an interface between a processor, 2 Ethernet controllers and a shared SRAM bank. The design contains 6 state machines, shift registers, counters and multiplexing circuitry for the data and address busses. It was implemented in an ACTEL 1280 FPGA which can fit roughly 8,000 gate array gates and has 140 user I/Os of which 125 were used in this design

Arbiter Description

A memory arbiter is used in this design to allow the 2 lances and the CPU fair access to a shared SRAM. The scheme used is first come first serve. If there is contention, the last requester to have been granted access to SRAM will wait. The arbiter was described in ABEL HDL in a "one hot" or state per bit encoding scheme. ABEL outputs a .PDS description format which is basically a Boolean description. This is read in by the Exemplar synthesis tool which will output Actel ACT2 gates. The Exemplar tool spends 5 minutes analyzing the Boolean description and providing different solution sets. For the Arbiter in this example, Exemplar was setup to optimize for fastest speed. The Exemplar tool made 9 passes with the best pass occurring on the first iteration. The resulting arbiter was implemented in 77 ACT2 modules with a worst case setup time to the FF of 46.3nS.

Problems with One Hot Encoding

Classroom state machine design techniques usually involve binary encoding for state machines. For example 4 bits can be used to represent 16 states. Decoding logic for the outputs or next state then look at the inputs and the previous state to determine what to do at the next clock edge. Frequently, the decoding logic necessary to decode the transition conditions for changing to the next state use multiple module levels. This can cause timing problems if the number of levels are high enough that the propagation delay through the decoding logic causes a setup violation at the input to the state bit flip flop.

One hot encoding is frequently recommended by FPGA vendors as a way to improve speed and module count. This is done by increasing the number of flip-flops used to represent the states. The state transition conditions usually include a very low number of logic levels in the decode

block, making the operating frequency high. Simple combinatorial logic is used to decode the state bits. For example, a case where an output needs to be low if it is in state 3 and state 4. A 2 input NOR gate can be tied to the Q output of each one hot flip flop.

This output would theoretically remain low in each state and during the transition between states. In reality though, the output will glitch during the transition due to the simple fact that the flip flop for state 3 may turn off before the flip flop for state 4. This glitch would cause a serious problem if the output were controlling an asynchronous input such as a chip select or output enable of a memory device.

A Better Approach

The problem descibed above was encountered in this Actel 1280 design for the signal UDTACK1. This signal is the data acknowledge handshake to the board CPU. The CPU runs at a different clock speed than the arbiter block. It is an asynchcronous signal which needs to remain glitch free. One option was to synchronize it with the CPU clock but this would of course have led to a performance hit equal to up approximately two CPU clocks. One approach is to use an HDL which decodes the inputs and previous states synchronously, to "look ahead" and provide glitch free transitions of the output on state machine clock changes. This is accomplished in the arbiter example. The disadvantage of this approach is that these outputs must now decode the inputs and previous states to switch. This can result in the same problems we had with the encoded state machines, because of the complexity of the logic needed in front of the flip/flop. An alternative approach is to use a pseudo- 'one hot' technique where more than one state is active at a time, insuring glitchless transition between states which require it. This usually will not require any additional logic, since the state flip/flop is simply cleared on exit of a different state.

