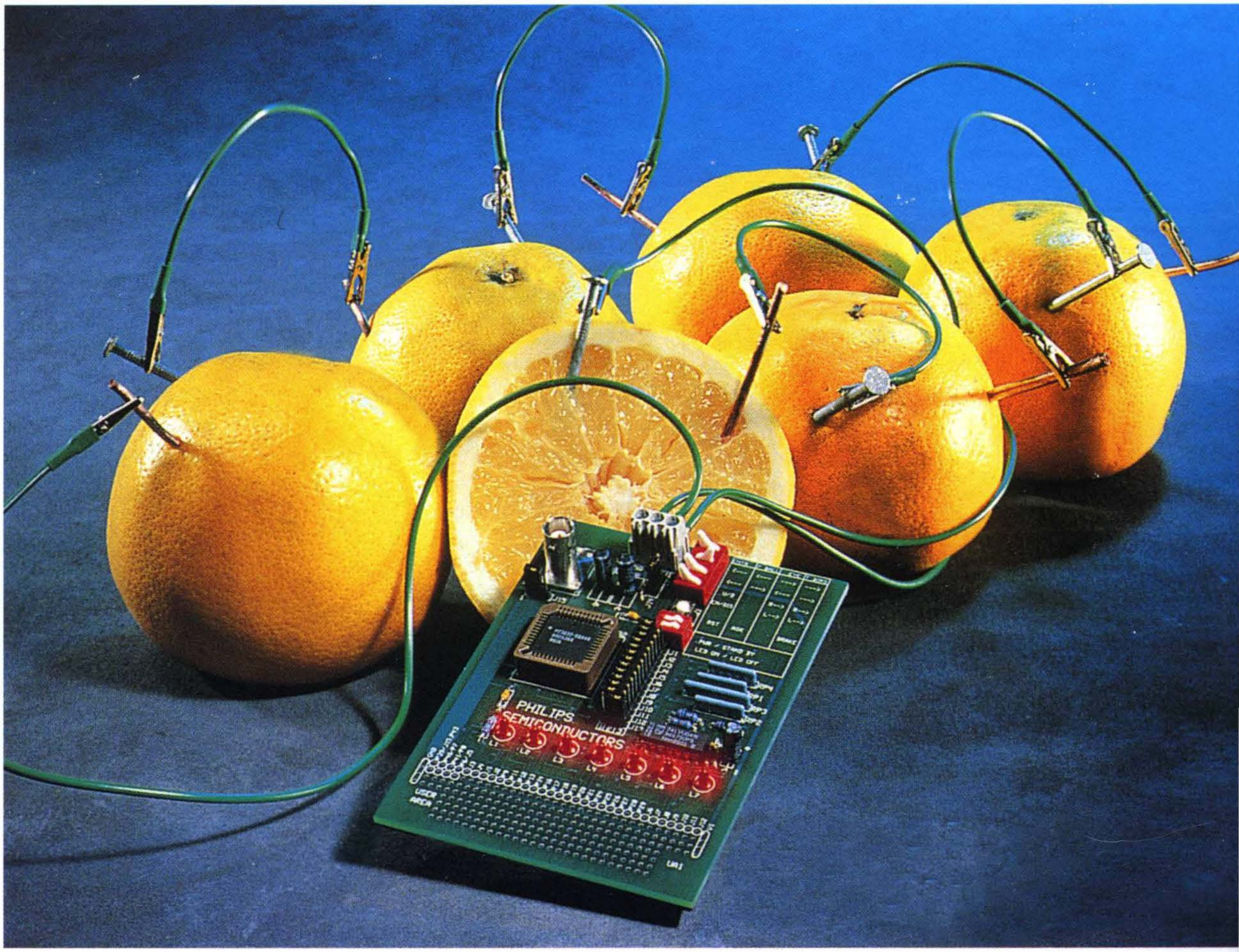
IC27 Complex Programmable Logic Devices 1997

INTEGRATED CIRCUITS

Complex Programmable Logic Devices



1997

Data Handbook IC27
CD-ROM included

Let's make things better.

Philips
Semiconductors

PHILIPS

Complex Programmable Logic Devices (CPLDs)

CONTENTS

DEFINITIONS

Data Sheet Identification	Product Status	Definition (Note)
Objective Specification	Formative or in Design	This data sheet contains the design target or goal specifications for product development. Specifications may change in any manner without notice.
Preliminary Specification	Preproduction Product	This data sheet contains preliminary data, and supplementary data will be published at a later date. Philips Semiconductors reserves the right to make changes at any time without notice in order to improve design and supply the best possible product.
Product Specification	Full Production	This data sheet contains Final Specifications. Philips Semiconductors reserves the right to make changes at any time without notice, in order to improve design and supply the best possible product.
Short-form specification	—	The data in this specification is extracted from a full data sheet with the same type number and title. For detailed information see the relevant data sheet or data handbook.
Limiting values		
Limiting values given are in accordance with the Absolute Maximum Rating System (IEC 134). Stress above one or more of the limiting values may cause permanent damage to the device. These are stress ratings only and operation of the device at these or at any other conditions above those given in the Characteristics sections of the specification is not implied. Exposure to limiting values for extended periods may affect device reliability.		
Application information		
Applications that are described herein for any of these products are for illustrative purposes only. Philips Semiconductors makes no representation or warranty that such applications will be suitable for the specified use without further testing or modification		

LIFE SUPPORT APPLICATIONS

These products are not designed for use in life support appliances, devices, or systems where malfunction of these products can reasonably be expected to result in personal injury. Philips Semiconductors customers using or selling these products for use in such applications do so at their own risk and agree to fully indemnify Philips Semiconductors for any damages resulting from such improper use or sale.

DISCLAIMER

Philips Semiconductors reserves the right to make changes, without notice, in the products, including circuits, standard cells, and/or software, described or contained herein in order to improve design and/or performance. Philips Semiconductors assumes no responsibility or liability for the use of any of these products, conveys no license or title under any patent, copyright, or mask work right to these products, and makes no representations or warranties that these products are free from patent, copyright, or mask work right infringement, unless otherwise specified.

NOTE: Always check with your local Philips Semiconductors Sales Office to be certain that you have the latest data sheet(s) before completing a design.

Preface

Introducing the only high-speed, zero-power CPLD solution

Sustaining a competitive edge in the high-performance arena requires you to maximize speed, minimize power use, and deliver more functionality—for both portable and mainstream PLD applications.

Philips Semiconductors gives you this edge with a superior solution that combines, for the first time ever, high speed and zero power into a CPLD. Using exclusive Philips technologies, including eXtended Programmable Logic Array (XPLA™), Fast Zero Power (FZP™), and standard In-System Programmability (ISP™), Philips has created new CPLD families. The families consist of ten devices in two process technologies across fifteen package types. Each solution offers you higher levels of integration than the competition, without sacrificing performance.

To ensure an easy-to-use CPLD solution, Philips offers world class software and application support. To enhance the support package, key partnerships with Synario Design Automation, Minc, IsData, and others have been formed to ensure state-of-the-art design solutions that are easily accessible to the designer. Whether it be Synario, Minc, Cadence, Viewlogic, Mentor or Synopsys, Philips provides design flows that allow you to complete designs on schedule.

Philips delivers the broadest range of 3V/5V CPLD solutions across all densities

The XPLA architecture serves as the foundation for all Philips' CPLD families. Current Philips CPLDs accommodate applications demanding industry-standard 32, 64 and 128 macrocell densities. As the demand for faster, larger, more power-efficient applications grows, future XPLA product families will use a combination of low-power techniques and state-of-the-art processing to extend densities up to 1,000 macrocells and beyond.

Now you can maximize density without sacrificing speed

Philips' new XPLA provides an architecture that enables you to efficiently allocate logic for 100% utilization without sacrificing speed. The XPLA provides a fast path to every macrocell, which includes five dedicated PAL product terms. Plus the XPLA's PLA structure and predictable sharing access times make it the fastest and most flexible solution available. Because the XPLA allocates additional logic exactly where you need it, it delivers key benefits competitive devices cannot. Benefits include no stealing of logic from neighbor macrocells, no density loss when sharing logic, no lost logic due to product-term granularity problems, no slow feedback logic, and no refitting or routing problems.

Philips shatters the myth—zero power no longer means slow speeds

With propagation delays of just 8ns at 3V and 6ns at 5V, and standby currents under 100 microamps, Philips' new FZP technology stands alone in breaking the paradigm that zero power equals slow speeds. When compared to today's standard zero power devices, with propagation delays of 20–25ns, Philips' CPLDs with FZP boast a 250% to 300% faster operating speed. And because no turbo-bits or sleep-modes are used, these CPLDs offer zero static power and low dynamic power all the time.

TRADEMARK LIST

Altera 7000 series devices

CoolRunner™

FZP™

AutoLogic™

Cadence PIC Designer™

Concept™

Composer™

Design Architect™

DSL™

ISP™

Hewlett-Packard HP-UX 9.05

Intel

Intelliflow™

Leapfrog™

Mentor Design Architect

Mentor Graphics Idea Station®

PLDSynthesis II™

Quicksim II

SpeedWave™

Sun MicroSystems SunOS

Sun MicroSystems Solaris

Synario

Synergy™

Synopsys®

Verilog-XL™

ViewPLD®

ViewDraw®

ViewSynthesis®

ViewSim®

Windows®95

Windows NT®

XPLA™

™, ® Product and Company names are registered trademarks of their respective organizations.

Section 1
General Information

CONTENTS

CONTENTS

IC27: Complex Programmable Logic Devices (CPLDs)

Section 1 – General Information

Section 2 – Introduction

Section 3 – In–System Programming (ISP™)

Section 4 – CPLD Datasheets

Section 5 – Related products

Section 6 – Application notes

Section 7 – Package information

Package outlines

ADDITIONAL MATERIAL ON ENCLOSED CD-ROM ONLY:

Application Notes/Design Models

Device pinouts

Family Selection Guide

PHILIPS CooLRunner™ FAMILY SELECTION GUIDE – COMMERCIAL TEMP RANGE (0 TO +70°C)

Part Number	Package(s)	Pkg Designator	Macrocells	Gates	Is & I/Os	Buried Macro cells	t_{PD}	Iddq	Iddd @1MHz	Supply Voltage
PZ3032	44 pin PLCC 44 pin TQFP	A44 BC	32	1000	36	0	8, 10, 12	35μA	0.5mA	3.0 – 3.3V
PZ5032	44 pin PLCC 44 pin TQFP	A44 BC	32	1000	36	0	6, 7.5, 10	75μA	3mA	4.75 – 5.25V
PZ3064	44 pin PLCC 44 pin TQFP 68 pin PLCC 84 pin PLCC 100 pin PQFP	A44 BC A68 A84 BB1	64	2000	36 52 68	32 16 0	10, 12	50μA	1mA	3.0 – 3.3V
PZ5064	44 pin PLCC 44 pin TQFP 68 pin PLCC 84 pin PLCC 100 pin PQFP	A44 BC A68 A84 BB1	64	2000	36 52 68	32 16 0	7.5, 10	80μA	3mA	4.75 – 5.25 V
PZ3128	84 pin PLCC 100 pin PQFP 100 pin TQFP 128 pin LQFP 160 pin PQFP	A84 BB1 BP BE BB2	128	4000	68 84 100	64 48 32	10, 12, 15	60μA	2mA	3.0 – 3.3V
PZ5128	84 pin PLCC 100 pin PQFP 100 pin TQFP 128 pin LQFP 160 pin PQFP	A84 BB1 BP BE BB2	128	4000	68 84 100	64 48 32	7.5, 10, 12	100μA	5mA	4.75 – 5.25 V

Family Selection Guide

PHILIPS CooLRunner™ FAMILY SELECTION GUIDE – INDUSTRIAL TEMP RANGE (–40 TO +85°C)

Part Number	Package(s)	Pkg Designator	Macrocells	Gates	Is & I/Os	Buried Macro cells	t_{PD}	Iddq	Iddd @1MHz	Supply Voltage
PZ3032I	44 pin PLCC 44 pin TQFP	A44 BC	32	1000	36	0	10, 12	45µA	0.5mA	3.0 – 3.3V
PZ5032I	44 pin PLCC 44 pin TQFP	A44 BC	32	1000	36	0	7.5, 10	95µA	4mA	4.5 – 5.5V
PZ3064I	44 pin PLCC 44 pin TQFP 68 pin PLCC 84 pin PLCC 100 pin PQFP	A44 BC A68 A84 BB1	64	2000	36 52 68	32 16 0	12, 15	50µA	1mA	3.0 – 3.3V
PZ5064I	44 pin PLCC 44 pin TQFP 68 pin PLCC 84 pin PLCC 100 pin PQFP	A44 BC A68 A84 BB1	64	2000	36 52 68	32 16 0	10, 12	100µA	4mA	4.5 – 5.5 V
PZ3128I	84 pin PLCC 100 pin PQFP 100 pin TQFP 128 pin LQFP 160 pin PQFP	A84 BB1 BP BE BB2	128	4000	68 84 100	64 48 32	15, 20	75µA	3mA	3.0 – 3.3V
PZ5128I	84 pin PLCC 100 pin PQFP 100 pin TQFP 128 pin LQFP 160 pin PQFP	A84 BB1 BP BE BB2	128	4000	68 84 100	64 48 32	12, 15	125µA	6mA	4.5 – 5.5 V

Ordering Information

PLD PRODUCTS – CPLD

Example: PZ x yyy –(S) zz YYY

PZ = Philips Zero power

x = Supply Voltage
3 = 3.3V
5 = 5V

yyy = Macrocell Count

k = Operating Temperature/Architecture
– = Commercial, original architecture
I = Industrial, original architecture
C = Commercial, enhanced clocking
N = Industrial, enhanced clocking

S = Designates ISP (if present)

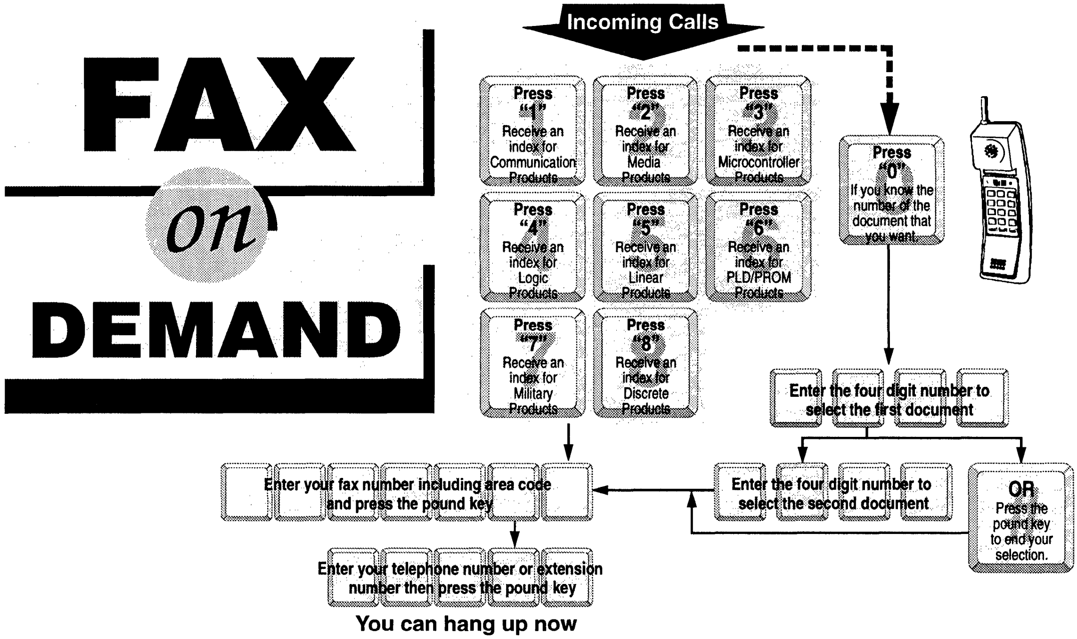
YYY = Package Code

A44	=	44-pin Plastic Leaded Chip Carrier (PLCC)
A68	=	68-pin Plastic Leaded Chip Carrier (PLCC)
A84	=	84-pin Plastic Leaded Chip Carrier (PLCC)
BB1	=	100-pin Plastic Quad Flat Pack (PQFP)
BB2	=	160-pin Plastic Quad Flat Pack (PQFP)
BC	=	44-pin Thin Quad Flat Pack (TQFP)
BE	=	128-pin Low profile Flat Pack (LQFP)
BP	=	100-pin Thin Quad Flat Pack (TQFP)

zz = Speed Grade (t_{PD})

6	=	6ns
7	=	7.5ns
8	=	8ns
10	=	10ns
12	=	12ns
15	=	15ns

FAX-on-DEMAND System


FAX
on
DEMAND
Incoming Calls
Press "1" Receive an index for Communication Products
Press "2" Receive an index for Media Products
Press "3" Receive an index for Microcontroller Products
Press "4" Receive an index for Logic Products
Press "5" Receive an index for Linear Products
Press "6" Receive an index for PLD/PROM Products
Press "7" Receive an index for Military Products
Press "8" Receive an index for Discrete Products
Press "0" If you know the number of the document that you want
Enter the four digit number to select the first document
Enter the four digit number to select the second document
OR Press the pound key to end your selection.
Enter your fax number including area code and press the pound key
Enter your telephone number or extension number then press the pound key
You can hang up now

What is it?

The FAX-on-DEMAND system is a computer facsimile system that allows customers to receive selected documents by fax automatically.

How does it work?

To order a document, you simply enter the document number. This number can be obtained by asking for an index of available documents to be faxed to you the first time you call the system.

Our system has a selection of the latest product data sheets from Philips with varying page counts. As you know, it takes approximately one minute to FAX one page. This isn't bad if the number of pages is less than 10. But if the document is 37 pages long, be ready for a long transmission!

Philips Semiconductors also maintains product information on the World-Wide Web. Our home page can be located at:
http://www.semiconductors.philips.com

Who do I contact if I have a question about FAX-on-DEMAND?

Contact your local Philips sales office.

FAX-on-DEMAND phone numbers:

England (United Kingdom, Ireland)	44-181-730-5020
France	33-1-40-99-60-60
Italy	39-167-295502
North America	1-800-282-2000

Locations soon to be in operation:

Hong Kong
Japan
The Netherlands

CPLD internet and support access

INTERNET ACCESS

Philips Semiconductors World Wide Web:

http://www.semiconductors.philips.com

Philips CPLD World Wide Web:

http://www.coolpld.com

Email CPLD Support Address:

coolpld@abq.sc.philips.com

ELECTROSTATIC CHARGES

Electrostatic charges can exist in many things; for example, man-made-fibre clothing, moving machinery, objects with air blowing across them, plastic storage bins, sheets of paper stored in plastic envelopes, paper from electrostatic copying machines, and people. The charges are caused by friction between two surfaces, at least one of which is non-conductive. The magnitude and polarity of the charges depend on the different affinities for electrons of the two materials rubbing together, the friction force and the humidity of the surrounding air.

Electrostatic discharge is the transfer of an electrostatic charge between bodies at different potentials and occurs with direct contact or when induced by an electrostatic field. All of our MOS devices are internally protected against electrostatic discharge but they **can** be damaged if the following precautions are not taken.

WORK STATION

Figure 1 shows a working area suitable for safely handling electrostatic sensitive devices. It has a work bench, the surface of which is conductive or covered by an antistatic sheet. Typical resistivity for the bench surface is between 1 and 500 kΩ per cm^2. The floor should also be covered with antistatic material. The following precautions should be observed:

- Persons at a work bench should be earthed via a wrist strap and a resistor
- All mains-powered electrical equipment should be connected via an earth leakage switch
- Equipment cases should be earthed
- Relative humidity should be maintained between 50 and 65%
- An ionizer should be used to neutralize objects with immobile static charges

RECEIPT AND STORAGE

MOS devices are packed for dispatch in antistatic/conductive containers, usually boxes, tubes or blister tape. The fact that the contents are sensitive to electrostatic discharge is shown by warning labels on both primary and secondary packing.

The devices should be kept in their original packing whilst in storage. If a bulk container is partially unpacked, the unpacking should be performed at a protected work station. Any MOS devices that are stored temporarily should be packed in conductive or antistatic packing or carriers.

ASSEMBLY

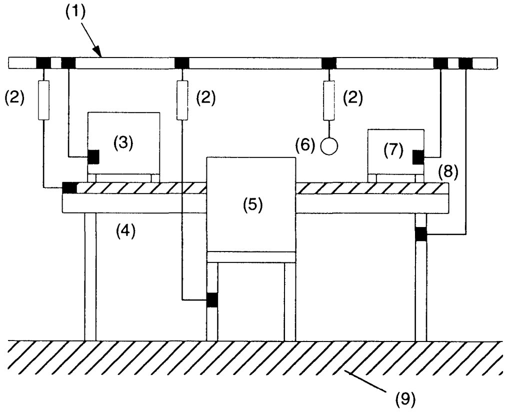
MOS devices must be removed from their protective packing with earthed component pincers or short-circuit clips. Short-circuit clips must remain in place during mounting, soldering and cleansing/drying processes. Do not remove more devices from the storage packing than are needed at any one time. Production/assembly documents should state that the product contains electrostatic sensitive devices and that special precautions need to be taken.

During assembly, endure that the MOS devices are the last of the components to be mounted and that this is done at a protected work station.

All tools used during assembly, including soldering tools and solder baths, must be earthed. All hand tools should be of conductive or antistatic material and, where possible, should not be insulated.

Measuring and testing of completed circuit boards must be done at a protected work station. Place the soldered side of the circuit board on conductive or antistatic foam and remove the short-circuit clips. Remove the circuit board from the foam, holding the board only at the edges. Make sure the circuit board does not touch the conductive surface of the work bench. After testing, replace the circuit board on the conductive foam to await packing.

Assembled circuit boards containing MOS devices should be handled in the same way a unmounted MOS devices. they should also carry waning labels and be packed in conductive or antistatic packing.



(1) Earthing rail.
(2) Resistor (500 kΩ ± 10%, 0.5 W).
(3) Ionizer.
(4) Work bench.
(5) Chair
(6) Wrist strap.
(7) Electrical equipment.
(8), (9) Conductive surface/antistatic sheet.

Figure 1. Protected work station

TOTAL QUALITY MANAGEMENT

Philips Semiconductors is a Quality Company, renowned for the high quality of our products and service. We keep alive this tradition by constantly aiming towards one ultimate standard, that of zero defects. This aim is guided by our Total Quality Management (TQM) system, the basis of which is described in the following paragraphs.

Quality assurance

Based on ISO 9000 standards, customer standards such as Ford TQE and IBM MDQ. Our factories are certified to ISO 9000 by external inspectorates.

Partnerships with customers

PPM co-operations, design-in agreements, ship-to-stock, just-in-time and self-qualification programmes, and application support.

Partnerships with suppliers

Ship-to-stock, statistical process control and ISO 9000 audits.

Quality improvement programme

Continuous process and system improvement, design improvement, complete use of statistical process control, realization of our final objective of zero defects, and logistics improvement by ship-to-stock and just-in-time agreements.

ADVANCED QUALITY PLANNING

During the design and development of new products and processes, quality is built-in by advanced quality planning. Through failure-mode-and-effect analysis the critical parameters are detected and measures taken to ensure good performance on these parameters. The capability of process steps is also planned in this phase.

PRODUCT CONFORMANCE

The assurance of product conformance is an integral part of our quality assurance (QA) practice. This is achieved by:

- Incoming material management through partnerships with suppliers.
- In-line quality assurance to monitor process reproducibility during manufacture and initiate any necessary corrective action. Critical process steps are 100% under statistical process control.
- Acceptance tests on finished products to verify conformance with the device specification. The test results are used for quality feedback and corrective actions. The inspection and test requirements are detailed in the general quality specifications.
- Periodic inspections to monitor and measure the conformance of products.

PRODUCT RELIABILITY

With the increasing complexity of Original Equipment Manufacturer (OEM) equipment, components reliability must be extremely high. Our research laboratories and development departments study the failure mechanisms of semiconductors. Their studies result in design rules and process optimization for the highest built-in product reliability. Highly accelerated tests are applied to the product reliability evaluation. Rejects from reliability tests and from customer complaints are submitted to failure analysis, to result in corrective action.

CUSTOMER RESPONSES

Our quality improvement depends on joint action with our customer. We need our customer's inputs and we invite constructive comments on all aspects of our performance. Please contact our local sales representative.

RECOGNITION

The high quality of our products and services is demonstrated by many Quality Awards granted by major customers and international organizations.

Section 2
Introduction

CONTENTS

XPLA™ ARCHITECTURE

Figure 1 gives a high level block diagram of the XPLA™ architecture. The XPLA™ architecture consist of Logic Blocks which are interconnected by a Zero-power Interconnect Array (ZIA). The ZIA is a virtual crosspoint switch. Each Logic Block is essentially a 36V16 device with 36 inputs from the ZIA and 16 macrocells. Each Logic Block also provides 32 ZIA feedback paths from the macrocells and I/O pins. The number of Logic Blocks contained within a device determines the macrocell count of the device. For example, devices containing 2, 4, and 8 Logic Blocks are 32, 64, and 128 macrocell devices, respectively.

From this point of view, this architecture looks like many other CPLD architectures. What makes the CoolRunner™ family unique is what's inside each Logic Block and the design technique used to implement these Logic Blocks. The contents of the Logic Block will be described next.

Logic Block Architecture

Figure 2 illustrates the Logic Block Architecture. Each Logic Block contains Control Terms, a PAL Array, a PLA Array, and 16 macrocells. The 6 Control Terms can individually be configured as either AND or SUM product terms and are used to control the preset/reset and output enables of the 16 macrocell's flip-flops. The PAL Array consists of a programmable AND array with a fixed OR array while the PLA array consist of a programmable AND array with a programmable OR array. The PAL array provides a high speed path through the array while the PLA array provides increased product term density.

Each macrocell has 5 dedicated product terms from the PAL array. If a macrocell needs more than 5 product terms, it simply gets the additional product terms from the PLA array. The PLA array consists of 32 product terms which are available for use by all 16 macrocells. For the 5V PZ5032 the additional propagation delay incurred by a macrocell using 1 or all 32 PLA product terms is just 2ns. So the total pin-to-pin t_{PD} for the PZ5032 using 6 to 37 product terms is 8ns (6ns for the PAL + 2ns for the PLA).

The XPLA™ architecture is very accommodating for implementing last minute design changes. In fact, 16 million worst case designs (designs which used all of the I/O Pins and all of the Macrocells) were implemented in the PZ5032 with fixed pins & macrocells and all but 30 designs were able to route. Therefore 99.998% of these worst case designs were able to route with the pins fixed after the design was changed.

The reason why the XPLA™ architecture accommodates last minute design changes is because the PAL product terms are dedicated to a given macrocell and in addition there is a free pool of 32 PLA product terms which can be used by any of the 16 macrocells. If a macrocell uses less than 5 product terms and the design change requires a total of 5 product terms, the design is guaranteed to fit because the 5 PAL product terms are dedicated to each macrocell. There is no borrowing between macrocells. Borrowing is a nice feature until the macrocell whose product terms were borrowed wants its product terms back because of a last minute design change. If a design change requires more than 5 product terms, unused PLA product terms are used by the macrocell. In an average design, less than 20 PLA product terms are used so there are typically 12 PLA product terms available to implement last minute design changes.

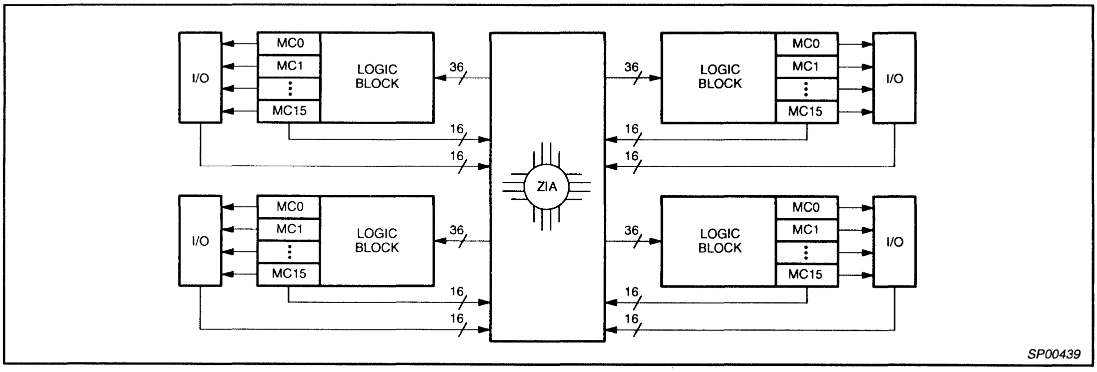


Figure 1. XPLA™ Block Diagram

XPLA™ architecture

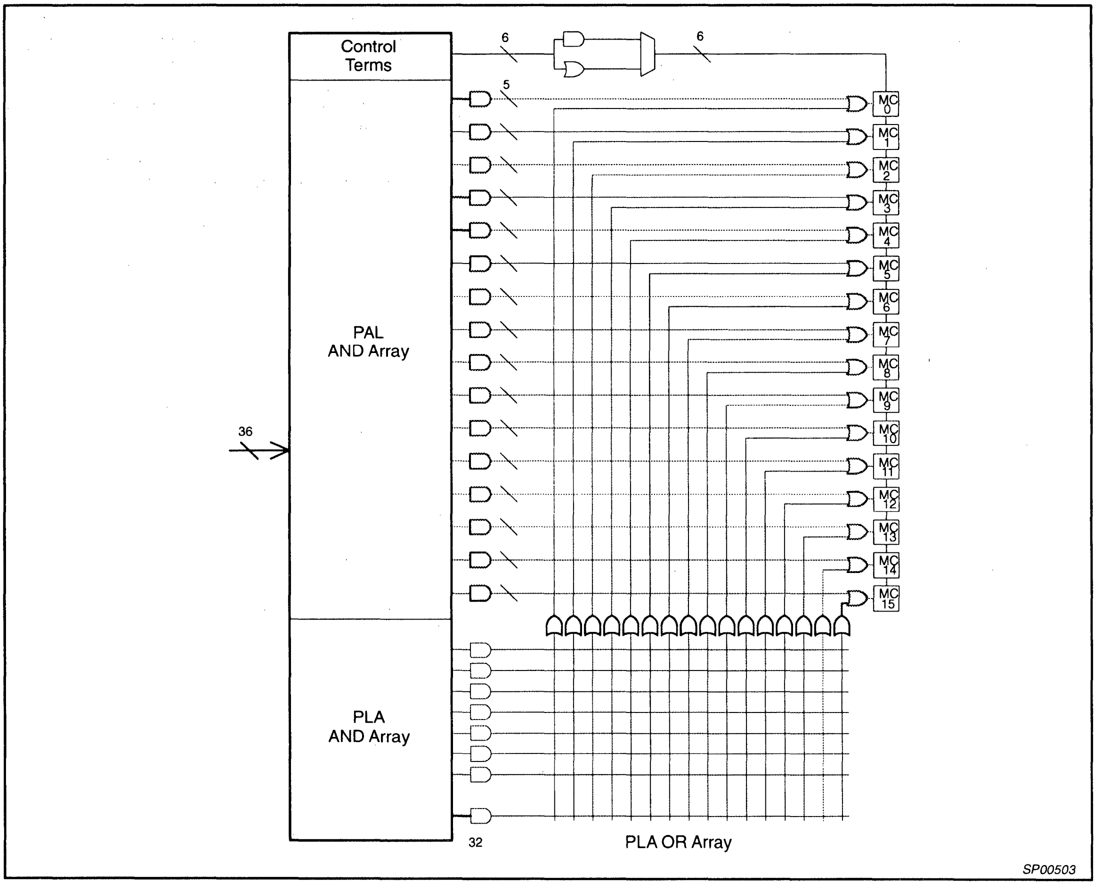


Figure 2. Logic Block Architecture

Macrocell Configuration

Figure 3 shows the architecture of the macrocell used in the CoolRunner™ family. The macrocell consists of a flip-flop which can be configured as either a D or T type. A D-Type flip-flop is generally more useful for implementing state machines and data buffering. A T-Type flip-flop is generally more useful in implementing counters. All CoolRunner™ family members provide both synchronous and asynchronous clocking and provide the ability to clock off either the falling or rising edges of these clocks. These devices are designed such that the skew between the rising and falling edges of a clock are minimized for clocking integrity. There are 2 clocks (CLK0 and CLK1) available on the PZ3032/PZ5032 devices and 4 clocks (CLK0 through CLK3) available in the PZ3064/PZ5064 and PZ3128/PZ5128 devices. Clock 0 (CLK0) in each of these devices is designated as the "synchronous" clock and must be driven by an external source. Clocks 1, 2, and 3 (CLK1, CLK2, and CLK3) can either be used as a synchronous clock (driven by an external source) or as an asynchronous clock (driven by a macrocell equation).

Two of the control terms (CT0 and CT1) are used to control the Preset/Reset of the macrocell's flip-flop. The Preset/Reset feature for each macrocell can also be disabled. The other 4 control terms (CT2-CT5) can be used to control the Output Enable of the macrocell's Output Buffers. The reason why there are so many control terms dedicated for the output enable of the macrocell is to insure that all CoolRunner™ devices are PCI compliant. The macrocell's output buffers can also be always enabled or disabled. All CoolRunner™ devices also provide a Global Three-State (GTS*) pin which, when pulled low, will three-state all the outputs of the device. This pin is provided to support "In-Circuit Testing" or "Bed-of Nails Testing".

There are two feedback paths to the ZIA; one from the macrocell and one from the I/O pin. The ZIA feedback path before the output buffer is the macrocell feedback path while the ZIA feedback path after the output buffer is the I/O pin ZIA path. When the macrocell is used as an output, the output buffer is enabled and the macrocell feedback path can be used to feedback the logic implemented in the macrocell. When the I/O pin is used as an input, the output buffer will be three-stated and the input signal will be fed into the ZIA via the I/O feedback path and the logic implemented in the buried macrocell can be fed back to the ZIA via the macrocell feedback path.

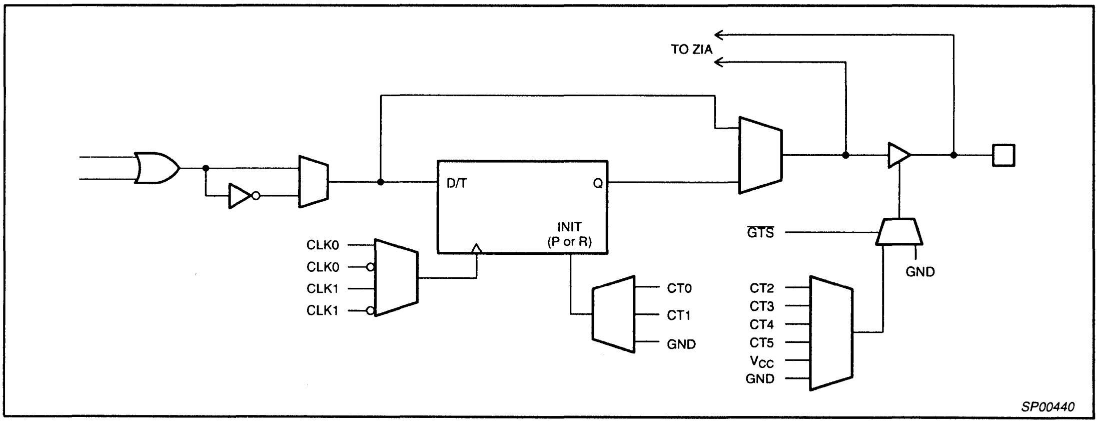


Figure 3. Macrocell Architecture

How to use the programmable, Global 3-State feature on CoolRunner™ devices

All CoolRunner™ devices include a Global 3-State (GTS) feature which supports "In-Circuit-Testing" or "Bed-of-Nails-Testing". The GTS feature is made available via a Global Tri-State pin (GTS*) which, when driven to a Low logic level, will 3-State all the outputs of the device. The GTS feature is optionally invoked as part of the user's design, and is provided through a dedicated pin on each CoolRunner™ device (please refer to the individual device data sheets). The default condition is that the GTS feature is not invoked, and that the related pin is instead available as a dedicated input pin (designated IN2 on all CoolRunner™ device packages).

The GTS feature is invoked by setting the device under test property in the user's design file. In the XPLA Designer software environment, this is achieved by including the following property statement in the header section of the design:

XPLA Property 'dut on';

When this property is set, the GTS feature can also be simulated in the XPLA Designer environment. After including the 'dut on' property in the design, the RESERVED_DUT signal automatically appears in the simulator signal list the first time a .SCL (Simulation Control Language) is created. If a new or subsequent .SCL file is created, the user must add a new signal and select RESERVED_DUT as its name to enable simulation of this feature. In either case, the RESERVED_DUT signal may be driven in the simulation by any waveform, similar to any other input. The difference, of course, is that all outputs are 3-stated whenever the RESERVED_DUT signal is driven to a Low logic level.

How to use synchronous and asynchronous clocks in CoolRunner™ devices

All CoolRunner™ devices provide multiple clock inputs, and support both synchronous and asynchronous clocking of the devices' internal registers. Clock inputs are associated with dedicated pins on CoolRunner™ device packages (please refer to the individual device data sheets), and the number of available clocks depends upon device density.

The clock inputs also have well-defined capabilities, depending upon whether their associated pin is a dedicated input or a general purpose I/O. CLK0 on any CoolRunner™ device is always associated with a dedicated input which, when used as a clock input, always serves as a synchronous clock driven solely by an externally-provided signal.

All other clocks (e.g., CLK1, CLK2, CLK3) are associated with a general purpose I/O pin and may be used to implement either synchronous or asynchronous (i.e., complex, or term-based) clocks. When used as a clock signal driven solely by an externally-provided input, these clocks perform similar to CLK0. Since these clocks are associated with a general purpose I/O, this isolates the related macrocell, although the macrocell's feedback path is still available, thus enabling use of the macrocell as a "buried" node or logic path.

When using CLK1, CLK2, or CLK3 to implement asynchronous clocks, there are some special considerations. In this case any or all I/O pins and feedback paths may be used to form the asynchronous clock. However, the clock must ultimately be driven by a macrocell associated with one of the general purpose I/O pins specified for clocking per the device data sheet. Because a specific macrocell ultimately drives the asynchronous clock, the associated general purpose I/O pin is no longer available for use (except as an external monitoring or distribution point for the asynchronous clock). Also, performance for asynchronous clocking is variable depending on the specific application and must be determined through simulation or analysis by the user. This is because the levels of logic complexity and number of feedback paths incurred in forming the asynchronous clock determine the delay in the clock path.

Please note that all of the clock signals in a CoolRunner™ device are available to all of the macrocells in that device. A multiplexer is associated with each macrocell for the purpose of providing any of the available clock signals to the macrocell's register element. Furthermore, clock polarity can be selected for each macrocell, thus allowing any macrocell's register element to be clocked from either the rising or falling edge of any clock signal in the device.

How to use Output Enables in CoolRunner™ devices

All CoolRunner™ devices provide the ability to control each general purpose I/O pin through the use of an output enable (OE) signal. OE signals allow for the 3-stating of an output pin in bus applications, and also allow for the implementation of bi-directional pins in the CoolRunner™ devices. Within each logic block (i.e., group of 16 macrocells), four control signals are provided to support OE generation. Because these control signals are provided at the logic block level, each logic block can independently generate OE control for its associated outputs.

Each of these four control signals are provided to the input of a multiplexer that is associated with each output buffer control, thus allowing each general purpose I/O pin in the logic block to have OE control based on one of these four control signals. Each of these four control signals may directly be either a sum term or product term of any or all of the 36 inputs into the logic (the 36 inputs into the logic block may be driven directly by external signals and/or by feedback paths from logic generated in the device). However, these four control signals cannot directly be a sum of products. For example:

A # B # C # D #	is a sum term, which is directly supported
A & B & C & D &	is a product term, which is directly supported
(A&B) # (C&D) #	is a sum of products term, which is **NOT** directly supported

Complex OE signals that are based on a sum of products term must use a 'buried' node that is not collapsed during the design compilation. The 'buried' node is driven by the sum of products equation, and the OE signal is assigned to be driven by the node.

How to use Presets and Resets in CoolRunner™ devices

All CoolRunner™ devices support asynchronous Preset/Reset control for the register elements associated with each macrocell. Within each logic block, two control signals are provided for Preset/Reset control. Like OE signals (described above), the Preset/Reset control signals may be a sum term or a product term of any/all of the inputs into the logic block.

Each of the two control signals drive multiplexers that are associated with each register element. The multiplexer, in turn, controls either a Preset or Reset function defined for the register. Please note that the user must select between asynchronous Preset and asynchronous Reset functionality for a register element; it is not possible to implement both asynchronous Preset and asynchronous Reset for the same register.

Synchronous Preset/Reset may be synthesized for each register element. This is achieved by 'gating' the logic input to a register element with an appropriate control signal, as specified in the logic design. In the XPLA Designer environment synchronous preset and reset dot extensions are available to automatically synthesize the required structures. For synchronous preset the .CLR or .SR extension can be utilized and for synchronous reset the .SET or .SP extensions may be used.

How to implement Clock Enable signals in CoolRunner™ devices

Clock Enable (CE) signals are indirectly supported in CoolRunner™ devices through synthesis. To implement a CE signal, an appropriate control is 'gated' with both the logic generated to drive the register element and the register's output feedback. In the XPLA Designer software environment, a .CE extension is provided for specifying a CE signal in the design and synthesis is automatic, such that it is not necessary to explicitly specify the 'gating'.

In other environments, it may be necessary to explicity implement the CE functionality. For a register element driven by the logic signal DAT_IN to produce the output signal DAT_OUT, the CE signal CLKEN is used in the definition of DAT_OUT as follows to implement the clock enable functionality:

```
DAT_OUT := (DAT_IN & CLKEN) # (!CLKEN & DAT_OUT.q)
```

where ':=' indicates that DAT_OUT is a registered signal, the '.q' extension indicates the feedback path from the register, and the signal CLKEN is a clock enable signal driven by external signals, internally-generated logic, or both.

How to determine output states upon power up in CoolRunner™ devices

When a CoolRunner™ device is powered-up, all output buffers are disabled, so that all outputs are 3-stated. The delay from valid V_{DD} to valid reset is specified in all CoolRunner™ datasheets as t_{INIT} with a maximum delay of 50 microseconds (μs). After t_{INIT}, all registered outputs are reset to a low logic level, and all combinatorial outputs are resolved to the appropriate state as determined by the inputs and/or internally-generated logic from which they are formed. However, please note that only a maximum delay is specified for t_{INIT} and no minimum delay is guaranteed. If external input signals are still at undeterminable (e.g., 3-state) logic states after the CoolRunner™ device has transitioned through its power-up reset sequence, dependent combinatorial outputs will be unknown.

PLA Product Term Sharing

Another feature offered by the XPLA™ architecture, which cannot be offered by other competing architectures, is product term sharing. In address decode circuits, some state machines, and other types of designs there are product terms which are common to a number of macrocells. The XPLA™ architecture allows sharing of PLA product terms between macrocells as shown in Figure 4. In this example, it shows one PLA product term being shared by two macrocells. In this case, there is "effectively" 33 PLA product terms because one of them is shared between two macrocells. PLA product term sharing increases the "effective" density of the device and allows larger designs to fit in the same macrocell count device. If needed, all 16 macrocells could share all 32 PLA product terms.

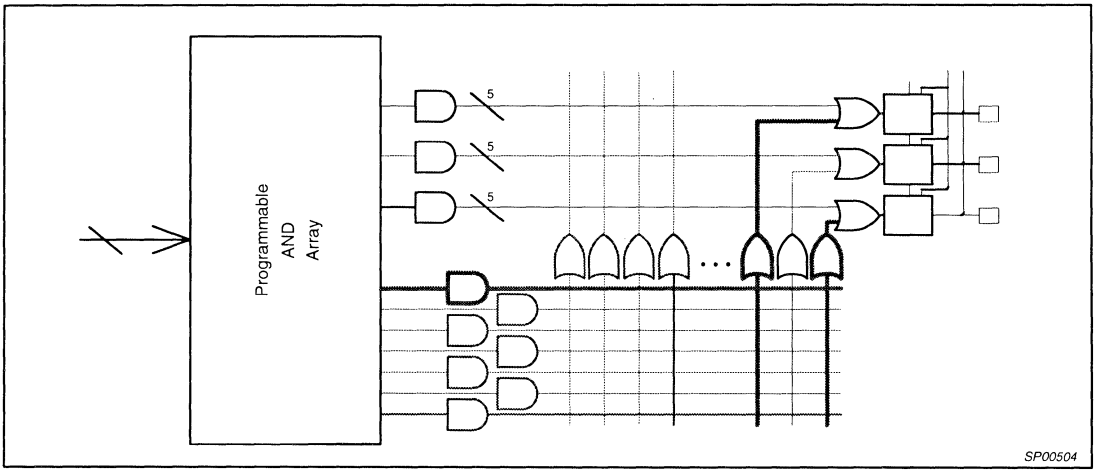


Figure 4.

Simple Timing Model

Figure 5 shows the CoolRunner™ Timing Model. As one can see from this illustration, the CoolRunner™ timing model looks very much like a 22V10 timing model in that there are three main timing parameters including t_{PD}, t_{SU}, and t_{CO}. In other competing architectures, the user may be able to fit the design into the CPLD but is not sure whether system timing requirements can be met until after the design has been fit into the device. This is because the timing models of competing architectures are very complex and include such things as: timing dependencies on the number of parallel expanders borrowed, sharable expanders, varying number of X and Y routing channels used, etc. In the XPLA™ architecture, the user knows up front whether the design will meet system timing requirements. This is due to the simplicity of the timing model. For example, in the PZ5032 device, the user knows up front that if a given output uses 5 product terms or less, the t_{PD} = 6ns, the t_{SU} = 4ns, and the t_{CO} = 5.5ns. If an output is using 6 to 37 product terms, an additional 2ns is added to the t_{PD} and t_{SU} timing parameters to account for the time to propagate through the PLA array—this is the only variation in timing that exists when using the XPLA™ architecture!

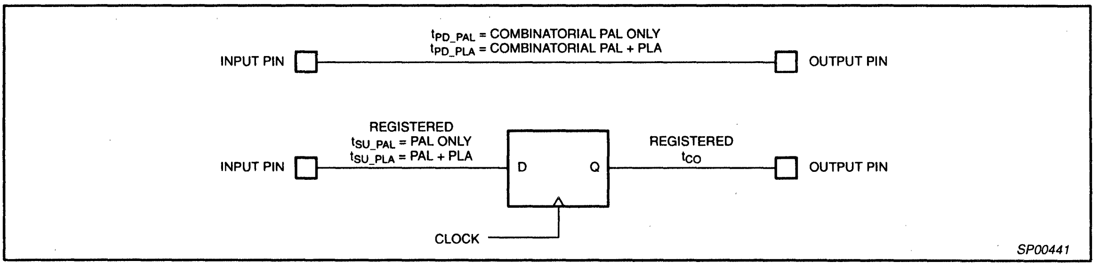


Figure 5. CoolRunner™ Timing Model

Fast Zero Power (FZP™)

In CPLD architectures, the methods implemented to propagate logic-level transitions in the product term array are derived from the original bipolar simple PLD devices. Within a CPLD, each product term in the array is "fed" by all of the inputs into the logic block (see Figure 1). The number of inputs to the logic block vary by CPLD supplier, but are generally 1.5-to-2.5 times the number of macrocells in the logic block. The most commonly implemented logic block size in existing CPLDs is the 36V16, meaning that there are 36 inputs into a logic block with 16 macrocells. Since all CPLD architectures make both true and complement forms of the logic block inputs accessible to the product terms, the number of capacitive loads on the product term "word line" becomes significant and can be modeled with an equivalent total capacitance of 2 pf.

Logic block inputs (and their complement) are connected to the product term "word line" via pass gates and the "word line" is biased with a "read" current of roughly 0.2 mA that can be modeled as the source voltage through a resistor. The time required to propagate a logic-level transition on the "word line" becomes a function of the 2 pf total capacitance of all the pass gates, the "read" current supplied for biasing the "word line", and the voltage differential that defines a logic-level transition.

Existing CPLDs implement sense amplifiers at the end of each "word line" to achieve fast propagation delays. These sense amplifiers operate in the linear region, and ensure fast propagation times by amplifying a 100 mV increase to the 1 Volt bias voltage on the "word line" such that it represents a full CMOS voltage swing. Using the equation I = C(dv/dt), the time required to increase the "word line" voltage can be calculated as follows:

dt = (C dv)/I = (2 pf) * (0.1 v) / 0.2 mA = 1 ns.

The sense amplifier itself contributes another 1 ns to the delay, and requires a current of about 0.05 mA. Because this is only a portion of the CPLD's total propagation delay (t_{PD}), it becomes necessary to reduce the "word line" delay to ensure high-performance for the CPLD. This is achieved by increasing the "read" current, as indicated in the calculations previously discussed.

The benefits of the sense amplifier are clear—a full voltage swing can be realized in a very short time by supplying the maximum current that the sense amplifier accommodates. In lower-density CPLDs (i.e., less than 128 macrocells), this maximum current is usually not prohibitive from a thermal dissipation and a supply voltage standpoint. As an alternative, consider what would happen if the sense amplifier was not used and it became necessary to increase the voltage by 4 volts to realize a full CMOS swing from the 1 volt bias. With a 0.2 mA "read" current, the time required to realize a 4 volt increase is calculated as follows:

dt = (2 pf) * (4.0 v) / 0.2 mA = 40 ns.

To get back to the 1 ns performance of the sense amplifier, the "read" current would have to be increased to 8 mA for each product term. This would translate to a standby current of more than 5 amps for a device with 5 product terms per macrocell, 16 macrocells per logic block, and 8 logic blocks per device (i.e., a common 128 macrocell CPLD).

The benefits of the sense amplifier are not free, though. As discussed above, each product term requires a standby current of 0.25 mA. This translates into a total standby current of 160 mA for the common 128 macrocell CPLD. To operate a 16-bit counter in every logic block at f_{cnt} = 50 MHz., the current requirement for this device increases to about 240 mA. Even though these currents are a vast improvement over not using a sense amplifier, they are still considerably high for today's digital designs—especially when the end-product is portable or battery-powered. When it is necessary to reduce power in these devices, the "read" current must be limited. This is achieved during device programming by setting the device to low-power, or "non-turbo" mode. However, as indicated in the equations above, reducing the "read" current causes an increase in the time it takes to propagate the logic-level transition through the product term. The end result is a significant decrease in the overall performance of the CPLD.

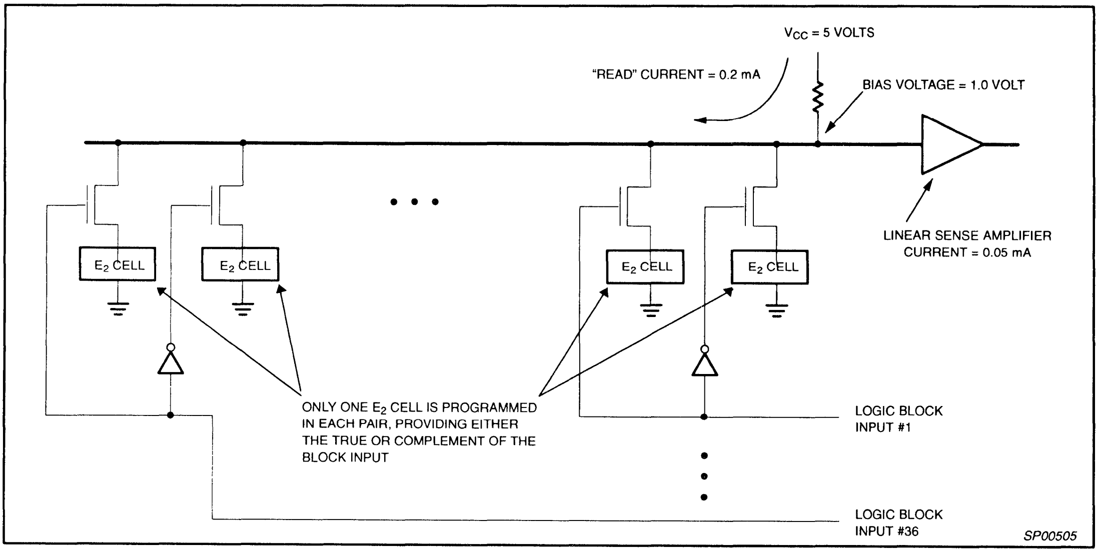
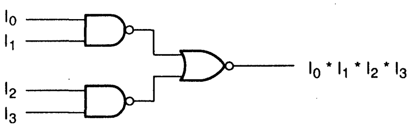


Figure 1. Product Term in Existing CPLDs

Fast Zero Power (FZP™)

The Philips FZP™ design technique takes a new and innovative approach to implementing the product term array.

Instead of employing a bipolar design technique (i.e., sense amplifiers), Philips instead becomes the first CPLD supplier to take a true CMOS design approach. The result is the first TotalCMOS™ CPLD; i.e., a CPLD that is CMOS both in process technology and design technique. In the FZP™ design technique, represented in Figure 2, the product term array is implemented by cascading the logic in a tree of full-CMOS gates. These full-CMOS gates switch in 200 picoseconds (ps), and are cascaded in a way that achieves speed performance comparable to the sense amplifier approach in existing CPLDs.



A four-input 'AND' function demonstrates how the product term is implemented with full-CMOS gates in the Philips Semiconductors' CPLDs.

SM00226

Figure 2. Representation of a Product Term using the Philips FZP™ Design Technique

Cascading the gates distributes the capacitance associated with the true and complement logic block inputs, so that this capacitance is no longer lumped on a single node. Furthermore, the switching current behaves in a manner similar to that of random logic in a gate array. The static current for each gate is incredibly small—about 1 pico-amp (pA). The total instantaneous dynamic current is also small, since only the gates in one path of the tree switch, and these gates switch in succession rather than all at the same time.

The advantages of FZP™ are numerous. Total standby current for these CPLDs is under 100 micro-amps—at least 1000 times less than that exhibited by CPLDs that use sense amplifiers. Dynamic power is also decreased relative to existing CPLDs—by as much as 70% for a device whose logic is fully-populated with 16-bit counters operating at 50 MHz. Best of all, these power savings are realized with no impact on performance and no externally-controlled provisions, such as power-down circuitry. Because power consumption is so low, TQFP packaging options can now be offered that were once considered impossible due to thermal limits. Also, CPLDs can now rival the gate densities that were previously attained only by FPGA's and ASIC's. As a result, FZP™ is an enabling technology for CPLDs since it makes their use possible in applications where their high-power-consumption and/or low-performance at low-power was previously prohibitive.

The disadvantages of FZP™ are trivial. It is possible to achieve slightly faster data paths by supplying high currents to sense amplifiers. This path is only a portion of the CPLDs total Tpd, though, such that the difference between a sense amplifier's speed performance and the speed of the CMOS gate chain becomes negligible. The CMOS gate chain also requires more die area in the core than sense amplifiers. In fact, this would likely preclude a full-CMOS gate approach in the older, two-layer metal processes used to manufacture most existing CPLDs. However, Philips has overcome this obstacle by manufacturing their CPLDs on a leading-edge, 0.5 micron, triple-layer metal process technology.

DEVELOPMENT SOFTWARE

Philips software strategy is based on three elements:

–Support for tools and design flows already in place at the customer site,

–Maintaining in-house expertise to facilitate optimum architecture and software interaction while adding valued utilities into the design process, and

–To offer cost effective solutions that enable users easy access to Philips silicon technologies.

By supporting tools that designers already own, Philips hopes that costs for both licensing and maintenance can be maintained at reasonable levels, and that designer productivity can be enhanced as minimal learning is required.

When considering the design process, it can be broken down into three elements: design creation, design verification, and design implementation. It is the Philips Semiconductors Programmable Product Group's belief that design creation tools are best handled by CAE companies whose charter it is to create the most useful software in completing this initial step. Philips supports the move to vendor independent design creation software and HDLs. Philips strategy involves focussing on design implementation software, so that design files can be efficiently and effectively implemented into Philips devices. In addition, Philips supports back annotation of delay information into software utilized for design verification.

The above strategy facilitates the need for partnerships, and Philips will continue to work with the CAE software industry to provide optimum, timely, and cost effective solutions. One example of a partnership is illustrated by Philips relationship with Minc Inc. to supply CPLD implementation software (i.e., fitters). As a result, Philips maintains redundant fitters for each device so that software support is always ensured for the design engineer.

Philips' CPLD devices are currently supported in the following design environments: Cadence, Mentor, Viewlogic, Synario, Minc, Synopsys via Minc, and Philips' own XPLA™ Designer on personal computer and Unix workstations. Detailed descriptions follow that outline the design flows and software tools available in these environments.

Philips/Cadence Design Environment

The Philips/MINC CPLD fitter is fully integrated into the Cadence PIC Designer™ software and the Cadence family of system design products. It is tightly integrated with both the Concept™ and Composer™ design entry systems, with Synergy™ VHDL and Verilog synthesis tools, and with Verilog-XL™ and Leapfrog™ VHDL simulation environments.

The Philips/Cadence design flow supports a top-down or mixed-level design methodology. Designs can be entered as schematics using Concept™ or Composer™, or in an industry standard hardware description languages (HDLs) including VHDL and Verilog HDL. Maximum flexibility is provided through the ability to enter designs as a mixture of schematic-based and HDL-based design entry methods.

PIC Designer™ provides the ability to simulate the programmable logic design before device selection and partitioning, and again after the implementation of the design. PIC Designer™ performs an optimization of the user's programmable logic, and automatically generates Verilog models for the blocks associated with HDL descriptions. These models contain unit delays for functional simulation of the programmable logic early in the design process. This makes it easier to detect and correct errors, and helps users ensure that their logic will function properly in their overall system design.

PIC Designer™ performs architecture-specific partitioning and fitting based on user-defined design constraints, to ensure the best possible design implementation. This includes automatic implementation of the design, with a resulting list of single or multiple device alternatives which meet the design constraints. After device fitting, PIC Designer™ automatically redraws fully annotated schematics. Designs can be re-simulated with final timing using Verilog or Leapfrog™ VHDL for a detailed analysis of system operation and performance.

System and Software Requirements

Cadence/Minc Software

- PIC Designer™
- Philips/MINC Fitter Option for PIC Designer™
- Synergy™ HDL synthesis
- Verilog or Leapfrog™ simulator

Recommended Operating Systems

PIC Designer™ has been tested with the following operating systems:

- Hewlett-Packard HP-UX 9.05
- Sun MicroSystems SunOS 4.1.4 (V4.1.3 compatible)
- Sun MicroSystems Solaris 2.4 (requires SunOS compatibility mode)

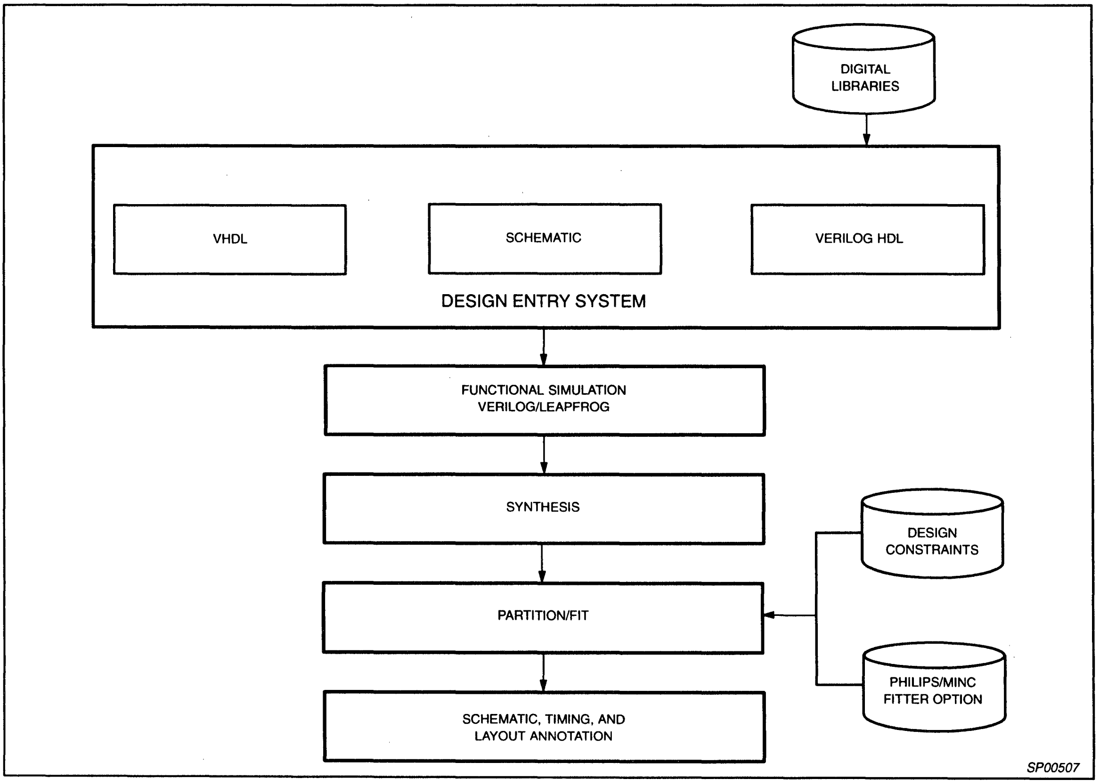


Figure 1.

Philips/Mentor Graphics Design Environment

The Philips/MINC CPLD fitter is fully integrated into the Mentor Graphics PLDSynthesis II™ (PLDS II) software. PLDS II offers high-level design entry, device-specific optimization, automatic device selection, and patented multiple-device partitioning. It also leverages the design and simulation capabilities of Mentor Graphics Idea Station® to provide a total programmable design environment.

The Philips/Mentor Graphics design flow supports a top-down or mixed-level design methodology. A wide variety of design entry methods allows the designer to quickly complete their design. Designs can be entered as TTL or GENLIB schematics using Mentor Graphics Design Architect, as an ASIC netlist using AutoLogic™, or in hardware description language (HDL) form using either AutoLogic™ VHDL or PLDS II's own hierarchical design language (DSL™). Maximum flexibility is provided through the ability to enter designs as a mixture of the various design entry methods.

PLDS II automatically builds functional simulation models, and supports simulation before device selection and synthesis, including VHDL functional simulation using QuickHDL. PLDS II performs automatic device selection based on user-defined engineering constraints to ensure the best possible design implementation. It also automatically performs partitioning, fitting, and pin mapping. These features, coupled with fully hierarchical synthesis for any combination of CPLDs, improves both cost-efficiency and time-to-market by facilitating fast and easy "What If" analysis for the design.

After device selection and synthesis, PLDS II generates JEDEC and Intel HEX files for programming the CPLD. It also automatically creates full documentation and physical schematics for the final implementation. These files are easily loaded into leading device models to assure convenient and accurate final simulation. Simulation support is available through Mentor Graphics QuickSim II using the Standard Timing Models that are included with PLDS II. The Standard Timing Models may also be used with Mentor Graphics QuickPath to perform static timing analysis.

System and Software Requirements
Mentor Graphics Software
- Idea Station™
- PLDSynthesis II™ Version 3.6
- Philips/MINC Fitter Option for PLDSynthesis II™

Recommended Operating Systems
PLDSynthesis II™ has been tested with the following operating systems:
- Hewlett-Packard HP-UX 9.05
- Sun MicroSystems SunOS 4.1.4 (V4.1.3 compatible)
- Sun MicroSystems Solaris 2.4 (requires SunOS compatibility mode)

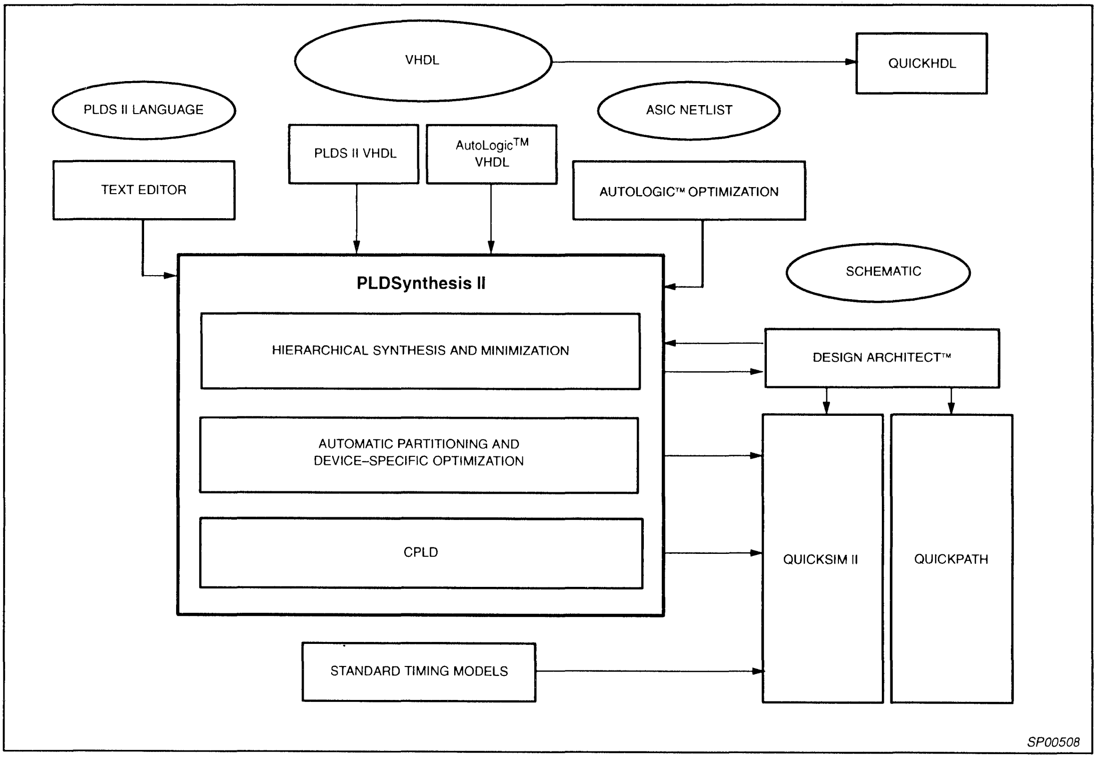


Figure 2.

Development software

Synario/Philips Design Environment

Synario offers a comprehensive suite of Windows-based tools for programmable logic design; incorporating best-of-class point tools that operate in a seamless design flow. The power behind Synario's integrated design system can be attributed to its Project Navigator. Its built-in knowledge of device architectures is leveraged each time a designer changes architectures, with an automatic reconfiguration of the design flow. After the designer selects a device, the Project Navigator presents the steps required to design, simulate, synthesize, and place-and-route the design. Synario users can move from entry to implementation in minutes, even when targeting an unfamiliar device architecture.

The value of Project Navigator increases for those designers transitioning to top-down design methodologies. HDLs, such as VHDL and Verilog, are well suited to handling the increasing complexity of CPLD and FPGA devices. The Project Navigator decreases the learning curve this transition creates, by supporting designers through each step in a pure HDL or mixed HDL-schematic based design process.

To support traditional or top-down design methods, Synario provides unsurpassed mixed-entry support along with functional and timing simulation options for both Verilog and VHDL users. Synthesis capabilities that interface to device-specific optimization and technology mapping are also available.

Synario is serious about good device support. Synario Device Kits include: schematic symbols; simulation models; logic synthesis and device-fitting technology; place-and-route software; device specific examples and on-line help. Designers can be assured, Synario Device Kits provide vendor-qualified tools tailored specifically to a chosen architecture.

Synario-Philips Device Kit incorporates everything the designer needs for a CoolRunner™ XPLA design: schematic symbols; functional and timing simulation models; Philips Semiconductors' XPLA Optimization and Fitting software; and process management, design examples and on-line help for Philips device families. It serves as the interface from within Synario's Project Navigator, offering designers a highly integrated front-to-back design solution tailored for Philips CPLDs.

System and Software Requirements

Synario Software:
- Synario Programmable IC Entry or Synario System Entry
- Synario VHDL, Verilog, or Verilog-Pro Simulators
- Synario ABEL, VHDL Synthesizers

Philips Software:
- Synario Philips Device Kit (Syn-XPLA-PR)

Operation Systems supported
- Windows 3.11, 95, NT 3.51or 4.0

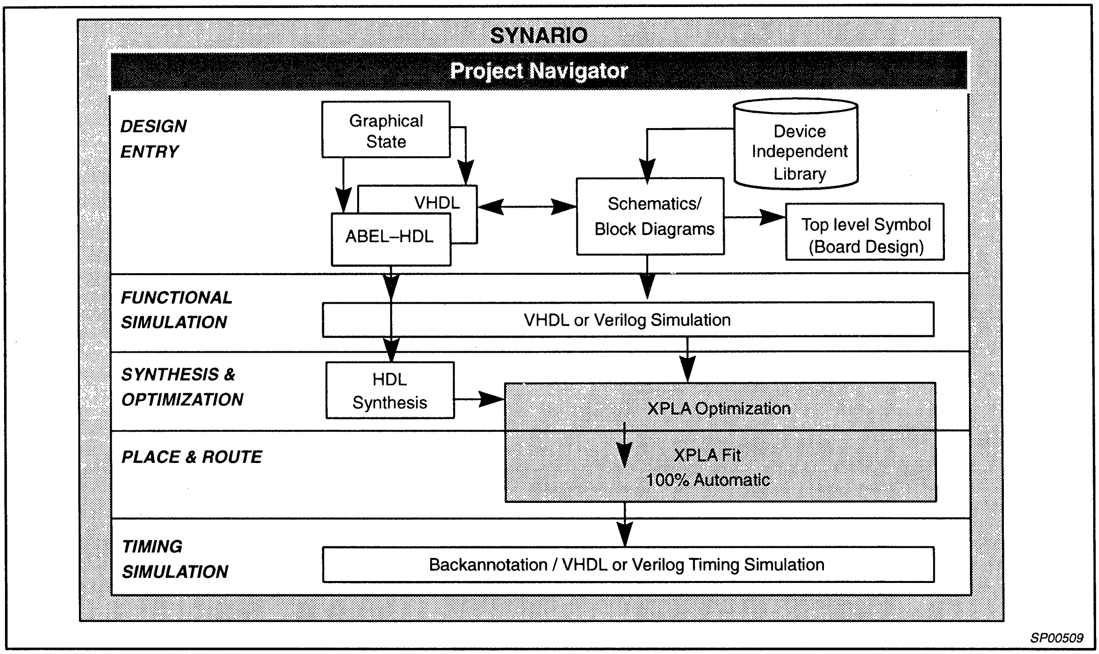


Figure 3.

Development software

Philips/Synopsys Design Environment

Philips CoolRunner CPLDs are supported in the Synopsys® design environment with the MINC Synopsys library interface for MINC's PLDesigner-XL. This interface includes the synthesis and simulation libraries for the Philips CoolRunner™ CPLDs, and describes the logic functions that may be used to implement designs in CoolRunner™ devices. Through the use of these libraries, VHDL or Verilog HDL design descriptions that target CoolRunner™ CPLDs can be entered in any of the following Synopsys tools: Synopsys Design Compiler family, Synopsys FPGA Compiler, and Synopsys Design Analyzer.

Design entry, simulation, and synthesis via the Synopsys design environment offers a number of advantages. In addition to simplifying the overall design task, the Synopsys tools support multiple HDLs with one compiler. As a result, there are no issues with language subsets or inconsistencies when moving between device architectures. The high-level, top-down Synopsys design environment increases productivity by providing a single front-end tool set for design, simulation, and synthesis of the overall system design. It also reduces development time for programmable logic devices by facilitating the re-use of designs previously developed in a Synopsys tool.

MINC's Synopsys library interface supports Philips CPLD's by targeting a library of MINC-specific components that includes the CoolRunner devices. This allows the designer to process a Synopsys design and generate an EDIF netlist that can, in turn, be read and processed by MINC's PLDesigner-XL. Within the PLDesigner-XL environment, the design can be further processed using all of the facilities this tool offers for physical implementation. This includes automatic, multiple device partitioning and fitting as well as device selection based upon the specification of multiple physical constraints, such as performance, power dissipation, and area.

System and Software Requirements

- Synopsys Design Compiler (Version 3.5b) or FPGA Compiler (Version 3.5a)
- MINC's PLDesigner-XL
- Philips Device Libaries for MINC's PLDesigner-XL

Recommended Operating Systems

- Hewlett-Packard HP-UX 9.05
- Sun MicroSystems SunOS 4.1.4 (V4.1.3 compatible)
- Sun MicroSystems Solaris 2.4 (requires SunOS compatibility mode)

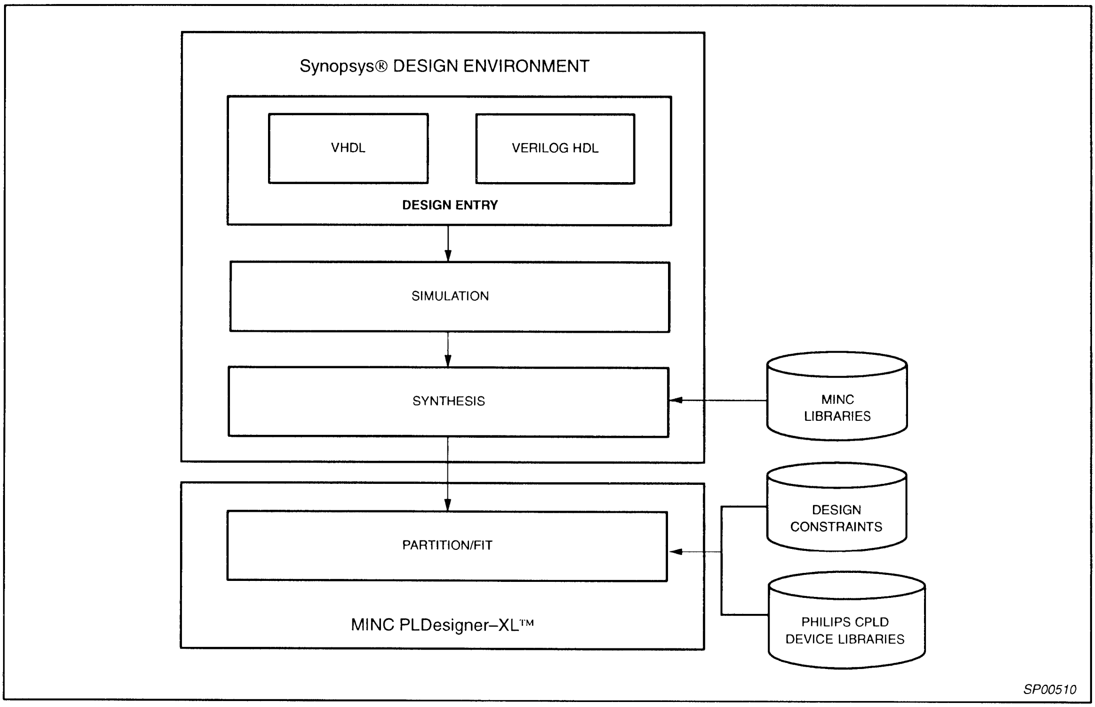


Figure 4.

Viewlogic/Philips Design Environment

The Philips CPLD Fitter is fully integrated into the Intelliflow™/ViewPLD® products and Workview Office/Powerview design environments of Viewlogic. It is tightly integrated to work with the ViewDraw® design entry system, the ViewSynthesis® and ABEL synthesis, and the ViewSim® and SpeedWave™ simulation environments.

The Viewlogic/Philips design flow follows a top-down or mixed-level design methodology. Designs can be entered as schematics using ViewDraw or in the industry-standard hardware description languages, VHDL and ABEL. The design flow is flexible in that it allow users to enter designs as a mixture of schematic and hardware description language.

Intelliflow/ViewPLD supports flows in which the user can functionally simulate the designs before they are targeted to a particular device and before the device is selected. The designs are optimized and VHDL models with unit timing are created for functional simulation. This enables the user to find any design problems at an early stage, thus reducing overall development time. A top level symbol can also be created for the HDL modules to enable their use in a schematic as part of a more complex design. After device selection and targeting, full timing simulation is available to verify the timing requirements.

Users can choose from a variety of available devices. The designs are implemented in the target device after device selection. With control of the entire process, through easily identified and explained options, the user can decide the best possible configuration for the design. After the design is fit into the device using the Philips fitter, post-route VHDL models are generated that enable the user to simulate the device with full timing information. The Philips fitter software is obtained directly from Philips, and is not supplied by Viewlogic. However, Philips and Viewlogic have jointly qualified the design flow to ensure full functionality.

System and Software Requirements

Viewlogic Software
- Intelliflow/ViewPLD
- ViewDraw/ViewSim/ViewTrace
- ViewSynthesis

Philips Software
- Philips Device Fitter for Viewlogic (PZVIEWMSC)

Recommended Operating Systems
- Windows®95 and Windows NT® 3.51 or 4.0
- SunOS 4.1.3_V1 or higher
- Solaris 2.4 or 2.5
- HP-UX 9.05 or 10.01

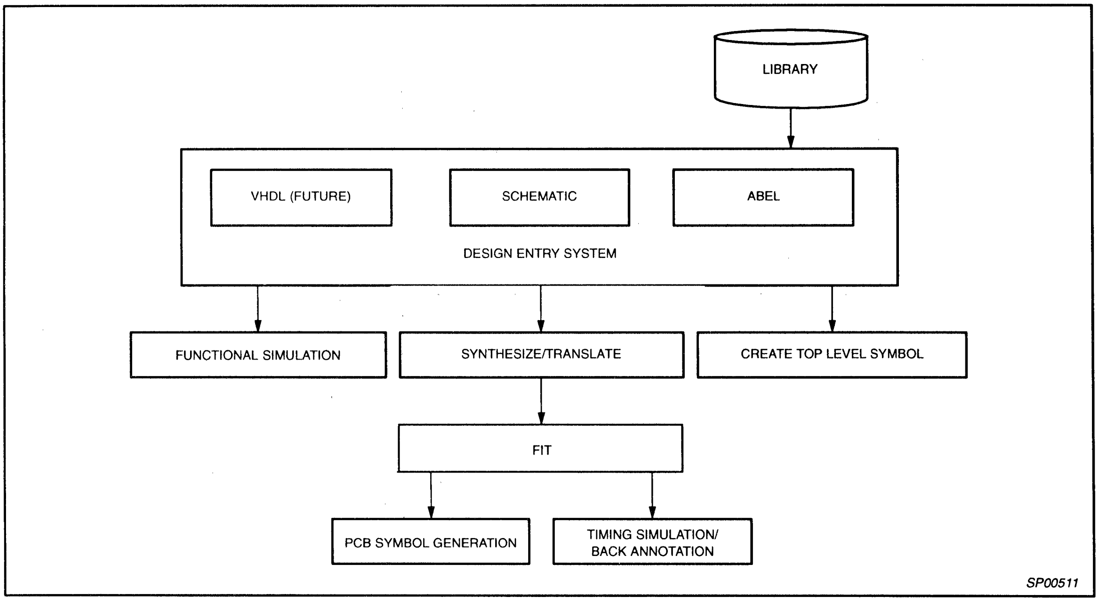


Figure 5.

Philips/MINC Design Environment

The Philips CPLD fitter is fully integrated into the MINC PLDesigner–XL software. Within the PLDesigner–XL environment, a design can be processed using all of the facilities this tool offers for physical implementation. This includes automatic, multiple device partitioning and fitting as well as device selection based upon the specifiction of multiple physical constraints, such as performance, power dissipation, and area. These features, coupled with fully hierarchical synthesis for any combination of CPLDs improves both cost–efficiency and time–to–market.

The Philips/MINC design flow supports a top–down or mixed–level design methodology. The integrated interface includes the synthesis and simulation libraries for the Philips CoolRunner CPLDs, along with the logic functions that may be used to implement designs in the CoolRunner devices. Device fitters for Philips CoolRunner CPLDs are included free–of–charge, or to installed users via maintenance in PLDesigner–XL for workstations. The specific device fitter maps the design source file into the Philips CPLD architecture, and translates design–specific timing back into the simulation environment.

System and Software Requirements

- MINC's PLDesigner–XL
- Philips Device Libraries for Mind's PLDesigner–XL

Recommended Operating Systems

- Hewlett–Packard HP–UX 9.05
- Sun MicroSystems SunOS 4.1.4(V4.1.3 compatible)
- Sun MicroSystems Solaris 2.4 (requires SunOS compatibility mode)

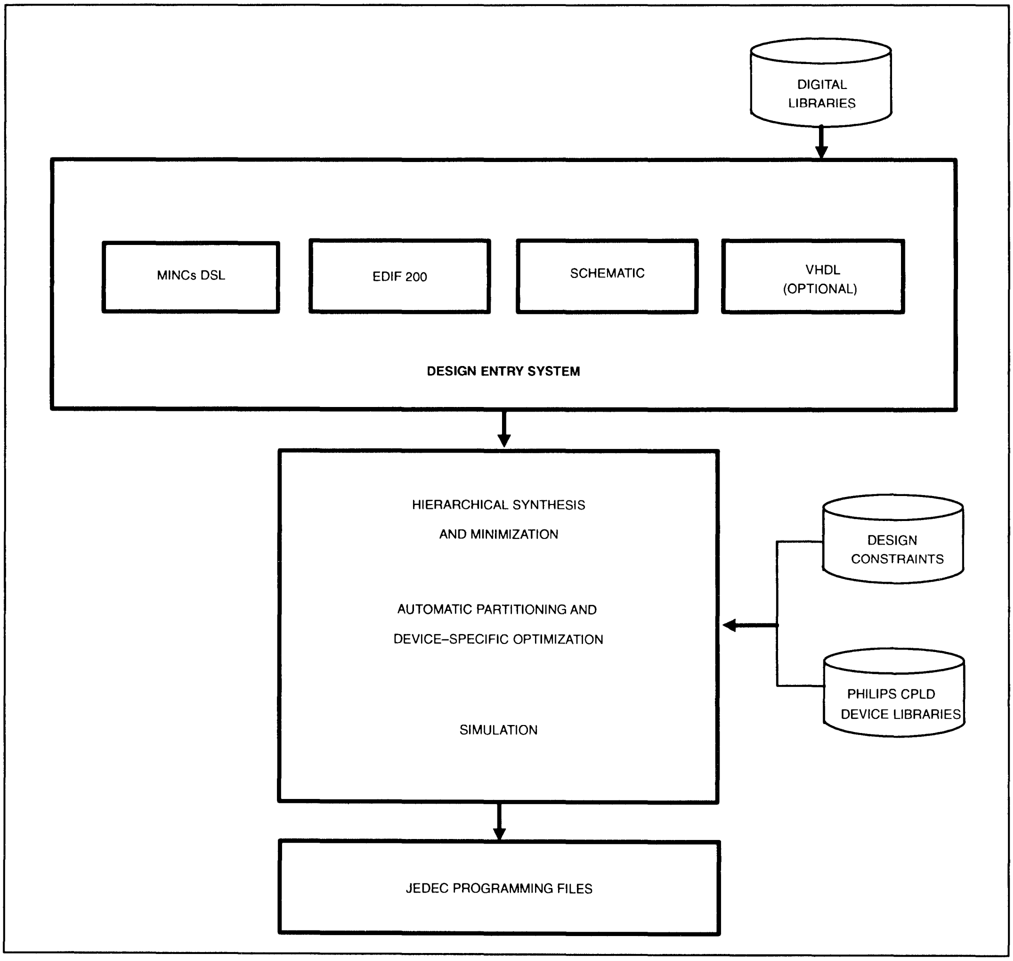


Figure 6.

PHILIPS XPLA™ DESIGNER

The overall CPLD design process consists of five steps: design definition, functional simulation, device fitting, post layout (timing) simulation, and programming. The XPLA™ Designer provides the first four of these five design steps and also supports the final step by producing a JEDEC file which can be used by most industry programmers to configure the device. The XPLA™ Designer also produces Verilog and VHDL timing models which can be used in board level simulations.

Supported Devices

XPLA™ Designer software supports the following devices:

- PZ3032 – 32 macrocell, 3.3 Volt CPLD
- PZ5032 – 32 macrocell, 5 Volt CPLD
- PZ3064 – 64 macrocell, 3.3 Volt CPLD
- PZ5064 – 64 macrocell, 5 Volt CPLD
- PZ3128 – 128 macrocell, 3.3 Volt CPLD
- PZ5128 – 128 macrocell, 5 Volt CPLD

The CPLD Design Process

Figure 7 shows the high-level CPLD design process. The manual walks the user through this design process. Each section of the manual gives a general description of the development stage and then gives the reader a brief introduction of how to use the XPLA Designer to complete each stage. A detailed description of each design stage is given in subsequent chapters of the manual.

Design Definition

The design definition stage is where the design is actually created. Starting with the knowledge of what the design must do, the designer enters that information to create a design. Within the limitations of the design package, the designer may use various methods of design entry including: schematics, textual models, state diagrams, state machines, and boolean equations. To the computer, all of these formats are equivalent because the software will eventually link them together. Having the ability to choose the design method gives designers a very powerful tool for their work. For example, designing a complex state machine using only registers and logic gates can be difficult; it may be easier to design it with an HDL (Hardware Description Language) that the computer can translate into those same registers and gates.

The XPLA™ Designer uses the PHDL (Philips Hardware Description Language) language to support the following design entry formats: boolean equations, state machines, and truth tables. A design can be created or selected by choosing the Design/New Design or the Design/Open Design command from the "Design pull-down Menu" on the XPLA Designer Interface shown in Figure 8. The design can be entered or modified by selecting the "Edit Button" on the XPLA Designer Interface. Once the design has been entered, the PHDL must be compiled to check for syntax errors and to minimize the user's logic. In order to activate the compiler, the "Compile Button" on the XPLA Designer Interface, as illustrated in Figure 8, must be selected.

Functional Simulation

Functional simulation verifies that the **design** is performing as intended. This is different from verifying that the **part** is performing as intended. It checks only that the logical response of the design to particular input stimuli is correct, but does not check any physical parameters such as speed or power. For example, a simple design that adds two plus two will functionally simulate correctly if the output is four, even if it took hours for the output to appear.

When the XPLA™ Designer simulator is run after the design has been compiled but before the design has been fitted into the device, the simulator will act as a functional simulator. If the simulator is employed after the logic has been fit into the design, the XPLA Designer simulator will act as a timing simulator and use the actual timing parameters of the target device. The simulator can be activated by selecting the "Simulate Button" on the XPLA Designer Interface as shown in Figure 8. The simulation input stimuli is defined in a "Simulation Control Language" (SCL) file.

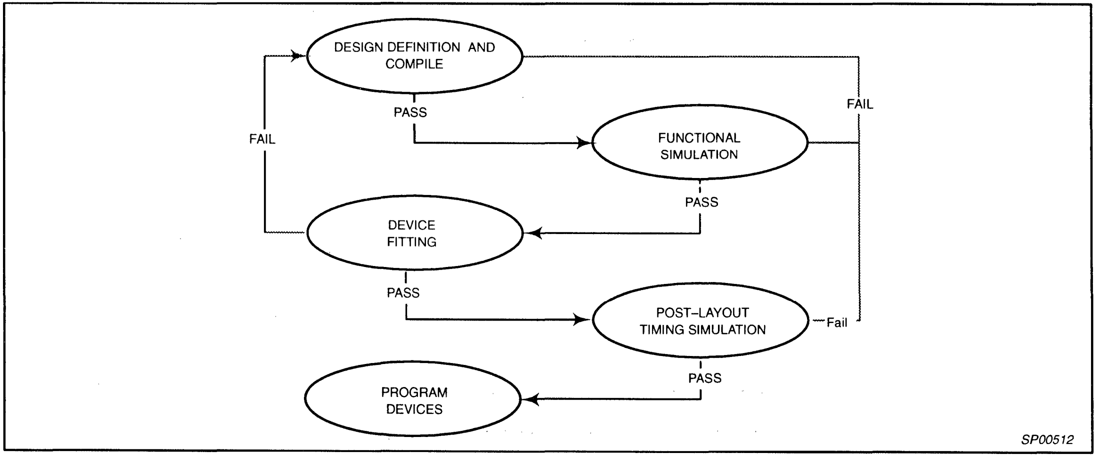
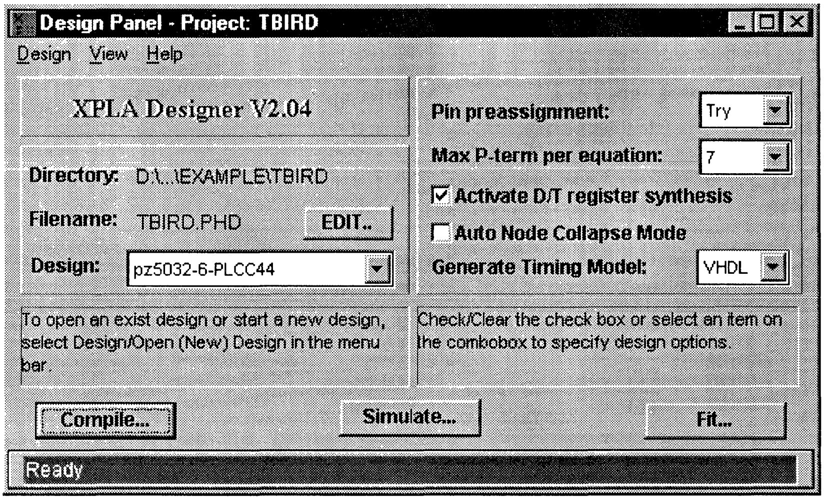


Figure 7.

Development software



Design Panel - Project: TBIRD

XPLA Designer V2.04

Directory: D:\..\EXAMPLE\TBIRD

Filename: TBIRD.PHD

Design: pz5032-6-PLCC44

Pin preassignment: Try

Max P-term per equation: 7

☑ Activate D/T register synthesis

□ Auto Node Collapse Mode

Generate Timing Model: VHDL

To open an exist design or start a new design, select Design/Open (New) Design in the menu bar.

Check/Clear the check box or select an item on the combobox to specify design options.

Ready

SP00513

Figure 8.

Device Fitting
Device fitting is where software translates the design into a file format that a part programmer can understand and then attempts to fit the user's logic into the target device's resources. The file format varies depending on the part you are using. A device can be selected by highlighting the chosen device in the "Device pull-down Menu" on the XPLA Designer Interface. Once the sources have been compiled and the design is functioning properly, the XPLA Designer fitter can be employed by selecting the "Fit Button" from the XPLA Designer Interface.

The user can control the manner in which the fitter places the design into the device by using the following options on the XPLA Designer Interface:

- Pin Assignments
- Max P-term per Equation
- Activate D/T Register Synthesis
- Auto Node Collapse
- Generate Timing Model
- A complete description on how to control the device fitting process can be found in Chapter 7 of the manual.

Post Layout Simulation
While the design may functionally simulate, the part may not function correctly due to physical limitations. For example, the part will not function correctly if you are using a 100 MHz clock, and there is a signal path in the part layout that takes more than 10 nanoseconds for the signal to reach the end of the path. To find this type of problem early in your design, you can do a second simulation that uses accurate delays from the datasheet specification of the devices to check the physical timing of the part. Like the functional simulation, this simulation also uses a test file that stimulates device inputs and records the outputs.

When fitting a design into the part, the XPLA Designer software generates a post-layout timing file. This file contains path delays for all signal routes based on real physical parameters for the selected CPLD. The XPLA Designer simulation uses this timing file to verify that the design will function correctly at the required frequency once it is programmed into the actual part. When the XPLA Designer simulator is employed after the design has been compiled and fitted into the device, the simulator will act as a timing simulator and use the actual timing parameters of the target device. The simulator can be activated by selecting the "Simulate Button" on the XPLA Designer Interface as shown in Figure 8. The simulation input stimuli is defined in a "Simulation Control Language" (SCL) file.

Programming
This final stage is generally done when all other steps have been completed and all design specifications have been met during the post-layout simulation. Designs that successfully fit into the selected CPLD are also translated into a JEDEC file for use in programming the part. The JEDEC file can be loaded into a part programmer which then configures the design into a part.

The JEDEC format is understood by many commercially available parts programmers. Refer to the user's manual of the specific programmer you are using for JEDEC compatibility and programming information.

The XPLA Help File
Included with the Philips XPLA Designer is a help menu. The help menu can be activated via the "Help Pull-Down Menu" in the XPLA Designer Interface and contains most of the information available in the manual.

Programming companies

The following is a listing of programming companies that support all or some of our CoolRunner™ CPLDs. Please visit our website at www.coolpld.com for up to date information on software revisions and hardware needs for individual part types.

COMPANY	ADDRESS	PHONE NUMBER
BP Microsystems	1000 North Post Oak Rd. Houston, TX 77055	800-225-2102
System General	1603A South Main St. Milpitas, CA 95035	408-263-6667
Tribal Microsystems	44388 S. Grimmer Blvd. Fremont, CA 94538	510-623-8859
SMS GmbH	Im Grund 15, 88239 Wangen, Germany	(49)7522 9728 0
Stag	Silver Court Watchmead Welwyn Garden City Herts AL7 1LT UK	011 44 1707 33214
Data I/O	10525 Willows Road N.E. Redmond, WA 98073	800-332-8246

Section 3
In–System Programming (ISP™)

CONTENTS

In-System Programming (ISP™)

JTAG BOUNDARY-SCAN AND ISP

Terminology

BC	Boundary-Scan Cell
BSDL	Boundary-Scan Description Language
BST	Boundary-Scan Test
CPLD	Complex Programmable Logic Device
IEEE	Institute of Electrical and Electronics Engineers
ISP	In-System Programmable
JEDEC	Joint Electron Device Engineering Council
JTAG	Joint Test Action Group
MC	Macrocell
PC	Personal Computer
PES	Programmer Electronic Signature
PZ3032/PZ5032	Philips CoolRunner™ 3V/5V 32 Macrocell Device
PZ3064/PZ5064	Philips CoolRunner™ 3V/5V 64 Macrocell Device
PZ3128/PZ5128	Philips CoolRunner™ 3V/5V 128 Macrocell Device
TAP	Test Access Port contained within the JTAG interface
UES	User Electronic Signature

JTAG Boundary-Scan Background

JTAG Boundary-Scan is the ability to test a set of devices using a standard JTAG interface instead of testing equipment which must make physical contact with the circuit pack. BST provides the ability to test the external connections of a device, the internal logic of the device, and the ability to capture data from the device during normal operation. BST provides many benefits including:

- Testability Benefits
  - No limits on the number of nets
  - Testability is designed in
  - Can force signals onto pins (Preload)
  - Can capture data from pin or core logic signals during normal operation
- Reliability Benefits
  - Not dependent on physical contact like existing test fixtures
  - Test fixtures do not deteriorate over time
  - Existing testing technologies have reached their limit
  - Better approach to handling smaller component size and components on both sides of the circuit pack
  - Shorter device interconnect routes (no fan-out required)
- Cost Benefits
  - Much cheaper test equipment
  - Reduced test preparation time
  - Reduced spare board inventories
  - Can incorporate denser technologies

ISP Background

ISP is the ability to reconfigure the logic and functionality of a device, circuit pack, or complete electronic system before, during, and after its manufacture and shipment to the end customer. ISP provides many benefits including:

- Design Benefits
  - Superior Prototyping Solution
  - Debug Partitioning
  - Circuit pack reconfiguration during debug
- Manufacturing Benefits
  - Multi-Functional Hardware
  - Reconfigurability for Test
  - Simplified Manufacturing Process
  - Reduced Inventory
  - Improved Manufacturing Quality
- Field Support Benefits
  - No need to replace devices for new features/bug fixes
  - Possible field upgrade of hardware

Philips' JTAG Boundary-Scan Features

Listed below are the JTAG Boundary-Scan features implemented within the Philips ISP CPLDs.

- JTAG Boundary-Scan supported mainly for high pin count devices.
- Use JTAG port for JTAG Boundary-Scan testing.
- JTAG Boundary-Scan devices can be daisy chained together to allow multiple devices to be tested from a single JTAG stream.
- JTAG TAP Controller implementation.
- IDCODE Register contains 32 bits and a pre-defined format.
- Support for the following JTAG Boundary-Scan instructions:
  - Sample/Preload
  - Extest
  - Bypass
- JTAG Boundary-Scan Register implementation.
- TAG USERCODE Register implementation.
- Support for the following JTAG Boundary-Scan instructions:
  - Idcode
  - Usercode (implemented by verifying EEPROM row 41 sides A & B)
  - HighZ

Philips' ISP Features

Listed below are the ISP features implemented within the Philips ISP CPLDs.

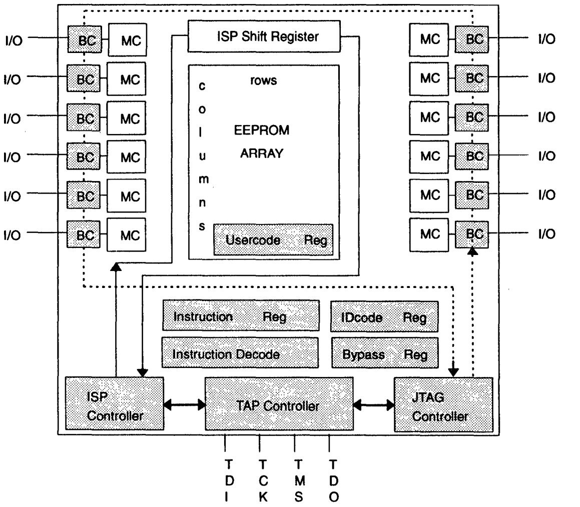
- ISP supported for high pin count devices.
- Use JTAG port for ISP programming.
- No external supervoltages are required to program/erase the devices. All supervoltages are generated internal to the device from the V_{DD} input voltage.

- ISP security bit is available to protect the user program information.
- ISP devices can be daisy chained together to allow multiple devices to be programmed from a single data stream.
- ISP programming support through
  - PC Parallel Port
  - Automated Test Equipment.
  - Third party Programmers.
  - Serial Port
  - an Embedded Processor
- Number of erase/program cycles equals 1000.
- Program retention time of 20 years.
- ISP programming time less than 10 seconds for a 256 Macrocell device.
- Simultaneous programming of multiple ISP devices daisy chained together.

Philips BST and ISP CPLD Architecture

All Philips high pin count CPLDs support both JTAG Boundary-Scan and ISP features through a standard JTAG interface. The JTAG interface consists of two parts: a TAP Port and a TAP Controller. The TAP Port consists of 4 required JTAG pins (TCK, TMS, TDI, TDO) plus 1 optional JTAG pin (TRST*). The TAP port signals are described in the section *"JTAG Interface*". The TAP Controller is a sequential circuit which is used to clock in and parse JTAG instructions. The TAP Controller defined by the IEEE 1149.1 JTAG Specification is illustrated in the section, *"JTAG TAP Controller."*

Please refer to Figure 1 for a high level block diagram of a Philips ISP CPLD using the JTAG interface to access the BST and ISP functionality. As illustrated by Figure 1, the additional circuitry needed to implement the BST and ISP functionality consists of a TAP Controller, a JTAG Controller, an ISP Controller, an ISP Shift Register, and a Boundary-Scan Register. The EEPROM array is made up of rows and columns of EEPROM cells. The ISP Shift Register contains the address to select individual EEPROM rows and contains the data which can be programmed into or read from the selected EEPROM row.



SP00487

Figure 1. ISP and BST CPLD High Level Architecture

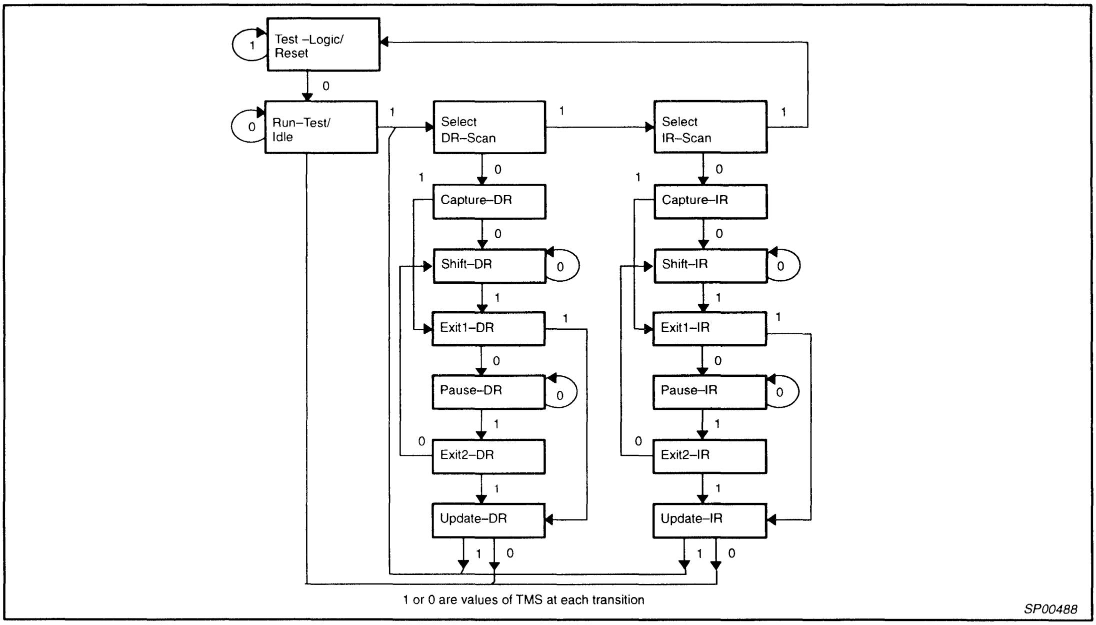


Figure 2. JTAG TAP Controller

JTAG TAP Controller

The TAP Controller is a synchronous finite state machine that responds to changes at the TMS and TCK signals of the TAP and controls the sequence of operations of the circuitry defined by the 1149.1 IEEE Standard. This TAP Controller, as seen in Figure 2, is implemented in the Philips CPLDs.

The behavior of the TAP controller and other BST and ISP logic in each of the controller states is briefly described below.

Test-Logic/Reset

The BST and ISP logic is disabled so that normal operation of the on-chip system logic (i.e. in response to stimuli received through the system pins only) can continue unimpeded. This is achieved by initializing the instruction register to contain the IDCODE instruction. No matter what the original state of the controller, it will enter *Test-Logic-Reset* when TMS is held high for at least 5 rising edges of TCK. The controller remains in this state while TMS is high. Note that the TAP controller will be forced to the *Test-Logic-Reset* controller state at power-up.

Run-Test/Idle

All of the instructions supported by the Philips CPLDs do not cause functions to execute in the *Run-Test/Idle* controller state. Thus, all BST and ISP data registers selected by the current instruction shall retain their previous state (i.e., Idle). The instruction does not change while the TAP controller is in this state.

Select-DR-Scan

This is a temporary controller state in which all BST and ISP data registers selected by the current instruction retain their previous state. The instruction does not change while the TAP controller is in this state.

Select-IR-Scan

This is a temporary controller state in which all BST and ISP data registers selected by the current instruction retain their previous state. The instruction does not change while the TAP controller is in this state.

Capture-DR

In this controller state, data may be parallel loaded into the BST data registers selected by the current instruction on the rising edge of TCK. If a BST data register selected by the current instruction does not have parallel input, or if capturing is not required for the selected test, then the register retains its previous state. The instruction does not change while the TAP controller is in this state.

Shift-DR

In this controller state, the BST or ISP data register connected between TDI and TDO as a result of the current instruction shifts data one stage towards its serial output on each rising edge of TCK. BST or ISP data registers that are selected by the current instruction, but are not placed in the serial path, retain their previous state. The instruction does not change while the TAP controller is in this state.

Exit1-DR

This is a temporary state. All BST or ISP data registers selected by the current instruction retain their previous state. The instruction does not change while the TAP controller is in this state.

Pause-DR

This controller state allows shifting of the BST or ISP data register in the serial path between TDI and TDO to be temporarily halted. All BST or ISP data registers selected by the current instruction retain their previous state. The instruction does not change while the TAP controller is in this state.

Exit2-DR

This is a temporary state. All BST or ISP data registers selected by the current instruction retain their previous state. The instruction does not change while the TAP controller is in this state.

Update-DR

Some BST data registers may be provided with a latched parallel output to prevent changes at the parallel output while data is shifted in the associated shift-register path in response to certain instructions (e.g. EXTEST). Data is latched onto the parallel output of these BST data registers from the shift-register path on the falling edge of TCK in the *Update-DR* controller state. The data held at the latched parallel outputs should not change. The instruction does not change while the TAP controller is in this state.

Capture-IR

In this controller state, the shift-register contained in the instruction register loads a pattern of fixed logic values on the rising edge of TCK. In addition, design-specific data may be loaded into shift-register stages that are not required to be set to fixed values. BST or ISP data registers selected by the current instruction retain their previous state. The instruction does not change while the TAP controller is in this state.

Shift-IR

In this controller state, the shift-register contained in the instruction register is connected between TDI and TDO and shifts data one stage towards its serial output on each rising edge of TCK. BST or ISP data registers that are selected by the current instruction, but are not placed in the serial path, retain their previous state. The instruction does not change while the TAP controller is in this state.

Exit1-IR

This is a temporary state. BST or ISP data registers selected by the current instruction retain their previous state. The instruction does not change while the TAP controller is in this state and the instruction register retains its state.

Pause-IR

This controller state allows shifting of the instruction register to be temporarily halted. BST or ISP data registers selected by the current instruction retain their previous state. The instruction does not change while the TAP controller is in this state and the instruction register retains its state.

Exit2-IR

This is a temporary state. BST or ISP data registers selected by the current instruction retain their previous state. The instruction does not change while the TAP controller is in this state and the instruction register retains its state.

Update-IR

The instruction shifted into the instruction register is latched onto the parallel output from the shift-register path on the falling edge of TCK in the *Update-IR* controller state. Once the new instruction has been latched, it becomes the current instruction. BST or ISP data registers selected by the current instruction retain their previous state.

JTAG Registers

Figure 3 illustrates the JTAG Registers that are incorporated into Philips CPLDs.

Table 1 gives a description of each JTAG Register for Philips CPLDs.

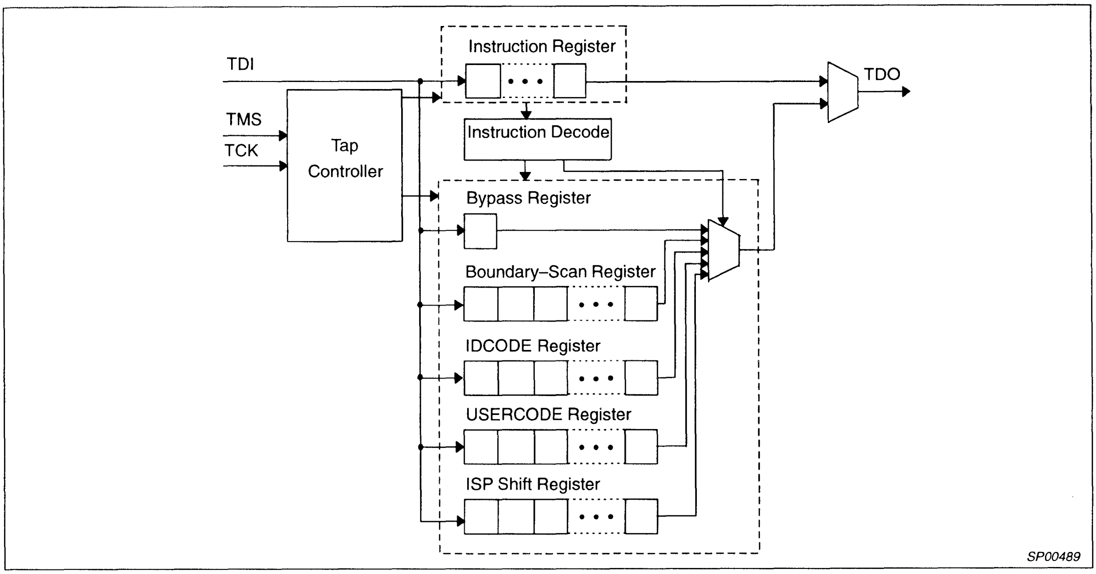


Figure 3. JTAG Registers

Table 1. JTAG Register Description

REGISTER	# OF BITS	REGISTER DESCRIPTION
Instruction Register	4	The Instruction Register is a shift-register-based design which allows an instruction to be shifted into a device. The instruction shifted into the register is latched at the completion of the shifting process. The instruction is used to select the BST or ISP operation and/or the data register to be accessed. The parallel output from the Instruction Register is latched to ensure that the BST and ISP logic is protected from the transient data patterns that will occur in its shift-register stages as new instruction data is entered.
Bypass Register	1	The Bypass Register contains a single shift-register stage and is used to provide a minimum length serial path between the TDI and TDO pins of a component when no test or program operation of that component is required. This allows more rapid movement of test/program data to and from other components on a circuit pack that are required to perform test/program operations.
Boundary-Scan Register	280	The Boundary-Scan Register allows testing of circuitry external to the CPLD and also permits the system signals flowing into and out of the CPLD logic to be sampled and examined without causing interference with the normal operation of the CPLD logic. The Boundary-Scan Register is a long shift register composed of all the Boundary-Scan cells at the pins of the device.
IDCODE Register	32	This register must consist of a 32 bit shift-register, parallel-in and serial out. The register contains the following information: **Bit(s)** **Usage** 0 1 - pre-defined 1-11 Manufacturing Identity 12-27 Part Number 28-31 Version
USERCODE Register	960	Contains the UES information. This Register is comprised of Row 41 of the EEPROM array.
ISP Shift Register	7 Address bits 1028 Data bits for PZ3128	Used to address the EEPROM row and contains the data that is being written into or read from the EEPROM array.

JTAG Boundary-Scan Registers

The following Boundary-Scan Cells were selected for Philips CPLD's dedicated input and bi-directional (I/O) pins.

An Observe-Only Dedicated Input BSR Cell

Figure 4 illustrates an Observe-Only BSR Cell for a dedicated input. The advantages associated with this Boundary-Scan Cell include:

- Does not require a multiplexer in the device's input speed path.
- Support for all mandatory BST instructions.
- Less silicon to implement than a BSR Cell with or without an Update Register.

Compliant BSR Cell with an Observe-Only Input for Bidirectional Pin

Figure 5 illustrates a Compliant BSR Cell with an observe-only input for a bi-directional pin. The advantages associated with this Boundary-Scan Cell include:

- Does not require a multiplexer in the device's input speed path.
- Support for all mandatory BST instructions.
- Less silicon to implement than a Compliant BSR Cell without an Observe-Only Input.
- The ability to control (latch) the data driving the device's output pin.
- The ability to sample and preload the output enable for the device's 3-State outputs.

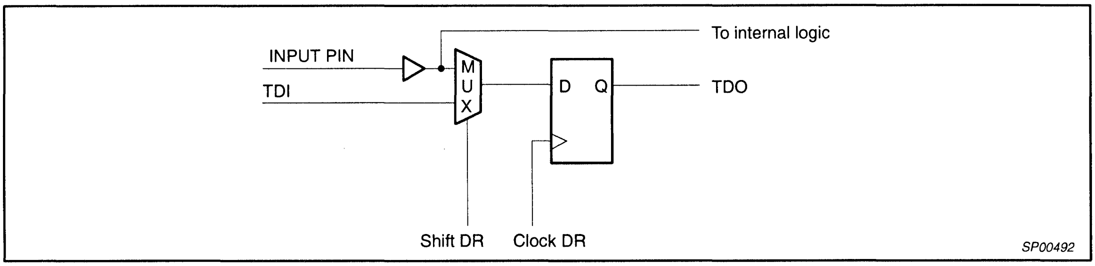


Figure 4. An Observe-Only Dedicated Input BSR Cell

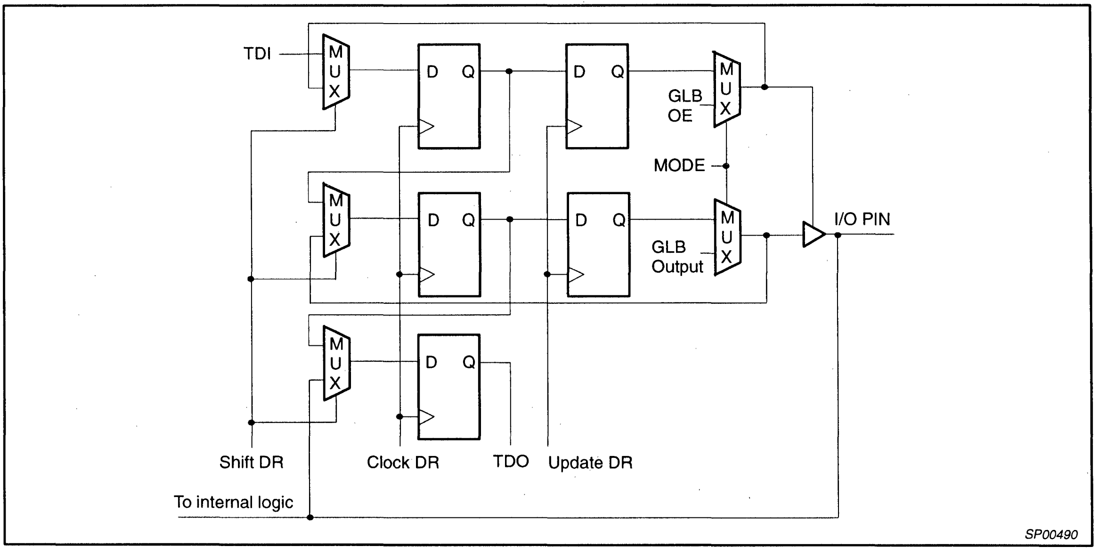


Figure 5. Compliant BSR Cell with an Observe–Only Input for a Bi–directional Pin

JTAG Interface

As mentioned before, the JTAG pins are used to support both the BST and ISP features. Table 2 gives a description of the JTAG pins to be used to support both BST and ISP. The optional TRST* (Test Reset) signal is not required for either the BST or ISP functionality. However, leaving out the TRST* pin requires that a power up reset circuit must be designed into the device. A power up reset circuit is included in all Philips CPLDs. These pins should contain an external pull resistor to keep the JTAG signals from floating when they are not being used.

For Philips' CPLDs, an ISP ENABLE instruction is sent to the device that enables the ISP functionality (instead of using a dedicated ISP ENABLE signal). This approach is similar to the approach taken by the Altera MAX7000S family. However in the MAX7000S family, these 4 signals use 4 pins which can be either used for ISP or as general I/O pins if ISP is not required by the user. Once these pins are used for ISP, these pins and the Macrocell logic associated with these pins are no longer available to the user. The selection of whether these pins are used for ISP or as general I/O is made when the JEDEC file is generated. With Philips' CPLDs, like the MAX700S family, the 4 JTAG signals use 4 pins which can be used for ISP or as general I/O pins if ISP is not required by the user. However, unlike the MAX7000S family, the Macrocells associated with these pins can be used as buried logic when these 4 pins are used for ISP.

A dedicated JTAG port is used for Philips CPLDs containing more than 128 Macrocells.

Table 2. JTAG Pin Description

PINS	NAME	DESCRIPTION
TCK	Test Clock Output	Clock pin to shift the serial data and instructions in and out of the TDI and TDO pins, respectively. TCK is also used to clock the TAP Controller state machine.
TMS	Test Mode Select	Serial input pin selects the JTAG instruction mode. TMS should be driven high during user mode operation.
TDI	Test Data Input	Serial input pin for instructions and test data. Data is shifted in on the rising edge of TCK.
TDO	Test Data Output	Serial output pin for instructions and test data. Data is shifted out on the falling edge of TCK. The signal is 3-Stated if data is not being shifted out of the device.

JTAG BOUNDARY-SCAN AND ISP COMMANDS

The TAP Controller will clock in and parse both JTAG and ISP instructions. The Philips ISP CPLDs contain a 4 bit Instruction Register which is large enough to contain all BST and ISP instructions. Since the Philips CPLDs support 6 JTAG Boundary-Scan instructions and 4 ISP instructions, the Instruction Register contains 4 bits to represent these instructions ($2^4 = 16 > 12$). All unused instruction codes, namely 0011, 0110, 0111, 1000, 1101, and 1110 are mapped into the Boundary-Scan BYPASS instruction which passes the incoming data (TDI) to the outgoing data (TDO).

The BST and ISP Commands supported by the Philips CPLDs are specified in the following subsections. The subsections are broken into low level and high level commands. The low level commands will be executed within the CPLD and the high level commands will be executed on a PC and/or Workstation. Please note that all high level commands are comprised of issuing a sequence of low level commands.

Low Level JTAG Boundary-Scan and ISP Commands

Low Level JTAG Boundary-Scan Commands

The low level JTAG Boundary-Scan Commands that are supported by the Philips ISP CPLDs are specified in Table 3. These are the only commands required to implement all of the required high level JTAG Boundary-Scan Commands.

Low Level ISP Commands

As noted above, the low level ISP commands are basic commands which are implemented inside the CPLD. The low level ISP commands which are supported by the Philips ISP CPLDs are specified in Table 4. Using the ENABLE instruction before the Erase, Program, and Verify instructions allows the user to specify the outputs of the device using the JTAG Boundary-Scan SAMPLE/PRELOAD command.

Please note that an ENABLE command must precede all ISP commands **unless** an ENABLE command has already been given for a preceding ISP command **and** the device has not gone through the Test-Logic/Reset TAP Controller State.

Table 3. Supported Low Level JTAG Boundary-Scan Commands

Instruction (Instr. Code) *Register Used*	**DESCRIPTION**
Sample/Preload (0010) *Boundary-Scan Register*	The mandatory SAMPLE/PRELOAD instruction allows a snapshot of the normal operation of the component to be taken and examined. It also allows data values to be loaded onto the latched parallel outputs of the Boundary-Scan Shift-Register prior to selection of the other boundary-scan test instructions.
Extest (0000) *Boundary-Scan Register*	The mandatory EXTEST instruction allows testing of off-chip circuitry and board level interconnections. Data would typically be loaded onto the latched parallel outputs of Boundary-Scan Shift-Register using the Sample/Preload instruction prior to selection of the EXTEST instruction.
Bypass (1111) *Bypass Register*	Places the 1 bit bypass register between the TDI and TDO pins, which allows the BST data to pass synchronously through the selected device to adjacent devices during normal device operation. The Bypass instruction can be entered by holding TDI at a high value and completing an Instruction-Scan cycle.
Idcode (0001) *Boundary-Scan Register*	Selects the IDCODE register and places it between TDI and TDO, allowing the IDCODE to be serially shifted out of TDO. The IDCODE instruction permits blind interrogation of the components assembled onto a circuit pack. Thus, in circumstances where the component population may vary, it is possible to determine what components exist in a product.
STCTEST (0100) *Boundary-Scan Register*	The STCTEST instruction is used by the Philips design community to check the integrity of the JTAG Boundary-Scan structure. This command will be for internal Philips use only and will not be advertised to customers.
HighZ (0101) *Bypass Register*	The HIGHZ instruction places the component in a state in which all of its system logic outputs are placed in an inactive drive state (e.g., high impedance). In this state, an in-circuit test system may drive signals onto the connections normally driven by a component output without incurring the risk of damage to the component. The HighZ instruction also forces the Bypass Register between TDI and TDO.

Table 4. Low Level ISP Commands

Instruction *(Register Used)*	**Instruction Code**	**DESCRIPTION**
Enable *(ISP Shift Register)*	1001	Enables the Erase, Program, and Verify commands. Using the ENABLE instruction before the Erase, Program, and Verify instructions allows the user to specify the outputs of the device using the JTAG Boundary-Scan SAMPLE/PRELOAD command.
Erase *(ISP Shift Register)*	1010	Erases the entire EEPROM array. The outputs during this operation can be defined by the user by using a sample preload command.
Program *(ISP Shift Register)*	1011	Programs the data in the ISP Shift Register into the addressed EEPROM row. The outputs during this operation can be defined by the user by using a sample preload command.
Verify *(ISP Shift Register)*	1100	Transfers the data from the addressed row to the ISP Shift Register. The data can then be shifted out and compared with the JEDEC file. The outputs during this operation can be defined by the user by using a sample preload command.

High-Level JTAG Boundary-Scan and ISP Commands
The High Level ISP commands are given in Table 5. Again, these commands are executed on a PC or Workstation environment and are not implemented inside the CPLD. Please note that all high level commands are comprised of issuing a sequence of low level commands.

A few examples of how high level commands are implemented using low level commands are given in Table 6.

Table 5. High Level JTAG Boundary-Scan and ISP Commands

INSTRUCTION	DESCRIPTION
BULK_ERASE	Erases the entire EEPROM array.
BLANK_CHECK	Verifies that the device is erased.
PROGRAM	Programs the data into the EEPROM array.
VERIFY	Verifies that the data programmed into the EEPROM array is correct.
PR_SECURITY	Programs the security cell of the device.
RD_SECURITY	Checks to see the device is secured.
PR_USER_ID	Programs the User Signature (UES) information.
USERCODE	Reads the User Electronic Signature (UES) information.
RD_PRGM_ID	Reads the Programmer Electronic Signature (PES) information.
PROGRAM_VERIFY	Simultaneously Programs and Verifies the EEPROM array.
BYPASS	Connects TDO to TDI with a one-half clock (TCK) cycle delay.
IDcode	Reads the IDcode information.
Pre-Condition_Outputs	Allows the user to specify the outputs during an ISP command.

Table 6. High Level Command Examples

COMMAND	SEQUENCE
Program	Shift in the ENABLE instruction Shift in the PROGRAM instruction Shift in the address and data for the EEPROM row being programmed. Execute the command (Program the data into the selected EEPROM row) Repeat steps 3 and 4 until all EEPROM rows have been programmed
Verify	Shift in the ENABLE instruction Shift in the VERIFY instruction Shift in the address of the EEPROM row being verified. Execute the command (this transfers the row data into the ISP Shift Register) Shift out the data from the ISP Shift Register Compare the shifted-out data to the expected data Repeat steps 3 though 6 until all EEPROM rows have been verified
USERCODE	Shift in the ENABLE instruction Shift in a the VERIFY instruction Shift in address for Row 41 side A Execute the command (transfers the UES data into the ISP Shift Register) Shift out the data from the ISP Shift Register Shift in address for Row 41 side B Execute the command (transfers the UES data into the ISP Shift Register) Shift out the data from the ISP Shift Register
Pr_Security	Shift in the ENABLE instruction Shift in the PROGRAM instruction Shift in the address of the EEPROM row containing the security bit and shift in the data with the corresponding security bit set to the programming state and the other bits set to the non-programming state Execute the command (Program the data into the selected EEPROM row)

JTAG BSDL AND ISP CHAIN DESCRIPTION FILE FORMATS

The following subsections give the BSDL and ISP Chain Description Files formats.

JTAG Boundary-Scan Description Language File Format

The JTAG Boundary-Scan Description Language file contains the following information:

- Mapping of Pin names to Pin Types, i.e. Inputs, Outputs, I/O, etc.
- Mapping of Pin names to physical Pin numbers.
- TAP Pin Constraints.
- OPCODES for all supported Instructions.
- Register Accessed during each Instruction.
- Boundary Register Description:
  - Sequence of Cells in Boundary Register.
  - Mapping of Cell numbers to Pin name.

To test a JTAG board, a collection of all of the BSDL files for the JTAG ICs is required along with a net list describing how these ICs are connected.

Generate an ISP File

A JEDEC file format to ISP file format program has to be written. The ISP file contains the same information as the JEDEC file but in ISP Shift Register format. The ISP file contains the following information in the following format:

- Row 0, left side data (D1027 ...... D0)
- Row 1, left side data (D1027 ...... D0)
- Row 41, left side data (D1027 ...... D0)
- Row 42, left side data (D1027 ...... D0)
- Row 0, right side data (D1027 ...... D0)
- Row 1, right side data (D1027 ...... D0)
- Row 41, right side data (D1027 ...... D0)
- Row 42, right side data (D1027 ...... D0)

ISP Chain Description File Format

This section addresses the issue of programming single or multiple ISP devices within the same JTAG chain. The JEDEC committee has put together a standard format (LtrB JC-42.1-95-97 Chain Description File) for programming multiple devices on a single JTAG chain. A brief description of this format is given in Table 7.

It is important to recognize that both programmable and non-programmable devices can be placed on the same JTAG chain. To program a single device in a JTAG chain, the programming software must put all other devices in the JTAG chain in the JTAG Boundary Scan BYPASS mode. When in the BYPASS mode, devices pass programming data from the TDI pin to the TDO pin through a single bypass register without being affected internally, thereby enabling the programming software to erase, program, or verify the target device.

Table 7. ISP Chain Description File Format

FIELD	DESCRIPTION
Begin	Marks the beginning of the JC42.1 Chain Description File.
File Revision	Identifies the JC42.1 Chain Description File standard revision.
Default Declarations (Optional)	**Default Path**: Specifies a directory on a host computer of where to read and/or write data. **Default Mfr.**: Specifies the default manufacturer's code for the JEDEC chain.
Chain Site Records	Describes the device and the operation to be performed on each device in the JTAG chain. The order of the Chain Description File records must correspond to the order of the devices in the chain. In other words, there must be a one to one mapping of the device in the Chain Description File and the physical routing of the device on the circuit pack. There are two different types of device records: one for programmable devices and one for non-programmable devices. **Action Codes**: Specifies the function to be performed by the appropriate device. Examples of action codes include: program, verify, erase, bypass, etc. **Device Records**: Supports the following information: Mfr's Code, Part Name, Path Name (optional), File Name (optional), Mfr's Specific Data (optional).
End	Marks the end of the JC42.1 Chain Description File.

In-System Programming (ISP™)

ISP Programming Algorithm

EEPROM technology requires a very small amount of current (60 nA) but requires a fairly lengthy programming pulse (10 msec) to program each EEPROM cell. Therefore, it is advantageous to program multiple EEPROM cells in parallel. The PZ3032/PZ5032 architecture is arranged with columns containing 255 bits of data. These 255 bits are all clocked into the device and then these 255 data bits are programmed in the EEPROM at the same time (a total of 15.3 µA). It is believed that these data bits can be clocked into the device at a maximum JTAG frequency of 10 MHz. Figure 6 illustrates the ISP Programming Flow Diagram is be used with the Philips CPLD architecture.

ISP Programming Times

Table 8 contains the theoretical programming time required for each Philips CPLD. These times were calculated using the *ISP Programming Flow Diagram* given in Figure 6. A conservative program cycle time of 12 msec was used to estimate the programming times for the various CPLDs. As predicted above, the total programming times are greatly dependent on the EEPROM array configuration. The more data bits that are programmed in parallel (larger data column), the fewer number of required programming pulses, and thus the shorter the total programming interval.

Please note that these programming times are better than the programming times of most CPLD vendors and that the PC or Workstation used to execute the high level commands will have a large impact on these ISP programming times.

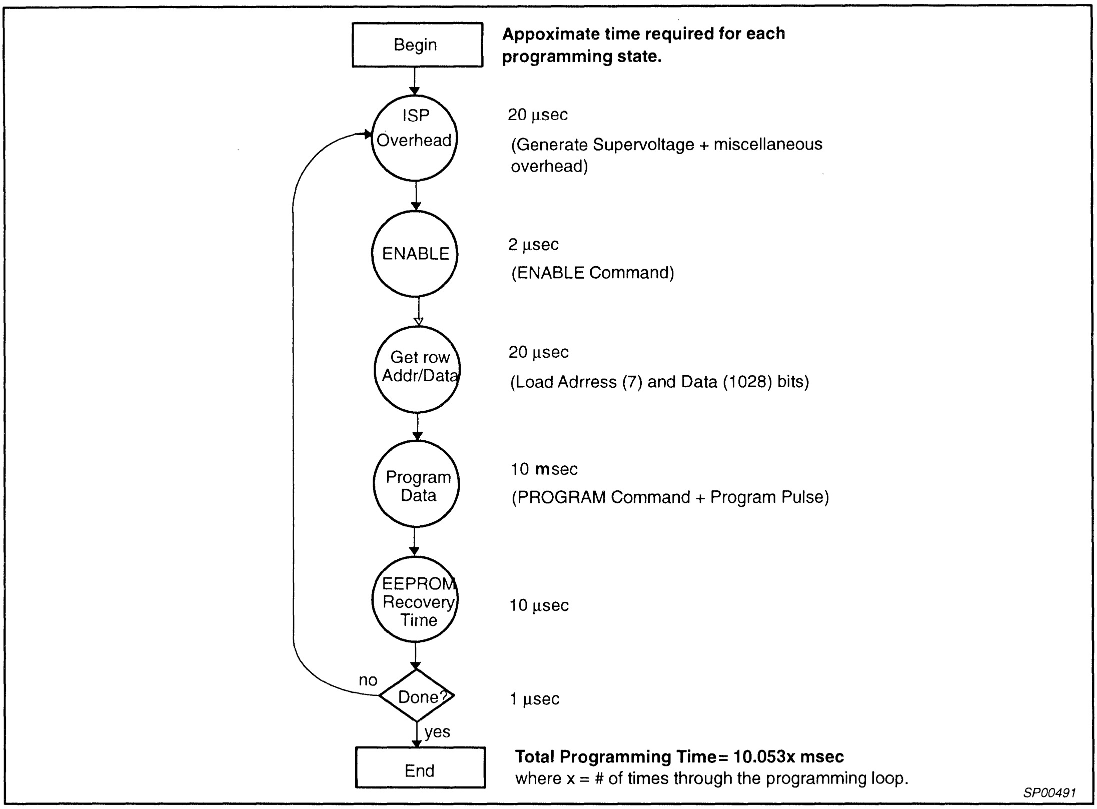

Begin
Appoximate time required for each programming state.
ISP Overhead
20 µsec
(Generate Supervoltage + miscellaneous overhead)
ENABLE
2 µsec
(ENABLE Command)
Get row Addr/Data
20 µsec
(Load Adrress (7) and Data (1028) bits)
Program Data
10 msec
(PROGRAM Command + Program Pulse)
EEPROM Recovery Time
10 µsec
no
Done?
1 µsec
yes
End
Total Programming Time= 10.053x msec
where x = # of times through the programming loop.
SP00491

Figure 6. PZ3128/PZ5128 ISP Programming Flow Diagram

Table 8. Theoretical Programming Times for Philips CPLDs

DEVICE	WIDTH OF COLUMN DATA			THEORETICAL PROGRAMMING TIMES		
PZ3032 / PZ5032	255			1.032 sec		
PZ3064 / PZ5064	256	512		2.064 sec	1.032 sec	
PZ3128 / PZ5128	256	512	1024	4.128 sec	2.064 sec	1.032 sec

Simultaneous (Parallel) ISP Programming Algorithm

Programming multiple devices on expensive Automated Testing Equipment will take some time and will be very costly. In order to keep ISP programming costs reasonable, it is required to be able to simultaneously program multiple devices. As illustrated in the section *"ISP Programming Algorithm,"* the number of programming pulses required to program a device or devices determines the time required to program the device(s). It is therefore advantageous to shift in data to multiple devices and then give each device a programming pulse at approximately the same time. There is approximately a 1% time increase required to simultaneously program two devices instead of a single device. Programming two devices in series would require twice the programming time than is required to program a single device. There is approximately a 22% time increase (1.22*x*)[1] required to simultaneously program twenty devices instead of a single device. Programming twenty devices in series would require twenty times (20*x*)[1] the programming time than is required to program a single device.

In order to program two devices in parallel, the designer first selects a row address for each device, then connects the ISP Shift Register through TDI and TDO. The data is then shifted in the order that the devices appear on the chain. Next, the instruction register is selected and two PROGRAM instructions are shifted in one after another. The selected row for both devices is programmed simultaneously. This algorithm can be expanded to simultaneously program multiple devices. These devices being simultaneously programmed do not have to be of the same size or type.

Simultaneously programming multiple devices requires that a complex file be produced. This file will consist of all ISP files from the devices involved with the appropriate low-level commands intermixed.

JTAG and ISP Interfacing

A number of industry-established methods exist for JTAG/ISP interfacing with CPLDs and other integrated circuits. Philips' CPLDs interface with the following methods:

- PC Parallel Port
- Workstation or PC Serial Port
- Embedded Processor
- Automated Test Equipment
- Third party Programmers
- High-End JTAG and ISP Tools

Boundary Scan Description Language (BSDL) descriptions of Philips' CPLDs are also available for use in test program development.

[1] Where *x* is the time required to program 1 device.

ISP Download Cable Specification

ISP DOWNLOAD CABLE SPECIFICATION

This chapter establishes the requirements for the In–System Programmable (ISP) Download Cable (a.k.a. HyperCable) used to program Philips CoolRunner CPLDs. The CoolRunner 128 Macrocell (CR128) device is the first of many Philips CPLD to contain ISP functionality. The ISP Download cable will not only support the CR128 but will support all Philips ISP CPLDs.

CPLD	Complex Programmable Logic Device
HyperCable	ISP Download Cable
ISP	In–System Programmable
JTAG	Joint Test Action Group
PC	Personal Computer
TAP	Test Access Port contained within the JTAG interface

Figure 1 illustrates the CoolRunner Download Cable.

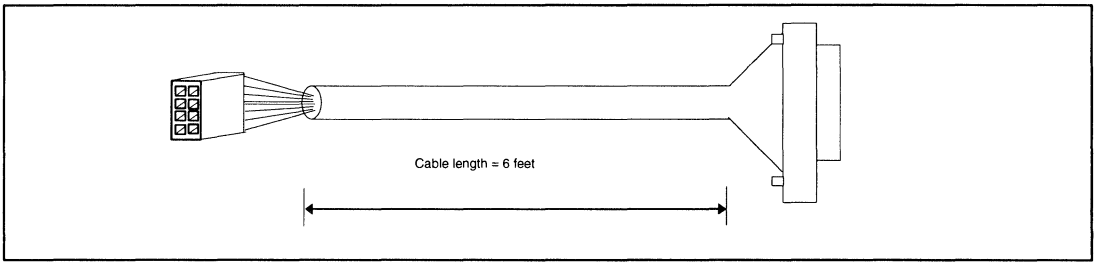


Figure 1. ISP Download Cable

Table 1. CoolRunner Download Cable Components

Components
DB 25 pin male connector
10 pin female connector (2x5) with plug on connector #10
10 stran .050" Ribbon Cable (length = 6 feet)
4 100 ohm resisters (Placed on TCK, TMS, TDI, and TDO near the DB25 Connector)

Table 2 gives the download cable pin description.

Table 2. CoolRunner Download Cable Pin Description

Pins	Name	Description
TCK	Test Clock Output	Clock pin to shift the serial data and instructions in and out of the TDI and TDO pins, respectively. TCK is also used to clock the TAP Controller state machine.
TMS	Test Mode Select	Serial input pin selects the JTAG instruction mode. TMS should be driven high during user mode operation.
TDI	Test Data Input	Serial input pin for instructions and test data. Data is shifted in on the rising edge of TCK.
TDO	Test Data Output	Serial output pin for instructions and test data. Data is shifted out on the falling edge of TCK. The signal is tri–stated if data is not being shifted out of the device.
GND	Ground	Ground

ISP Download Cable Specification

Figure 2 gives the ISP Download Cable connections.

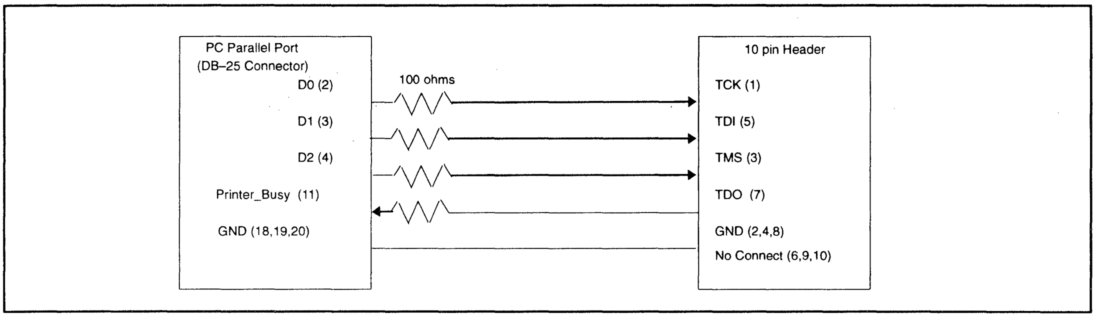


Figure 2. Download Cable Connections

Figure 3 gives the 10 pin header pin orientation from a cable header point of view.

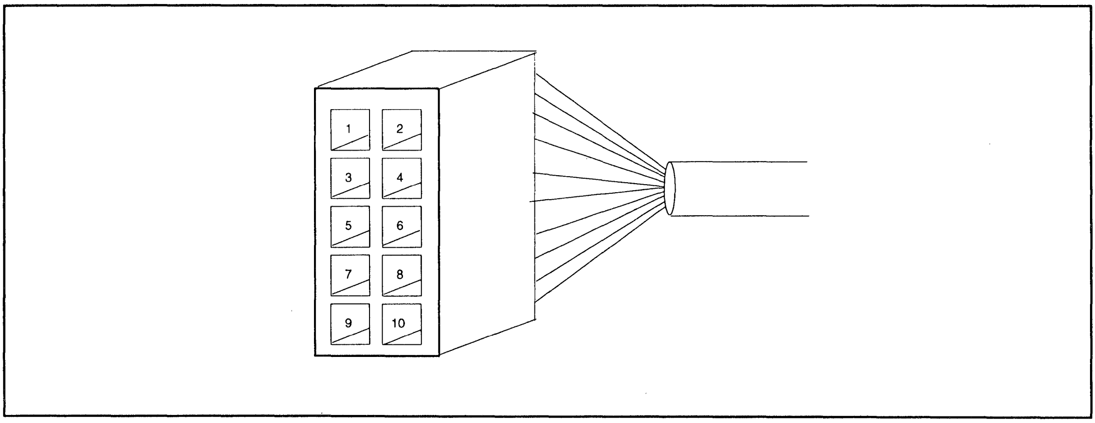


Figure 3. 10 Pin Header Pin Orientation (From a Cable Header Point of View)

Section 4
CPLD Datasheets

CONTENTS

32 macrocell CPLD PZ3032

FEATURES

- Industry's first TotalCMOS™ PLD – both CMOS design and process technologies
- Fast Zero Power (FZP™) design technique provides ultra-low power and very high speed
- High speed pin-to-pin delays of 8ns
- Ultra-low static power of less than 35µA
- Dynamic power that is 70% lower at 50MHz than competing devices
- 100% routable with 100% utilization while all pins and all macrocells are fixed
- Deterministic timing model that is extremely simple to use
- 2 clocks with programmable polarity at every macrocell
- Support for complex asynchronous clocking
- Innovative XPLA™ architecture combines high speed with extreme flexibility
- 1000 erase/program cycles guaranteed
- 20 years data retention guaranteed
- Logic expandable to 37 product terms
- PCI compliant
- Advanced 0.5µ E^2CMOS process
- Security bit prevents unauthorized access
- Design entry and verification using industry standard and Philips CAE tools
- Reprogrammable using industry standard device programmers
- Innovative Control Term structure provides either sum terms or product terms in each logic block for:
  - Programmable 3-State buffer
  - Asynchronous macrocell register preset/reset
- Programmable global 3-State pin facilitates 'bed of nails' testing without using logic resources
- Available in both PLCC and TQFP packages

Table 1. PZ3032 Features

	PZ3032
Usable gates	1000
Maximum inputs	36
Maximum I/Os	32
Number of macrocells	32
I/O macrocells	32
Buried macrocells	0
Propagation delay (ns)	8.0
Packages	44-pin PLCC, 44-pin TQFP

DESCRIPTION

The PZ3032 CPLD (Complex Programmable Logic Device) is the first in a family of Fast Zero Power (FZP™) CPLDs from Philips Semiconductors. These devices combine high speed and zero power in a 32 macrocell CPLD. With the FZP™ design technique, the PZ3032 offers true pin-to-pin speeds of 8ns, while simultaneously delivering power that is less than 35µA at standby without the need for 'turbo bits' or other power down schemes. By replacing conventional sense amplifier methods for implementing product terms (a technique that has been used in PLDs since the bipolar era) with a cascaded chain of pure CMOS gates, the dynamic power is also substantially lower than any competing CPLD – 70% lower at 50MHz. These devices are the first TotalCMOS™ PLDs, as they use both a CMOS process technology **and** the patented full CMOS FZP™ design technique. For 5V applications, Philips also offers the high speed PZ5032 CPLD that offers pin-to-pin speeds of 6ns.

The Philips FZP™ CPLDs introduce the new patent-pending XPLA™ (eXtended Programmable Logic Array) architecture. The XPLA™ architecture combines the best features of both PLA and PAL™ type structures to deliver high speed and flexible logic allocation that results in superior ability to make design changes with fixed pinouts. The XPLA™ structure in each logic block provides a fast 8ns PAL™ path with 5 dedicated product terms per output. This PAL™ path is joined by an additional PLA structure that deploys a pool of 32 product terms to a fully programmable OR array that can allocate the PLA product terms to any output in the logic block. This combination allows logic to be allocated efficiently throughout the logic block and supports as many as 37 product terms on an output. The speed with which logic is allocated from the PLA array to an output is only 2.5ns, regardless of the number of PLA product terms used, which results in worst case t_{PD}'s of only 10.5ns from any pin to any other pin. In addition, logic that is common to multiple outputs can be placed on a single PLA product term and shared across multiple outputs via the OR array, effectively increasing design density.

The PZ3032 CPLDs are supported by industry standard CAE tools (Cadence, Mentor, Synopsys, Synario, Viewlogic, OrCAD), using text (Abel, VHDL, Verilog) and/or schematic entry. Design verification uses industry standard simulators for functional and timing simulation. Development is supported on personal computer, Sparc, and HP platforms. Device fitting uses either Minc or Philips Semiconductors-developed tools.

The PZ3032 CPLD is reprogrammable using industry standard device programmers from vendors such as Data I/O, BP Microsystems, SMS, and others.

PAL is a registered trademark of Advanced Micro Devices, Inc.

ORDERING INFORMATION

ORDER CODE	DESCRIPTION	DESCRIPTION	DRAWING NUMBER
PZ3032–8A44	44-pin PLCC, 8ns t_{PD}	Commercial temp range, 3.3 volt power supply, ± 10%	SOT187-2
PZ3032–10A44	44-pin PLCC, 10ns t_{PD}	Commercial temp range, 3.3 volt power supply, ± 10%	SOT187-2
PZ3032–12A44	44-pin PLCC, 12ns t_{PD}	Commercial temp range, 3.3 volt power supply, ± 10%	SOT187-2
PZ3032I10A44	44-pin PLCC, 10ns t_{PD}	Industrial temp range, 3.3 volt power supply, ± 10%	SOT187-2
PZ3032I12A44	44-pin PLCC, 12ns t_{PD}	Industrial temp range, 3.3 volt power supply, ± 10%	SOT187-2
PZ3032–8BC	44-pin TQFP, 8ns t_{PD},	Commercial temp range, 3.3 volt power supply, ± 10%	SOT376-1
PZ3032–10BC	44-pin TQFP, 10ns t_{PD}	Commercial temp range, 3.3 volt power supply, ± 10%	SOT376-1
PZ3032–12BC	44-pin TQFP, 12ns t_{PD}	Commercial temp range, 3.3 volt power supply, ± 10%	SOT376-1
PZ3032I10BC	44-pin TQFP, 10ns t_{PD}	Industrial temp range, 3.3 volt power supply, ± 10%	SOT376-1
PZ3032I12BC	44-pin TQFP, 12ns t_{PD}	Industrial temp range, 3.3 volt power supply, ± 10%	SOT376-1

XPLA™ ARCHITECTURE

Figure 1 shows a high level block diagram of a 64 macrocell device implementing the XPLA™ architecture. The XPLA™ architecture consists of logic blocks that are interconnected by a Zero-power Interconnect Array (ZIA). The ZIA is a virtual crosspoint switch. Each logic block is essentially a 36V16 device with 36 inputs from the ZIA and 16 macrocells. Each logic block also provides 32 ZIA feedback paths from the macrocells and I/O pins.

From this point of view, this architecture looks like many other CPLD architectures. What makes the CoolRunner™ family unique is what is inside each logic block and the design technique used to implement these logic blocks. The contents of the logic block will be described next.

Logic Block Architecture

Figure 2 illustrates the logic block architecture. Each logic block contains control terms, a PAL array, a PLA array, and 16 macrocells. The 6 control terms can individually be configured as either SUM or PRODUCT terms, and are used to control the preset/reset and output enables of the 16 macrocells' flip-flops. The PAL array consists of a programmable AND array with a fixed OR array, while the PLA array consists of a programmable AND array with a programmable OR array. The PAL array provides a high speed path through the array, while the PLA array provides increased product term density.

Each macrocell has 5 dedicated product terms from the PAL array. The pin-to-pin t_{PD} of the PZ3032 device through the PAL array is 8ns. This performance is the fastest 3 volt CPLD available today. If a macrocell needs more than 5 product terms, it simply gets the additional product terms from the PLA array. The PLA array consists of 32 product terms, which are available for use by all 16 macrocells. The additional propagation delay incurred by a macrocell using 1 or all 32 PLA product terms is just 2.5ns. So the total pin-to-pin t_{PD} for the PZ3032 using 6 to 37 product terms is 10.5ns (8ns for the PAL + 2.5ns for the PLA).

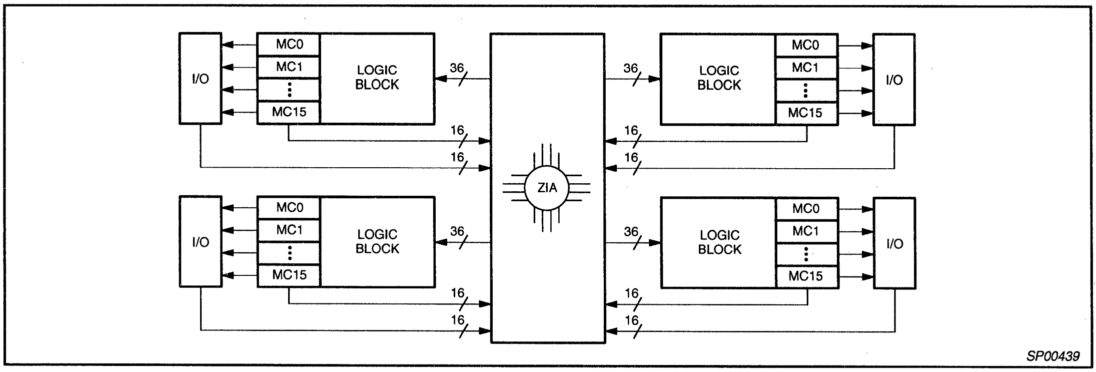


Figure 1. Philips XPLA CPLD Architecture

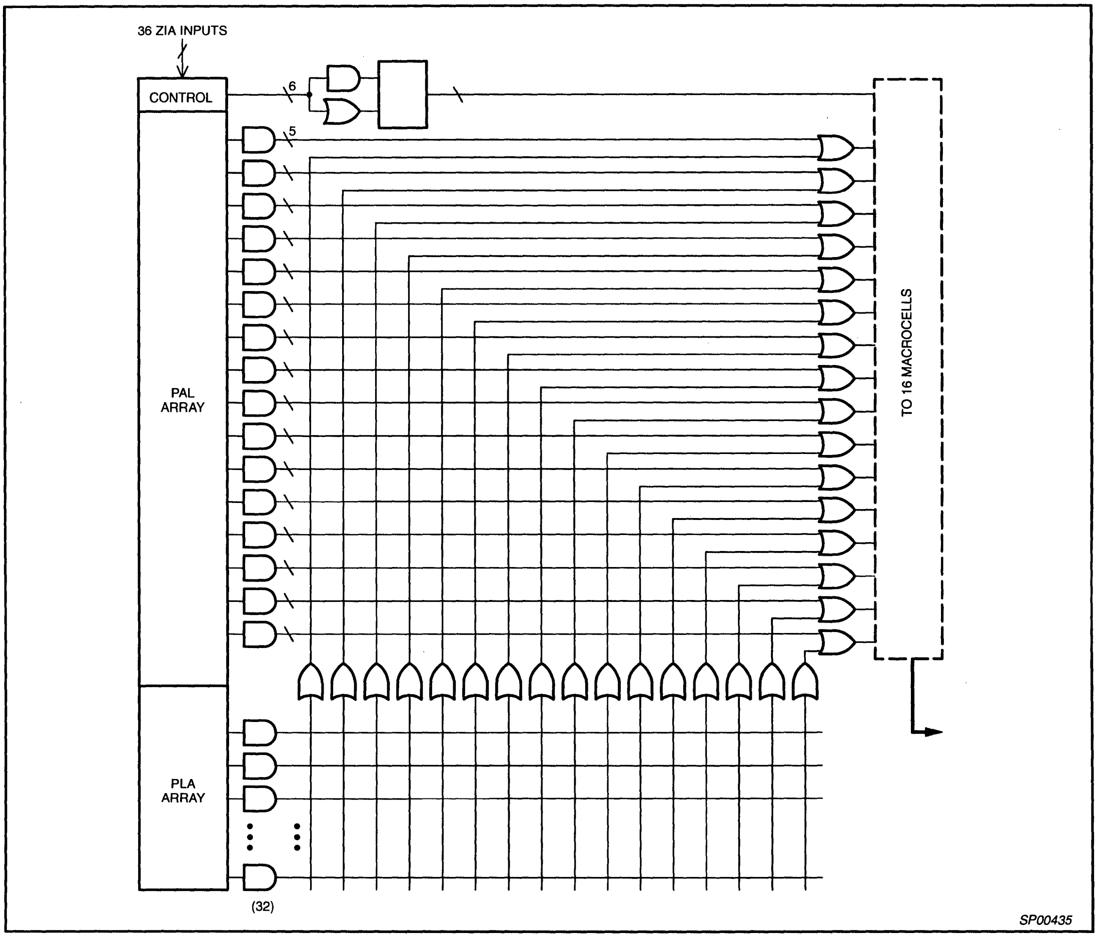


Figure 2. Philips Logic Block Architecture

Macrocell Architecture

Figure 3 shows the architecture of the macrocell used in the CoolRunner™ family. The macrocell consists of a flip-flop that can be configured as either a D or T type. A D-type flip-flop is generally more useful for implementing state machines and data buffering. A T-type flip-flop is generally more useful in implementing counters. All CoolRunner™ family members provide both synchronous and asynchronous clocking and provide the ability to clock off either the falling or rising edges of these clocks. These devices are designed such that the skew between the rising and falling edges of a clock are minimized for clocking integrity. There are 2 clocks (CLK0 and CLK1) available on the PZ3032 device. Clock 0 (CLK0) is designated as the "synchronous" clock and must be driven by an external source. Clock 1 (CLK1) can either be used as a synchronous clock (driven by an external source) or as an asynchronous clock (driven by a macrocell equation).

Two of the control terms (CT0 and CT1) are used to control the Preset/Reset of the macrocell's flip-flop. The Preset/Reset feature for each macrocell can also be disabled. Note that the Power-on Reset leaves all macrocells in the "zero" state when power is properly applied. The other 4 control terms (CT2–CT5) can be used to control the Output Enable of the macrocell's output buffers. The reason there are as many control terms dedicated for the Output Enable of the macrocell is to insure that all CoolRunner™ devices are PCI compliant. The macrocell's output buffers can also be always enabled or disabled. All CoolRunner™ devices also provide a Global Tri-State ($\overline{\text{GTS}}$) pin, which, when pulled Low, will 3-State all the outputs of the device. This pin is provided to support "In-Circuit Testing" or "Bed-of-Nails Testing".

There are two feedback paths to the ZIA: one from the macrocell, and one from the I/O pin. The ZIA feedback path before the output buffer is the macrocell feedback path, while the ZIA feedback path after the output buffer is the I/O pin ZIA path. When the macrocell is used as an output, the output buffer is enabled, and the macrocell feedback path can be used to feedback the logic implemented in the macrocell. When the I/O pin is used as an input, the output buffer will be 3-Stated and the input signal will be fed into the ZIA via the I/O feedback path, and the logic implemented in the buried macrocell can be fed back to the ZIA via the macrocell feedback path. It should be noted that unused inputs or I/Os should be properly terminated.

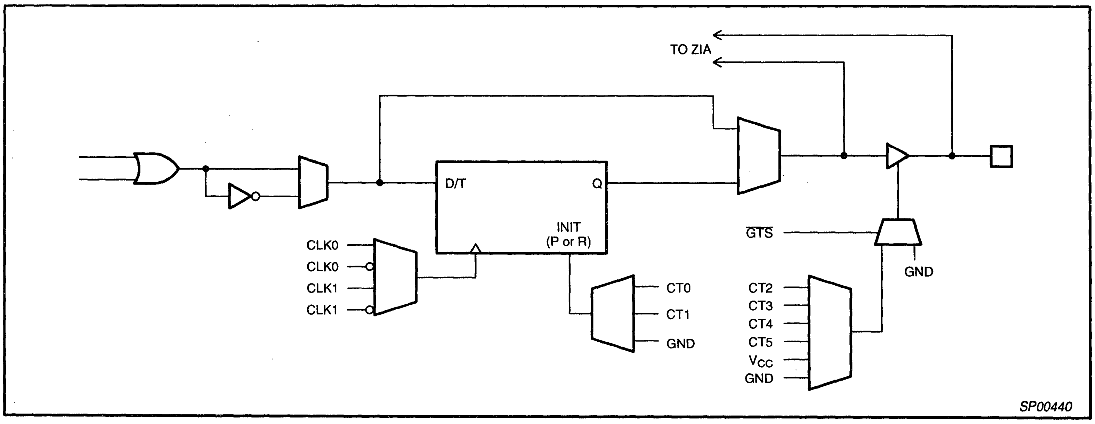


Figure 3. PZ3032 Macrocell Architecture

Simple Timing Model

Figure 4 shows the CoolRunner™ Timing Model. The CoolRunner™ timing model looks very much like a 22V10 timing model in that there are three main timing parameters, including t_{PD}, t_{SU}, and t_{CO}. In other competing architectures, the user may be able to fit the design into the CPLD, but is not sure whether system timing requirements can be met until after the design has been fit into the device. This is because the timing models of competing architectures are very complex and include such things as timing dependencies on the number of parallel expanders borrowed, sharable expanders, varying number of X and Y routing channels used, etc. In the XPLA™ architecture, the user knows up front whether the design will meet system timing requirements. This is due to the simplicity of the timing model. For example, in the PZ3032 device, the user knows up front that if a given output uses 5 product terms or less, the t_{PD} = 8ns, the t_{SU} = 6.5ns, and the t_{CO} = 7.5ns. If an output is using 6 to 37 product terms, an additional 2.5ns must be added to the t_{PD} and t_{SU} timing parameters to account for the time to propagate through the PLA array.

TotalCMOS™ Design Technique for Fast Zero Power

Philips is the first to offer a TotalCMOS™ CPLD, both in process technology and design technique. Philips employs a cascade of CMOS gates to implement its Sum of Products instead of the traditional sense amp approach. This CMOS gate implementation allows Philips to offer CPLDs which are both high performance and low power, breaking the paradigm that to have low power, you must have low performance. Refer to Figure 5 and Table 2 showing the I_{DD} vs. Frequency of our PZ3032 TotalCMOS™ CPLD.

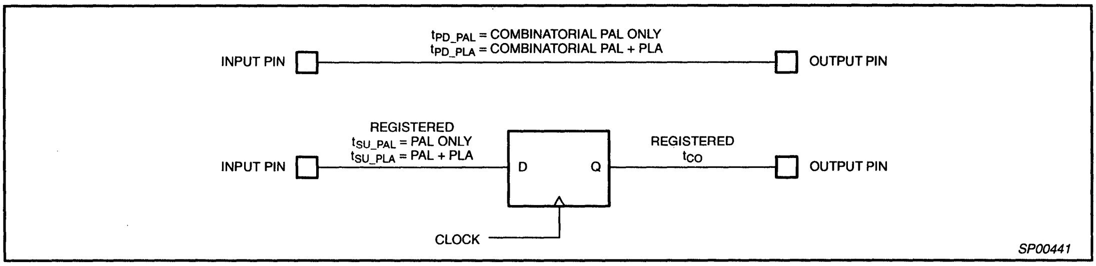


Figure 4. CoolRunner™ Timing Model

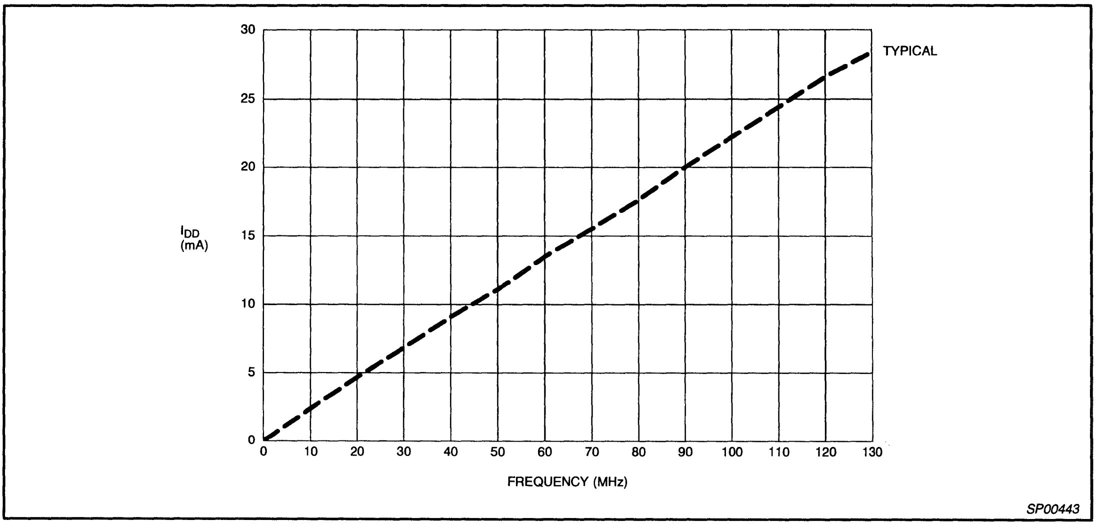


Figure 5. I_{DD} vs. Frequency @ V_{DD} = 3.3V

Table 2. I_{DD} vs Frequency

V_{DD} = 3.3V

FREQ (MHz)	0	10	20	30	40	50	60	70	80	90	100	110	120	130
Typical I_{DD} (mA)	0.01	2.37	4.65	6.80	9.06	11.1	13.5	15.5	17.4	20.0	22.1	24.4	26.6	28.5

ABSOLUTE MAXIMUM RATINGS[1]

SYMBOL	PARAMETER	MIN.	MAX.	UNIT
V_{DD}	Supply voltage	–0.5	7.0	V
V_I	Input voltage	–1.2	V_{DD}+0.5	V
V_{OUT}	Output voltage	–0.5	V_{DD}+0.5	V
I_{IN}	Input current	–30	30	mA
I_{OUT}	Output current	–100	100	mA
T_J	Maximum junction temperature	–40	150	°C
T_{str}	Storage temperature	–65	150	°C

NOTES:

1. Stresses above those listed may cause malfunction or permanent damage to the device. This is a stress rating only. Functional operation at these or any other condition above those indicated in the operational and programming specification is not implied.

OPERATING RANGE

PRODUCT GRADE	TEMPERATURE	VOLTAGE
Commercial	0 to +70°C	3.3 ±10% V
Industrial	–40 to +85°C	3.3 ±10% V

DC ELECTRICAL CHARACTERISTICS FOR COMMERCIAL GRADE DEVICES

Commercial: 0°C ≤ T_{amb} ≤ +70°C; 3.0V ≤ V_{DD} ≤ 3.6V

SYMBOL	PARAMETER	TEST CONDITIONS	MIN.	MAX.	UNIT
V_{IL}	Input voltage low	V_{DD} = 3.0V		0.8	V
V_{IH}	Input voltage high	V_{DD} = 3.6V	2.0		V
V_I	Input clamp voltage	V_{DD} = 3.0V, I_{IN} = –18mA		–1.2	V
V_{OL}	Output voltage low	V_{DD} = 3.0V, I_{OL} = 8mA		0.5	V
V_{OH}	Output voltage high	V_{DD} = 3.0V, I_{OH} = –8mA	2.4		V
I_{IL}	Input leakage current low	V_{DD} = 3.6V (except CKO), V_{IN} = 0V	–10	10	µA
I_{IH}	Input leakage current high	V_{DD} = 3.6V, V_{IN} = 3.0V	–10	10	µA
I_{IL}	Clock input leakage current	V_{DD} = 3.6V, V_{IN} = 0.4V	–10	10	µA
I_{OZL}	3-Stated output leakage current low	V_{DD} = 3.6V, V_{IN} = 0.4V	–10	10	µA
I_{OZH}	3-Stated output leakage current high	V_{DD} = 3.6V, V_{IN} = 3.0V	–10	10	µA
I_{DDQ}	Standby current	V_{DD} = 3.6V, T_{amb} = 0°C		35	µA
I_{DDD}[1]	Dynamic current	V_{DD} = 3.6V, T_{amb} = 0°C @ 1MHz		0.5	mA
		V_{DD} = 3.6V, T_{amb} = 0°C @ 50MHz		18	mA
I_{OS}	Short circuit output current	1 pin at a time for no longer than 1 second	–5	–100	mA
C_{IN}	Input pin capacitance	T_{amb} = 25°C, f = 1MHz		8	pF
C_{CLK}	Clock input capacitance	T_{amb} = 25°C, f = 1MHz	5	12	pF
$C_{I/O}$	I/O pin capacitance	T_{amb} = 25°C, f = 1MHz		10	pF

NOTE:

1. This parameter measured with a 16–bit, loadable up/down counter loaded into every logic block, with all outputs enabled and unloaded. Inputs are tied to V_{DD} or ground. This parameter guaranteed by design and characterization, not testing.

AC ELECTRICAL CHARACTERISTICS[1] FOR COMMERCIAL GRADE DEVICES

Commercial: 0°C ≤ T_{amb} ≤ +70°C; 3.0V ≤ V_{DD} ≤ 3.6V

SYMBOL	PARAMETER	–8		–10		–12		UNIT
		MIN.	MAX.	MIN.	MAX.	MIN.	MAX.	
t_{PD_PAL}	Propagation delay time, input (or feedback node) to output through PAL	2	8	2	10	2	12	ns
t_{PD_PLA}	Propagation delay time, input (or feedback node) to output through PAL & PLA	3	10.5	3	13	3	15	ns
t_{CO}	Clock to out delay time	2	7	2	9	2	11	ns
t_{SU_PAL}	Setup time (from input or feedback node) through PAL	6.5		8.5		10.5		ns
t_{SU_PLA}	Setup time (from input or feedback node) through PAL + PLA	9		11.5		13.5		ns
t_H	Hold time		0		0		0	ns
t_{CH}	Clock High time	3		4		5		ns
t_{CL}	Clock Low time	3		4		5		ns
t_R	Input rise time		20		20		20	ns
t_F	Input fall time		20		20		20	ns
f_{MAX1}	Maximum FF toggle rate[2] (1/t_{CH} + t_{CL})	167		125		100		MHz
f_{MAX2}	Maximum internal frequency[2] (1/t_{SUPAL} + t_{CF})	83		63		50		MHz
f_{MAX3}	Maximum external frequency[2] (1/t_{SUPAL} + t_{CO})	74		57		47		MHz
t_{BUF}	Output buffer delay time		1.5		1.5		1.5	ns
t_{PDF_PAL}	Input (or feedback node) to internal feedback node delay time through PAL		6.5		8.5		10.5	ns
t_{PDF_PLA}	Input (or feedback node) to internal feedback node delay time through PAL + PLA		9		11.5		13.5	ns
t_{CF}	Clock to internal feedback node delay time		5.5		7.5		9.5	ns
t_{INIT}	Delay from valid V_{DD} to valid reset		50		50		50	µs
t_{ER}	Input to output disable[3]		15		17		19	ns
t_{EA}	Input to output valid		15		17		19	ns
t_{RP}	Input to register preset		16		18		20	ns
t_{RR}	Input to register reset		19		21		23	ns

NOTES:

1. Specifications measured with one output switching. See Figure 6 and Table 3 for derating.
2. This parameter guaranteed by design and characterization, not by test.
3. Output C_L = 5pF.

DC ELECTRICAL CHARACTERISTICS FOR INDUSTRIAL GRADE DEVICES

Industrial: –40°C ≤ T_{amb} ≤ +85°C; 3.0V ≤ V_{DD} ≤ 3.6V

SYMBOL	PARAMETER	TEST CONDITIONS	MIN.	MAX.	UNIT
V_{IL}	Input voltage low	V_{DD} = 3.0V		0.8	V
V_{IH}	Input voltage high	V_{DD} = 3.6V	2.0		V
V_I	Input clamp voltage	V_{DD} = 3.0V, I_{IN} = –18mA		–1.2	V
V_{OL}	Output voltage low	V_{DD} = 3.0V, I_{OL} = 8mA		0.5	V
V_{OH}	Output voltage high	V_{DD} = 3.0V, I_{OH} = –8mA	2.4		V
I_{IL}	Input leakage current low	V_{DD} = 3.6V (except CKO), V_{IN} = 0.4V	–10	10	µA
I_{IH}	Input leakage current high	V_{DD} = 3.6V, V_{IN} = 3.0V	–10	10	µA
I_{IL}	Clock input leakage current	V_{DD} = 3.6V, V_{IN} = 0.4V	–10	10	µA
I_{OZL}	3-Stated output leakage current low	V_{DD} = 3.6V, V_{IN} = 0.4V	–10	10	µA
I_{OZH}	3-Stated output leakage current high	V_{DD} = 3.6V, V_{IN} = 3.0V	–10	10	µA
I_{DDQ}	Standby current	V_{DD} = 3.6V, T_{amb} = –40°C		45	µA
I_{DDD}[1]	Dynamic current	V_{DD} = 3.6V, T_{amb} = –40°C @ 1MHz		0.5	mA
		V_{DD} = 3.6V, T_{amb} = –40°C @ 50MHz		18	mA
I_{OS}	Short circuit output current	1 pin at a time for no longer than 1 second	–5	–120	mA
C_{IN}	Input pin capacitance	T_{amb} = 25°C, f = 1MHz		8	pF
C_{CLK}	Clock input capacitance	T_{amb} = 25°C, f = 1MHz	5	12	pF
$C_{I/O}$	I/O pin capacitance	T_{amb} = 25°C, f = 1MHz		10	pF

NOTE:

1. This parameter measured with a 16–bit, loadable up/down counter loaded into every logic block, with all outputs enabled and unloaded. Inputs are tied to V_{DD} or ground. This parameter guaranteed by design and characterization, not testing.

AC ELECTRICAL CHARACTERISTICS[1] FOR INDUSTRIAL GRADE DEVICES

Industrial: –40°C ≤ T_{amb} ≤ +85°C; 3.0V ≤ V_{DD} ≤ 3.6V

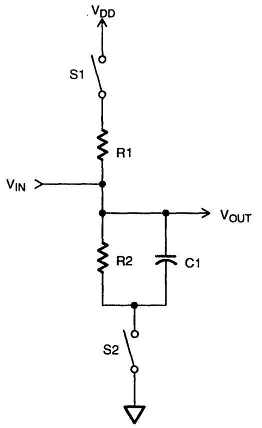
SYMBOL	PARAMETER	I10		I12		UNIT
		MIN.	MAX.	MIN.	MAX.	
t_{PD_PAL}	Propagation delay time, input (or feedback node) to output through PAL	2	10	2	12	ns
t_{PD_PLA}	Propagation delay time, input (or feedback node) to output through PAL & PLA	3	12.5	3	15	ns
t_{CO}	Clock to out delay time	2	9	2	11	ns
t_{SU_PAL}	Setup time (from input or feedback node) through PAL	8		10.5		ns
t_{SU_PLA}	Setup time (from input or feedback node) through PAL + PLA	10.5		13.5		ns
t_H	Hold time		0		0	ns
t_{CH}	Clock High time	4		5		ns
t_{CL}	Clock Low time	4		5		ns
t_R	Input rise time		20		20	ns
t_F	Input fall time		20		20	ns
f_{MAX1}	Maximum FF toggle rate[2] (1/t_{CH} + t_{CL})	125		100		MHz
f_{MAX2}	Maximum internal frequency[2] (1/t_{SUPAL} + t_{CF})	64.5		50		MHz
f_{MAX3}	Maximum external frequency[2] (1/t_{SUPAL} + t_{CO})	58.8		47		MHz
t_{BUF}	Output buffer delay time		1.5		1.5	ns
t_{PDF_PAL}	Input (or feedback node) to internal feedback node delay time through PAL		8		10.5	ns
t_{PDF_PLA}	Input (or feedback node) to internal feedback node delay time through PAL + PLA		10.5		13.5	ns
t_{CF}	Clock to internal feedback delay time		7.5		9.5	ns
t_{INIT}	Delay from valid V_{DD} to valid reset		50		50	µs
t_{ER}	Input to output disable[3]		16		19	ns
t_{EA}	Input to output valid		16		19	ns
t_{RP}	Input to register preset		17		20	ns
t_{RR}	Input to register reset		20		23	ns

NOTES:

1. Specifications measured with one output switching. See Figure 6 and Table 3 for derating.
2. This parameter guaranteed by design and characterization, not by test.
3. Output C_L = 5pF.

SWITCHING CHARACTERISTICS

The test load circuit and load values for the AC Electrical Characteristics are illustrated below.



COMPONENT	VALUES
R1	390Ω
R2	390Ω
C1	35pF

MEASUREMENT	S1	S2
t_{PZH}	Open	Closed
t_{PZL}	Closed	Closed
t_P	Closed	Closed

NOTE: For t_{PHZ} and t_{PLZ} C = 5pF, and 3-State levels are measured 0.5V from steady-state active level.

SP00477

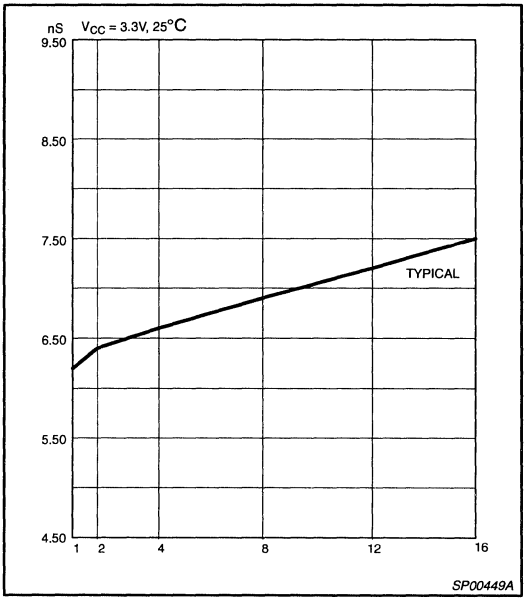
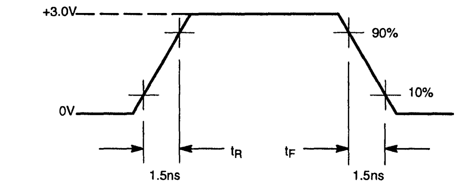


Figure 6. t_{PD_PAL} vs. Outputs switching

VOLTAGE WAVEFORM



MEASUREMENTS:
All circuit delays are measured at the +1.5V level of inputs and outputs, unless otherwise specified.

Input Pulses

SP00368

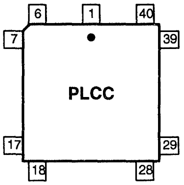
Table 3. t_{PD_PAL} vs. # of Outputs switching

V_{DD} = 3.30V

# of Outputs	1	2	4	8	12	16
Typical (ns)	6.2	6.4	6.6	6.9	7.2	7.5

PIN DESCRIPTIONS

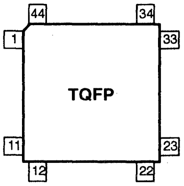
PZ3032 – 44-Pin Plastic Leaded Chip Carrier



Pin	Function	Pin	Function	Pin	Function
1	IN1	16	I/O–A10	31	I/O–B9
2	IN3	17	I/O–A11	32	I/O–B8
3	V_{DD}	18	I/O–A12	33	I/O–B7
4	I/O–A0–CK1	19	I/O–A13	34	I/O–B6
5	I/O–A1	20	I/O–A14	35	V_{DD}
6	I/O–A2	21	I/O–A15	36	I/O–B5
7	I/O–A3	22	GND	37	I/O–B4
8	I/O–A4	23	V_{DD}	38	I/O–B3
9	I/O–A5	24	I/O–B15	39	I/O–B2
10	GND	25	I/O–B14	40	I/O–B1
11	I/O–A6	26	I/O–B13	41	I/O–B0
12	I/O–A7	27	I/O–B12	42	GND
13	I/O–A8	28	I/O–B11	43	IN0–CK0
14	I/O–A9	29	I/O–B10	44	IN2–gtsn
15	V_{DD}	30	GND		

SP00420

PZ3032 – 44-Pin Thin Quad Flat Package



Pin	Function	Pin	Function	Pin	Function
1	I/O–A3	16	GND	31	I/O–B4
2	I/O–A4	17	V_{DD}	32	I/O–B3
3	I/O–A5	18	I/O–B15	33	I/O–B2
4	GND	19	I/O–B14	34	I/O–B1
5	I/O–A6	20	I/O–B13	35	I/O–B0
6	I/O–A7	21	I/O–B12	36	GND
7	I/O–A8	22	I/O–B11	37	IN0/CK0
8	I/O–A9	23	I/O–B10	38	IN2–gtsn
9	V_{DD}	24	GND	39	IN1
10	I/O–A10	25	I/O–B9	40	IN3
11	I/O–A11	26	I/O–B8	41	V_{DD}
12	I/O–A12	27	I/O–B7	42	I/O–A0–CK1
13	I/O–A13	28	I/O–B6	43	I/O–A1
14	I/O–A14	29	V_{DD}	44	I/O–A2
15	I/O–A15	30	I/O–B5		

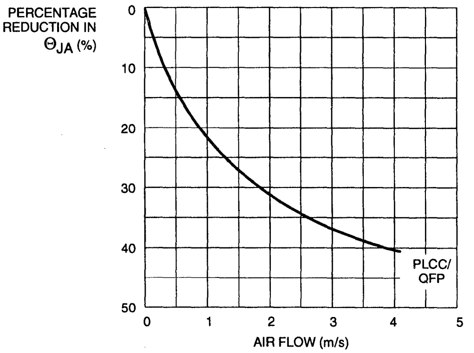
SP00433

Package Thermal Characteristics

Philips Semiconductors uses the Temperature Sensitive Parameter (TSP) method to test thermal resistance. This method meets Mil-Std-883C Method 1012.1 and is described in Philips *1995 IC Package Databook*. Thermal resistance varies slightly as a function of input power. As input power increases, thermal resistance changes approximately 5% for a 100% change in power.

Figure 7 is a derating curve for the change in Θ_{JA} with airflow based on wind tunnel measurements. It should be noted that the wind flow dynamics are more complex and turbulent in actual applications than in a wind tunnel. Also, the test boards used in the wind tunnel contribute significantly to forced convection heat transfer, and may not be similar to the actual circuit board, especially in size.

Package	Θ_{JA}
44-pin PLCC	49.8°C/W
44-pin TQFP	66.3°C/W



SP00419A

Figure 7. Average Effect of Airflow on Θ_{JA}

32 macrocell CPLD

PZ5032

FEATURES

- Industry's first TotalCMOS™ PLD – both CMOS design and process technologies
- Fast Zero Power (FZP™) design technique provides ultra-low power and very high speed
- High speed pin-to-pin delays of 6ns
- Ultra-low static power of less than 75µA
- Dynamic power that is 70% lower at 50MHz than competing devices
- 100% routable with 100% utilization while all pins and all macrocells are fixed
- Deterministic timing model that is extremely simple to use
- 2 clocks with programmable polarity at every macrocell
- Support for complex asynchronous clocking
- Innovative XPLA™ architecture combines high speed with extreme flexibility
- 1000 erase/program cycles guaranteed
- 20 years data retention guaranteed
- Logic expandable to 37 product terms
- PCI compliant
- Advanced 0.5µ E^2CMOS process
- Security bit prevents unauthorized access
- Design entry and verification using industry standard and Philips CAE tools
- Reprogrammable using industry standard device programmers
- Innovative Control Term structure provides either sum terms or product terms in each logic block for:
  - Programmable 3-State buffer
  - Asynchronous macrocell register preset/reset
- Programmable global 3-State pin facilitates 'bed of nails' testing without using logic resources
- Available in both PLCC and TQFP packages
- Available in both Commercial and Industrial grades

Table 1. PZ5032 Features

	PZ5032
Usable gates	1000
Maximum inputs	36
Maximum I/Os	32
Number of macrocells	32
I/O macrocells	32
Buried macrocells	0
Propagation delay (ns)	6.0
Packages	44-pin PLCC, 44-pin TQFP

DESCRIPTION

The PZ5032 CPLD (Complex Programmable Logic Device) is the first in a family of Fast Zero Power (FZP™) CPLDs from Philips Semiconductors. These devices combine high speed and zero power in a 32 macrocell CPLD. With the FZP™ design technique, the PZ5032 offers true pin-to-pin speeds of 6ns, while simultaneously delivering power that is less than 75µA at standby without the need for 'turbo bits' or other power down schemes. By replacing conventional sense amplifier methods for implementing product terms (a technique that has been used in PLDs since the bipolar era) with a cascaded chain of pure CMOS gates, the dynamic power is also substantially lower than any competing CPLD – 70% lower at 50MHz. These devices are the first TotalCMOS™ PLDs, as they use both a CMOS process technology **and** the patented full CMOS FZP™ design technique. For 3V applications, Philips also offers the high speed PZ3032 CPLD that offers these features in a full 3V implementation.

The Philips FZP™ CPLDs introduce the new patent-pending XPLA™ (eXtended Programmable Logic Array) architecture. The XPLA™ architecture combines the best features of both PLA and PAL™ type structures to deliver high speed and flexible logic allocation that results in superior ability to make design changes with fixed pinouts. The XPLA™ structure in each logic block provides a fast 6ns PAL™ path with 5 dedicated product terms per output. This PAL™ path is joined by an additional PLA structure that deploys a pool of 32 product terms to a fully programmable OR array that can allocate the PLA product terms to any output in the logic block. This combination allows logic to be allocated efficiently throughout the logic block and supports as many as 37 product terms on an output. The speed with which logic is allocated from the PLA array to an output is only 2ns, regardless of the number of PLA product terms used, which results in worst case t_{PD}'s of only 8ns from any pin to any other pin. In addition, logic that is common to multiple outputs can be placed on a single PLA product term and shared across multiple outputs via the OR array, effectively increasing design density.

The PZ5032 CPLDs are supported by industry standard CAE tools (Cadence, Mentor, Synopsys, Synario, Viewlogic, OrCAD), using text (Abel, VHDL, Verilog) and/or schematic entry. Design verification uses industry standard simulators for functional and timing simulation. Development is supported on personal computer, Sparc, and HP platforms. Device fitting uses either Minc or Philips Semiconductors-developed tools.

The PZ5032 CPLD is reprogrammable using industry standard device programmers from vendors such as Data I/O, BP Microsystems, SMS, and others.

PAL is a registered trademark of Advanced Micro Devices, Inc.