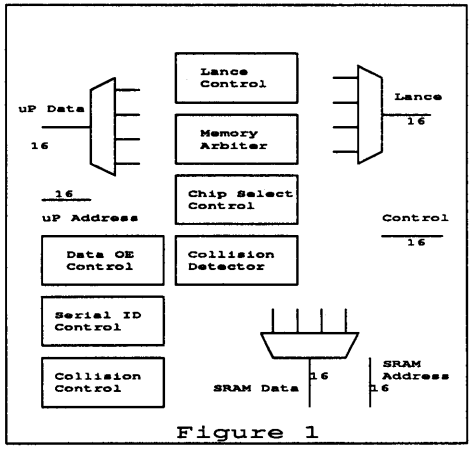


Figure 1

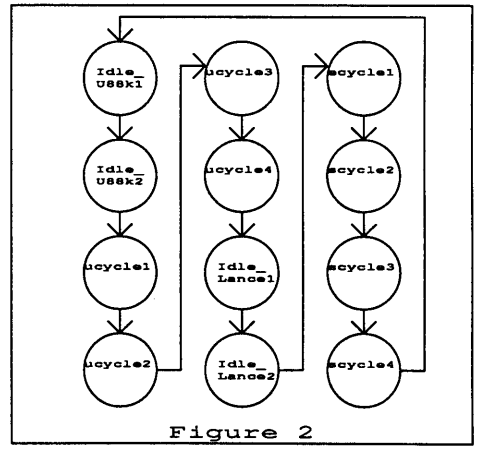


Figure 2

1.1.3.C

Arbiter Abel file

```
MODULE memarb
title 'Dual Port Memory Arbiter
Ira D. Hart     Copyright CHIPCOM Co. 1992 '
declarations
"InputsClk20pin istype 'buffer';
Resetlpin       istype 'buffer';
SRAMCSlpin istype 'buffer';
"Outputs
UDTACKlpin istype 'reg, buffer' ;"68k
ADDMUX1pin istype 'reg, buffer'
EDTACKlpin istype 'reg, buffer'
s12pin istype 'reg, buffer' ; "states
s11 pin istype 'reg, buffer' ; "states
s10 pin istype 'reg, buffer' ; "states
s9pin istype 'reg, buffer' ; "states
s8pin istype 'reg, buffer' ; "states
s7pin istype 'reg, buffer' ; "states
s6pin istype 'reg, buffer' ; "states
s5pin istype 'reg, buffer' ; "states
s4pin istype 'reg, buffer' ; "states
s3pin istype 'reg, buffer' ; "states
s2pin istype 'reg, buffer' ; "states
s1pin istype 'reg, buffer' ; "states

ADDMUX =[ADDMUX1,ADDMUX0];
U68K  =[ 0,   0    ];
Lance =[ 0,   1    ];
PP    =[ 1,   0    ];
CC    =[ 1,   1    ];
High,Low   =1,0;
H,L,c,x,z  =1,0,.C.,.X.,.Z.; "test vector characters

Qstate=[s12,s11,s10,s9,s8,s7,s6,s5,s4,s3,s2,s1]; "
Idle_U68K1= ^b000000000001; "s1
Idle_U68K2= ^b000000000010; "s2
Idle_Lance1= ^b000000000100; "s3
Idle_Lance2= ^b000000001000; "s4
uCycle1=    ^b000000010000; "s5
uCycle2=    ^b000000100000; "s6
uCycle3=    ^b000001000000; "s7
uCycle4=    ^b000010000000; "s8
eCycle1=    ^b000100000000; "s9
eCycle2=    ^b001000000000; "s10
eCycle3=    ^b010000000000; "s11
eCycle4=    ^b100000000000; "s12
```

```
1.1.3.C

equations
[UDTACK1,EDTACK1,ADDMUX1,ADDMUX0,memcycl,s12,s11,s10,s9,s8,s7,s6,s5,s4,s3,s2,s1
s2,s1].CLK =                                    Clk20;
[ADDMUX1,s12,s11,s10,s9,s8,s7,s6,s5,s4,s3,s2].RE=    !Reset1;
[memcycl,ADDMUX0,EDTACK1,UDTACK1,s1].PR=    !Reset1;

state_diagram Qstate

State Idle_U68K1:
EDTACK1:=!(!EDTACK1 & (!EAS1 # !clk10)); "Sustain till end of EAS. w/clk10
If (!SRAMCS1) Then uCycle1                    "68K has first priority.

WithADDMUX :=U68K;
memcycl:=0;
UDTACK1 :=1;
Endwith;

Else Idle_U68K2
WithADDMUX  := Lance;
memcycl := 1;
UDTACK1 := 1;
Endwith;

State Idle_U68K2:
If (!SRAMCS1) Then uCycle1      "68K has first priority.

WithADDMUX:=U68K;
memcycl:=0;
UDTACK1 :=1;
EDTACK1:=!(!EDTACK1 & !EAS1); "Sustain till end of EAS.
Endwith;

Else If (!EAS1 & clk10 & EDTACK1) Then eCycle1
                        "Lance has 2nd priority.
WithADDMUX:=Lance;
memcycl:=0;
UDTACK1 :=1;
EDTACK1 :=1;
Endwith;
```