ORDERING INFORMATION

ORDER CODE	DESCRIPTION	DESCRIPTION	DRAWING NUMBER
PZ5032–6A44	44-pin PLCC, 6ns t_{PD}	Commercial temp range, 5 volt power supply, ± 5%	SOT187-2
PZ5032–7A44	44-pin PLCC, 7.5ns t_{PD}	Commercial temp range, 5 volt power supply, ± 5%	SOT187-2
PZ5032–10A44	44-pin PLCC, 10ns t_{PD}	Commercial temp range, 5 volt power supply, ± 5%	SOT187-2
PZ5032I7A44	44-pin PLCC, 7.5ns t_{PD}	Industrial temp range, 5 volt power supply, ± 10%	SOT187-2
PZ5032I10A44	44-pin PLCC, 10ns t_{PD}	Industrial temp range, 5 volt power supply, ± 10%	SOT187-2
PZ5032–6BC	44-pin TQFP, 6ns t_{PD},	Commercial temp range, 5 volt power supply, ± 5%	SOT376-1
PZ5032–7BC	44-pin TQFP, 7.5ns t_{PD}	Commercial temp range, 5 volt power supply, ± 5%	SOT376-1
PZ5032–10BC	44-pin TQFP, 10ns t_{PD}	Commercial temp range, 5 volt power supply, ± 5%	SOT376-1
PZ5032I7BC	44-pin TQFP, 7.5ns t_{PD}	Industrial temp range, 5 volt power supply, ± 10%	SOT376-1
PZ5032I10BC	44-pin TQFP, 10ns t_{PD}	Industrial temp range, 5 volt power supply, ± 10%	SOT376-1

XPLA™ ARCHITECTURE

Figure 1 shows a high level block diagram of a 64 macrocell device implementing the XPLA™ architecture. The XPLA™ architecture consists of logic blocks that are interconnected by a Zero-power Interconnect Array (ZIA). The ZIA is a virtual crosspoint switch. Each logic block is essentially a 36V16 device with 36 inputs from the ZIA and 16 macrocells. Each logic block also provides 32 ZIA feedback paths from the macrocells and I/O pins.

From this point of view, this architecture looks like many other CPLD architectures. What makes the CoolRunner™ family unique is what is inside each logic block and the design technique used to implement these logic blocks. The contents of the logic block will be described next.

Logic Block Architecture

Figure 2 illustrates the logic block architecture. Each logic block contains control terms, a PAL array, a PLA array, and 16 macrocells. the 6 control terms can individually be configured as either SUM or PRODUCT terms, and are used to control the preset/reset and output enables of the 16 macrocells' flip-flops. The PAL array consists of a programmable AND array with a fixed OR array, while the PLA array consists of a programmable AND array with a programmable OR array. The PAL array provides a high speed path through the array, while the PLA array provides increased product term density.

Each macrocell has 5 dedicated product terms from the PAL array. The pin-to-pin t_{PD} of the PZ5032 device through the PAL array is 6ns. This performance is equivalent to the fastest 5 volt CPLD available today. If a macrocell needs more than 5 product terms, it simply gets the additional product terms from the PLA array. The PLA array consists of 32 product terms, which are available for use by all 16 macrocells. The additional propagation delay incurred by a macrocell using 1 or all 32 PLA product terms is just 2ns. So the total pin-to-pin t_{PD} for the PZ5032 using 6 to 37 product terms is 8ns (6ns for the PAL + 2ns for the PLA).

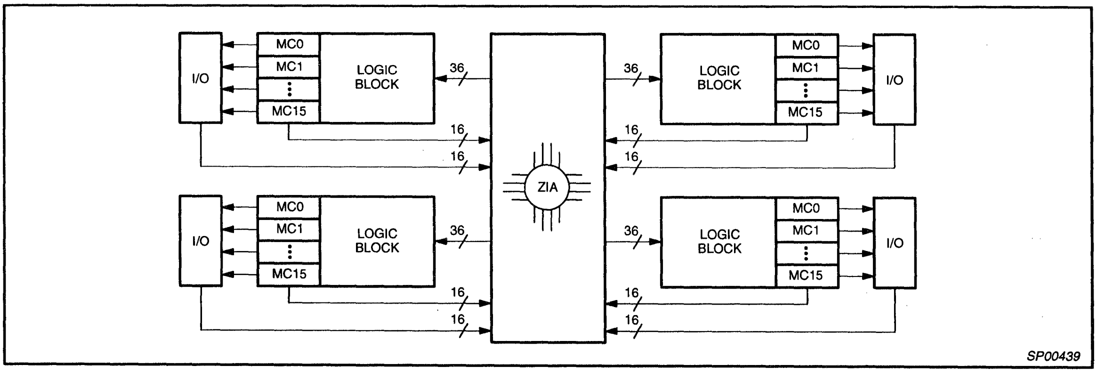


Figure 1. Philips XPLA CPLD Architecture

32 macrocell CPLD

PZ5032

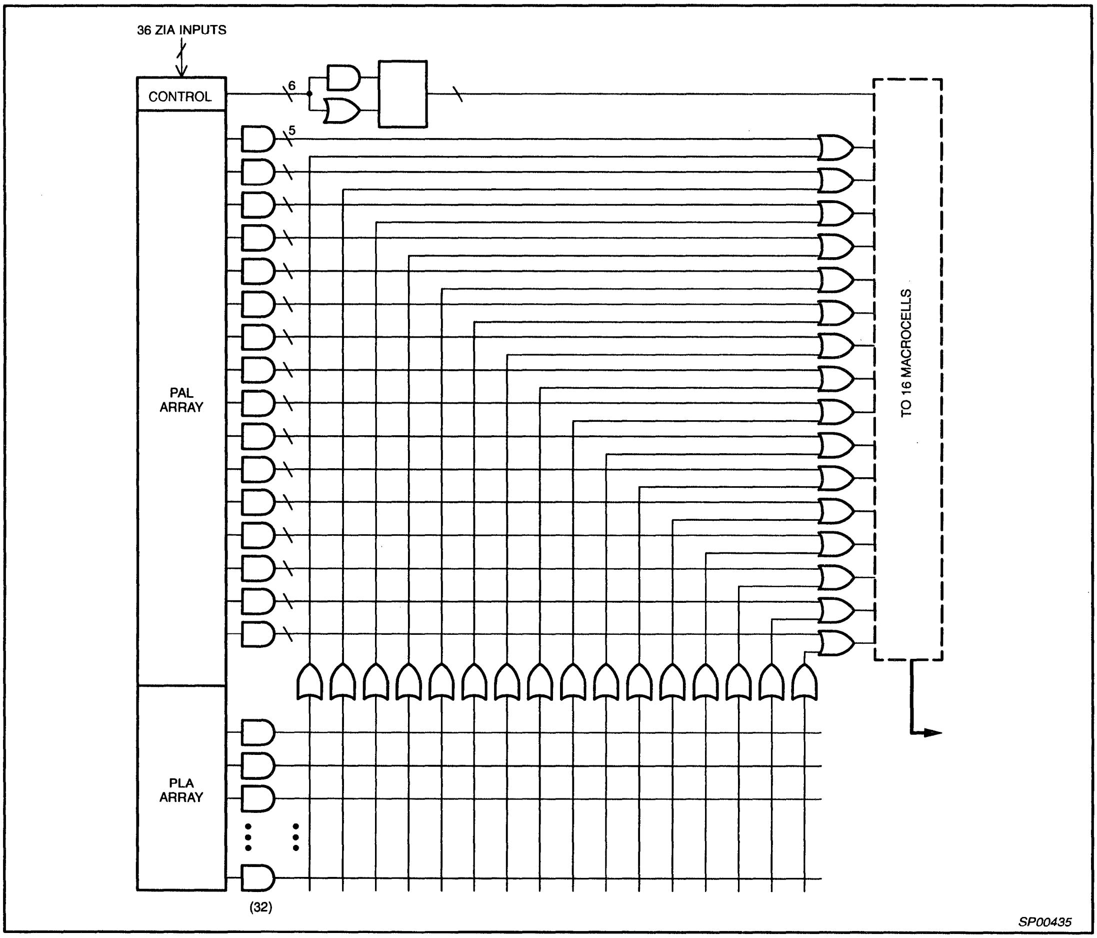


Figure 2. Philips Logic Block Architecture

Macrocell Architecture

Figure 3 shows the architecture of the macrocell used in the CoolRunner™ family. The macrocell consists of a flip-flop that can be configured as either a D or T type. A D-type flip-flop is generally more useful for implementing state machines and data buffering. A T-type flip-flop is generally more useful in implementing counters. All CoolRunner™ family members provide both synchronous and asynchronous clocking and provide the ability to clock off either the falling or rising edges of these clocks. These devices are designed such that the skew between the rising and falling edges of a clock are minimized for clocking integrity. There are 2 clocks (CLK0 and CLK1) available on the PZ5032 device. Clock 0 (CLK0) is designated as the "synchronous" clock and must be driven by an external source. Clock 1 (CLK1) can either be used as a synchronous clock (driven by an external source) or as an asynchronous clock (driven by a macrocell equation).

Two of the control terms (CT0 and CT1) are used to control the Preset/Reset of the macrocell's flip-flop. The Preset/Reset feature for each macrocell can also be disabled. Note that the Power-on Reset leaves all macrocells in the "zero" state when power is properly applied. The other 4 control terms (CT2–CT5) can be used to control the Output Enable of the macrocell's output buffers. The reason there are as many control terms dedicated for the Output Enable of the macrocell is to insure that all CoolRunner™ devices are PCI compliant. The macrocell's output buffers can also be always enabled or disabled. All CoolRunner™ devices also provide a Global Tri-State ($\overline{GTS}$) pin, which, when pulled Low, will 3-State all the outputs of the device. This pin is provided to support "In-Circuit Testing" or "Bed-of-Nails Testing".

There are two feedback paths to the ZIA: one from the macrocell, and one from the I/O pin. The ZIA feedback path before the output buffer is the macrocell feedback path, while the ZIA feedback path after the output buffer is the I/O pin ZIA path. When the macrocell is used as an output, the output buffer is enabled, and the macrocell feedback path can be used to feedback the logic implemented in the macrocell. When the I/O pin is used as an input, the output buffer will be 3-Stated and the input signal will be fed into the ZIA via the I/O feedback path, and the logic implemented in the buried macrocell can be fed back to the ZIA via the macrocell feedback path. It should be noted that unused inputs or I/Os should be properly terminated.

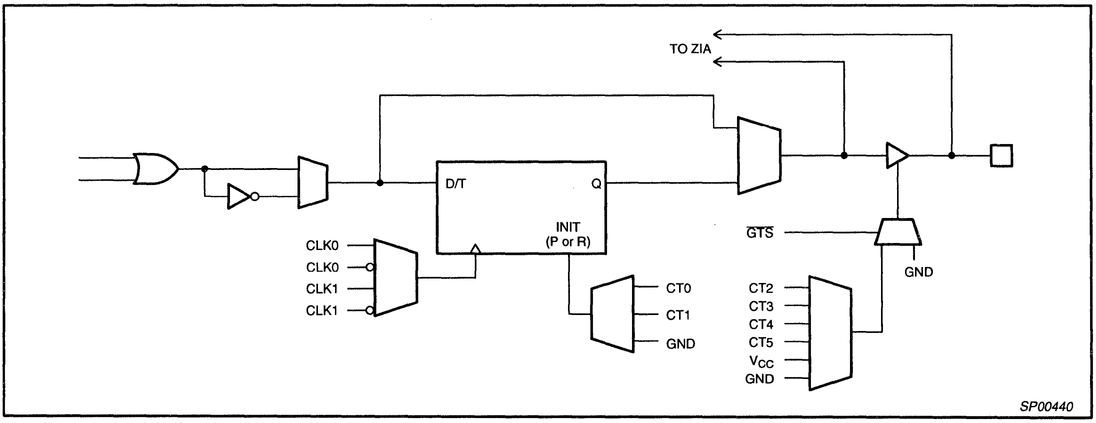


Figure 3. PZ5032 Macrocell Architecture

Simple Timing Model

Figure 4 shows the CoolRunner™ Timing Model. The CoolRunner™ timing model looks very much like a 22V10 timing model in that there are three main timing parameters, including t_{PD}, t_{SU}, and t_{CO}. In other competing architectures, the user may be able to fit the design into the CPLD, but is not sure whether system timing requirements can be met until after the design has been fit into the device. This is because the timing models of competing architectures are very complex and include such things as timing dependencies on the number of parallel expanders borrowed, sharable expanders, varying number of X and Y routing channels used, etc. In the XPLA™ architecture, the user knows up front whether the design will meet system timing requirements. This is due to the simplicity of the timing model. For example, in the PZ5032 device, the user knows up front that if a given output uses 5 product terms or less, the t_{PD} = 6ns, the t_{SU} = 4.5ns, and the t_{CO} = 5ns. If an output is using 6 to 37 product terms, an additional 2ns must be added to the t_{PD} and t_{SU} timing parameters to account for the time to propagate through the PLA array.

TotalCMOS™ Design Technique for Fast Zero Power

Philips is the first to offer a TotalCMOS™ CPLD, both in process technology and design technique. Philips employs a cascade of CMOS gates to implement its Sum of Products instead of the traditional sense amp approach. This CMOS gate implementation allows Philips to offer CPLDs which are both high performance and low power, breaking the paradigm that to have low power, you must have low performance. Refer to Figure 5 and Table 2 showing the I_{DD} vs. Frequency of our PZ5032 TotalCMOS™ CPLD.

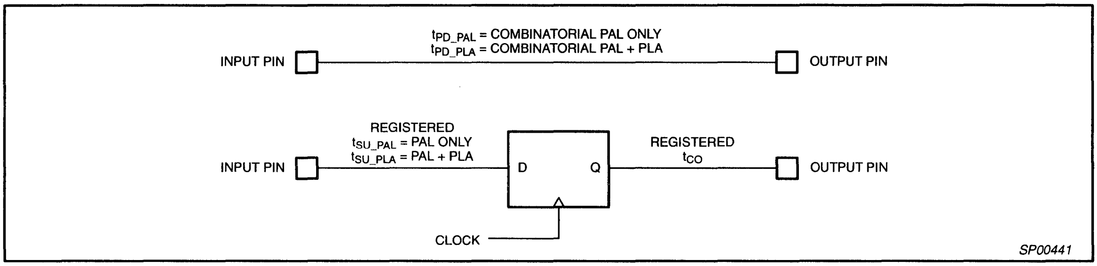


Figure 4. CoolRunner™ Timing Model

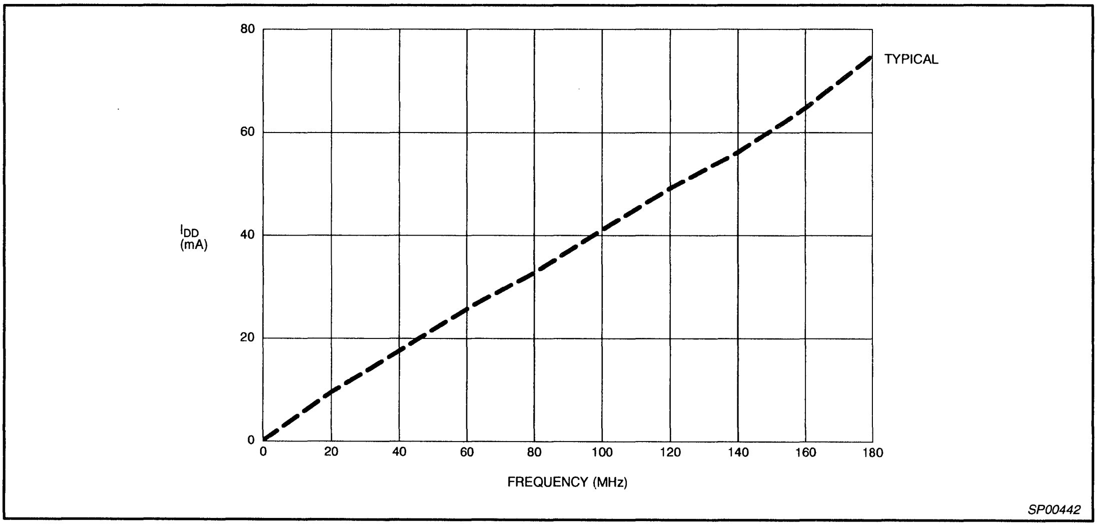


Figure 5. I_{DD} vs. Frequency @ V_{DD} = 5.0V, 25°C

Table 2. I_{DD} vs Frequency

V_{DD} = 5.00V

FREQ (MHz)	0	20	40	60	80	100	120	140	160	180
Typical I_{DD}(mA)	0.05	9.62	17.5	25.6	32.5	40.8	49.0	55.9	64.2	75.2

ABSOLUTE MAXIMUM RATINGS[1]

SYMBOL	PARAMETER	MIN.	MAX.	UNIT
V_{DD}	Supply voltage	–0.5	7.0	V
V_I	Input voltage	–1.2	V_{DD}+0.5	V
V_{OUT}	Output voltage	–0.5	V_{DD}+0.5	V
I_{IN}	Input current	–30	30	mA
I_{OUT}	Output current	–100	100	mA
T_J	Maximum junction temperature	–40	150	°C
T_{str}	Storage temperature	–65	150	°C

NOTE:

1. Stresses above those listed may cause malfunction or permanent damage to the device. This is a stress rating only. Functional operation at these or any other condition above those indicated in the operational and programming specification is not implied.

OPERATING RANGE

PRODUCT GRADE	TEMPERATURE	VOLTAGE
Commercial	0 to +70°C	5.0 ±5% V
Industrial	–40 to +85°C	5.0 ±10% V

DC ELECTRICAL CHARACTERISTICS FOR COMMERCIAL GRADE DEVICES

Commercial: 0°C ≤ T_{amb} ≤ +70°C; 4.75V ≤ V_{DD} ≤ 5.25V

SYMBOL	PARAMETER	TEST CONDITIONS	MIN.	MAX.	UNIT
V_{IL}	Input voltage low	V_{DD} = 4.75V		0.8	V
V_{IH}	Input voltage high	V_{DD} = 5.25V	2.0		V
V_I	Input clamp voltage	V_{DD} = 4.75V, I_{IN} = –18mA		–1.2	V
V_{OL}	Output voltage low	V_{DD} = 4.75V, I_{OL} = 12mA		0.5	V
V_{OH}	Output voltage high	V_{DD} = 4.75V, I_{OH} = –12mA	2.4		V
I_{IL}	Input leakage current low	V_{DD} = 5.25V (except CKO), V_{IN} = 0.4V	–10	10	µA
I_{IH}	Input leakage current high	V_{DD} = 5.25V, V_{IN} = 3.0V	–10	10	µA
I_{IL}	Clock input leakage current	V_{DD} = 5.25V, V_{IN} = 0.4V	–10	10	µA
I_{OZL}	3-Stated output leakage current low	V_{DD} = 5.25V, V_{IN} = 0.4V	–10	10	µA
I_{OZH}	3-Stated output leakage current high	V_{DD} = 5.25V, V_{IN} = 3.0V	–10	10	µA
I_{DDQ}	Standby current	V_{DD} = 5.25V, T_{amb} = 0°C		75	µA
I_{DDD}[1]	Dynamic current	V_{DD} = 5.25V, T_{amb} = 0°C @ 1MHz		3	mA
		V_{DD} = 5.25V, T_{amb} = 0°C @ 50MHz		30	mA
I_{OS}	Short circuit output current	1 pin at a time for no longer than 1 second	–50	–200	mA
C_{IN}	Input pin capacitance	T_{amb} = 25°C, f = 1MHz		8	pF
C_{CLK}	Clock input capacitance	T_{amb} = 25°C, f = 1MHz	5	12	pF
$C_{I/O}$	I/O pin capacitance	T_{amb} = 25°C, f = 1MHz		10	pF

NOTE:

1. This parameter measured with a 16–bit, loadable up/down counter loaded into every logic block, with all outputs enabled and unloaded. Inputs are tied to V_{DD} or ground. This parameter guaranteed by design and characterization, not testing.

AC ELECTRICAL CHARACTERISTICS[1] FOR COMMERCIAL GRADE DEVICES

Commercial: 0°C ≤ T_{amb} ≤ +70°C; 4.75V ≤ V_{DD} ≤ 5.25V

SYMBOL	PARAMETER	–6		–7		–10		UNIT
		MIN.	MAX.	MIN.	MAX.	MIN.	MAX.	
t_{PD_PAL}	Propagation delay time, input (or feedback node) to output through PAL	2	6	2	7.5	2	10	ns
t_{PD_PLA}	Propagation delay time, input (or feedback node) to output through PAL & PLA	3	8	3	10	3	12.5	ns
t_{CO}	Clock to out delay time	2	5.5	2	7	2	9	ns
t_{SU_PAL}	Setup time (from input or feedback node) through PAL	4		5.5		8		ns
t_{SU_PLA}	Setup time (from input or feedback node) through PAL + PLA	6		8		10.5		ns
t_H	Hold time		0		0		0	ns
t_{CH}	Clock High time	3		4		5		ns
t_{CL}	Clock Low time	3		4		5		ns
t_R	Input rise time		20		20		20	ns
t_F	Input fall time		20		20		20	ns
f_{MAX1}	Maximum FF toggle rate[2] (1/t_{CH} + t_{CL})	167		125		100		MHz
f_{MAX2}	Maximum internal frequency[2] (1/t_{SUPAL} + t_{CF})	125		91		64		MHz
f_{MAX3}	Maximum external frequency[2] (1/t_{SUPAL} + t_{CO})	105		80		59		MHz
t_{BUF}	Output buffer delay time		1.5		1.5		1.5	ns
t_{PDF_PAL}	Input (or feedback node) to internal feedback node delay time through PAL		4.5		6		8.5	ns
t_{PDF_PLA}	Input (or feedback node) to internal feedback node delay time through PAL + PLA		6.5		8.5		11	ns
t_{CF}	Clock to internal feedback node delay time		4		5.5		7.5	ns
t_{INIT}	Delay from valid V_{DD} to valid reset		50		50		50	µs
t_{ER}	Input to output disable[3]		11		12.5		15	ns
t_{EA}	Input to output valid		11		12.5		15	ns
t_{RP}	Input to register preset		11		12.5		15	ns
t_{RR}	Input to register reset		14		15.5		18	ns

NOTES:

1. Specifications measured with one output switching. See Figure 6 and Table 3 for derating.
2. This parameter guaranteed by design and characterization, not by test.
3. Output C_L = 5pF.

DC ELECTRICAL CHARACTERISTICS FOR INDUSTRIAL GRADE DEVICES

Industrial: $-40°C \le T_{amb} \le +85°C$; $4.5V \le V_{DD} \le 5.5V$

SYMBOL	PARAMETER	TEST CONDITIONS	MIN.	MAX.	UNIT
V_{IL}	Input voltage low	$V_{DD} = 4.5V$		0.8	V
V_{IH}	Input voltage high	$V_{DD} = 5.5V$	2.0		V
V_I	Input clamp voltage	$V_{DD} = 4.5V$, $I_{IN} = -18mA$		–1.2	V
V_{OL}	Output voltage low	$V_{DD} = 4.5V$, $I_{OL} = 12mA$		0.5	V
V_{OH}	Output voltage high	$V_{DD} = 4.5V$, $I_{OH} = -12mA$	2.4		V
I_{IL}	Input leakage current low	$V_{DD} = 5.5V$ (except CKO), $V_{IN} = 0.4V$	–10	10	µA
I_{IH}	Input leakage current high	$V_{DD} = 5.5V$, $V_{IN} = 3.0V$	–10	10	µA
I_{IL}	Clock input leakage current	$V_{DD} = 5.5V$, $V_{IN} = 0.4V$	–10	10	µA
I_{OZL}	3-Stated output leakage current low	$V_{DD} = 5.5V$, $V_{IN} = 0.4V$	–10	10	µA
I_{OZH}	3-Stated output leakage current high	$V_{DD} = 5.5V$, $V_{IN} = 3.0V$	–10	10	µA
I_{DDQ}	Standby current	$V_{DD} = 5.5V$, $T_{amb} = -40°C$		95	µA
I_{DDD}[1]	Dynamic current	$V_{DD} = 5.5V$, $T_{amb} = -40°C$ @ 1MHz		4	mA
		$V_{DD} = 5.5V$, $T_{amb} = -40°C$ @ 50MHz		35	mA
I_{OS}	Short circuit output current	1 pin at a time for no longer than 1 second	–50	–230	mA
C_{IN}	Input pin capacitance	$T_{amb} = 25°C$, f = 1MHz		8	pF
C_{CLK}	Clock input capacitance	$T_{amb} = 25°C$, f = 1MHz	5	12	pF
$C_{I/O}$	I/O pin capacitance	$T_{amb} = 25°C$, f = 1MHz		10	pF

NOTE:

1. This parameter measured with a 16–bit, loadable up/down counter loaded into every logic block, with all outputs enabled and unloaded. Inputs are tied to V_{DD} or ground. This parameter guaranteed by design and characterization, not testing.

AC ELECTRICAL CHARACTERISTICS[1] FOR INDUSTRIAL GRADE DEVICES

Industrial: $-40°C \le T_{amb} \le +85°C$; $4.5V \le V_{DD} \le 5.5V$

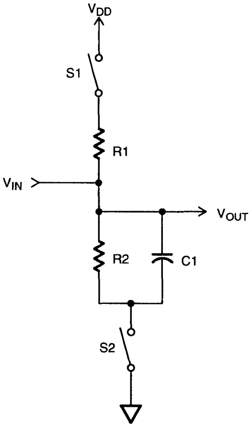
SYMBOL	PARAMETER	I7		I10		UNIT
		MIN.	MAX.	MIN.	MAX.	
t_{PD_PAL}	Propagation delay time, input (or feedback node) to output through PAL	2	7.5	2	10	ns
t_{PD_PLA}	Propagation delay time, input (or feedback node) to output through PAL & PLA	3	9.5	3	12.5	ns
t_{CO}	Clock to out delay time	2	6	2	9	ns
t_{SU_PAL}	Setup time (from input or feedback node) through PAL	5		8		ns
t_{SU_PLA}	Setup time (from input or feedback node) through PAL + PLA	7		10.5		ns
t_H	Hold time		0		0	ns
t_{CH}	Clock High time	4		5		ns
t_{CL}	Clock Low time	4		5		ns
t_R	Input rise time		20		20	ns
t_F	Input fall time		20		20	ns
f_{MAX1}	Maximum FF toggle rate[2] $(1/t_{CH} + t_{CL})$	125		100		MHz
f_{MAX2}	Maximum internal frequency[2] $(1/t_{SUPAL} + t_{CF})$	105		64		MHz
f_{MAX3}	Maximum external frequency[2] $(1/t_{SUPAL} + t_{CO})$	91		59		MHz
t_{BUF}	Output buffer delay time		1.5		1.5	ns
t_{PDF_PAL}	Input (or feedback node) to internal feedback node delay time through PAL		6		8.5	ns
t_{PDF_PLA}	Input (or feedback node) to internal feedback node delay time through PAL + PLA		8		11	ns
t_{CF}	Clock to internal feedback node delay time		4.5		7.5	ns
t_{INIT}	Delay from valid V_{DD} to valid reset		50		50	µs
t_{ER}	Input to output disable[3]		12		15	ns
t_{EA}	Input to output valid		12		15	ns
t_{RP}	Input to register preset		12		15	ns
t_{RR}	Input to register reset		14		18	ns

NOTES:

1. Specifications measured with one output switching. See Figure 6 and Table 3 for derating.
2. This parameter guaranteed by design and characterization, not by test.
3. Output C_L = 5pF.

SWITCHING CHARACTERISTICS

The test load circuit and load values for the AC Electrical Characteristics are illustrated below.



COMPONENT	VALUES
R1	470Ω
R2	250Ω
C1	35pF

MEASUREMENT	S1	S2
t_{PZH}	Open	Closed
t_{PZL}	Closed	Closed
t_P	Closed	Closed

NOTE: For t_{PHZ} and t_{PLZ} C = 5pF, and 3-State levels are measured 0.5V from steady state active level.

SP00476

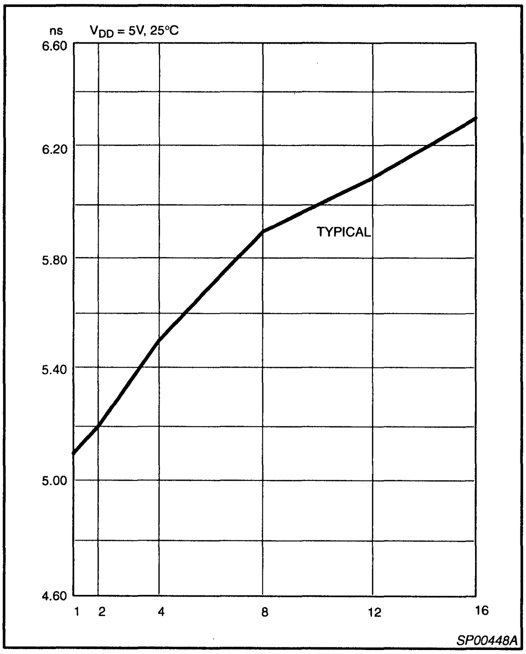
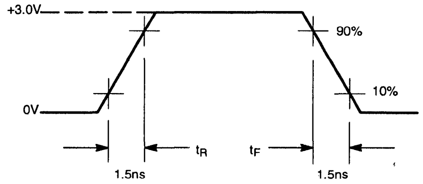


Figure 6. t_{PD_PAL} **vs Outputs switching**

VOLTAGE WAVEFORM



MEASUREMENTS:
All circuit delays are measured at the +1.5V level of inputs and outputs, unless otherwise specified.

Input Pulses

SP00368

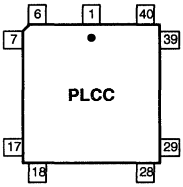
Table 3. t_{PD_PAL} vs # of Outputs switching

V$_{DD}$ = 5.00V

# of Outputs	1	2	4	8	12	16
Typical (ns)	5.1	5.2	5.5	5.9	6.1	6.3

PIN DESCRIPTIONS

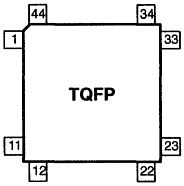
PZ5032 – 44-Pin Plastic Leaded Chip Carrier



Pin	Function	Pin	Function	Pin	Function
1	IN1	16	I/O–A10	31	I/O–B9
2	IN3	17	I/O–A11	32	I/O–B8
3	V_{DD}	18	I/O–A12	33	I/O–B7
4	I/O–A0–CK1	19	I/O–A13	34	I/O–B6
5	I/O–A1	20	I/O–A14	35	V_{DD}
6	I/O–A2	21	I/O–A15	36	I/O–B5
7	I/O–A3	22	GND	37	I/O–B4
8	I/O–A4	23	V_{DD}	38	I/O–B3
9	I/O–A5	24	I/O–B15	39	I/O–B2
10	GND	25	I/O–B14	40	I/O–B1
11	I/O–A6	26	I/O–B13	41	I/O–B0
12	I/O–A7	27	I/O–B12	42	GND
13	I/O–A8	28	I/O–B11	43	IN0–CK0
14	I/O–A9	29	I/O–B10	44	IN2–gtsn
15	V_{DD}	30	GND		

SP00420

PZ5032 – 44-Pin Thin Quad Flat Package



Pin	Function	Pin	Function	Pin	Function
1	I/O–A3	16	GND	31	I/O–B4
2	I/O–A4	17	V_{DD}	32	I/O–B3
3	I/O–A5	18	I/O–B15	33	I/O–B2
4	GND	19	I/O–B14	34	I/O–B1
5	I/O–A6	20	I/O–B13	35	I/O–B0
6	I/O–A7	21	I/O–B12	36	GND
7	I/O–A8	22	I/O–B11	37	IN0/CK0
8	I/O–A9	23	I/O–B10	38	IN2–gtsn
9	V_{DD}	24	GND	39	IN1
10	I/O–A10	25	I/O–B9	40	IN3
11	I/O–A11	26	I/O–B8	41	V_{DD}
12	I/O–A12	27	I/O–B7	42	I/O–A0–CK1
13	I/O–A13	28	I/O–B6	43	I/O–A1
14	I/O–A14	29	V_{DD}	44	I/O–A2
15	I/O–A15	30	I/O–B5		

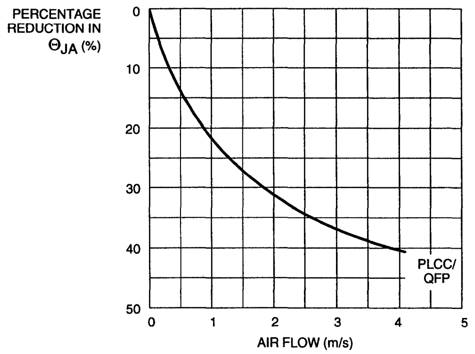
SP00433

Package Thermal Characteristics

Philips Semiconductors uses the Temperature Sensitive Parameter (TSP) method to test thermal resistance. This method meets Mil-Std-883C Method 1012.1 and is described in Philips *1995 IC Package Databook*. Thermal resistance varies slightly as a function of input power. As input power increases, thermal resistance changes approximately 5% for a 100% change in power.

Figure 7 is a derating curve for the change in Θ_{JA} with airflow based on wind tunnel measurements. It should be noted that the wind flow dynamics are more complex and turbulent in actual applications than in a wind tunnel. Also, the test boards used in the wind tunnel contribute significantly to forced convection heat transfer, and may not be similar to the actual circuit board, especially in size.

Package	Θ_{JA}
44-pin PLCC	49.8°C/W
44-pin TQFP	66.3°C/W



SP00419A

Figure 7. Average Effect of Airflow on Θ_{JA}

64 macrocell CPLD PZ3064

FEATURES

- Industry's first TotalCMOS™ PLD – both CMOS design and process technologies
- Fast Zero Power (FZP™) design technique provides ultra-low power and very high speed
- High speed pin-to-pin delays of 10ns
- Ultra-low static power of less than 50μA
- Dynamic power that is 70% lower at 50MHz than competing devices
- 100% routable with 100% utilization while all pins and all macrocells are fixed
- Deterministic timing model that is extremely simple to use
- 4 clocks with programmable polarity at every macrocell
- Support for complex asynchronous clocking
- Innovative XPLA™ architecture combines high speed with extreme flexibility
- 1000 erase/program cycles guaranteed
- 20 years data retention guaranteed
- Logic expandable to 37 product terms
- PCI compliant
- Advanced 0.5μ E^2CMOS process
- Security bit prevents unauthorized access
- Design entry and verification using industry standard and Philips CAE tools
- Reprogrammable using industry standard device programmers
- Innovative Control Term structure provides either sum terms or product terms in each logic block for:
  - Programmable 3-State buffer
  - Asynchronous macrocell register preset/reset
- Programmable global 3-State pin facilitates 'bed of nails' testing without using logic resources
- Available in PLCC, TQFP, and PQFP packages
- Available in both Commercial and Industrial grades

Table 1. PZ3064 Features

	PZ3064
Usable gates	2000
Maximum inputs	68
Maximum I/Os	64
Number of macrocells	64
Propagation delay (ns)	10
Packages	44-pin PLCC, 44-pin TQFP, 68-pin PLCC, 84-pin PLCC, 100-pin PQFP

DESCRIPTION

The PZ3064 CPLD (Complex Programmable Logic Device) is the second in a family of Fast Zero Power (FZP™) CPLDs from Philips Semiconductors. These devices combine high speed and zero power in a 64 macrocell CPLD. With the FZP™ design technique, the PZ3064 offers true pin-to-pin speeds of 10ns, while simultaneously delivering power that is less than 50μA at standby without the need for 'turbo bits' or other power down schemes. By replacing conventional sense amplifier methods for implementing product terms (a technique that has been used in PLDs since the bipolar era) with a cascaded chain of pure CMOS gates, the dynamic power is also substantially lower than any competing CPLD – 70% lower at 50MHz. These devices are the first TotalCMOS™ PLDs, as they use both a CMOS process technology **and** the patented full CMOS FZP™ design technique. For 5V applications, Philips also offers the high speed PZ5064 CPLD that offers these features in a full 5V implementation.

The Philips FZP™ CPLDs introduce the new patent-pending XPLA™ (eXtended Programmable Logic Array) architecture. The XPLA™ architecture combines the best features of both PLA and PAL™ type structures to deliver high speed and flexible logic allocation that results in superior ability to make design changes with fixed pinouts. The XPLA™ structure in each logic block provides a fast 10ns PAL™ path with 5 dedicated product terms per output. This PAL™ path is joined by an additional PLA structure that deploys a pool of 32 product terms to a fully programmable OR array that can allocate the PLA product terms to any output in the logic block. This combination allows logic to be allocated efficiently throughout the logic block and supports as many as 37 product terms on an output. The speed with which logic is allocated from the PLA array to an output is only 2.5ns, regardless of the number of PLA product terms used, which results in worst case t_{PD}'s of only 12.5ns from any pin to any other pin. In addition, logic that is common to multiple outputs can be placed on a single PLA product term and shared across multiple outputs via the OR array, effectively increasing design density.

The PZ3064 CPLDs are supported by industry standard CAE tools (Cadence, Mentor, Synopsys, Synario, Viewlogic, MINC), using text (Abel, VHDL, Verilog) and/or schematic entry. Design verification uses industry standard simulators for functional and timing simulation. Development is supported on personal computer, Sparc, and HP platforms. Device fitting uses either Minc or Philips Semiconductors-developed tools.

The PZ3064 CPLD is reprogrammable using industry standard device programmers from vendors such as Data I/O, BP Microsystems, SMS, and others.

PAL is a registered trademark of Advanced Micro Devices, Inc.

ORDERING INFORMATION

ORDER CODE	DESCRIPTION	DESCRIPTION	DRAWING NUMBER
PZ3064-10A44	44-pin PLCC, 10ns t_{PD}	Commercial temp range, 3.3 volt power supply, ± 10%	SOT187-2
PZ3064-12A44	44-pin PLCC, 12ns t_{PD}	Commercial temp range, 3.3 volt power supply, ± 10%	SOT187-2
PZ3064I12A44	44-pin PLCC, 12ns t_{PD}	Industrial temp range, 3.3 volt power supply, ± 10%	SOT187-2
PZ3064I15A44	44-pin PLCC, 15ns t_{PD}	Industrial temp range, 3.3 volt power supply, ± 10%	SOT187-2
PZ3064-10BC	44-pin TQFP, 10ns t_{PD}	Commercial temp range, 3.3 volt power supply, ± 10%	SOT376-1
PZ3064-12BC	44-pin TQFP, 12ns t_{PD}	Commercial temp range, 3.3 volt power supply, ± 10%	SOT376-1
PZ3064I12BC	44-pin TQFP, 12ns t_{PD}	Industrial temp range, 3.3 volt power supply, ± 10%	SOT376-1
PZ3064I15BC	44-pin TQFP, 15ns t_{PD}	Industrial temp range, 3.3 volt power supply, ± 10%	SOT376-1
PZ3064-10A68	68-pin PLCC, 10ns t_{PD}	Commercial temp range, 3.3 volt power supply, ± 10%	SOT188-3
PZ3064-12A68	68-pin PLCC, 12ns t_{PD}	Commercial temp range, 3.3 volt power supply, ± 10%	SOT188-3
PZ3064I12A68	68-pin PLCC, 12ns t_{PD}	Industrial temp range, 3.3 volt power supply, ± 10%	SOT188-3
PZ3064I15A68	68-pin PLCC, 15ns t_{PD}	Industrial temp range, 3.3 volt power supply, ± 10%	SOT188-3
PZ3064-10A84	84-pin PLCC, 10ns t_{PD}	Commercial temp range, 3.3 volt power supply, ± 10%	SOT189-3
PZ3064-12A84	84-pin PLCC, 12ns t_{PD}	Commercial temp range, 3.3 volt power supply, ± 10%	SOT189-3
PZ3064I12A84	84-pin PLCC, 12ns t_{PD}	Industrial temp range, 3.3 volt power supply, ± 10%	SOT189-3
PZ3064I15A84	84-pin PLCC, 15ns t_{PD}	Industrial temp range, 3.3 volt power supply, ± 10%	SOT189-3
PZ3064-10BB1	100-pin PQFP, 10ns t_{PD}	Commercial temp range, 3.3 volt power supply, ± 10%	SOT382-1
PZ3064-12BB1	100-pin PQFP, 12ns t_{PD}	Commercial temp range, 3.3 volt power supply, ± 10%	SOT382-1
PZ3064I12BB1	100-pin PQFP, 12ns t_{PD}	Industrial temp range, 3.3 volt power supply, ± 10%	SOT382-1
PZ3064I15BB1	100-pin PQFP, 15ns t_{PD}	Industrial temp range, 3.3 volt power supply, ± 10%	SOT382-1

XPLA™ ARCHITECTURE

Figure 1 shows a high level block diagram of a 64 macrocell device implementing the XPLA™ architecture. The XPLA™ architecture consists of logic blocks that are interconnected by a Zero-power Interconnect Array (ZIA). The ZIA is a virtual crosspoint switch. Each logic block is essentially a 36V16 device with 36 inputs from the ZIA and 16 macrocells. Each logic block also provides 32 ZIA feedback paths from the macrocells and I/O pins.

From this point of view, this architecture looks like many other CPLD architectures. What makes the CoolRunner™ family unique is what is inside each logic block and the design technique used to implement these logic blocks. The contents of the logic block will be described next.

Logic Block Architecture

Figure 2 illustrates the logic block architecture. Each logic block contains control terms, a PAL array, a PLA array, and 16 macrocells. the 6 control terms can individually be configured as either SUM or PRODUCT terms, and are used to control the preset/reset and output enables of the 16 macrocells' flip-flops. The PAL array consists of a programmable AND array with a fixed OR array, while the PLA array consists of a programmable AND array with a programmable OR array. The PAL array provides a high speed path through the array, while the PLA array provides increased product term density.

Each macrocell has 5 dedicated product terms from the PAL array. The pin-to-pin t_{PD} of the PZ3064 device through the PAL array is 10ns. If a macrocell needs more than 5 product terms, it simply gets the additional product terms from the PLA array. The PLA array consists of 32 product terms, which are available for use by all 16 macrocells. The additional propagation delay incurred by a macrocell using 1 or all 32 PLA product terms is just 2.5ns. So the total pin-to-pin t_{PD} for the PZ3064 using 6 to 37 product terms is 12.5ns (10ns for the PAL + 2.5ns for the PLA).

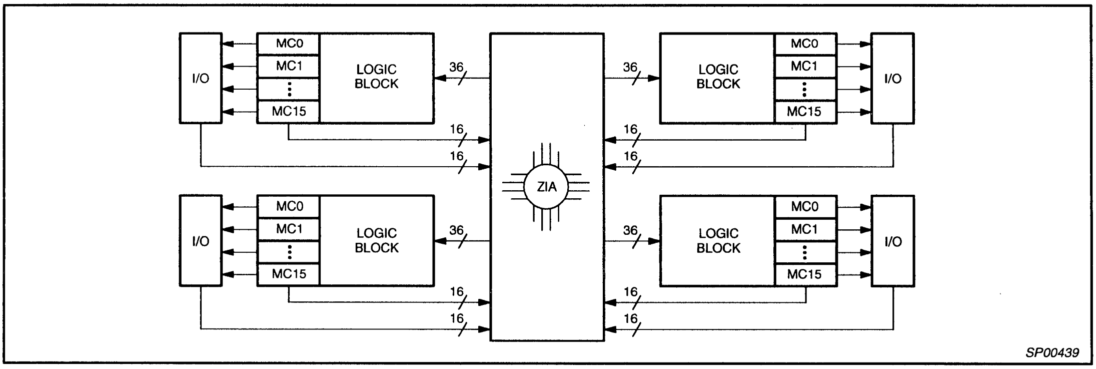


Figure 1. Philips XPLA CPLD Architecture

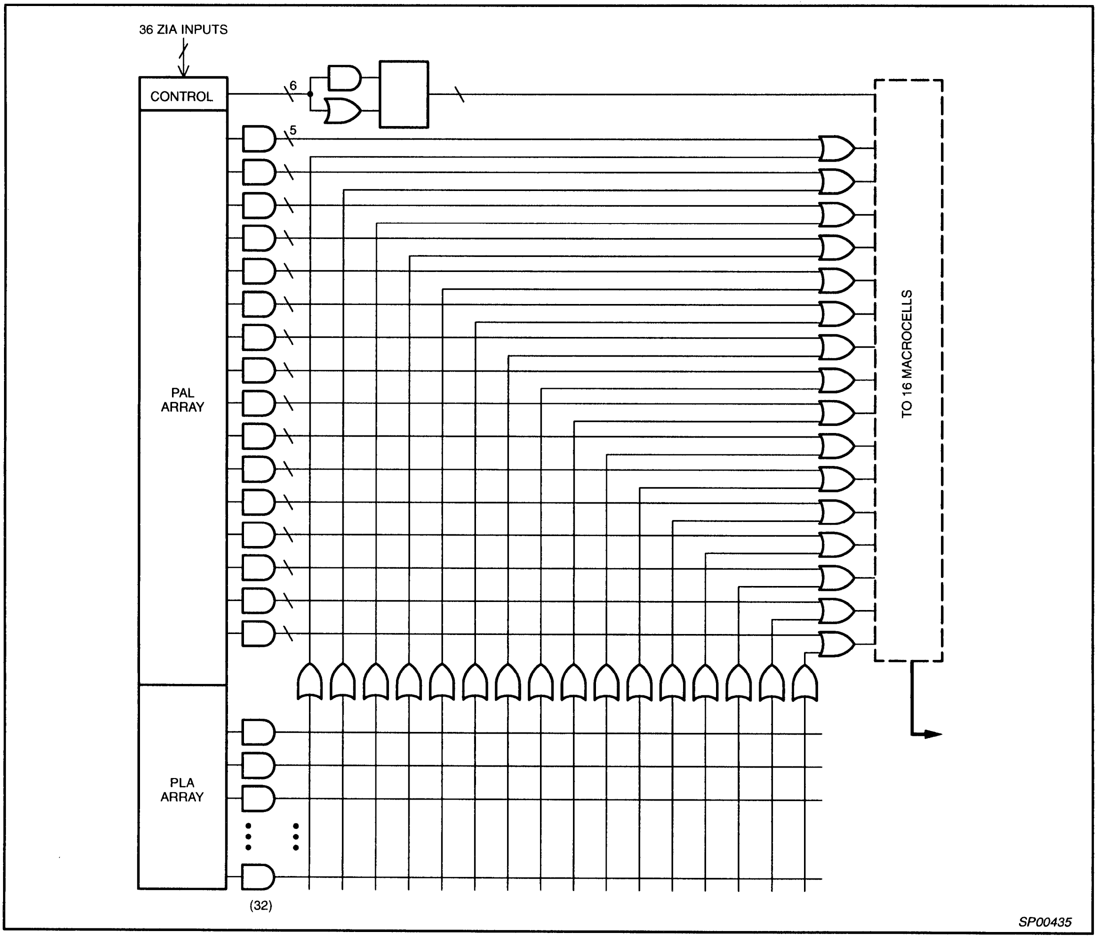


Figure 2. Philips Logic Block Architecture

Macrocell Architecture

Figure 3 shows the architecture of the macrocell used in the CoolRunner™ family. The macrocell consists of a flip-flop that can be configured as either a D or T type. A D-type flip-flop is generally more useful for implementing state machines and data buffering. A T-type flip-flop is generally more useful in implementing counters. All CoolRunner™ family members provide both synchronous and asynchronous clocking and provide the ability to clock off either the falling or rising edges of these clocks. These devices are designed such that the skew between the rising and falling edges of a clock are minimized for clocking integrity. There are 4 clocks available on the PZ3064 device. Clock 0 (CLK0) is designated as the "synchronous" clock and must be driven by an external source. Clock 1 (CLK1), Clock 2 (CLK2), and Clock 3 (CLK3) can either be used as a synchronous clock (driven by an external source) or as an asynchronous clock (driven by a macrocell equation).

Two of the control terms (CT0 and CT1) are used to control the Preset/Reset of the macrocell's flip-flop. The Preset/Reset feature for each macrocell can also be disabled. Note that the Power-on Reset leaves all macrocells in the "zero" state when power is properly applied. The other 4 control terms (CT2–CT5) can be used to control the Output Enable of the macrocell's output buffers. The reason there are as many control terms dedicated for the Output Enable of the macrocell is to insure that all CoolRunner™ devices are PCI compliant. The macrocell's output buffers can also be always enabled or disabled. All CoolRunner™ devices also provide a Global Tri-State ($\overline{\text{GTS}}$) pin, which, when pulled Low, will 3-State all the outputs of the device. This pin is provided to support "In-Circuit Testing" or "Bed-of-Nails Testing".

There are two feedback paths to the ZIA: one from the macrocell, and one from the I/O pin. The ZIA feedback path before the output buffer is the macrocell feedback path, while the ZIA feedback path after the output buffer is the I/O pin ZIA path. When the macrocell is used as an output, the output buffer is enabled, and the macrocell feedback path can be used to feedback the logic implemented in the macrocell. When the I/O pin is used as an input, the output buffer will be 3-Stated and the input signal will be fed into the ZIA via the I/O feedback path, and the logic implemented in the buried macrocell can be fed back to the ZIA via the macrocell feedback path. It should be noted that unused inputs or I/Os should be properly terminated.

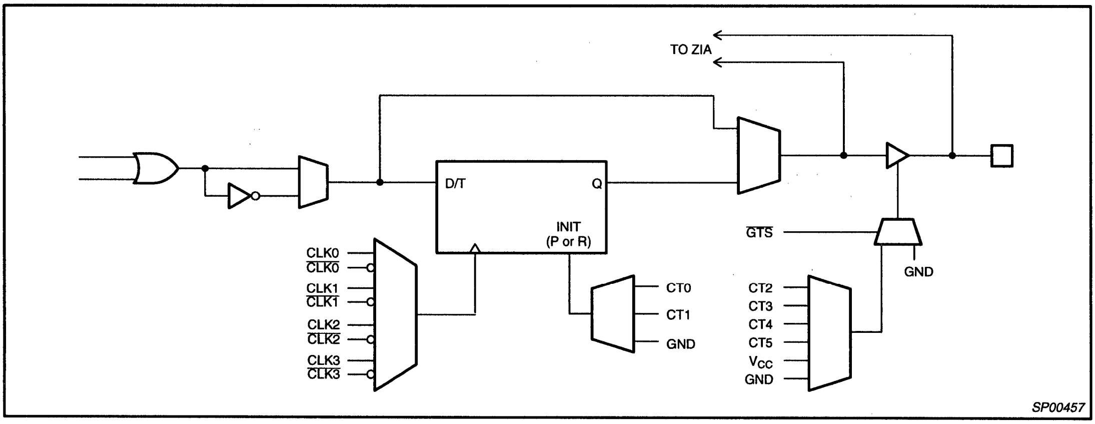


Figure 3. PZ3064 Macrocell Architecture

Simple Timing Model

Figure 4 shows the CoolRunner™ Timing Model. The CoolRunner™ timing model looks very much like a 22V10 timing model in that there are three main timing parameters, including t_{PD}, t_{SU}, and t_{CO}. In other competing architectures, the user may be able to fit the design into the CPLD, but is not sure whether system timing requirements can be met until after the design has been fit into the device. This is because the timing models of competing architectures are very complex and include such things as timing dependencies on the number of parallel expanders borrowed, sharable expanders, varying number of X and Y routing channels used, etc. In the XPLA™ architecture, the user knows up front whether the design will meet system timing requirements. This is due to the simplicity of the timing model. For example, in the PZ3064 device, the user knows up front that if a given output uses 5 product terms or less, the t_{PD} = 10ns, the t_{SU_PAL} = 6ns, and the t_{CO} = 7ns. If an output is using 6 to 37 product terms, an additional 2ns must be added to the t_{PD} and t_{SU} timing parameters to account for the time to propagate through the PLA array.

TotalCMOS™ Design Technique for Fast Zero Power

Philips is the first to offer a TotalCMOS™ CPLD, both in process technology and design technique. Philips employs a cascade of CMOS gates to implement its Sum of Products instead of the traditional sense amp approach. This CMOS gate implementation allows Philips to offer CPLDs which are both high performance and low power, breaking the paradigm that to have low power, you must have low performance. Refer to Figure 5 and Table 2 showing the I_{DD} vs. Frequency of our PZ3064 TotalCMOS™ CPLD.

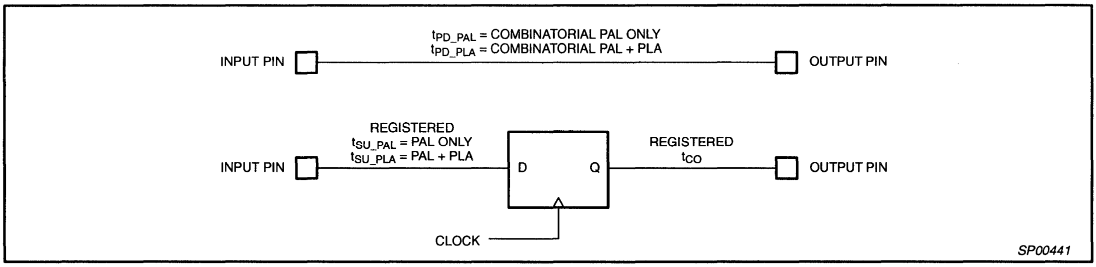


Figure 4. CoolRunner™ Timing Model

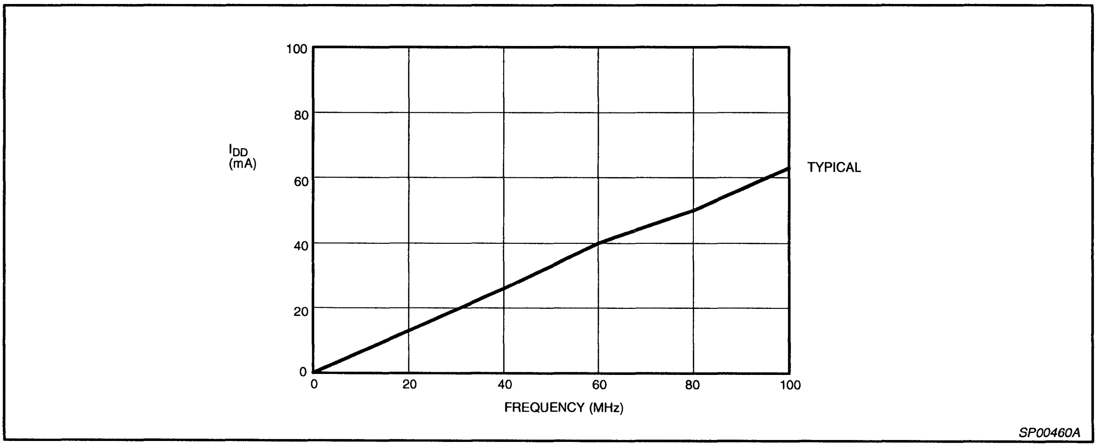


Figure 5. I_{DD} vs. Frequency @ V_{DD} = 3.3V, 25°C

Table 2. I_{DD} vs. Frequency

V_{DD} = 3.3V

FREQUENCY (MHz)	0	20	40	60	80	100
Typical I_{DD} (mA)	0.04	13	26	40	50	63

ABSOLUTE MAXIMUM RATINGS[1]

SYMBOL	PARAMETER	MIN.	MAX.	UNIT
V_{DD}	Supply voltage	–0.5	7.0	V
V_I	Input voltage	–1.2	V_{DD}+0.5	V
V_{OUT}	Output voltage	–0.5	V_{DD}+0.5	V
I_{IN}	Input current	–30	30	mA
I_{OUT}	Output current	–100	100	mA
T_J	Maximum junction temperature	–40	150	°C
T_{str}	Storage temperature	–65	150	°C

NOTES:

1. Stresses above those listed may cause malfunction or permanent damage to the device. This is a stress rating only. Functional operation at these or any other condition above those indicated in the operational and programming specification is not implied.

OPERATING RANGE

PRODUCT GRADE	TEMPERATURE	VOLTAGE
Commercial	0 to +70°C	3.3 ±10% V
Industrial	–40 to +85°C	3.3 ±10% V

DC ELECTRICAL CHARACTERISTICS FOR COMMERCIAL GRADE DEVICES

Commercial: 0°C ≤ T_{amb} ≤ +70°C; 3.0V ≤ V_{DD} ≤ 3.6V

SYMBOL	PARAMETER	TEST CONDITIONS	MIN.	MAX.	UNIT
V_{IL}	Input voltage low	V_{DD} = 3.0V		0.8	V
V_{IH}	Input voltage high	V_{DD} = 3.6V	2.0		V
V_I	Input clamp voltage	V_{DD} = 3.0V, I_{IN} = –18mA		–1.2	V
V_{OL}	Output voltage low	V_{DD} = 3.0V, I_{OL} = 8mA		0.5	V
V_{OH}	Output voltage high	V_{DD} = 3.0V, I_{OH} = –8mA	2.4		V
I_I	Input leakage current	V_{IN} = 0 to V_{DD}	–10	10	μA
I_{OZ}	3-Stated output leakage current	V_{IN} = 0 to V_{DD}	–10	10	μA
I_{DDQ}	Standby current	V_{DD} = 3.6V, T_{amb} = 0°C		50	μA
I_{DDD}[1]	Dynamic current	V_{DD} = 3.6V, T_{amb} = 0°C @ 1MHz		1	mA
		V_{DD} = 3.6V, T_{amb} = 0°C @ 50MHz		40	mA
I_{OS}	Short circuit output current	1 pin at a time for no longer than 1 second	–5	–100	mA
C_{IN}	Input pin capacitance	T_{amb} = 25°C, f = 1MHz		8	pF
C_{CLK}	Clock input capacitance	T_{amb} = 25°C, f = 1MHz	5	12	pF
$C_{I/O}$	I/O pin capacitance	T_{amb} = 25°C, f = 1MHz		10	pF

NOTE:

1. This parameter measured with a 16-bit, loadable up/down counter loaded into every logic block, with all outputs enabled and unloaded. Inputs are tied to V_{DD} or ground. This parameter guaranteed by design and characterization, not testing.

AC ELECTRICAL CHARACTERISTICS[1] FOR COMMERCIAL GRADE DEVICES

Commercial: 0°C ≤ T_{amb} ≤ +70°C; 3.0V ≤ V_{DD} ≤ 3.6V

SYMBOL	PARAMETER	–10		–12		UNIT
		MIN.	MAX.	MIN.	MAX.	
t_{PD_PAL}	Propagation delay time, input (or feedback node) to output through PAL	2	10	2	12	ns
t_{PD_PLA}	Propagation delay time, input (or feedback node) to output through PAL & PLA	3	12.5	3	14.5	ns
t_{CO}	Clock to out delay time	2	7	2	8	ns
t_{SU_PAL}	Setup time (from input or feedback node) through PAL	5.5		7		ns
t_{SU_PLA}	Setup time (from input or feedback node) through PAL + PLA	8		9.5		ns
t_H	Hold time		0		0	ns
t_{CH}	Clock High time	4		5		ns
t_{CL}	Clock Low time	4		5		ns
t_R	Input Rise time		20		20	ns
t_F	Input Fall time		20		20	ns
f_{MAX1}	Maximum FF toggle rate[2] (1/t_{CH} + t_{CL})	125		100		MHz
f_{MAX2}	Maximum internal frequency[2] (1/t_{SUPAL} + t_{CF})	91		74		MHz
f_{MAX3}	Maximum external frequency[2] (1/t_{SUPAL} + t_{CO})	80		67		MHz
t_{BUF}	Output buffer delay time		1.5		1.5	ns
t_{PDF_PAL}	Input (or feedback node) to internal feedback node delay time through PAL		8.5		10.5	ns
t_{PDF_PLA}	Input (or feedback node) to internal feedback node delay time through PAL+PLA		11		13	ns
t_{CF}	Clock to internal feedback node delay time		5.5		6.5	ns
t_{INIT}	Delay from valid V_{DD} to valid reset		50		50	μs
t_{ER}	Input to output disable[3]		12.5		14	ns
t_{EA}	Input to output valid		12.5		14	ns
t_{RP}	Input to register preset		15		16	ns
t_{RR}	Input to register reset		15		16	ns

NOTES:

1. Specifications measured with one output switching. See Figure 6 and Table 3 for derating.
2. This parameter guaranteed by design and characterization, not by test.
3. Output C_L = 5pF.

DC ELECTRICAL CHARACTERISTICS FOR INDUSTRIAL GRADE DEVICES

Industrial: $-40°C \leq T_{amb} \leq +85°C$; $3.0V \leq V_{DD} \leq 3.6V$

SYMBOL	PARAMETER	TEST CONDITIONS	MIN.	MAX.	UNIT
V_{IL}	Input voltage low	V_{DD} = 3.0V		0.8	V
V_{IH}	Input voltage high	V_{DD} = 3.6V	2.0		V
V_I	Input clamp voltage	V_{DD} = 3.0V, I_{IN} = –18mA		–1.2	V
V_{OL}	Output voltage low	V_{DD} = 3.0V, I_{OL} = 8mA		0.5	V
V_{OH}	Output voltage high	V_{DD} = 3.0V, I_{OH} = –8mA	2.4		V
I_I	Input leakage current	V_{IN} = 0 to V_{DD}	–10	10	µA
I_{OZ}	3-Stated output leakage current	V_{IN} = 0 to V_{DD}	–10	10	µA
I_{DDQ}	Standby current	V_{DD} = 3.6V, T_{amb} = –40°C		50	µA
I_{DDD}[1]	Dynamic current	V_{DD} = 3.6V, T_{amb} = –40°C @ 1MHz		1	mA
		V_{DD} = 3.6V, T_{amb} = –40°C @ 50MHz		40	mA
I_{OS}	Short circuit output current	1 pin at a time for no longer than 1 second	–5	–130	mA
C_{IN}	Input pin capacitance	T_{amb} = 25°C, f = 1MHz		8	pF
C_{CLK}	Clock input capacitance	T_{amb} = 25°C, f = 1MHz	5	12	pF
$C_{I/O}$	I/O pin capacitance	T_{amb} = 25°C, f = 1MHz		10	pF

NOTE:

1. This parameter measured with a 16–bit, loadable up/down counter loaded into every logic block, with all outputs enabled and unloaded. Inputs are tied to V_{DD} or ground. This parameter guaranteed by design and characterization, not testing.

AC ELECTRICAL CHARACTERISTICS[1] FOR INDUSTRIAL GRADE DEVICES

Industrial: $-40°C \leq T_{amb} \leq +85°C$; $3.0V \leq V_{DD} \leq 3.6V$

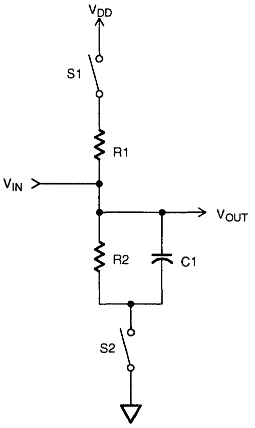
SYMBOL	PARAMETER	I12		I15		UNIT
		MIN.	MAX.	MIN.	MAX.	
t_{PD_PAL}	Propagation delay time, input (or feedback node) to output through PAL	2	12	2	15	ns
t_{PD_PLA}	Propagation delay time, input (or feedback node) to output through PAL & PLA	3	14.5	3	17.5	ns
t_{CO}	Clock to out delay time	2	8	2	9	ns
t_{SU_PAL}	Setup time (from input or feedback node) through PAL	7		8		ns
t_{SU_PLA}	Setup time (from input or feedback node) through PAL + PLA	9.5		10.5		ns
t_H	Hold time		0		0	ns
t_{CH}	Clock High time	5		5		ns
t_{CL}	Clock Low time	5		5		ns
t_R	Input Rise time		20		20	ns
t_F	Input Fall time		20		20	ns
f_{MAX1}	Maximum FF toggle rate[2] (1/t_{CH} + t_{CL})	100		100		MHz
f_{MAX2}	Maximum internal frequency[2] (1/t_{SUPAL} + t_{CF})	74		65		MHz
f_{MAX3}	Maximum external frequency[2] (1/t_{SUPAL} + t_{CO})	67		58		MHz
t_{BUF}	Output buffer delay time		1.5		1.5	ns
t_{PDF_PAL}	Input (or feedback node) to internal feedback node delay time through PAL		10.5		13.5	ns
t_{PDF_PLA}	Input (or feedback node) to internal feedback node delay time through PAL+PLA		13		16	ns
t_{CF}	Clock to internal feedback node delay time		6.5		7.5	ns
t_{INIT}	Delay from valid V_{DD} to valid reset		50		50	µs
t_{ER}	Input to output disable[3]		14		15	ns
t_{EA}	Input to output valid		14		15	ns
t_{RP}	Input to register preset		16		17	ns
t_{RR}	Input to register reset		16		17	ns

NOTES:

1. Specifications measured with one output switching. See Figure 6 and Table 3 for derating.
2. This parameter guaranteed by design and characterization, not by test.
3. Output C_L = 5pF.

SWITCHING CHARACTERISTICS

The test load circuit and load values for the AC Electrical Characteristics are illustrated below.

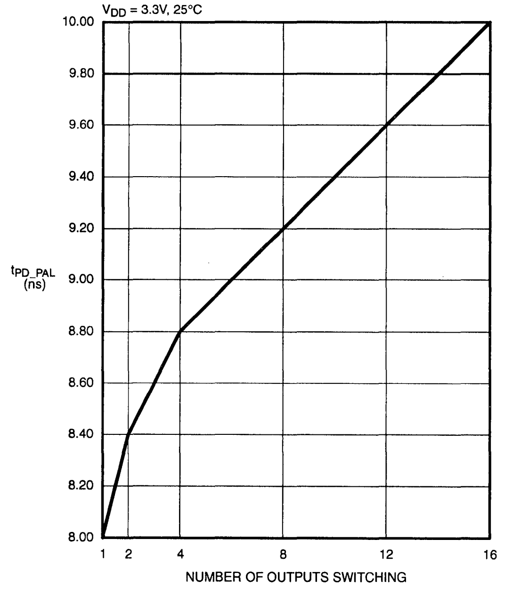


COMPONENT	VALUES
R1	390Ω
R2	390Ω
C1	35pF

MEASUREMENT	S1	S2
t_{PZH}	Open	Closed
t_{PZL}	Closed	Open
t_P	Closed	Closed

NOTE: For t_{PHZ} and t_{PLZ} C = 5pF

SP00461A



SP00462

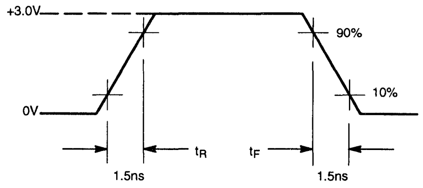
Figure 6. t_{PD_PAL} **vs. Outputs Switching**

Table 3. t_{PD_PAL} **vs. Number of Outputs Switching**

V_{DD} = 3.3V

NUMBER OF OUTPUTS	1	2	4	8	12	16
Typical (ns)	8.0	8.4	8.8	9.2	9.6	10.0

VOLTAGE WAVEFORM



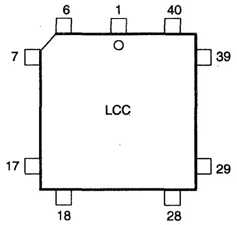
MEASUREMENTS:
All circuit delays are measured at the +1.5V level of inputs and outputs, unless otherwise specified.

Input Pulses

SP00368

PIN DESCRIPTIONS

PZ3064 – 44-Pin Plastic Leaded Chip Carrier

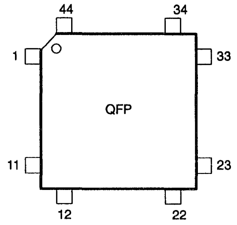


Pin	Function	Pin	Function	Pin	Function
1	IN1	16	I/O-B10	31	I/O-C13
2	IN3	17	I/O-B8	32	I/O-C15 (TCK)
3	V_{DD}	18	I/O-B4	33	I/O-D15
4	I/O-A0/CK3	19	I/O-B3	34	I/O-D13
5	I/O-A2	20	I/O-B2	35	V_{DD}
6	I/O-A5	21	I/O-B0/CK2	36	I/O-D12
7	I/O-A8 (TDI)	22	GND	37	I/O-D11
8	I/O-A11	23	V_{DD}	38	I/O-D8 (TDO)
9	I/O-A12	24	I/O-C0/CK1	39	I/O-D7
10	GND	25	I/O-C2	40	I/O-D2
11	I/O-A13	26	I/O-C3	41	I/O-D0
12	I/O-A15	27	I/O-C4	42	GND
13	I/O-B15 (TMS)*	28	I/O-C7	43	IN0-CK0
14	I/O-B13	29	I/O-C8	44	IN2-gtsn
15	V_{DD}	30	GND		

* THE TEST MODE SELECT (TMS) FUNCTION IS INACTIVE ON NON-ISR ARCHITECTURES.

SP00452A

PZ3064 – 44-Pin Thin Quad Flat Package

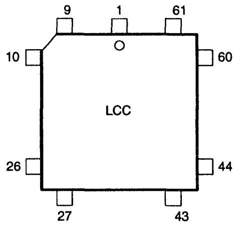


Pin	Function	Pin	Function	Pin	Function
1	I/O-A8	16	GND	31	I/O-D11
2	I/O-A11	17	V_{DD}	32	I/O-D8 (TDO)
3	I/O-A12	18	I/O-C0/CK1	33	I/O-D7
4	GND	19	I/O-C2	34	I/O-D2
5	I/O-A13	20	I/O-C3	35	I/O-D0
6	I/O-A15	21	I/O-C4	36	GND
7	I/O-B15 (TMS)*	22	I/O-C7	37	IN0/CK0
8	I/O-B13	23	I/O-C8	38	IN2-gtsn
9	V_{DD}	24	GND	39	IN1
10	I/O-B10	25	I/O-C13	40	IN3
11	I/O-B8	26	I/O-C15 (TCK)	41	V_{DD}
12	I/O-B4	27	I/O-D15	42	I/O-A0/CK3
13	I/O-B3	28	I/O-D13	43	I/O-A2
14	I/O-B2	29	V_{DD}	44	I/O-A5
15	I/O-B0/CK2	30	I/O-D12		

* THE TEST MODE SELECT (TMS) FUNCTION IS INACTIVE ON NON-ISR ARCHITECTURES.

SP00453

PZ3064 – 68-Pin Plastic Leaded Chip Carrier

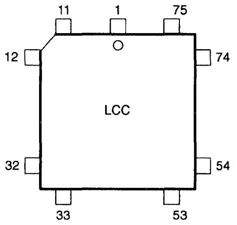


Pin	Function	Pin	Function	Pin	Function
1	IN1	24	I/O-B10	47	I/O-C12
2	IN3	25	I/O-B8	48	GND
3	V_{DD}	26	GND	49	I/O-D13
4	I/O-A0/CK3	27	I/O-B7	50	I/O-C15 (TCK)
5	I/O-A2	28	I/O-B5	51	I/O-D15
6	GND	29	I/O-B4	52	I/O-D13
7	I/O-A3	30	I/O-B3	53	V_{DD}
8	I/O-A4	31	V_{DD}	54	I/O-D12
9	I/O-A5	32	I/O-B2	55	I/O-D11
10	I/O-A7	33	I/O-B0/CK2	56	I/O-D9
11	V_{DD}	34	GND	57	I/O-D8 (TDO)
12	I/O-A8 (TDI)	35	V_{DD}	58	GND
13	I/O-A10	36	I/O-C0/CK1	59	I/O-D7
14	I/O-A11	37	I/O-C2	60	I/O-D6
15	I/O-A12	38	GND	61	I/O-D4
16	GND	39	I/O-C3	62	I/O-D3
17	I/O-A13	40	I/O-C4	63	V_{DD}
18	I/O-A15	41	I/O-C5	64	I/O-D2
19	I/O-B15 (TMS)*	42	I/O-C7	65	I/O-D0
20	I/O-B13	43	V_{DD}	66	GND
21	V_{DD}	44	I/O-C8	67	IN0/CK0
22	I/O-B12	45	I/O-C10	68	IN2-gtsn
23	I/O-B11	46	I/O-C11		

* THE TEST MODE SELECT (TMS) FUNCTION IS INACTIVE ON NON-ISR ARCHITECTURES.

SP00454

PZ3064 – 84-Pin Plastic Leaded Chip Carrier

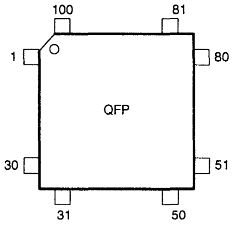


Pin	Function	Pin	Function	Pin	Function
1	IN1	29	I/O-B10	57	I/O-C11
2	IN3	30	I/O-B9	58	I/O-C12
3	V_{DD}	31	I/O-B8	59	GND
4	I/O-A0/CK3	32	GND	60	I/O-C13
5	I/O-A1	33	I/O-B7	61	I/O-C14
6	I/O-A2	34	I/O-B6	62	I/O-C15 (TCK)
7	GND	35	I/O-B5	63	I/O-D15
8	I/O-A3	36	I/O-B4	64	I/O-D14
9	I/O-A4	37	I/O-B3	65	I/O-D13
10	I/O-A5	38	V_{DD}	66	V_{DD}
11	I/O-A6	39	I/O-B2	67	I/O-D12
12	I/O-A7	40	I/O-B1	68	I/O-D11
13	V_{CC}	41	I/O-B0/CK2	69	I/O-D10
14	I/O-A8 (TDI)	42	GND	70	I/O-D9
15	I/O-A9	43	V_{DD}	71	I/O-D8 (TDO)
16	I/O-A10	44	I/O-C0/CK1	72	GND
17	I/O-A11	45	I/O-C1	73	I/O-D7
18	I/O-A12	46	I/O-C2	74	I/O-D6
19	GND	47	GND	75	I/O-D5
20	I/O-A13 *	48	I/O-C3	76	I/O-D4
21	I/O-A14	49	I/O-C4	77	I/O-D3
22	I/O-B15	50	I/O-C5	78	V_{DD}
23	I/O-B15 (TMS)*	51	I/O-C6	79	I/O-D2
24	I/O-B14	52	I/O-C7	80	I/O-D1
25	I/O-B13	53	V_{DD}	81	I/O-D0
26	V_{DD}	54	I/O-C8	82	GND
27	I/O-B12	55	I/O-C9	83	IN0/CK0
28	I/O-B11	56	I/O-C10	84	IN2-gtsn

* THE TEST MODE SELECT (TMS) FUNCTION IS INACTIVE ON NON-ISR ARCHITECTURES.

SP00455

PZ3064 – 100-Pin Plastic Quad Flat Package



Pin	Function	Pin	Function	Pin	Function
1	NC	35	I/O-B3	69	I/O-D12
2	NC	36	V_{DD}	70	I/O-D11
3	I/O-A6	37	I/O-B2	71	I/O-D10
4	I/O-A7	38	I/O-B1	72	NC
5	V_{DD}	39	I/O-B0/CK2	73	I/O-D9
6	I/O-A8 (TDI)	40	GND	74	NC
7	NC	41	V_{DD}	75	I/O-D8 (TDO)
8	I/O-A9	42	I/O-C0/CK1	76	GND
9	NC	43	I/O-C1	77	I/O-D7
10	I/O-A10	44	I/O-C2	78	I/O-D6
11	I/O-A11	45	GND	79	NC
12	I/O-A12	46	I/O-C3	80	NC
13	GND	47	I/O-C4	81	I/O-D5
14	I/O-A13	48	I/O-C5	82	I/O-D4
15	I/O-A14	49	I/O-C6	83	I/O-D3
16	I/O-A15	50	I/O-C7	84	V_{DD}
17	I/O-B15 (TMS)*	51	NC	85	I/O-D2
18	I/O-B14	52	NC	86	I/O-D1
19	I/O-B13	53	V_{DD}	87	I/O-D0
20	V_{DD}	54	I/O-C8	88	GND
21	I/O-B12	55	NC	89	IN0/CK0
22	I/O-B11	56	I/O-C9	90	IN2-gtsn
23	I/O-B10	57	NC	91	IN1
24	NC	58	I/O-C10	92	IN3
25	I/O-B9	59	I/O-C11	93	V_{DD}
26	NC	60	I/O-C12	94	I/O-A0/CK3
27	I/O-B8	61	GND	95	I/O-A1
28	GND	62	I/O-C13	96	I/O-A2
29	NC	63	I/O-C14	97	GND
30	NC	64	I/O-C15 (TCK)	98	I/O-A3
31	I/O-B7	65	I/O-D15	99	I/O-A4
32	I/O-B6	66	I/O-D14	100	I/O-A5
33	I/O-B5	67	I/O-D13		
34	I/O-B4	68	V_{DD}		

* THE TEST MODE SELECT (TMS) FUNCTION IS INACTIVE ON NON-ISR ARCHITECTURES.

SP00456

Package Thermal Characteristics

Philips Semiconductors uses the Temperature Sensitive Parameter (TSP) method to test thermal resistance. This method meets Mil-Std-883C Method 1012.1 and is described in Philips *1995 IC Package Databook*. Thermal resistance varies slightly as a function of input power. As input power increases, thermal resistance changes approximately 5% for a 100% change in power.

Figure 7 is a derating curve for the change in Θ_{JA} with airflow based on wind tunnel measurements. It should be noted that the wind flow dynamics are more complex and turbulent in actual applications than in a wind tunnel. Also, the test boards used in the wind tunnel contribute significantly to forced convection heat transfer, and may not be similar to the actual circuit board, especially in size.

Package	Θ_{JA}
44-pin PLCC	44.8°C/W
44-pin TQFP	60.8°C/W
68-pin PLCC	44.9°C/W
84-pin PLCC	34.7°C/W
100-pin PQFP	44.5°C/W

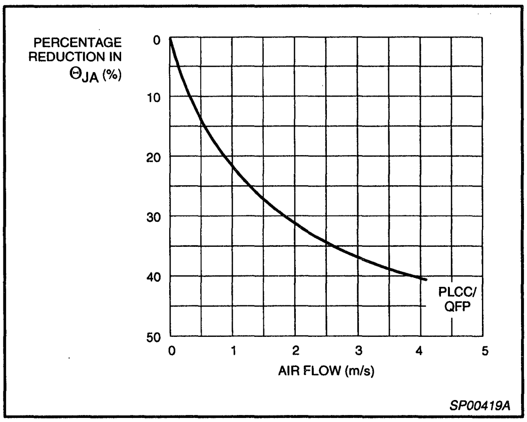


Figure 7. Average Effect of Airflow on Θ_{JA}

64 macrocell CPLD PZ5064

FEATURES

- Industry's first TotalCMOS™ PLD – both CMOS design and process technologies
- Fast Zero Power (FZP™) design technique provides ultra-low power and very high speed
- High speed pin-to-pin delays of 7.5ns
- Ultra-low static power of less than 100μA
- Dynamic power that is 70% lower at 50MHz than competing devices
- 100% routable with 100% utilization while all pins and all macrocells are fixed
- Deterministic timing model that is extremely simple to use
- 4 clocks with programmable polarity at every macrocell
- Support for complex asynchronous clocking
- Innovative XPLA™ architecture combines high speed with extreme flexibility
- 1000 erase/program cycles guaranteed
- 20 years data retention guaranteed
- Logic expandable to 37 product terms
- PCI compliant
- Advanced 0.5μ E^2CMOS process
- Security bit prevents unauthorized access
- Design entry and verification using industry standard and Philips CAE tools
- Reprogrammable using industry standard device programmers
- Innovative Control Term structure provides either sum terms or product terms in each logic block for:
  - Programmable 3-State buffer
  - Asynchronous macrocell register preset/reset
- Programmable global 3-State pin facilitates 'bed of nails' testing without using logic resources
- Available in PLCC, TQFP, and PQFP packages
- Available in both Commercial and Industrial grades

Table 1. PZ5064 Features

	PZ5064
Usable gates	2000
Maximum inputs	68
Maximum I/Os	64
Number of macrocells	64
Propagation delay (ns)	7.5
Packages	44-pin PLCC, 44-pin TQFP, 68-pin PLCC, 84-pin PLCC, 100-pin PQFP

DESCRIPTION

The PZ5064 CPLD (Complex Programmable Logic Device) is the second in a family of Fast Zero Power (FZP™) CPLDs from Philips Semiconductors. These devices combine high speed and zero power in a 64 macrocell CPLD. With the FZP™ design technique, the PZ5064 offers true pin-to-pin speeds of 7.5ns, while simultaneously delivering power that is less than 100μA at standby without the need for 'turbo bits' or other power down schemes. By replacing conventional sense amplifier methods for implementing product terms (a technique that has been used in PLDs since the bipolar era) with a cascaded chain of pure CMOS gates, the dynamic power is also substantially lower than any competing CPLD – 70% lower at 50MHz. These devices are the first TotalCMOS™ PLDs, as they use both a CMOS process technology **and** the patented full CMOS FZP™ design technique. For 3V applications, Philips also offers the high speed PZ3064 CPLD that offers these features in a full 3V implementation.

The Philips FZP™ CPLDs introduce the new patent-pending XPLA™ (eXtended Programmable Logic Array) architecture. The XPLA™ architecture combines the best features of both PLA and PAL™ type structures to deliver high speed and flexible logic allocation that results in superior ability to make design changes with fixed pinouts. The XPLA™ structure in each logic block provides a fast 7.5ns PAL™ path with 5 dedicated product terms per output. This PAL™ path is joined by an additional PLA structure that deploys a pool of 32 product terms to a fully programmable OR array that can allocate the PLA product terms to any output in the logic block. This combination allows logic to be allocated efficiently throughout the logic block and supports as many as 37 product terms on an output. The speed with which logic is allocated from the PLA array to an output is only 2.0ns, regardless of the number of PLA product terms used, which results in worst case t_{PD}'s of only 9.5ns from any pin to any other pin. In addition, logic that is common to multiple outputs can be placed on a single PLA product term and shared across multiple outputs via the OR array, effectively increasing design density.

The PZ5064 CPLDs are supported by industry standard CAE tools (Cadence, Mentor, Synopsys, Synario, Viewlogic, MINC), using text (Abel, VHDL, Verilog) and/or schematic entry. Design verification uses industry standard simulators for functional and timing simulation. Development is supported on personal computer, Sparc, and HP platforms. Device fitting uses either Minc or Philips Semiconductors-developed tools.

The PZ5064 CPLD is reprogrammable using industry standard device programmers from vendors such as Data I/O, BP Microsystems, SMS, and others.

PAL is a registered trademark of Advanced Micro Devices, Inc.

ORDERING INFORMATION

ORDER CODE	DESCRIPTION	DESCRIPTION	DRAWING NUMBER
PZ5064-7A44	44-pin PLCC, 7.5ns t_{PD}	Commercial temp range, 5 volt power supply, ± 5%	SOT187-2
PZ5064-10A44	44-pin PLCC, 10ns t_{PD}	Commercial temp range, 5 volt power supply, ± 5%	SOT187-2
PZ5064I10A44	44-pin PLCC, 10ns t_{PD}	Industrial temp range, 5 volt power supply, ± 10%	SOT187-2
PZ5064I12A44	44-pin PLCC, 12ns t_{PD}	Industrial temp range, 5 volt power supply, ± 10%	SOT187-2
PZ5064-7BC	44-pin TQFP, 7.5ns t_{PD}	Commercial temp range, 5 volt power supply, ± 5%	SOT376-1
PZ5064-10BC	44-pin TQFP, 10ns t_{PD}	Commercial temp range, 5 volt power supply, ± 5%	SOT376-1
PZ5064I10BC	44-pin TQFP, 10ns t_{PD}	Industrial temp range, 5 volt power supply, ± 10%	SOT376-1
PZ5064I12BC	44-pin TQFP, 12ns t_{PD}	Industrial temp range, 5 volt power supply, ± 10%	SOT376-1
PZ5064-7A68	68-pin PLCC, 7.5ns t_{PD}	Commercial temp range, 5 volt power supply, ± 5%	SOT188-3
PZ5064-10A68	68-pin PLCC, 10ns t_{PD}	Commercial temp range, 5 volt power supply, ± 5%	SOT188-3
PZ5064I10A68	68-pin PLCC, 10ns t_{PD}	Industrial temp range, 5 volt power supply, ± 10%	SOT188-3
PZ5064I12A68	68-pin PLCC, 12ns t_{PD}	Industrial temp range, 5 volt power supply, ± 10%	SOT188-3
PZ5064-7A84	84-pin PLCC, 7.5ns t_{PD}	Commercial temp range, 5 volt power supply, ± 5%	SOT189-3
PZ5064-10A84	84-pin PLCC, 10ns t_{PD}	Commercial temp range, 5 volt power supply, ± 5%	SOT189-3
PZ5064I10A84	84-pin PLCC, 10ns t_{PD}	Industrial temp range, 5 volt power supply, ± 10%	SOT189-3
PZ5064I12A84	84-pin PLCC, 12ns t_{PD}	Industrial temp range, 5 volt power supply, ± 10%	SOT189-3
PZ5064-7BB1	100-pin PQFP, 7.5ns t_{PD}	Commercial temp range, 5 volt power supply, ± 5%	SOT382-1
PZ5064-10BB1	100-pin PQFP, 10ns t_{PD}	Commercial temp range, 5 volt power supply, ± 5%	SOT382-1
PZ5064I10BB1	100-pin PQFP, 10ns t_{PD}	Industrial temp range, 5 volt power supply, ± 10%	SOT382-1
PZ5064I12BB1	100-pin PQFP, 12ns t_{PD}	Industrial temp range, 5 volt power supply, ± 10%	SOT382-1

XPLA™ ARCHITECTURE

Figure 1 shows a high level block diagram of a 64 macrocell device implementing the XPLA™ architecture. The XPLA™ architecture consists of logic blocks that are interconnected by a Zero-power Interconnect Array (ZIA). The ZIA is a virtual crosspoint switch. Each logic block is essentially a 36V16 device with 36 inputs from the ZIA and 16 macrocells. Each logic block also provides 32 ZIA feedback paths from the macrocells and I/O pins.

From this point of view, this architecture looks like many other CPLD architectures. What makes the CoolRunner™ family unique is what is inside each logic block and the design technique used to implement these logic blocks. The contents of the logic block will be described next.

Logic Block Architecture

Figure 2 illustrates the logic block architecture. Each logic block contains control terms, a PAL array, a PLA array, and 16 macrocells. the 6 control terms can individually be configured as either SUM or PRODUCT terms, and are used to control the preset/reset and output enables of the 16 macrocells' flip-flops. The PAL array consists of a programmable AND array with a fixed OR array, while the PLA array consists of a programmable AND array with a programmable OR array. The PAL array provides a high speed path through the array, while the PLA array provides increased product term density.

Each macrocell has 5 dedicated product terms from the PAL array. The pin-to-pin t_{PD} of the PZ5064 device through the PAL array is 7.5ns. If a macrocell needs more than 5 product terms, it simply gets the additional product terms from the PLA array. The PLA array consists of 32 product terms, which are available for use by all 16 macrocells. The additional propagation delay incurred by a macrocell using 1 or all 32 PLA product terms is just 2.0ns. So the total pin-to-pin t_{PD} for the PZ5064 using 6 to 37 product terms is 9.5ns (7.5ns for the PAL + 2.0ns for the PLA).

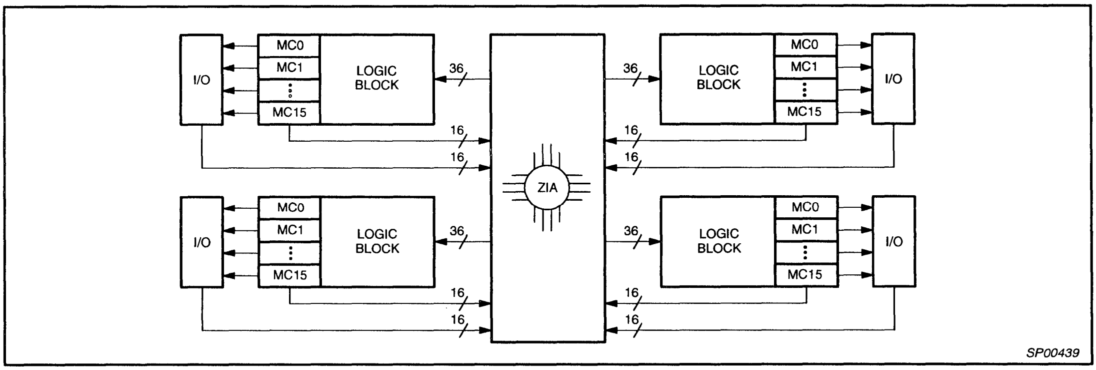


Figure 1. Philips XPLA CPLD Architecture

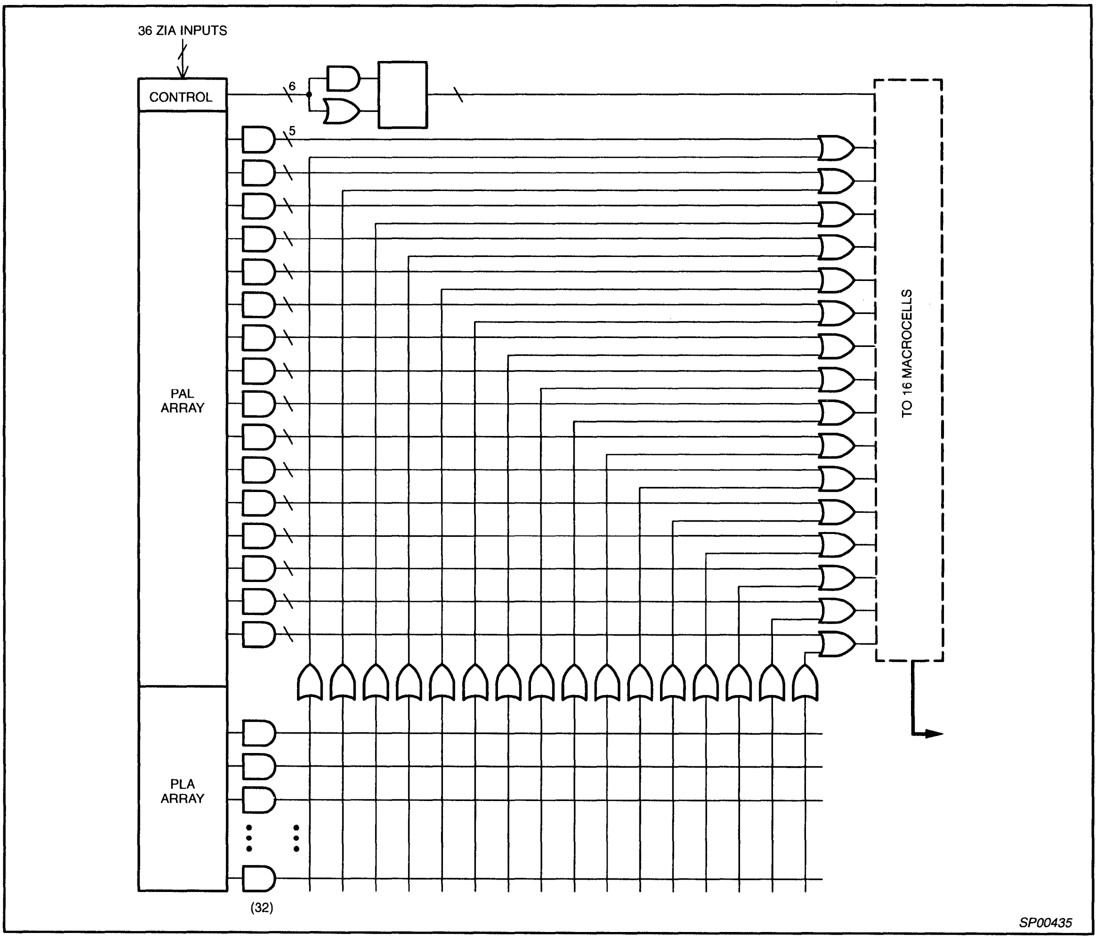


Figure 2. Philips Logic Block Architecture

Macrocell Architecture

Figure 3 shows the architecture of the macrocell used in the CoolRunner™ family. The macrocell consists of a flip-flop that can be configured as either a D or T type. A D-type flip-flop is generally more useful for implementing state machines and data buffering. A T-type flip-flop is generally more useful in implementing counters. All CoolRunner™ family members provide both synchronous and asynchronous clocking and provide the ability to clock off either the falling or rising edges of these clocks. These devices are designed such that the skew between the rising and falling edges of a clock are minimized for clocking integrity. There are 4 clocks available on the PZ5064 device. Clock 0 (CLK0) is designated as the "synchronous" clock and must be driven by an external source. Clock 1 (CLK1), Clock 2 (CLK2), and Clock 3 (CLK3) can either be used as a synchronous clock (driven by an external source) or as an asynchronous clock (driven by a macrocell equation).