1.1.3.C

Arbiter Exemplar Summary File

Exemplar Logic Synthesis System Thu Nov 19 13:58:14 1992

Pass	Area (modules)	Estimated Delay (ns)	CPU min:sec
1	77	46.3	00:30
2	85	55.2	00:32
3	81	47.1	00:30
4	81	47.1	00:31
5	81	46.3	00:30
6	117	55.2	01:31
7	81	47.1	00:31
8	81	47.1	00:32
9	81	47.1	00:31

Resource Use Estimate
Design: memarb
Technology:act2
File: memarb.pds
Area: 77.0
Critical Path: 46.3 ns

Device	Area avl/usd /pct	Registers avl/usd/pct	i/o avl/usd/pct
A1225	451/ 77/ 17%	341/ 0/ 0%	82/ 24/ 29%
A1240	684/ 77/ 11%	565/ 0/ 0%	104/ 24/ 23%
A1280	1232/ 77/ 6%	998/ 0/ 0%	140/ 24/ 17%

Delay Summary

Node:	Slack rise	Slack fall	Arrival rise	Arrival fall	Required rise	Required fall	Load
S4	: 31.00	15.30	15.30	46.30	46.30		1.60
S1	: 32.60	13.70	13.70	46.30	46.30		1.60
S3	: 32.60	13.70	13.70	46.30	46.30		1.60

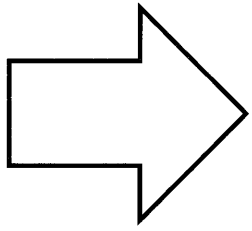
1.1.3.C

Cell Usage Summary

Cell	Uses	Cost	Total
AND2	4 uses(s)	1.00 modules	4.00 modules
AND2A	2 uses(s)	1.00 modules	2.00 modules
AND4A	1 uses(s)	1.00 modules	1.00 modules
AO1C	2 uses(s)	1.00 modules	2.00 modules
AO2	3 uses(s)	1.00 modules	3.00 modules
AO3	4 uses(s)	1.00 modules	4.00 modules
AO6A	2 uses(s)	1.00 modules	2.00 modules
AO7	2 uses(s)	1.00 modules	2.00 modules
AOI1	1 uses(s)	1.00 modules	1.00 modules
AOI2B	1 uses(s)	1.00 modules	1.00 modules
AOI4A	1 uses(s)	1.00 modules	1.00 modules
BUF	9 uses(s)	1.00 modules	9.00 modules
DFC1B	13 uses(s)	1.00 modules	13.00 modules
		Total =	77.00

Number of combinable modules is: 5

Technical Support Services

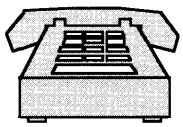


Technical Support Services

Actel Technical Support

Actel's application engineers have developed many ways to meet your needs. Our services provide technical assistance to all of our customers worldwide. Customers can obtain technical assistance by means of the Technical Support Hotline, the Bulletin Board Service (BBS), Customer Training, International email, the technical newsletter, and our fax-back system, Action Facts.