Two of the control terms (CT0 and CT1) are used to control the Preset/Reset of the macrocell's flip-flop. The Preset/Reset feature for each macrocell can also be disabled. Note that the Power-on Reset leaves all macrocells in the "zero" state when power is properly applied. The other 4 control terms (CT2–CT5) can be used to control the Output Enable of the macrocell's output buffers. The reason there are as many control terms dedicated for the Output Enable of the macrocell is to insure that all CoolRunner™ devices are PCI compliant. The macrocell's output buffers can also be always enabled or disabled. All CoolRunner™ devices also provide a Global Tri-State ($\overline{\text{GTS}}$) pin, which, when pulled Low, will 3-State all the outputs of the device. This pin is provided to support "In-Circuit Testing" or "Bed-of-Nails Testing".

There are two feedback paths to the ZIA: one from the macrocell, and one from the I/O pin. The ZIA feedback path before the output buffer is the macrocell feedback path, while the ZIA feedback path after the output buffer is the I/O pin ZIA path. When the macrocell is used as an output, the output buffer is enabled, and the macrocell feedback path can be used to feedback the logic implemented in the macrocell. When the I/O pin is used as an input, the output buffer will be 3-Stated and the input signal will be fed into the ZIA via the I/O feedback path, and the logic implemented in the buried macrocell can be fed back to the ZIA via the macrocell feedback path. It should be noted that unused inputs or I/Os should be properly terminated.

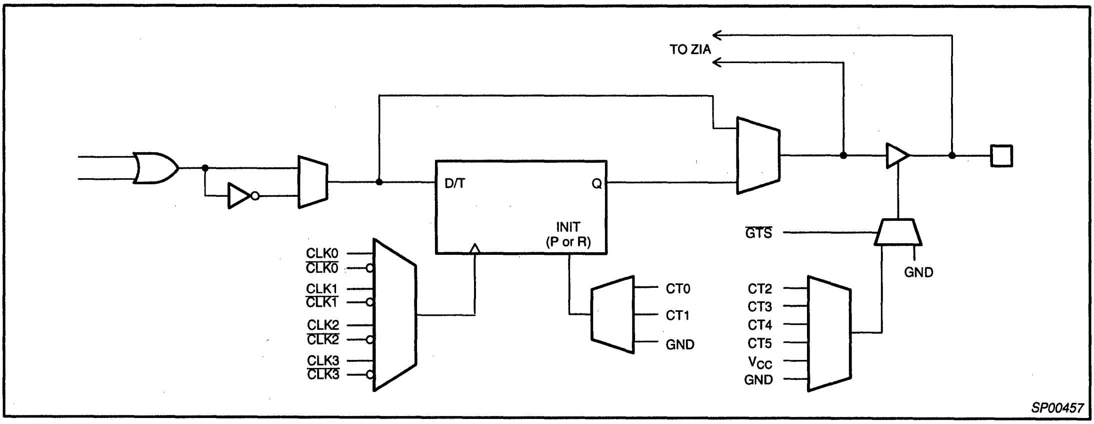


Figure 3. PZ5064 Macrocell Architecture

Simple Timing Model

Figure 4 shows the CoolRunner™ Timing Model. The CoolRunner™ timing model looks very much like a 22V10 timing model in that there are three main timing parameters, including t_{PD}, t_{SU}, and t_{CO}. In other competing architectures, the user may be able to fit the design into the CPLD, but is not sure whether system timing requirements can be met until after the design has been fit into the device. This is because the timing models of competing architectures are very complex and include such things as timing dependencies on the number of parallel expanders borrowed, sharable expanders, varying number of X and Y routing channels used, etc. In the XPLA™ architecture, the user knows up front whether the design will meet system timing requirements. This is due to the simplicity of the timing model. For example, in the PZ5064 device, the user knows up front that if a given output uses 5 product terms or less, the t_{PD} = 7.5ns, the t_{SU_PAL} = 4ns, and the t_{CO} = 5.5ns. If an output is using 6 to 37 product terms, an additional 2ns must be added to the t_{PD} and t_{SU} timing parameters to account for the time to propagate through the PLA array.

TotalCMOS™ Design Technique for Fast Zero Power

Philips is the first to offer a TotalCMOS™ CPLD, both in process technology and design technique. Philips employs a cascade of CMOS gates to implement its Sum of Products instead of the traditional sense amp approach. This CMOS gate implementation allows Philips to offer CPLDs which are both high performance and low power, breaking the paradigm that to have low power, you must have low performance. Refer to Figure 5 and Table 2 showing the I_{DD} vs. Frequency of our PZ5064 TotalCMOS™ CPLD.

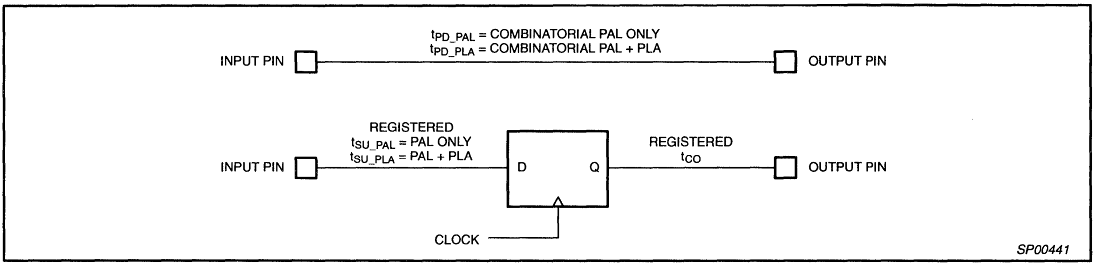


Figure 4. CoolRunner™ Timing Model

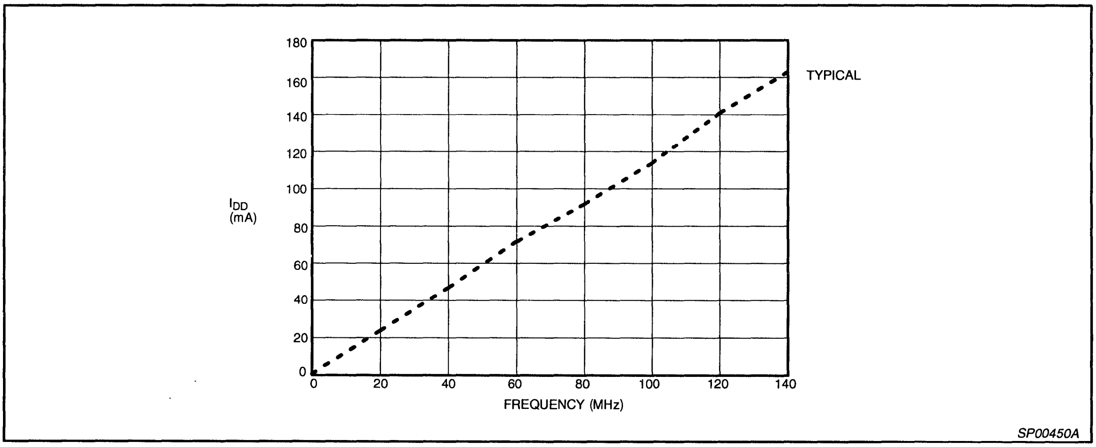


Figure 5. I_{DD} vs. Frequency @ V_{DD} = 5.0V, 25°C

Table 2. I_{DD} vs. Frequency

V_{DD} = 5.00V

FREQUENCY (MHz)	0	20	40	60	80	100	120	140
Typical I_{DD} (mA)	0.08	24	47	72	92	114	141	163

ABSOLUTE MAXIMUM RATINGS[1]

SYMBOL	PARAMETER	MIN.	MAX.	UNIT
V_{DD}	Supply voltage	–0.5	7.0	V
V_I	Input voltage	–1.2	V_{DD}+0.5	V
V_{OUT}	Output voltage	–0.5	V_{DD}+0.5	V
I_{IN}	Input current	–30	30	mA
I_{OUT}	Output current	–100	100	mA
T_J	Maximum junction temperature	–40	150	°C
T_{str}	Storage temperature	–65	150	°C

NOTE:

1. Stresses above those listed may cause malfunction or permanent damage to the device. This is a stress rating only. Functional operation at these or any other condition above those indicated in the operational and programming specification is not implied.

OPERATING RANGE

PRODUCT GRADE	TEMPERATURE	VOLTAGE
Commercial	0 to +70°C	5.0 ±5% V
Industrial	–40 to +85°C	5.0 ±10% V

DC ELECTRICAL CHARACTERISTICS FOR COMMERCIAL GRADE DEVICES

Commercial: 0°C ≤ T_{amb} ≤ +70°C; 4.75V ≤ V_{DD} ≤ 5.25V

SYMBOL	PARAMETER	TEST CONDITIONS	MIN.	MAX.	UNIT
V_{IL}	Input voltage low	V_{DD} = 4.75V		0.8	V
V_{IH}	Input voltage high	V_{DD} = 5.25V	2.0		V
V_I	Input clamp voltage	V_{DD} = 4.75V, I_{IN} = –18mA		–1.2	V
V_{OL}	Output voltage low	V_{DD} = 4.75V, I_{OL} = 12mA		0.5	V
V_{OH}	Output voltage high	V_{DD} = 4.75V, I_{OH} = –12mA	2.4		V
I_I	Input leakage current	V_{IN} = 0 to V_{DD}	–10	10	µA
I_{OZ}	3-Stated output leakage current	V_{IN} = 0 to V_{DD}	–10	10	µA
I_{DDQ}	Standby current	V_{DD} = 5.25V, T_{amb} = 0°C		80	µA
I_{DDD}[1]	Dynamic current	V_{DD} = 5.25V, T_{amb} = 0°C @ 1MHz		3	mA
		V_{DD} = 5.25V, T_{amb} = 0°C @ 50MHz		65	mA
I_{OS}	Short circuit output current	1 pin at a time for no longer than 1 second	–50	–200	mA
C_{IN}	Input pin capacitance	T_{amb} = 25°C, f = 1MHz		8	pF
C_{CLK}	Clock input capacitance	T_{amb} = 25°C, f = 1MHz	5	12	pF
$C_{I/O}$	I/O pin capacitance	T_{amb} = 25°C, f = 1MHz		10	pF

NOTE:

1. This parameter measured with a 16-bit, loadable up/down counter loaded into every logic block, with all outputs enabled and unloaded. Inputs are tied to V_{DD} or ground. This parameter guaranteed by design and characterization, not testing.

AC ELECTRICAL CHARACTERISTICS[1] FOR COMMERCIAL GRADE DEVICES

Commercial: 0°C ≤ T_{amb} ≤ +70°C; 4.75V ≤ V_{DD} ≤ 5.25V

SYMBOL	PARAMETER	–7		–10		UNIT
		MIN.	MAX.	MIN.	MAX.	
t_{PD_PAL}	Propagation delay time, input (or feedback node) to output through PAL	2	7.5	2	10	ns
t_{PD_PLA}	Propagation delay time, input (or feedback node) to output through PAL & PLA	3	9.5	3	12.5	ns
t_{CO}	Clock to out delay time	2	5.5	2	7	ns
t_{SU_PAL}	Setup time (from input or feedback node) through PAL	4		6		ns
t_{SU_PLA}	Setup time (from input or feedback node) through PAL + PLA	6		8.5		ns
t_H	Hold time		0		0	ns
t_{CH}	Clock High time	4		5		ns
t_{CL}	Clock Low time	4		5		ns
t_R	Input Rise time		20		20	ns
t_F	Input Fall time		20		20	ns
f_{MAX1}	Maximum FF toggle rate[2] (1/t_{CH} + t_{CL})	125		100		MHz
f_{MAX2}	Maximum internal frequency[2] (1/t_{SUPAL} + t_{CF})	125		87		MHz
f_{MAX3}	Maximum external frequency[2] (1/t_{SUPAL} + t_{CO})	105		77		MHz
t_{BUF}	Output buffer delay time		1.5		1.5	ns
t_{PDF_PAL}	Input (or feedback node) to internal feedback node delay time through PAL		6		8.5	ns
t_{PDF_PLA}	Input (or feedback node) to internal feedback node delay time through PAL+PLA		8		11	ns
t_{CF}	Clock to internal feedback node delay time		4		5.5	ns
t_{INIT}	Delay from valid V_{DD} to valid reset		50		50	µs
t_{ER}	Input to output disable[3]		10		12	ns
t_{EA}	Input to output valid		10		12	ns
t_{RP}	Input to register preset		10		12.5	ns
t_{RR}	Input to register reset		10		12.5	ns

NOTES:

1. Specifications measured with one output switching. See Figure 6 and Table 3 for derating.
2. This parameter guaranteed by design and characterization, not by test.
3. Output C_L = 5pF.

DC ELECTRICAL CHARACTERISTICS FOR INDUSTRIAL GRADE DEVICES

Industrial: –40°C ≤ T_{amb} ≤ +85°C; 4.5V ≤ V_{DD} ≤ 5.5V

SYMBOL	PARAMETER	TEST CONDITIONS	MIN.	MAX.	UNIT
V_{IL}	Input voltage low	V_{DD} = 4.5V		0.8	V
V_{IH}	Input voltage high	V_{DD} = 5.5V	2.0		V
V_I	Input clamp voltage	V_{DD} = 4.5V, I_{IN} = –18mA		–1.2	V
V_{OL}	Output voltage low	V_{DD} = 4.5V, I_{OL} = 12mA		0.5	V
V_{OH}	Output voltage high	V_{DD} = 4.5V, I_{OH} = –12mA	2.4		V
I_I	Input leakage current	V_{IN} = 0 to V_{DD}	–10	10	µA
I_{OZ}	3-Stated output leakage current	V_{IN} = 0 to V_{DD}	–10	10	µA
I_{DDQ}	Standby current	V_{DD} = 5.5V, T_{amb} = –40°C		100	µA
I_{DDD}[1]	Dynamic current	V_{DD} = 5.5V, T_{amb} = –40°C @ 1MHz		4	mA
		V_{DD} = 5.5V, T_{amb} = –40°C @ 50MHz		70	mA
I_{OS}	Short circuit output current	1 pin at a time for no longer than 1 second	–50	–230	mA
C_{IN}	Input pin capacitance	T_{amb} = 25°C, f = 1MHz		8	pF
C_{CLK}	Clock input capacitance	T_{amb} = 25°C, f = 1MHz	5	12	pF
$C_{I/O}$	I/O pin capacitance	T_{amb} = 25°C, f = 1MHz		10	pF

NOTE:

1. This parameter measured with a 16–bit, loadable up/down counter loaded into every logic block, with all outputs enabled and unloaded. Inputs are tied to V_{DD} or ground. This parameter guaranteed by design and characterization, not testing.

AC ELECTRICAL CHARACTERISTICS[1] FOR INDUSTRIAL GRADE DEVICES

Industrial: –40°C ≤ T_{amb} ≤ +85°C; 4.5V ≤ V_{DD} ≤ 5.5V

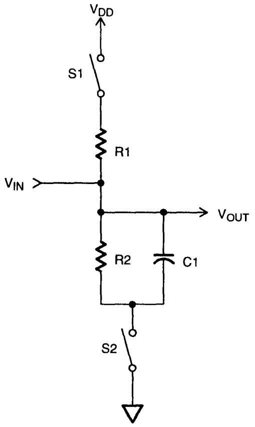
SYMBOL	PARAMETER	I10		I12		UNIT
		MIN.	MAX.	MIN.	MAX.	
t_{PD_PAL}	Propagation delay time, input (or feedback node) to output through PAL	2	10	2	12	ns
t_{PD_PLA}	Propagation delay time, input (or feedback node) to output through PAL & PLA	3	12.5	3	14.5	ns
t_{CO}	Clock to out delay time	2	7	2	8	ns
t_{SU_PAL}	Setup time (from input or feedback node) through PAL	6		7		ns
t_{SU_PLA}	Setup time (from input or feedback node) through PAL + PLA	8.5		9.5		ns
t_H	Hold time		0		0	ns
t_{CH}	Clock High time	5		5		ns
t_{CL}	Clock Low time	5		5		ns
t_R	Input Rise time		20		20	ns
t_F	Input Fall time		20		20	ns
f_{MAX1}	Maximum FF toggle rate[2] (1/t_{CH} + t_{CL})	100		100		MHz
f_{MAX2}	Maximum internal frequency[2] (1/t_{SUPAL} + t_{CF})	87		74		MHz
f_{MAX3}	Maximum external frequency[2] (1/t_{SUPAL} + t_{CO})	77		67		MHz
t_{BUF}	Output buffer delay time		1.5		1.5	ns
t_{PDF_PAL}	Input (or feedback node) to internal feedback node delay time through PAL		8.5		10.5	ns
t_{PDF_PLA}	Input (or feedback node) to internal feedback node delay time through PAL+PLA		11		13	ns
t_{CF}	Clock to internal feedback node delay time		5.5		6.5	ns
t_{INIT}	Delay from valid V_{DD} to valid reset		50		50	µs
t_{ER}	Input to output disable[3]		12		13	ns
t_{EA}	Input to output valid		12		13	ns
t_{RP}	Input to register preset		12.5		13.5	ns
t_{RR}	Input to register reset		12.5		13.5	ns

NOTES:

1. Specifications measured with one output switching. See Figure 6 and Table 3 for derating.
2. This parameter guaranteed by design and characterization, not by test.
3. Output C_L = 5pF.

SWITCHING CHARACTERISTICS

The test load circuit and load values for the AC Electrical Characteristics are illustrated below.

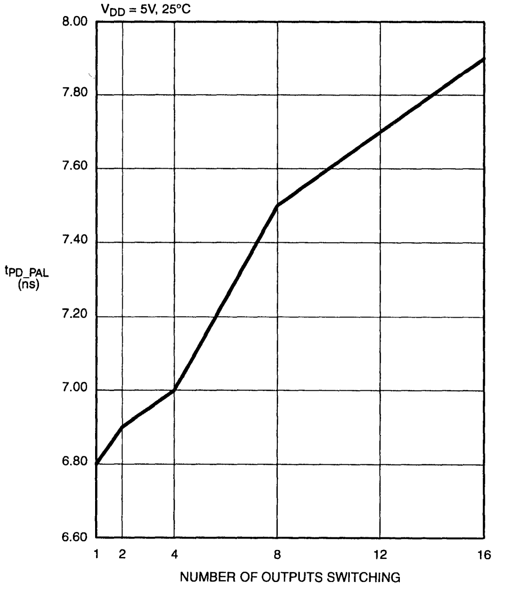


COMPONENT	VALUES
R1	470Ω
R2	250Ω
C1	35pF

MEASUREMENT	S1	S2
t_{PZH}	Open	Closed
t_{PZL}	Closed	Open
t_P	Closed	Closed

NOTE: For t_{PHZ} and t_{PLZ} C = 5pF

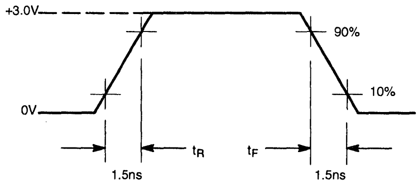
SP00458A



SP00451A

Figure 6. t_{PD_PAL} vs. Outputs Switching

VOLTAGE WAVEFORM



MEASUREMENTS:
All circuit delays are measured at the +1.5V level of inputs and outputs, unless otherwise specified.

Input Pulses

SP00368

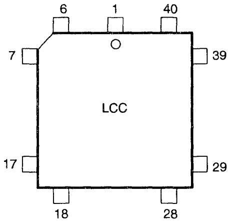
Table 3. t_{PD_PAL} vs. Number of Outputs Switching

V_{DD} = 5.00V

NUMBER OF OUTPUTS	1	2	4	8	12	16
Typical (ns)	6.8	6.9	7.0	7.5	7.7	7.9

PIN DESCRIPTIONS

PZ5064 – 44-Pin Plastic Leaded Chip Carrier

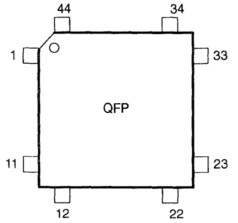


Pin	Function	Pin	Function	Pin	Function
1	IN1	16	I/O-B10	31	I/O-C13
2	IN3	17	I/O-B8	32	I/O-C15 (TCK)
3	V_{DD}	18	I/O-B4	33	I/O-D15
4	I/O-A0/CK3	19	I/O-B3	34	I/O-D13
5	I/O-A2	20	I/O-B2	35	V_{DD}
6	I/O-A5	21	I/O-B0/CK2	36	I/O-D12
7	I/O-A8 (TDI)	22	GND	37	I/O-D11
8	I/O-A11	23	V_{DD}	38	I/O-D8 (TDO)
9	I/O-A12	24	I/O-C0/CK1	39	I/O-D7
10	GND	25	I/O-C2	40	I/O-D2
11	I/O-A13	26	I/O-C3	41	I/O-D0
12	I/O-A15	27	I/O-C4	42	GND
13	I/O-B15 (TMS)*	28	I/O-C7	43	IN0-CK0
14	I/O-B13	29	I/O-C8	44	IN2-gtsn
15	V_{DD}	30	GND		

* THE TEST MODE SELECT (TMS) FUNCTION IS INACTIVE ON NON-ISR ARCHITECTURES.

SP00452A

PZ5064 – 44-Pin Thin Quad Flat Package

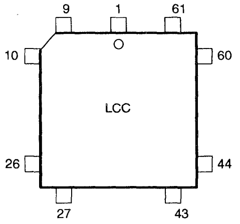


Pin	Function	Pin	Function	Pin	Function
1	I/O-A8	16	GND	31	I/O-D11
2	I/O-A11	17	V_{DD}	32	I/O-D8 (TDO)
3	I/O-A12	18	I/O-C0/CK1	33	I/O-D7
4	GND	19	I/O-C2	34	I/O-D2
5	I/O-A13	20	I/O-C3	35	I/O-D0
6	I/O-A15	21	I/O-C4	36	GND
7	I/O-B15 (TMS)*	22	I/O-C7	37	IN0/CK0
8	I/O-B13	23	I/O-C8	38	IN2-gtsn
9	V_{DD}	24	GND	39	IN1
10	I/O-B10	25	I/O-C13	40	IN3
11	I/O-B8	26	I/O-C15 (TCK)	41	V_{DD}
12	I/O-B4	27	I/O-D15	42	I/O-A0/CK3
13	I/O-B3	28	I/O-D13	43	I/O-A2
14	I/O-B2	29	V_{DD}	44	I/O-A5
15	I/O-B0/CK2	30	I/O-D12		

* THE TEST MODE SELECT (TMS) FUNCTION IS INACTIVE ON NON-ISR ARCHITECTURES.

SP00453

PZ5064 – 68-Pin Plastic Leaded Chip Carrier

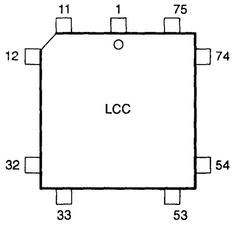


Pin	Function	Pin	Function	Pin	Function
1	IN1	24	I/O-B10	47	I/O-C12
2	IN3	25	I/O-B8	48	GND
3	V_{DD}	26	GND	49	I/O-D13
4	I/O-A0/CK3	27	I/O-B7	50	I/O-C15 (TCK)
5	I/O-A2	28	I/O-B5	51	I/O-D15
6	GND	29	I/O-B4	52	I/O-D13
7	I/O-A3	30	I/O-B3	53	V_{DD}
8	I/O-A4	31	V_{DD}	54	I/O-D12
9	I/O-A5	32	I/O-B2	55	I/O-D11
10	I/O-A7	33	I/O-B0/CK2	56	I/O-D9
11	V_{DD}	34	GND	57	I/O-D8 (TDO)
12	I/O-A8 (TDI)	35	V_{DD}	58	GND
13	I/O-A10	36	I/O-C0/CK1	59	I/O-D7
14	I/O-A11	37	I/O-C2	60	I/O-D6
15	I/O-A12	38	GND	61	I/O-D4
16	GND	39	I/O-C3	62	I/O-D3
17	I/O-A13	40	I/O-C4	63	V_{DD}
18	I/O-A15	41	I/O-C5	64	I/O-D2
19	I/O-B15 (TMS)*	42	I/O-C7	65	I/O-D0
20	I/O-B13	43	V_{DD}	66	GND
21	V_{DD}	44	I/O-C8	67	IN0/CK0
22	I/O-B12	45	I/O-C10	68	IN2-gtsn
23	I/O-B11	46	I/O-C11		

* THE TEST MODE SELECT (TMS) FUNCTION IS INACTIVE ON NON-ISR ARCHITECTURES.

SP00454

PZ5064 – 84-Pin Plastic Leaded Chip Carrier

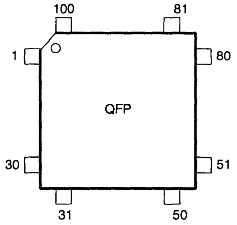


Pin	Function	Pin	Function	Pin	Function
1	IN1	29	I/O-B10	57	I/O-C11
2	IN3	30	I/O-B9	58	I/O-C12
3	V_{DD}	31	I/O-B8	59	GND
4	I/O-A0/CK3	32	GND	60	I/O-C13
5	I/O-A1	33	I/O-B7	61	I/O-C14
6	I/O-A2	34	I/O-B6	62	I/O-C15 (TCK)
7	GND	35	I/O-B5	63	I/O-D15
8	I/O-A3	36	I/O-B4	64	I/O-D14
9	I/O-A4	37	I/O-B3	65	I/O-D13
10	I/O-A5	38	V_{DD}	66	V_{DD}
11	I/O-A6	39	I/O-B2	67	I/O-D12
12	I/O-A7	40	I/O-B1	68	I/O-D11
13	V_{CC}	41	I/O-B0/CK2	69	I/O-D10
14	I/O-A8 (TDI)	42	GND	70	I/O-D9
15	I/O-A9	43	V_{DD}	71	I/O-D8 (TDO)
16	I/O-A10	44	I/O-C0/CK1	72	GND
17	I/O-A11	45	I/O-C1	73	I/O-D7
18	I/O-A12	46	I/O-C2	74	I/O-D6
19	GND	47	GND	75	I/O-D5
20	I/O-A13	48	I/O-C3	76	I/O-D4
21	I/O-A14	49	I/O-C4	77	I/O-D3
22	I/O-B15	50	I/O-C5	78	V_{DD}
23	I/O-B15 (TMS)*	51	I/O-C6	79	I/O-D2
24	I/O-B14	52	I/O-C7	80	I/O-D1
25	I/O-B13	53	V_{DD}	81	I/O-D0
26	V_{DD}	54	I/O-C8	82	GND
27	I/O-B12	55	I/O-C9	83	IN0/CK0
28	I/O-B11	56	I/O-C10	84	IN2-gtsn

* THE TEST MODE SELECT (TMS) FUNCTION IS INACTIVE ON NON-ISR ARCHITECTURES.

SP00455

PZ5064 – 100-Pin Plastic Quad Flat Package



Pin	Function	Pin	Function	Pin	Function
1	NC	35	I/O-B3	69	I/O-D12
2	NC	36	V_{DD}	70	I/O-D11
3	I/O-A6	37	I/O-B2	71	I/O-D10
4	I/O-A7	38	I/O-B1	72	NC
5	V_{DD}	39	I/O-B0/CK2	73	I/O-D9
6	I/O-A8 (TDI)	40	GND	74	NC
7	NC	41	V_{DD}	75	I/O-D8 (TDO)
8	I/O-A9	42	I/O-C0/CK1	76	GND
9	NC	43	I/O-C1	77	I/O-D7
10	I/O-A10	44	I/O-C2	78	I/O-D6
11	I/O-A11	45	GND	79	NC
12	I/O-A12	46	I/O-C3	80	NC
13	GND	47	I/O-C4	81	I/O-D5
14	I/O-A13	48	I/O-C5	82	I/O-D4
15	I/O-A14	49	I/O-C6	83	I/O-D3
16	I/O-A15	50	I/O-C7	84	V_{DD}
17	I/O-B15 (TMS)*	51	NC	85	I/O-D2
18	I/O-B14	52	NC	86	I/O-D1
19	I/O-B13	53	V_{DD}	87	I/O-D0
20	V_{DD}	54	I/O-C8	88	GND
21	I/O-B12	55	NC	89	IN0/CK0
22	I/O-B11	56	I/O-C9	90	IN2-gtsn
23	I/O-B10	57	NC	91	IN1
24	NC	58	I/O-C10	92	IN3
25	I/O-B9	59	I/O-C11	93	V_{DD}
26	NC	60	I/O-C12	94	I/O-A0/CK3
27	I/O-B8	61	GND	95	I/O-A1
28	GND	62	I/O-C13	96	I/O-A2
29	NC	63	I/O-C14	97	GND
30	NC	64	I/O-C15 (TCK)	98	I/O-A3
31	I/O-B7	65	I/O-D15	99	I/O-A4
32	I/O-B6	66	I/O-D14	100	I/O-A5
33	I/O-B5	67	I/O-D13		
34	I/O-B4	68	V_{DD}		

* THE TEST MODE SELECT (TMS) FUNCTION IS INACTIVE ON NON-ISR ARCHITECTURES.

SP00456

Package Thermal Characteristics

Philips Semiconductors uses the Temperature Sensitive Parameter (TSP) method to test thermal resistance. This method meets Mil-Std-883C Method 1012.1 and is described in Philips *1995 IC Package Databook*. Thermal resistance varies slightly as a function of input power. As input power increases, thermal resistance changes approximately 5% for a 100% change in power.

Figure 7 is a derating curve for the change in Θ_{JA} with airflow based on wind tunnel measurements. It should be noted that the wind flow dynamics are more complex and turbulent in actual applications than in a wind tunnel. Also, the test boards used in the wind tunnel contribute significantly to forced convection heat transfer, and may not be similar to the actual circuit board, especially in size.

Package	Θ_{JA}
44-pin PLCC	44.9°C/W
44-pin TQFP	60.8°C/W
68-pin PLCC	44.9°C/W
84-pin PLCC	34.7°C/W
100-pin PQFP	44.5°C/W

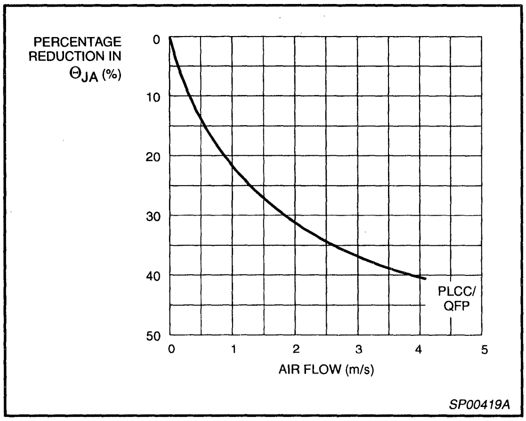


Figure 7. Average Effect of Airflow on Θ_{JA}

128 macrocell CPLD

PZ3128

FEATURES

- Industry's first TotalCMOS™ PLD – both CMOS design and process technologies
- Fast Zero Power (FZP™) design technique provides ultra-low power and very high speed
- IEEE 1149.1–compliant, JTAG Testing Capability
  - 4 pin JTAG interface (TCK, TMS, TDI, TDO)
  - IEEE 1149.1 TAP Controller
  - JTAG commands include: Bypass, Sample/Preload, Extest, Usercode, Idcode, HighZ
- 3.3 Volt, In–System Programmable (ISP) using the JTAG interface
  - On–chip supervoltage generation
  - ISP commands include: Enable, Erase, Program, Verify
  - Supported by multiple ISP programming platforms
- High speed pin-to-pin delays of 12ns
- Ultra-low static power of less than 100μA
- Dynamic power that is 70% lower at 50MHz than competing devices
- 100% routable with 100% utilization while all pins and all macrocells are fixed
- Deterministic timing model that is extremely simple to use
- 4 clocks with programmable polarity at every macrocell
- Support for complex asynchronous clocking
- Innovative XPLA™ architecture combines high speed with extreme flexibility
- 1000 erase/program cycles guaranteed
- 20 years data retention guaranteed
- Logic expandable to 37 product terms
- PCI compliant
- Advanced 0.5μ E^{2}CMOS process
- Security bit prevents unauthorized access
- Design entry and verification using industry standard and Philips CAE tools
- Reprogrammable using industry standard device programmers
- Innovative Control Term structure provides either sum terms or product terms in each logic block for:
  - Programmable 3-State buffer
  - Asynchronous macrocell register preset/reset
- Programmable global 3-State pin facilitates 'bed of nails' testing without using logic resources
- Available in PLCC, TQFP, and PQFP packages
- Available in both Commercial and Industrial grades

Table 1. PZ3128 Features

	PZ3128
Usable gates	4000
Maximum inputs	100
Maximum I/Os	96
Number of macrocells	128
Propagation delay (ns)	12.0
Packages	84-pin PLCC, 100-pin PQFP, 100-pin TQFP, 128-pin LQFP, 160-pin PQFP

DESCRIPTION

The PZ3128 CPLD (Complex Programmable Logic Device) is the third in a family of Fast Zero Power (FZP™) CPLDs from Philips Semiconductors. These devices combine high speed and zero power in a 128 macrocell CPLD. With the FZP™ design technique, the PZ3128 offers true pin-to-pin speeds of 12ns, while simultaneously delivering power that is less than 100μA at standby without the need for 'turbo bits' or other power down schemes. By replacing conventional sense amplifier methods for implementing product terms (a technique that has been used in PLDs since the bipolar era) with a cascaded chain of pure CMOS gates, the dynamic power is also substantially lower than any competing CPLD – 70% lower at 50MHz. These devices are the first TotalCMOS™ PLDs, as they use both a CMOS process technology **and** the patented full CMOS FZP™ design technique. For 5V applications, Philips also offers the high speed PZ5128 CPLD that offers these features in a full 5V implementation.

The Philips FZP™ CPLDs introduce the new patent-pending XPLA™ (eXtended Programmable Logic Array) architecture. The XPLA™ architecture combines the best features of both PLA and PAL™ type structures to deliver high speed and flexible logic allocation that results in superior ability to make design changes with fixed pinouts. The XPLA™ structure in each logic block provides a fast 12ns PAL™ path with 5 dedicated product terms per output. This PAL™ path is joined by an additional PLA structure that deploys a pool of 32 product terms to a fully programmable OR array that can allocate the PLA product terms to any output in the logic block. This combination allows logic to be allocated efficiently throughout the logic block and supports as many as 37 product terms on an output. The speed with which logic is allocated from the PLA array to an output is only 2.5ns, regardless of the number of PLA product terms used, which results in worst case t_{PD}'s of only 14.5ns from any pin to any other pin. In addition, logic that is common to multiple outputs can be placed on a single PLA product term and shared across multiple outputs via the OR array, effectively increasing design density.

The PZ3128 CPLDs are supported by industry standard CAE tools (Cadence, Mentor, Synopsys, Synario, Viewlogic, MINC), using text (Abel, VHDL, Verilog) and/or schematic entry. Design verification uses industry standard simulators for functional and timing simulation. Development is supported on personal computer, Sparc, and HP platforms. Device fitting uses either MINC or Philips Semiconductors-developed tools.

The PZ3128 CPLD is electrically reprogrammable using industry standard device programmers from vendors such as Data I/O, BP Microsystems, SMS, and others. The PZ3128 also includes an industry-standard, IEEE 1149.1, JTAG interface through which in-system programming (ISP) and reprogramming of the device is supported.

PAL is a registered trademark of Advanced Micro Devices, Inc.

ORDERING INFORMATION

ORDER CODE	DESCRIPTION	DESCRIPTION	DRAWING NUMBER
PZ3128-S12A84	84-pin PLCC, 12ns t_{PD}	Commercial temp range, 3.3 volt power supply, ± 10%	SOT189-3
PZ3128-S15A84	84-pin PLCC, 15ns t_{PD}	Commercial temp range, 3.3 volt power supply, ± 10%	SOT189-3
PZ3128IS15A84	84-pin PLCC, 15ns t_{PD}	Industrial temp range, 3.3 volt power supply, ± 10%	SOT189-3
PZ3128-S12BB1	100-pin PQFP, 12ns t_{PD}	Commercial temp range, 3.3 volt power supply, ± 10%	SOT382-1
PZ3128-S15BB1	100-pin PQFP, 15ns t_{PD}	Commercial temp range, 3.3 volt power supply, ± 10%	SOT382-1
PZ3128IS15BB1	100-pin PQFP, 15ns t_{PD}	Industrial temp range, 3.3 volt power supply, ± 10%	SOT382-1
PZ3128-S12BP	100-pin TQFP, 12ns t_{PD}	Commercial temp range, 3.3 volt power supply, ± 10%	SOT386-1
PZ3128-S15BP	100-pin TQFP, 15ns t_{PD}	Commercial temp range, 3.3 volt power supply, ± 10%	SOT386-1
PZ3128IS15BP	100-pin TQFP, 15ns t_{PD}	Industrial temp range, 3.3 volt power supply, ± 10%	SOT386-1
PZ3128-S12BE	128-pin LQFP, 12ns t_{PD}	Commercial temp range, 3.3 volt power supply, ± 10%	SOT425-1
PZ3128-S15BE	128-pin LQFP, 15ns t_{PD}	Commercial temp range, 3.3 volt power supply, ± 10%	SOT425-1
PZ3128IS15BE	128-pin LQFP, 15ns t_{PD}	Industrial temp range, 3.3 volt power supply, ± 10%	SOT425-1
PZ3128-S12BB2	160-pin PQFP, 12ns t_{PD}	Commercial temp range, 3.3 volt power supply, ± 10%	SOT322-2
PZ3128-S15BB2	160-pin PQFP, 15ns t_{PD}	Commercial temp range, 3.3 volt power supply, ± 10%	SOT322-2
PZ3128IS15BB2	160-pin PQFP, 15ns t_{PD}	Industrial temp range, 3.3 volt power supply, ± 10%	SOT322-2

XPLA™ ARCHITECTURE

Figure 1 shows a high level block diagram of a 128 macrocell device implementing the XPLA™ architecture. The XPLA™ architecture consists of logic blocks that are interconnected by a Zero-power Interconnect Array (ZIA). The ZIA is a virtual crosspoint switch. Each logic block is essentially a 36V16 device with 36 inputs from the ZIA and 16 macrocells. Each logic block also provides 32 ZIA feedback paths from the macrocells and I/O pins.

From this point of view, this architecture looks like many other CPLD architectures. What makes the CoolRunner™ family unique is what is inside each logic block and the design technique used to implement these logic blocks. The contents of the logic block will be described next.

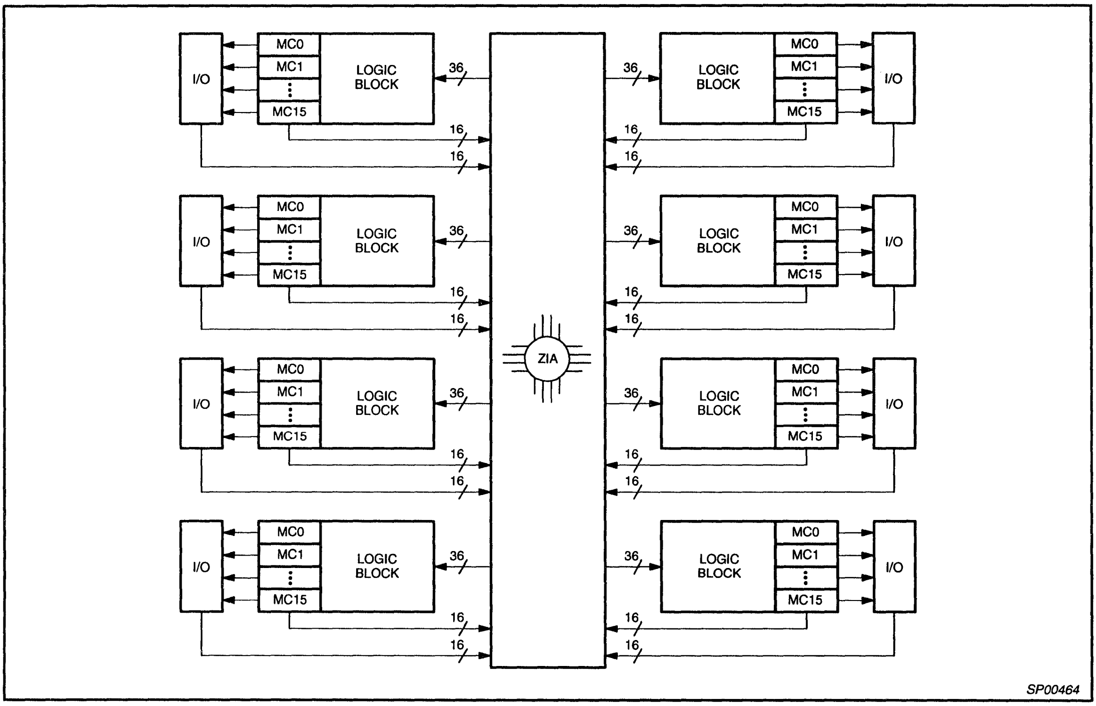


Figure 1. Philips XPLA CPLD Architecture

Logic Block Architecture

Figure 2 illustrates the logic block architecture. Each logic block contains control terms, a PAL array, a PLA array, and 16 macrocells. the 6 control terms can individually be configured as either SUM or PRODUCT terms, and are used to control the preset/reset and output enables of the 16 macrocells' flip-flops. The PAL array consists of a programmable AND array with a fixed OR array, while the PLA array consists of a programmable AND array with a programmable OR array. The PAL array provides a high speed path through the array, while the PLA array provides increased product term density.

Each macrocell has 5 dedicated product terms from the PAL array. The pin-to-pin t_{PD} of the PZ3128 device through the PAL array is 12ns. If a macrocell needs more than 5 product terms, it simply gets the additional product terms from the PLA array. The PLA array consists of 32 product terms, which are available for use by all 16 macrocells. The additional propagation delay incurred by a macrocell using 1 or all 32 PLA product terms is just 2.5ns. So the total pin-to-pin t_{PD} for the PZ3128 using 6 to 37 product terms is 14.5ns (12ns for the PAL + 2.5ns for the PLA).

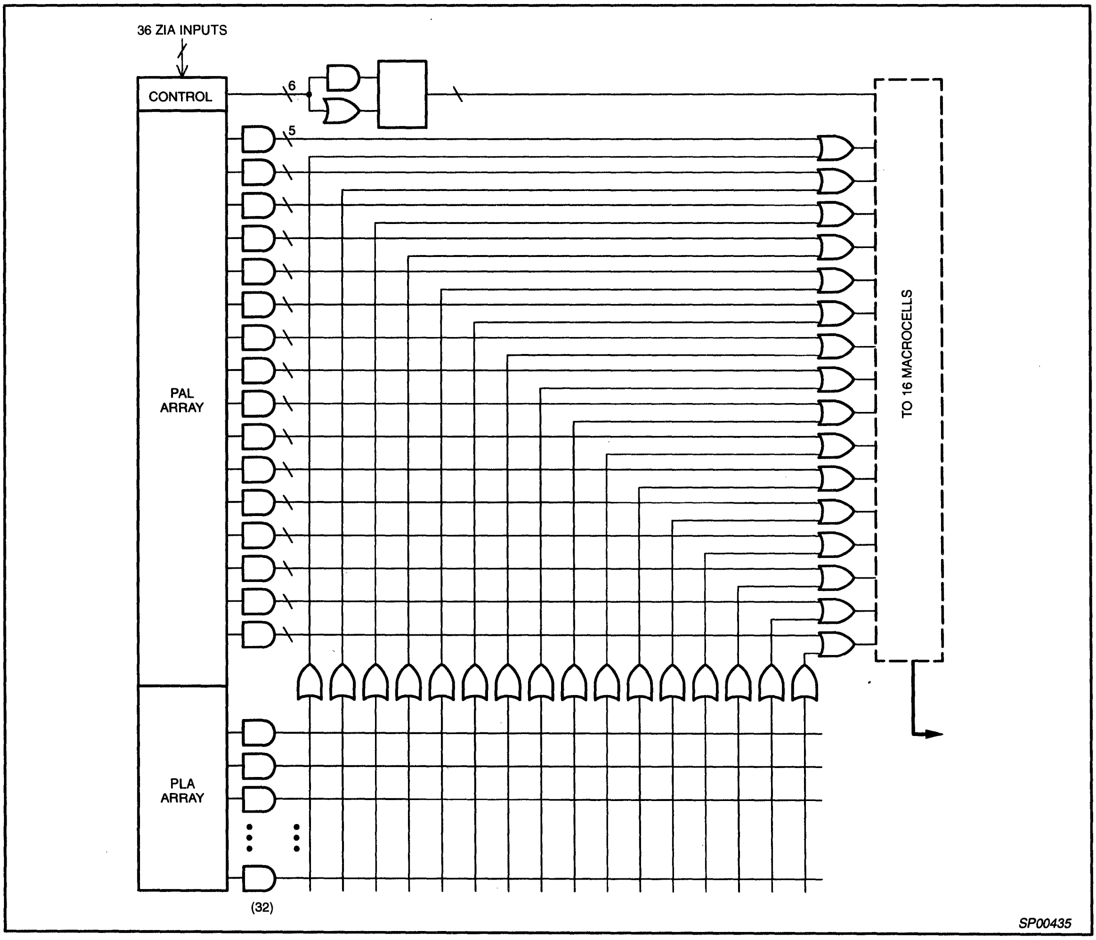


Figure 2. Philips Logic Block Architecture

Macrocell Architecture

Figure 3 shows the architecture of the macrocell used in the CoolRunner™ family. The macrocell consists of a flip-flop that can be configured as either a D or T type. A D-type flip-flop is generally more useful for implementing state machines and data buffering. A T-type flip-flop is generally more useful in implementing counters. All CoolRunner™ family members provide both synchronous and asynchronous clocking and provide the ability to clock off either the falling or rising edges of these clocks. These devices are designed such that the skew between the rising and falling edges of a clock are minimized for clocking integrity. There are 4 clocks available on the PZ3128 device. Clock 0 (CLK0) is designated as the "synchronous" clock and must be driven by an external source. Clock 1 (CLK1), Clock 2 (CLK2), and Clock 3 (CLK3) can either be used as a synchronous clock (driven by an external source) or as an asynchronous clock (driven by a macrocell equation).

Two of the control terms (CT0 and CT1) are used to control the Preset/Reset of the macrocell's flip-flop. The Preset/Reset feature for each macrocell can also be disabled. Note that the Power-on Reset leaves all macrocells in the "zero" state when power is properly applied. The other 4 control terms (CT2–CT5) can be used to control the Output Enable of the macrocell's output buffers. The reason there are as many control terms dedicated for the Output Enable of the macrocell is to insure that all CoolRunner™ devices are PCI compliant. The macrocell's output buffers can also be always enabled or disabled. All CoolRunner™ devices also provide a Global Tri-State ($\overline{GTS}$) pin, which, when pulled Low, will 3-State all the outputs of the device. This pin is provided to support "In-Circuit Testing" or "Bed-of-Nails Testing".

There are two feedback paths to the ZIA: one from the macrocell, and one from the I/O pin. The ZIA feedback path before the output buffer is the macrocell feedback path, while the ZIA feedback path after the output buffer is the I/O pin ZIA path. When the macrocell is used as an output, the output buffer is enabled, and the macrocell feedback path can be used to feedback the logic implemented in the macrocell. When the I/O pin is used as an input, the output buffer will be 3-Stated and the input signal will be fed into the ZIA via the I/O feedback path, and the logic implemented in the buried macrocell can be fed back to the ZIA via the macrocell feedback path. It should be noted that unused inputs or I/Os should be properly terminated.

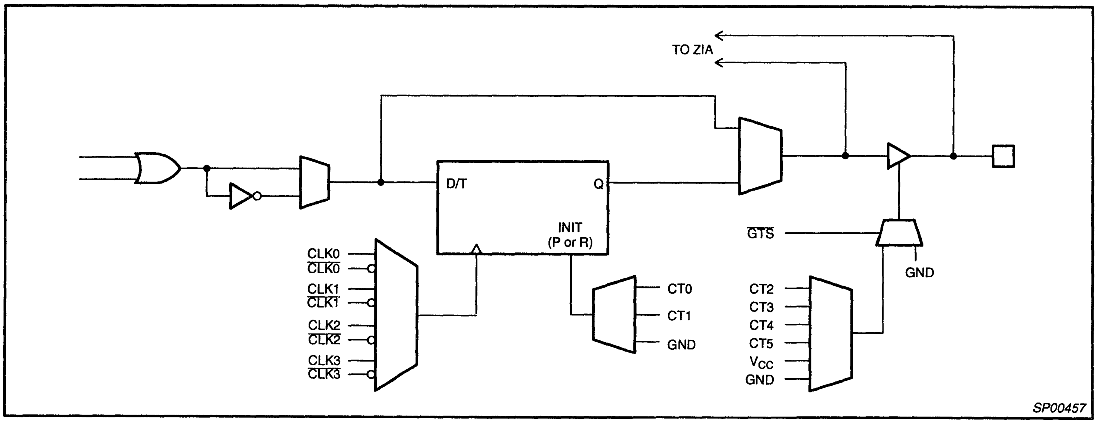


Figure 3. PZ3128 Macrocell Architecture

Simple Timing Model

Figure 4 shows the CoolRunner™ Timing Model. The CoolRunner™ timing model looks very much like a 22V10 timing model in that there are three main timing parameters, including t_{PD}, t_{SU}, and t_{CO}. In other competing architectures, the user may be able to fit the design into the CPLD, but is not sure whether system timing requirements can be met until after the design has been fit into the device. This is because the timing models of competing architectures are very complex and include such things as timing dependencies on the number of parallel expanders borrowed, sharable expanders, varying number of X and Y routing channels used, etc. In the XPLA™ architecture, the user knows up front whether the design will meet system timing requirements. This is due to the simplicity of the timing model.

TotalCMOS™ Design Technique for Fast Zero Power

Philips is the first to offer a TotalCMOS™ CPLD, both in process technology and design technique. Philips employs a cascade of CMOS gates to implement its Sum of Products instead of the traditional sense amp approach. This CMOS gate implementation allows Philips to offer CPLDs which are both high performance and low power, breaking the paradigm that to have low power, you must have low performance. Refer to Figure 5 and Table 2 showing the I_{DD} vs. Frequency of our PZ3128 TotalCMOS™ CPLD.

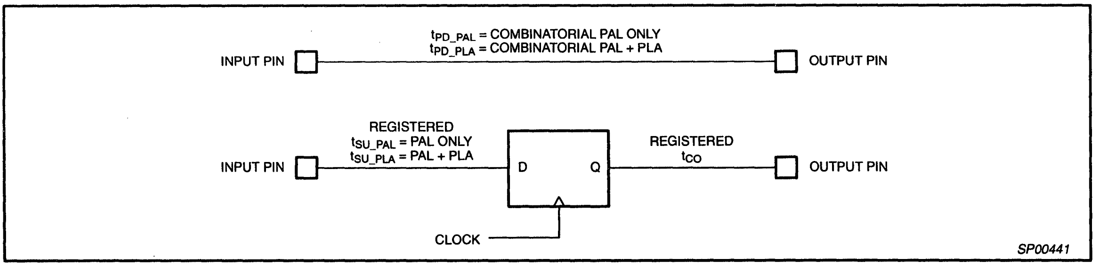


Figure 4. CoolRunner™ Timing Model

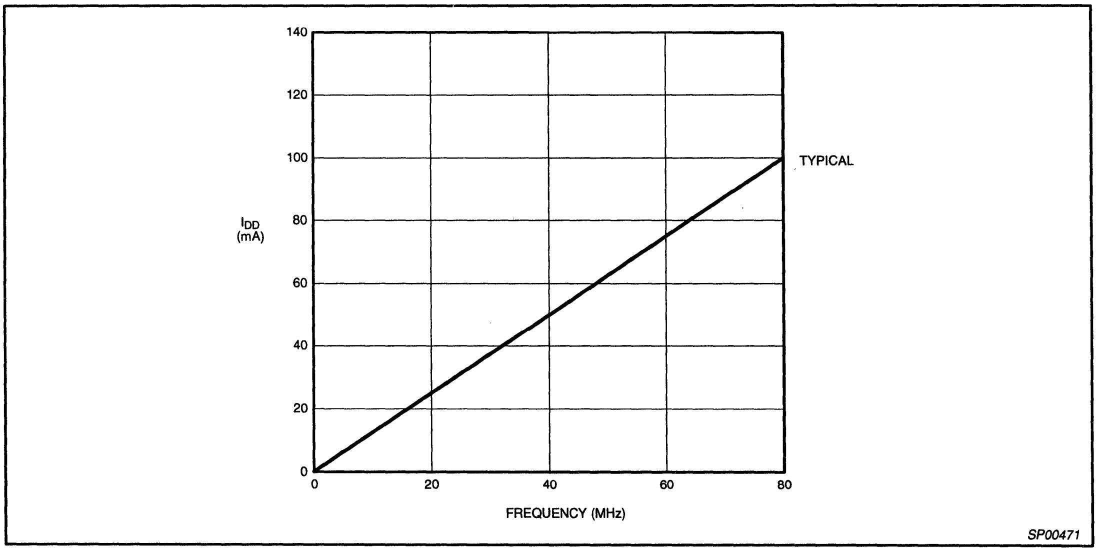


Figure 5. I_{DD} vs. Frequency @ V_{DD} = 3.3V, 25°C

Table 2. I_{DD} vs. Frequency

V_{DD} = 3.3V

FREQUENCY (MHz)	0	20	40	60	80
Typical I_{DD} (mA)	0.10	25	50	75	100

JTAG Testing Capability

JTAG is the commonly-used acronym for the Boundary Scan Test (BST) feature defined for integrated circuits by IEEE Standard 1149.1. This standard defines input/output pins, logic control functions, and commands which facilitate both board and device level testing without the use of specialized test equipment. BST provides the ability to test the external connections of a device, test the internal logic of the device, and capture data from the device during normal operation. BST provides a number of benefits in each of the following areas:

- Testability
  - Allows testing of an unlimited number of interconnects on the printed circuit board
  - Testability is designed in at the component level
  - Enables desired signal levels to be set at specific pins (Preload)
  - Data from pin or core logic signals can be examined during normal operation
- Reliability
  - Eliminates physical contacts common to existing test fixtures (e.g., "bed-of-nails")
  - Degradation of test equipment is no longer a concern
  - Facilitates the handling of smaller, surface-mount components
  - Allows for testing when components exist on both sides of the printed circuit board
- Cost
  - Reduces/eliminates the need for expensive test equipment
  - Reduces test preparation time
  - Reduces spare board inventories

The Philips PZ3128's JTAG interface includes a TAP Port and a TAP Controller, both of which are defined by the IEEE 1149.1 JTAG Specification. As implemented in the Philips PZ3128, the TAP Port includes four of the five pins (refer to Table 3) described in the JTAG specification: TCK, TMS, TDI, and TDO. The fifth signal defined by the JTAG specification is TRST* (Test Reset). TRST* is considered an optional signal, since it is not actually required to perform BST or ISP. The Philips PZ3128 saves an I/O pin for general purpose use by not implementing the optional TRST* signal in the JTAG interface. Instead, the Philips PZ3128 supports the test reset functionality through the use of its power up reset circuit, which is included in all Philips CPLDs. The pins associated with the power up reset circuit should connect to an external pull-up resistor to keep the JTAG signals from floating when they are not being used.

In the Philips PZ3128, the four mandatory JTAG pins each require a unique, dedicated pin on the device. However, if JTAG and ISP are not desired in the end-application, these pins may instead be used as additional general I/O pins. The decision as to whether these pins are used for JTAG/ISP or as general I/O is made when the JEDEC file is generated. If the use of JTAG/ISP is selected, the dedicated pins are not available for general purpose use. However, unlike competing CPLD's, the Philips PZ3128 does allow the macrocell logic associated with these dedicated pins to be used as buried logic even when JTAG/ISP is selected. Table 4 defines the dedicated pins used by the four mandatory JTAG signals for each of the PZ3128 package types.

The JTAG specifications defines two sets of commands to support boundary-scan testing: high-level commands and low-level commands. High-level commands are executed via board test software on an a user test station such as automated test equipment, a PC, or an engineering workstation (EWS). Each high-level command comprises a sequence of low level commands. These low-level commands are executed within the component under test, and therefore must be implemented as part of the TAP Controller design. The set of low-level boundary-scan commands implemented in the Philips PZ3128 is defined in Table 5. By supporting this set of low-level commands, the PZ3128 allows execution of all high-level boundary-scan commands.

Table 3. JTAG Pin Description

PIN	NAME	DESCRIPTION
TCK	Test Clock Output	Clock pin to shift the serial data and instructions in and out of the TDI and TDO pins, respectively. TCK is also used to clock the TAP Controller state machine.
TMS	Test Mode Select	Serial input pin selects the JTAG instruction mode. TMS should be driven high during user mode operation.
TDI	Test Data Input	Serial input pin for instructions and test data. Data is shifted in on the rising edge of TCK.
TDO	Test Data Output	Serial output pin for instructions and test data. Data is shifted out on the falling edge of TCK. The signal is tri-stated if data is not being shifted out of the device.

Table 4. PZ3128 JTAG Pinout by Package Type

DEVICE	(PIN NUMBER / MACROCELL #)			
	TCK	TMS	TDI	TDO
PZ3128				
84-pin PLCC	62 / 96 (F15)	23 / 48 (C15)	14 / 32 (B15)	71 / 112 (G15)
100-pin PQFP	64 / 96 (F15)	17 / 48 (C15)	6 / 32 (B15)	75 / 112 (G15)
100-pin TQFP	62 / 96 (F15)	15 / 48 (C15)	4 / 32 (B15)	73 / 112 (G15)
128-pin LQFP	82 / 96 (F15)	21 / 48 (C15)	8 / 32 (B15)	95 / 112 (G15)
160-pin PQFP	99 / 96 (F15)	22 / 48 (C15)	9 / 32 (B15)	112/ 112 (G15)

Table 5. PZ3128 Low-Level JTAG Boundary-Scan Commands

INSTRUCTION **(Instruction Code)** ***Register Used***	**DESCRIPTION**
Sample/Preload (0010) *Boundary–Scan Register*	The mandatory SAMPLE/PRELOAD instruction allows a snapshot of the normal operation of the component to be taken and examined. It also allows data values to be loaded onto the latched parallel outputs of the Boundary-Scan Shift-Register prior to selection of the other boundary-scan test instructions.
Extest (0000) *Boundary-Scan Register*	The mandatory EXTEST instruction allows testing of off-chip circuitry and board level interconnections. Data would typically be loaded onto the latched parallel outputs of Boundary-Scan Shift-Register using the Sample/Preload instruction prior to selection of the EXTEST instruction.
Bypass (1111) *Bypass Register*	Places the 1 bit bypass register between the TDI and TDO pins, which allows the BST data to pass synchronously through the selected device to adjacent devices during normal device operation. The Bypass instruction can be entered by holding TDI at a constant high value and completing an Instruction-Scan cycle.
Idcode (0001) *Boundary-Scan Register*	Selects the IDCODE register and places it between TDI and TDO, allowing the IDCODE to be serially shifted out of TDO. The IDCODE instruction permits blind interrogation of the components assembled onto a printed circuit board. Thus, in circumstances where the component population may vary, it is possible to determine what components exist in a product.
HighZ (0101) *Bypass Register*	The HIGHZ instruction places the component in a state in which all of its system logic outputs are placed in an inactive drive state (e.g., high impedance). In this state, an in-circuit test system may drive signals onto the connections normally driven by a component output without incurring the risk of damage to the component. The HighZ instruction also forces the Bypass Register between TDI and TDO.

3.3-Volt, In-System Programming (ISP)

ISP is the ability to reconfigure the logic and functionality of a device, printed circuit board, or complete electronic system before, during, and after its manufacture and shipment to the end customer. ISP provides substantial benefits in each of the following areas:

- Design
  - Faster time-to-market
  - Debug partitioning and simplified prototyping
  - Printed circuit board reconfiguration during debug
  - Better device and board level testing
- Manufacturing
  - Multi-Functional hardware
  - Reconfiguarability for Test
  - Eliminates handling of “fine lead-pitch” components for programming
  - Reduced Inventory and manufacturing costs
  - Improved quality and reliability
- Field Support
  - Easy remote upgrades and repair
  - Support for field configuration, re-configuration, and customization

The Philips PZ3128 allows for 3.3-Volt, in-system programming/reprogramming of its EEPROM cells via its JTAG interface. An on-chip charge pump eliminates the need for externally-provided supervoltages, so that the PZ3128 may be easily programmed on the circuit board using only the 3.3-volt supply required by the device for normal operation. A set of low-level ISP basic commands implemented in the PZ3128 enable this feature. The ISP commands implemented in the Philips PZ3128 are specified in Table 6. Please note that an ENABLE command must precede all ISP commands **unless** an ENABLE command has already been given for a preceding ISP command **and** the device has not gone through a Test-Logic/Rest TAP Controller State.

Table 6. Low Level ISP Commands

INSTRUCTION ***(Register Used)***	**INSTRUCTION CODE**	**DESCRIPTION**
Enable *(ISP Shift Register)*	1001	Enables the Erase, Program, and Verify commands. Using the ENABLE instruction before the Erase, Program, and Verify instructions allows the user to specify the outputs the device using the JTAG Boundary–Scan SAMPLE/PRELOAD command.
Erase *(ISP Shift Register)*	1010	Erases the entire EEPROM array. The outputs during this operation can be defined by user by using the JTAG SAMPLE/PRELOAD command.
Program *(ISP Shift Register)*	1011	Programs the data in the ISP Shift Register into the addressed EEPROM row. The outputs during this operation can be defined by user by using the JTAG SAMPLE/PRELOAD command.
Verify *(ISP Shift Register)*	1100	Transfers the data from the addressed row to the ISP Shift Register. The data can then be shifted out and compared with the JEDEC file. The outputs during this operation can be defined by user by using the JTAG SAMPLE/PRELOAD command.

JTAG and ISP Interfacing

A number of industry-established methods exist for JTAG/ISP interfacing with CPLD's and other integrated circuits. The Philips PZ3128 supports the following methods:

- PC Parallel Port
- Workstation or PC Serial Port
- Embedded Processor
- Automated Test Equipment
- Third party Programmers
- High-End JTAG and ISP Tools

A Boundary-Scan Description Language (BSDL) description of the PZ3128 is also available from Philips for use in test program development. For more details on JTAG and ISP for the PZ3128, refer to the related application note: *JTAG and ISP in Philips CPLDs.*

Table 7. Programming Specifications

SYMBOL	PARAMETER	MIN.	MAX.	UNIT
DC Parameters				
V_{CCP}	V_{CC} supply program/verify	3.0	3.6	V
I_{CCP}	I_{CC} limit program/verify		200	mA
V_{IH}	Input voltage (High)	2.0		V
V_{IL}	Input voltage (Low)		0.8	V
V_{SOL}	Output voltage (Low)		0.5	V
V_{SOH}	Output voltage (High)	2.4		V
TDO_I_{OL}	Output current (Low)	8		mA
TDO_I_{OH}	Output current (High)	–8		mA
AC Parameters				
f_{MAX}	CLK maximum frequency	10		MHz
PWE	Pulse width erase	100		ms
PWP	Pulse width program	10		ms
PWV	Pulse width verify	10		µs
INIT	Initialization time	100		µs
TMS_SU	TMS setup time before TCK ↑	10		ns
TDI_SU	TDI setup time before TCK ↑	10		ns
TMS_H	TMS hold time after TCK ↑	25		ns
TDI_H	TDI hold time after TCK ↑	25		ns
TDO_CO	TDO valid after TCK ↓		40	ns

ABSOLUTE MAXIMUM RATINGS[1]

SYMBOL	PARAMETER	MIN.	MAX.	UNIT
V_{DD}	Supply voltage	–0.5	7.0	V
V_I	Input voltage	–1.2	V_{DD}+0.5	V
V_{OUT}	Output voltage	–0.5	V_{DD}+0.5	V
I_{IN}	Input current	–30	30	mA
I_{OUT}	Output current	–100	100	mA
T_J	Maximum junction temperature	–40	150	°C
T_{str}	Storage temperature	–65	150	°C

NOTE:

1. Stresses above those listed may cause malfunction or permanent damage to the device. This is a stress rating only. Functional operation at these or any other condition above those indicated in the operational and programming specification is not implied.

OPERATING RANGE

PRODUCT GRADE	TEMPERATURE	VOLTAGE
Commercial	0 to +70°C	3.3 ±10% V
Industrial	–40 to +85°C	3.3 ±10% V

DC ELECTRICAL CHARACTERISTICS FOR COMMERCIAL GRADE DEVICES

Commercial: 0°C ≤ T_{amb} ≤ +70°C; 3.0V ≤ V_{DD} ≤ 3.6V

SYMBOL	PARAMETER	TEST CONDITIONS	MIN.	MAX.	UNIT
V_{IL}	Input voltage low	V_{DD} = 3.0V		0.8	V
V_{IH}	Input voltage high	V_{DD} = 3.6V	2.0		V
V_I	Input clamp voltage	V_{DD} = 3.0V, I_{IN} = –18mA		–1.2	V
V_{OL}	Output voltage low	V_{DD} = 3.0V, I_{OL} = 8mA		0.5	V
V_{OH}	Output voltage high	V_{DD} = 3.0V, I_{OH} = –8mA	2.4		V
I_I	Input leakage current	V_{IN} = 0 to V_{DD}	–10	10	µA
I_{OZ}	3-Stated output leakage current	V_{IN} = 0 to V_{DD}	–10	10	µA
I_{DDQ}	Standby current	V_{DD} = 3.6V, T_{amb} = 0°C		60	µA
I_{DDD}[1]	Dynamic current	V_{DD} = 3.6V, T_{amb} = 0°C @ 1MHz		2	mA
		V_{DD} = 3.6V, T_{amb} = 0°C @ 50MHz		75	mA
I_{OS}	Short circuit output current[2]	1 pin at a time for no longer than 1 second	–50	–100	mA
C_{IN}	Input pin capacitance[2]	T_{amb} = 25°C, f = 1MHz		8	pF
C_{CLK}	Clock input capacitance[2]	T_{amb} = 25°C, f = 1MHz	5	12	pF
$C_{I/O}$	I/O pin capacitance[2]	T_{amb} = 25°C, f = 1MHz		10	pF

NOTES:

1. This parameter measured with a 16-bit, loadable up/down counter loaded into every logic block, with all outputs enabled and unloaded. Inputs are tied to V_{DD} or ground. This parameter guaranteed by design and characterization, not testing.
2. Typical values, not tested.

AC ELECTRICAL CHARACTERISTICS[1] FOR COMMERCIAL GRADE DEVICES

Commercial: 0°C ≤ T_{amb} ≤ +70°C; 3.0V ≤ V_{DD} ≤ 3.6V

SYMBOL	PARAMETER	–12		–15		UNIT
		MIN.	MAX.	MIN.	MAX.	
t_{PD_PAL}	Propagation delay time, input (or feedback node) to output through PAL	2	12	2	15	ns
t_{PD_PLA}	Propagation delay time, input (or feedback node) to output through PAL & PLA	3	14.5	3	17.5	ns
t_{CO}	Clock to out delay time	2	8	2	9	ns
t_{SU_PAL}	Setup time (from input or feedback node) through PAL	7		8		ns
t_{SU_PLA}	Setup time (from input or feedback node) through PAL + PLA	9.5		10.5		ns
t_H	Hold time		0		0	ns
t_{CH}	Clock High time	4		4		ns
t_{CL}	Clock Low time	4		4		ns
t_R	Input Rise time		20		20	ns
t_F	Input Fall time		20		20	ns
f_{MAX1}	Maximum FF toggle rate[2] 1/(t_{CH} + t_{CL})	125		125		MHz
f_{MAX2}	Maximum internal frequency[2] 1/(t_{SUPAL} + t_{CF})	74		65		MHz
f_{MAX3}	Maximum external frequency[2] 1/(t_{SUPAL} + t_{CO})	66		59		MHz
t_{BUF}	Output buffer delay time		1.5		1.5	ns
t_{PDF_PAL}	Input (or feedback node) to internal feedback node delay time through PAL	2	10.5	2	13.5	ns
t_{PDF_PLA}	Input (or feedback node) to internal feedback node delay time through PAL+PLA	3	13	3	16	ns
t_{CF}	Clock to internal feedback node delay time		6.5		7.5	ns
t_{INIT}	Delay from valid V_{DD} to valid reset		50		50	µs
t_{ER}	Input to output disable[3]		14		17	ns
t_{EA}	Input to output valid		14		17	ns
t_{RP}	Input to register preset		16		19	ns
t_{RR}	Input to register reset		16		19	ns

NOTES:

1. Specifications measured with one output switching. See Figure 6 and Table 8 for derating.
2. This parameter guaranteed by design and characterization, not by test.
3. Output C_L = 5pF.

DC ELECTRICAL CHARACTERISTICS FOR INDUSTRIAL GRADE DEVICES

Industrial: −40°C ≤ T_{amb} ≤ +85°C; 3.0V ≤ V_{DD} ≤ 3.6V

SYMBOL	PARAMETER	TEST CONDITIONS	MIN.	MAX.	UNIT
V_{IL}	Input voltage low	V_{DD} = 3.0V		0.8	V
V_{IH}	Input voltage high	V_{DD} = 3.6V	2.0		V
V_I	Input clamp voltage	V_{DD} = 3.0V, I_{IN} = −18mA		−1.2	V
V_{OL}	Output voltage low	V_{DD} = 3.0V, I_{OL} = 8mA		0.5	V
V_{OH}	Output voltage high	V_{DD} = 3.0V, I_{OH} = −8mA	2.4		V
I_I	Input leakage current	V_{IN} = 0 to V_{DD}	−10	10	µA
I_{OZ}	3-Stated output leakage current	V_{IN} = 0 to V_{DD}	−10	10	µA
I_{DDQ}	Standby current	V_{DD} = 3.6V, T_{amb} = −40°C		75	µA
I_{DDD}[1]	Dynamic current	V_{DD} = 3.6V, T_{amb} = −40°C @ 1MHz		2	mA
		V_{DD} = 3.6V, T_{amb} = −40°C @ 50MHz		75	mA
I_{OS}	Short circuit output current[2]	1 pin at a time for no longer than 1 second	−50	−130	mA
C_{IN}	Input pin capacitance[2]	T_{amb} = 25°C, f = 1MHz		8	pF
C_{CLK}	Clock input capacitance[2]	T_{amb} = 25°C, f = 1MHz	5	12	pF
$C_{I/O}$	I/O pin capacitance[2]	T_{amb} = 25°C, f = 1MHz		10	pF

NOTES:

1. This parameter measured with a 16–bit, loadable up/down counter loaded into every logic block, with all outputs enabled and unloaded. Inputs are tied to V_{DD} or ground. This parameter guaranteed by design and characterization, not testing.
2. Typical values, not tested.

AC ELECTRICAL CHARACTERISTICS[1] FOR INDUSTRIAL GRADE DEVICES

Industrial: −40°C ≤ T_{amb} ≤ +85°C; 3.0V ≤ V_{DD} ≤ 3.6V

SYMBOL	PARAMETER	I15		UNIT
		MIN.	MAX.	
t_{PD_PAL}	Propagation delay time, input (or feedback node) to output through PAL	2	15	ns
t_{PD_PLA}	Propagation delay time, input (or feedback node) to output through PAL & PLA	3	17.5	ns
t_{CO}	Clock to out delay time	2	9	ns
t_{SU_PAL}	Setup time (from input or feedback node) through PAL	8		ns
t_{SU_PLA}	Setup time (from input or feedback node) through PAL + PLA	10.5		ns
t_H	Hold time		0	ns
t_{CH}	Clock High time	4		ns
t_{CL}	Clock Low time	4		ns
t_R	Input Rise time		20	ns
t_F	Input Fall time		20	ns
f_{MAX1}	Maximum FF toggle rate[2] $1/(t_{CH} + t_{CL})$	125		MHz
f_{MAX2}	Maximum internal frequency[2] $1/(t_{SUPAL} + t_{CF})$	65		MHz
f_{MAX3}	Maximum external frequency[2] $1/(t_{SUPAL} + t_{CO})$	59		MHz
t_{BUF}	Output buffer delay time		1.5	ns
t_{PDF_PAL}	Input (or feedback node) to internal feedback node delay time through PAL	2	13.5	ns
t_{PDF_PLA}	Input (or feedback node) to internal feedback node delay time through PAL+PLA	3	16	ns
t_{CF}	Clock to internal feedback node delay time		7.5	ns
t_{INIT}	Delay from valid V_{DD} to valid reset		50	µs
t_{ER}	Input to output disable[3]		15.5	ns
t_{EA}	Input to output valid		15.5	ns
t_{RP}	Input to register preset		17	ns
t_{RR}	Input to register reset		17	ns

NOTES:

1. Specifications measured with one output switching. See Figure 6 and Table 8 for derating.
2. This parameter guaranteed by design and characterization, not by test.
3. Output C_L = 5pF.

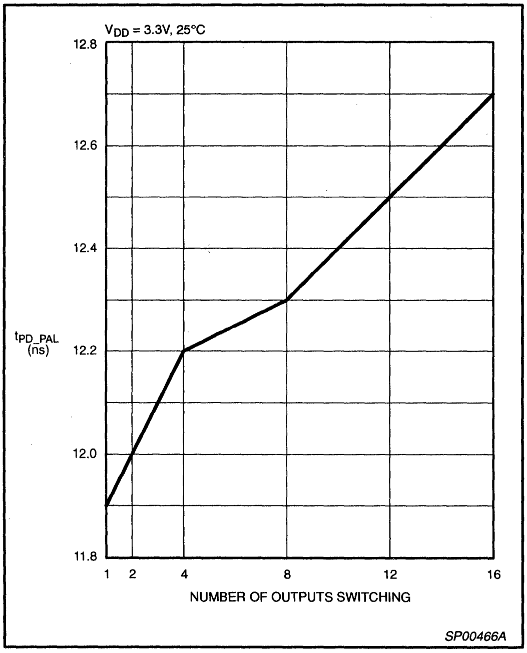


Figure 6. t_{PD_PAL} vs. Outputs Switching

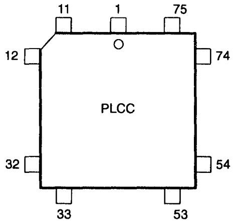
Table 8. t_{PD_PAL} vs. Number of Outputs Switching

V_{DD} = 3.3V

NUMBER OF OUTPUTS	1	2	4	8	12	16
Typical (ns)	11.9	12	12.2	12.3	12.5	12.7

PIN DESCRIPTIONS

84-Pin Plastic Leaded Chip Carrier

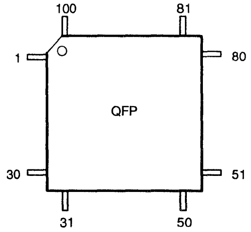


Pin	Function	Pin	Function	Pin	Function
1	IN1	29	I/O-C5	57	I/O-F7
2	IN3	30	I/O-C4	58	I/O-F10
3	V_{DD}	31	I/O-C2	59	GND
4	I/O-A15/CLK3	32	GND	60	I/O-F12
5	I/O-A13	33	I/O-D15	61	I/O-F13
6	I/O-A12	34	I/O-D12	62	I/O-F15 (TCK)
7	GND	35	I/O-D10	63	I/O-G0
8	I/O-A10	36	I/O-D8	64	I/O-G2
9	I/O-A7	37	I/O-D7	65	I/O-G4
10	I/O-A5	38	V_{DD}	66	V_{DD}
11	I/O-A4	39	I/O-D4	67	I/O-G7
12	I/O-A2	40	I/O-D2	68	I/O-G8
13	V_{DD}	41	I/O-D0/CLK2	69	I/O-G10
14	I/O-B15 (TDI)	42	GND	70	I/O-G12
15	I/O-B12	43	V_{DD}	71	I/O-G15 (TDO)
16	I/O-B10	44	I/O-E0/CLK1	72	GND
17	I/O-B8	45	I/O-E2	73	I/O-H2
18	I/O-B7	46	I/O-E4	74	I/O-H4
19	GND	47	GND	75	I/O-H5
20	I/O-B4	48	I/O-E7	76	I/O-H7
21	I/O-B2	49	I/O-E8	77	I/O-H10
22	I/O-B0	50	I/O-E10	78	V_{DD}
23	I/O-C15 (TMS)*	51	I/O-E12	79	I/O-H12
24	I/O-C13	52	I/O-E15	80	I/O-H13
25	I/O-C12	53	V_{DD}	81	I/O-H15
26	V_{DD}	54	I/O-F2	82	GND
27	I/O-C10	55	I/O-F4	83	IN0/CLK0
28	I/O-C7	56	I/O-F5	84	IN2-gtsn

* THE TEST MODE SELECT (TMS) FUNCTION IS INACTIVE ON NON-ISR ARCHITECTURES.

SP00467

100-Pin Plastic Quad Flat Package

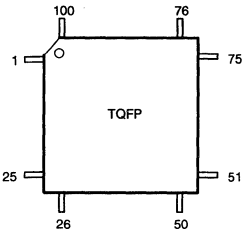


Pin	Function	Pin	Function	Pin	Function
1	I/O-A5	35	I/O-D5	69	I/O-G5
2	I/O-A4	36	V_{DD}	70	I/O-G7
3	I/O-A2	37	I/O-D4	71	I/O-G8
4	I/O-A0	38	I/O-D2	72	I/O-G10
5	V_{DD}	39	I/O-B0/CLK2	73	I/O-G12
6	I/O-B15 (TDI)	40	GND	74	I/O-G13
7	I/O-B13	41	V_{DD}	75	I/O-G15 (TDO)
8	I/O-B12	42	I/O-E0/CLK1	76	GND
9	I/O-B10	43	I/O-E2	77	I/O-H0
10	I/O-B8	44	I/O-E4	78	I/O-H2
11	I/O-B7	45	GND	79	I/O-H4
12	I/O-B5	46	I/O-E5	80	I/O-H5
13	GND	47	I/O-E7	81	I/O-H7
14	I/O-B4	48	I/O-E8	82	I/O-H8
15	I/O-B2	49	I/O-E10	83	I/O-H10
16	I/O-B0	50	I/O-E12	84	V_{DD}
17	I/O-C15 (TMS)*	51	I/O-E13	85	I/O-H12
18	I/O-C13	52	I/O-E15	86	I/O-H13
19	I/O-C12	53	V_{DD}	87	I/O-H15
20	V_{DD}	54	I/O-F0	88	GND
21	I/O-C10	55	I/O-F2	89	IN0/CLK0
22	I/O-C8	56	I/O-F4	90	IN2-gtsn
23	I/O-C7	57	I/O-F5	91	IN1
24	I/O-C5	58	I/O-F7	92	IN3
25	I/O-B9	59	I/O-F8	93	V_{DD}
26	I/O-C2	60	I/O-F10	94	I/O-A15/CLK3
27	I/O-C0	61	GND	95	I/O-A13
28	GND	62	I/O-F12	96	I/O-A12
29	I/O-D15	63	I/O-F13	97	GND
30	I/O-D13	64	I/O-F15 (TCK)	98	I/O-A10
31	I/O-D12	65	I/O-G0	99	I/O-A8
32	I/O-D10	66	I/O-G2	100	I/O-A7
33	I/O-D8	67	I/O-G4		
34	I/O-D7	68	V_{DD}		

* THE TEST MODE SELECT (TMS) FUNCTION IS INACTIVE ON NON-ISR ARCHITECTURES.

SP00468

100-Pin Thin Quad Flat Package

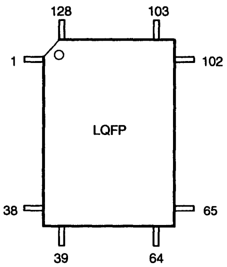


Pin	Function	Pin	Function	Pin	Function
1	I/O-A2	35	I/O-D4	69	I/O-G8
2	I/O-A0	36	I/O-D2	70	I/O-G10
3	V_{DD}	37	I/O-D0/CLK2	71	I/O-G12
4	I/O-B15 (TDI)	38	GND	72	I/O-G13
5	I/O-B13	39	V_{DD}	73	I/O-G15 (TDO)
6	I/O-B12	40	I/O-E0/CLK1	74	GND
7	I/O-B10	41	I/O-E2	75	I/O-H0
8	I/O-B8	42	I/O-E4	76	I/O-H2
9	I/O-B7	43	GND	77	I/O-H4
10	I/O-B5	44	I/O-E5	78	I/O-H5
11	GND	45	I/O-E7	79	I/O-H7
12	I/O-B4	46	I/O-E8	80	I/O-H8
13	I/O-B2	47	I/O-E10	81	I/O-H10
14	I/O-B0	48	I/O-E12	82	V_{DD}
15	I/O-C15 (TMS)*	49	I/O-E13	83	I/O-H12
16	I/O-C13	50	I/O-E15	84	I/O-H13
17	I/O-C12	51	V_{DD}	85	I/O-H15
18	V_{DD}	52	I/O-F0	86	GND
19	I/O-C10	53	I/O-F2	87	IN0/CLK0
20	I/O-C8	54	I/O-F4	88	IN2-gtsn
21	I/O-C7	55	I/O-F5	89	IN1
22	I/O-C5	56	I/O-F7	90	IN3
23	I/O-C4	57	I/O-F8	91	V_{DD}
24	I/O-C2	58	I/O-F10	92	I/O-A15/CLK3
25	I/O-C0	59	GND	93	I/O-A13
26	GND	60	I/O-F12	94	I/O-A12
27	I/O-D15	61	I/O-F13	95	GND
28	I/O-D13	62	I/O-F15 (TCK)	96	I/O-A10
29	I/O-D12	63	I/O-G0	97	I/O-A8
30	I/O-D10	64	I/O-G2	98	I/O-A7
31	I/O-D8	65	I/O-G4	99	I/O-A5
32	I/O-D7	66	V_{DD}	100	I/O-A4
33	I/O-D5	67	I/O-G5		
34	V_{DD}	68	I/O-G7		

* THE TEST MODE SELECT (TMS) FUNCTION IS INACTIVE ON NON-ISR ARCHITECTURES.

SP00485

128-Pin Low Profile Quad Flat Package

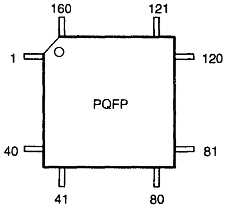


Pin	Function	Pin	Function	Pin	Function
1	I/O-A3	44	I/O-D7	87	V_{DD}
2	I/O-A2	45	I/O-D5	88	I/O-G5
3	I/O-A0	46	V_{DD}	89	I/O-G7
4	NC	47	I/O-D4	90	I/O-G8
5	NC	48	I/O-D3	91	I/O-G10
6	NC	49	I/O-D2	92	I/O-G11
7	V_{DD}	50	I/O-D0/CLK2	93	I/O-G12
8	I/O-B15 (TDI)	51	GND	94	I/O-G13
9	I/O-B13	52	V_{DD}	95	I/O-G15 (TDO)
10	I/O-B12	53	I/O-E0/CLK1	96	GND
11	I/O-B11	54	I/O-E2	97	NC
12	I/O-B10	55	I/O-E3	98	NC
13	I/O-B8	56	I/O-E4	99	NC
14	I/O-B7	57	GND	100	I/O-H0
15	I/O-B5	58	I/O-E5	101	I/O-H2
16	GND	59	I/O-E7	102	I/O-H3
17	I/O-B4	60	I/O-E8	103	I/O-H4
18	I/O-B3	61	I/O-E10	104	I/O-H5
19	I/O-B2	62	I/O-E11	105	I/O-H7
20	I/O-B0	63	I/O-E12	106	I/O-H8
21	I/O-C15 (TMS)*	64	I/O-E13	107	I/O-H10
22	I/O-C13	65	I/O-E15	108	V_{DD}
23	I/O-C12	66	V_{DD}	109	I/O-H11
24	I/O-C11	67	I/O-F0	110	I/O-H12
25	V_{DD}	68	NC	111	I/O-H13
26	I/O-C10	69	NC	112	I/O-H15
27	I/O-C8	70	NC	113	GND
28	I/O-C7	71	I/O-F2	114	IN0/CLK0
29	I/O-C5	72	I/O-F3	115	IN2-gtsn
30	I/O-C4	73	I/O-F4	116	IN1
31	I/O-C3	74	I/O-F5	117	IN3
32	I/O-C2	75	I/O-F7	118	V_{DD}
33	NC	76	I/O-F8	119	I/O-A15/CLK3
34	NC	77	I/O-F10	120	I/O-A13
35	NC	78	GND	121	I/O-A12
36	I/O-C0	79	I/O-F11	122	I/O-A11
37	GND	80	I/O-F12	123	GND
38	I/O-D15	81	I/O-F13	124	I/O-A10
39	I/O-D13	82	I/O-F15(TCK)	125	I/O-A8
40	I/O-D12	83	I/O-G0	126	I/O-A7
41	I/O-D11	84	I/O-G2	127	I/O-A5
42	I/O-D10	85	I/O-G3	128	I/O-A4
43	I/O-D8	86	I/O-G4		

* THE TEST MODE SELECT (TMS) FUNCTION IS INACTIVE ON NON-ISR ARCHITECTURES.

SP00469A

160-Pin Plastic Quad Flat Package



Pin	Function	Pin	Function	Pin	Function
1	NC	54	I/O-D5	107	I/O-G8
2	NC	55	V_{DD}	108	I/O-G10
3	NC	56	I/O-D4	109	I/O-G11
4	NC	57	I/O-D3	110	I/O-G12
5	NC	58	I/O-D2	111	I/O-G13
6	NC	59	I/O-D0/CLK2	112	I/O-G15 (TDO)
7	NC	60	GND	113	GND
8	V_{DD}	61	V_{DD}	114	NC
9	I/O-B15 (TDI)	62	I/O-E0/CLK1	115	NC
10	I/O-B13	63	I/O-E2	116	NC
11	I/O-B12	64	I/O-E3	117	NC
12	I/O-B11	65	I/O-E4	118	NC
13	I/O-B10	66	GND	119	NC
14	I/O-B8	67	I/O-E5	120	NC
15	I/O-B7	68	I/O-E7	121	I/O-H0
16	I/O-B5	69	I/O-E8	122	I/O-H2
17	GND	70	I/O-E10	123	I/O-H3
18	I/O-B4	71	I/O-E11	124	NC
19	I/O-B3	72	I/O-E12	125	NC
20	I/O-B2	73	I/O-E13	126	NC
21	I/O-B0	74	NC	127	NC
22	I/O-C15 (TMS)	75	NC	128	I/O-H4
23	I/O-C13	76	NC	129	I/O-H5
24	I/O-C12	77	NC	130	I/O-H7
25	I/O-C11	78	I/O-E15	131	I/O-H8
26	V_{DD}	79	V_{DD}	132	I/O-H10
27	I/O-C10	80	I/O-F0	133	V_{DD}
28	I/O-C8	81	NC	134	I/O-H11
29	I/O-C7	82	NC	135	I/O-H12
30	I/O-C5	83	NC	136	I/O-H13
31	I/O-C4	84	NC	137	I/O-H15
32	I/O-C3	85	NC	138	GND
33	I/O-C2	86	NC	139	IN0/CLK0
34	NC	87	NC	140	IN2-gtsn
35	NC	88	I/O-F2	141	IN1
36	NC	89	I/O-F3	142	IN3
37	NC	90	I/O-F4	143	V_{DD}
38	NC	91	I/O-F5	144	I/O-A15/CLK3
39	NC	92	I/O-F7	145	I/O-A13
40	NC	93	I/O-F8	146	I/O-A12
41	I/O-C0	94	I/O-F10	147	I/O-A11
42	GND	95	GND	148	GND
43	I/O-D15	96	I/O-F11	149	I/O-A10
44	NC	97	I/O-F12	150	I/O-A8
45	NC	98	I/O-F13	151	I/O-A7
46	NC	99	I/O-F15 (TCK)	152	I/O-A5
47	NC	100	I/O-G0	153	I/O-A4
48	I/O-D13	101	I/O-G2	154	NC
49	I/O-D12	102	I/O-G3	155	NC
50	I/O-D11	103	I/O-G4	156	NC
51	I/O-D10	104	V_{DD}	157	NC
52	I/O-D8	105	I/O-G5	158	I/O-A3
53	I/O-D7	106	I/O-G7	159	I/O-A2
				160	I/O-A0

* THE TEST MODE SELECT (TMS) FUNCTION IS INACTIVE ON NON-ISR ARCHITECTURES.