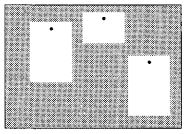
Technical Support Hotline



The Technical Support Hotline provides technical information on Actel hardware and software products. Questions regarding software authorization, availability, and pricing are handled through Actel customer service. All hotline calls are answered by our Technical Message Center. The center retrieves information such as the caller's name, the company name, the phone number, and the caller's question, and then issues a case number. The center then fowards the information to a queue where the application engineers receive the data and return the customer call. Our goal is to return all customer inquiries within one hour. The hotline's phone hours are from 7:00 a.m. to 5:30 p.m. PST. In addition to answering the Technical Support Hotline calls, our applications engineers develop other ways of assisting our customers, such as writing application notes, and user guides, creating design examples, and evaluating software.

The Technical Support Hotline number is 800-262-1060.

Electronic Bulletin Board Service (BBS)



Actel currently offers information access and transfer via a 24-hour worldwide bulletin board. Customers can download information such as new soft macros and software bug fixes. In addition, our applications engineers use the BBS to help solve design problems. Sometimes a customer's issue can not be solved by the technical hotline. In these cases, our applications engineers may request the customer to upload their design to the BBS. All files uploaded to the Actel BBS are automatically stored in a private directory. This way the applications engineers can examine the software first hand and make the necessary adjustments.

The telephone number for the BBS is 408-739-6397. To connect to the BBS by modem, the following equipment and configuration are required:

- Baud rate of 9600
- Data format: 8 data bits, 1 stop bit, no parity

The following file transfer protocols are supported:

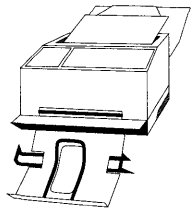
- Xmodem
- Ymodem
- Zmodem
- ASCII

First-time callers need to establish an account. Once connected to the BBS, the caller is then prompted for his/her name, company, phone number and so on. After the account is established, the customer calls the Technical Support Hotline 800-262-1060 and requests file transfer class. Our Technical Message Center verifies the name and company and then upgrades the account's security level.

Customer Training

Actel offers an introductory two-day course covering all aspects of designing an Actel FPGA. The class covers a discussion of the architecture of ACT™ 1, ACT 2 and ACT 3 families design methodologies, a brief look at the Viewlogic® schematic capture and simulation tools, and a thorough examination of the Actel FPGA design software. The course is a mixture of lectures and lab exercises. Classes are offered to a minimum of three persons at a time. Customers can register for our training course by calling the Technical Support Hotline and asking about training class registration.

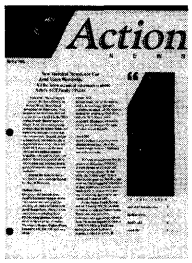
Action Facts



The Action Facts system is a 24-hour fax-back service. Customers can obtain a list of current software bugs and workarounds, application notes, design hints, package pinouts, and much more information. Simply call the toll free number and request that a catalogue be faxed to your office. Review the document and call the number again to request up to five documents per call. The Action Facts Catalogue is updated bimonthly.

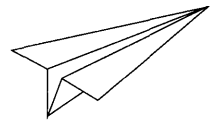
The Action Facts phone number is 800-262-1062.

Technical Newsletter



In spring 1992 Actel started a quarterly technical newsletter. The newsletter contains the latest Technical Support Hotline news, information about new hardware and software products, the answers to the most commonly asked questions, and detailed discussions from Actel's experts about hardware and software tools. All customers who have software maintenance automatically receive the newsletter.