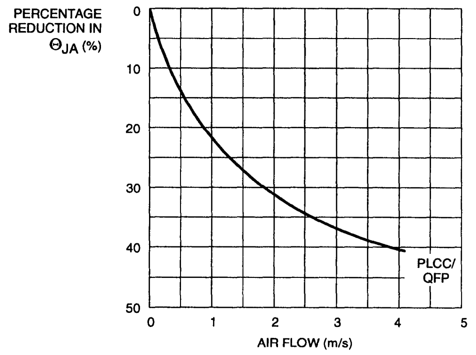
SP00470A

Package Thermal Characteristics

Philips Semiconductors uses the Temperature Sensitive Parameter (TSP) method to test thermal resistance. This method meets Mil-Std-883C Method 1012.1 and is described in Philips *1995 IC Package Databook*. Thermal resistance varies slightly as a function of input power. As input power increases, thermal resistance changes approximately 5% for a 100% change in power.

Figure 7 is a derating curve for the change in Θ_{JA} with airflow based on wind tunnel measurements. It should be noted that the wind flow dynamics are more complex and turbulent in actual applications than in a wind tunnel. Also, the test boards used in the wind tunnel contribute significantly to forced convection heat transfer, and may not be similar to the actual circuit board, especially in size.

Package	Θ_{JA}
84-pin PLCC	32.8°C/W
100-pin PQFP	41.2°C/W
100-pin TQFP	47.4°C/W
128-pin LQFP	45.0°C/W
160-pin PQFP	31.9°C/W



SP00419A

Figure 7. Average Effect of Airflow on Θ_{JA}

128 macrocell CPLD PZ5128

FEATURES

- Industry's first TotalCMOS™ PLD – both CMOS design and process technologies
- Fast Zero Power (FZP™) design technique provides ultra-low power and very high speed
- IEEE 1149.1–compliant, JTAG Testing Capability
  - 4 pin JTAG interface (TCK, TMS, TDI, TDO)
  - IEEE 1149.1 TAP Controller
  - JTAG commands include: Bypass, Sample/Preload, Extest, Usercode, Idcode, HighZ
- 5 Volt, In–System Programmable (ISP) using the JTAG interface
  - On–chip supervoltage generation
  - ISP commands include: Enable, Erase, Program, Verify
  - Supported by multiple ISP programming platforms
- High speed pin-to-pin delays of 10ns
- Ultra-low static power of less than 100μA
- Dynamic power that is 70% lower at 50MHz than competing devices
- 100% routable with 100% utilization while all pins and all macrocells are fixed
- Deterministic timing model that is extremely simple to use
- 4 clocks with programmable polarity at every macrocell
- Support for complex asynchronous clocking
- Innovative XPLA™ architecture combines high speed with extreme flexibility
- 1000 erase/program cycles guaranteed
- 20 years data retention guaranteed
- Logic expandable to 37 product terms
- PCI compliant
- Advanced 0.5μ E^2CMOS process
- Security bit prevents unauthorized access
- Design entry and verification using industry standard and Philips CAE tools
- Reprogrammable using industry standard device programmers
- Innovative Control Term structure provides either sum terms or product terms in each logic block for:
  - Programmable 3-State buffer
  - Asynchronous macrocell register preset/reset
- Programmable global 3-State pin facilitates 'bed of nails' testing without using logic resources
- Available in PLCC, TQFP, and PQFP packages
- Available in both Commercial and Industrial grades

Table 1. PZ5128 Features

	PZ5128
Usable gates	4000
Maximum inputs	100
Maximum I/Os	96
Number of macrocells	128
Propagation delay (ns)	10.0
Packages	84-pin PLCC, 100-pin PQFP, 100-pin TQFP 128-pin LQFP, 160-pin PQFP

DESCRIPTION

The PZ5128 CPLD (Complex Programmable Logic Device) is the third in a family of Fast Zero Power (FZP™) CPLDs from Philips Semiconductors. These devices combine high speed and zero power in a 128 macrocell CPLD. With the FZP™ design technique, the PZ5128 offers true pin-to-pin speeds of 10ns, while simultaneously delivering power that is less than 100μA at standby without the need for 'turbo bits' or other power down schemes. By replacing conventional sense amplifier methods for implementing product terms (a technique that has been used in PLDs since the bipolar era) with a cascaded chain of pure CMOS gates, the dynamic power is also substantially lower than any competing CPLD – 70% lower at 50MHz. These devices are the first TotalCMOS™ PLDs, as they use both a CMOS process technology **and** the patented full CMOS FZP™ design technique. For 3V applications, Philips also offers the high speed PZ3128 CPLD that offers these features in a full 3V implementation.

The Philips FZP™ CPLDs introduce the new patent-pending XPLA™ (eXtended Programmable Logic Array) architecture. The XPLA™ architecture combines the best features of both PLA and PAL™ type structures to deliver high speed and flexible logic allocation that results in superior ability to make design changes with fixed pinouts. The XPLA™ structure in each logic block provides a fast 10ns PAL™ path with 5 dedicated product terms per output. This PAL™ path is joined by an additional PLA structure that deploys a pool of 32 product terms to a fully programmable OR array that can allocate the PLA product terms to any output in the logic block. This combination allows logic to be allocated efficiently throughout the logic block and supports as many as 37 product terms on an output. The speed with which logic is allocated from the PLA array to an output is only 2ns, regardless of the number of PLA product terms used, which results in worst case t_{PD}'s of only 12ns from any pin to any other pin. In addition, logic that is common to multiple outputs can be placed on a single PLA product term and shared across multiple outputs via the OR array, effectively increasing design density.

The PZ5128 CPLDs are supported by industry standard CAE tools (Cadence, Mentor, Synopsys, Synario, Viewlogic, MINC), using text (Abel, VHDL, Verilog) and/or schematic entry. Design verification uses industry standard simulators for functional and timing simulation. Development is supported on personal computer, Sparc, and HP platforms. Device fitting uses either MINC or Philips Semiconductors-developed tools.

The PZ5128 CPLD is electrically reprogrammable using industry standard device programmers from vendors such as Data I/O, BP Microsystems, SMS, and others. The PZ5128 also includes an industry-standard, IEEE 1149.1, JTAG interface through which in-system programming (ISP) and reprogramming of the device is supported.

PAL is a registered trademark of Advanced Micro Devices, Inc.

ORDERING INFORMATION

ORDER CODE	DESCRIPTION	DESCRIPTION	DRAWING NUMBER
PZ5128-S10A84	84-pin PLCC, 10ns t_{PD}	Commercial temp range, 5 volt power supply, ± 5%	SOT189-3
PZ5128-S12A84	84-pin PLCC, 12ns t_{PD}	Commercial temp range, 5 volt power supply, ± 5%	SOT189-3
PZ5128IS15A84	84-pin PLCC, 15ns t_{PD}	Industrial temp range, 5 volt power supply, ± 10%	SOT189-3
PZ5128-S10BB1	100-pin PQFP, 10ns t_{PD}	Commercial temp range, 5 volt power supply, ± 5%	SOT382-1
PZ5128-S12BB1	100-pin PQFP, 12ns t_{PD}	Commercial temp range, 5 volt power supply, ± 5%	SOT382-1
PZ5128IS15BB1	100-pin PQFP, 15ns t_{PD}	Industrial temp range, 5 volt power supply, ± 10%	SOT382-1
PZ5128-S10BP	100-pin TQFP, 10ns t_{PD}	Commercial temp range, 5 volt power supply, ± 5%	SOT386-1
PZ5128-S12BP	100-pin TQFP, 12ns t_{PD}	Commercial temp range, 5 volt power supply, ± 5%	SOT386-1
PZ5128IS15BP	100-pin TQFP, 15ns t_{PD}	Industrial temp range, 5 volt power supply, ± 10%	SOT386-1
PZ5128-S10BE	128-pin LQFP, 10ns t_{PD}	Commercial temp range, 5 volt power supply, ± 5%	SOT425-1
PZ5128-S12BE	128-pin LQFP, 12ns t_{PD}	Commercial temp range, 5 volt power supply, ± 5%	SOT425-1
PZ5128IS15BE	128-pin LQFP, 15ns t_{PD}	Industrial temp range, 5 volt power supply, ± 10%	SOT425-1
PZ5128-S10BB2	160-pin PQFP, 10ns t_{PD}	Commercial temp range, 5 volt power supply, ± 5%	SOT322-2
PZ5128-S12BB2	160-pin PQFP, 12ns t_{PD}	Commercial temp range, 5 volt power supply, ± 5%	SOT322-2
PZ5128IS15BB2	160-pin PQFP, 15ns t_{PD}	Industrial temp range, 5 volt power supply, ± 10%	SOT322-2

XPLA™ ARCHITECTURE

Figure 1 shows a high level block diagram of a 128 macrocell device implementing the XPLA™ architecture. The XPLA™ architecture consists of logic blocks that are interconnected by a Zero-power Interconnect Array (ZIA). The ZIA is a virtual crosspoint switch. Each logic block is essentially a 36V16 device with 36 inputs from the ZIA and 16 macrocells. Each logic block also provides 32 ZIA feedback paths from the macrocells and I/O pins.

From this point of view, this architecture looks like many other CPLD architectures. What makes the CoolRunner™ family unique is what is inside each logic block and the design technique used to implement these logic blocks. The contents of the logic block will be described next.

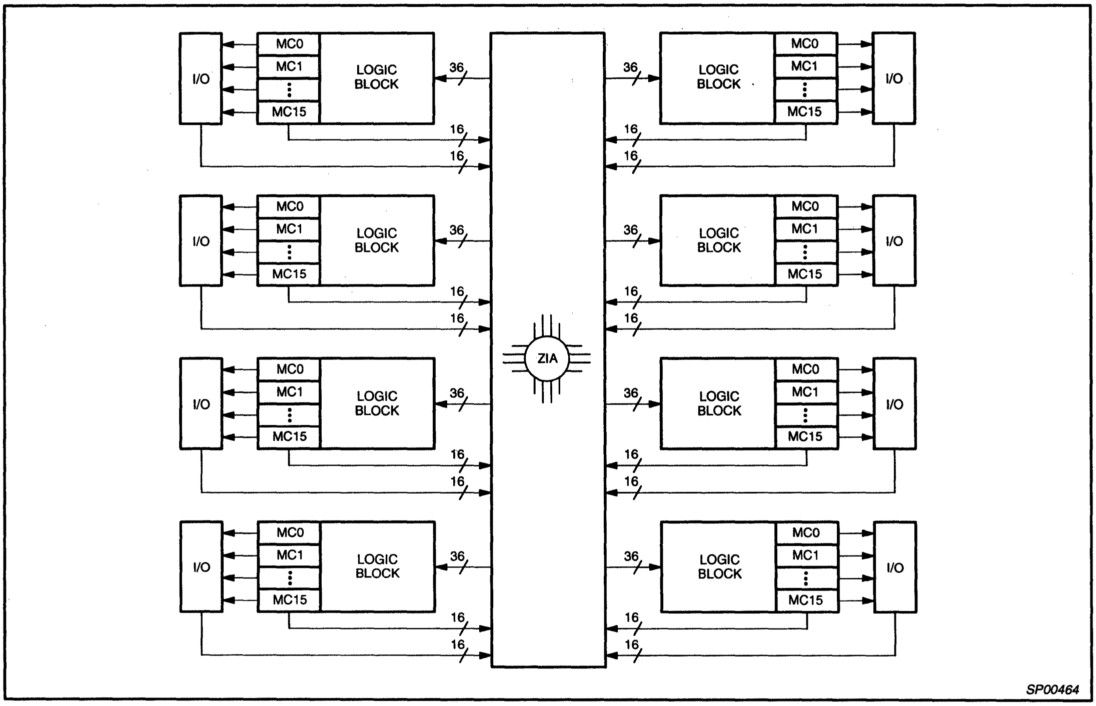


Figure 1. Philips XPLA CPLD Architecture

Logic Block Architecture

Figure 2 illustrates the logic block architecture. Each logic block contains control terms, a PAL array, a PLA array, and 16 macrocells. the 6 control terms can individually be configured as either SUM or PRODUCT terms, and are used to control the preset/reset and output enables of the 16 macrocells' flip-flops. The PAL array consists of a programmable AND array with a fixed OR array, while the PLA array consists of a programmable AND array with a programmable OR array. The PAL array provides a high speed path through the array, while the PLA array provides increased product term density.

Each macrocell has 5 dedicated product terms from the PAL array. The pin-to-pin t_{PD} of the PZ5128 device through the PAL array is 10ns. If a macrocell needs more than 5 product terms, it simply gets the additional product terms from the PLA array. The PLA array consists of 32 product terms, which are available for use by all 16 macrocells. The additional propagation delay incurred by a macrocell using 1 or all 32 PLA product terms is just 2ns. So the total pin-to-pin t_{PD} for the PZ5128 using 6 to 37 product terms is 12ns (10ns for the PAL + 2ns for the PLA).

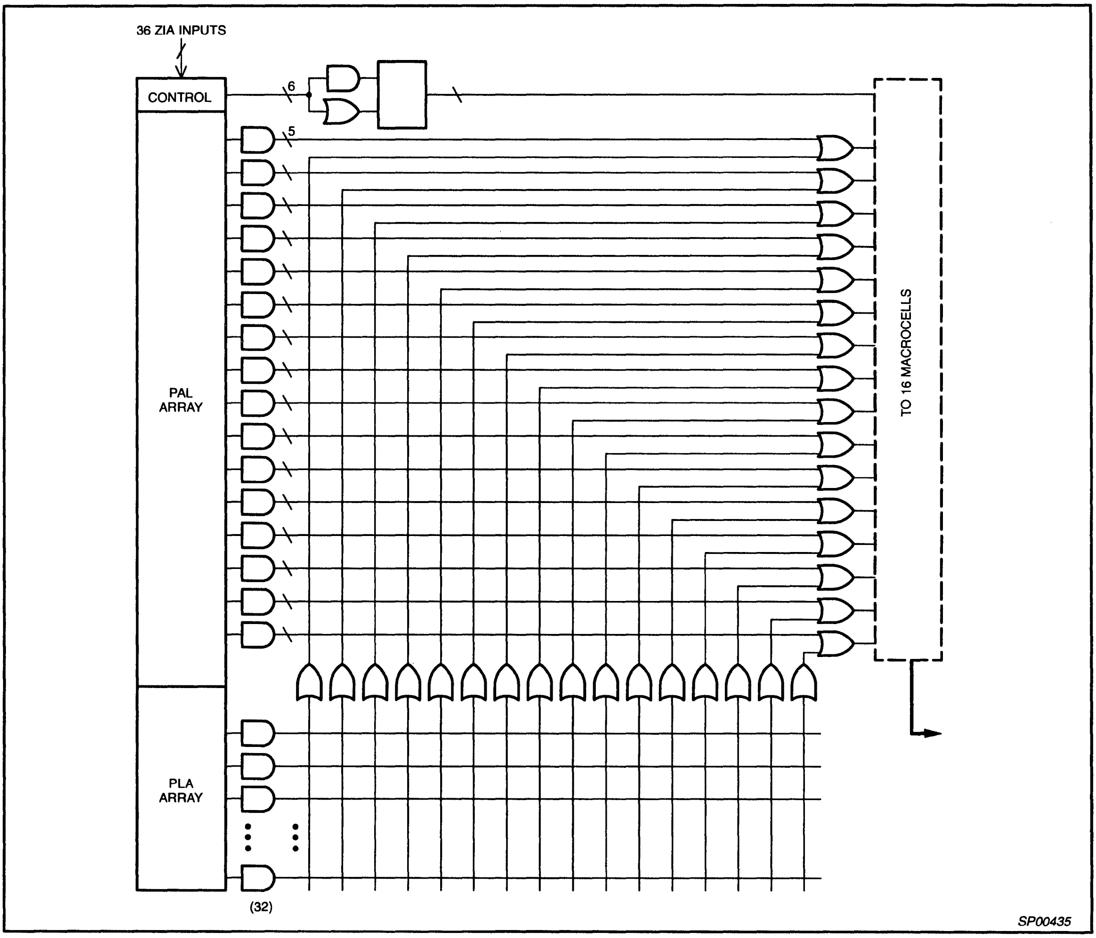


Figure 2. Philips Logic Block Architecture

Macrocell Architecture

Figure 3 shows the architecture of the macrocell used in the CoolRunner™ family. The macrocell consists of a flip-flop that can be configured as either a D or T type. A D-type flip-flop is generally more useful for implementing state machines and data buffering. A T-type flip-flop is generally more useful in implementing counters. All CoolRunner™ family members provide both synchronous and asynchronous clocking and provide the ability to clock off either the falling or rising edges of these clocks. These devices are designed such that the skew between the rising and falling edges of a clock are minimized for clocking integrity. There are 4 clocks available on the PZ5128 device. Clock 0 (CLK0) is designated as the "synchronous" clock and must be driven by an external source. Clock 1 (CLK1), Clock 2 (CLK2), and Clock 3 (CLK3) can either be used as a synchronous clock (driven by an external source) or as an asynchronous clock (driven by a macrocell equation).

Two of the control terms (CT0 and CT1) are used to control the Preset/Reset of the macrocell's flip-flop. The Preset/Reset feature for each macrocell can also be disabled. Note that the Power-on Reset leaves all macrocells in the "zero" state when power is properly applied. The other 4 control terms (CT2–CT5) can be used to control the Output Enable of the macrocell's output buffers. The reason there are as many control terms dedicated for the Output Enable of the macrocell is to insure that all CoolRunner™ devices are PCI compliant. The macrocell's output buffers can also be always enabled or disabled. All CoolRunner™ devices also provide a Global Tri-State ($\overline{\text{GTS}}$) pin, which, when pulled Low, will 3-State all the outputs of the device. This pin is provided to support "In-Circuit Testing" or "Bed-of-Nails Testing".

There are two feedback paths to the ZIA: one from the macrocell, and one from the I/O pin. The ZIA feedback path before the output buffer is the macrocell feedback path, while the ZIA feedback path after the output buffer is the I/O pin ZIA path. When the macrocell is used as an output, the output buffer is enabled, and the macrocell feedback path can be used to feedback the logic implemented in the macrocell. When the I/O pin is used as an input, the output buffer will be 3-Stated and the input signal will be fed into the ZIA via the I/O feedback path, and the logic implemented in the buried macrocell can be fed back to the ZIA via the macrocell feedback path. It should be noted that unused inputs or I/Os should be properly terminated.

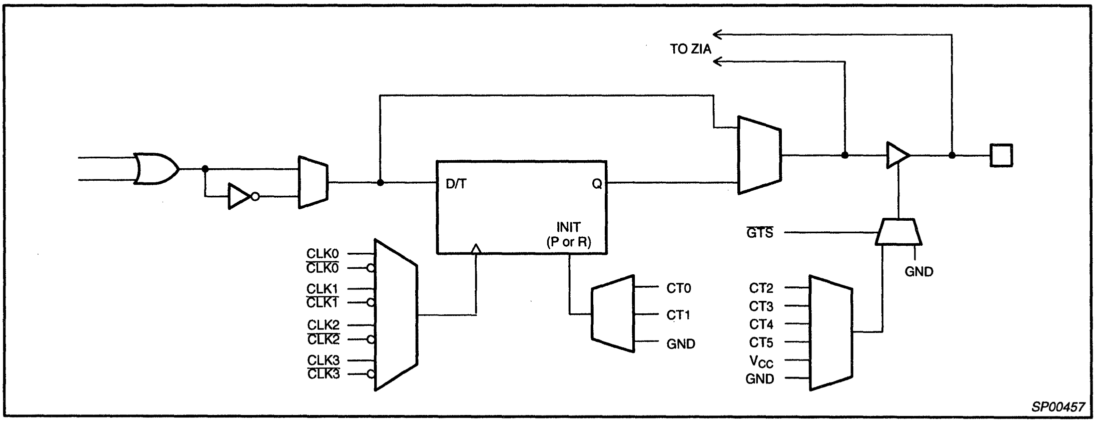


Figure 3. PZ5128 Macrocell Architecture

Simple Timing Model

Figure 4 shows the CoolRunner™ Timing Model. The CoolRunner™ timing model looks very much like a 22V10 timing model in that there are three main timing parameters, including t_{PD}, t_{SU}, and t_{CO}. In other competing architectures, the user may be able to fit the design into the CPLD, but is not sure whether system timing requirements can be met until after the design has been fit into the device. This is because the timing models of competing architectures are very complex and include such things as timing dependencies on the number of parallel expanders borrowed, sharable expanders, varying number of X and Y routing channels used, etc. In the XPLA™ architecture, the user knows up front whether the design will meet system timing requirements. This is due to the simplicity of the timing model.

TotalCMOS™ Design Technique for Fast Zero Power

Philips is the first to offer a TotalCMOS™ CPLD, both in process technology and design technique. Philips employs a cascade of CMOS gates to implement its Sum of Products instead of the traditional sense amp approach. This CMOS gate implementation allows Philips to offer CPLDs which are both high performance and low power, breaking the paradigm that to have low power, you must have low performance. Refer to Figure 5 and Table 2 showing the I_{DD} vs. Frequency of our PZ5128 TotalCMOS™ CPLD.

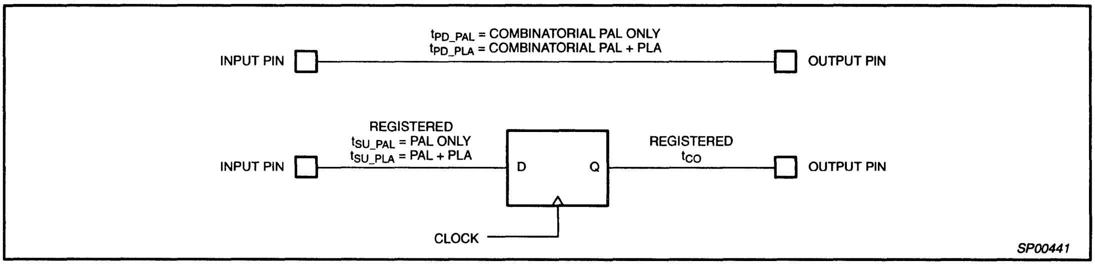


Figure 4. CoolRunner™ Timing Model

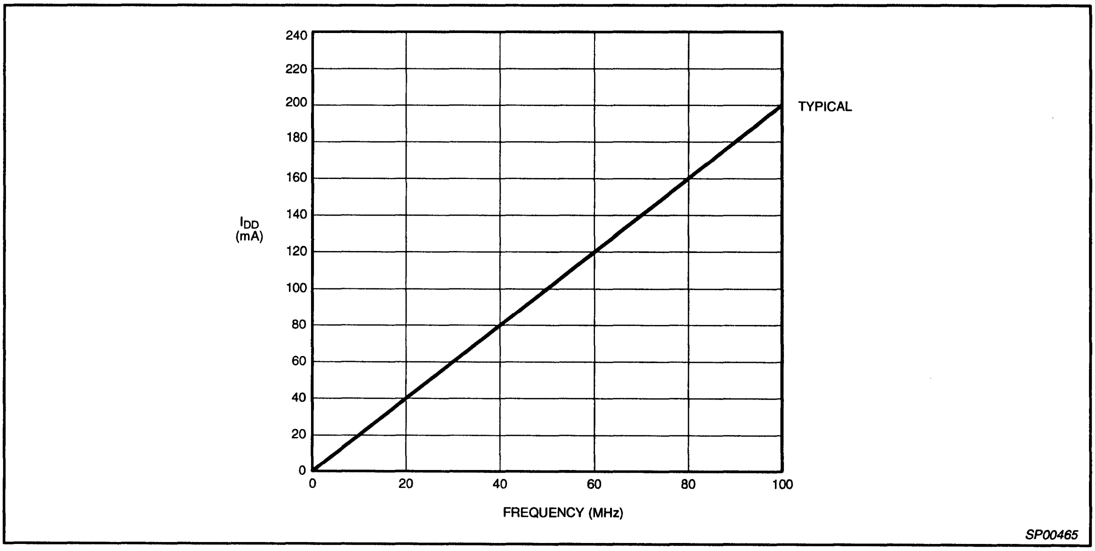


Figure 5. I_{DD} vs. Frequency @ V_{DD} = 5.0V, 25°C

Table 2. I_{DD} vs. Frequency

V_{DD} = 5.00V

FREQUENCY (MHz)	0	20	40	60	80	100
Typical I_{DD} (mA)	0.10	40	80	120	160	200

JTAG Testing Capability

JTAG is the commonly-used acronym for the Boundary Scan Test (BST) feature defined for integrated circuits by IEEE Standard 1149.1. This standard defines input/output pins, logic control functions, and commands which facilitate both board and device level testing without the use of specialized test equipment. BST provides the ability to test the external connections of a device, test the internal logic of the device, and capture data from the device during normal operation. BST provides a number of benefits in each of the following areas:

- Testability
  - Allows testing of an unlimited number of interconnects on the printed circuit board
  - Testability is designed in at the component level
  - Enables desired signal levels to be set at specific pins (Preload)
  - Data from pin or core logic signals can be examined during normal operation
- Reliability
  - Eliminates physical contacts common to existing test fixtures (e.g., "bed-of-nails")
  - Degradation of test equipment is no longer a concern
  - Facilitates the handling of smaller, surface-mount components
  - Allows for testing when components exist on both sides of the printed circuit board
- Cost
  - Reduces/eliminates the need for expensive test equipment
  - Reduces test preparation time
  - Reduces spare board inventories

The Philips PZ5128's JTAG interface includes a TAP Port and a TAP Controller, both of which are defined by the IEEE 1149.1 JTAG Specification. As implemented in the Philips PZ5128, the TAP Port includes four of the five pins (refer to Table 3) described in the JTAG specification: TCK, TMS, TDI, and TDO. The fifth signal defined by the JTAG specification is TRST* (Test Reset). TRST* is considered an optional signal, since it is not actually required to perform BST or ISP. The Philips PZ5128 saves an I/O pin for general purpose use by not implementing the optional TRST* signal in the JTAG interface. Instead, the Philips PZ5128 supports the test reset functionality through the use of its power up reset circuit, which is included in all Philips CPLDs. The pins associated with the power up reset circuit should connect to an external pull-up resistor to keep the JTAG signals from floating when they are not being used.

In the Philips PZ5128, the four mandatory JTAG pins each require a unique, dedicated pin on the device. However, if JTAG and ISP are not desired in the end-application, these pins may instead be used as additional general I/O pins. The decision as to whether these pins are used for JTAG/ISP or as general I/O is made when the JEDEC file is generated. If the use of JTAG/ISP is selected, the dedicated pins are not available for general purpose use. However, unlike competing CPLD's, the Philips PZ5128 does allow the macrocell logic associated with these dedicated pins to be used as buried logic even when JTAG/ISP is selected. Table 4 defines the dedicated pins used by the four mandatory JTAG signals for each of the PZ5128 package types.

The JTAG specifications defines two sets of commands to support boundary-scan testing: high-level commands and low-level commands. High-level commands are executed via board test software on an a user test station such as automated test equipment, a PC, or an engineering workstation (EWS). Each high-level command comprises a sequence of low level commands. These low-level commands are executed within the component under test, and therefore must be implemented as part of the TAP Controller design. The set of low-level boundary-scan commands implemented in the Philips PZ5128 is defined in Table 5. By supporting this set of low-level commands, the PZ5128 allows execution of all high-level boundary-scan commands.

Table 3. JTAG Pin Description

PIN	NAME	DESCRIPTION
TCK	Test Clock Output	Clock pin to shift the serial data and instructions in and out of the TDI and TDO pins, respectively. TCK is also used to clock the TAP Controller state machine.
TMS	Test Mode Select	Serial input pin selects the JTAG instruction mode. TMS should be driven high during user mode operation.
TDI	Test Data Input	Serial input pin for instructions and test data. Data is shifted in on the rising edge of TCK.
TDO	Test Data Output	Serial output pin for instructions and test data. Data is shifted out on the falling edge of TCK. The signal is tri-stated if data is not being shifted out of the device.

Table 4. PZ5128 JTAG Pinout by Package Type

DEVICE	(PIN NUMBER / MACROCELL #)			
	TCK	TMS	TDI	TDO
PZ5128				
84-pin PLCC	62 / 96 (F15)	23 / 48 (C15)	14 / 32 (B15)	71 / 112 (G15)
100-pin PQFP	64 / 96 (F15)	17 / 48 (C15)	6 / 32 (B15)	75 / 112 (G15)
100-pin TQFP	62 / 96 (F15)	15 / 48 (C15)	4 / 32 (B15)	73 / 112 (G15)
128-pin LQFP	82 / 96 (F15)	21 / 48 (C15)	8 / 32 (B15)	95 / 112 (G15)
160-pin PQFP	99 / 96 (F15)	22 / 48 (C15)	9 / 32 (B15)	112/ 112 (G15)

Table 5. PZ5128 Low-Level JTAG Boundary-Scan Commands

INSTRUCTION **(Instruction Code)** ***Register Used***	**DESCRIPTION**
Sample/Preload (0010) *Boundary–Scan Register*	The mandatory SAMPLE/PRELOAD instruction allows a snapshot of the normal operation of the component to be taken and examined. It also allows data values to be loaded onto the latched parallel outputs of the Boundary-Scan Shift-Register prior to selection of the other boundary-scan test instructions.
Extest (0000) *Boundary-Scan Register*	The mandatory EXTEST instruction allows testing of off-chip circuitry and board level interconnections. Data would typically be loaded onto the latched parallel outputs of Boundary-Scan Shift-Register using the Sample/Preload instruction prior to selection of the EXTEST instruction.
Bypass (1111) *Bypass Register*	Places the 1 bit bypass register between the TDI and TDO pins, which allows the BST data to pass synchronously through the selected device to adjacent devices during normal device operation. The Bypass instruction can be entered by holding TDI at a constant high value and completing an Instruction-Scan cycle.
Idcode (0001) *Boundary-Scan Register*	Selects the IDCODE register and places it between TDI and TDO, allowing the IDCODE to be serially shifted out of TDO. The IDCODE instruction permits blind interrogation of the components assembled onto a printed circuit board. Thus, in circumstances where the component population may vary, it is possible to determine what components exist in a product.
HighZ (0101) *Bypass Register*	The HIGHZ instruction places the component in a state in which all of its system logic outputs are placed in an inactive drive state (e.g., high impedance). In this state, an in-circuit test system may drive signals onto the connections normally driven by a component output without incurring the risk of damage to the component. The HighZ instruction also forces the Bypass Register between TDI and TDO.

5-Volt, In-System Programming (ISP)

ISP is the ability to reconfigure the logic and functionality of a device, printed circuit board, or complete electronic system before, during, and after its manufacture and shipment to the end customer. ISP provides substantial benefits in each of the following areas:

- Design
  - Faster time-to-market
  - Debug partitioning and simplified prototyping
  - Printed circuit board reconfiguration during debug
  - Better device and board level testing
- Manufacturing
  - Multi-Functional hardware
  - Reconfiguarability for Test
  - Eliminates handling of “fine lead-pitch” components for programming
  - Reduced Inventory and manufacturing costs
  - Improved quality and reliability
- Field Support
  - Easy remote upgrades and repair
  - Support for field configuration, re-configuration, and customization

The Philips PZ5128 allows for 5-Volt, in-system programming/reprogramming of its EEPROM cells via its JTAG interface. An on-chip charge pump eliminates the need for externally-provided supervoltages, so that the PZ5128 may be easily programmed on the circuit board using only the 5-volt supply required by the device for normal operation. A set of low-level ISP basic commands implemented in the PZ5128 enable this feature. The ISP commands implemented in the Philips PZ5128 are specified in Table 6. Please note that an ENABLE command must precede all ISP commands **unless** an ENABLE command has already been given for a preceding ISP command **and** the device has not gone through a Test-Logic/Rest TAP Controller State.

Table 6. Low Level ISP Commands

INSTRUCTION ***(Register Used)***	**INSTRUCTION CODE**	**DESCRIPTION**
Enable *(ISP Shift Register)*	1001	Enables the Erase, Program, and Verify commands. Using the ENABLE instruction before the Erase, Program, and Verify instructions allows the user to specify the outputs the device using the JTAG Boundary–Scan SAMPLE/PRELOAD command.
Erase *(ISP Shift Register)*	1010	Erases the entire EEPROM array. The outputs during this operation can be defined by user by using the JTAG SAMPLE/PRELOAD command.
Program *(ISP Shift Register)*	1011	Programs the data in the ISP Shift Register into the addressed EEPROM row. The outputs during this operation can be defined by user by using the JTAG SAMPLE/PRELOAD command.
Verify *(ISP Shift Register)*	1100	Transfers the data from the addressed row to the ISP Shift Register. The data can then be shifted out and compared with the JEDEC file. The outputs during this operation can be defined by user by using the JTAG SAMPLE/PRELOAD command.

JTAG and ISP Interfacing

A number of industry-established methods exist for JTAG/ISP interfacing with CPLD's and other integrated circuits. The Philips PZ5128 supports the following methods:

- PC Parallel Port
- Workstation or PC Serial Port
- Embedded Processor
- Automated Test Equipment
- Third party Programmers
- High-End JTAG and ISP Tools

A Boundary-Scan Description Language (BSDL) description of the PZ5128 is also available from Philips for use in test program development. For more details on JTAG and ISP for the PZ5128, refer to the related application note: *JTAG and ISP in Philips CPLDs*.

Table 7. Programming Specifications

SYMBOL	PARAMETER	MIN.	MAX.	UNIT
DC Parameters				
V_{CCP}	V_{CC} supply program/verify	4.5	5.5	V
I_{CCP}	I_{CC} limit program/verify		200	mA
V_{IH}	Input voltage (High)	2.0		V
V_{IL}	Input voltage (Low)		0.8	V
V_{SOL}	Output voltage (Low)		0.5	V
V_{SOH}	Output voltage (High)	2.4		V
TDO_I_{OL}	Output current (Low)	12		mA
TDO_I_{OH}	Output current (High)	–12		mA
AC Parameters				
f_{MAX}	CLK maximum frequency	10		MHz
PWE	Pulse width erase	100		ms
PWP	Pulse width program	10		ms
PWV	Pulse width verify	10		µs
INIT	Initialization time	100		µs
TMS_SU	TMS setup time before TCK ↑	10		ns
TDI_SU	TDI setup time before TCK ↑	10		ns
TMS_H	TMS hold time after TCK ↑	20		ns
TDI_H	TDI hold time after TCK ↑	20		ns
TDO_CO	TDO valid after TCK ↓		30	ns

ABSOLUTE MAXIMUM RATINGS[1]

SYMBOL	PARAMETER	MIN.	MAX.	UNIT
V_{DD}	Supply voltage	–0.5	7.0	V
V_I	Input voltage	–1.2	V_{DD}+0.5	V
V_{OUT}	Output voltage	–0.5	V_{DD}+0.5	V
I_{IN}	Input current	–30	30	mA
I_{OUT}	Output current	–100	100	mA
T_J	Maximum junction temperature	–40	150	°C
T_{str}	Storage temperature	–65	150	°C

NOTE:

1. Stresses above those listed may cause malfunction or permanent damage to the device. This is a stress rating only. Functional operation at these or any other condition above those indicated in the operational and programming specification is not implied.

OPERATING RANGE

PRODUCT GRADE	TEMPERATURE	VOLTAGE
Commercial	0 to +70°C	5.0 ±5% V
Industrial	–40 to +85°C	5.0 ±10% V

DC ELECTRICAL CHARACTERISTICS FOR COMMERCIAL GRADE DEVICES

Commercial: 0°C ≤ T_{amb} ≤ +70°C; 4.75V ≤ V_{DD} ≤ 5.25V

SYMBOL	PARAMETER	TEST CONDITIONS	MIN.	MAX.	UNIT
V_{IL}	Input voltage low	V_{DD} = 4.75V		0.8	V
V_{IH}	Input voltage high	V_{DD} = 5.25V	2.0		V
V_I	Input clamp voltage	V_{DD} = 4.75V, I_{IN} = –18mA		–1.2	V
V_{OL}	Output voltage low	V_{DD} = 4.75V, I_{OL} = 12mA		0.5	V
V_{OH}	Output voltage high	V_{DD} = 4.75V, I_{OH} = –12mA	2.4		V
I_I	Input leakage current	V_{IN} = 0 to V_{DD}	–10	10	µA
I_{OZ}	3-Stated output leakage current	V_{IN} = 0 to V_{DD}	–10	10	µA
I_{DDQ}	Standby current	V_{DD} = 5.25V, T_{amb} = 0°C		100	µA
I_{DDD}[1]	Dynamic current	V_{DD} = 5.25V, T_{amb} = 0°C @ 1MHz		5	mA
		V_{DD} = 5.25V, T_{amb} = 0°C @ 50MHz		120	mA
I_{OS}	Short circuit output current[2]	1 pin at a time for no longer than 1 second	–50	–200	mA
C_{IN}	Input pin capacitance[2]	T_{amb} = 25°C, f = 1MHz		8	pF
C_{CLK}	Clock input capacitance[2]	T_{amb} = 25°C, f = 1MHz	5	12	pF
$C_{I/O}$	I/O pin capacitance[2]	T_{amb} = 25°C, f = 1MHz		10	pF

NOTES:

[1] This parameter measured with a 16-bit, loadable up/down counter loaded into every logic block, with all outputs enabled and unloaded. Inputs are tied to V_{DD} or ground. This parameter guaranteed by design and characterization, not testing.

[2] Typical values, not tested.

AC ELECTRICAL CHARACTERISTICS[1] FOR COMMERCIAL GRADE DEVICES

Commercial: 0°C ≤ T_{amb} ≤ +70°C; 4.75V ≤ V_{DD} ≤ 5.25V

SYMBOL	PARAMETER	–10		–12		UNIT
		MIN.	MAX.	MIN.	MAX.	
t_{PD_PAL}	Propagation delay time, input (or feedback node) to output through PAL	2	10	2	12	ns
t_{PD_PLA}	Propagation delay time, input (or feedback node) to output through PAL & PLA	3	12	3	14.5	ns
t_{CO}	Clock to out delay time	2	7	2	8	ns
t_{SU_PAL}	Setup time (from input or feedback node) through PAL	7		8		ns
t_{SU_PLA}	Setup time (from input or feedback node) through PAL + PLA	9		10.5		ns
t_H	Hold time		0		0	ns
t_{CH}	Clock High time	4		4		ns
t_{CL}	Clock Low time	4		4		ns
t_R	Input Rise time		20		20	ns
t_F	Input Fall time		20		20	ns
f_{MAX1}	Maximum FF toggle rate[2] 1/(t_{CH} + t_{CL})	125		125		MHz
f_{MAX2}	Maximum internal frequency[2] 1/(t_{SUPAL} + t_{CF})	80		69		MHz
f_{MAX3}	Maximum external frequency[2] 1/(t_{SUPAL} + t_{CO})	71		63		MHz
t_{BUF}	Output buffer delay time		1.5		1.5	ns
t_{PDF_PAL}	Input (or feedback node) to internal feedback node delay time through PAL	2	8.5	2	10.5	ns
t_{PDF_PLA}	Input (or feedback node) to internal feedback node delay time through PAL+PLA	3	10.5	3	13	ns
t_{CF}	Clock to internal feedback node delay time		5.5		6.5	ns
t_{INIT}	Delay from valid V_{DD} to valid reset		50		50	µs
t_{ER}	Input to output disable[3]		12		15	ns
t_{EA}	Input to output valid		12		15	ns
t_{RP}	Input to register preset		12.5		15	ns
t_{RR}	Input to register reset		12.5		15	ns

NOTES:

[1] Specifications measured with one output switching. See Figure 6 and Table 8 for derating.

[2] This parameter guaranteed by design and characterization, not by test.

[3] Output C_L = 5pF.

DC ELECTRICAL CHARACTERISTICS FOR INDUSTRIAL GRADE DEVICES

Industrial: $-40°C \le T_{amb} \le +85°C$; $4.5V \le V_{DD} \le 5.5V$

SYMBOL	PARAMETER	TEST CONDITIONS	MIN.	MAX.	UNIT
V_{IL}	Input voltage low	$V_{DD} = 4.5V$		0.8	V
V_{IH}	Input voltage high	$V_{DD} = 5.5V$	2.0		V
V_I	Input clamp voltage	$V_{DD} = 4.5V$, $I_{IN} = -18mA$		–1.2	V
V_{OL}	Output voltage low	$V_{DD} = 4.5V$, $I_{OL} = 12mA$		0.5	V
V_{OH}	Output voltage high	$V_{DD} = 4.5V$, $I_{OH} = -12mA$	2.4		V
I_I	Input leakage current	$V_{IN} = 0$ to V_{DD}	–10	10	µA
I_{OZ}	3-Stated output leakage current	$V_{IN} = 0$ to V_{DD}	–10	10	µA
I_{DDQ}	Standby current	$V_{DD} = 5.5V$, $T_{amb} = -40°C$		125	µA
I_{DDD}[1]	Dynamic current	$V_{DD} = 5.5V$, $T_{amb} = -40°C$ @ 1MHz		6	mA
		$V_{DD} = 5.5V$, $T_{amb} = -40°C$ @ 50MHz		125	mA
I_{OS}	Short circuit output current[2]	1 pin at a time for no longer than 1 second	–50	–230	mA
C_{IN}	Input pin capacitance[2]	$T_{amb} = 25°C$, f = 1MHz		8	pF
C_{CLK}	Clock input capacitance[2]	$T_{amb} = 25°C$, f = 1MHz	5	12	pF
$C_{I/O}$	I/O pin capacitance[2]	$T_{amb} = 25°C$, f = 1MHz		10	pF

NOTES:

1. This parameter measured with a 16–bit, loadable up/down counter loaded into every logic block, with all outputs enabled and unloaded. Inputs are tied to V_{DD} or ground. This parameter guaranteed by design and characterization, not testing.
2. Typical values, not tested.

AC ELECTRICAL CHARACTERISTICS[1] FOR INDUSTRIAL GRADE DEVICES

Industrial: $-40°C \le T_{amb} \le +85°C$; $4.5V \le V_{DD} \le 5.5V$

SYMBOL	PARAMETER	I15		UNIT
		MIN.	MAX.	
t_{PD_PAL}	Propagation delay time, input (or feedback node) to output through PAL	2	15	ns
t_{PD_PLA}	Propagation delay time, input (or feedback node) to output through PAL & PLA	3	17.5	ns
t_{CO}	Clock to out delay time	2	8	ns
t_{SU_PAL}	Setup time (from input or feedback node) through PAL	8		ns
t_{SU_PLA}	Setup time (from input or feedback node) through PAL + PLA	10.5		ns
t_H	Hold time		0	ns
t_{CH}	Clock High time	5		ns
t_{CL}	Clock Low time	5		ns
t_R	Input Rise time		20	ns
t_F	Input Fall time		20	ns
f_{MAX1}	Maximum FF toggle rate[2] $1/(t_{CH} + t_{CL})$	100		MHz
f_{MAX2}	Maximum internal frequency[2] $1/(t_{SUPAL} + t_{CF})$	69		MHz
f_{MAX3}	Maximum external frequency[2] $1/(t_{SUPAL} + t_{CO})$	63		MHz
t_{BUF}	Output buffer delay time		1.5	ns
t_{PDF_PAL}	Input (or feedback node) to internal feedback node delay time through PAL	2	13.5	ns
t_{PDF_PLA}	Input (or feedback node) to internal feedback node delay time through PAL+PLA	3	16	ns
t_{CF}	Clock to internal feedback node delay time		6.5	ns
t_{INIT}	Delay from valid V_{DD} to valid reset		50	µs
t_{ER}	Input to output disable[3]		15	ns
t_{EA}	Input to output valid		15	ns
t_{RP}	Input to register preset		17	ns
t_{RR}	Input to register reset		17	ns

NOTES:

1. Specifications measured with one output switching. See Figure 6 and Table 8 for derating.
2. This parameter guaranteed by design and characterization, not by test.
3. Output C_L = 5pF.

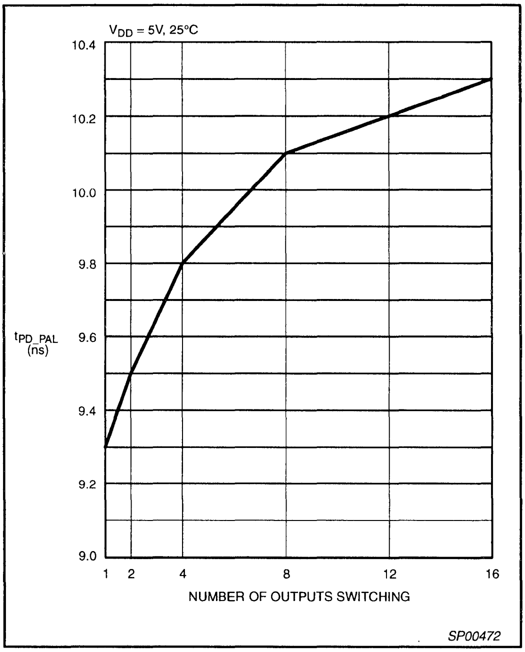


Figure 6. t_{PD_PAL} **vs. Outputs Switching**

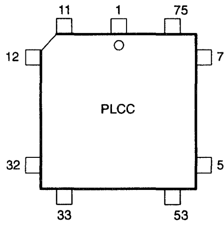
Table 8. t_{PD_PAL} vs. Number of Outputs Switching

V_{DD} = 5.00V

NUMBER OF OUTPUTS	**1**	**2**	**4**	**8**	**12**	**16**
Typical (ns)	9.3	9.5	9.8	10.1	10.2	10.3

PIN DESCRIPTIONS

84-Pin Plastic Leaded Chip Carrier

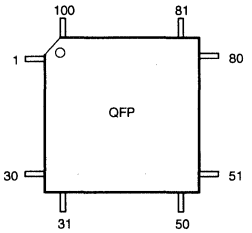


Pin	Function	Pin	Function	Pin	Function
1	IN1	29	I/O-C5	57	I/O-F7
2	IN3	30	I/O-C4	58	I/O-F10
3	V_{DD}	31	I/O-C2	59	GND
4	I/O-A15/CLK3	32	GND	60	I/O-F12
5	I/O-A13	33	I/O-D15	61	I/O-F13
6	I/O-A12	34	I/O-D12	62	I/O-F15 (TCK)
7	GND	35	I/O-D10	63	I/O-G0
8	I/O-A10	36	I/O-D8	64	I/O-G2
9	I/O-A7	37	I/O-D7	65	I/O-G4
10	I/O-A5	38	V_{DD}	66	V_{DD}
11	I/O-A4	39	I/O-D4	67	I/O-G7
12	I/O-A2	40	I/O-D2	68	I/O-G8
13	V_{DD}	41	I/O-D0/CLK2	69	I/O-G10
14	I/O-B15 (TDI)	42	GND	70	I/O-G12
15	I/O-B12	43	V_{DD}	71	I/O-G15 (TDO)
16	I/O-B10	44	I/O-E0/CLK1	72	GND
17	I/O-B8	45	I/O-E2	73	I/O-H2
18	I/O-B7	46	I/O-E4	74	I/O-H4
19	GND	47	GND	75	I/O-H5
20	I/O-B4	48	I/O-E7	76	I/O-H7
21	I/O-B2	49	I/O-E8	77	I/O-H10
22	I/O-B0	50	I/O-E10	78	V_{DD}
23	I/O-C15 (TMS)*	51	I/O-E12	79	I/O-H12
24	I/O-C13	52	I/O-E15	80	I/O-H13
25	I/O-C12	53	V_{DD}	81	I/O-H15
26	V_{DD}	54	I/O-F2	82	GND
27	I/O-C10	55	I/O-F4	83	IN0/CLK0
28	I/O-C7	56	I/O-F5	84	IN2-gtsn

* THE TEST MODE SELECT (TMS) FUNCTION IS INACTIVE ON NON-ISR ARCHITECTURES.

SP00467

100-Pin Plastic Quad Flat Package

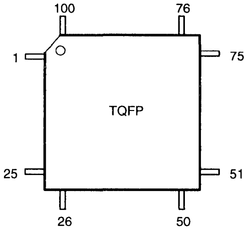


Pin	Function	Pin	Function	Pin	Function
1	I/O-A5	35	I/O-D5	69	I/O-G5
2	I/O-A4	36	V_{DD}	70	I/O-G7
3	I/O-A2	37	I/O-D4	71	I/O-G8
4	I/O-A0	38	I/O-D2	72	I/O-G10
5	V_{DD}	39	I/O-B0/CLK2	73	I/O-G12
6	I/O-B15 (TDI)	40	GND	74	I/O-G13
7	I/O-B13	41	V_{DD}	75	I/O-G15 (TDO)
8	I/O-B12	42	I/O-E0/CLK1	76	GND
9	I/O-B10	43	I/O-E2	77	I/O-H0
10	I/O-B8	44	I/O-E4	78	I/O-H2
11	I/O-B7	45	GND	79	I/O-H4
12	I/O-B5	46	I/O-E5	80	I/O-H5
13	GND	47	I/O-E7	81	I/O-H7
14	I/O-B4	48	I/O-E8	82	I/O-H8
15	I/O-B2	49	I/O-E10	83	I/O-H10
16	I/O-B0	50	I/O-E12	84	V_{DD}
17	I/O-C15 (TMS)*	51	I/O-E13	85	I/O-H12
18	I/O-C13	52	I/O-E15	86	I/O-H13
19	I/O-C12	53	V_{DD}	87	I/O-H15
20	V_{DD}	54	I/O-F0	88	GND
21	I/O-C10	55	I/O-F2	89	IN0/CLK0
22	I/O-C8	56	I/O-F4	90	IN2-gtsn
23	I/O-C7	57	I/O-F5	91	IN1
24	I/O-C5	58	I/O-F7	92	IN3
25	I/O-B9	59	I/O-F8	93	V_{DD}
26	I/O-C2	60	I/O-F10	94	I/O-A15/CLK3
27	I/O-C0	61	GND	95	I/O-A13
28	GND	62	I/O-F12	96	I/O-A12
29	I/O-D15	63	I/O-F13	97	GND
30	I/O-D13	64	I/O-F15 (TCK)	98	I/O-A10
31	I/O-D12	65	I/O-G0	99	I/O-A8
32	I/O-D10	66	I/O-G2	100	I/O-A7
33	I/O-D8	67	I/O-G4		
34	I/O-D7	68	V_{DD}		

* THE TEST MODE SELECT (TMS) FUNCTION IS INACTIVE ON NON-ISR ARCHITECTURES.

SP00468

100-Pin Thin Quad Flat Package

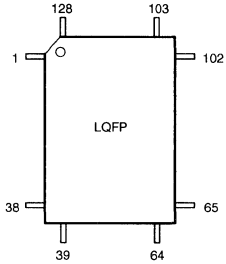


Pin	Function	Pin	Function	Pin	Function
1	I/O-A2	35	I/O-D4	69	I/O-G8
2	I/O-A0	36	I/O-D2	70	I/O-G10
3	V_{DD}	37	I/O-D0/CLK2	71	I/O-G12
4	I/O-B15 (TDI)	38	GND	72	I/O-G13
5	I/O-B13	39	V_{DD}	73	I/O-G15 (TDO)
6	I/O-B12	40	I/O-E0/CLK1	74	GND
7	I/O-B10	41	I/O-E2	75	I/O-H0
8	I/O-B8	42	I/O-E4	76	I/O-H2
9	I/O-B7	43	GND	77	I/O-H4
10	I/O-B5	44	I/O-E5	78	I/O-H5
11	GND	45	I/O-E7	79	I/O-H7
12	I/O-B4	46	I/O-E8	80	I/O-H8
13	I/O-B2	47	I/O-E10	81	I/O-H10
14	I/O-B0	48	I/O-E12	82	V_{DD}
15	I/O-C15 (TMS)*	49	I/O-E13	83	I/O-H12
16	I/O-C13	50	I/O-E15	84	I/O-H13
17	I/O-C12	51	V_{DD}	85	I/O-H15
18	V_{DD}	52	I/O-F0	86	GND
19	I/O-C10	53	I/O-F2	87	IN0/CLK0
20	I/O-C8	54	I/O-F4	88	IN2-gtsn
21	I/O-C7	55	I/O-F5	89	IN1
22	I/O-C5	56	I/O-F7	90	IN3
23	I/O-C4	57	I/O-F8	91	V_{DD}
24	I/O-C2	58	I/O-F10	92	I/O-A15/CLK3
25	I/O-C0	59	GND	93	I/O-A13
26	GND	60	I/O-F12	94	I/O-A12
27	I/O-D15	61	I/O-F13	95	GND
28	I/O-D13	62	I/O-F15 (TCK)	96	I/O-A10
29	I/O-D12	63	I/O-G0	97	I/O-A8
30	I/O-D10	64	I/O-G2	98	I/O-A7
31	I/O-D8	65	I/O-G4	99	I/O-A5
32	I/O-D7	66	V_{DD}	100	I/O-A4
33	I/O-D5	67	I/O-G5		
34	V_{DD}	68	I/O-G7		

* THE TEST MODE SELECT (TMS) FUNCTION IS INACTIVE ON NON-ISR ARCHITECTURES.

SP00485

128-Pin Low Profile Quad Flat Package

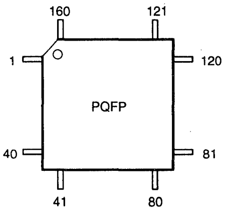


Pin	Function	Pin	Function	Pin	Function
1	I/O-A3	44	I/O-D7	87	V_{DD}
2	I/O-A2	45	I/O-D5	88	I/O-G5
3	I/O-A0	46	V_{DD}	89	I/O-G7
4	NC	47	I/O-D4	90	I/O-G8
5	NC	48	I/O-D3	91	I/O-G10
6	NC	49	I/O-D2	92	I/O-G11
7	V_{DD}	50	I/O-D0/CLK2	93	I/O-G12
8	I/O-B15 (TDI)	51	GND	94	I/O-G13
9	I/O-B13	52	V_{DD}	95	I/O-G15 (TDO)
10	I/O-B12	53	I/O-E0/CLK1	96	GND
11	I/O-B11	54	I/O-E2	97	NC
12	I/O-B10	55	I/O-E3	98	NC
13	I/O-B8	56	I/O-E4	99	NC
14	I/O-B7	57	GND	100	I/O-H0
15	I/O-B5	58	I/O-E5	101	I/O-H2
16	GND	59	I/O-E7	102	I/O-H3
17	I/O-B4	60	I/O-E8	103	I/O-H4
18	I/O-B3	61	I/O-E10	104	I/O-H5
19	I/O-B2	62	I/O-E11	105	I/O-H7
20	I/O-B0	63	I/O-E12	106	I/O-H8
21	I/O-C15 (TMS)*	64	I/O-E13	107	I/O-H10
22	I/O-C13	65	I/O-E15	108	V_{DD}
23	I/O-C12	66	V_{DD}	109	I/O-H11
24	I/O-C11	67	I/O-F0	110	I/O-H12
25	V_{DD}	68	NC	111	I/O-H13
26	I/O-C10	69	NC	112	I/O-H15
27	I/O-C8	70	NC	113	GND
28	I/O-C7	71	I/O-F2	114	IN0/CLK0
29	I/O-C5	72	I/O-F3	115	IN2-gtsn
30	I/O-C4	73	I/O-F4	116	IN1
31	I/O-C3	74	I/O-F5	117	IN3
32	I/O-C2	75	I/O-F7	118	V_{DD}
33	NC	76	I/O-F8	119	I/O-A15/CLK3
34	NC	77	I/O-F10	120	I/O-A13
35	NC	78	GND	121	I/O-A12
36	I/O-C0	79	I/O-F11	122	I/O-A11
37	GND	80	I/O-F12	123	GND
38	I/O-D15	81	I/O-F13	124	I/O-A10
39	I/O-D13	82	I/O-F15(TCK)	125	I/O-A8
40	I/O-D12	83	I/O-G0	126	I/O-A7
41	I/O-D11	84	I/O-G2	127	I/O-A5
42	I/O-D10	85	I/O-G3	128	I/O-A4
43	I/O-D8	86	I/O-G4		

* THE TEST MODE SELECT (TMS) FUNCTION IS INACTIVE ON NON-ISR ARCHITECTURES.

SP00469A

160-Pin Plastic Quad Flat Package



Pin	Function	Pin	Function	Pin	Function
1	NC	54	I/O-D5	107	I/O-G8
2	NC	55	V_{DD}	108	I/O-G10
3	NC	56	I/O-D4	109	I/O-G11
4	NC	57	I/O-D3	110	I/O-G12
5	NC	58	I/O-D2	111	I/O-G13
6	NC	59	I/O-D0/CLK2	112	I/O-G15 (TDO)
7	NC	60	GND	113	GND
8	V_{DD}	61	V_{DD}	114	NC
9	I/O-B15 (TDI)	62	I/O-E0/CLK1	115	NC
10	I/O-B13	63	I/O-E2	116	NC
11	I/O-B12	64	I/O-E3	117	NC
12	I/O-B11	65	I/O-E4	118	NC
13	I/O-B10	66	GND	119	NC
14	I/O-B8	67	I/O-E5	120	NC
15	I/O-B7	68	I/O-E7	121	I/O-H0
16	I/O-B5	69	I/O-E8	122	I/O-H2
17	GND	70	I/O-E10	123	I/O-H3
18	I/O-B4	71	I/O-E11	124	NC
19	I/O-B3	72	I/O-E12	125	NC
20	I/O-B2	73	I/O-E13	126	NC
21	I/O-B0	74	NC	127	NC
22	I/O-C15 (TMS)	75	NC	128	I/O-H4
23	I/O-C13	76	NC	129	I/O-H5
24	I/O-C12	77	NC	130	I/O-H7
25	I/O-C11	78	I/O-E15	131	I/O-H8
26	V_{DD}	79	V_{DD}	132	I/O-H10
27	I/O-C10	80	I/O-F0	133	V_{DD}
28	I/O-C8	81	NC	134	I/O-H11
29	I/O-C7	82	NC	135	I/O-H12
30	I/O-C5	83	NC	136	I/O-H13
31	I/O-C4	84	NC	137	I/O-H15
32	I/O-C3	85	NC	138	GND
33	I/O-C2	86	NC	139	IN0/CLK0
34	NC	87	NC	140	IN2-gtsn
35	NC	88	I/O-F2	141	IN1
36	NC	89	I/O-F3	142	IN3
37	NC	90	I/O-F4	143	V_{DD}
38	NC	91	I/O-F5	144	I/O-A15/CLK3
39	NC	92	I/O-F7	145	I/O-A13
40	NC	93	I/O-F8	146	I/O-A12
41	I/O-C0	94	I/O-F10	147	I/O-A11
42	GND	95	GND	148	GND
43	I/O-D15	96	I/O-F11	149	I/O-A10
44	NC	97	I/O-F12	150	I/O-A8
45	NC	98	I/O-F13	151	I/O-A7
46	NC	99	I/O-F15 (TCK)	152	I/O-A5
47	NC	100	I/O-G0	153	I/O-A4
48	I/O-D13	101	I/O-G2	154	NC
49	I/O-D12	102	I/O-G3	155	NC
50	I/O-D11	103	I/O-G4	156	NC
51	I/O-D10	104	V_{DD}	157	NC
52	I/O-D8	105	I/O-G5	158	I/O-A3
53	I/O-D7	106	I/O-G7	159	I/O-A2
				160	I/O-A0

* THE TEST MODE SELECT (TMS) FUNCTION IS INACTIVE ON NON-ISR ARCHITECTURES.

SP00470A

Package Thermal Characteristics

Philips Semiconductors uses the Temperature Sensitive Parameter (TSP) method to test thermal resistance. This method meets Mil-Std-883C Method 1012.1 and is described in Philips *1995 IC Package Databook*. Thermal resistance varies slightly as a function of input power. As input power increases, thermal resistance changes approximately 5% for a 100% change in power.

Figure 7 is a derating curve for the change in Θ_{JA} with airflow based on wind tunnel measurements. It should be noted that the wind flow dynamics are more complex and turbulent in actual applications than in a wind tunnel. Also, the test boards used in the wind tunnel contribute significantly to forced convection heat transfer, and may not be similar to the actual circuit board, especially in size.

Package	Θ_{JA}
84-pin PLCC	32.8 °C/W
100-pin PQFP	41.2 °C/W
100-pin TQFP	47.4 °C/W
128-pin LQFP	45.0 °C/W
160-pin PQFP	31.4 °C/W

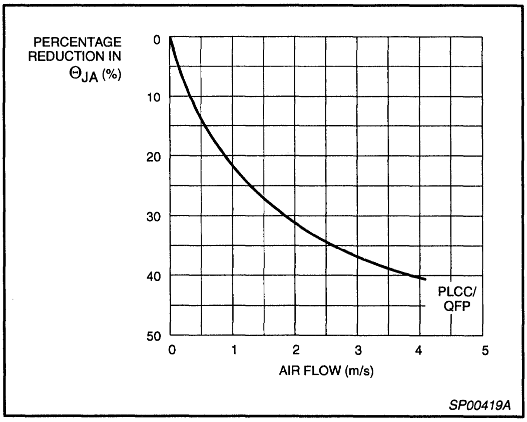


Figure 7. Average Effect of Airflow on Θ_{JA}

128 macrocell CPLD PZ3128–10

FEATURES

- Industry's first TotalCMOS™ PLD – both CMOS design and process technologies
- Fast Zero Power (FZP™) design technique provides ultra-low power and very high speed
- IEEE 1149.1–compliant, JTAG Testing Capability
  - 4 pin JTAG interface (TCK, TMS, TDI, TDO)
  - IEEE 1149.1 TAP Controller
  - JTAG commands include: Bypass, Sample/Preload, Extest, Usercode, Idcode, HighZ
- 3.3 Volt, In–System Programmable (ISP) using the JTAG interface
  - On–chip supervoltage generation
  - ISP commands include: Enable, Erase, Program, Verify
  - Supported by multiple ISP programming platforms
- High speed pin-to-pin delays of 10ns
- Ultra-low static power of less than 100μA
- Dynamic power that is 70% lower at 50MHz than competing devices
- 100% routable with 100% utilization while all pins and all macrocells are fixed
- Deterministic timing model that is extremely simple to use
- 4 clocks with programmable polarity at every macrocell
- Support for complex asynchronous clocking
- Innovative XPLA™ architecture combines high speed with extreme flexibility
- 1000 erase/program cycles guaranteed
- 20 years data retention guaranteed
- Logic expandable to 37 product terms
- PCI compliant
- Advanced 0.5μ E^2CMOS process
- Security bit prevents unauthorized access
- Design entry and verification using industry standard and Philips CAE tools
- Reprogrammable using industry standard device programmers
- Innovative Control Term structure provides either sum terms or product terms in each logic block for:
  - Programmable 3-State buffer
  - Asynchronous macrocell register preset/reset
- Programmable global 3-State pin facilitates 'bed of nails' testing without using logic resources
- Available in PLCC, TQFP, and PQFP packages
- Available in both Commercial and Industrial grades

Table 1. PZ3128 Features

	PZ3128
Usable gates	4000
Maximum inputs	100
Maximum I/Os	96
Number of macrocells	128
Propagation delay (ns)	10.0
Packages	84-pin PLCC, 100-pin PQFP, 100-pin TQFP, 128-pin LQFP, 160-pin PQFP

DESCRIPTION

The PZ3128 CPLD (Complex Programmable Logic Device) is the third in a family of Fast Zero Power (FZP™) CPLDs from Philips Semiconductors. These devices combine high speed and zero power in a 128 macrocell CPLD. With the FZP™ design technique, the PZ3128 offers true pin-to-pin speeds of 10ns, while simultaneously delivering power that is less than 100μA at standby without the need for 'turbo bits' or other power down schemes. By replacing conventional sense amplifier methods for implementing product terms (a technique that has been used in PLDs since the bipolar era) with a cascaded chain of pure CMOS gates, the dynamic power is also substantially lower than any competing CPLD – 70% lower at 50MHz. These devices are the first TotalCMOS™ PLDs, as they use both a CMOS process technology **and** the patented full CMOS FZP™ design technique. For 5V applications, Philips also offers the high speed PZ5128 CPLD that offers these features in a full 5V implementation.

The Philips FZP™ CPLDs introduce the new patent-pending XPLA™ (eXtended Programmable Logic Array) architecture. The XPLA™ architecture combines the best features of both PLA and PAL™ type structures to deliver high speed and flexible logic allocation that results in superior ability to make design changes with fixed pinouts. The XPLA™ structure in each logic block provides a fast 12ns PAL™ path with 5 dedicated product terms per output. This PAL™ path is joined by an additional PLA structure that deploys a pool of 32 product terms to a fully programmable OR array that can allocate the PLA product terms to any output in the logic block. This combination allows logic to be allocated efficiently throughout the logic block and supports as many as 37 product terms on an output. The speed with which logic is allocated from the PLA array to an output is only 2.5ns, regardless of the number of PLA product terms used, which results in worst case t_{PD}'s of only 12.5ns from any pin to any other pin. In addition, logic that is common to multiple outputs can be placed on a single PLA product term and shared across multiple outputs via the OR array, effectively increasing design density.

The PZ3128 CPLDs are supported by industry standard CAE tools (Cadence, Mentor, Synopsys, Synario, Viewlogic, MINC), using text (Abel, VHDL, Verilog) and/or schematic entry. Design verification uses industry standard simulators for functional and timing simulation. Development is supported on personal computer, Sparc, and HP platforms. Device fitting uses either MINC or Philips Semiconductors-developed tools.

The PZ3128 CPLD is electrically reprogrammable using industry standard device programmers from vendors such as Data I/O, BP Microsystems, SMS, and others. The PZ3128 also includes an industry-standard, IEEE 1149.1, JTAG interface through which in-system programming (ISP) and reprogramming of the device is supported.

PAL is a registered trademark of Advanced Micro Devices, Inc.

ORDERING INFORMATION

ORDER CODE	DESCRIPTION	DESCRIPTION	DRAWING NUMBER
PZ3128-S10A84	84-pin PLCC, 10ns t_{PD}	Commercial temp range, 3.3 volt power supply, ± 10%	SOT189-3
PZ3128IS12A84	84-pin PLCC, 12ns t_{PD}	Industrial temp range, 3.3 volt power supply, ± 10%	SOT189-3
PZ3128-S10BB1	100-pin PQFP, 10ns t_{PD}	Commercial temp range, 3.3 volt power supply, ± 10%	SOT382-1
PZ3128IS12BB1	100-pin PQFP, 12ns t_{PD}	Industrial temp range, 3.3 volt power supply, ± 10%	SOT382-1
PZ3128-S10BP	100-pin TQFP, 10ns t_{PD}	Commercial temp range, 3.3 volt power supply, ± 10%	SOT386-1
PZ3128IS12BP	100-pin TQFP, 12ns t_{PD}	Industrial temp range, 3.3 volt power supply, ± 10%	SOT386-1
PZ3128-S10BE	128-pin LQFP, 10ns t_{PD}	Commercial temp range, 3.3 volt power supply, ± 10%	SOT425-1
PZ3128IS12BE	128-pin LQFP, 12ns t_{PD}	Industrial temp range, 3.3 volt power supply, ± 10%	SOT425-1
PZ3128-S10BB2	160-pin PQFP, 10ns t_{PD}	Commercial temp range, 3.3 volt power supply, ± 10%	SOT322-2
PZ3128IS12BB2	160-pin PQFP, 12ns t_{PD}	Industrial temp range, 3.3 volt power supply, ± 10%	SOT322-2

XPLA™ ARCHITECTURE

Figure 1 shows a high level block diagram of a 128 macrocell device implementing the XPLA™ architecture. The XPLA™ architecture consists of logic blocks that are interconnected by a Zero-power Interconnect Array (ZIA). The ZIA is a virtual crosspoint switch. Each logic block is essentially a 36V16 device with 36 inputs from the ZIA and 16 macrocells. Each logic block also provides 32 ZIA feedback paths from the macrocells and I/O pins.

From this point of view, this architecture looks like many other CPLD architectures. What makes the CoolRunner™ family unique is what is inside each logic block and the design technique used to implement these logic blocks. The contents of the logic block will be described next.

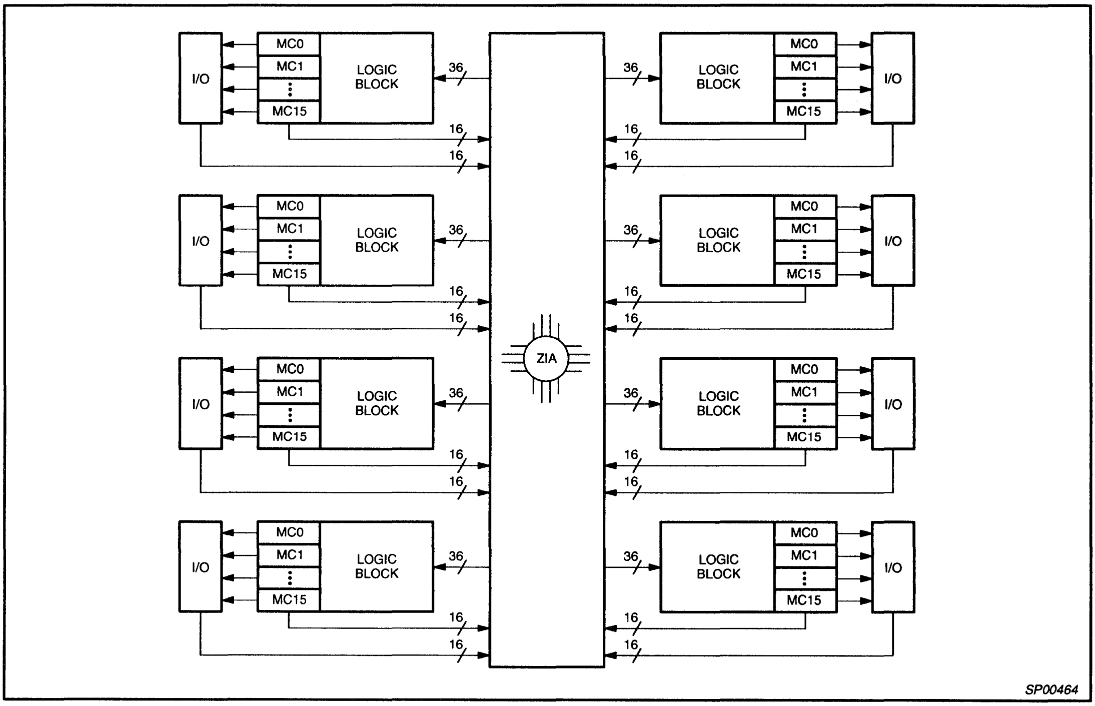


Figure 1. Philips XPLA CPLD Architecture

Logic Block Architecture

Figure 2 illustrates the logic block architecture. Each logic block contains control terms, a PAL array, a PLA array, and 16 macrocells. the 6 control terms can individually be configured as either SUM or PRODUCT terms, and are used to control the preset/reset and output enables of the 16 macrocells' flip-flops. The PAL array consists of a programmable AND array with a fixed OR array, while the PLA array consists of a programmable AND array with a programmable OR array. The PAL array provides a high speed path through the array, while the PLA array provides increased product term density.

Each macrocell has 5 dedicated product terms from the PAL array. The pin-to-pin t_{PD} of the PZ3128 device through the PAL array is 12ns. If a macrocell needs more than 5 product terms, it simply gets the additional product terms from the PLA array. The PLA array consists of 32 product terms, which are available for use by all 16 macrocells. The additional propagation delay incurred by a macrocell using 1 or all 32 PLA product terms is just 2.5ns. So the total pin-to-pin t_{PD} for the PZ3128 using 6 to 37 product terms is 12.5ns (10ns for the PAL + 2.5ns for the PLA).

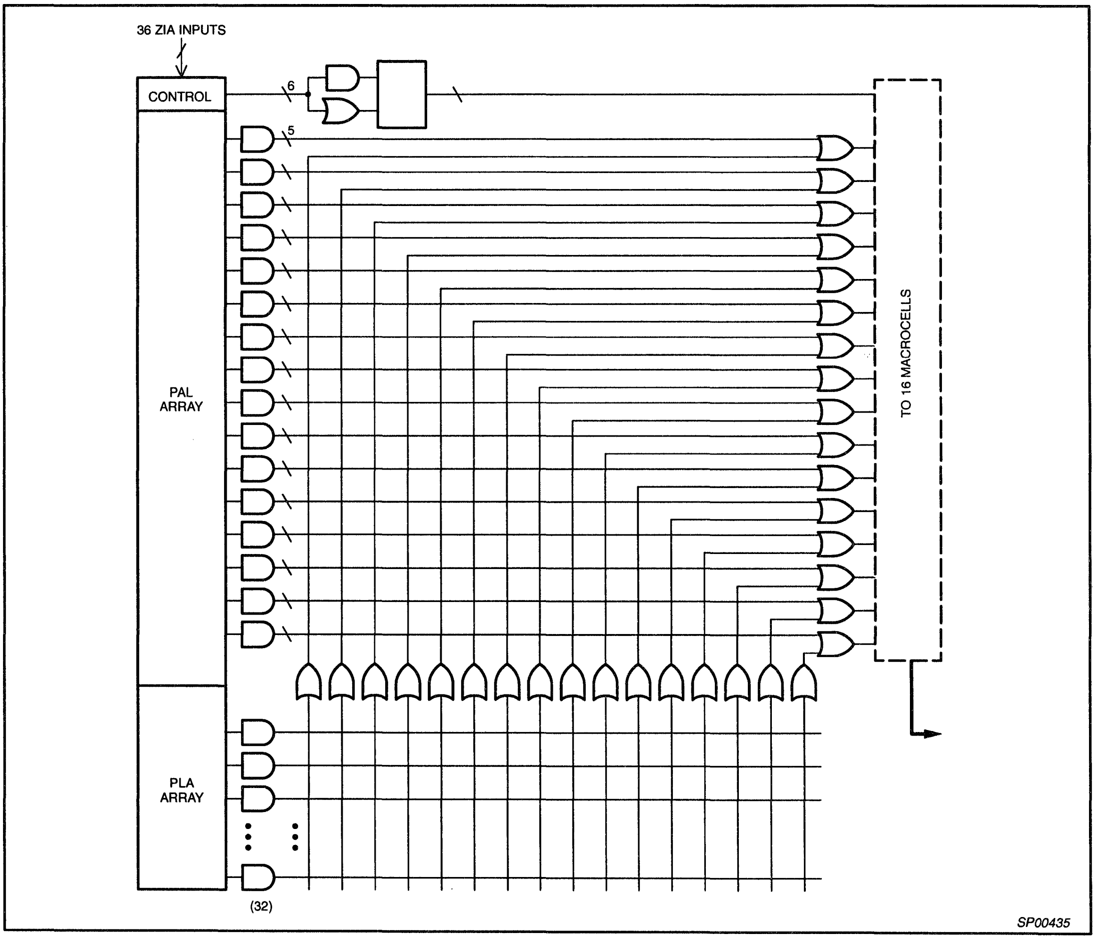


Figure 2. Philips Logic Block Architecture

Macrocell Architecture

Figure 3 shows the architecture of the macrocell used in the CoolRunner™ family. The macrocell consists of a flip-flop that can be configured as either a D or T type. A D-type flip-flop is generally more useful for implementing state machines and data buffering. A T-type flip-flop is generally more useful in implementing counters. All CoolRunner™ family members provide both synchronous and asynchronous clocking and provide the ability to clock off either the falling or rising edges of these clocks. These devices are designed such that the skew between the rising and falling edges of a clock are minimized for clocking integrity. There are 4 clocks available on the PZ3128 device. Clock 0 (CLK0) is designated as the "synchronous" clock and must be driven by an external source. Clock 1 (CLK1), Clock 2 (CLK2), and Clock 3 (CLK3) can either be used as a synchronous clock (driven by an external source) or as an asynchronous clock (driven by a macrocell equation).

Two of the control terms (CT0 and CT1) are used to control the Preset/Reset of the macrocell's flip-flop. The Preset/Reset feature for each macrocell can also be disabled. Note that the Power-on Reset leaves all macrocells in the "zero" state when power is properly applied. The other 4 control terms (CT2–CT5) can be used to control the Output Enable of the macrocell's output buffers. The reason there are as many control terms dedicated for the Output Enable of the macrocell is to insure that all CoolRunner™ devices are PCI compliant. The macrocell's output buffers can also be always enabled or disabled. All CoolRunner™ devices also provide a Global Tri-State (GTS) pin, which, when pulled Low, will 3-State all the outputs of the device. This pin is provided to support "In-Circuit Testing" or "Bed-of-Nails Testing".

There are two feedback paths to the ZIA: one from the macrocell, and one from the I/O pin. The ZIA feedback path before the output buffer is the macrocell feedback path, while the ZIA feedback path after the output buffer is the I/O pin ZIA path. When the macrocell is used as an output, the output buffer is enabled, and the macrocell feedback path can be used to feedback the logic implemented in the macrocell. When the I/O pin is used as an input, the output buffer will be 3-Stated and the input signal will be fed into the ZIA via the I/O feedback path, and the logic implemented in the buried macrocell can be fed back to the ZIA via the macrocell feedback path. It should be noted that unused inputs or I/Os should be properly terminated.

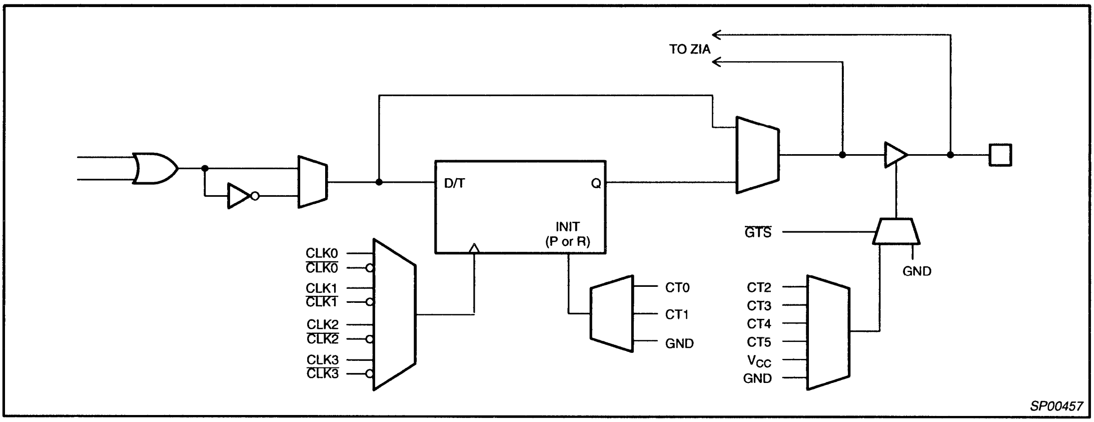


Figure 3. PZ3128 Macrocell Architecture

Simple Timing Model

Figure 4 shows the CoolRunner™ Timing Model. The CoolRunner™ timing model looks very much like a 22V10 timing model in that there are three main timing parameters, including t_{PD}, t_{SU}, and t_{CO}. In other competing architectures, the user may be able to fit the design into the CPLD, but is not sure whether system timing requirements can be met until after the design has been fit into the device. This is because the timing models of competing architectures are very complex and include such things as timing dependencies on the number of parallel expanders borrowed, sharable expanders, varying number of X and Y routing channels used, etc. In the XPLA™ architecture, the user knows up front whether the design will meet system timing requirements. This is due to the simplicity of the timing model.

TotalCMOS™ Design Technique for Fast Zero Power

Philips is the first to offer a TotalCMOS™ CPLD, both in process technology and design technique. Philips employs a cascade of CMOS gates to implement its Sum of Products instead of the traditional sense amp approach. This CMOS gate implementation allows Philips to offer CPLDs which are both high performance and low power, breaking the paradigm that to have low power, you must have low performance. Refer to Figure 5 and Table 2 showing the I_{DD} vs. Frequency of our PZ3128 TotalCMOS™ CPLD.

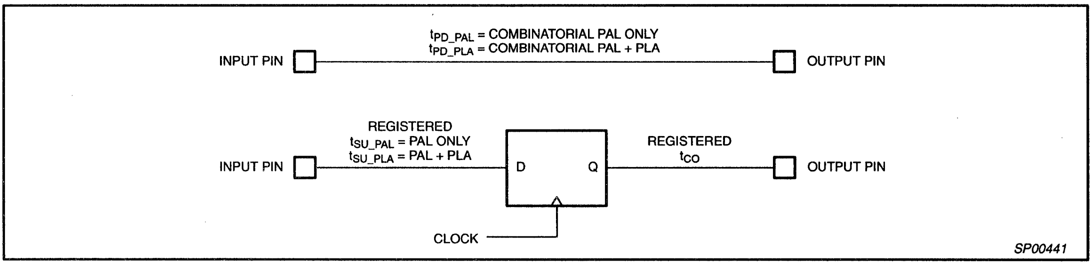


Figure 4. CoolRunner™ Timing Model

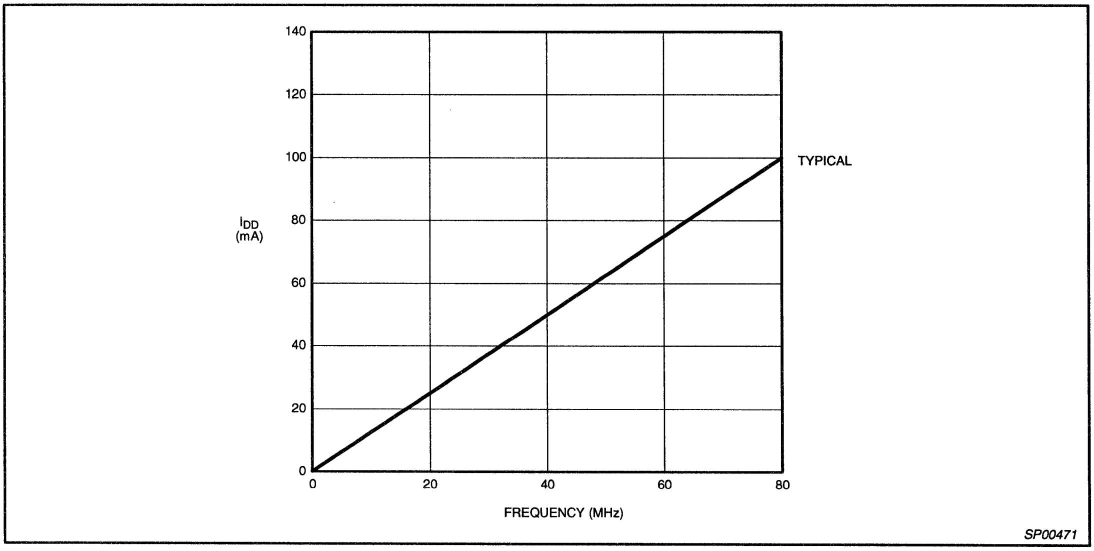


Figure 5. I_{DD} vs. Frequency @ V_{DD} = 3.3V, 25°C

Table 2. I_{DD} vs. Frequency

V_{DD} = 3.3V

FREQUENCY (MHz)	0	20	40	60	80
Typical I_{DD} (mA)	0.10	25	50	75	100

JTAG Testing Capability

JTAG is the commonly-used acronym for the Boundary Scan Test (BST) feature defined for integrated circuits by IEEE Standard 1149.1. This standard defines input/output pins, logic control functions, and commands which facilitate both board and device level testing without the use of specialized test equipment. BST provides the ability to test the external connections of a device, test the internal logic of the device, and capture data from the device during normal operation. BST provides a number of benefits in each of the following areas:

- Testability
  - Allows testing of an unlimited number of interconnects on the printed circuit board
  - Testability is designed in at the component level
  - Enables desired signal levels to be set at specific pins (Preload)
  - Data from pin or core logic signals can be examined during normal operation
- Reliability
  - Eliminates physical contacts common to existing test fixtures (e.g., “bed-of-nails”)
  - Degradation of test equipment is no longer a concern
  - Facilitates the handling of smaller, surface-mount components
  - Allows for testing when components exist on both sides of the printed circuit board
- Cost
  - Reduces/eliminates the need for expensive test equipment
  - Reduces test preparation time
  - Reduces spare board inventories

The Philips PZ3128’s JTAG interface includes a TAP Port and a TAP Controller, both of which are defined by the IEEE 1149.1 JTAG Specification. As implemented in the Philips PZ3128, the TAP Port includes four of the five pins (refer to Table 3) described in the JTAG specification: TCK, TMS, TDI, and TDO. The fifth signal defined by the JTAG specification is TRST* (Test Reset). TRST* is considered an optional signal, since it is not actually required to perform BST or ISP. The Philips PZ3128 saves an I/O pin for general purpose use by not implementing the optional TRST* signal in the JTAG interface. Instead, the Philips PZ3128 supports the test reset functionality through the use of its power up reset circuit, which is included in all Philips CPLDs. The pins associated with the power up reset circuit should connect to an external pull-up resistor to keep the JTAG signals from floating when they are not being used.

In the Philips PZ3128, the four mandatory JTAG pins each require a unique, dedicated pin on the device. However, if JTAG and ISP are not desired in the end-application, these pins may instead be used as additional general I/O pins. The decision as to whether these pins are used for JTAG/ISP or as general I/O is made when the JEDEC file is generated. If the use of JTAG/ISP is selected, the dedicated pins are not available for general purpose use. However, unlike competing CPLD’s, the Philips PZ3128 does allow the macrocell logic associated with these dedicated pins to be used as buried logic even when JTAG/ISP is selected. Table 4 defines the dedicated pins used by the four mandatory JTAG signals for each of the PZ3128 package types.

The JTAG specifications defines two sets of commands to support boundary-scan testing: high-level commands and low-level commands. High-level commands are executed via board test software on an a user test station such as automated test equipment, a PC, or an engineering workstation (EWS). Each high-level command comprises a sequence of low level commands. These low-level commands are executed within the component under test, and therefore must be implemented as part of the TAP Controller design. The set of low-level boundary-scan commands implemented in the Philips PZ3128 is defined in Table 5. By supporting this set of low-level commands, the PZ3128 allows execution of all high-level boundary-scan commands.

Table 3. JTAG Pin Description

PIN	NAME	DESCRIPTION
TCK	Test Clock Output	Clock pin to shift the serial data and instructions in and out of the TDI and TDO pins, respectively. TCK is also used to clock the TAP Controller state machine.
TMS	Test Mode Select	Serial input pin selects the JTAG instruction mode. TMS should be driven high during user mode operation.
TDI	Test Data Input	Serial input pin for instructions and test data. Data is shifted in on the rising edge of TCK.
TDO	Test Data Output	Serial output pin for instructions and test data. Data is shifted out on the falling edge of TCK. The signal is tri-stated if data is not being shifted out of the device.

Table 4. PZ3128 JTAG Pinout by Package Type

DEVICE	(PIN NUMBER / MACROCELL #)			
	TCK	TMS	TDI	TDO
PZ3128				
84-pin PLCC	62 / 96 (F15)	23 / 48 (C15)	14 / 32 (B15)	71 / 112 (G15)
100-pin PQFP	64 / 96 (F15)	17 / 48 (C15)	6 / 32 (B15)	75 / 112 (G15)
100-pin TQFP	62 / 96 (F15)	15 / 48 (C15)	4 / 32 (B15)	73 / 112 (G15)
128-pin LQFP	82 / 96 (F15)	21 / 48 (C15)	8 / 32 (B15)	95 / 112 (G15)
160-pin PQFP	99 / 96 (F15)	22 / 48 (C15)	9 / 32 (B15)	112/ 112 (G15)

Table 5. PZ3128 Low-Level JTAG Boundary-Scan Commands

INSTRUCTION **(Instruction Code)** ***Register Used***	**DESCRIPTION**
Sample/Preload (0010) *Boundary–Scan Register*	The mandatory SAMPLE/PRELOAD instruction allows a snapshot of the normal operation of the component to be taken and examined. It also allows data values to be loaded onto the latched parallel outputs of the Boundary-Scan Shift-Register prior to selection of the other boundary-scan test instructions.
Extest (0000) *Boundary-Scan Register*	The mandatory EXTEST instruction allows testing of off-chip circuitry and board level interconnections. Data would typically be loaded onto the latched parallel outputs of Boundary-Scan Shift-Register using the Sample/Preload instruction prior to selection of the EXTEST instruction.
Bypass (1111) *Bypass Register*	Places the 1 bit bypass register between the TDI and TDO pins, which allows the BST data to pass synchronously through the selected device to adjacent devices during normal device operation. The Bypass instruction can be entered by holding TDI at a constant high value and completing an Instruction-Scan cycle.
Idcode (0001) *Boundary-Scan Register*	Selects the IDCODE register and places it between TDI and TDO, allowing the IDCODE to be serially shifted out of TDO. The IDCODE instruction permits blind interrogation of the components assembled onto a printed circuit board. Thus, in circumstances where the component population may vary, it is possible to determine what components exist in a product.
HighZ (0101) *Bypass Register*	The HIGHZ instruction places the component in a state in which all of its system logic outputs are placed in an inactive drive state (e.g., high impedance). In this state, an in-circuit test system may drive signals onto the connections normally driven by a component output without incurring the risk of damage to the component. The HighZ instruction also forces the Bypass Register between TDI and TDO.