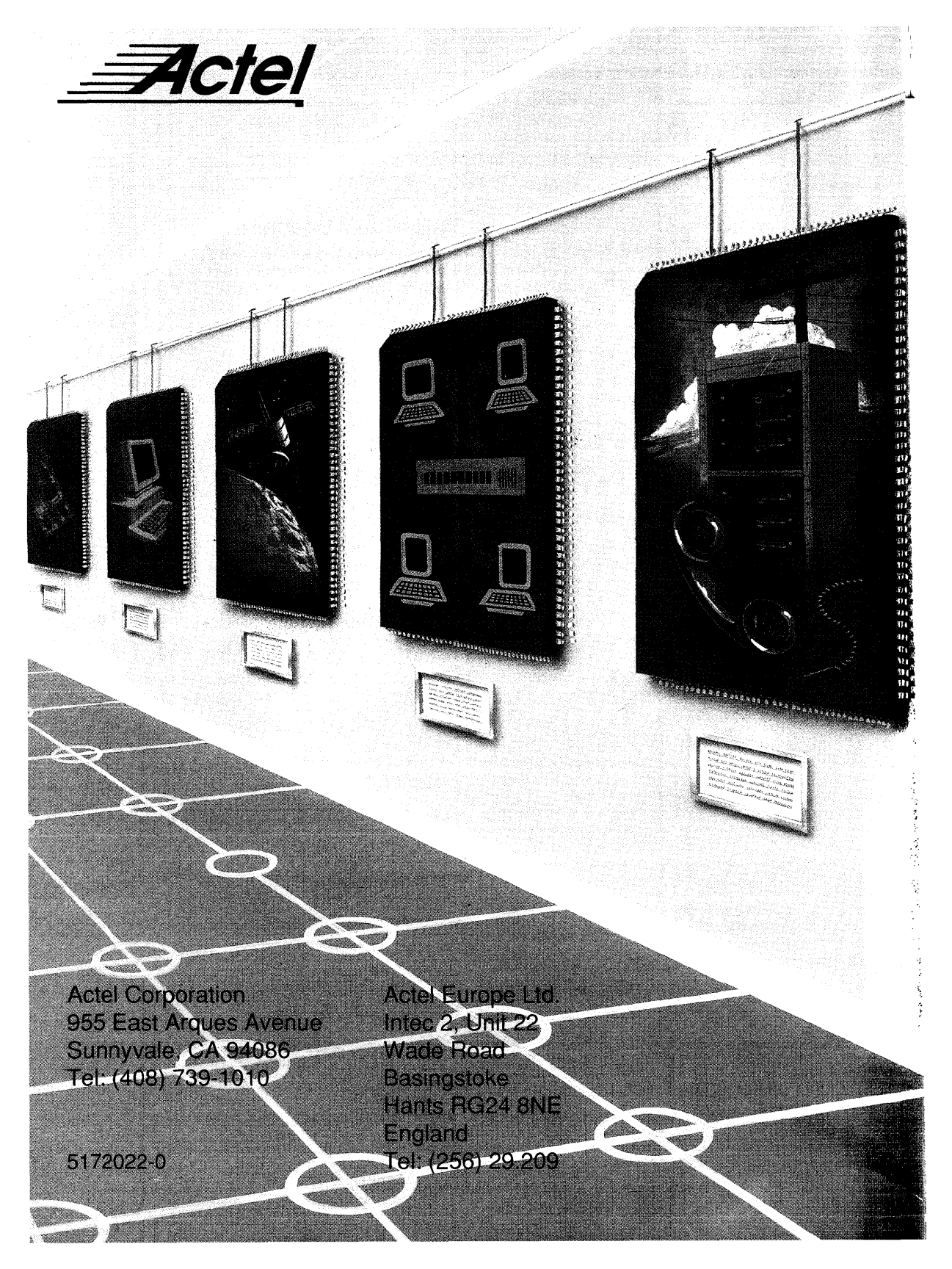
International E-mail Address



Actel has recently introduced a new method for customers to communicate with our applications engineers. Customers can e-mail their technical questions to our e-mail address and receive answers back either by e-mail, fax, or phone call. In addition, customers can quickly e-mail their design files to receive assistance. The e-mail account is monitored several times throughout the day, and our goal for responding is two working days after receiving the message.

Technical support's e-mail address is

tech@actel.com



Actel Corporation
955 East Arques Avenue
Sunnyvale, CA 94086
Tel: (408) 739-1010

Actel Europe Ltd.
Intec 2, Unit 22
Wade Road
Basingstoke
Hants RG24 8NE
England
Tel: (256) 29.209

5172022-0