3.3-Volt, In-System Programming (ISP)

ISP is the ability to reconfigure the logic and functionality of a device, printed circuit board, or complete electronic system before, during, and after its manufacture and shipment to the end customer. ISP provides substantial benefits in each of the following areas:

- Design
  - Faster time-to-market
  - Debug partitioning and simplified prototyping
  - Printed circuit board reconfiguration during debug
  - Better device and board level testing
- Manufacturing
  - Multi-Functional hardware
  - Reconfiguarability for Test
  - Eliminates handling of “fine lead-pitch” components for programming
  - Reduced Inventory and manufacturing costs
  - Improved quality and reliability
- Field Support
  - Easy remote upgrades and repair
  - Support for field configuration, re-configuration, and customization

The Philips PZ3128 allows for 3.3-Volt, in-system programming/reprogramming of its EEPROM cells via its JTAG interface. An on-chip charge pump eliminates the need for externally-provided supervoltages, so that the PZ3128 may be easily programmed on the circuit board using only the 3.3-volt supply required by the device for normal operation. A set of low-level ISP basic commands implemented in the PZ3128 enable this feature. The ISP commands implemented in the Philips PZ3128 are specified in Table 6. Please note that an ENABLE command must precede all ISP commands **unless** an ENABLE command has already been given for a preceding ISP command **and** the device has not gone through a Test-Logic/Rest TAP Controller State.

Table 6. Low Level ISP Commands

INSTRUCTION ***(Register Used)***	**INSTRUCTION CODE**	**DESCRIPTION**
Enable *(ISP Shift Register)*	1001	Enables the Erase, Program, and Verify commands. Using the ENABLE instruction before the Erase, Program, and Verify instructions allows the user to specify the outputs the device using the JTAG Boundary–Scan SAMPLE/PRELOAD command.
Erase *(ISP Shift Register)*	1010	Erases the entire EEPROM array. The outputs during this operation can be defined by user by using the JTAG SAMPLE/PRELOAD command.
Program *(ISP Shift Register)*	1011	Programs the data in the ISP Shift Register into the addressed EEPROM row. The outputs during this operation can be defined by user by using the JTAG SAMPLE/PRELOAD command.
Verify *(ISP Shift Register)*	1100	Transfers the data from the addressed row to the ISP Shift Register. The data can then be shifted out and compared with the JEDEC file. The outputs during this operation can be defined by user by using the JTAG SAMPLE/PRELOAD command.

JTAG and ISP Interfacing

A number of industry-established methods exist for JTAG/ISP interfacing with CPLD's and other integrated circuits. The Philips PZ3128 supports the following methods:

- PC Parallel Port
- Workstation or PC Serial Port
- Embedded Processor
- Automated Test Equipment
- Third party Programmers
- High-End JTAG and ISP Tools

A Boundary-Scan Description Language (BSDL) description of the PZ3128 is also available from Philips for use in test program development. For more details on JTAG and ISP for the PZ3128, refer to the related application note: *JTAG and ISP in Philips CPLDs*.

PROGRAMMING SPECIFICATIONS

SYMBOL	PARAMETER	MIN.	MAX.	UNIT
DC Parameters				
V_{CCP}	V_{CC} supply program/verify	3.0	3.6	V
I_{CCP}	I_{CC} limit program/verify		200	mA
V_{IH}	Input voltage (High)	2.0		V
V_{IL}	Input voltage (Low)		0.8	V
V_{SOL}	Output voltage (Low)		0.5	V
V_{SOH}	Output voltage (High)	2.4		V
TDO_I_{OL}	Output current (Low)	8		mA
TDO_I_{OH}	Output current (High)	–8		mA
AC Parameters				
f_{MAX}	CLK maximum frequency	10		MHz
PWE	Pulse width erase	100		ms
PWP	Pulse width program	10		ms
PWV	Pulse width verify	10		µs
INIT	Initialization time	100		µs
TMS_SU	TMS setup time before TCK ↑	10		ns
TDI_SU	TDI setup time before TCK ↑	10		ns
TMS_H	TMS hold time after TCK ↑	25		ns
TDI_H	TDI hold time after TCK ↑	25		ns
TDO_CO	TDO valid after TCK ↓		40	ns

ABSOLUTE MAXIMUM RATINGS[1]

SYMBOL	PARAMETER	MIN.	MAX.	UNIT
V_{DD}	Supply voltage	–0.5	7.0	V
V_I	Input voltage	–1.2	V_{DD}+0.5	V
V_{OUT}	Output voltage	–0.5	V_{DD}+0.5	V
I_{IN}	Input current	–30	30	mA
I_{OUT}	Output current	–100	100	mA
T_J	Maximum junction temperature	–40	150	°C
T_{str}	Storage temperature	–65	150	°C

NOTE:

1. Stresses above those listed may cause malfunction or permanent damage to the device. This is a stress rating only. Functional operation at these or any other condition above those indicated in the operational and programming specification is not implied.

OPERATING RANGE

PRODUCT GRADE	TEMPERATURE	VOLTAGE
Commercial	0 to +70°C	3.3 ±10% V
Industrial	–40 to +85°C	3.3 ±10% V

DC ELECTRICAL CHARACTERISTICS FOR COMMERCIAL GRADE DEVICES

Commercial: 0°C ≤ T_{amb} ≤ +70°C; 3.0V ≤ V_{DD} ≤ 3.6V

SYMBOL	PARAMETER	TEST CONDITIONS	MIN.	MAX.	UNIT
V_{IL}	Input voltage low	V_{DD} = 3.0V		0.8	V
V_{IH}	Input voltage high	V_{DD} = 3.6V	2.0		V
V_I	Input clamp voltage	V_{DD} = 3.0V, I_{IN} = –18mA		–1.2	V
V_{OL}	Output voltage low	V_{DD} = 3.0V, I_{OL} = 8mA		0.5	V
V_{OH}	Output voltage high	V_{DD} = 3.0V, I_{OH} = –8mA	2.4		V
I_I	Input leakage current	V_{IN} = 0 to V_{DD}	–10	10	µA
I_{OZ}	3-Stated output leakage current	V_{IN} = 0 to V_{DD}	–10	10	µA
I_{DDQ}	Standby current	V_{DD} = 3.6V, T_{amb} = 0°C		60	µA
I_{DDD}[1]	Dynamic current	V_{DD} = 3.6V, T_{amb} = 0°C @ 1MHz		2	mA
		V_{DD} = 3.6V, T_{amb} = 0°C @ 50MHz		75	mA
I_{OS}	Short circuit output current[2]	1 pin at a time for no longer than 1 second	–50	–100	mA
C_{IN}	Input pin capacitance[2]	T_{amb} = 25°C, f = 1MHz		8	pF
C_{CLK}	Clock input capacitance[2]	T_{amb} = 25°C, f = 1MHz	5	12	pF
$C_{I/O}$	I/O pin capacitance[2]	T_{amb} = 25°C, f = 1MHz		10	pF

NOTES:

1. This parameter measured with a 16-bit, loadable up/down counter loaded into every logic block, with all outputs enabled and unloaded. Inputs are tied to V_{DD} or ground. This parameter guaranteed by design and characterization, not testing.
2. Typical values, not tested.

AC ELECTRICAL CHARACTERISTICS[1] FOR COMMERCIAL GRADE DEVICES

Commercial: 0°C ≤ T_{amb} ≤ +70°C; 3.0V ≤ V_{DD} ≤ 3.6V

SYMBOL	PARAMETER	–10		UNIT
		MIN.	MAX.	
t_{PD_PAL}	Propagation delay time, input (or feedback node) to output through PAL	2	10	ns
t_{PD_PLA}	Propagation delay time, input (or feedback node) to output through PAL & PLA	3	12.5	ns
t_{CO}	Clock to out delay time	2	7	ns
t_{SU_PAL}	Setup time (from input or feedback node) through PAL	6.5		ns
t_{SU_PLA}	Setup time (from input or feedback node) through PAL + PLA	9		ns
t_H	Hold time		0	ns
t_{CH}	Clock High time	4		ns
t_{CL}	Clock Low time	4		ns
t_R	Input Rise time		20	ns
t_F	Input Fall time		20	ns
f_{MAX1}	Maximum FF toggle rate[2] 1/(t_{CH} + t_{CL})	125		MHz
f_{MAX2}	Maximum internal frequency[2] 1/(t_{SUPAL} + t_{CF})	83		MHz
f_{MAX3}	Maximum external frequency[2] 1/(t_{SUPAL} + t_{CO})	74		MHz
t_{BUF}	Output buffer delay time		1.5	ns
t_{PDF_PAL}	Input (or feedback node) to internal feedback node delay time through PAL	2	8.5	ns
t_{PDF_PLA}	Input (or feedback node) to internal feedback node delay time through PAL+PLA	3	11	ns
t_{CF}	Clock to internal feedback node delay time		5.5	ns
t_{INIT}	Delay from valid V_{DD} to valid reset		50	µs
t_{ER}	Input to output disable[3]		13	ns
t_{EA}	Input to output valid		13	ns
t_{RP}	Input to register preset		15	ns
t_{RR}	Input to register reset		15	ns

NOTES:

1. Specifications measured with one output switching. See Figure 6 and Table 8 for derating.
2. This parameter guaranteed by design and characterization, not by test.
3. Output C_L = 5pF.

DC ELECTRICAL CHARACTERISTICS FOR INDUSTRIAL GRADE DEVICES

Industrial: $-40°C \le T_{amb} \le +85°C$; $3.0V \le V_{DD} \le 3.6V$

SYMBOL	PARAMETER	TEST CONDITIONS	MIN.	MAX.	UNIT
V_{IL}	Input voltage low	V_{DD} = 3.0V		0.8	V
V_{IH}	Input voltage high	V_{DD} = 3.6V	2.0		V
V_I	Input clamp voltage	V_{DD} = 3.0V, I_{IN} = –18mA		–1.2	V
V_{OL}	Output voltage low	V_{DD} = 3.0V, I_{OL} = 8mA		0.5	V
V_{OH}	Output voltage high	V_{DD} = 3.0V, I_{OH} = –8mA	2.4		V
I_I	Input leakage current	V_{IN} = 0 to V_{DD}	–10	10	µA
I_{OZ}	3-Stated output leakage current	V_{IN} = 0 to V_{DD}	–10	10	µA
I_{DDQ}	Standby current	V_{DD} = 3.6V, T_{amb} = –40°C		75	µA
I_{DDD}[1]	Dynamic current	V_{DD} = 3.6V, T_{amb} = –40°C @ 1MHz		2	mA
		V_{DD} = 3.6V, T_{amb} = –40°C @ 50MHz		75	mA
I_{OS}	Short circuit output current[2]	1 pin at a time for no longer than 1 second	–50	–130	mA
C_{IN}	Input pin capacitance[2]	T_{amb} = 25°C, f = 1MHz		8	pF
C_{CLK}	Clock input capacitance[2]	T_{amb} = 25°C, f = 1MHz	5	12	pF
$C_{I/O}$	I/O pin capacitance[2]	T_{amb} = 25°C, f = 1MHz		10	pF

NOTES:

1. This parameter measured with a 16–bit, loadable up/down counter loaded into every logic block, with all outputs enabled and unloaded. Inputs are tied to V_{DD} or ground. This parameter guaranteed by design and characterization, not testing.
2. Typical values, not tested.

AC ELECTRICAL CHARACTERISTICS[1] FOR INDUSTRIAL GRADE DEVICES

Industrial: $-40°C \le T_{amb} \le +85°C$; $3.0V \le V_{DD} \le 3.6V$

SYMBOL	PARAMETER	I12		UNIT
		MIN.	MAX.	
t_{PD_PAL}	Propagation delay time, input (or feedback node) to output through PAL	2	12	ns
t_{PD_PLA}	Propagation delay time, input (or feedback node) to output through PAL & PLA	3	14.5	ns
t_{CO}	Clock to out delay time	2	8	ns
t_{SU_PAL}	Setup time (from input or feedback node) through PAL	7		ns
t_{SU_PLA}	Setup time (from input or feedback node) through PAL + PLA	9.5		ns
t_H	Hold time		0	ns
t_{CH}	Clock High time	4		ns
t_{CL}	Clock Low time	4		ns
t_R	Input Rise time		20	ns
t_F	Input Fall time		20	ns
f_{MAX1}	Maximum FF toggle rate[2] $1/(t_{CH} + t_{CL})$	125		MHz
f_{MAX2}	Maximum internal frequency[2] $1/(t_{SUPAL} + t_{CF})$	77		MHz
f_{MAX3}	Maximum external frequency[2] $1/(t_{SUPAL} + t_{CO})$	66		MHz
t_{BUF}	Output buffer delay time		1.5	ns
t_{PDF_PAL}	Input (or feedback node) to internal feedback node delay time through PAL	2	10.5	ns
t_{PDF_PLA}	Input (or feedback node) to internal feedback node delay time through PAL+PLA	3	13	ns
t_{CF}	Clock to internal feedback node delay time		6	ns
t_{INIT}	Delay from valid V_{DD} to valid reset		50	µs
t_{ER}	Input to output disable[3]		14	ns
t_{EA}	Input to output valid		14	ns
t_{RP}	Input to register preset		16	ns
t_{RR}	Input to register reset		16	ns

NOTES:

1. Specifications measured with one output switching. See Figure 6 and Table 8 for derating.
2. This parameter guaranteed by design and characterization, not by test.
3. Output C_L = 5pF.

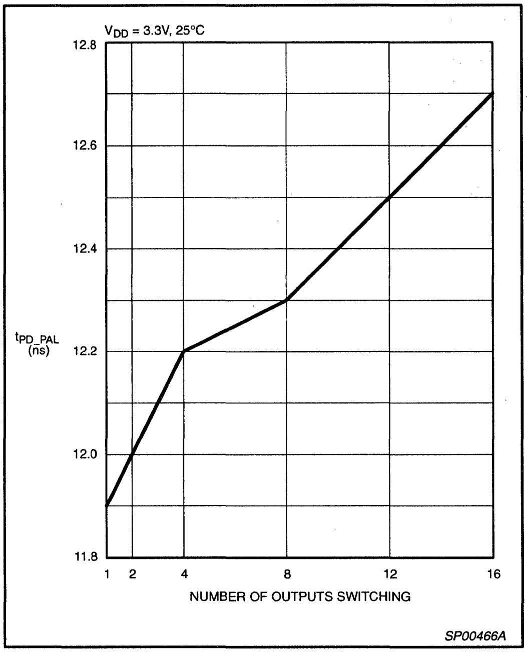


Figure 6. t_{PD_PAL} **vs. Outputs Switching**

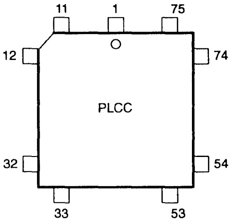
Table 7. t_{PD_PAL} vs. Number of Outputs Switching

V_{DD} = 3.3V

NUMBER OF OUTPUTS	1	2	4	8	12	16
Typical (ns)						

PIN DESCRIPTIONS

84-Pin Plastic Leaded Chip Carrier

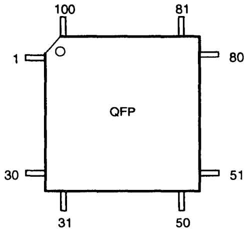


Pin	Function	Pin	Function	Pin	Function
1	IN1	29	I/O-C5	57	I/O-F7
2	IN3	30	I/O-C4	58	I/O-F10
3	V_{DD}	31	I/O-C2	59	GND
4	I/O-A15/CLK3	32	GND	60	I/O-F12
5	I/O-A13	33	I/O-D15	61	I/O-F13
6	I/O-A12	34	I/O-D12	62	I/O-F15 (TCK)
7	GND	35	I/O-D10	63	I/O-G0
8	I/O-A10	36	I/O-D8	64	I/O-G2
9	I/O-A7	37	I/O-D7	65	I/O-G4
10	I/O-A5	38	V_{DD}	66	V_{DD}
11	I/O-A4	39	I/O-D4	67	I/O-G7
12	I/O-A2	40	I/O-D2	68	I/O-G8
13	V_{DD}	41	I/O-D0/CLK2	69	I/O-G10
14	I/O-B15 (TDI)	42	GND	70	I/O-G12
15	I/O-B12	43	V_{DD}	71	I/O-G15 (TDO)
16	I/O-B10	44	I/O-E0/CLK1	72	GND
17	I/O-B8	45	I/O-E2	73	I/O-H2
18	I/O-B7	46	I/O-E4	74	I/O-H4
19	GND	47	GND	75	I/O-H5
20	I/O-B4	48	I/O-E7	76	I/O-H7
21	I/O-B2	49	I/O-E8	77	I/O-H10
22	I/O-B0	50	I/O-E10	78	V_{DD}
23	I/O-C15 (TMS)*	51	I/O-E12	79	I/O-H12
24	I/O-C13	52	I/O-E15	80	I/O-H13
25	I/O-C12	53	V_{DD}	81	I/O-H15
26	V_{DD}	54	I/O-F2	82	GND
27	I/O-C10	55	I/O-F4	83	IN0/CLK0
28	I/O-C7	56	I/O-F5	84	IN2-gtsn

* THE TEST MODE SELECT (TMS) FUNCTION IS INACTIVE ON NON-ISR ARCHITECTURES.

SP00467

100-Pin Plastic Quad Flat Package

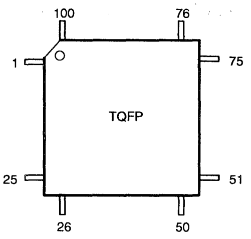


Pin	Function	Pin	Function	Pin	Function
1	I/O-A5	35	I/O-D5	69	I/O-G5
2	I/O-A4	36	V_{DD}	70	I/O-G7
3	I/O-A2	37	I/O-D4	71	I/O-G8
4	I/O-A0	38	I/O-D2	72	I/O-G10
5	V_{DD}	39	I/O-B0/CLK2	73	I/O-G12
6	I/O-B15 (TDI)	40	GND	74	I/O-G13
7	I/O-B13	41	V_{DD}	75	I/O-G15 (TDO)
8	I/O-B12	42	I/O-E0/CLK1	76	GND
9	I/O-B10	43	I/O-E2	77	I/O-H0
10	I/O-B8	44	I/O-E4	78	I/O-H2
11	I/O-B7	45	GND	79	I/O-H4
12	I/O-B5	46	I/O-E5	80	I/O-H5
13	GND	47	I/O-E7	81	I/O-H7
14	I/O-B4	48	I/O-E8	82	I/O-H8
15	I/O-B2	49	I/O-E10	83	I/O-H10
16	I/O-B0	50	I/O-E12	84	V_{DD}
17	I/O-C15 (TMS)*	51	I/O-E13	85	I/O-H12
18	I/O-C13	52	I/O-E15	86	I/O-H13
19	I/O-C12	53	V_{DD}	87	I/O-H15
20	V_{DD}	54	I/O-F0	88	GND
21	I/O-C10	55	I/O-F2	89	IN0/CLK0
22	I/O-C8	56	I/O-F4	90	IN2-gtsn
23	I/O-C7	57	I/O-F5	91	IN1
24	I/O-C5	58	I/O-F7	92	IN3
25	I/O-B9	59	I/O-F8	93	V_{DD}
26	I/O-C2	60	I/O-F10	94	I/O-A15/CLK3
27	I/O-C0	61	GND	95	I/O-A13
28	GND	62	I/O-F12	96	I/O-A12
29	I/O-D15	63	I/O-F13	97	GND
30	I/O-D13	64	I/O-F15 (TCK)	98	I/O-A10
31	I/O-D12	65	I/O-G0	99	I/O-A8
32	I/O-D10	66	I/O-G2	100	I/O-A7
33	I/O-D8	67	I/O-G4		
34	I/O-D7	68	V_{DD}		

* THE TEST MODE SELECT (TMS) FUNCTION IS INACTIVE ON NON-ISR ARCHITECTURES.

SP00468

100-Pin Thin Quad Flat Package

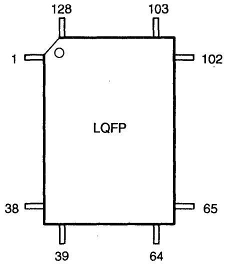


Pin	Function	Pin	Function	Pin	Function
1	I/O-A2	35	I/O-D4	69	I/O-G8
2	I/O-A0	36	I/O-D2	70	I/O-G10
3	V_{DD}	37	I/O-D0/CLK2	71	I/O-G12
4	I/O-B15 (TDI)	38	GND	72	I/O-G13
5	I/O-B13	39	V_{DD}	73	I/O-G15 (TDO)
6	I/O-B12	40	I/O-E0/CLK1	74	GND
7	I/O-B10	41	I/O-E2	75	I/O-H0
8	I/O-B8	42	I/O-E4	76	I/O-H2
9	I/O-B7	43	GND	77	I/O-H4
10	I/O-B5	44	I/O-E5	78	I/O-H5
11	GND	45	I/O-E7	79	I/O-H7
12	I/O-B4	46	I/O-E8	80	I/O-H8
13	I/O-B2	47	I/O-E10	81	I/O-H10
14	I/O-B0	48	I/O-E12	82	V_{DD}
15	I/O-C15 (TMS)*	49	I/O-E13	83	I/O-H12
16	I/O-C13	50	I/O-E15	84	I/O-H13
17	I/O-C12	51	V_{DD}	85	I/O-H15
18	V_{DD}	52	I/O-F0	86	GND
19	I/O-C10	53	I/O-F2	87	IN0/CLK0
20	I/O-C8	54	I/O-F4	88	IN2-gtsn
21	I/O-C7	55	I/O-F5	89	IN1
22	I/O-C5	56	I/O-F7	90	IN3
23	I/O-C4	57	I/O-F8	91	V_{DD}
24	I/O-C2	58	I/O-F10	92	I/O-A15/CLK3
25	I/O-C0	59	GND	93	I/O-A13
26	GND	60	I/O-F12	94	I/O-A12
27	I/O-D15	61	I/O-F13	95	GND
28	I/O-D13	62	I/O-F15 (TCK)	96	I/O-A10
29	I/O-D12	63	I/O-G0	97	I/O-A8
30	I/O-D10	64	I/O-G2	98	I/O-A7
31	I/O-D8	65	I/O-G4	99	I/O-A5
32	I/O-D7	66	V_{DD}	100	I/O-A4
33	I/O-D5	67	I/O-G5		
34	V_{DD}	68	I/O-G7		

* THE TEST MODE SELECT (TMS) FUNCTION IS INACTIVE ON NON-ISR ARCHITECTURES.

SP00485

128-Pin Low Profile Quad Flat Package

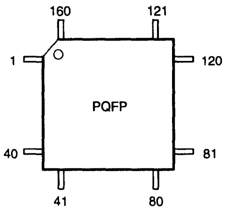


Pin	Function	Pin	Function	Pin	Function
1	I/O-A3	44	I/O-D7	87	V_{DD}
2	I/O-A2	45	I/O-D5	88	I/O-G5
3	I/O-A0	46	V_{DD}	89	I/O-G7
4	NC	47	I/O-D4	90	I/O-G8
5	NC	48	I/O-D3	91	I/O-G10
6	NC	49	I/O-D2	92	I/O-G11
7	V_{DD}	50	I/O-D0/CLK2	93	I/O-G12
8	I/O-B15 (TDI)	51	GND	94	I/O-G13
9	I/O-B13	52	V_{DD}	95	I/O-G15 (TDO)
10	I/O-B12	53	I/O-E0/CLK1	96	GND
11	I/O-B11	54	I/O-E2	97	NC
12	I/O-B10	55	I/O-E3	98	NC
13	I/O-B8	56	I/O-E4	99	NC
14	I/O-B7	57	GND	100	I/O-H0
15	I/O-B5	58	I/O-E5	101	I/O-H2
16	GND	59	I/O-E7	102	I/O-H3
17	I/O-B4	60	I/O-E8	103	I/O-H4
18	I/O-B3	61	I/O-E10	104	I/O-H5
19	I/O-B2	62	I/O-E11	105	I/O-H7
20	I/O-B0	63	I/O-E12	106	I/O-H8
21	I/O-C15 (TMS)*	64	I/O-E13	107	I/O-H10
22	I/O-C13	65	I/O-E15	108	V_{DD}
23	I/O-C12	66	V_{DD}	109	I/O-H11
24	I/O-C11	67	I/O-F0	110	I/O-H12
25	V_{DD}	68	NC	111	I/O-H13
26	I/O-C10	69	NC	112	I/O-H15
27	I/O-C8	70	NC	113	GND
28	I/O-C7	71	I/O-F2	114	IN0/CLK0
29	I/O-C5	72	I/O-F3	115	IN2-gtsn
30	I/O-C4	73	I/O-F4	116	IN1
31	I/O-C3	74	I/O-F5	117	IN3
32	I/O-C2	75	I/O-F7	118	V_{DD}
33	NC	76	I/O-F8	119	I/O-A15/CLK3
34	NC	77	I/O-F10	120	I/O-A13
35	NC	78	GND	121	I/O-A12
36	I/O-C0	79	I/O-F11	122	I/O-A11
37	GND	80	I/O-F12	123	GND
38	I/O-D15	81	I/O-F13	124	I/O-A10
39	I/O-D13	82	I/O-F15(TCK)	125	I/O-A8
40	I/O-D12	83	I/O-G0	126	I/O-A7
41	I/O-D11	84	I/O-G2	127	I/O-A5
42	I/O-D10	85	I/O-G3	128	I/O-A4
43	I/O-D8	86	I/O-G4		

* THE TEST MODE SELECT (TMS) FUNCTION IS INACTIVE ON NON-ISR ARCHITECTURES.

SP00469A

160-Pin Plastic Quad Flat Package



Pin	Function	Pin	Function	Pin	Function
1	NC	54	I/O-D5	107	I/O-G8
2	NC	55	V_{DD}	108	I/O-G10
3	NC	56	I/O-D4	109	I/O-G11
4	NC	57	I/O-D3	110	I/O-G12
5	NC	58	I/O-D2	111	I/O-G13
6	NC	59	I/O-D0/CLK2	112	I/O-G15 (TDO)
7	NC	60	GND	113	GND
8	V_{DD}	61	V_{DD}	114	NC
9	I/O-B15 (TDI)	62	I/O-E0/CLK1	115	NC
10	I/O-B13	63	I/O-E2	116	NC
11	I/O-B12	64	I/O-E3	117	NC
12	I/O-B11	65	I/O-E4	118	NC
13	I/O-B10	66	GND	119	NC
14	I/O-B8	67	I/O-E5	120	NC
15	I/O-B7	68	I/O-E7	121	I/O-H0
16	I/O-B5	69	I/O-E8	122	I/O-H2
17	GND	70	I/O-E10	123	I/O-H3
18	I/O-B4	71	I/O-E11	124	NC
19	I/O-B3	72	I/O-E12	125	NC
20	I/O-B2	73	I/O-E13	126	NC
21	I/O-B0	74	NC	127	NC
22	I/O-C15 (TMS)	75	NC	128	I/O-H4
23	I/O-C13	76	NC	129	I/O-H5
24	I/O-C12	77	NC	130	I/O-H7
25	I/O-C11	78	I/O-E15	131	I/O-H8
26	V_{DD}	79	V_{DD}	132	I/O-H10
27	I/O-C10	80	I/O-F0	133	V_{DD}
28	I/O-C8	81	NC	134	I/O-H11
29	I/O-C7	82	NC	135	I/O-H12
30	I/O-C5	83	NC	136	I/O-H13
31	I/O-C4	84	NC	137	I/O-H15
32	I/O-C3	85	NC	138	GND
33	I/O-C2	86	NC	139	IN0/CLK0
34	NC	87	NC	140	IN2-gtsn
35	NC	88	I/O-F2	141	IN1
36	NC	89	I/O-F3	142	IN3
37	NC	90	I/O-F4	143	V_{DD}
38	NC	91	I/O-F5	144	I/O-A15/CLK3
39	NC	92	I/O-F7	145	I/O-A13
40	NC	93	I/O-F8	146	I/O-A12
41	I/O-C0	94	I/O-F10	147	I/O-A11
42	GND	95	GND	148	GND
43	I/O-D15	96	I/O-F11	149	I/O-A10
44	NC	97	I/O-F12	150	I/O-A8
45	NC	98	I/O-F13	151	I/O-A7
46	NC	99	I/O-F15 (TCK)	152	I/O-A5
47	NC	100	I/O-G0	153	I/O-A4
48	I/O-D13	101	I/O-G2	154	NC
49	I/O-D12	102	I/O-G3	155	NC
50	I/O-D11	103	I/O-G4	156	NC
51	I/O-D10	104	V_{DD}	157	NC
52	I/O-D8	105	I/O-G5	158	I/O-A3
53	I/O-D7	106	I/O-G7	159	I/O-A2
				160	I/O-A0

* THE TEST MODE SELECT (TMS) FUNCTION IS INACTIVE ON NON-ISR ARCHITECTURES.

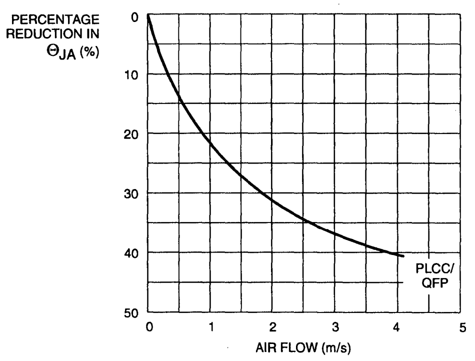
SP00470A

Package Thermal Characteristics

Philips Semiconductors uses the Temperature Sensitive Parameter (TSP) method to test thermal resistance. This method meets Mil-Std-883C Method 1012.1 and is described in Philips *1995 IC Package Databook*. Thermal resistance varies slightly as a function of input power. As input power increases, thermal resistance changes approximately 5% for a 100% change in power.

Figure 7 is a derating curve for the change in Θ_{JA} with airflow based on wind tunnel measurements. It should be noted that the wind flow dynamics are more complex and turbulent in actual applications than in a wind tunnel. Also, the test boards used in the wind tunnel contribute significantly to forced convection heat transfer, and may not be similar to the actual circuit board, especially in size.

Package	Θ_{JA}
84-pin PLCC	32.8°C/W
100-pin PQFP	41.2°C/W
100-pin TQFP	47.4°C/W
128-pin LQFP	45.0°C/W
160-pin PQFP	31.9°C/W



SP00419A

Figure 7. Average Effect of Airflow on Θ_{JA}

128 macrocell CPLD PZ5128–7

FEATURES

- Industry's first TotalCMOS™ PLD – both CMOS design and process technologies
- Fast Zero Power (FZP™) design technique provides ultra-low power and very high speed
- IEEE 1149.1–compliant, JTAG Testing Capability
  - 4 pin JTAG interface (TCK, TMS, TDI, TDO)
  - IEEE 1149.1 TAP Controller
  - JTAG commands include: Bypass, Sample/Preload, Extest, Usercode, Idcode, HighZ
- 5 Volt, In–System Programmable (ISP) using the JTAG interface
  - On–chip supervoltage generation
  - ISP commands include: Enable, Erase, Program, Verify
  - Supported by multiple ISP programming platforms
- High speed pin-to-pin delays of 7.5ns
- Ultra-low static power of less than 100µA
- Dynamic power that is 70% lower at 50MHz than competing devices
- 100% routable with 100% utilization while all pins and all macrocells are fixed
- Deterministic timing model that is extremely simple to use
- 4 clocks with programmable polarity at every macrocell
- Support for complex asynchronous clocking
- Innovative XPLA™ architecture combines high speed with extreme flexibility
- 1000 erase/program cycles guaranteed
- 20 years data retention guaranteed
- Logic expandable to 37 product terms
- PCI compliant
- Advanced 0.5µ E^{2}CMOS process
- Security bit prevents unauthorized access
- Design entry and verification using industry standard and Philips CAE tools
- Reprogrammable using industry standard device programmers
- Innovative Control Term structure provides either sum terms or product terms in each logic block for:
  - Programmable 3-State buffer
  - Asynchronous macrocell register preset/reset
- Programmable global 3-State pin facilitates 'bed of nails' testing without using logic resources
- Available in PLCC, TQFP, and PQFP packages
- Available in both Commercial and Industrial grades

Table 1. PZ5128 Features

	PZ5128
Usable gates	4000
Maximum inputs	100
Maximum I/Os	96
Number of macrocells	128
Propagation delay (ns)	7.5
Packages	84-pin PLCC, 100-pin PQFP, 100-pin TQFP 128-pin LQFP, 160-pin PQFP

DESCRIPTION

The PZ5128 CPLD (Complex Programmable Logic Device) is the third in a family of Fast Zero Power (FZP™) CPLDs from Philips Semiconductors. These devices combine high speed and zero power in a 128 macrocell CPLD. With the FZP™ design technique, the PZ5128 offers true pin-to-pin speeds of 7.5ns, while simultaneously delivering power that is less than 100µA at standby without the need for 'turbo bits' or other power down schemes. By replacing conventional sense amplifier methods for implementing product terms (a technique that has been used in PLDs since the bipolar era) with a cascaded chain of pure CMOS gates, the dynamic power is also substantially lower than any competing CPLD – 70% lower at 50MHz. These devices are the first TotalCMOS™ PLDs, as they use both a CMOS process technology **and** the patented full CMOS FZP™ design technique. For 3V applications, Philips also offers the high speed PZ3128 CPLD that offers these features in a full 3V implementation.

The Philips FZP™ CPLDs introduce the new patent-pending XPLA™ (eXtended Programmable Logic Array) architecture. The XPLA™ architecture combines the best features of both PLA and PAL™ type structures to deliver high speed and flexible logic allocation that results in superior ability to make design changes with fixed pinouts. The XPLA™ structure in each logic block provides a fast 7.5ns PAL™ path with 5 dedicated product terms per output. This PAL™ path is joined by an additional PLA structure that deploys a pool of 32 product terms to a fully programmable OR array that can allocate the PLA product terms to any output in the logic block. This combination allows logic to be allocated efficiently throughout the logic block and supports as many as 37 product terms on an output. The speed with which logic is allocated from the PLA array to an output is only 2ns, regardless of the number of PLA product terms used, which results in worst case t_{PD}'s of only 9.5ns from any pin to any other pin. In addition, logic that is common to multiple outputs can be placed on a single PLA product term and shared across multiple outputs via the OR array, effectively increasing design density.

The PZ5128 CPLDs are supported by industry standard CAE tools (Cadence, Mentor, Synopsys, Synario, Viewlogic, MINC), using text (Abel, VHDL, Verilog) and/or schematic entry. Design verification uses industry standard simulators for functional and timing simulation. Development is supported on personal computer, Sparc, and HP platforms. Device fitting uses either MINC or Philips Semiconductors-developed tools.

The PZ5128 CPLD is electrically reprogrammable using industry standard device programmers from vendors such as Data I/O, BP Microsystems, SMS, and others. The PZ5128 also includes an industry-standard, IEEE 1149.1, JTAG interface through which in-system programming (ISP) and reprogramming of the device is supported.

PAL is a registered trademark of Advanced Micro Devices, Inc.

ORDERING INFORMATION

ORDER CODE	DESCRIPTION	DESCRIPTION	DRAWING NUMBER
PZ5128-S7A84	84-pin PLCC, 7.5ns t_{PD}	Commercial temp range, 5 volt power supply, ± 5%	SOT189-3
PZ5128IS10A84	84-pin PLCC, 10ns t_{PD}	Industrial temp range, 5 volt power supply, ± 10%	SOT189-3
PZ5128-S7BB1	100-pin PQFP, 7.5ns t_{PD}	Commercial temp range, 5 volt power supply, ± 5%	SOT382-1
PZ5128IS10BB1	100-pin PQFP, 10ns t_{PD}	Industrial temp range, 5 volt power supply, ± 10%	SOT382-1
PZ5128-S7BBP	100-pin TQFP, 7.5ns t_{PD}	Commercial temp range, 5 volt power supply, ± 5%	SOT386-1
PZ5128IS10BP	100-pin TQFP, 10ns t_{PD}	Industrial temp range, 5 volt power supply, ± 10%	SOT386-1
PZ5128-S7BE	128-pin LQFP, 7.5ns t_{PD}	Commercial temp range, 5 volt power supply, ± 5%	SOT425-1
PZ5128IS10BE	128-pin LQFP, 10ns t_{PD}	Industrial temp range, 5 volt power supply, ± 10%	SOT425-1
PZ5128-S7BB2	160-pin PQFP, 7.5ns t_{PD}	Commercial temp range, 5 volt power supply, ± 5%	SOT322-2
PZ5128IS10BB2	160-pin PQFP, 10ns t_{PD}	Industrial temp range, 5 volt power supply, ± 10%	SOT322-2

XPLA™ ARCHITECTURE

Figure 1 shows a high level block diagram of a 128 macrocell device implementing the XPLA™ architecture. The XPLA™ architecture consists of logic blocks that are interconnected by a Zero-power Interconnect Array (ZIA). The ZIA is a virtual crosspoint switch. Each logic block is essentially a 36V16 device with 36 inputs from the ZIA and 16 macrocells. Each logic block also provides 32 ZIA feedback paths from the macrocells and I/O pins.

From this point of view, this architecture looks like many other CPLD architectures. What makes the CoolRunner™ family unique is what is inside each logic block and the design technique used to implement these logic blocks. The contents of the logic block will be described next.

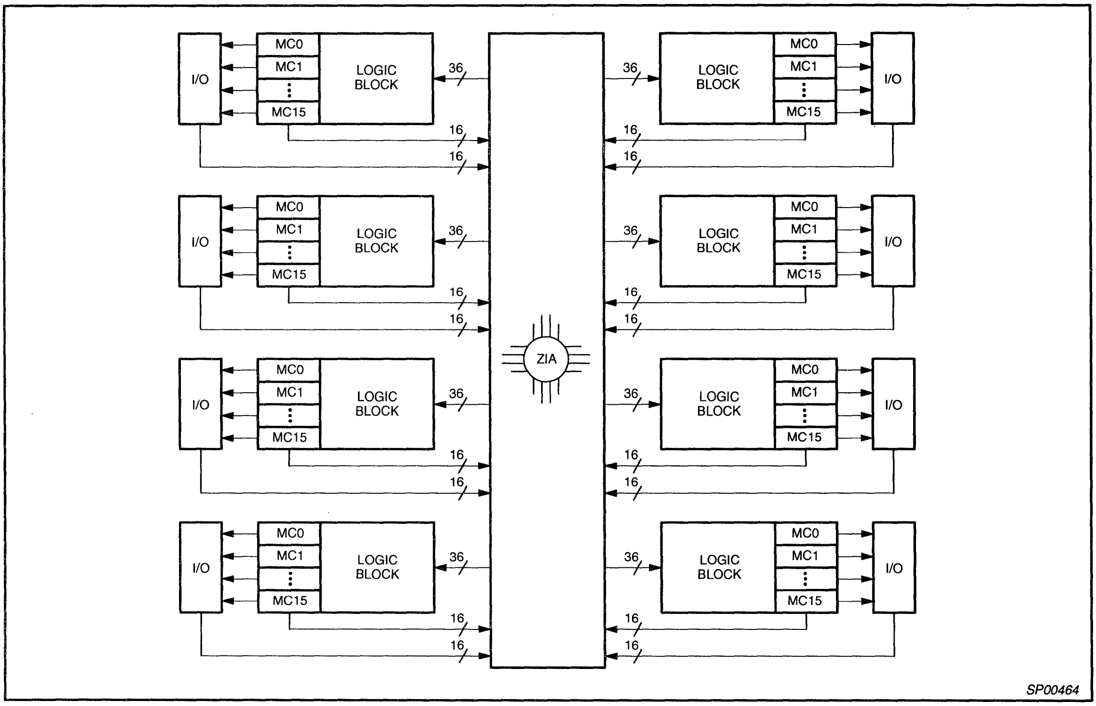


Figure 1. Philips XPLA CPLD Architecture

Logic Block Architecture

Figure 2 illustrates the logic block architecture. Each logic block contains control terms, a PAL array, a PLA array, and 16 macrocells. the 6 control terms can individually be configured as either SUM or PRODUCT terms, and are used to control the preset/reset and output enables of the 16 macrocells' flip-flops. The PAL array consists of a programmable AND array with a fixed OR array, while the PLA array consists of a programmable AND array with a programmable OR array. The PAL array provides a high speed path through the array, while the PLA array provides increased product term density.

Each macrocell has 5 dedicated product terms from the PAL array. The pin-to-pin t_{PD} of the PZ5128 device through the PAL array is 7.5ns. If a macrocell needs more than 5 product terms, it simply gets the additional product terms from the PLA array. The PLA array consists of 32 product terms, which are available for use by all 16 macrocells. The additional propagation delay incurred by a macrocell using 1 or all 32 PLA product terms is just 2ns. So the total pin-to-pin t_{PD} for the PZ5128 using 6 to 37 product terms is 9.5ns (7.5ns for the PAL + 2ns for the PLA).

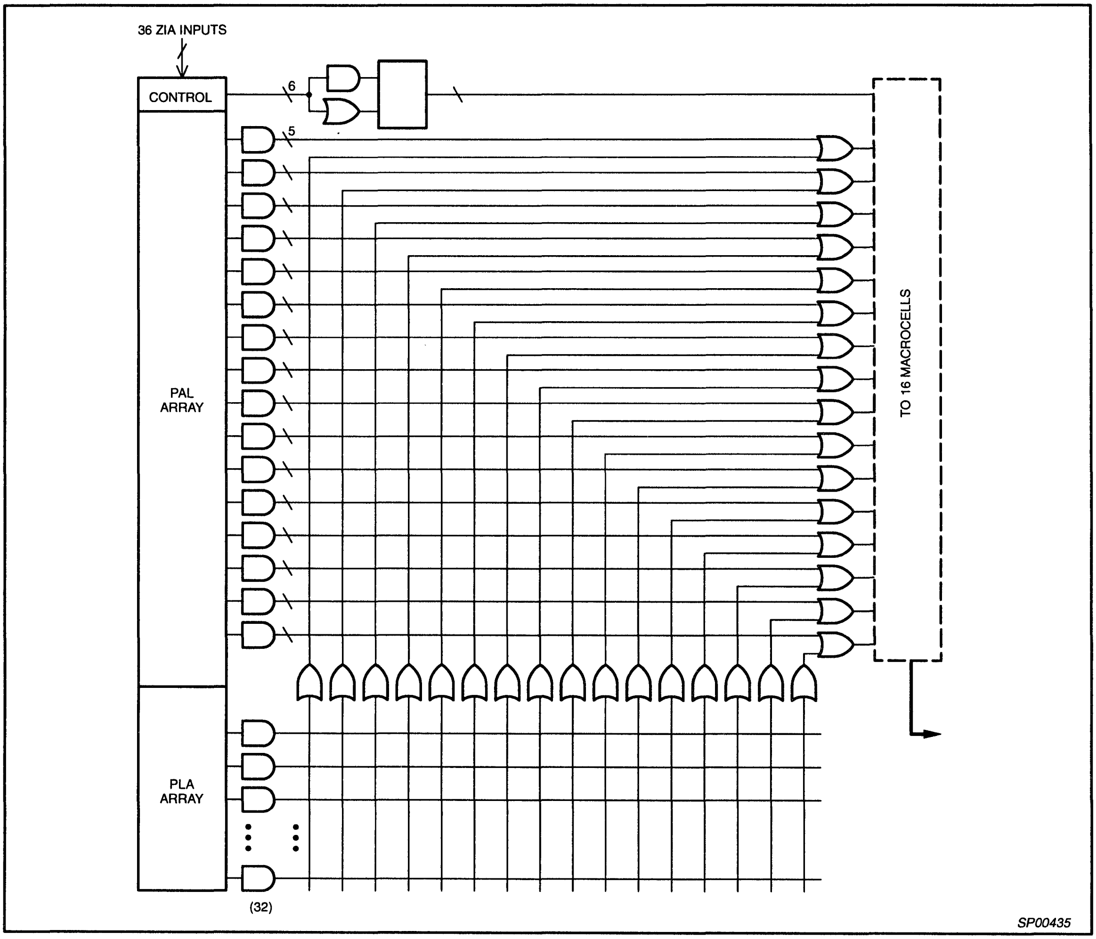


Figure 2. Philips Logic Block Architecture

Macrocell Architecture

Figure 3 shows the architecture of the macrocell used in the CoolRunner™ family. The macrocell consists of a flip-flop that can be configured as either a D or T type. A D-type flip-flop is generally more useful for implementing state machines and data buffering. A T-type flip-flop is generally more useful in implementing counters. All CoolRunner™ family members provide both synchronous and asynchronous clocking and provide the ability to clock off either the falling or rising edges of these clocks. These devices are designed such that the skew between the rising and falling edges of a clock are minimized for clocking integrity. There are 4 clocks available on the PZ5128 device. Clock 0 (CLK0) is designated as the "synchronous" clock and must be driven by an external source. Clock 1 (CLK1), Clock 2 (CLK2), and Clock 3 (CLK3) can either be used as a synchronous clock (driven by an external source) or as an asynchronous clock (driven by a macrocell equation).

Two of the control terms (CT0 and CT1) are used to control the Preset/Reset of the macrocell's flip-flop. The Preset/Reset feature for each macrocell can also be disabled. Note that the Power-on Reset leaves all macrocells in the "zero" state when power is properly applied. The other 4 control terms (CT2–CT5) can be used to control the Output Enable of the macrocell's output buffers. The reason there are as many control terms dedicated for the Output Enable of the macrocell is to insure that all CoolRunner™ devices are PCI compliant. The macrocell's output buffers can also be always enabled or disabled. All CoolRunner™ devices also provide a Global Tri-State ($\overline{\text{GTS}}$) pin, which, when pulled Low, will 3-State all the outputs of the device. This pin is provided to support "In-Circuit Testing" or "Bed-of-Nails Testing".

There are two feedback paths to the ZIA: one from the macrocell, and one from the I/O pin. The ZIA feedback path before the output buffer is the macrocell feedback path, while the ZIA feedback path after the output buffer is the I/O pin ZIA path. When the macrocell is used as an output, the output buffer is enabled, and the macrocell feedback path can be used to feedback the logic implemented in the macrocell. When the I/O pin is used as an input, the output buffer will be 3-Stated and the input signal will be fed into the ZIA via the I/O feedback path, and the logic implemented in the buried macrocell can be fed back to the ZIA via the macrocell feedback path. It should be noted that unused inputs or I/Os should be properly terminated.

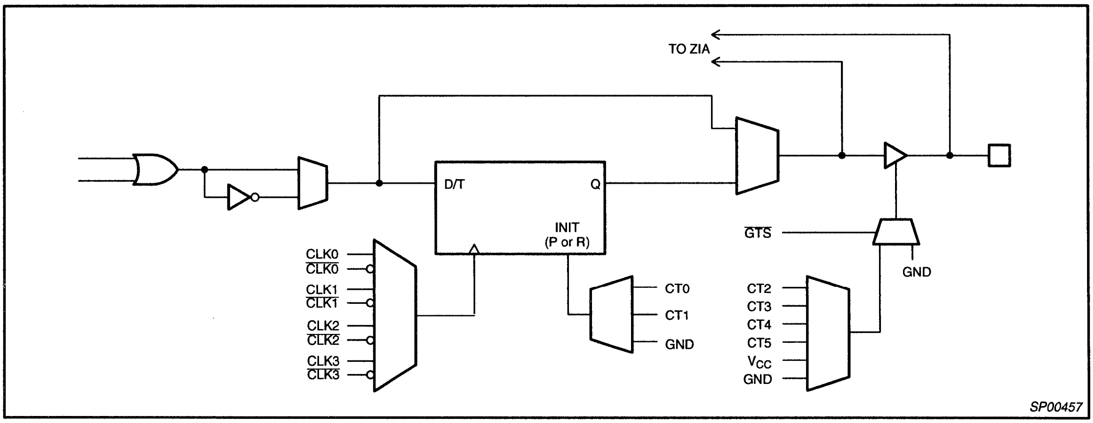


Figure 3. PZ5128 Macrocell Architecture

Simple Timing Model

Figure 4 shows the CoolRunner™ Timing Model. The CoolRunner™ timing model looks very much like a 22V10 timing model in that there are three main timing parameters, including t_{PD}, t_{SU}, and t_{CO}. In other competing architectures, the user may be able to fit the design into the CPLD, but is not sure whether system timing requirements can be met until after the design has been fit into the device. This is because the timing models of competing architectures are very complex and include such things as timing dependencies on the number of parallel expanders borrowed, sharable expanders, varying number of X and Y routing channels used, etc. In the XPLA™ architecture, the user knows up front whether the design will meet system timing requirements. This is due to the simplicity of the timing model.

TotalCMOS™ Design Technique for Fast Zero Power

Philips is the first to offer a TotalCMOS™ CPLD, both in process technology and design technique. Philips employs a cascade of CMOS gates to implement its Sum of Products instead of the traditional sense amp approach. This CMOS gate implementation allows Philips to offer CPLDs which are both high performance and low power, breaking the paradigm that to have low power, you must have low performance. Refer to Figure 5 and Table 2 showing the I_{DD} vs. Frequency of our PZ5128 TotalCMOS™ CPLD.

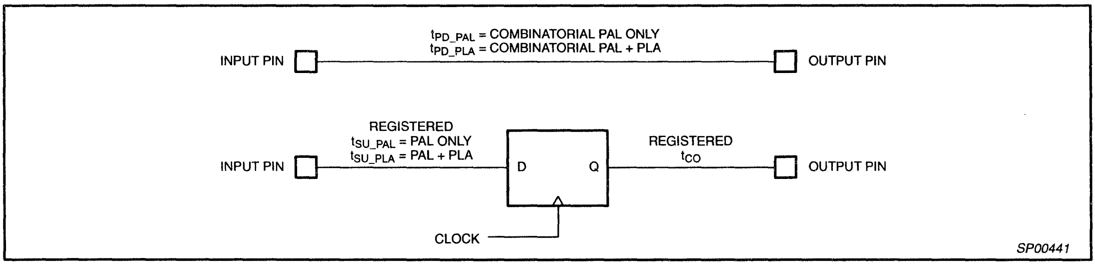


Figure 4. CoolRunner™ Timing Model

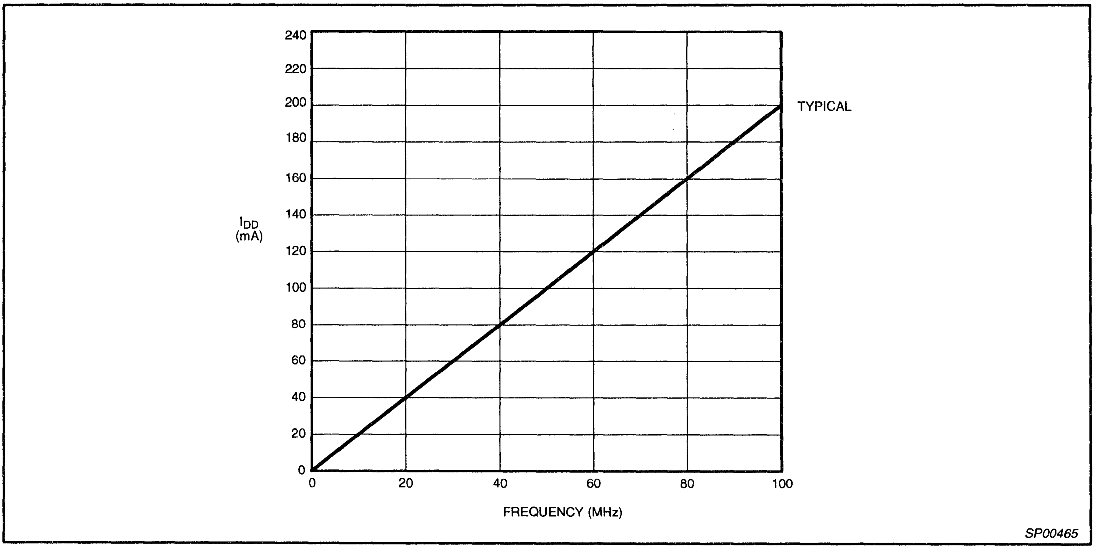


Figure 5. I_{DD} vs. Frequency @ V_{DD} = 5.0V, 25°C

Table 2. I_{DD} vs. Frequency

V_{DD} = 5.00V

FREQUENCY (MHz)	0	20	40	60	80	100
Typical I_{DD} (mA)	0.10	40	80	120	160	200

JTAG Testing Capability

JTAG is the commonly-used acronym for the Boundary Scan Test (BST) feature defined for integrated circuits by IEEE Standard 1149.1. This standard defines input/output pins, logic control functions, and commands which facilitate both board and device level testing without the use of specialized test equipment. BST provides the ability to test the external connections of a device, test the internal logic of the device, and capture data from the device during normal operation. BST provides a number of benefits in each of the following areas:

- Testability
  - Allows testing of an unlimited number of interconnects on the printed circuit board
  - Testability is designed in at the component level
  - Enables desired signal levels to be set at specific pins (Preload)
  - Data from pin or core logic signals can be examined during normal operation
- Reliability
  - Eliminates physical contacts common to existing test fixtures (e.g., “bed-of-nails”)
  - Degradation of test equipment is no longer a concern
  - Facilitates the handling of smaller, surface-mount components
  - Allows for testing when components exist on both sides of the printed circuit board
- Cost
  - Reduces/eliminates the need for expensive test equipment
  - Reduces test preparation time
  - Reduces spare board inventories

The Philips PZ5128's JTAG interface includes a TAP Port and a TAP Controller, both of which are defined by the IEEE 1149.1 JTAG Specification. As implemented in the Philips PZ5128, the TAP Port includes four of the five pins (refer to Table 3) described in the JTAG specification: TCK, TMS, TDI, and TDO. The fifth signal defined by the JTAG specification is TRST* (Test Reset). TRST* is considered an optional signal, since it is not actually required to perform BST or ISP. The Philips PZ5128 saves an I/O pin for general purpose use by not implementing the optional TRST* signal in the JTAG interface. Instead, the Philips PZ5128 supports the test reset functionality through the use of its power up reset circuit, which is included in all Philips CPLDs. The pins associated with the power up reset circuit should connect to an external pull-up resistor to keep the JTAG signals from floating when they are not being used.

In the Philips PZ5128, the four mandatory JTAG pins each require a unique, dedicated pin on the device. However, if JTAG and ISP are not desired in the end-application, these pins may instead be used as additional general I/O pins. The decision as to whether these pins are used for JTAG/ISP or as general I/O is made when the JEDEC file is generated. If the use of JTAG/ISP is selected, the dedicated pins are not available for general purpose use. However, unlike competing CPLD's, the Philips PZ5128 does allow the macrocell logic associated with these dedicated pins to be used as buried logic even when JTAG/ISP is selected. Table 4 defines the dedicated pins used by the four mandatory JTAG signals for each of the PZ5128 package types.

The JTAG specifications defines two sets of commands to support boundary-scan testing: high-level commands and low-level commands. High-level commands are executed via board test software on an a user test station such as automated test equipment, a PC, or an engineering workstation (EWS). Each high-level command comprises a sequence of low level commands. These low-level commands are executed within the component under test, and therefore must be implemented as part of the TAP Controller design. The set of low-level boundary-scan commands implemented in the Philips PZ5128 is defined in Table 5. By supporting this set of low-level commands, the PZ5128 allows execution of all high-level boundary-scan commands.

Table 3. JTAG Pin Description

PIN	NAME	DESCRIPTION
TCK	Test Clock Output	Clock pin to shift the serial data and instructions in and out of the TDI and TDO pins, respectively. TCK is also used to clock the TAP Controller state machine.
TMS	Test Mode Select	Serial input pin selects the JTAG instruction mode. TMS should be driven high during user mode operation.
TDI	Test Data Input	Serial input pin for instructions and test data. Data is shifted in on the rising edge of TCK.
TDO	Test Data Output	Serial output pin for instructions and test data. Data is shifted out on the falling edge of TCK. The signal is tri-stated if data is not being shifted out of the device.

Table 4. PZ5128 JTAG Pinout by Package Type

DEVICE	(PIN NUMBER / MACROCELL #)			
	TCK	TMS	TDI	TDO
PZ5128				
84-pin PLCC	62 / 96 (F15)	23 / 48 (C15)	14 / 32 (B15)	71 / 112 (G15)
100-pin PQFP	64 / 96 (F15)	17 / 48 (C15)	6 / 32 (B15)	75 / 112 (G15)
100-pin TQFP	62 / 96 (F15)	15 / 48 (C15)	4 / 32 (B15)	73 / 112 (G15)
128-pin LQFP	82 / 96 (F15)	21 / 48 (C15)	8 / 32 (B15)	95 / 112 (G15)
160-pin PQFP	99 / 96 (F15)	22 / 48 (C15)	9 / 32 (B15)	112/ 112 (G15)

Table 5. PZ5128 Low-Level JTAG Boundary-Scan Commands

INSTRUCTION **(Instruction Code)** ***Register Used***	**DESCRIPTION**
Sample/Preload (0010) *Boundary–Scan Register*	The mandatory SAMPLE/PRELOAD instruction allows a snapshot of the normal operation of the component to be taken and examined. It also allows data values to be loaded onto the latched parallel outputs of the Boundary-Scan Shift-Register prior to selection of the other boundary-scan test instructions.
Extest (0000) *Boundary-Scan Register*	The mandatory EXTEST instruction allows testing of off-chip circuitry and board level interconnections. Data would typically be loaded onto the latched parallel outputs of Boundary-Scan Shift-Register using the Sample/Preload instruction prior to selection of the EXTEST instruction.
Bypass (1111) *Bypass Register*	Places the 1 bit bypass register between the TDI and TDO pins, which allows the BST data to pass synchronously through the selected device to adjacent devices during normal device operation. The Bypass instruction can be entered by holding TDI at a constant high value and completing an Instruction-Scan cycle.
Idcode (0001) *Boundary-Scan Register*	Selects the IDCODE register and places it between TDI and TDO, allowing the IDCODE to be serially shifted out of TDO. The IDCODE instruction permits blind interrogation of the components assembled onto a printed circuit board. Thus, in circumstances where the component population may vary, it is possible to determine what components exist in a product.
HighZ (0101) *Bypass Register*	The HIGHZ instruction places the component in a state in which all of its system logic outputs are placed in an inactive drive state (e.g., high impedance). In this state, an in-circuit test system may drive signals onto the connections normally driven by a component output without incurring the risk of damage to the component. The HighZ instruction also forces the Bypass Register between TDI and TDO.

5-Volt, In-System Programming (ISP)

ISP is the ability to reconfigure the logic and functionality of a device, printed circuit board, or complete electronic system before, during, and after its manufacture and shipment to the end customer. ISP provides substantial benefits in each of the following areas:

- Design
  - Faster time-to-market
  - Debug partitioning and simplified prototyping
  - Printed circuit board reconfiguration during debug
  - Better device and board level testing
- Manufacturing
  - Multi-Functional hardware
  - Reconfiguarability for Test
  - Eliminates handling of "fine lead-pitch" components for programming
  - Reduced Inventory and manufacturing costs
  - Improved quality and reliability
- Field Support
  - Easy remote upgrades and repair
  - Support for field configuration, re-configuration, and customization

The Philips PZ5128 allows for 5-Volt, in-system programming/reprogramming of its EEPROM cells via its JTAG interface. An on-chip charge pump eliminates the need for externally-provided supervoltages, so that the PZ5128 may be easily programmed on the circuit board using only the 5-volt supply required by the device for normal operation. A set of low-level ISP basic commands implemented in the PZ5128 enable this feature. The ISP commands implemented in the Philips PZ5128 are specified in Table 6. Please note that an ENABLE command must precede all ISP commands **unless** an ENABLE command has already been given for a preceding ISP command **and** the device has not gone through a Test-Logic/Rest TAP Controller State.

Table 6. Low Level ISP Commands

INSTRUCTION ***(Register Used)***	**INSTRUCTION CODE**	**DESCRIPTION**
Enable *(ISP Shift Register)*	1001	Enables the Erase, Program, and Verify commands. Using the ENABLE instruction before the Erase, Program, and Verify instructions allows the user to specify the outputs the device using the JTAG Boundary–Scan SAMPLE/PRELOAD command.
Erase *(ISP Shift Register)*	1010	Erases the entire EEPROM array. The outputs during this operation can be defined by user by using the JTAG SAMPLE/PRELOAD command.
Program *(ISP Shift Register)*	1011	Programs the data in the ISP Shift Register into the addressed EEPROM row. The outputs during this operation can be defined by user by using the JTAG SAMPLE/PRELOAD command.
Verify *(ISP Shift Register)*	1100	Transfers the data from the addressed row to the ISP Shift Register. The data can then be shifted out and compared with the JEDEC file. The outputs during this operation can be defined by user by using the JTAG SAMPLE/PRELOAD command.

JTAG and ISP Interfacing

A number of industry-established methods exist for JTAG/ISP interfacing with CPLD's and other integrated circuits. The Philips PZ5128 supports the following methods:

- PC Parallel Port
- Workstation or PC Serial Port
- Embedded Processor
- Automated Test Equipment
- Third party Programmers
- High-End JTAG and ISP Tools

A Boundary-Scan Description Language (BSDL) description of the PZ5128 is also available from Philips for use in test program development. For more details on JTAG and ISP for the PZ5128, refer to the related application note: *JTAG and ISP in Philips CPLDs.*

Programming Specifications

SYMBOL	PARAMETER	MIN.	MAX.	UNIT
DC Parameters				
V_{CCP}	V_{CC} supply program/verify	4.5	5.5	V
I_{CCP}	I_{CC} limit program/verify		200	mA
V_{IH}	Input voltage (High)	2.0		V
V_{IL}	Input voltage (Low)		0.8	V
V_{SOL}	Output voltage (Low)		0.5	V
V_{SOH}	Output voltage (High)	2.4		V
TDO_I_{OL}	Output current (Low)	12		mA
TDO_I_{OH}	Output current (High)	–12		mA
AC Parameters				
f_{MAX}	CLK maximum frequency	10		MHz
PWE	Pulse width erase	100		ms
PWP	Pulse width program	10		ms
PWV	Pulse width verify	10		µs
INIT	Initialization time	100		µs
TMS_SU	TMS setup time before TCK ↑	10		ns
TDI_SU	TDI setup time before TCK ↑	10		ns
TMS_H	TMS hold time after TCK ↑	20		ns
TDI_H	TDI hold time after TCK ↑	20		ns
TDO_CO	TDO valid after TCK ↓		30	ns

ABSOLUTE MAXIMUM RATINGS[1]

SYMBOL	PARAMETER	MIN.	MAX.	UNIT
V_{DD}	Supply voltage	–0.5	7.0	V
V_I	Input voltage	–1.2	V_{DD}+0.5	V
V_{OUT}	Output voltage	–0.5	V_{DD}+0.5	V
I_{IN}	Input current	–30	30	mA
I_{OUT}	Output current	–100	100	mA
T_J	Maximum junction temperature	–40	150	°C
T_{str}	Storage temperature	–65	150	°C

NOTE:
1. Stresses above those listed may cause malfunction or permanent damage to the device. This is a stress rating only. Functional operation at these or any other condition above those indicated in the operational and programming specification is not implied.

OPERATING RANGE

PRODUCT GRADE	TEMPERATURE	VOLTAGE
Commercial	0 to +70°C	5.0 ±5% V
Industrial	–40 to +85°C	5.0 ±10% V

DC ELECTRICAL CHARACTERISTICS FOR COMMERCIAL GRADE DEVICES

Commercial: 0°C ≤ T_{amb} ≤ +70°C; 4.75V ≤ V_{DD} ≤ 5.25V

SYMBOL	PARAMETER	TEST CONDITIONS	MIN.	MAX.	UNIT
V_{IL}	Input voltage low	V_{DD} = 4.75V		0.8	V
V_{IH}	Input voltage high	V_{DD} = 5.25V	2.0		V
V_I	Input clamp voltage	V_{DD} = 4.75V, I_{IN} = –18mA		–1.2	V
V_{OL}	Output voltage low	V_{DD} = 4.75V, I_{OL} = 12mA		0.5	V
V_{OH}	Output voltage high	V_{DD} = 4.75V, I_{OH} = –12mA	2.4		V
I_I	Input leakage current	V_{IN} = 0 to V_{DD}	–10	10	µA
I_{OZ}	3-Stated output leakage current	V_{IN} = 0 to V_{DD}	–10	10	µA
I_{DDQ}	Standby current	V_{DD} = 5.25V, T_{amb} = 0°C		100	µA
I_{DDD}[1]	Dynamic current	V_{DD} = 5.25V, T_{amb} = 0°C @ 1MHz		5	mA
		V_{DD} = 5.25V, T_{amb} = 0°C @ 50MHz		120	mA
I_{OS}	Short circuit output current[2]	1 pin at a time for no longer than 1 second	–50	–200	mA
C_{IN}	Input pin capacitance[2]	T_{amb} = 25°C, f = 1MHz		8	pF
C_{CLK}	Clock input capacitance[2]	T_{amb} = 25°C, f = 1MHz	5	12	pF
$C_{I/O}$	I/O pin capacitance[2]	T_{amb} = 25°C, f = 1MHz		10	pF

NOTES:

1. This parameter measured with a 16-bit, loadable up/down counter loaded into every logic block, with all outputs enabled and unloaded. Inputs are tied to V_{DD} or ground. This parameter guaranteed by design and characterization, not testing.
2. Typical values, not tested.

AC ELECTRICAL CHARACTERISTICS[1] FOR COMMERCIAL GRADE DEVICES

Commercial: 0°C ≤ T_{amb} ≤ +70°C; 4.75V ≤ V_{DD} ≤ 5.25V

SYMBOL	PARAMETER	–7		UNIT
		MIN.	MAX.	
t_{PD_PAL}	Propagation delay time, input (or feedback node) to output through PAL	2	7.5	ns
t_{PD_PLA}	Propagation delay time, input (or feedback node) to output through PAL & PLA	3	9.5	ns
t_{CO}	Clock to out delay time	2	6	ns
t_{SU_PAL}	Setup time (from input or feedback node) through PAL	5.5		ns
t_{SU_PLA}	Setup time (from input or feedback node) through PAL + PLA	7.5		ns
t_H	Hold time		0	ns
t_{CH}	Clock High time	4		ns
t_{CL}	Clock Low time	4		ns
t_R	Input Rise time		20	ns
t_F	Input Fall time		20	ns
f_{MAX1}	Maximum FF toggle rate[2] 1/(t_{CH} + t_{CL})	125		MHz
f_{MAX2}	Maximum internal frequency[2] 1/(t_{SUPAL} + t_{CF})	100		MHz
f_{MAX3}	Maximum external frequency[2] 1/(t_{SUPAL} + t_{CO})	87		MHz
t_{BUF}	Output buffer delay time		1.5	ns
t_{PDF_PAL}	Input (or feedback node) to internal feedback node delay time through PAL	2	6	ns
t_{PDF_PLA}	Input (or feedback node) to internal feedback node delay time through PAL+PLA	3	8	ns
t_{CF}	Clock to internal feedback node delay time		4.5	ns
t_{INIT}	Delay from valid V_{DD} to valid reset		50	µs
t_{ER}	Input to output disable[3]		10	ns
t_{EA}	Input to output valid		10	ns
t_{RP}	Input to register preset		10	ns
t_{RR}	Input to register reset		10	ns

NOTES:

1. Specifications measured with one output switching. See Figure 6 and Table 8 for derating.
2. This parameter guaranteed by design and characterization, not by test.
3. Output C_L = 5pF.

DC ELECTRICAL CHARACTERISTICS FOR INDUSTRIAL GRADE DEVICES

Industrial: $-40°C \le T_{amb} \le +85°C$; $4.5V \le V_{DD} \le 5.5V$

SYMBOL	PARAMETER	TEST CONDITIONS	MIN.	MAX.	UNIT
V_{IL}	Input voltage low	$V_{DD} = 4.5V$		0.8	V
V_{IH}	Input voltage high	$V_{DD} = 5.5V$	2.0		V
V_I	Input clamp voltage	$V_{DD} = 4.5V$, $I_{IN} = -18mA$		–1.2	V
V_{OL}	Output voltage low	$V_{DD} = 4.5V$, $I_{OL} = 12mA$		0.5	V
V_{OH}	Output voltage high	$V_{DD} = 4.5V$, $I_{OH} = -12mA$	2.4		V
I_I	Input leakage current	$V_{IN} = 0$ to V_{DD}	–10	10	µA
I_{OZ}	3-Stated output leakage current	$V_{IN} = 0$ to V_{DD}	–10	10	µA
I_{DDQ}	Standby current	$V_{DD} = 5.5V$, $T_{amb} = -40°C$		125	µA
I_{DDD}[1]	Dynamic current	$V_{DD} = 5.5V$, $T_{amb} = -40°C$ @ 1MHz		6	mA
		$V_{DD} = 5.5V$, $T_{amb} = -40°C$ @ 50MHz		125	mA
I_{OS}	Short circuit output current[2]	1 pin at a time for no longer than 1 second	–50	–230	mA
C_{IN}	Input pin capacitance[2]	$T_{amb} = 25°C$, f = 1MHz		8	pF
C_{CLK}	Clock input capacitance[2]	$T_{amb} = 25°C$, f = 1MHz	5	12	pF
$C_{I/O}$	I/O pin capacitance[2]	$T_{amb} = 25°C$, f = 1MHz		10	pF

NOTES:

1. This parameter measured with a 16–bit, loadable up/down counter loaded into every logic block, with all outputs enabled and unloaded. Inputs are tied to V_{DD} or ground. This parameter guaranteed by design and characterization, not testing.
2. Typical values, not tested.

AC ELECTRICAL CHARACTERISTICS[1] FOR INDUSTRIAL GRADE DEVICES

Industrial: $-40°C \le T_{amb} \le +85°C$; $4.5V \le V_{DD} \le 5.5V$

SYMBOL	PARAMETER	I10		UNIT
		MIN.	MAX.	
t_{PD_PAL}	Propagation delay time, input (or feedback node) to output through PAL	2	10	ns
t_{PD_PLA}	Propagation delay time, input (or feedback node) to output through PAL & PLA	3	12	ns
t_{CO}	Clock to out delay time	2	7	ns
t_{SU_PAL}	Setup time (from input or feedback node) through PAL	8		ns
t_{SU_PLA}	Setup time (from input or feedback node) through PAL + PLA	10		ns
t_H	Hold time		0	ns
t_{CH}	Clock High time	5		ns
t_{CL}	Clock Low time	5		ns
t_R	Input Rise time		20	ns
t_F	Input Fall time		20	ns
f_{MAX1}	Maximum FF toggle rate[2] $1/(t_{CH} + t_{CL})$	100		MHz
f_{MAX2}	Maximum internal frequency[2] $1/(t_{SUPAL} + t_{CF})$	71		MHz
f_{MAX3}	Maximum external frequency[2] $1/(t_{SUPAL} + t_{CO})$	66		MHz
t_{BUF}	Output buffer delay time		1.5	ns
t_{PDF_PAL}	Input (or feedback node) to internal feedback node delay time through PAL	2	8.5	ns
t_{PDF_PLA}	Input (or feedback node) to internal feedback node delay time through PAL+PLA	3	10.5	ns
t_{CF}	Clock to internal feedback node delay time		6	ns
t_{INIT}	Delay from valid V_{DD} to valid reset		50	µs
t_{ER}	Input to output disable[3]		15	ns
t_{EA}	Input to output valid		15	ns
t_{RP}	Input to register preset		15	ns
t_{RR}	Input to register reset		15	ns

NOTES:

1. Specifications measured with one output switching. See Figure 6 and Table 8 for derating.
2. This parameter guaranteed by design and characterization, not by test.
3. Output C_L = 5pF.

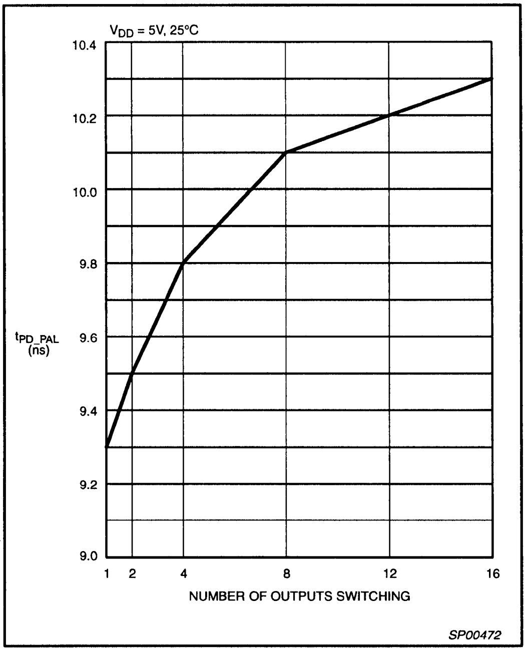


Figure 6. t_{PD_PAL} **vs. Outputs Switching**

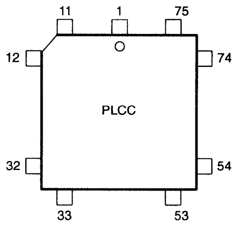
Table 7. t_{PD_PAL} vs. Number of Outputs Switching

V_{DD} = 5.00V

NUMBER OF OUTPUTS	1	2	4	8	12	16
Typical (ns)						

PIN DESCRIPTIONS

84-Pin Plastic Leaded Chip Carrier

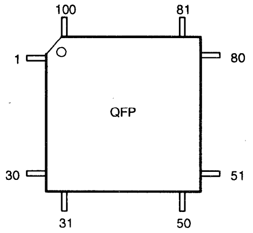


Pin	Function	Pin	Function	Pin	Function
1	IN1	29	I/O-C5	57	I/O-F7
2	IN3	30	I/O-C4	58	I/O-F10
3	V_{DD}	31	I/O-C2	59	GND
4	I/O-A15/CLK3	32	GND	60	I/O-F12
5	I/O-A13	33	I/O-D15	61	I/O-F13
6	I/O-A12	34	I/O-D12	62	I/O-F15 (TCK)
7	GND	35	I/O-D10	63	I/O-G0
8	I/O-A10	36	I/O-D8	64	I/O-G2
9	I/O-A7	37	I/O-D7	65	I/O-G4
10	I/O-A5	38	V_{DD}	66	V_{DD}
11	I/O-A4	39	I/O-D4	67	I/O-G7
12	I/O-A2	40	I/O-D2	68	I/O-G8
13	V_{DD}	41	I/O-D0/CLK2	69	I/O-G10
14	I/O-B15 (TDI)	42	GND	70	I/O-G12
15	I/O-B12	43	V_{DD}	71	I/O-G15 (TDO)
16	I/O-B10	44	I/O-E0/CLK1	72	GND
17	I/O-B8	45	I/O-E2	73	I/O-H2
18	I/O-B7	46	I/O-E4	74	I/O-H4
19	GND	47	GND	75	I/O-H5
20	I/O-B4	48	I/O-E7	76	I/O-H7
21	I/O-B2	49	I/O-E8	77	I/O-H10
22	I/O-B0	50	I/O-E10	78	V_{DD}
23	I/O-C15 (TMS)*	51	I/O-E12	79	I/O-H12
24	I/O-C13	52	I/O-E15	80	I/O-H13
25	I/O-C12	53	V_{DD}	81	I/O-H15
26	V_{DD}	54	I/O-F2	82	GND
27	I/O-C10	55	I/O-F4	83	IN0/CLK0
28	I/O-C7	56	I/O-F5	84	IN2-gtsn

* THE TEST MODE SELECT (TMS) FUNCTION IS INACTIVE ON NON-ISR ARCHITECTURES.

SP00467

100-Pin Plastic Quad Flat Package

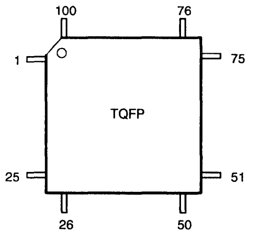


Pin	Function	Pin	Function	Pin	Function
1	I/O-A5	35	I/O-D5	69	I/O-G5
2	I/O-A4	36	V_{DD}	70	I/O-G7
3	I/O-A2	37	I/O-D4	71	I/O-G8
4	I/O-A0	38	I/O-D2	72	I/O-G10
5	V_{DD}	39	I/O-B0/CLK2	73	I/O-G12
6	I/O-B15 (TDI)	40	GND	74	I/O-G13
7	I/O-B13	41	V_{DD}	75	I/O-G15 (TDO)
8	I/O-B12	42	I/O-E0/CLK1	76	GND
9	I/O-B10	43	I/O-E2	77	I/O-H0
10	I/O-B8	44	I/O-E4	78	I/O-H2
11	I/O-B7	45	GND	79	I/O-H4
12	I/O-B5	46	I/O-E5	80	I/O-H5
13	GND	47	I/O-E7	81	I/O-H7
14	I/O-B4	48	I/O-E8	82	I/O-H8
15	I/O-B2	49	I/O-E10	83	I/O-H10
16	I/O-B0	50	I/O-E12	84	V_{DD}
17	I/O-C15 (TMS)*	51	I/O-E13	85	I/O-H12
18	I/O-C13	52	I/O-E15	86	I/O-H13
19	I/O-C12	53	V_{DD}	87	I/O-H15
20	V_{DD}	54	I/O-F0	88	GND
21	I/O-C10	55	I/O-F2	89	IN0/CLK0
22	I/O-C8	56	I/O-F4	90	IN2-gtsn
23	I/O-C7	57	I/O-F5	91	IN1
24	I/O-C5	58	I/O-F7	92	IN3
25	I/O-B9	59	I/O-F8	93	V_{DD}
26	I/O-C2	60	I/O-F10	94	I/O-A15/CLK3
27	I/O-C0	61	GND	95	I/O-A13
28	GND	62	I/O-F12	96	I/O-A12
29	I/O-D15	63	I/O-F13	97	GND
30	I/O-D13	64	I/O-F15 (TCK)	98	I/O-A10
31	I/O-D12	65	I/O-G0	99	I/O-A8
32	I/O-D10	66	I/O-G2	100	I/O-A7
33	I/O-D8	67	I/O-G4		
34	I/O-D7	68	V_{DD}		

* THE TEST MODE SELECT (TMS) FUNCTION IS INACTIVE ON NON-ISR ARCHITECTURES.

SP00468

100-Pin Thin Quad Flat Package

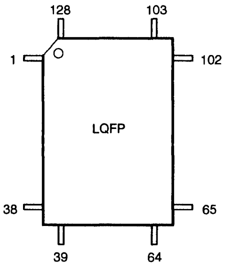


Pin	Function	Pin	Function	Pin	Function
1	I/O-A2	35	I/O-D4	69	I/O-G8
2	I/O-A0	36	I/O-D2	70	I/O-G10
3	V_{DD}	37	I/O-D0/CLK2	71	I/O-G12
4	I/O-B15 (TDI)	38	GND	72	I/O-G13
5	I/O-B13	39	V_{DD}	73	I/O-G15 (TDO)
6	I/O-B12	40	I/O-E0/CLK1	74	GND
7	I/O-B10	41	I/O-E2	75	I/O-H0
8	I/O-B8	42	I/O-E4	76	I/O-H2
9	I/O-B7	43	GND	77	I/O-H4
10	I/O-B5	44	I/O-E5	78	I/O-H5
11	GND	45	I/O-E7	79	I/O-H7
12	I/O-B4	46	I/O-E8	80	I/O-H8
13	I/O-B2	47	I/O-E10	81	I/O-H10
14	I/O-B0	48	I/O-E12	82	V_{DD}
15	I/O-C15 (TMS)*	49	I/O-E13	83	I/O-H12
16	I/O-C13	50	I/O-E15	84	I/O-H13
17	I/O-C12	51	V_{DD}	85	I/O-H15
18	V_{DD}	52	I/O-F0	86	GND
19	I/O-C10	53	I/O-F2	87	IN0/CLK0
20	I/O-C8	54	I/O-F4	88	IN2-gtsn
21	I/O-C7	55	I/O-F5	89	IN1
22	I/O-C5	56	I/O-F7	90	IN3
23	I/O-C4	57	I/O-F8	91	V_{DD}
24	I/O-C2	58	I/O-F10	92	I/O-A15/CLK3
25	I/O-C0	59	GND	93	I/O-A13
26	GND	60	I/O-F12	94	I/O-A12
27	I/O-D15	61	I/O-F13	95	GND
28	I/O-D13	62	I/O-F15 (TCK)	96	I/O-A10
29	I/O-D12	63	I/O-G0	97	I/O-A8
30	I/O-D10	64	I/O-G2	98	I/O-A7
31	I/O-D8	65	I/O-G4	99	I/O-A5
32	I/O-D7	66	V_{DD}	100	I/O-A4
33	I/O-D5	67	I/O-G5		
34	V_{DD}	68	I/O-G7		

* THE TEST MODE SELECT (TMS) FUNCTION IS INACTIVE ON NON-ISR ARCHITECTURES.

SP00485

128-Pin Low Profile Quad Flat Package

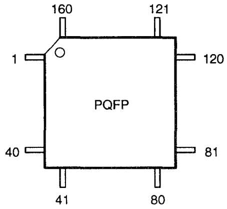


Pin	Function	Pin	Function	Pin	Function
1	I/O-A3	44	I/O-D7	87	V_{DD}
2	I/O-A2	45	I/O-D5	88	I/O-G5
3	I/O-A0	46	V_{DD}	89	I/O-G7
4	NC	47	I/O-D4	90	I/O-G8
5	NC	48	I/O-D3	91	I/O-G10
6	NC	49	I/O-D2	92	I/O-G11
7	V_{DD}	50	I/O-D0/CLK2	93	I/O-G12
8	I/O-B15 (TDI)	51	GND	94	I/O-G13
9	I/O-B13	52	V_{DD}	95	I/O-G15 (TDO)
10	I/O-B12	53	I/O-E0/CLK1	96	GND
11	I/O-B11	54	I/O-E2	97	NC
12	I/O-B10	55	I/O-E3	98	NC
13	I/O-B8	56	I/O-E4	99	NC
14	I/O-B7	57	GND	100	I/O-H0
15	I/O-B5	58	I/O-E5	101	I/O-H2
16	GND	59	I/O-E7	102	I/O-H3
17	I/O-B4	60	I/O-E8	103	I/O-H4
18	I/O-B3	61	I/O-E10	104	I/O-H5
19	I/O-B2	62	I/O-E11	105	I/O-H7
20	I/O-B0	63	I/O-E12	106	I/O-H8
21	I/O-C15 (TMS)*	64	I/O-E13	107	I/O-H10
22	I/O-C13	65	I/O-E15	108	V_{DD}
23	I/O-C12	66	V_{DD}	109	I/O-H11
24	I/O-C11	67	I/O-F0	110	I/O-H12
25	V_{DD}	68	NC	111	I/O-H13
26	I/O-C10	69	NC	112	I/O-H15
27	I/O-C8	70	NC	113	GND
28	I/O-C7	71	I/O-F2	114	IN0/CLK0
29	I/O-C5	72	I/O-F3	115	IN2-gtsn
30	I/O-C4	73	I/O-F4	116	IN1
31	I/O-C3	74	I/O-F5	117	IN3
32	I/O-C2	75	I/O-F7	118	V_{DD}
33	NC	76	I/O-F8	119	I/O-A15/CLK3
34	NC	77	I/O-F10	120	I/O-A13
35	NC	78	GND	121	I/O-A12
36	I/O-C0	79	I/O-F11	122	I/O-A11
37	GND	80	I/O-F12	123	GND
38	I/O-D15	81	I/O-F13	124	I/O-A10
39	I/O-D13	82	I/O-F15(TCK)	125	I/O-A8
40	I/O-D12	83	I/O-G0	126	I/O-A7
41	I/O-D11	84	I/O-G2	127	I/O-A5
42	I/O-D10	85	I/O-G3	128	I/O-A4
43	I/O-D8	86	I/O-G4		

* THE TEST MODE SELECT (TMS) FUNCTION IS INACTIVE ON NON-ISR ARCHITECTURES.

SP00469A

160-Pin Plastic Quad Flat Package



Pin	Function	Pin	Function	Pin	Function
1	NC	54	I/O-D5	107	I/O-G8
2	NC	55	V_{DD}	108	I/O-G10
3	NC	56	I/O-D4	109	I/O-G11
4	NC	57	I/O-D3	110	I/O-G12
5	NC	58	I/O-D2	111	I/O-G13
6	NC	59	I/O-D0/CLK2	112	I/O-G15 (TDO)
7	NC	60	GND	113	GND
8	V_{DD}	61	V_{DD}	114	NC
9	I/O-B15 (TDI)	62	I/O-E0/CLK1	115	NC
10	I/O-B13	63	I/O-E2	116	NC
11	I/O-B12	64	I/O-E3	117	NC
12	I/O-B11	65	I/O-E4	118	NC
13	I/O-B10	66	GND	119	NC
14	I/O-B8	67	I/O-E5	120	NC
15	I/O-B7	68	I/O-E7	121	I/O-H0
16	I/O-B5	69	I/O-E8	122	I/O-H2
17	GND	70	I/O-E10	123	I/O-H3
18	I/O-B4	71	I/O-E11	124	NC
19	I/O-B3	72	I/O-E12	125	NC
20	I/O-B2	73	I/O-E13	126	NC
21	I/O-B0	74	NC	127	NC
22	I/O-C15 (TMS)	75	NC	128	I/O-H4
23	I/O-C13	76	NC	129	I/O-H5
24	I/O-C12	77	NC	130	I/O-H7
25	I/O-C11	78	I/O-E15	131	I/O-H8
26	V_{DD}	79	V_{DD}	132	I/O-H10
27	I/O-C10	80	I/O-F0	133	V_{DD}
28	I/O-C8	81	NC	134	I/O-H11
29	I/O-C7	82	NC	135	I/O-H12
30	I/O-C5	83	NC	136	I/O-H13
31	I/O-C4	84	NC	137	I/O-H15
32	I/O-C3	85	NC	138	GND
33	I/O-C2	86	NC	139	IN0/CLK0
34	NC	87	NC	140	IN2-gtsn
35	NC	88	I/O-F2	141	IN1
36	NC	89	I/O-F3	142	IN3
37	NC	90	I/O-F4	143	V_{DD}
38	NC	91	I/O-F5	144	I/O-A15/CLK3
39	NC	92	I/O-F7	145	I/O-A13
40	NC	93	I/O-F8	146	I/O-A12
41	I/O-C0	94	I/O-F10	147	I/O-A11
42	GND	95	GND	148	GND
43	I/O-D15	96	I/O-F11	149	I/O-A10
44	NC	97	I/O-F12	150	I/O-A8
45	NC	98	I/O-F13	151	I/O-A7
46	NC	99	I/O-F15 (TCK)	152	I/O-A5
47	NC	100	I/O-G0	153	I/O-A4
48	I/O-D13	101	I/O-G2	154	NC
49	I/O-D12	102	I/O-G3	155	NC
50	I/O-D11	103	I/O-G4	156	NC
51	I/O-D10	104	V_{DD}	157	NC
52	I/O-D8	105	I/O-G5	158	I/O-A3
53	I/O-D7	106	I/O-G7	159	I/O-A2
				160	I/O-A0

* THE TEST MODE SELECT (TMS) FUNCTION IS INACTIVE ON NON-ISR ARCHITECTURES.

SP00470A

Package Thermal Characteristics

Philips Semiconductors uses the Temperature Sensitive Parameter (TSP) method to test thermal resistance. This method meets Mil-Std-883C Method 1012.1 and is described in Philips *1995 IC Package Databook*. Thermal resistance varies slightly as a function of input power. As input power increases, thermal resistance changes approximately 5% for a 100% change in power.

Figure 7 is a derating curve for the change in Θ_{JA} with airflow based on wind tunnel measurements. It should be noted that the wind flow dynamics are more complex and turbulent in actual applications than in a wind tunnel. Also, the test boards used in the wind tunnel contribute significantly to forced convection heat transfer, and may not be similar to the actual circuit board, especially in size.

Package	Θ_{JA}
84-pin PLCC	32.8 °C/W
100-pin PQFP	41.2 °C/W
100-pin TQFP	47.4 °C/W
128-pin LQFP	45.0 °C/W
160-pin PQFP	31.4 °C/W

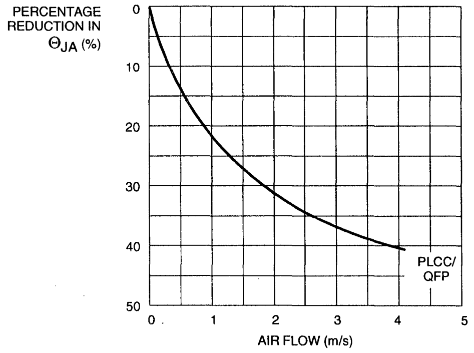


Figure 7. Average Effect of Airflow on Θ_{JA}

Section 5
Related Products

CONTENTS

3V zero power, TotalCMOS™, universal PLD device P3Z22V10

FEATURES

- Industry's first TotalCMOS™ 22V10 – both CMOS design and process technologies
- Fast Zero Power (FZP™) design technique provides ultra-low power and high speed
  - Static current of less than 45µA
  - Dynamic current 1/10 to 1/1000 that of competitive devices
  - Pin-to-pin delay of only 10ns
- True Zero Power device with no turbo bits or power down schemes
- Function/JEDEC map compatible with Bipolar UVCMOS EECMOS 22V10s
- Multiple packaging options featuring PCB-friendly flow-through pinouts (SOL and TSSOP)
  - 24-pin TSSOP—uses 93% less in-system space than a 28-pin PLCC
  - 24-pin SOL
  - 28-pin PLCC with standard JEDEC pin-out
- Available in commercial and industrial operating ranges
- Supports mixed voltage systems–5V tolerant I/Os
- Advanced 0.5µ E^2CMOS process
- 1000 erase/program cycles guaranteed
- 20 years data retention guaranteed
- Varied product term distribution with up to 16 product terms per output for complex functions
- Programmable output polarity
- Synchronous preset/asynchronous reset capability
- Security bit prevents unauthorized access
- Electronic signature for identification
- Design entry and verification using industry standard CAE tools
- Reprogrammable using industry standard device programmers

DESCRIPTION

The P3Z22V10 is the first SPLD to combine high performance with low power, without the need for "turbo bits" or other power down schemes. To achieve this, Philips Semiconductors has used their FZP™ design technique, which replaces conventional sense amplifier methods for implementing product terms (a technique that has been used in PLDs since the bipolar era) with a cascaded chain of pure CMOS gates. This results in the combination of low power and high speed that has previously been unattainable in the PLD arena. For 5V operation, Philips Semiconductors offers the P5Z22V10 that offers high speed and low power in a 5V implementation.

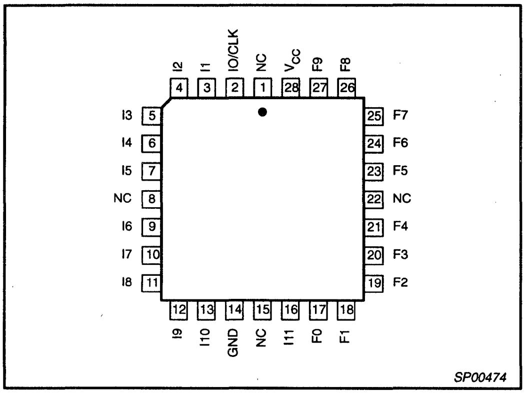
The P3Z22V10 uses the familiar AND/OR logic array structure, which allows direct implementation of sum-of-products equations. This device has a programmable AND array which drives a fixed OR array. The OR sum of products feeds an "Output Macro Cell" (OMC), which can be individually configured as a dedicated input, a combinatorial output, or a registered output with internal feedback.

ORDERING INFORMATION

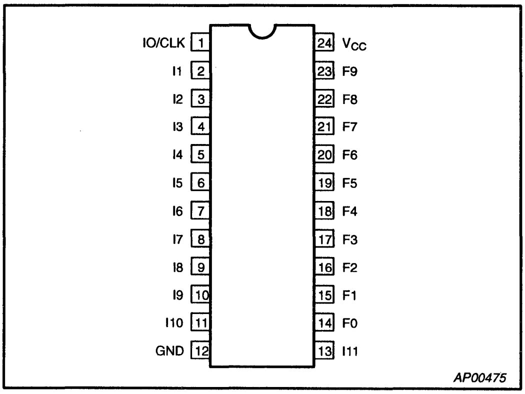
ORDER CODE	PACKAGE	PROPAGATION DELAY	TEMPERATURE RANGE	OPERATING RANGE	DRAWING NUMBER
P3Z22V10-DA	28-pin PLCC	10ns	0 to +70°C	V_{CC} = 3.3V ±10%	SOT261-3
P3Z22V10-DD	24-pin SOL	10ns	0 to +70°C	V_{CC} = 3.3V ±10%	SOT137-1
P3Z22V10-DDH	24-pin TSSOP	10ns	0 to +70°C	V_{CC} = 3.3V ±10%	SOT355-1
P3Z22V10-BA	28-pin PLCC	15ns	0 to +70°C	V_{CC} = 3.3V ±10%	SOT261-3
P3Z22V10-BD	24-pin SOL	15ns	0 to +70°C	V_{CC} = 3.3V ±10%	SOT137-1
P3Z22V10-BDH	24-pin TSSOP	15ns	0 to +70°C	V_{CC} = 3.3V ±10%	SOT355-1
P3Z22V10IBA	28-pin PLCC	15ns	–40 to +85°C	V_{CC} = 3.3V ±10%	SOT261-3
P3Z22V10IBD	24-pin SOL	15ns	–40 to +85°C	V_{CC} = 3.3V ±10%	SOT137-1
P3Z22V10IBDH	24-pin TSSOP	15ns	–40 to +85°C	V_{CC} = 3.3V ±10%	SOT355-1

PIN CONFIGURATIONS

28-Pin PLCC



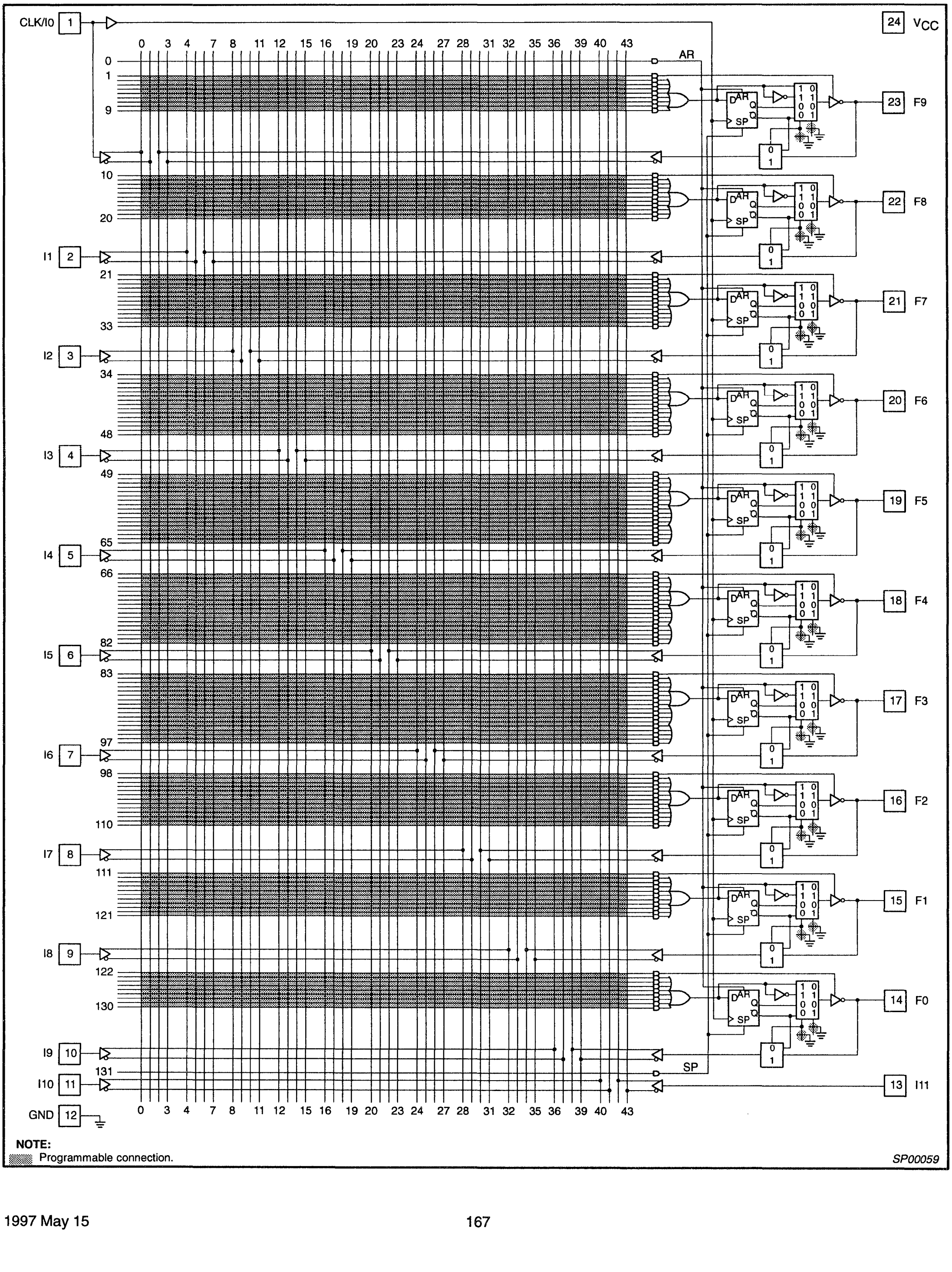
24-Pin SOL and 24-Pin TSSOP



PIN DESCRIPTIONS

PIN LABEL	DESCRIPTION
I1 – I11	Dedicated Input
NC	Not Connected
F0 – F9	Macrocell Input/Output
I0/CLK	Dedicated Input/Clock Input
V_{CC}	Supply Voltage
GND	Ground

LOGIC DIAGRAM



NOTE:
▒ Programmable connection.

SP00059

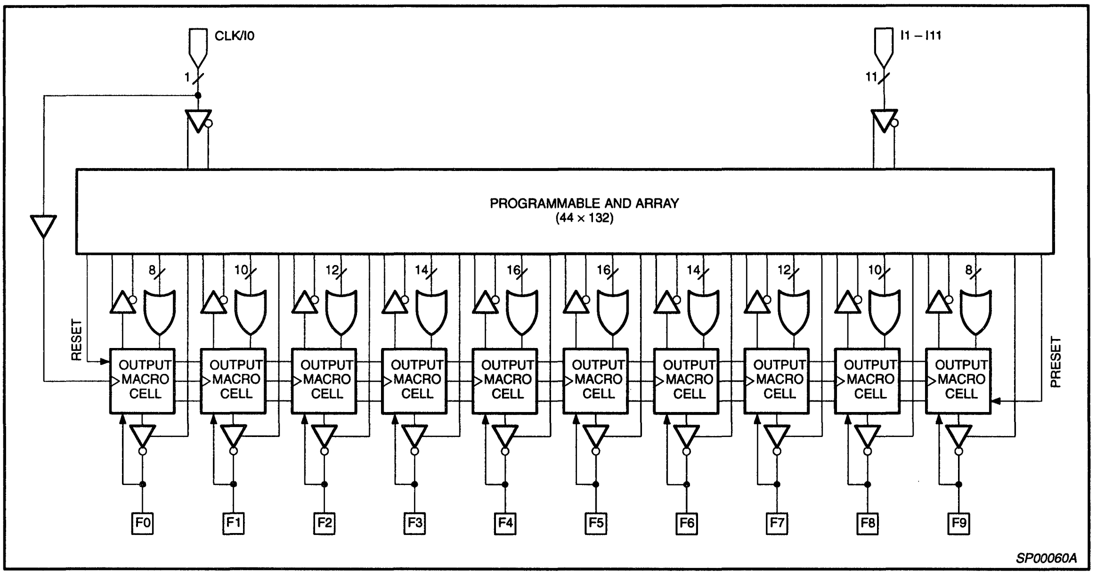


Figure 1. Functional Diagram

FUNCTIONAL DESCRIPTION

The P3Z22V10 implements logic functions as sum-of-products expressions in a programmable-AND/fixed-OR logic array. User-defined functions are created by programming the connections of input signals into the array. User-configurable output structures in the form of I/O macrocells further increase logic flexibility.

ARCHITECTURE OVERVIEW

The P3Z22V10 architecture is illustrated in Figure 1. Twelve dedicated inputs and 10 I/Os provide up to 22 inputs and 10 outputs for creation of logic functions. At the core of the device is a programmable electrically-erasable AND array which drives a fixed-OR array. With this structure, the P3Z22V10 can implement up to 10 sum-of-products logic expressions.

Associated with each of the 10 OR functions is an I/O macrocell which can be independently programmed to one of 4 different configurations. The programmable macrocells allow each I/O to create sequential or combinatorial logic functions with either Active-High or Active-Low polarity.

AND/OR Logic Array

The programmable AND array of the P3Z22V10 (shown in the Logic Diagram) is formed by input lines intersecting product terms. The input lines and product terms are used as follows:

44 input lines:

- 24 input lines carry the True and Complement of the signals applied to the 12 input pins
- 20 additional lines carry the True and Complement values of feedback or input signals from the 10 I/Os

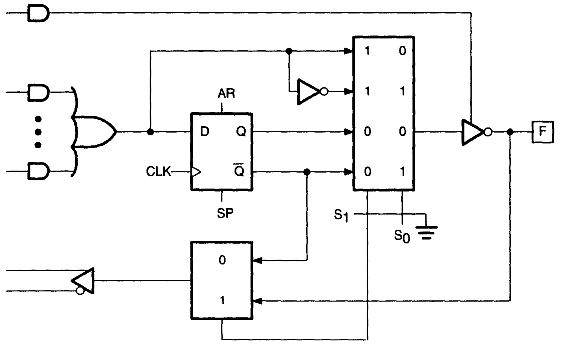
132 product terms:

- 120 product terms (arranged in 2 groups of 8, 10, 12, 14, and 16) used to form logical sums
- 10 output enable terms (one for each I/O)
- 1 global synchronous preset product term
- 1 global asynchronous clear product term

At each input-line/product-term intersection there is an EEPROM memory cell which determines whether or not there is a logical connection at that intersection. Each product term is essentially a 44-input AND gate. A product term which is connected to both the True and Complement of an input signal will always be FALSE, and thus will not affect the OR function that it drives. When all the connections on a product term are opened, a Don't Care state exists and that term will always be TRUE.

Variable Product Term Distribution

The P3Z22V10 provides 120 product terms to drive the 10 OR functions. These product terms are distributed among the outputs in groups of 8, 10, 12, 14, and 16 to form logical sums (see Logic Diagram). This distribution allows optimum use of device resources.

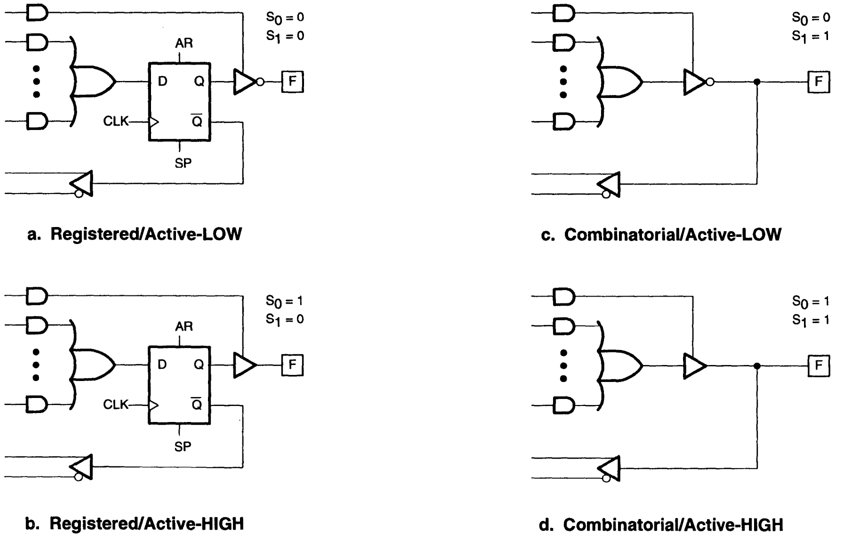


S_1	S_0	OUTPUT CONFIGURATION
0	0	Registered/Active-LOW/Macrocell feedback
0	1	Registered/Active-HIGH/Macrocell feedback
1	0	Combinatorial/Active-LOW/Pin feedback
1	1	Combinatorial/Active-HIGH/Pin feedback

0 = Unprogrammed fuse
1 = Programmed fuse

SP00484

Figure 2. Output Macro Cell Logic Diagram



a. Registered/Active-LOW

c. Combinatorial/Active-LOW

b. Registered/Active-HIGH

d. Combinatorial/Active-HIGH

SP00376

Figure 3. Output Macro Cell Configurations

Programmable I/O Macrocell

The output macrocell provides complete control over the architecture of each output. the ability to configure each output independently permits users to tailor the configuration of the P3Z22V10 to the precise requirements of their designs.

Macrocell Architecture

Each I/O macrocell, as shown in Figure 2, consists of a D-type flip-flop and two signal-select multiplexers. The configuration of each macrocell of the P3Z22V10 is determined by the two EEPROM bits controlling these multiplexers. These bits determine output polarity, and output type (registered or non-registered). Equivalent circuits for the macrocell configurations are illustrated in Figure 3.

Output type

The signal from the OR array can be fed directly to the output pin (combinatorial function) or latched in the D-type flip-flop (registered function). The D-type flip-flop latches data on the rising edge of the clock and is controlled by the global preset and clear terms. When the synchronous preset term is satisfied, the Q output of the register will be set HIGH at the next rising edge of the clock input. Satisfying the asynchronous clear term will set Q LOW, regardless of the clock state. If both terms are satisfied simultaneously, the clear will override the preset.

Program/Erase Cycles

The P3Z22V10 is 100% testable, erases/programs in seconds, and guarantees 1000 program/erase cycles.

Output Polarity

Each macrocell can be configured to implement Active-High or Active-Low logic. Programmable polarity eliminates the need for external inverters.

Output Enable

The output of each I/O macrocell can be enabled or disabled under the control of its associated programmable output enable product term. When the logical conditions programmed on the output enable term are satisfied, the output signal is propagated to the I/O pin. Otherwise, the output buffer is driven into the high-impedance state.

Under the control of the output enable term, the I/O pin can function as a dedicated input, a dedicated output, or a bi-directional I/O. Opening every connection on the output enable term will permanently enable the output buffer and yield a dedicated output. Conversely, if every connection is intact, the enable term will always be logically FALSE and the I/O will function as a dedicated input.

Register Feedback Select

When the I/O macrocell is configured to implement a registered function (S1 = 0) (Figures 3a or 3b), the feedback signal to the AND array is taken from the $\overline{Q}$ output.

Bi-directional I/O Select

When configuring an I/O macrocell to implement a combinatorial function (S1 = 1) (Figures 3c or 3d), the feedback signal is taken from the I/O pin. In this case, the pin can be used as a dedicated input, a dedicated output, or a bi-directional I/O.

Power-On Reset

To ease system initialization, all flip-flops will power-up to a reset condition and the Q output will be low. The actual output of the P3Z22V10 will depend on the programmed output polarity. The V_{CC} rise must be monotonic.

Design Security

The P3Z22V10 provides a special EEPROM security bit that prevents unauthorized reading or copying of designs programmed into the device. The security bit is set by the PLD programmer, either at the conclusion of the programming cycle or as a separate step, after the device has been programmed. Once the security bit is set, it is impossible to verify (read) or program the P3Z22V10 until the entire device has first been erased with the bulk-erase function.

TotalCMOS™ Design Technique for Fast Zero Power

Philips is the first to offer a TotalCMOS™ SPLD, both in process technology and design technique. Philips employs a cascade of CMOS gates to implement its Sum of Products instead of the traditional sense amp approach. This CMOS gate implementation allows Philips to offer SPLDs which are both high performance and low power, breaking the paradigm that to have low power, you must accept low performance. Refer to Figure 4 and Table 1 showing the I_{DD} vs. Frequency of our P3Z22V10 TotalCMOS™ SPLD.

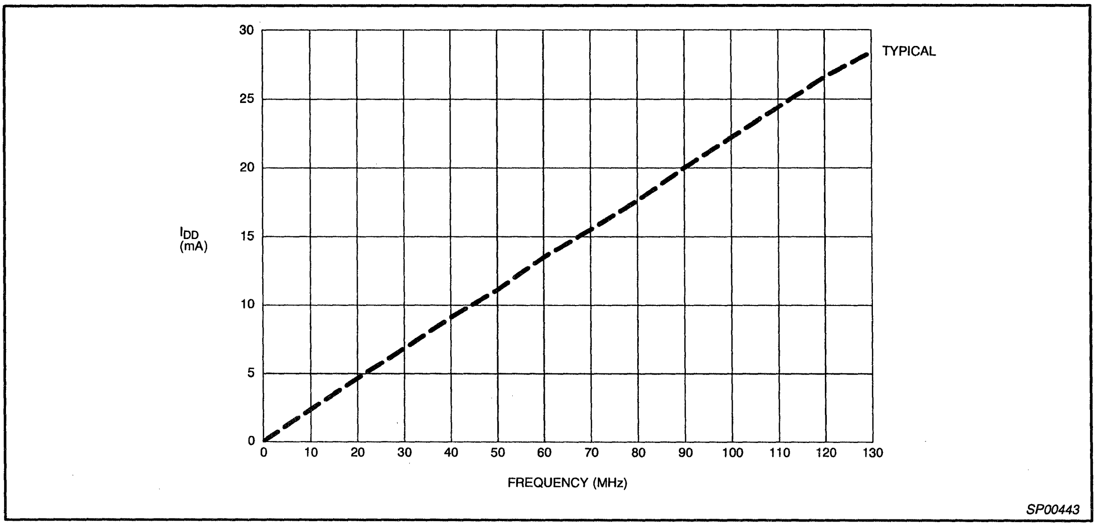


Figure 4. I_{DD} vs. Frequency @ V_{DD} = 3.3V (10-bit counter)

Table 1. I_{DD} vs. Frequency

V_{DD} = 3.3V

FREQ (MHz)	0	10	20	30	40	50	60	70	80	90	100	110	120	130
Typical I_{DD} (mA)														

ABSOLUTE MAXIMUM RATINGS[1]

SYMBOL	PARAMETER	LIMITS		UNIT
		MIN.	MAX.	
V_{DD}	Supply voltage	–0.5	4.6	V
V_I	Input voltage	–0.5	5.5[2]	V
V_{OUT}	Output voltage	–0.5	5.5[2]	V
I_{IN}	Input current	–30	30	mA
I_{OUT}	Output current	–100	100	mA
T_R	Allowable thermal rise ambient to junction	0	75	°C
T_J	Junction temperature range	–40	150	°C
T_{STG}	Storage temperature range	–65	150	°C

NOTES:
1. Stresses above those listed may cause malfunction or permanent damage to the device. This is a stress rating only. Functional operation at these or any other condition above those indicated in the operational and programming specification of the device is not implied.
2. Except F7, where max = V_{DD} + 0.5V.

OPERATING RANGE

PRODUCT GRADE	TEMPERATURE	VOLTAGE
Commercial	0 to +70°C	3.3 ± 10% V
Industrial	–40 to +85°C	3.3 ± 10% V

DC ELECTRICAL CHARACTERISTICS FOR COMMERCIAL GRADE DEVICES

Commercial: 0°C ≤ T_{amb} ≤ +70°C; 3.0 ≤ V_{DD} ≤ 3.6V

SYMBOL	PARAMETER	TEST CONDITIONS	LIMITS		UNITS
			MIN.	MAX.	
V_{IL}	Input voltage low	V_{DD} = 3.0V		0.8	V
V_{IH}	Input voltage high	V_{DD} = 3.6V	2		V
V_I	Input clamp voltage	V_{DD} = 3.0V; I_{IN} = –18mA		–1.2	V
V_{OL}	Output voltage low	V_{DD} = 3.0V; I_{OL} = 8mA		0.5	V
V_{OH}	Output voltage high	V_{DD} = 3.0V; I_{OL} = –4mA	2.4		V
I_I	Input leakage current	V_{IN} = 0 to V_{DD}	–10	10	µA
		V_{IN} = V_{DD} to 5.5V[2]			
I_{OZ}	3-Stated output leakage current	V_{IN} = 0 to V_{DD}	–10	10	µA
		V_{IN} = V_{DD} to 5.5V[2]			
I_{DDQ}	Standby current	V_{DD} = 3.6V; T_{amb} = 0°C		45	µA
I_{DDD} [1]	Dynamic current	V_{DD} = 3.6V; T_{amb} = 0°C @ 1MHz			mA
		V_{DD} = 3.6V; T_{amb} = 0°C @ 50MHz			mA
I_{SC}	Short circuit output current	1 pin/time for no longer than 1 second	–30	–100	mA
C_{IN}	Input pin capacitance	T_{amb} = 25°C; f = 1MHz		8	pF
C_{CLK}	Clock input capacitance	T_{amb} = 25°C; f = 1MHz	5	12	pF
$C_{I/O}$	I/O pin capacitance	T_{amb} = 25°C; f = 1MHz		10	pF

NOTES:

1. These parameters measured with a 10-bit up counter, with all outputs enabled and unloaded. Inputs are tied to V_{DD} or ground. These parameters are not 100% tested, but are calculated at initial characterization and at any time the design is modified where current may be affected.
2. Does not apply to F7.

AC ELECTRICAL CHARACTERISTICS FOR COMMERCIAL GRADE DEVICES

Commercial: 0°C ≤ T_{amb} ≤ +70°C; 3.0 ≤ V_{DD} ≤ 3.6V

SYMBOL	PARAMETER	–B		–D		UNIT
		MIN.	MAX.	MIN.	MAX.	
t_{PD}	Input or feedback to non-registered output		15		10	ns
t_{SU}	Setup time from input, feedback or SP to Clock	4.5		3.5		ns
t_{CO}	Clock to output		10		8.5	ns
t_{CF}	Clock to feedback[1]		6		4.5	ns
t_H	Hold time		0		0	ns
t_{AR}	Asynchronous Reset to registered output		17		17	ns
t_{ARW}	Asynchronous Reset width	5		5		ns
t_{ARR}	Asynchronous Reset recovery time		6		6	ns
t_{SPR}	Synchronous Preset recovery time		6		6	ns
t_{WL}	Width of Clock LOW	3		3		ns
t_{WH}	Width of Clock HIGH	3		3		ns
t_R	Input rise time		20		20	ns
t_F	Input fall time		20		20	ns
f_{MAX1}	Maximum internal frequency[2] (1/t_{SU} + t_{CF})	95		125		MHz
f_{MAX2}	Maximum external frequency[1] (1/t_{SU} + t_{CO})	69		83		MHz
f_{MAX3}	Maximum clock frequency[1] (1/t_{WL} + t_{WH})	167		167		MHz
t_{EA}	Input to Output Enable		9		9	ns
t_{ER}	Input to Output Disable		9		9	ns
Capacitance						
C_{IN}	Input pin capacitance		10		10	pF
C_{OUT}	Output capacitance		10		10	pF

NOTES:

1. These parameters are not 100% tested, but are calculated at initial characterization and at any time the design is modified where frequency may be affected.
2. These parameters measured with a 10-bit up counter, with all outputs enabled and unloaded. Inputs are tied to V_{DD} or ground. These parameters are not 100% tested, but are calculated at initial characterization and at any time the design is modified where frequency may be affected.

DC ELECTRICAL CHARACTERISTICS FOR INDUSTRIAL GRADE DEVICES

Industrial: –40°C ≤ T_{amb} ≤ +85°C; 3.0 ≤ V_{DD} ≤ 3.6V

SYMBOL	PARAMETER	TEST CONDITIONS	LIMITS		UNITS
			MIN.	MAX.	
V_{IL}	Input voltage low	V_{DD} = 3.0V		0.8	V
V_{IH}	Input voltage high	V_{DD} = 3.6V	2		V
V_I	Input clamp voltage	V_{DD} = 3.0V; I_{IN} = –18mA		–1.2	V
V_{OL}	Output voltage low	V_{DD} = 3.0V; I_{OL} = 8mA		0.5	V
V_{OH}	Output voltage high	V_{DD} = 3.0V; I_{OL} = –4mA	2.4		V
I_I	Input leakage current	V_{IN} = 0 to V_{DD}	–10	10	µA
		V_{IN} = V_{DD} to 5.5V[2]			
I_{SC}	3-Stated output leakage current	V_{IN} = 0 to V_{DD}	–10	10	µA
		V_{IN} = V_{DD} to 5.5V[2]			
I_{DDQ}	Standby current	V_{DD} = 3.6V; T_{amb} = –40°C		45	µA
I_{DDD} [1]	Dynamic current	V_{DD} = 3.6V; T_{amb} = –40°C @ 1MHz			mA
		V_{DD} = 3.6V; T_{amb} = –40°C @ 50MHz			mA
I_{OS}	Short circuit output current	1 pin/time for no longer than 1 second	–30	–100	mA
C_{IN}	Input pin capacitance	T_{amb} = 25°C; f = 1MHz		8	pF
C_{CLK}	Clock input capacitance	T_{amb} = 25°C; f = 1MHz	5	12	pF
$C_{I/O}$	I/O pin capacitance	T_{amb} = 25°C; f = 1MHz		10	pF

NOTES:

1. These parameters measured with a 10-bit up counter, with all outputs enabled and unloaded. Inputs are tied to V_{DD} or ground. These parameters are not 100% tested, but are calculated at initial characterization and at any time the design is modified where current may be affected.
2. Does not apply to F7.

AC ELECTRICAL CHARACTERISTICS FOR INDUSTRIAL GRADE DEVICES

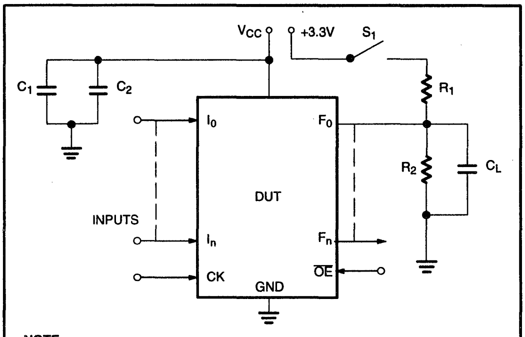
Industrial: –40°C ≤ T_{amb} ≤ +85°C; 3.0 ≤ V_{DD} ≤ 3.6V

SYMBOL	PARAMETER	LIMITS		UNIT
		MIN.	MAX.	
t_{PD}	Input or feedback to non-registered output		15	ns
t_{SU}	Setup time from input, feedback or SP to Clock	5		ns
t_{CO}	Clock to output		10.5	ns
t_{CF}	Clock to feedback[1]		6	ns
t_H	Hold time		0	ns
t_{AR}	Asynchronous Reset to registered output		17	ns
t_{ARW}	Asynchronous Reset width	5		ns
t_{ARR}	Asynchronous Reset recovery time		6	ns
t_{SPR}	Synchronous Preset recovery time		6	ns
t_{WL}	Width of Clock LOW	3		ns
t_{WH}	Width of Clock HIGH	3		ns
t_R	Input rise time		20	ns
t_F	Input fall time		20	ns
f_{MAX1}	Maximum internal frequency[2] $(1/t_{SU} + t_{CF})$	91		MHz
f_{MAX2}	Maximum external frequency[1] $(1/t_{SU} + t_{CO})$	65		MHz
f_{MAX3}	Maximum clock frequency[1] $(1/t_{WL} + t_{WH})$	167		MHz
t_{EA}	Input to Output Enable		11	ns
t_{ER}	Input to Output Disable		11	ns
Capacitance				
C_{IN}	Input pin capacitance		10	pF
C_{OUT}	Output capacitance		12	pF

NOTES:

1. These parameters are not 100% tested, but are calculated at initial characterization and at any time the design is modified where frequency may be affected.
2. These parameters measured with a 10-bit up counter, with all outputs enabled and unloaded. Inputs are tied to V_{DD} or ground. These parameters are not 100% tested, but are calculated at initial characterization and at any time the design is modified where frequency may be affected.

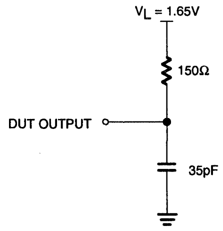
TEST LOAD CIRCUIT



NOTE:
C_1 and C_2 are to bypass V_{CC} to GND.
R_1 = 300Ω, R_2 = 300Ω, C_L = 35pF.

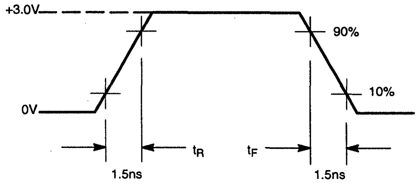
SP00478

THEVENIN EQUIVALENT



SP00479A

VOLTAGE WAVEFORM

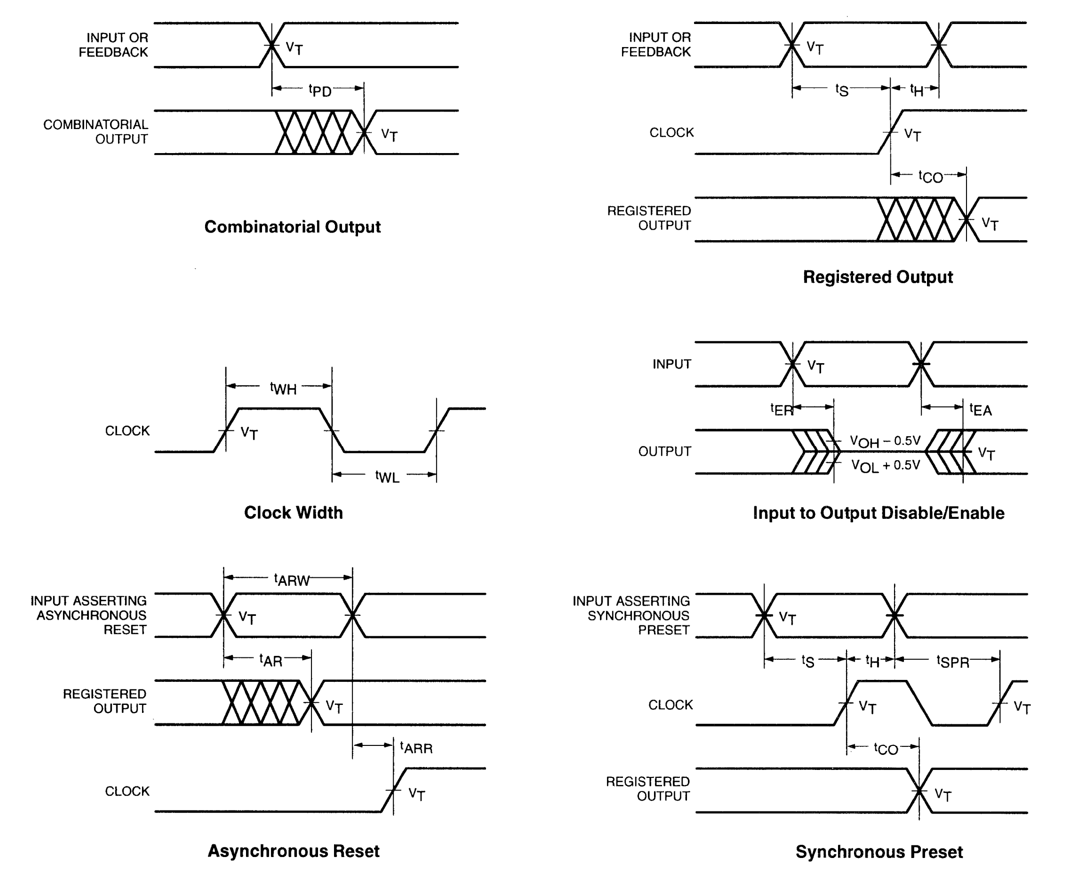


MEASUREMENTS:
All circuit delays are measured at the +1.5V level of inputs and outputs, unless otherwise specified.

Input Pulses

SP00368

SWITCHING WAVEFORMS



Combinatorial Output

Registered Output

Clock Width

Input to Output Disable/Enable

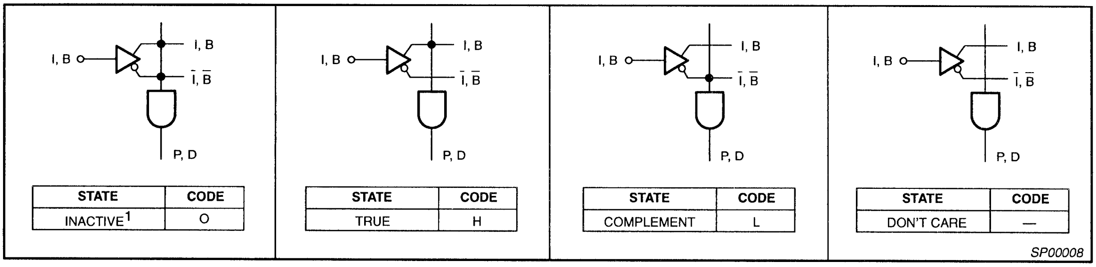
Asynchronous Reset

Synchronous Preset

NOTES:

1. $V_T = 1.5V$.
2. Input pulse amplitude 0V to 3.0V.
3. Input rise and fall times 2.0ns max.

"AND" ARRAY – (I, B)



STATE	CODE
INACTIVE[1]	O

STATE	CODE
TRUE	H

STATE	CODE
COMPLEMENT	L

STATE	CODE
DON'T CARE	—

NOTE:

1. This is the initial state.

5V zero power, TotalCMOS™, universal PLD device P5Z22V10

FEATURES

- Industry's first TotalCMOS™ 22V10 – both CMOS design and process technologies
- Fast Zero Power (FZP™) design technique provides ultra-low power and high speed
  - Static current of less than 75µA
  - Dynamic current 1/10 to 1/1000 that of competing devices
  - Pin-to-pin delay of only 7.5ns
- True Zero Power device with no turbo bits or power down schemes
- Function/JEDEC map compatible with Bipolar UVCMOS EECMOS 22V10s
- Multiple packaging options featuring PCB-friendly flow-through pinouts (SOL and TSSOP)
  - 24-pin TSSOP—uses 93% less in-system space than a 28-pin PLCC
  - 24-pin SOL
  - 28-pin PLCC with standard JEDEC pin-out
- Available in commercial and industrial operating ranges
- Advanced 0.5µ E^{2}CMOS process
- 1000 erase/program cycles guaranteed
- 20 years data retention guaranteed
- Varied product term distribution with up to 16 product terms per output for complex functions
- Programmable output polarity
- Synchronous preset/asynchronous reset capability
- Security bit prevents unauthorized access
- Electronic signature for identification
- Design entry and verification using industry standard CAE tools
- Reprogrammable using industry standard device programmers

DESCRIPTION

The P5Z22V10 is the first SPLD to combine high performance with low power, without the need for "turbo bits" or other power down schemes. To achieve this, Philips Semiconductors has used their FZP™ design technique, which replaces conventional sense amplifier methods for implementing product terms (a technique that has been used in PLDs since the bipolar era) with a cascaded chain of pure CMOS gates. This results in the combination of low power and high speed that has previously been unattainable in the PLD arena. For 3V operation, Philips Semiconductors offers the P3Z22V10 that offers high speed and low power in a 3V implementation.

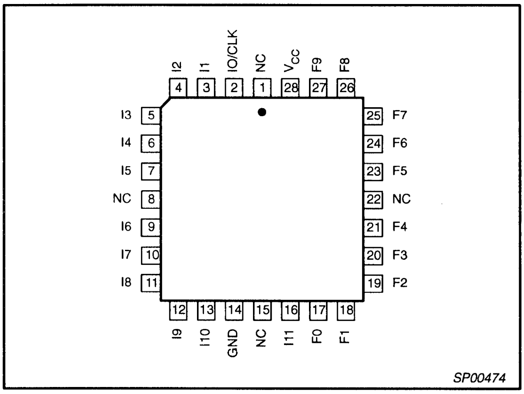
The P5Z22V10 uses the familiar AND/OR logic array structure, which allows direct implementation of sum-of-products equations. This device has a programmable AND array which drives a fixed OR array. The OR sum of products feeds an "Output Macro Cell" (OMC), which can be individually configured as a dedicated input, a combinatorial output, or a registered output with internal feedback.

ORDERING INFORMATION

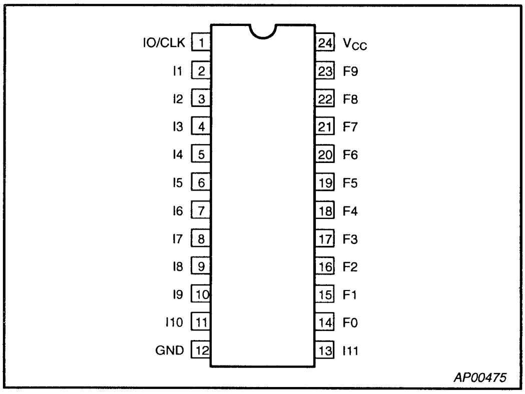
ORDER CODE	PACKAGE	PROPAGATION DELAY	TEMPERATURE RANGE	OPERATING RANGE	DRAWING NUMBER
P5Z22V10-7A	28-pin PLCC	7.5ns	0 to +70°C	V_{CC} = 5.0V ±5%	SOT261-3
P5Z22V10-7D	24-pin SOL	7.5ns	0 to +70°C	V_{CC} = 5.0V ±5%	SOT137-1
P5Z22V10-7DH	24-pin TSSOP	7.5ns	0 to +70°C	V_{CC} = 5.0V ±5%	SOT355-1
P5Z22V10–DA	28-pin PLCC	10ns	0 to +70°C	V_{CC} = 5.0V ±5%	SOT261-3
P5Z22V10–DD	24-pin SOL	10ns	0 to +70°C	V_{CC} = 5.0V ±5%	SOT137-1
P5Z22V10–DDH	24-pin TSSOP	10ns	0 to +70°C	V_{CC} = 5.0V ±5%	SOT355-1
P5Z22V10IDA	28-pin PLCC	10ns	–40 to +85°C	V_{CC} = 5.0V ±10%	SOT261-3
P5Z22V10IDD	24-pin SOL	10ns	–40 to +85°C	V_{CC} = 5.0V ±10%	SOT137-1
P5Z22V10IDDH	24-pin TSSOP	10ns	–40 to +85°C	V_{CC} = 5.0V ±10%	SOT355-1

PIN CONFIGURATIONS

28-Pin PLCC



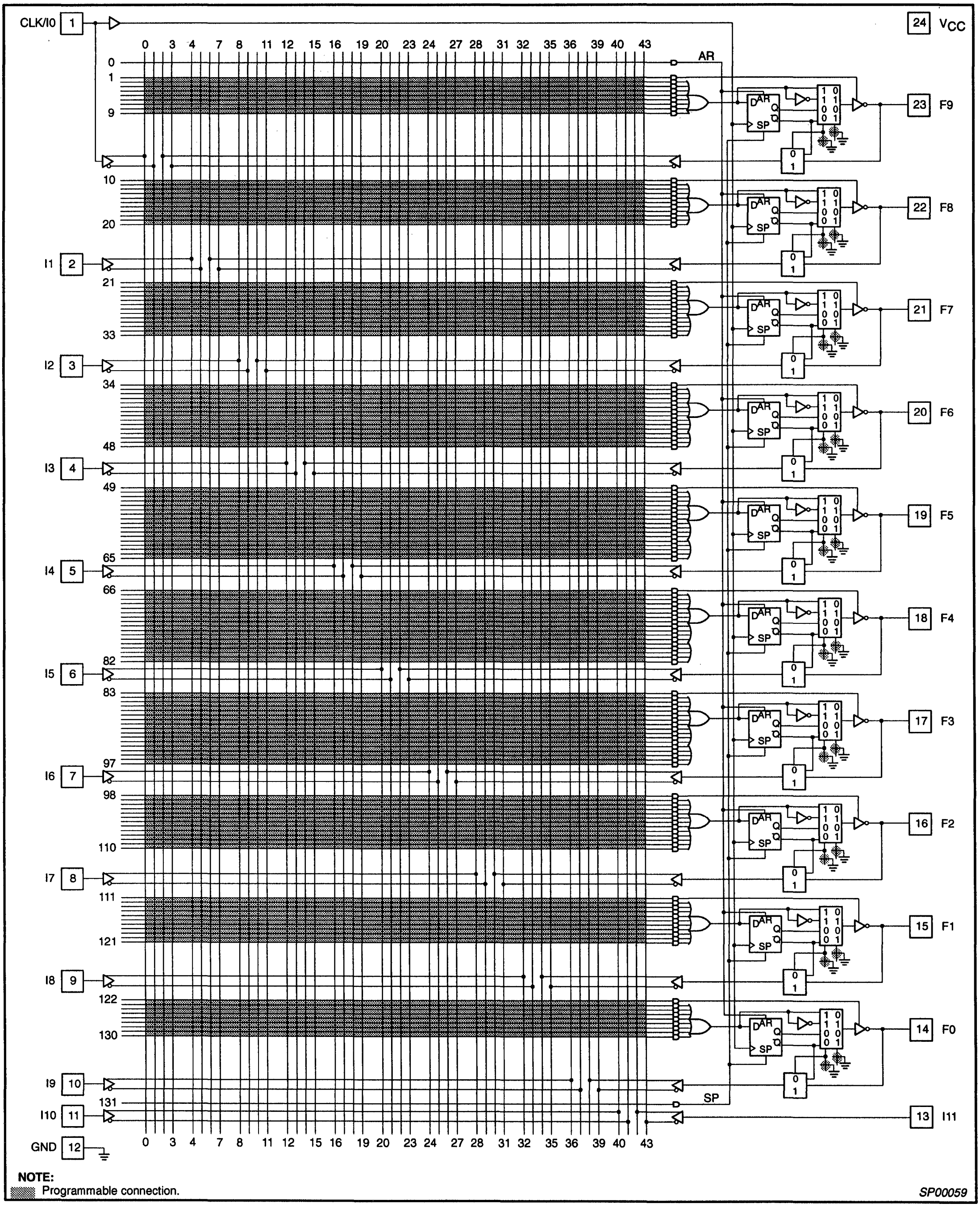
24-Pin SOL and 24-Pin TSSOP



PIN DESCRIPTIONS

PIN LABEL	DESCRIPTION
I1 – I11	Dedicated Input
NC	Not Connected
F0 – F9	Macrocell Input/Output
I0/CLK	Dedicated Input/Clock Input
V_{CC}	Supply Voltage
GND	Ground

LOGIC DIAGRAM



NOTE:
▨ Programmable connection.

SP00059

5V zero power, TotalCMOS™, universal PLD device P5Z22V10

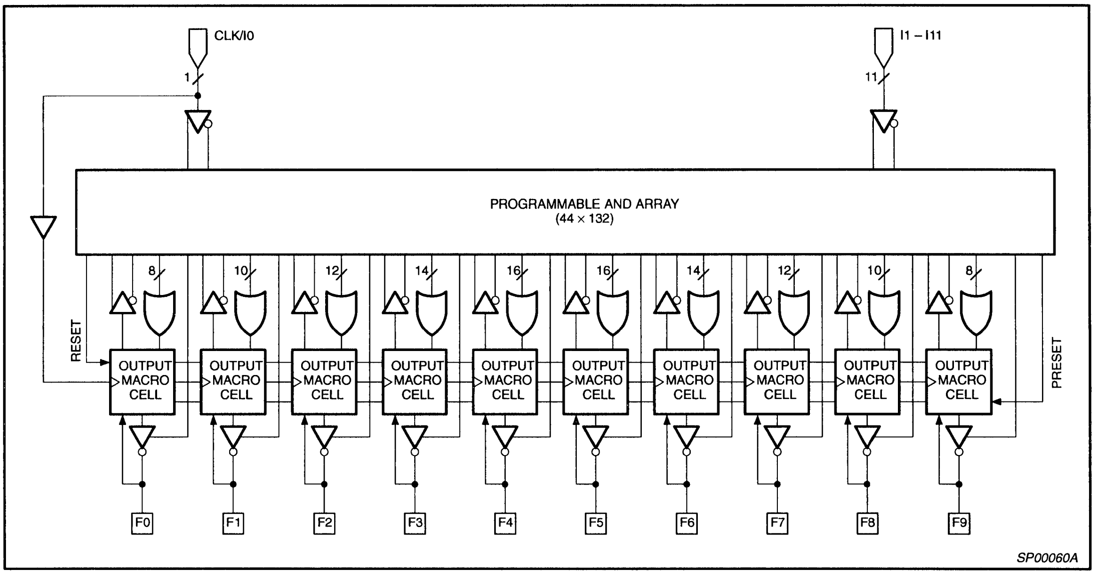


Figure 1. Functional Diagram

FUNCTIONAL DESCRIPTION

The P5Z22V10 implements logic functions as sum-of-products expressions in a programmable-AND/fixed-OR logic array. User-defined functions are created by programming the connections of input signals into the array. User-configurable output structures in the form of I/O macrocells further increase logic flexibility.

ARCHITECTURE OVERVIEW

The P5Z22V10 architecture is illustrated in Figure 1. Twelve dedicated inputs and 10 I/Os provide up to 22 inputs and 10 outputs for creation of logic functions. At the core of the device is a programmable electrically-erasable AND array which drives a fixed-OR array. With this structure, the P5Z22V10 can implement up to 10 sum-of-products logic expressions.

Associated with each of the 10 OR functions is an I/O macrocell which can be independently programmed to one of 4 different configurations. The programmable macrocells allow each I/O to create sequential or combinatorial logic functions with either Active-High or Active-Low polarity.

AND/OR Logic Array

The programmable AND array of the P5Z22V10 (shown in the Logic Diagram) is formed by input lines intersecting product terms. The input lines and product terms are used as follows:

44 input lines:

- 24 input lines carry the True and Complement of the signals applied to the 12 input pins
- 20 additional lines carry the True and Complement values of feedback or input signals from the 10 I/Os

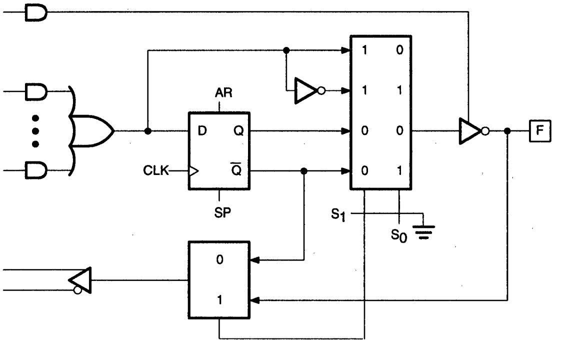
132 product terms:

- 120 product terms (arranged in 2 groups of 8, 10, 12, 14, and 16) used to form logical sums
- 10 output enable terms (one for each I/O)
- 1 global synchronous preset product term
- 1 global asynchronous clear product term

At each input-line/product-term intersection there is an EEPROM memory cell which determines whether or not there is a logical connection at that intersection. Each product term is essentially a 44-input AND gate. A product term which is connected to both the True and Complement of an input signal will always be FALSE, and thus will not affect the OR function that it drives. When all the connections on a product term are opened, a Don't Care state exists and that term will always be TRUE.

Variable Product Term Distribution

The P5Z22V10 provides 120 product terms to drive the 10 OR functions. These product terms are distributed among the outputs in groups of 8, 10, 12, 14, and 16 to form logical sums (see Logic Diagram). This distribution allows optimum use of device resources.

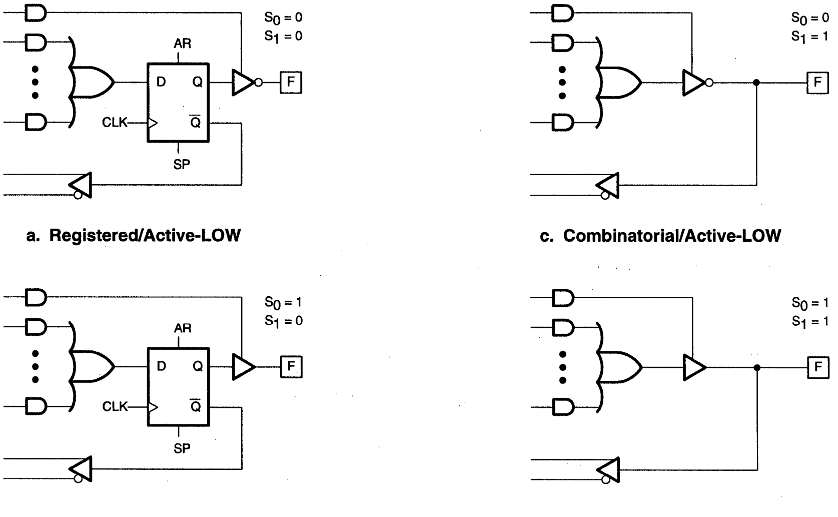


S_1	S_0	OUTPUT CONFIGURATION
0	0	Registered/Active-LOW/Macrocell feedback
0	1	Registered/Active-HIGH/Macrocell feedback
1	0	Combinatorial/Active-LOW/Pin feedback
1	1	Combinatorial/Active-HIGH/Pin feedback

0 = Unprogrammed fuse
1 = Programmed fuse

SP00484

Figure 2. Output Macro Cell Logic Diagram



a. Registered/Active-LOW

c. Combinatorial/Active-LOW

b. Registered/Active-HIGH

d. Combinatorial/Active-HIGH

SP00376

Figure 3. Output Macro Cell Configurations

Programmable I/O Macrocell

The output macrocell provides complete control over the architecture of each output. the ability to configure each output independently permits users to tailor the configuration of the P5Z22V10 to the precise requirements of their designs.

Macrocell Architecture

Each I/O macrocell, as shown in Figure 2, consists of a D-type flip-flop and two signal-select multiplexers. The configuration of each macrocell of the P5Z22V10 is determined by the two EEPROM bits controlling these multiplexers. These bits determine output polarity, and output type (registered or non-registered). Equivalent circuits for the macrocell configurations are illustrated in Figure 3.

Output type

The signal from the OR array can be fed directly to the output pin (combinatorial function) or latched in the D-type flip-flop (registered function). The D-type flip-flop latches data on the rising edge of the clock and is controlled by the global preset and clear terms. When the synchronous preset term is satisfied, the Q output of the register will be set HIGH at the next rising edge of the clock input. Satisfying the asynchronous clear term will set Q LOW, regardless of the clock state. If both terms are satisfied simultaneously, the clear will override the preset.

Program/Erase Cycles

The P5Z22V10 is 100% testable, erases/programs in seconds, and guarantees 1000 program/erase erase cycles.

Output Polarity

Each macrocell can be configured to implement Active-High or Active-Low logic. Programmable polarity eliminates the need for external inverters.

Output Enable

The output of each I/O macrocell can be enabled or disabled under the control of its associated programmable output enable product term. When the logical conditions programmed on the output enable term are satisfied, the output signal is propagated to the I/O pin. Otherwise, the output buffer is driven into the high-impedance state.

Under the control of the output enable term, the I/O pin can function as a dedicated input, a dedicated output, or a bi-directional I/O. Opening every connection on the output enable term will permanently enable the output buffer and yield a dedicated output. Conversely, if every connection is intact, the enable term will always be logically FALSE and the I/O will function as a dedicated input.

Register Feedback Select

When the I/O macrocell is configured to implement a registered function (S1 = 0) (Figures 3a or 3b), the feedback signal to the AND array is taken from the $\overline{Q}$ output.

Bi-directional I/O Select

When configuring an I/O macrocell to implement a combinatorial function (S1 = 1) (Figures 3c or 3d), the feedback signal is taken from the I/O pin. In this case, the pin can be used as a dedicated input, a dedicated output, or a bi-directional I/O.

Power-On Reset

To ease system initialization, all flip-flops will power-up to a reset condition and the Q output will be low. The actual output of the P5Z22V10 will depend on the programmed output polarity. The V_{CC} rise must be monotonic.

Design Security

The P5Z22V10 provides a special EEPROM security bit that prevents unauthorized reading or copying of designs programmed into the device. The security bit is set by the PLD programmer, either at the conclusion of the programming cycle or as a separate step, after the device has been programmed. Once the security bit is set, it is impossible to verify (read) or program the P5Z22V10 until the entire device has first been erased with the bulk-erase function.

TotalCMOS™ Design Technique for Fast Zero Power

Philips is the first to offer a TotalCMOS™ SPLD, both in process technology and design technique. Philips employs a cascade of CMOS gates to implement its Sum of Products instead of the traditional sense amp approach. This CMOS gate implementation allows Philips to offer SPLDs which are both high performance and low power, breaking the paradigm that to have low power, you must accept low performance. Refer to Figure 4 and Table 1 showing the I_{DD} vs. Frequency of our P5Z22V10 TotalCMOS™ SPLD.

Table 1. Typical I_{DD} vs. Frequency

V_{DD} = 5V @ 25°C

FREQUENCY (MHz)	TYPICAL I_{DD} (mA)
1	0.5
10	1.9
20	3.5
30	5.0
40	6.5
50	8.1
60	9.5
70	10.9
80	12.4
90	13.9
100	15.4
110	16.7
120	18.1
130	19.4
140	20.7
150	22.1
160	23.5
170	24.8
180	26.2
190	27.5
200	28.7

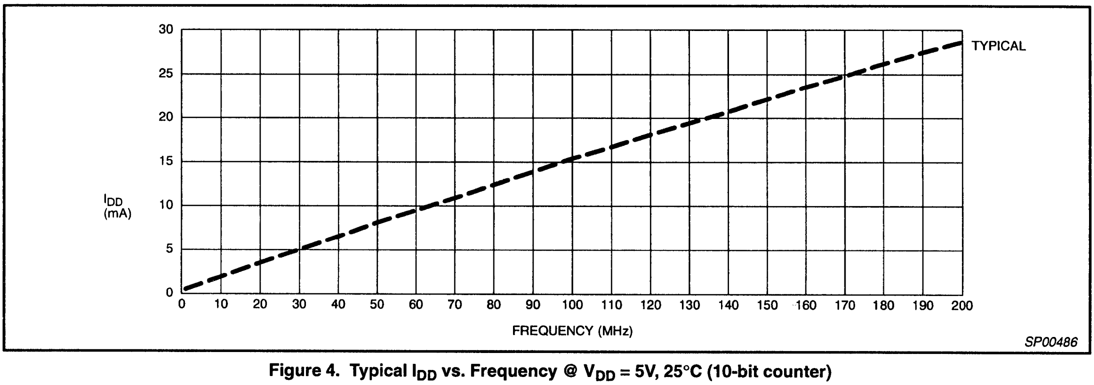


Figure 4. Typical I_{DD} vs. Frequency @ V_{DD} = 5V, 25°C (10-bit counter)

ABSOLUTE MAXIMUM RATINGS[1]

SYMBOL	PARAMETER	LIMITS		UNIT
		MIN.	MAX.	
V_{DD}	Supply voltage	–0.5	7.0	V
V_I	Input voltage	–1.2	V_{DD} + 0.5	V
V_{OUT}	Output voltage	–0.5	V_{DD} + 0.5	V
I_{IN}	Input current	–30	30	mA
I_{OUT}	Output current	–100	100	mA
T_R	Allowable thermal rise ambient to junction	0	75	°C
T_J	Junction temperature range	–40	150	°C
T_{STG}	Storage temperature range	–65	150	°C
ESD	Static discharge voltage (human body)		1000	V

NOTE:

1. Stresses above those listed may cause malfunction or permanent damage to the device. This is a stress rating only. Functional operation at these or any other condition above those indicated in the operational and programming specification of the device is not implied.

OPERATING RANGE

PRODUCT GRADE	TEMPERATURE	VOLTAGE
Commercial	0 to +70°C	5.0 ± 5% V
Industrial	–40 to +85°C	5.0 ± 10% V

DC ELECTRICAL CHARACTERISTICS FOR COMMERCIAL GRADE DEVICES

Commercial: 0°C ≤ T_{amb} ≤ +70°C; 4.75 ≤ V_{DD} ≤ 5.25V

SYMBOL	PARAMETER	TEST CONDITIONS	LIMITS			UNITS
			MIN.	TYP.	MAX.	
V_{IL}	Input voltage low	V_{DD} = 4.75V			0.8	V
V_{IH}	Input voltage high	V_{DD} = 5.25V	2			V
V_I	Input clamp voltage	V_{DD} = 4.75V; I_{IN} = –18mA			–1.2	V
V_{OL}	Output voltage low	V_{DD} = 4.75V; I_{OL} = 8mA			0.5	V
V_{OH}	Output voltage high	V_{DD} = 4.75V; I_{OL} = –4mA	2.4			V
I_I	Input leakage current	V_{IN} = 0 to V_{DD}	–10		10	µA
I_{OZ}	3-Stated output leakage current	V_{IN} = 0 to V_{DD}	–10		10	µA
I_{DDQ}	Standby current	V_{DD} = 5.25V; T_{amb} = 0°C		60	75	µA
I_{DDD} [1]	Dynamic current	V_{DD} = 5.25V; T_{amb} = 0°C @ 1MHz		1	2	mA
		V_{DD} = 5.25V; T_{amb} = 0°C @ 50MHz		10	15	mA
I_{SC}	Short circuit output current	1 pin/time for no longer than 1 second	–30		–100	mA
C_{IN}	Input pin capacitance	T_{amb} = 25°C; f = 1MHz			10	pF
C_{CLK}	Clock input capacitance	T_{amb} = 25°C; f = 1MHz	5		12	pF
$C_{I/O}$	I/O pin capacitance	T_{amb} = 25°C; f = 1MHz			10	pF

NOTE:

1. These parameters measured with a 10-bit up counter, with all outputs enabled and unloaded. Inputs are tied to V_{DD} or ground. These parameters are not 100% tested, but are calculated at initial characterization and at any time the design is modified where current may be affected.

AC ELECTRICAL CHARACTERISTICS FOR COMMERCIAL GRADE DEVICES

Commercial: 0°C ≤ T_{amb} ≤ +70°C; 4.75 ≤ V_{DD} ≤ 5.25V

SYMBOL	PARAMETER	–7		D		UNIT
		MIN.	MAX.	MIN.	MAX.	
t_{PD}	Input or feedback to non-registered output		7.5		10	ns
t_{SU}	Setup time from input, feedback or SP to Clock	3		4		ns
t_{CO}	Clock to output		6.75		8	ns
t_{CF}	Clock to feedback[1]		2		3	ns
t_H	Hold time		0		0	ns
t_{AR}	Asynchronous Reset to registered output		15		15	ns
t_{ARW}	Asynchronous Reset width	5		5		ns
t_{ARR}	Asynchronous Reset recovery time		5		5	ns
t_{SPR}	Synchronous Preset recovery time		5		5	ns
t_{WL}	Width of Clock LOW	3		3		ns
t_{WH}	Width of Clock HIGH	3		3		ns
t_R	Input rise time		20		20	ns
t_F	Input fall time		20		20	ns
f_{MAX1}	Maximum internal frequency[2] $1/(t_{SU} + t_{CF})$	200		143		MHz
f_{MAX2}	Maximum external frequency[1] $1/(t_{SU} + t_{CO})$	103		83		MHz
f_{MAX3}	Maximum clock frequency[1] $1/(t_{WL} + t_{WH})$	167		167		MHz
t_{EA}	Input to Output Enable		9		10	ns
t_{ER}	Input to Output Disable		9		10	ns
Capacitance						
C_{IN}	Input pin capacitance		10		10	pF
C_{OUT}	Output capacitance		12		12	pF

NOTES:

1. These parameters are not 100% tested, but are calculated at initial characterization and at any time the design is modified where frequency may be affected.
2. These parameters measured with a 10-bit up counter, with all outputs enabled and unloaded. Inputs are tied to V_{DD} or ground. These parameters are not 100% tested, but are calculated at initial characterization and at any time the design is modified where frequency may be affected.

DC ELECTRICAL CHARACTERISTICS FOR INDUSTRIAL GRADE DEVICES

Industrial: –40°C ≤ T_{amb} ≤ +85°C; 4.5 ≤ V_{DD} ≤ 5.5V

SYMBOL	PARAMETER	TEST CONDITIONS	LIMITS			UNITS
			MIN.	TYP.	MAX.	
V_{IL}	Input voltage low	V_{DD} = 4.5V			0.8	V
V_{IH}	Input voltage high	V_{DD} = 5.5V	2			V
V_I	Input clamp voltage	V_{DD} = 4.5V; I_{IN} = –18mA			–1.2	V
V_{OL}	Output voltage low	V_{DD} = 4.5V; I_{OL} = 8mA			0.5	V
V_{OH}	Output voltage high	V_{DD} = 4.5V; I_{OL} = –4mA	2.4			V
I_I	Input leakage current	V_{IN} = 0 to V_{DD}	–10		10	µA
I_{OZ}	3-Stated output leakage current	V_{IN} = 0 to V_{DD}	–10		10	µA
I_{DDQ}	Standby current	V_{DD} = 5.5V; T_{amb} = –40°C		70	95	µA
I_{DDD} [1]	Dynamic current	V_{DD} = 5.5V; T_{amb} = –40°C @ 1MHz		1	3	mA
		V_{DD} = 5.5V; T_{amb} = –40°C @ 50MHz		10	20	mA
I_{SC}	Short circuit output current	1 pin/time for no longer than 1 second	–30		–100	mA
C_{IN}	Input pin capacitance	T_{amb} = 25°C; f = 1MHz			10	pF
C_{CLK}	Clock input capacitance	T_{amb} = 25°C; f = 1MHz	5		12	pF
$C_{I/O}$	I/O pin capacitance	T_{amb} = 25°C; f = 1MHz			10	pF

NOTE:

1. These parameters measured with a 10-bit up counter, with all outputs enabled and unloaded. Inputs are tied to V_{DD} or ground. These parameters are not 100% tested, but are calculated at initial characterization and at any time the design is modified where current may be affected.

AC ELECTRICAL CHARACTERISTICS FOR INDUSTRIAL GRADE DEVICES

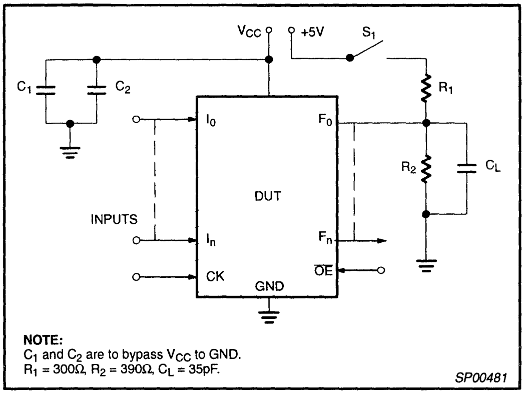
Industrial: –40°C ≤ T_{amb} ≤ +85°C; 4.5 ≤ V_{DD} ≤ 5.5V

SYMBOL	PARAMETER	LIMITS		UNIT
		MIN.	MAX.	
t_{PD}	Input or feedback to non-registered output		10	ns
t_{SU}	Setup time from input, feedback or SP to Clock	5		ns
t_{CO}	Clock to output		8.5	ns
t_{CF}	Clock to feedback[1]		4	ns
t_H	Hold time		0	ns
t_{AR}	Asynchronous Reset to registered output		15	ns
t_{ARW}	Asynchronous Reset width	5		ns
t_{ARR}	Asynchronous Reset recovery time		5	ns
t_{SPR}	Synchronous Preset recovery time		5	ns
t_{WL}	Width of Clock LOW	3		ns
t_{WH}	Width of Clock HIGH	3		ns
t_R	Input rise time		20	ns
t_F	Input fall time		20	ns
f_{MAX1}	Maximum internal frequency[2] $1/(t_{SU} + t_{CF})$	111		MHz
f_{MAX2}	Maximum external frequency[1] $1/(t_{SU} + t_{CO})$	74		MHz
f_{MAX3}	Maximum clock frequency[1] $1/(t_{WL} + t_{WH})$	167		MHz
t_{EA}	Input to Output Enable		11	ns
t_{ER}	Input to Output Disable		11	ns
Capacitance				
C_{IN}	Input pin capacitance		10	pF
C_{OUT}	Output capacitance		12	pF

NOTES:

1. These parameters are not 100% tested, but are calculated at initial characterization and at any time the design is modified where frequency may be affected.
2. These parameters measured with a 10-bit up counter, with all outputs enabled and unloaded. Inputs are tied to V_{DD} or ground. These parameters are not 100% tested, but are calculated at initial characterization and at any time the design is modified where frequency may be affected.

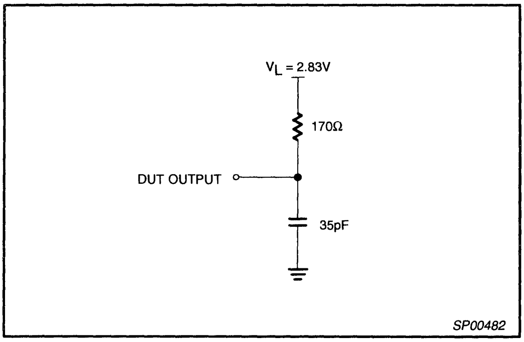
TEST LOAD CIRCUIT



NOTE:
C_1 and C_2 are to bypass V_{CC} to GND.
$R_1 = 300\Omega$, $R_2 = 390\Omega$, $C_L = 35pF$.

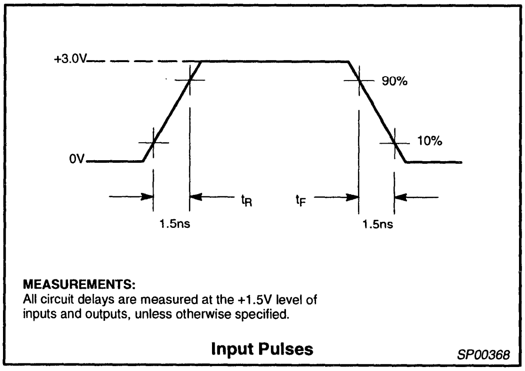
SP00481

THEVENIN EQUIVALENT



SP00482

VOLTAGE WAVEFORM

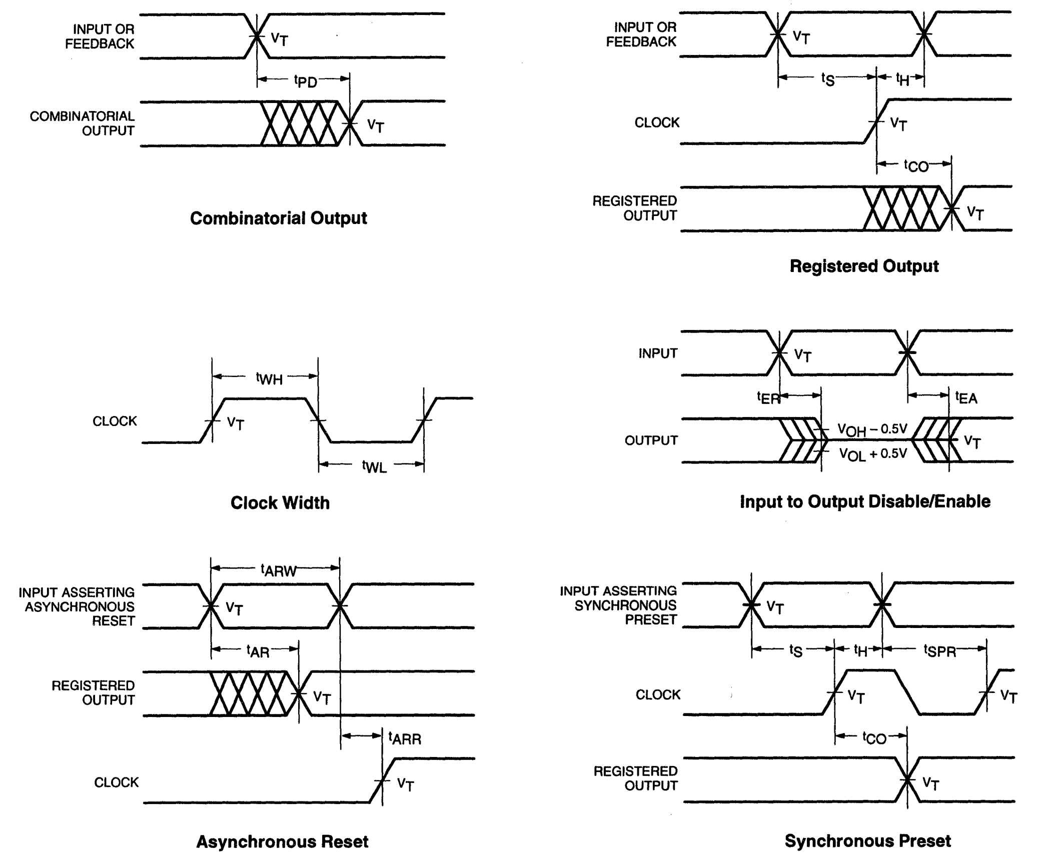


MEASUREMENTS:
All circuit delays are measured at the +1.5V level of inputs and outputs, unless otherwise specified.

Input Pulses

SP00368

SWITCHING WAVEFORMS



Combinatorial Output

Registered Output

Clock Width

Input to Output Disable/Enable

Asynchronous Reset

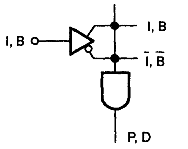
Synchronous Preset

NOTES:

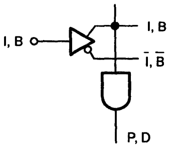
1. $V_T = 1.5V$.
2. Input pulse amplitude 0V to 3.0V.
3. Input rise and fall times 2.0ns max.

SP00065

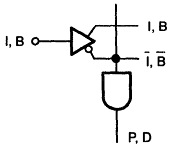
"AND" ARRAY – (I, B)



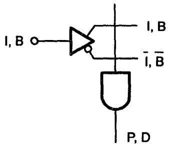
STATE	CODE
INACTIVE[1]	O



STATE	CODE
TRUE	H



STATE	CODE
COMPLEMENT	L



STATE	CODE
DON'T CARE	—

SP00008

NOTE:

1. This is the initial state.

Section 6
Application notes

CONTENTS

Metastability Characteristics for Philips CPLDs – PZ3032

AN055

Introduction

When using a latch or flip-flop in normal circumstances (i.e., when the devices set-up and hold times are not being violated) the outputs will respond to a latch enable or clock pulse within some specified time. These are the propagation delays found in the data sheets. If, however, the set-up and hold times are violated so that the data input is not a clear one or zero, there is a finite chance that the flip-flop will not immediately latch a high or low but get caught half way in between. This is the metastable state and it is manifested in a bistable device by the outputs glitching, going into an undefined state somewhere between a high and low, oscillating, or by the output transition being delayed for an indeterminable time.

Once the flip-flop has entered the metastable state, the probability that it will still be metastable some time later has been shown to be an exponentially decreasing function. Because of this property, a designer can simply wait for some added time after the specified propagation delay before sampling the flip-flop output so that he can be assured that the likelihood of metastable failure is remote enough to be tolerable. On the other hand, one consequence of this is that there is some probability (albeit vanishingly small) that the device will remain in a metastable state forever. The designer needs to know the characteristics of metastability so that he can determine how long he must wait to achieve his design goals. The following information on the PZ3032 is provided to fill this basic need to know how the device operates in situations where metastability may be a problem. It is important in evaluating the reliability of your system that you obtain and evaluate this information from any programmable logic supplier you may be using. Also note that metastable characteristics are different at operating corners of supply voltage and temperature ranges—be wary of data that is presented only at room temperature and nominal V_{CC}. For more detailed background information on metastability, refer to application note AN219, 'A Metastability Primer'. For applications where metastability is a critical concern, the designer may want to investigate Philips ABT22V10-7, which employs patented 'metastable-immune' flip-flops.

PZ3032 Metastable Characteristics

Table 1 presents the metastability data for Philips PZ3032 CoolRunner CPLD.

As shown, Philips provides complete data on the PZ3032's metastable characteristics. While the PZ3032 does not employ Philips patented metastable immune flip-flops, its metastable characteristics are still quite favorable relative to competitive devices. For information on metastable immune PLDs, refer to the datasheet for the ABT22V10-7.

Design Example

Suppose a designer wants to use the PZ3032 for synchronizing asynchronous data that is arriving at 10MHz (as measured by a frequency counter), in a 3.3V system that has a clock frequency of 50MHz, at an ambient temperature of 25°C. The next device in the system samples the output of the PZ3032 8ns after the clock edge to ensure that any metastable conditions that occur have time to resolve to the correct state. The MTBF for this situation can be calculated by using equation below:

$$\mathrm{MTBF} = \frac{e^{(t'/\tau)}}{(T_0\ F_c\ F_1)}$$

In this formula, F_c is the frequency of the clock, F1 is the average input event frequency, and t' is the time after the clock pulse that the output is sampled (t' > Tco). T_0 and τ are device parameters provided by the semiconductor manufacturer (refer to the previous table for the PZ3032 metastability specifications). T_0 and τ are derived from tests and can be most nearly defined as follows: τ is a function of the rate at which a latch in a metastable state resolves that condition. T_0 is a function of the measurement of the propensity of a latch to enter a metastable state. T_0 is also a normalization constant which is a very strong function of the normal propagation delay of the device.

In this situation the F_1 will be twice the data frequency, or 20 MHz, because input events consist of both low and high transitions. Thus in this case F_c is 50MHz, F_1 is 20Mhz, τ is 90.3ps, t' is 8ns, and T_0 is 1.98×10^{13} seconds. Using the above formula the actual MTBF for this situation is 1.51×10^{10} seconds or 478.6 years for the PZ3032.

Table 1. PZ3032 metastability data

	0°C		25°C		70°C	
	τ	T_0	τ	T_0	τ	T_0
3.0V	95.00ps	1.43E+13	101.0ps	4.83E+12	113.0ps	5.91E+11
3.3V	86.70ps	1.53E+13	90.30PS	1.98E+13	103.0ps	1.41E+12
3.6V	80.70ps	2.50E+17	84.10ps	1.17E+15	93.70ps	7.75E+12

Metastability Characteristics for Philips CPLDs – PZ5032

AN056

Introduction

When using a latch or flip-flop in normal circumstances (i.e., when the devices set-up and hold times are not being violated) the outputs will respond to a latch enable or clock pulse within some specified time. These are the propagation delays found in the data sheets. If, however, the set-up and hold times are violated so that the data input is not a clear one or zero, there is a finite chance that the flip-flop will not immediately latch a high or low but get caught half way in between. This is the metastable state and it is manifested in a bistable device by the outputs glitching, going into an undefined state somewhere between a high and low, oscillating, or by the output transition being delayed for an indeterminable time.

Once the flip-flop has entered the metastable state, the probability that it will still be metastable some time later has been shown to be an exponentially decreasing function. Because of this property, a designer can simply wait for some added time after the specified propagation delay before sampling the flip-flop output so that he can be assured that the likelihood of metastable failure is remote enough to be tolerable. On the other hand, one consequence of this is that there is some probability (albeit vanishing small) that the device will remain in a metastable state forever. The designer needs to know the characteristics of metastability so that he can determine how long he must wait to achieve his design goals. The following information on the PZ5032 is provided to fill this basic need to know how the device operates in situations where metastability may be a problem. It is important in evaluating the reliability of your system that you obtain and evaluate this information from any programmable logic supplier you may be using. Also note that metastable characteristics are different at operating corners of supply voltage and temperature ranges—be wary of data that is presented only at room temperature and nominal V_{CC}. For more detailed background information on metastability, refer to application note AN219, 'A Metastability Primer'. For applications where metastability is a critical concern, the designer may want to investigate Philips ABT22V10-7, which employs patented 'metastable-immune' flip-flops.

PZ5032 Metastable Characteristics

Table 1 presents the metastability data for Philips PZ5032 CoolRunner CPLD.

As shown, Philips provides complete data on the PZ5032's metastable characteristics. While the PZ5032 does not employ Philips patented metastable immune flip-flops, its metastable characteristics are still quite favorable relative to competitive devices. For information on metastable immune PLDs, refer to the datasheet for the ABT22V10-7.

Design Example

Suppose a designer wants to use the PZ5032 for synchronizing asynchronous data that is arriving at 10MHz (as measured by a frequency counter), in a 5V system that has a clock frequency of 50MHz, at an ambient temperature of 25ºC. The next device in the system samples the output of the PZ5032 6.5ns after the clock edge to ensure that any metastable conditions that occur have time to resolve to the correct state. The MTBF for this situation can be calculated by using equation below:

$$\mathrm{MTBF} = \frac{e^{(t'/\tau)}}{(T_0\ F_c\ F_1)}$$

In this formula, F_c is the frequency of the clock, F_1 is the average input event frequency, and t' is the time after the clock pulse that the output is sampled ($t' > T_{co}$). T0 and τ are device parameters provided by the semiconductor manufacturer (refer to the previous table for the PZ5032 metastability specifications). T_0 and τ are derived from tests and can be most nearly defined as follows: τ is a function of the rate at which a latch in a metastable state resolves that condition. T_0 is a function of the measurement of the propensity of a latch to enter a metastable state. T_0 is also a normalization constant which is a very strong function of the normal propagation delay of the device.

In this situation the F_1 will be twice the data frequency, or 20 MHz, because input events consist of both low and high transitions. Thus in this case F_c is 50MHz, F_1 is 20Mhz, τ is 69.9ps, t' is 6.5ns, and T_0 is 3.75×10^{14} seconds. Using the above formula the actual MTBF for this situation is 6.47×10^{10} seconds or 2050 years for the PZ5032.

Table 1. PZ5032 metastability data

	0°C		25°C		70°C	
	τ	T_0	τ	T_0	τ	T_0
4.75V	68.40ps	2.87E+14	71.30ps	3.11E+14	76.60ps	2.45E+14
5.0V	66.60ps	8.47E+14	69.90ps	3.75E+14	74.80ps	4.90E+14
5.25V	66.20ps	9.07E+14	68.50ps	1.08E+15	73.70ps	8.38E+14

Altera (AHDL) to Philips (PHDL) design conversion guidelines

AN057

Author: Reno Sanchez, Senior CPLD Applications Specialist

DOCUMENT SCOPE

This document provides information required to translate an Altera Hardware Description Language (AHDL) based design into a Philips Hardware Description Language (PHDL) based design. Designs which should be targeted for conversions are ones in which the customer system needs require one of Philips CoolRunner CPLD advanced features including: dramatic power savings, increased routability with fixed pins, and higher logic density, etc.

This memorandum first gives the key conversion factors which determine if the conversion is feasible. Next, the structural and language syntax differences between the AHDL and PHDL languages are given. Finally, a design example written in both AHDL and PHDL is given.

Terminology

AHDL	Altera Hardware Description Language
CPLD	Complex Programmable Logic Device
CR32	CoolRunner 32 Macrocell CPLD
CR64	CoolRunner 64 Macrocell CPLD
CR128	CoolRunner 128 Macrocell CPLD
DFF	D type Flip–Flop
DFFE	D–type Flip–Flop with Clock Enable
JKFF	JK type Flip–Flop
JKFFE	JK–type Flip–Flop with Clock Enable
OE	Output Enable
PHDL	Philips Hardware Description Language
SRFFE	SR–type Flip–Flop with Clock Enable
TFF	T type Flip–Flop
TFFE	T–type Flip–Flop with Clock Enable

KEY CONVERSION FACTORS

This section gives the key conversion factors which must be addressed before the design conversion is attempted. If these key conversion factors are not met, the design conversion has no possibility of success and the designer should not attempt the conversion. In other words, the factors given in this section must be satisfied or the design conversion is not possible (with fixed pins). If these factors are satisfied, then the designer should attempt the design conversion.

Number of Macrocells

First and foremost, one must ensure that the number of macrocells between an Altera CPLD and a Philips CPLD are equivalent. One should attempt to convert a 32 macrocell Altera CPLD (EPM7032) into a 32 macrocell Philips CPLD (PZ5032). In some cases however, it may be possible to fit a larger macrocell Altera CPLD (i.e. EPM7064) into a smaller Philips CPLD (i.e. PZ5032) if the design is logic (product term) constrained and not macrocell constrained.

Clocking

Before starting the design conversion, one must ensure that all clocks used in the design conform to the Philips CoolRunner pinout. Table 1 gives the Philips CoolRunner clock pinout for the CR32, CR64, and CR128 CPLDs. Please note that CLK0 is a Synchronous clock (must be driven by an external source) while CLK1, CLK2, and CLK3 can be used as either Synchronous clocks (driven by an external source) or Asynchronous clocks (driven by a macrocell equation).

If the design uses any pin as a clock that is different than the ones specified in Table 1, the design will not be pin compatible with the CoolRunner CPLD. However, as long as the number of clocks in the design does not exceed the number of clocks offered by the specific CoolRunner CPLD, the design will probably still fit. What you lose in this case is pin compatibility with the corresponding Altera MAX7000 CPLD. Your decision on whether or not to proceed with the design conversion depends on whether pin compatibility is important. If pin compatibility is a requirement, then there is no reason to convert the design. However, if pin compatibility is not a requirement, then you should convert the design.

If the number of clocks used in the design exceeds the number of clocks contained within the targeted Philips CPLD, then the design will not fit. It may be possible to convert asynchronous clocks in an Altera design to synchronous clocks by modifying the original design. Of course, any design modifications must be tested to insure functionality.

Table 1. Philips CoolRunner™ Clock Pinouts

Package/ Device	Clock Number	Clock Type	44 PLCC	44 TQFP	68 PLCC	84 PLCC	100 PQFP	128 LQFP	160 PQFP
CR32	CLK0	Sync	43	37					
	CLK1	Sync/Async	4	42					
CR64	CLK0	Sync	43	37	67	83	89		
	CLK1	Sync/Async	24	18	36	44	42		
	CLK2	Sync/Async	21	15	33	41	39		
	CLK3	Sync/Async	4	42	4	4	94		
CR128	CLK0	Sync				83	89	114	139
	CLK1	Sync/Async				44	42	53	62
	CLK2	Sync/Async				41	39	50	59
	CLK3	Sync/Async				4	94	119	144

Reset / Preset / Output Enable

The final restriction is reset/preset/oe functionality. The Philips CoolRunner CPLDs have the ability to provide either 36 AND terms or 36 SUM terms (control terms) for each preset/reset/oe function without using any of the macrocells. For example:

```
signal.ar  = A & B & C & ...      – up to 36 terms
signal.oe  = A # B # C # ...      – up to 36 terms
```

However, if a combination of AND and SUM terms are needed to control reset/preset/oe, then a macrocell must be used as an internal node to generate the sum of product signal. For example:

```
node      = (A & B) # (C & D);
signal.ar = node;
```

In the Altera MAX7000 family, a macrocell is not needed for a very limited set of combination of AND and SUM terms which control the reset/preset/oe function.

STRUCTURE TRANSLATION

This section describes the structural differences between the AHDL and PHDL languages.

AHDL Structure

An AHDL file is broken into several sections including: a Header section, a Design Section, a Subdesign section, a Variable section, and a Logic section. An example of the AHDL structure is given in Figure 1. Listed below is a brief description of each section:

- The Header section can contain the following items: Title Statement, a Constant Statement, a Function Prototype Statement, an Include Statement, and an Options Statement.
- The optional Design Section specifies pin, buried logic cell, chip, clique, logic option, and device assignments.
- The Subdesign Section declares the input, output, and bi–directional ports of the file.
- The Variable Section is used to declare any variable used in the Logic Sections. These variables include both external and internal logic.
- The Logic Section specifies the logical operations of the design file and is the body of the Subdesign Section.

PHDL Structure

A PHDL file is broken into four distinct sections: the Header, the Declarations, the Logic Description, and the End. An example of the PHDL structure is given in Figure 2. Listed below is a brief description of each section:

- The Header section contains descriptive information about the design. This section must contain a name for the PHDL file and it can contain title and property statements.
- The Declaration section is where constants, variables, signals, and macro functions are declared and initialized. The start of the declaration section is indicated by the reserved word "Declarations" placed by itself on one line and the declarations that follow.
- The Logic section is where the design is defined by establishing relationships between the inputs and outputs created in the Declaration section. The design may be defined using equations, state machines, or truth tables.
- All PHDL files must close with the reserved word "end".

Key Structural Differences

The key differences between the PHDL and AHDL design file structure is the way pins and outputs of logic functions are defined. In PHDL, if the output of the logic function drives an external pin, the name of function can be the same as the pin name and only needs to be declared once. In AHDL, if the output of the logic function drives an external pin, the name of function must be different from the pin name and the two must be equated in the Logic section. This can be confusing in AHDL because it is not obvious which logic will drive external pins and which are used as buried logic until you examine the entire AHDL file. In PHDL, all node and pin declarations are made in the Declaration section near the front of the PHDL file where the user can easily distinguish between logic which drives external pins and logic which drives internal nodes.

LANGUAGE TRANSLATION

The following subsections give the PHDL equivalent for keywords, operators, and ports for primary inputs. These subsections can serve as a quick reference as you begin creating designs with PHDL.

Keyword cross–reference

Table 2 lists AHDL keywords in the first column and gives the PHDL equivalents in the second column.

Table 2. AHDL – PHDL Keyword Cross Reference

AHDL Keyword	**PHDL Equivalent or Cross Reference**
CASE	CASE
DEVICE IS	DEVICE
ELSE, ELSIF, and END IF	ELSE
END CASE	ENDCASE
AHDL State Machine section	ENDWITH
AHDL Logic Section	EQUATIONS
AHDL Options Statement	FLAG
None	FUSES
AHDL State Machine Section	GOTO
IF	IF
AHDL Subdesign Section	IN
AHDL Primitives	ISTYPE
AHDL Function Prototype Statement	LIBRARY
AHDL Function Prototype Statement	MACRO
DESIGN IS	MODULE
NODE	NODE
@	PIN
WITH STATES	STATE
MACHINE OF BITS	STATE_DIAGRAM
THEN	THEN
TITLE	TITLE
TABLE	TRUTH_TABLE
WHEN	WHEN
AHDL State Machine Section	WITH

Operator Equivalents

Table 3 shows AHDL operators and their PHDL equivalents. Each operator's priority is listed in parentheses beside the symbol for both AHDL and PHDL operators. The operators are similar in AHDL and PHDL.

Table 3. AHDL – PHDL Operator Equivalents

AHDL	PHDL	Operation
– (1)	– (1)	negation
! (1)	! (1)	NOT (invert)
+ (2)	+ (3)	arithmetic addition
– (2)	– (3)	arithmetic subtraction
= = (3)	= = (4)	equal to
! = (3)	! = (4)	not equal to
< (3)	< (4)	less than
< = (3)	< = (4)	less than or equal to
> (3)	> (4)	greater than
> = (3)	> = (4)	greater than or equal to
& (4)	& (2)	AND
!& (4)	none	NAND (invert AND)
$ (5)	$ (3)	XOR (exclusive OR)
!$ (5)	!$ (3)	XNOR (exclusive NOR)
# (6)	# (6)	OR
! # (6)	none	NOR (invert OR)

Dot Extensions

In both AHDL and PHDL, dot extensions are used to connect the features of the macrocell. The ports of an instance of a function are declared in the following format:

<macrocell>.<dot extension>

Table 4 shows the AHDL and PHDL dot extension notations for connections to macrocell logic.

Table 4. AHDL – PHDL Dot Extensions

AHDL	**PHDL**	**Function**
.d	.D	D input to D flip–flop
.t	.T	T input to T flip–flop
.q	.Q	Register feedback
.j	.J	J input to JK flip–flop
.k	.K	K input to JK flip–flop
.s	.S	K input to SR flip–flop
.r	.R	R input to SR flip–flop
.clk	.C	Clock to flip–flop
.prn	.AP	Preset
.clrn	.AR	Clear
TRI	.OE	Tri–state buffer

AHDL has active–low Preset and Clear signals to all flip–flop types: DFF, DFFE, TFF, TFFE, JKFF, JKFFE, and SRFFE. You must explicitly use the TRI primitive when you create a tri–state output. If no port is explicitly used in AHDL and PHDL, the default port on the left–hand side of an equation is the primary data input to the instance of the primitive; the default port on the right–hand side of the equation is the primary output. JK and SR flip–flops always require an explicit port for all inputs.

PITFALLS

This section describes potential pitfalls (language differences) when converting between AHDL and PHDL.

Logic Synthesis

Both PHDL and AHDL do not need to represent a physical "polarity control" for an output. Both compilers automatically apply De Morgan's inversion to all functions as part of logic synthesis. These compilers then compute the most appropriate configuration to obtain the logical behavior that has been defined.

Identifiers

Identifiers are case–sensitive in PHDL designs, while AHDL identifiers (symbolic names) are case insensitive.

Groups

You declare a set of signals in PHDL with an identifier followed by square brackets that enclose a comma–separated list of set members. Subsequent references to the set are made using the identifier only. AHDL notation is slightly different. In AHDL, you declare a group of signals with an identifier followed by an empty pair of square brackets.

The following examples show how a group a seven associated D flip–flops are declared and used in AHDL and PHDL.

AHDL Declaration:

```
countq[6..0] : DFF
```

AHDL Reference:

```
countq[] = countq[] &  incadr & clrqdr & !carryq
            # countq[] & !incadr & clrard
```

PHDL Declaration:

```
countq = [q6..q0];
```

PHDL Reference:

```
countq := countq & incadr & clradr & carryq
            # countq & !incadr & clradr
```

Equations

All flip–flops must be explicitly specified before being used in AHDL. Equations in AHDL are used only to describe combinatorial logic. The explicit declaration of flip–flops in AHDL is somewhat analogous to PHDL's ISTYPE declaration and makes registered assignment operators superfluous.

Comments

Comments in AHDL can span multiple lines and a comment must begin and end with a percent character (%). Comments in PHDL begin with either a quotation mark (") or double slash (//) and continue to the end of the line. Comments in PHDL can span multiple lines with a /* */ enclosing the comments.

Active–Low Specification

The exclamation mark (!) in PHDL is used to declare active–low ports. In AHDL, however, you cannot create active–low ports in the Subdesign Section with the NOT (!) operator. In AHDL, the Logic Section may refer to signals that are either actual device pins or ports that connect to the next higher level of hierarchy. Therefore, the names of ports and their logical sense must agree for the design to compile without errors.

DESIGN CONVERSION EXAMPLE

This section contains a Counter design witch is implemented in AHDL and then re–implemented using PHDL. This should give the reader an idea of how to convert other designs.

AHDL Design Example

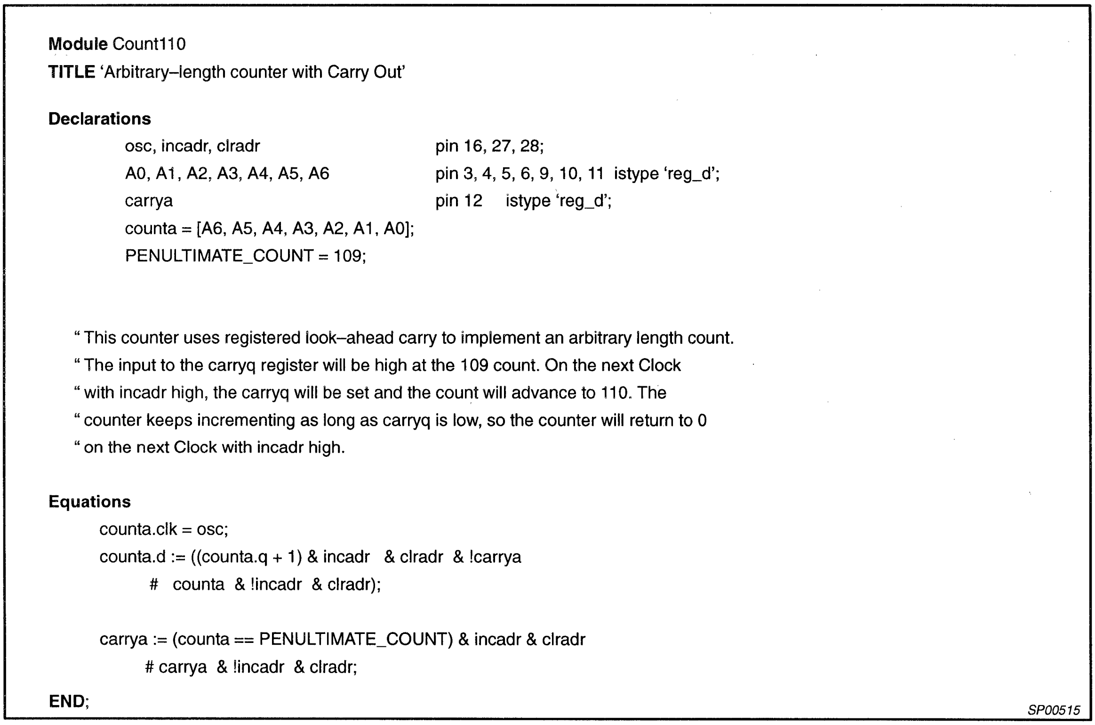
Figure 1 gives a Counter implemented in AHDL.

```
TITLE "Arbitrary–length counter with Carry Out";
CONSTANT PENULTIMATE_COUNT = 109

DESIGN IS "count110"
BEGIN
      DEVICE IS EPM5032"
      BEGIN
            clradr @ 28, incadr @ 27, osc @ 16        : INPUT
            a0 @3, a1 @ 4, a2 @ 5, a3 @ 6             :OUTPUT
            a4 @ 9, a5 @ 10, a6 @ 11                  :OUTPUT
            carrya @ 12                               :OUTPUT
         END;
END;
%-----------------------------------------------------------------------------------
      This counter uses registered look–ahead carry to implement an arbitrary length count.
      The input to the carryq register will be high at the 109 count. On the next Clock
      with incadr high, the carryq will be set and the count will advance to 110. The
      counter keeps incrementing as long as carryq is low, so the counter will return to 0
      on the next Clock with incadr high.
-----------------------------------------------------------------------------------%
SUBDESIGN Count110
(
      osc, incadr, clradr                     :INPUT
      a[6..0], carrya                         :OUTPUT
)
VARIABLE
         carryq, count[6..0]                  :DFF

BEGIN
         countq[].clk = osc;
         countq[]  = (count[] + 1) & incadr  & clradr  & !carryq
                 # countq[]  & !incadr  & clradr;
         a[]  = countq[];
         carryq  = (countq[] == PENULTIMATE_COUNT) & incadr & clradr
                 # carryq  & !incadr  & clradr;
         carrya  = carryq;
END;                                                                       SP00514
```

Figure 1. AHDL Counter Design

PHDL Design Example

Figure 2 gives a Counter implemented in PHDL.



```
Module Count110
TITLE 'Arbitrary–length counter with Carry Out'

Declarations
            osc, incadr, clradr                    pin 16, 27, 28;
            A0, A1, A2, A3, A4, A5, A6             pin 3, 4, 5, 6, 9, 10, 11  istype 'reg_d';
            carrya                                 pin 12     istype 'reg_d';
            counta = [A6, A5, A4, A3, A2, A1, A0];
            PENULTIMATE_COUNT = 109;


    " This counter uses registered look–ahead carry to implement an arbitrary length count.
    " The input to the carryq register will be high at the 109 count. On the next Clock
    " with incadr high, the carryq will be set and the count will advance to 110. The
    " counter keeps incrementing as long as carryq is low, so the counter will return to 0
    " on the next Clock with incadr high.

Equations
        counta.clk = osc;
        counta.d := ((counta.q + 1) & incadr  & clradr  & !carrya
                #  counta  & !incadr  & clradr);

        carrya := (counta == PENULTIMATE_COUNT) & incadr & clradr
                # carrya  & !incadr  & clradr;
END;
```

SP00515

Figure 2. PHDL Counter Design

TECHNICAL SUPPORT

With these guidelines, you should be well on your way to converting designs written in AHDL into design utilizing the PHDL language. This will enable you to take advantage of all the great features the Philips CoolRunner CPLDs offer. If you wish to learn more about PHDL, please refer to the XPLA Designer Users Manual.

This document was authored by Reno Sanchez, Sr. CPLD Applications Specialist. If you need more information, please contact me at 505–858–2790 or call the Philips CPLD Technical Support Line at 1–888–COOLPLD (1–888–2665753) or 505–858–2996; or send email to coolpld@scs.philips.com.

Cadence/Synopsys design flows for targeting Philips CPLDs

AN058

Author Lester Sanders, CPLD Applications Engineer

INTRODUCTION

The Programmable Logic Group of Philips Semiconductor is developing a family of advanced 3-volt and 5-volt complex programmable logic devices(CPLDs). The XPLA series, designated as the PZ5000 - (5-volt) and PZ3000 (3-volt) series devices, is footprint compatible with the Altera 7000 series devices. The principle advantage of Philips CPLDs over all existing CPLDs is that they consume zero static power. The other advantages are 25% higher logic capacity and a better ability to fit logic with fixed pinouts. The PZ5128/PZ3128 are in-system programmable. All devices are all programmable on Data I/O and BP Microsystems programmers.

Minc Inc has developed fitters for the PZ5000/PZ3000 series for up to 128 macrocells. This allows workstation users to target Philips CPLDs within workstation environents. The software is capable of automatically partitioning across multiple CPLDs. Verilog and VHDL models are generated for timing simulation and post fit board-level simulation.

This note provides scripts for using this capability. The Minc fitter can be used with most workstation flows which use VHDL or Verilog from Cadence, Synopsys, Mentor Graphics, and Exemplar Logic. It can be used with Composer and Concept schematic editors from Cadence and Design Architect from Mentor. In this application note, an example of a design flow for simulation with Verilog-XL and synthesis with Synergy and Synopsys is given. This flow can be used with minor edits for VHDL synthesis.

For additional information, telephone Philips Applications Support at 888-coolpld or browse http://www.coolpld.com.The following documentation is available either through the web server or calling (888) coolpld.

PLDesigner-XL User's Guide

Cadence Openbook

Synopsys Online Doc

Synopsys Library Interface for PLDesigner-XL

PZ5000/PZ3000 Series Data Sheets

Design Flows

The software required to target Philips CPLDs depends on the design flow. The software should be installed as provided in Chapter 1 in the PLDesigner-XL User's Guide.

In the steps listed below, $1 is used to represent the design name, and $_tf the testfixture name.Generally there are a number of different methods to design with each set of CAE tools, and scripts will usually vary based on the design.

Synthesis using Synopsys HDL Compiler/Design Compiler

The steps given below do the following:

Setup environment

Generate an edif file using Synopsys

Compile the edif file to a jedec file anp produce a delay-annotated verilog model from the jedec file.

To use Synopsys for Philips CPLDs, the .synopsys_dc.setup in the user's home or project directory should contain the following :

```
designer = “Lester Sanders”
company = “Philips Semiconductors”
search_path = {., /cadappl/packages/synopsys/3.3b/libraries/syn,~} ;

link_library = {minc.db} ;
target_library = {minc.db};
symbol_library = {generic.sdb} ;
bussing_no_ladder = “true”;
edifout_netlist_only = “true”;
edifout_no_array = “true”;
edifout_power_and_ground_representetion = “cell”;
edifout_power_name = “POWER_1”;
edifout_ground_name = “POWER_0”;
```

```
edifout_power_pin_name = "POWER_1";
```

To use Design Compiler, enter dc_shell -f $1.script[1] where $1.script is in the project directory. The contents of $1.script are:

```
read -format verilog $1.v
check_design
set_structure false
set_flatten true
current_design
compile -map_effort low
write -format edif -output $1.edf
exit
```

When this script is complete, the project directory should contain a $1.edf file. The next step is to enter

```
minc.script $1.
```

The contents of minc.script are

```
cp $MINC_PATH/default.pi .
cp $MINC_PATH/default.cst .
mv default.pi $1.pi
mv default.cst $1.cst
make_src $1.edf
plcomp $1.src
#plsim $1.stm
plopt $1.afb
plscan $1
```

[1] At the Albuquerque site, this is done with q.script $1, where q.script is qsh -t 1000 -p synopsys -j dc_shell dc_shell -f $1.script | tee $1.log

```
plfit $1
plfuse $1
pldoc $1
modgen $1.j1 pz3032-8-qfp44 -verilog
```

This produces a jedec file ($1.j1) and a delay annotated verilog model ($1.vo). The jedec file can be used to program a PZ3000 or PZ5000 series device.

As an alternative to the compilation flow above, the minc script verilog2dsl can be used. If this is used, comment the make_src line in minc.script. The details of this flow are provided in Synopsys Library Interface for PLDesigner-XL.

To use verilog2dsl enter

```
verilog2dsl ad_decoder.v -d ad_decoder
```

Design Flow within the Cadence Environment

If Synergy is used to synthesize the design, PIC Designer licenses are needed for Cadence. The pic_lib library is used as the target library. This design flow used below uses a Verilog description of the design.Synergy is used to generate a schematic. This means that the flow cannot be executed (without x routines) using scripts only. The schematic can be either Composer or Concept based, and its generation is automatic.

The flow is as follows:

Enter synergy -verilog -text

Within Synergy, enter the following commands.

```
add_path ~/.caddata/cadence/share/library/minc/data/cds/
add_path ~/.caddata/cadence/share/library/minc/data/verilog/
```

```
select_design $1.v
library -target pic_lib
run_synthesizer
generate -schematic
generate -edif $1 -net
exit
```

Steps 2, 3, and 5 can be put in .cdsinit.

The following section is used to produce a Composer schematic and to write an edif file in temps of pic_lib primitives.

At the shell prompt, enter cd synthesis.run1

At the shell prompt, enter icds &

From CIW, select Design Manager -> Design Flow -> PIC Design to open the pic design flow.

Click Setup and browse to opt library $1

Open schematic by left clicking on Edit Schematic

Check and save the schematic

From PIC Designer, select Edit Schematic -> Translate Schematic in pic designer to produce a $.src file.

At this stage, run minc.script to produce a jedec file. Sinc $1.src has been created, the make_src line in minc.script must be commented.

Selecting a specific Philips CPLD

The Philips CPLD used is specified in the <design>.pi and <design>.cst files. This allows a user to direct PLDesigner to either target a specific device as the PZ3032 or to scan all devices and provide multiple solutions. The use of these files is described in detail in Chapters 14-16 of the PLDesigner-XL User's Guide. To target the PZ3032, the following can be used .

<design>.cst

```
TEMPLATE =  XPLA32_32;
```

<design>.pi

```
DEVICE
TARGET 'TEMPLATE XPLA32_32 TQFP-44-P32';
default ;
END DEVICE;
```

Post Fitting Simulation

The output of modgen command is used to simulate the design after compilation. If the port order of the modgen-created verilog file ($1.vo) and testfixturedo not match, replace the module record in the $1.vo file with module record from the testfixture file. The post-fit verilog model breaks busses into discrete signals, so some edits may be required so that the signals correlate. . Using the revised testfixture, enter

```
verilog $1_tf.v $1.vo
```

Example

An example of a flow using Synopsys dc_shell and the Cadence Verilog-XL simulator is given below. The example is an address decoder.

The verilog source is

```
// address decoder
// lester sanders
// 1/4/96
module ad_decoder (a,io1,mem1,mem2,mem3) ;
input [15:0] a ;
output io1;
output mem1 ;
output mem2 ;
output mem3 ;
wire mem1 ;
wire mem2 ;
wire mem3 ;

assign io1 = (a <= 16'hdfff ) ;
assign mem1 = ((a >= 16'he000) && (a <= 16'he7ff)) ;
assign mem2 = ((a >= 16'hf000) && (a <= 16'hf6ff)) ;
assign mem3 = (a >= 16'hf800) ;
endmodule
```

The code for testfixture ad_decoder_tf.v is

```
module ad_decoder_tf;
reg [15:0] a ;
wire io1 ;
wire mem1 ;
wire mem2 ;
wire mem3 ;
ad_decoder u1 (a,io1,mem1,mem2,mem3) ;
initial begin
a = 16’h0000 ;
end

integer ad_decoder_chann ;
initial begin
ad_decoder_chann= $fopen (“ad_decoder.rpt”) ;
end
initial begin
$fmonitor (ad_decoder_chann,”time %d a=%h io1=%b mem1=%b mem2=%b mem3=%b”,
$time, a,io1,mem1,mem2,mem3) ;
#10 a =   16’h0000 ;
#10 a =   16’hffff ;
#10 a =   16’hc000 ;
#10 a =   16’he010 ;
#10 a =   16’hf600 ;
#10 a =   16’h0000 ;
$fdisplay (ad_decoder_chann,”\nSimulation of address decoder is complete.”) ;
$finish ;
end
endmodule
```

Using this test fixture and source, a functional simulation in Verilog-XL is done by entering

```
verilog ad_decoder_tf.v ad_decoder.v
```

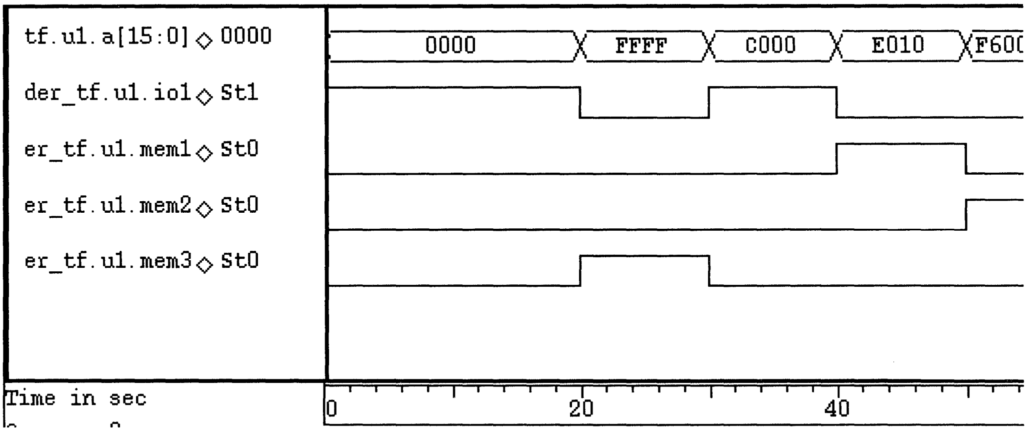
Tho output report file ad_decoder.rpt is

```
time         0 a=0000 io1=1 mem1=0 mem2=0 mem3=0
time        20 a=ffff io1=0 mem1=0 mem2=0 mem3=1
time        30 a=c000 io1=1 mem1=0 mem2=0 mem3=0
time        40 a=e010 io1=0 mem1=1 mem2=0 mem3=0
time        50 a=f600 io1=0 mem1=0 mem2=1 mem3=0
Simulation of address decoder is complete.
```

Users of cWaves can view waveforms of the simulation output by adding thefjollowing lines in ad_decoder_tf.v.

```
$shm_open("ad_decoder.shm");
$shm_probe("AS");
```

The cWaves output resembles the figure below.



To compile this design, Synopsys is used to generate an edif file, and then Minc's PLD Designer is used to produce an jedec file This is done by

```
dc_shell -f ad_decoder.script

minc.script ad_decoder
```

The first few lines of the fitter report file is given below. There is also ad_decoder.doc and otherdesign related files in the project directory.

```
DATE:    Tue May 21 09:11:29 1996
DESIGN:  ad_decoder.afb
DEVICE:  XPLA32_32:1
SUMMARY STATISTICS:
  8 Inputs
  4 Outputs
  0 Tri-states
  0 Nodes
Functions by block:
  A:  4
  B:  0
D Register Macrocells    0
T Register Macrocells    0
Combinatorial Macrocells 4
Single-Pterm Equations   3
Total Pterms Required    6
DEVICE RESOURCE UTILIZATION:
Resource   Available    Used    Remaining      %
DEVICE
Input/Clock Pins:       4        1        3       25
I/O Pins:       32        8        24       25
```

```
Macrocells:        32       4       28       12
Control Pterms:       12       0       12       0
PAL Pterms:       160       6       154       3
PLA Pterms:        64       0       64       0
Signal Sources:       64       5       59       7
Array Inputs:      80       5       75       6
CONTROL BLOCK 'A'
Clock Pins:        4       1       3       25
Blk Clocks:        4       1       3       25
Enable Pterms:        4       0       4       0
SR Pterms:         2       0       2       0
MACROCELL BLOCK 'A'
I/O Pins:       16       8       8       50
Macrocells:        16       4       12       25
AL Pterms:         80       6       74       7
PLA Pterms:        32       0       32       0
Signal Sources:       32       5       27       15
Array Inputs:      40       5       35       12
```

Post-Route Simulation

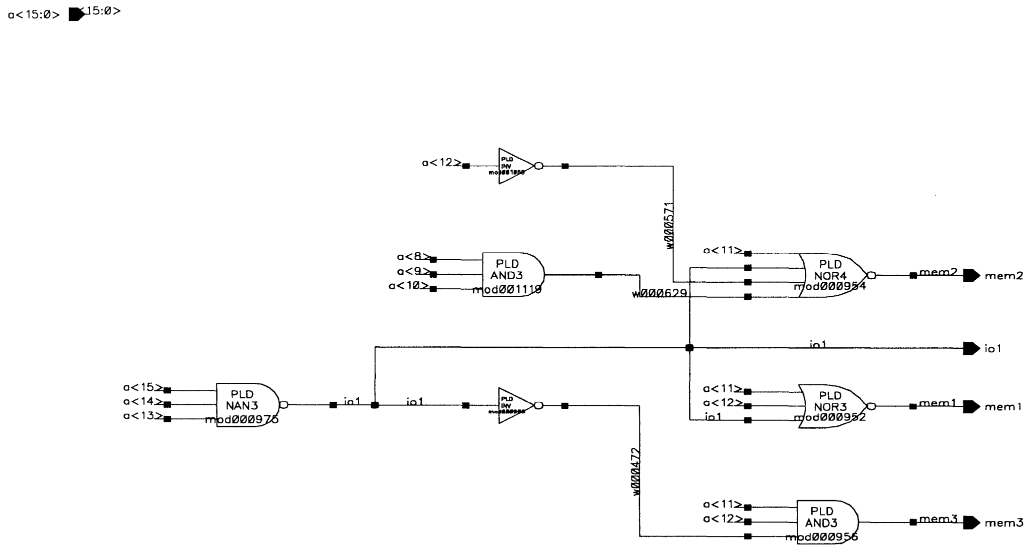
The verilog model generated from the jedec map of the ad_decoder is given below. It can be used to re-simulate the CPLD with timing delays and/or as a module in a PCB simulation. The first section of the verilog model of the address decoder is given below.

```
//----------------------------------------------------------------------
// Verilog Timing Model
// Converted from JEDEC file
// Created by Philips Semiconductors
// Design Name = ad_decoder
// Device Name = pz3032-8-qfp44
// May 22 08:55:02 1996
//-------------------------------------------------------------------
//`timescale 1 ns / 100 ps
module ad_decoder(io1, a_11_, a_10_, a_9_, a_8_, a_15_, a_13_, a_14_, a_12_,
               mem3, mem2, mem1);
input a_11_, a_10_, a_9_, a_8_, a_15_, a_13_, a_14_, a_12_;
inout io1, mem3, mem2, mem1;
wire  io1_D, io1_OE, mem3_D, mem3_OE, mem2_D, mem2_OE, mem1_D, mem1_OE;
parameter tpd0 = 0.0;
parameter tpd1 =  6.5;
parameter tpd2 =  9.0;
parameter tclk =  0.0;

// Equations:

//----( io1 )-------------------------------------------------------------
assign #tpd1 io1_D = !(a_15_ && a_14_ && a_13_);
assign #tpd0 io1_OE = (1);
pxa_bufif1 io1_buf(io1, io1_D, io1_OE);
//----( mem3 )-----------------------------------------------------------
assign #tpd1 mem3_D = (!io1_D && a_12_ && a_11_);
assign #tpd0 mem3_OE = (1);
pxa_bufif1 mem3_buf(mem3, mem3_D, mem3_OE);
//----( mem2 )-----------------------------------------------------------
assign #tpd1 mem2_D = (!io1_D && a_12_ && !a_11_ && !a_10_
|| !io1_D && a_12_ && !a_11_ && !a_8_
assign #tpd0 mem2_OE = (1);
```

```
pxa_bufif1 mem2_buf(mem2, mem2_D, mem2_OE);
//----( mem1 )------------------------------------------------------------
assign #tpd1 mem1_D = (!io1_D && !a_12_ && !a_11_);
assign #tpd0 mem1_OE = (1);
pxa_bufif1 mem1_buf(mem1, mem1_D, mem1_OE);
endmodule
```

Generating a schematic

If Synergy is used for synthesis rather than HDL Compiler/Design Compiler, the schematic produced from the Verilog source resembles that below.



A schematic can also be generated based on the modgen-generted verilog model.

Mentor Graphics Design Flow for targeting Philips Semiconductors CPLDs

AN059

Author Lester Sanders, CPLD Applications Engineer

INTRODUCTION

The Programmable Logic Group of Philips Semiconductor is developing a family of advanced 3-volt and 5-volt complex programmable logic devices(CPLDs). The XPLA series, designated as the PZ5000 - (5-volt) and PZ3000 (3-volt) series devices, is footprint compatible with the Altera 7000 series devices. The principle advantage of Philips CPLDs over all existing CPLDs is that they consume zero static power. The other advantages are 25% higher logic capacity and a better ability to fit logic with fixed pinouts. The first devices, the 32-macrocell PZ3032 and PZ5032, began sampling in Feb 1996. The PZ3064/PZ5064 iand the PZ5128/PZ3128 are scheduled to sample in Q4 1996. The PZ5128/PZ3128 are in-system programmable. All devices are all programmable on Data I/O and BP Microsystems programmers.

Minc Inc has developed fitters for the PZ5000/PZ3000 series for up to 128 macrocells. This allows workstation users to target Philips CPLDs within workstation environments. The software is capable of automatically partitioning across multiple CPLDs. Verilog and VHDL models are generated for timing simulation and post fit board-level simulation.

This note provides scripts for using this capability. The Minc fitter can be used with most workstation flows which use VHDL or Verilog from Cadence, Synopsys, Mentor Graphics, and Exemplar Logic. It can be used with Composer and Concept schematic editors from Cadence and Design Architect from Mentor. In this application note, an example of a design flow for using Mentor Graphics softwarefor simulation and compilation of VHDL designs is given. This flow can be used with minor edits for Verilog synthesis.

For additional information, telephone Philips Applications Support at 888-coolpld or browse http://www.coolpld.com.The following documentation is available either through the web server or telephoning 888 coolpld.

PLDesigner-XL User's Guide

PLDSynthesis II User's Manual

PZ5000/PZ3000 Series Data Sheets

DESIGN FLOWS

The software needed to target Philips CPLDs is available from Mentor Graphics. The software required depends on the design flow. This software should be installed as provided in Chapter 1 in the PLDSII User's Manual.

In the steps listed below, $1 is used to represent the design name, and $_tbthe testbench name. Generally, there are a number of different methods to design using Mentor tools , and scripts will usually vary based on the design.

Mentor Graphics may have access to different tools, including

Synthesis

Autologic - VHDL or Verilog

Synopsys - VHDL or Verilog

Exemplar - VHDL or Verilog

Simulation

Quicksim

QuickVHDL

This note provides a design flow for using the Autologic synthesis tool. The flow and examples for both Quicksim and QuickVHDL simulators are given.

Synthesis using the Autologic Flow

The steps given below do the following:

1. Setup environment
2. Create library and compile VHDL source.
3. Synthesize using Autologic.
4. Open and check the schematic in Design Architect.
5. Generate and check the symbol.
6. Export to PDSII to generate a $1.src file
7. Compile the edif file to a jedec file and produce a delay-annotated VHDL model from the jedec file.
8. Simulate using Quicksim or Quickvhdl.

To use Autologic for Philips CPLDs, the mgc_location_map in the user's home or project directory should contain the following :

```
MGC_LOCATION_MAP_2
#MGC Synlib
$MGC_SYNLIB
/export/home2/lib/synlib
/tmp_mnt/export/home2/lib/synlib
$MGC_GENLIB
/export/home2/lib/gen_lib
/tmp_mnt/export/home2/lib/gen_lib
$MINC_PATH
#$PLDS2_SYN_LIB
```

The flow for Mentor tools is as follows, broken down into Quickvhdl, Autologic, Design Architect, PLDSII, and Quicksim functions.:

<u>Quickvhdl steps</u>

```
mkdir src
cp $1.vhd src
export MGC_LOCATION_MAP=/export/home/lss/designs/mgc/mgc_location_map
qvlib work
qvmap work"<project_directory>/work"
qvcom src/$1.vhd -work work -synth
```

In some cases, testbenches do not compile with the -synth option to the qvcom command.

The flow for a functional simulation in Quickvhdl is as follows. Although recommended, a test bench is not required.

```
qvcom $1_tb.vhd -work designs
qvsim -lib <library> <entity>
File-Load testbench
view wave signals list
```

Autologic steps

To invoke Autologic to use the command line intrface, enter

```
alui -nodisplay
```

The autologic session will include the following steps.

```
opn design -vhdl <path>/src/$1.vhd
env dst gen_lib
syn vhdl <library> <entity> -arch <architecture>
opt area -low
sav design -eddm -model <entity> -map <library> eddm -schematic
quit -f
```

Design Architect steps

To invoke Design Architect, enter

```
da &
```

The steps in Design Architect are :

open sheet in eddm

check schematic

generate and check symbol

re-open schematic

export to pldsII produces *.src in eddm/design/minc directory

PLDSII steps

Either the PLDesigner graphical user interface or a script can be used. For use ot the PLDesigner GUI, see the PLDesigner User Manual. As of this writing, Philips CPLDs are not released with PLDSII. Below is a script for compiling to Philips CPLDs which requires release of only Philips device files. These are available through Mentor Graphics. To run the script, enter

```
minc.script $1
```

The contents of minc.script are

```
cp $MINC_PATH/default.pi .
cp $MINC_PATH/default.cst .
mv default.pi $1.pi
mv default.cst $1.cst
make_src $1.edf
plcomp $1.src
#plsim $1.stm
plopt $1.afb
plscan $1
plfit $1
plfuse $1
pldoc $1
modgen $1.j1 pz3032-8-qfp44 -vhdl
```

This produces a jedec file ($1.j1) and a delay annotated vhdl model ($1.vho). The jedec file can be used to program a PZ3000 or PZ5000 series device[1]. This file can be read into Quickvhdl but there may be modifications to the toriginal testbench for a simulation.

Selecting a Philips CPLD

The Philips CPLD used is specified in the <design>.pi and <design>.cst files. This allows a user to direct PLDesigner to either target a specific device as the PZ3032 or to scan all devices and provide multiple solutions. The use of these files is described in detail in Chapters 14-16 of the PLDesigner-XL User's Guide. To target the PZ3032, the following can be used .

<design>.cst

```
TEMPLATE =  XPLA32_32 or XPLA64_64 ;
```

<design>.pi

```
DEVICE
TARGET 'TEMPLATE XPLA32_32 TQFP-44-P32';
default ;
END DEVICE;
```

Quicksim II simulation steps

From the <design>/eddm/<design> directory, enter quicksim -timing_mode typ plds2_vpt

File-Open design_sheet

Select signals to be monitored in schematic

Invoke Trace, List

[1] Depending on the .pi file, PLDesigner can partition a design across multiple devices, so that $1.j1, $1,j2,... are produced.

Generate Stimuli

Run

DESIGN FLOW EXAMPLE

An example of a flow using Autologic, Design Architect, and either Quicksim II or the Quickvhdl simulator is given below. The example is an 4 to 1 multiplexer of 6-bit busses.

The vhdl source is

```
-- Philips CPLD Applications
-- 6-bit 4 to 1 multiplexer
-- August 20, 1995
library ieee ;
use ieee.std_logic_1164.all ;
entity m41 is
port (a,b,c,d: in std_logic_vector (5 downto 0);
 sel: in std_logic_vector (1 downto 0);
 z : out std_logic_vector (5 downto 0));
end m41 ;
architecture v1 of m41 is
begin
z <= a when sel = “00” else
b when sel = “01” else
c when sel = “10” else
d ;
end v1 ;
```

The testbench for Quickvhdl is

```
-- Philips CPLD Applications
-- m41_tb.vhd
-- 17 oct 1995
library ieee ;
use ieee.std_logic_1164.all ;
entity testbench is end ;
architecture tb of testbench is
component m41
```

```
port (a: in std_logic_vector (5 downto 0);
b: in std_logic_vector (5 downto 0);
c: in std_logic_vector (5 downto 0);
d: in std_logic_vector (5 downto 0);
sel: in std_logic_vector (1 downto 0);
z: out std_logic_vector (5 downto 0));

end component ;
signal a : std_logic_vector (5 downto 0) ;
signal b : std_logic_vector (5 downto 0) ;
signal c : std_logic_vector (5 downto 0) ;
signal d : std_logic_vector (5 downto 0) ;
signal sel : std_logic_vector (1 downto 0) ;
signal z : std_logic_vector (5 downto 0) ;
signal vector_cnt : integer := 1 ;
signaltype test_record is record
a : std_logic_vector (5 downto 0) ;
b : std_logic_vector (5 downto 0) ;
c : std_logic_vector (5 downto 0) ;
d : std_logic_vector (5 downto 0) ;
sel : std_logic_vector (1 downto 0) ;
z : std_logic_vector (5 downto 0) ;
end record ;
type test_array is array(positive range<>) of test_record ;
constant test_vectors : test_array := (
-- a, b, c, d, sel, z
(“000000”,  “000111”, “111000”, “111111”, “00”, “000000”),
(“000000”,  “000111”, “111000”, “111111”, “01”, “000111”),
(“000000”,  “000111”, “111000”, “111111”, “10”, “111000”),
(“000000”,  “000111”, “111000”, “111111”, “11”, “111111”)
);
begin
dut: m41 port map (a => a,
b => b,
c => c,
d => d,
sel => sel,
z => z);
testrun:  process
variable vector : test_record ;
begin
for index in test_vectors'range loop
vector_cnt <= index ;
vector := test_vectors(index);
a <= vector.a ;
b <= vector.b ;
c <= vector.c ;
d <= vector.d ;
sel <= vector.sel ;
wait for 50 ns ;
if ( z /= vector.z) then
error_flag <= '1' ;
assert false
report "Output did not match." ;
else
error_flag <= '0' ;
end if ;
end loop ;
wait ;
end process ;
end;
```

To begin the design flow, from the project directory enter

```
mkdir src
cp <path>/m41.vhd src
cp <path>/m41_tb.vhd src
```

Then create and map the llibrary

```
qvlib work
qvmap work "<path>/work"
```

Now compile the files

```
qvcom src/m41.vhd -work work -synth
qvcom src/m41_tb.vhd -work work -synth
```

Now start autologic for synthesis by entering alui -nodisplay.

```
/export/home/lss/designs/mgc/vhdl/practice3->alui -nodisplay
```

This will cause the following to be displayed. The Autologic session is given in the next three pages. A prompt precedes user input. Lines without a prompt are Autologic output.

```
AutoLogic II Optimizer v8.4_3.2  Tue Jun 27 10:34:53 PDT 1995
Autologic idea license granted
GENIE version 9.16
Loading library -- /export/home2/mgc/pkgs/se_any/userware/default/ipc.ma
Loading library -- /export/home2/mgc/pkgs/syn_any/userware/default/autologic.ma
Loading AutoLogic Timing Driven Layout Library.
Loading library -- /export/home2/mgc/pkgs/syn_any/userware/default/opt_cli.ma
Loading library -- /export/home2/mgc/pkgs/syn_any/userware/default/complib.ma
Loading library -- /export/home2/mgc/pkgs/np_any/userware/default/gc_util.ma
Loading file -- /export/home2/mgc/pkgs/syn_any/userware/default/lo.m
loading /export/home2/mgc/pkgs/np_any/userware/default/synrc.m

    Copyright (c) Mentor Graphics Corporation, 1990-1995
              All Rights Reserved.
          UNPUBLISHED, LICENSED SOFTWARE
       CONFIDENTIAL AND PROPRIETARY INFORMATION
  PROPERTY OF MENTOR GRAPHICS CORPORATION OR ITS LICENSORS

Loading Generic Technology
loading /export/home2/mgc/pkgs/aui_any/userware/default/opt.m
Redefinition of function update_all_vhdl_library_browser
Redefinition of function env_INI
Redefinition of function read_vhdl
>
> opn design -vhdl src/m41.vhd
# opn design -vhdl src/m41.vhd
```

```
alx-hdl - v8.4_1.22
Messages will be logged to file '/export/home2/mgc/tmp/gn842537606.log'
-- Reading file /export/home2/mgc/pkgs/qvhdl_libs/data/standard.vhd
-- Loading package STANDARD into library STD
-- Reading root vhdl file src/m41.vhd
-- Reading file /export/home2/mgc/pkgs/qvhdl_libs/data/std_1164.vhd
-- Loading package STD_LOGIC_1164 into library IEEE
-- Loading entity M41 into library MGC_WORK
-- Loading architecture V1 of M41 into library MGC_WORK
-- Compiling root entity M41(V1)
-- VHDL source successfully analyzed

## Loading $MGC_HOME/lib/autologic.ini
Warning: Overwriting netlist mgc_operators.eq_2u_2u(INTERFACE)
Warning: Overwriting netlist mgc_work.m41(v1)
0
> > env dst gen_lib
# env dst gen_lib
Loading Technology -- /export/home2/lib/synlib/gen_lib
*** gen_lib Library - Version 1.3 - 01Nov93
Loading Cell Definition -- bin/celldef.ma
 Loading Database -- /export/home2/lib/synlib/gen_lib/celldb
Warning: Cannot find auxiliary rules
     Consider creation of auxiliary rules using "do_aux_rules <libname>"
0
> syn vhdl work m41 -arch v1
Synthesizing ....
alx-hdl - v8.4_1.22
Messages will be logged to file '/export/home2/mgc/tmp/synth842537685.log'
-- Reading file /export/home2/mgc/pkgs/qvhdl_libs/data/standard.vhd
-- Loading package STANDARD into library STD
-- Reading file /export/home/lss/designs/mgc/vhdl/practice3/work/m41/m41__alx.vhd into library work
-- Reading file /export/home2/mgc/pkgs/qvhdl_libs/data/std_1164.vhd
-- Loading package STD_LOGIC_1164 into library IEEE
-- Loading entity M41 into library work
-- Reading root vhdl file /export/home/lss/designs/mgc/vhdl/practice3/work/m41/v1__arch__alx.vhd
-- Loading architecture V1 of M41 into library work
-- Compiling root entity M41(V1)
-- VHDL source successfully analyzed

## Loading $MGC_HOME/lib/autologic.ini
Warning: Overwriting netlist mgc_operators.eq_2u_2u(INTERFACE)
Warning: Overwriting netlist work.m41(v1)
netlist:work:m41.v1

> opt area -low
> opt area -low
# opt area -low
DMAG library version 1.0 installed
  Creating new view (mgc_operators eq_2u_2u_Zd6dc662 INTERFACE)
  Creating new view (mgc_operators eq_2u_2u_Z1fb8c43 INTERFACE)
Starting top-level cell: m41 v1

Optimizing /v1
Netlist Count 1 of 1
Starting area optimization, effort:  low, factoring option:  factor
      original area  litweight    current area  litweight
  top:          0      890 ==>          0       890
Starting combinational optimization
      original area  litweight    current area  litweight
  top:          0      890 ==>          0       890
 local:         0      570 ==>          0       570
End of combinational optimization
      original area  litweight    current area  litweight
  top:          0      890 ==>      34000        320
 local:         0      570 ==>      34000          0
End of area optimization, effort low
      original area  litweight    current area  litweight
  top:          0      890 ==>      34000        320
0
```

```
>sav design -eddm -model m41 -map eddm -schematic
```

This step produces alot of output and indicates that a schematic has been written.

```
> quit -f
```

Invoke Design Architect steps by entering

```
da &
```

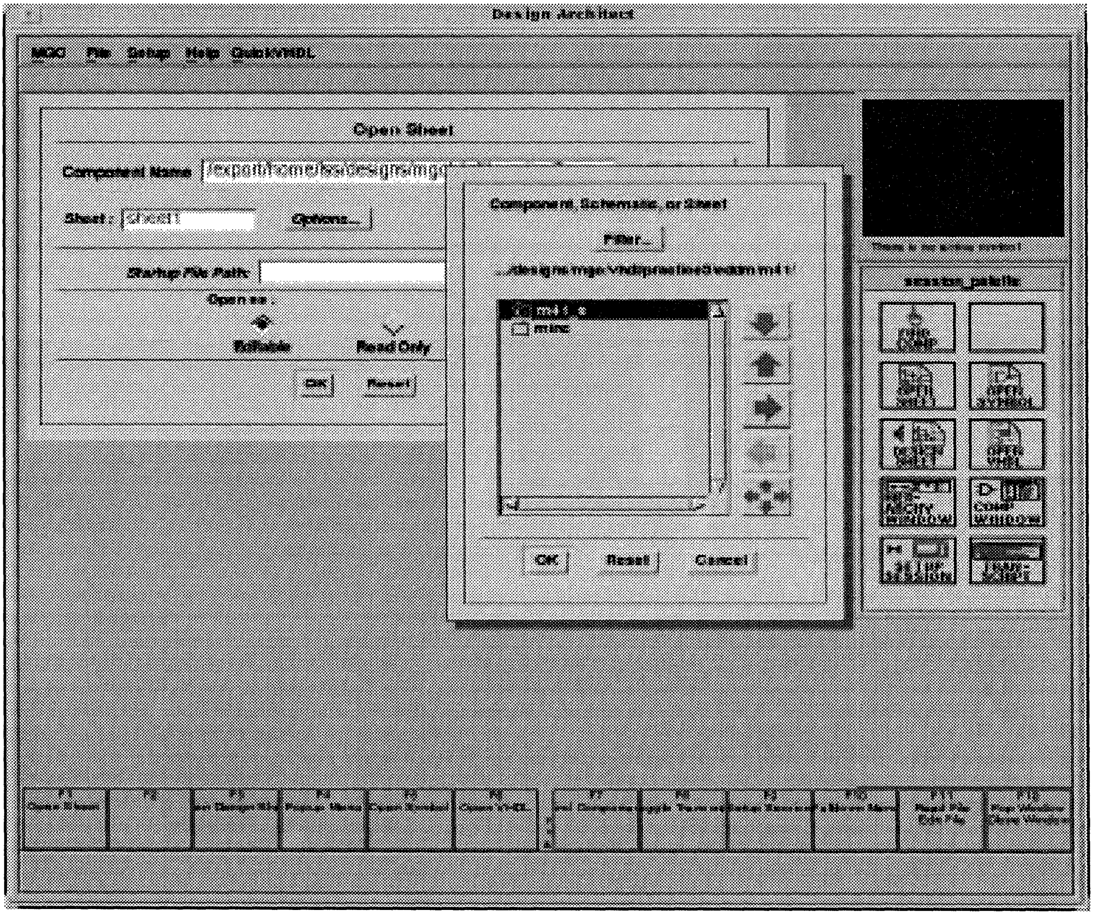


Figure 1. Opening a schematic in Design Architect..

Open the schemaitc by selecting Open Sheet in the palette window and selecting m41_s as shown.

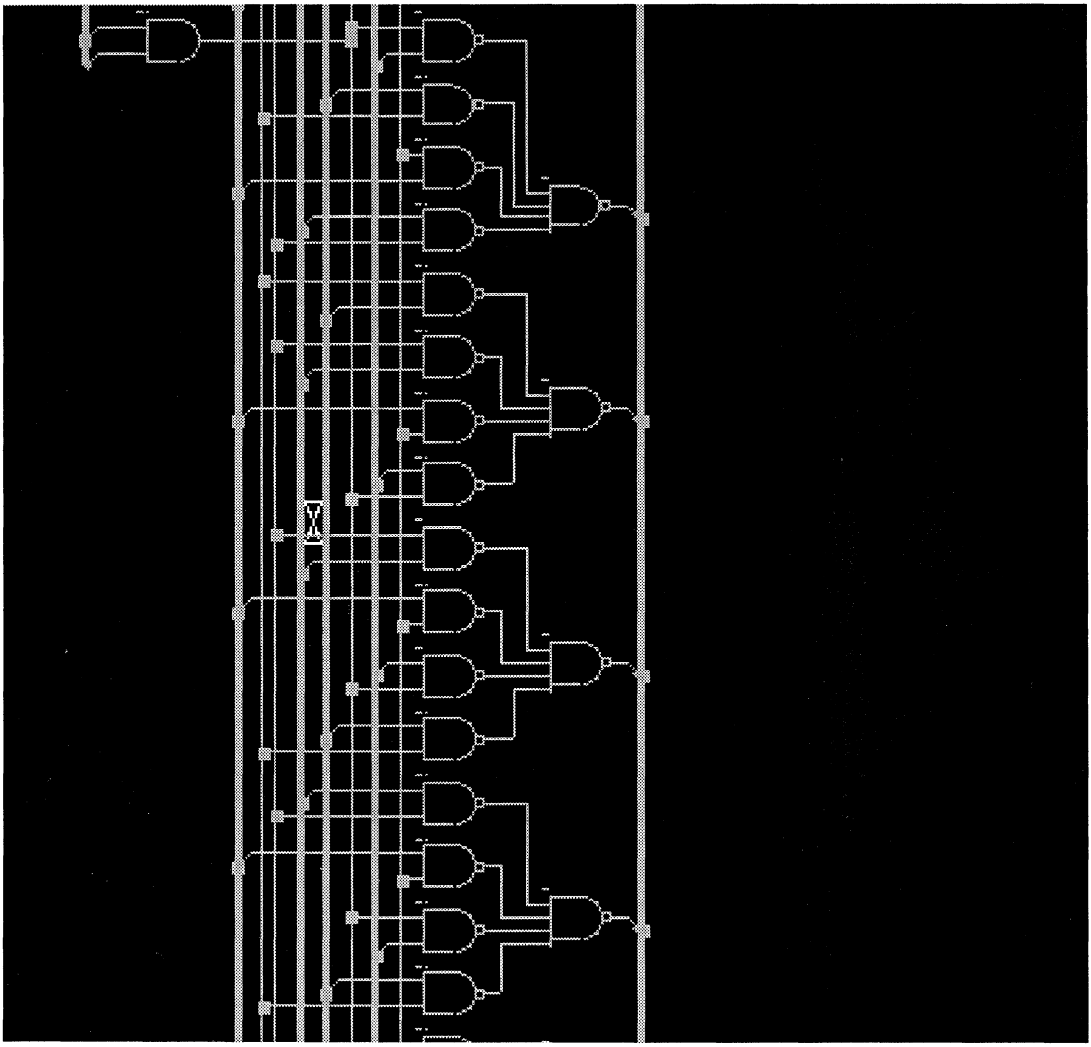
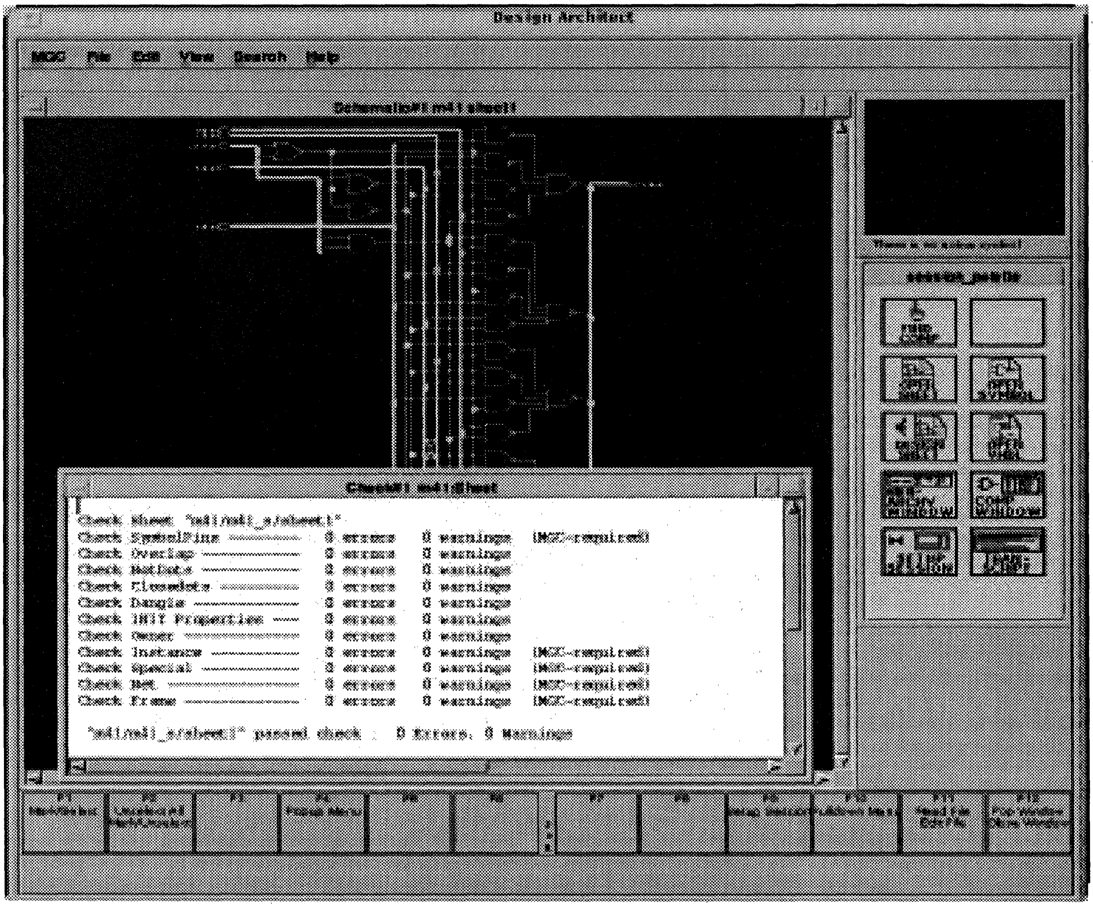


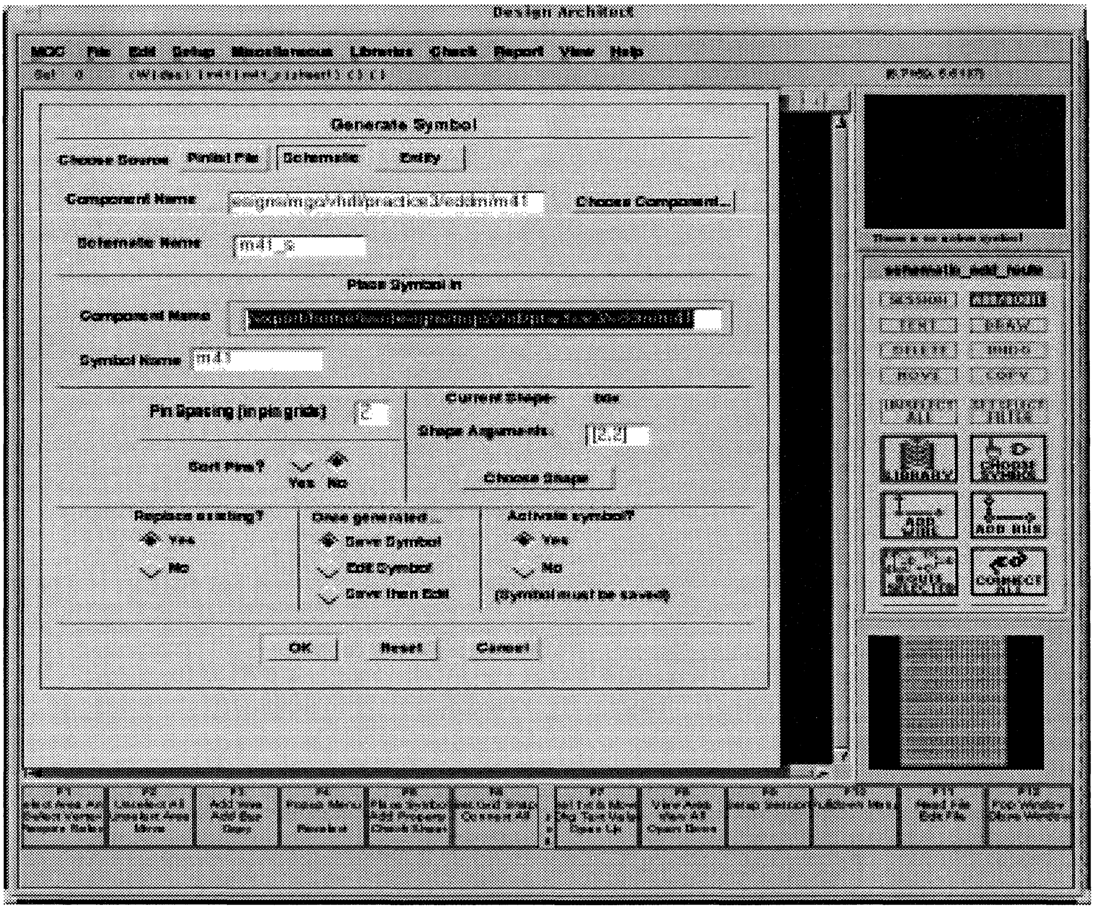
Figure 2. m41_s schematic



```
Check Sheet "m41/m41_s/sheet1"
Check SymbolPins ------------   0 errors   0 warnings   (MGC-required)
Check Overlap ---------------   0 errors   0 warnings
Check Notdots ---------------   0 errors   0 warnings
Check Closedots -------------   0 errors   0 warnings
Check Dangle ----------------   0 errors   0 warnings
Check INIT Properties ----      0 errors   0 warnings
Check Owner -----------------   0 errors   0 warnings
Check Instance --------------   0 errors   0 warnings   (MGC-required)
Check Special ---------------   0 errors   0 warnings   (MGC-required)
Check Net -------------------   0 errors   0 warnings   (MGC-required)
Check Frame -----------------   0 errors   0 warnings   (MGC-required)

 "m41/m41_s/sheet1" passed check :  0 Errors, 0 Warnings
```

Figure 3. Electrical rules check results

With the schematic open, use the pull down menus from the Menu bar Check to ensure that there are no electrical rule violations.



Component Name [illegible]

Schematic Name m41_s

Component Name [illegible]

Symbol Name m41

Pin Spacing (in pin grids) 2

Sort Pins? ○ Yes ● No

Current Shape: box

Shape Arguments [2,2]

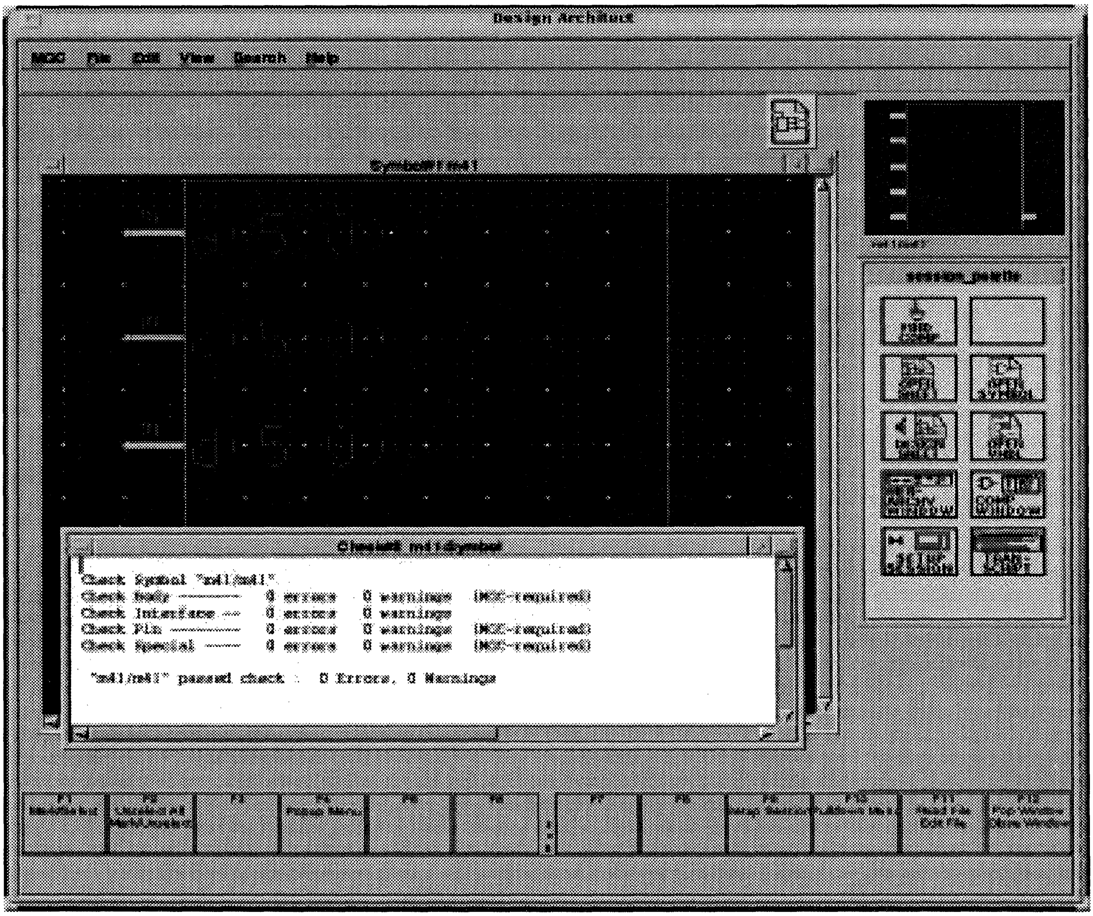
Replace existing? ● Yes ○ No

Once generated... ● Save Symbol ○ Edit Symbol ○ Save then Edit

Activate symbol? ● Yes ○ No (Symbol must be saved)

Figure 4. Generating a symbol

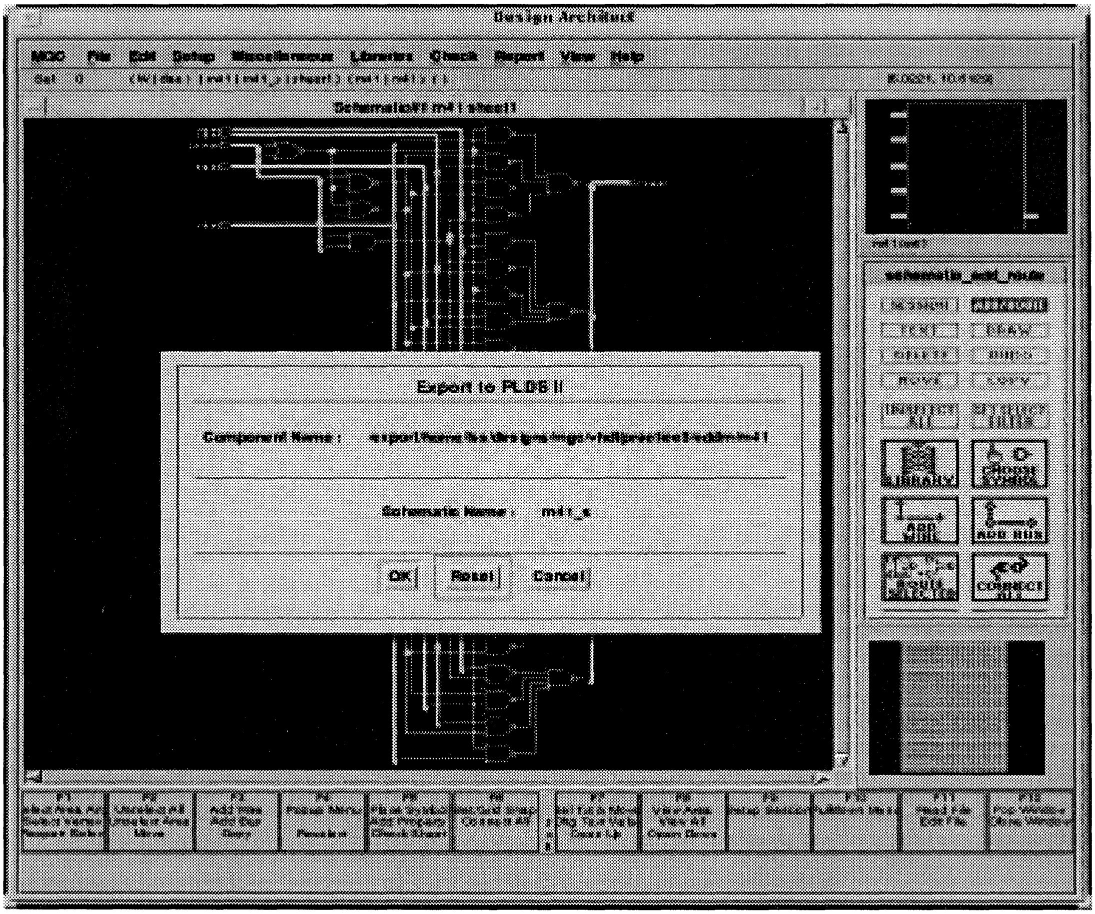
After the schematic is checked without errors, use the Menu bar to select Miscellaneous-Generate Symbol, and change the radio buttons in the dialog box to Replace Existing, Save, and Activare the symbol.



```
Check Symbol "m41/m41"
Check Body --------   0 errors   0 warnings  (MGC-required)
Check Interface --    0 errors   0 warnings
Check Pin --------    0 errors   0 warnings  (MGC-required)
Check Special ----    0 errors   0 warnings  (MGC-required)

 "m41/m41" passed check :   0 Errors, 0 Warnings
```

Figure 5. Check the symbol

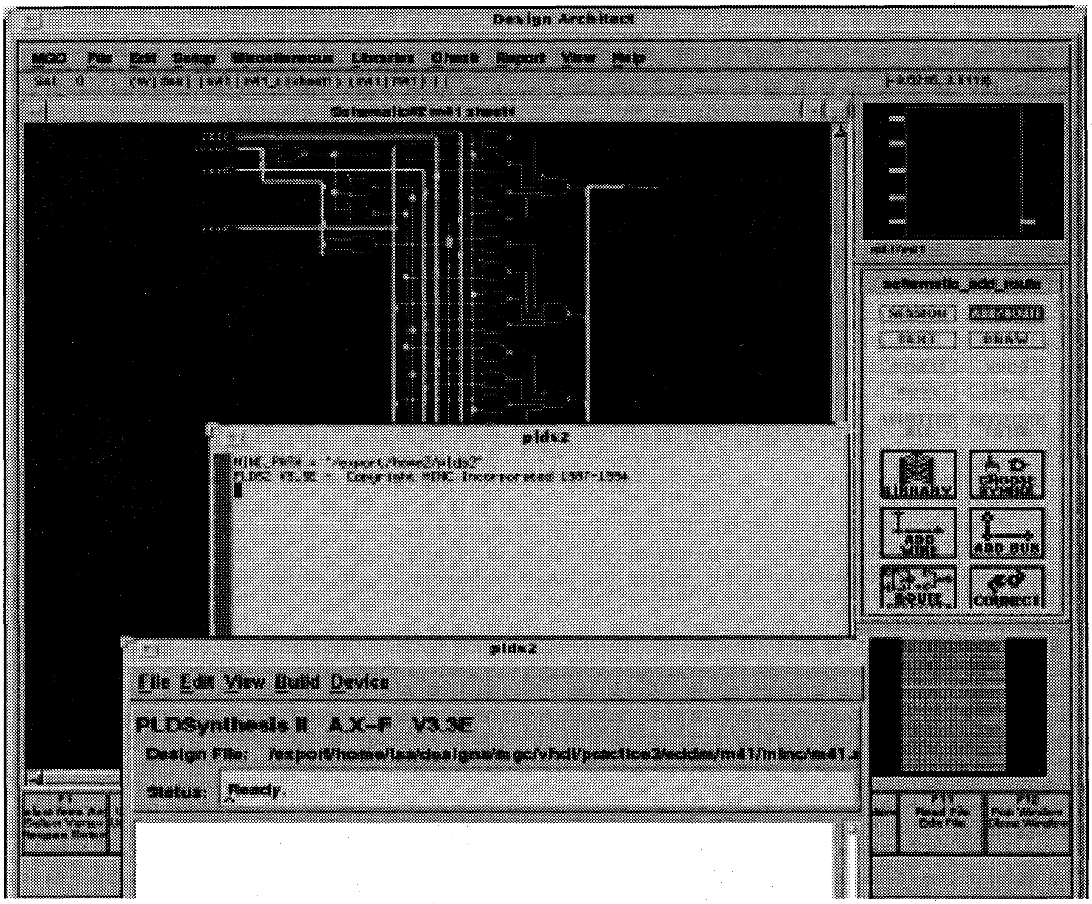
Close the schematic (upper right corner). Use the palette to Open Symbol. From the Menu bar, check the symbol with default registration.



Component Name : [illegible]

Schematic Name : m41_s

Figure 6. Export to PLDSII



```
MINC_PATH = "/export/home2/plds2"
PLDS2 V3.3E - Copyright MINC Incorporated 1987-1994
```

PLDSynthesis II A.X-F V3.3E

Design File: /export/home/isa/designs/mgc/vhdl/practice2/eddie/m41/minc/m41.s

Status: Ready.

Figure 7. Invoking plds2

This shows that PLDSII ha successfully written a.src file which can be compiled to a Philips CPLD. The PLDesigner GUI is shown in the lower part of the screen shot.

Mentor Graphics Design Flow for targeting Philips Semiconductors CPLDs

AN059

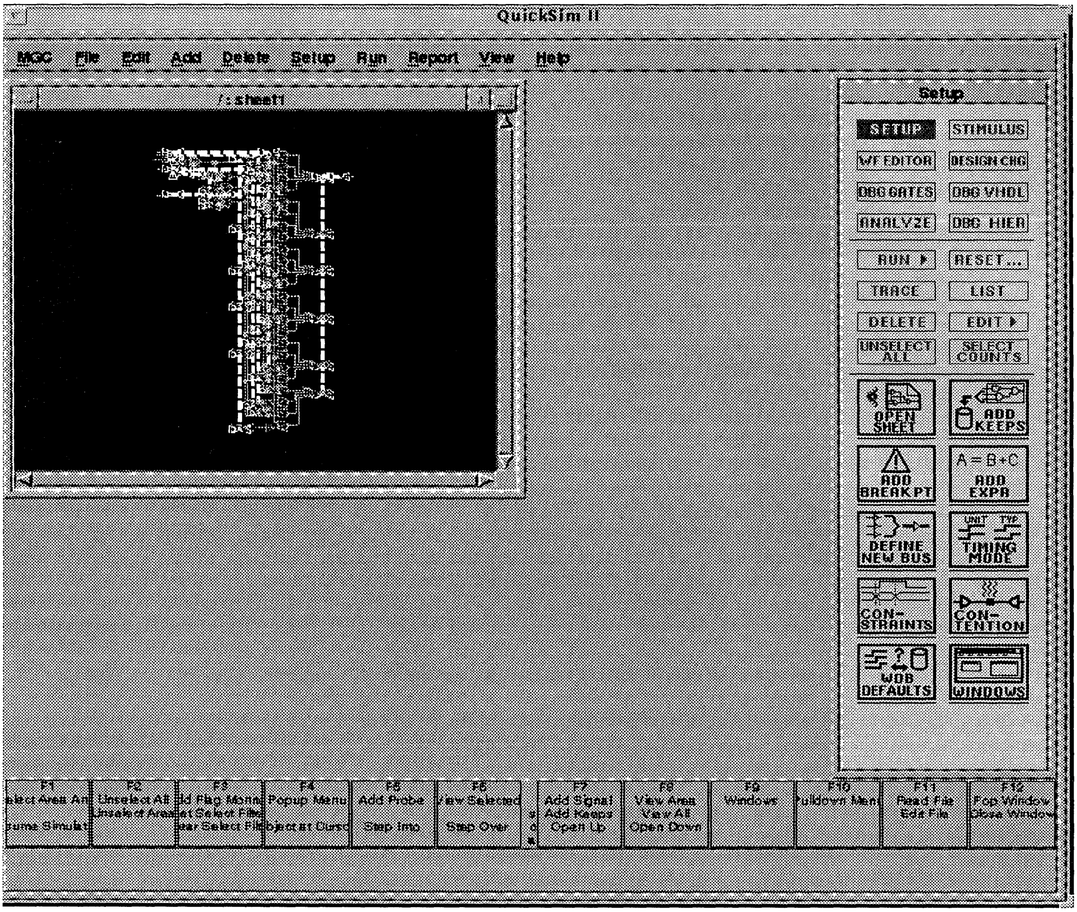
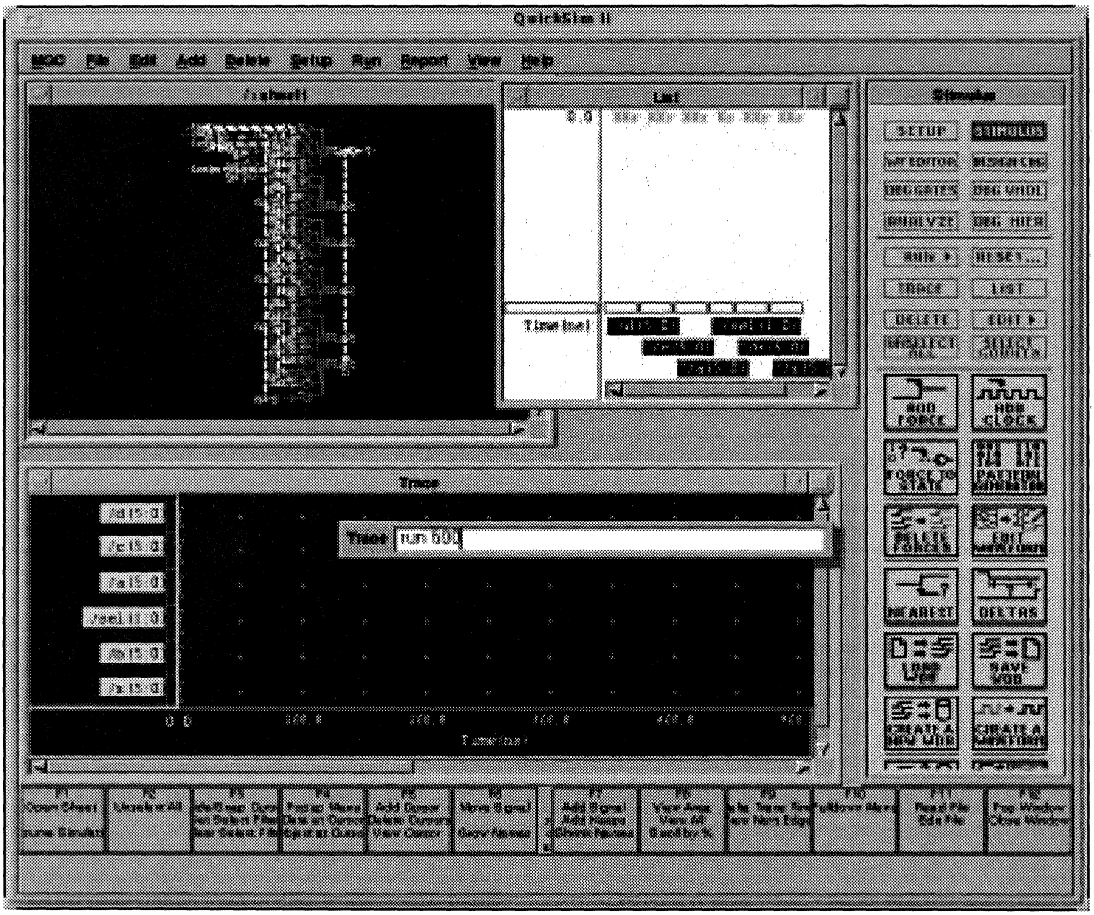


Figure 8. Using Quicksim II

To simulate with Quicksim II, enter quicksim -timing_mode typ plds2_vpt. From the palette menu, select Open Sheet. Using the mouse, elect the nets to monitor in the schematic. Invoke Trace and List from the palette.



Trace run 500

Figure 9. Schematic, Trace, and List windows displayed.

To run a simulation, from the palette menu, enter Stimulus and provide the input stimuli. Enter run 500 to run a simulation.

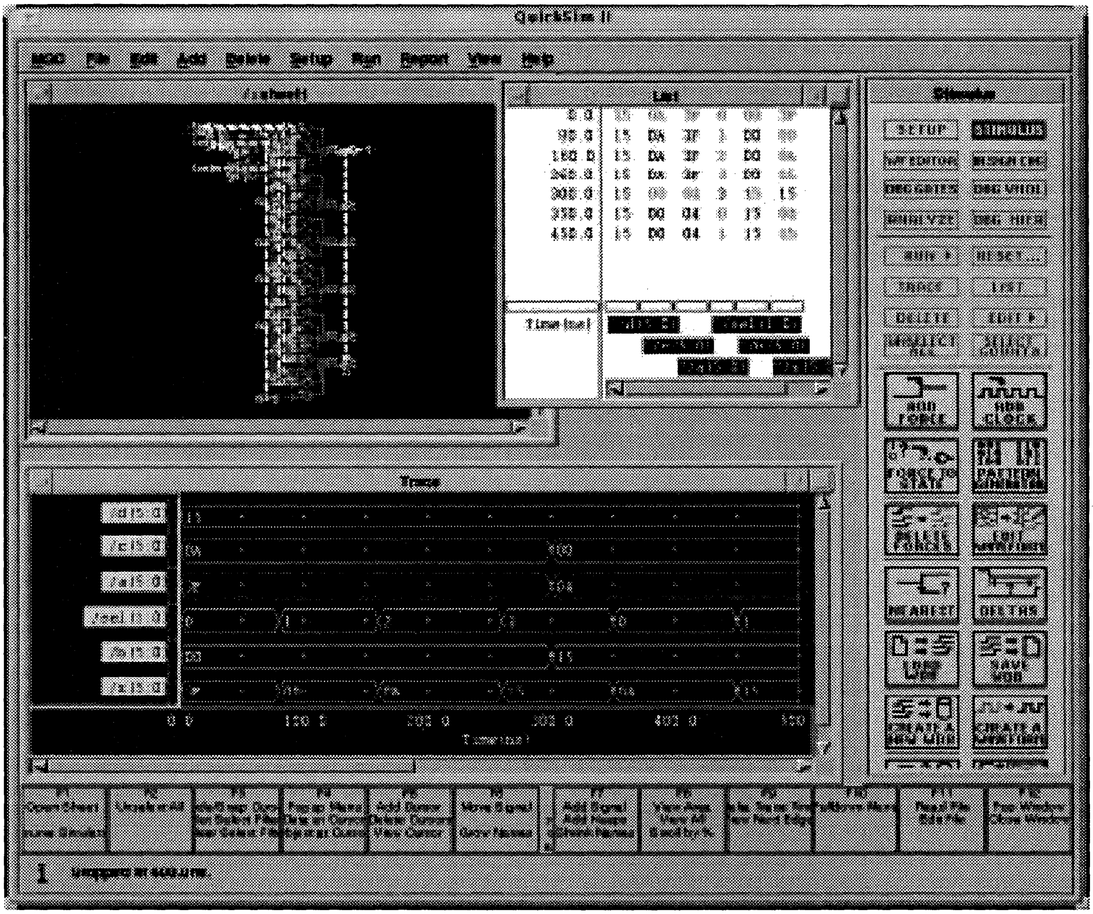


Figure 10. Simulation results

Using Data I/O-Model Technology VHDL tools to target Philips Semiconductors CPLDs

AN060

Author Lester Sanders, CPLD Applications Engineer

INTRODUCTION

This note provides the steps for using the Data I/O-Model Technology VHDL tools to simulate and compile a digital design into either Philips' complex Programable Logic Device (PLDs). This design is generated using schematic and VHDL tools from Data I/O. The design is simulated using the VHDL simulatior from Model Technology.

This note uses VHDL text entry and VHDL simulation. Other methods of design entry include Abel and Verilog text entry and a variety of Verilog and VHDL simulators. Support for Philips PLDs is available on workstaions as well as PCs.

Technical Assistance

Telephone no. 888-coolpld

email - support@coolpld.com

web site - http://www.coolpld.com

Fax on Demand - 800 282-2000

REFERENCES

VHDL User Manual - Synario Universal FPGA Design System Schematic Entry Reference - Synario Capture and ECS Schematic Entry User Manual Project Navigator User Manual

V-System/VHDL Windows User's Manual

INSTALLATION REQUIREMENTS

This design requires the following PC-based CAE tools:

Synario with ECS v 2.1

Synario VHDL Simulator v 2.1 or Model Technology V-System v 4.3g

This design targets the Philips PZ3032 complex programmable logic device. This requires the XPLA fitter, which is available from Data I/O.

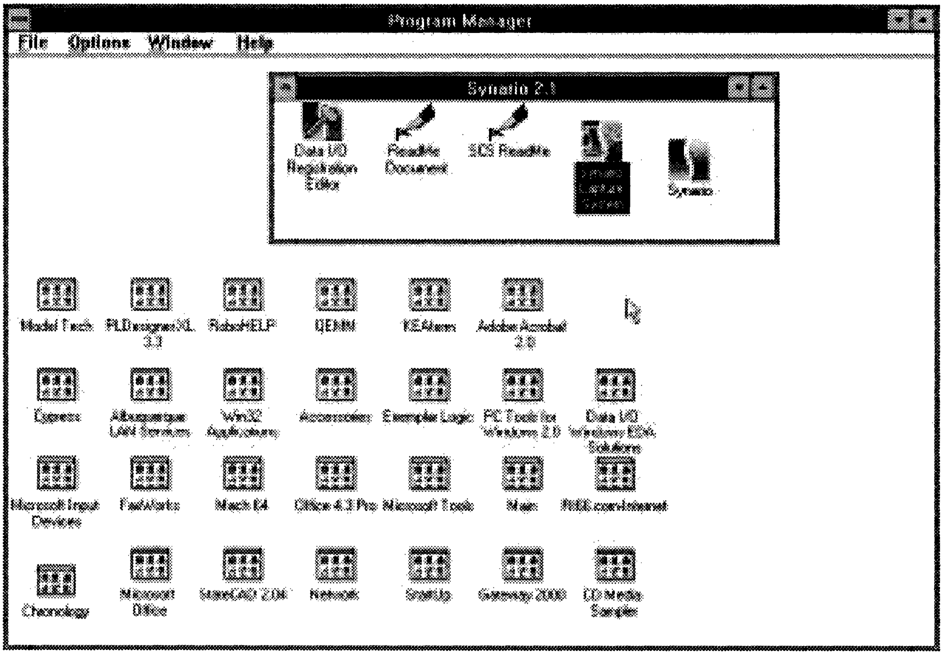
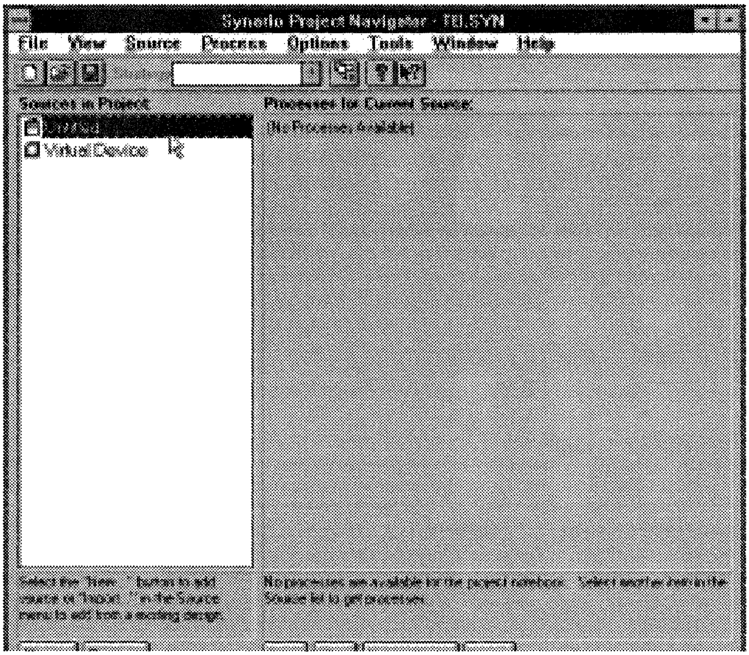


Figure 1: Program Manager and Synario 2.1 Group

To begin, from the Windows Program Manager (Figure 1), double-click on the Synario icon to invoke the Project Navigator. The Project Navigator (Figure 2) window is split into two halves. The left half is associated with sources and the right half is associated with processes. A source is an object such ad a project, a schematic, or a symbol. A process is an action that is performed on a source (compiling, simulating, etc.). When a source in the left window is selected by clicking on it once, the processes that may be performed on that source are listed in the right window. To edit a source or to execute a process, place the cursor on the text and double-click.


Synario Project Navigator - TB.SYN
File View Source Process Options Tools Window Help
Sources in Project:
TB.SYN
Virtual Device
Processes for Current Source:

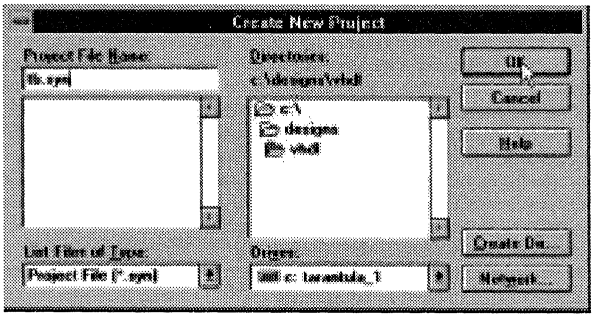
(No Processes Available)

Select the "New..." button to add source or "Import..." in the Source menu to add from a existing design.

No processes are available for the project notebook. Select another item in the Source list to get processes.

Figure 2: Project Navigator window

This design is a state machine for controlling the tail lights on a 1964 Thunderbird. This car had six tail lights. The driver provides turn left (tl), turn right (tr), and brake inputs. When turning, the tail lights are lit sequentially.



Project File Name: tb.syn

Directories: c:\designs\vhdl

c:\
designs
vhdl

List Files of Type: Project File (*.syn)

Drives: c: tarantula_1

Figure 3 Project Navigator Dialog Box

In the dialog box provided, title the project. Double-click on Untitled in the source window and enter “Tbird Tail Lights” for the title.

Select File/New Project and create a directory “c:\designs\vhdl where your project will reside. Use “tb.syn” for the title.

To select a device, double click on Virtual Device, select Philips XPLA CPLDs, then PZ3032, and then OK. Respond Yes when prompted to confirm that you wish to change device kits Schematic Creation

As seen in Figure 2, the Project Navigator menu bar includes File, View, Source, Process, Options, Tools, Windows, Help entries. Create a new schematic by selecting Source/New/Schematic from the Project Navigator

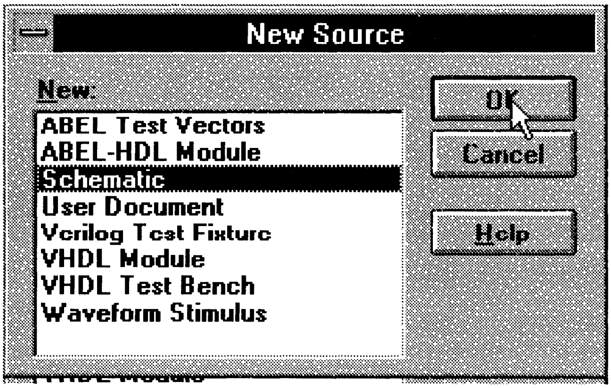
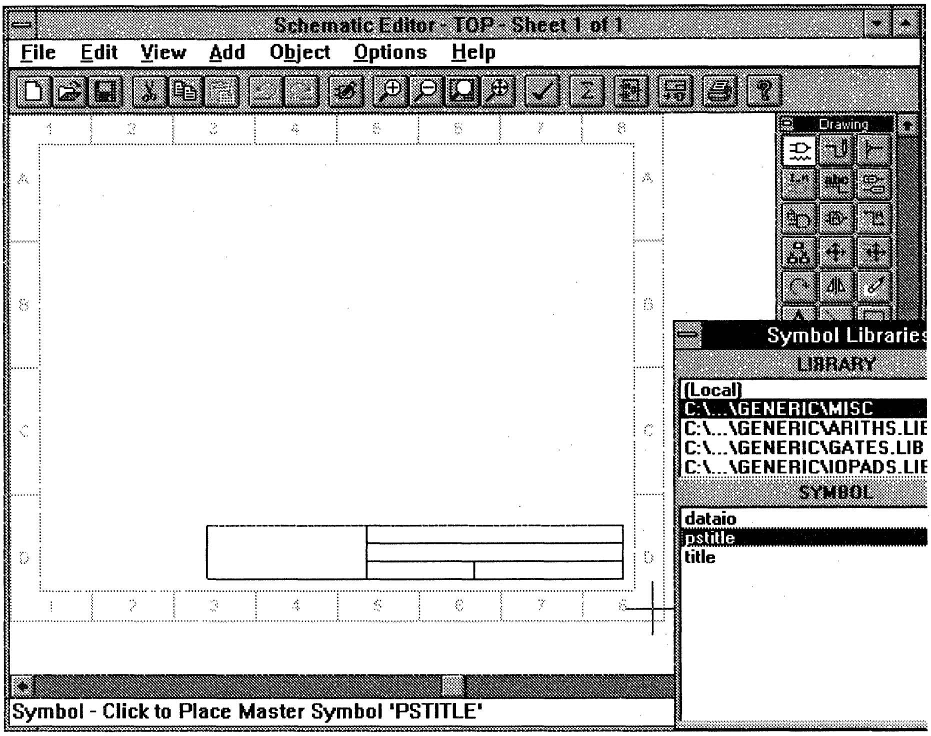


Figure 4: Creating a schematic

As shown in Figure 4, electing New Source causes a dialog box to be displayed. In the dialog box, name the schematic “top.sch” and select OK when finished. A blank schematic page with a sheet border is generated and opened for editing.


Schematic Editor - TOP - Sheet 1 of 1
File Edit View Add Object Options Help
Drawing
Symbol Libraries
LIBRARY
(Local)
C:\...\GENERIC\MISC
C:\...\GENERIC\ARITHS.LIE
C:\...\GENERIC\GATES.LIB
C:\...\GENERIC\IOPADS.LIE
SYMBOL
dataio
pstitle
title

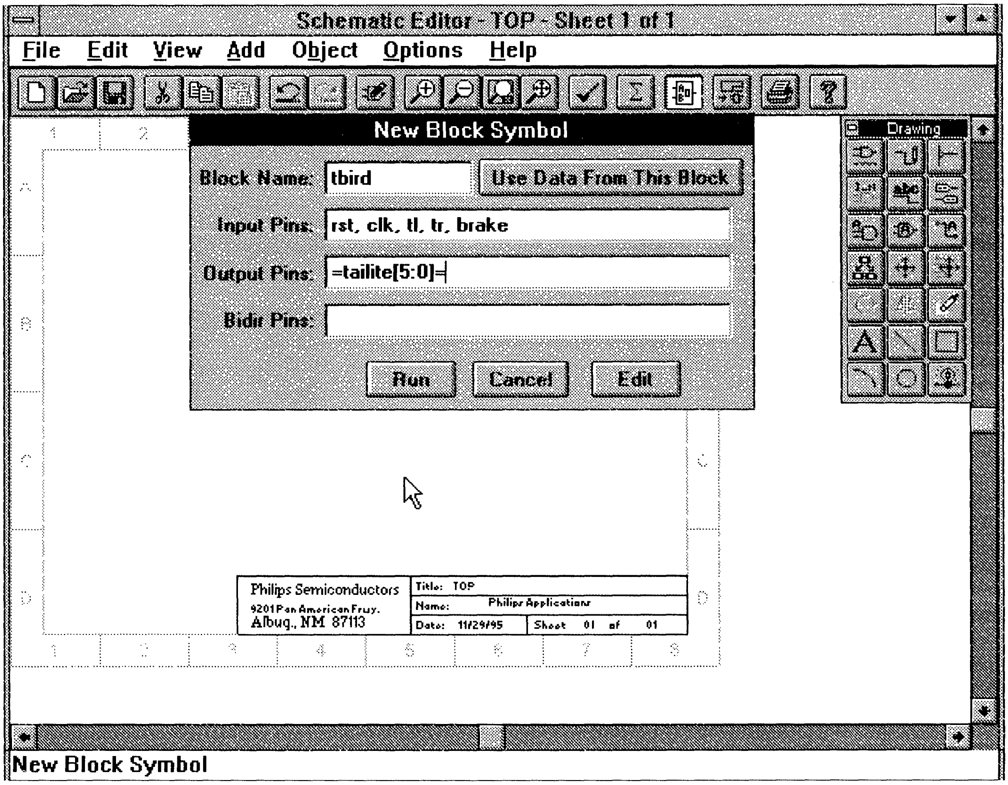
Symbol - Click to Place Master Symbol 'PSTITLE'

Figure 5: Adding a title box in the schematic editor.

To draw the schematic, refer to Figure 5. This example uses the pull-down menus for all actions. The actions may also be performed by selecting the appropriate icon from the Fixed Menu shown on the screen.

Add the title box by selecting Add/Symbol/C:\...\GENERIC\MISC/pstitle from the Schematic editor menu. Position the box in the lower right corner of the screen and click the left mouse button once to place the object. It may be necessary to move the Symbol/Miscellaneous window to properly place the title box. Close the Add/Symbol window.

Select Add/Text from the schematic editor menu and enter your name, which appears at the bottom left corner of the screen. Press Enter when finished typing, position the text in the title box, and click the left mouse button once.



Block Name: tbird

Input Pins: rst, clk, tl, tr, brake

Output Pins: =tailite[5:0]=

Bidir Pins:

Philips Semiconductors
9201 Pan American Frwy.
Albuq., NM 87113

Title: TOP	
Name: Philips Applications	
Date: 11/29/95	Sheet 01 of 01

Figure 6: Schematic Editor - Creating a symbol

To create the top level symbol, select Add/New Block Symbol from the Schematic Editor menu. Fill in the fields as shown in Figure 6. Use the tab key or the mouse to move between fields and select Run.

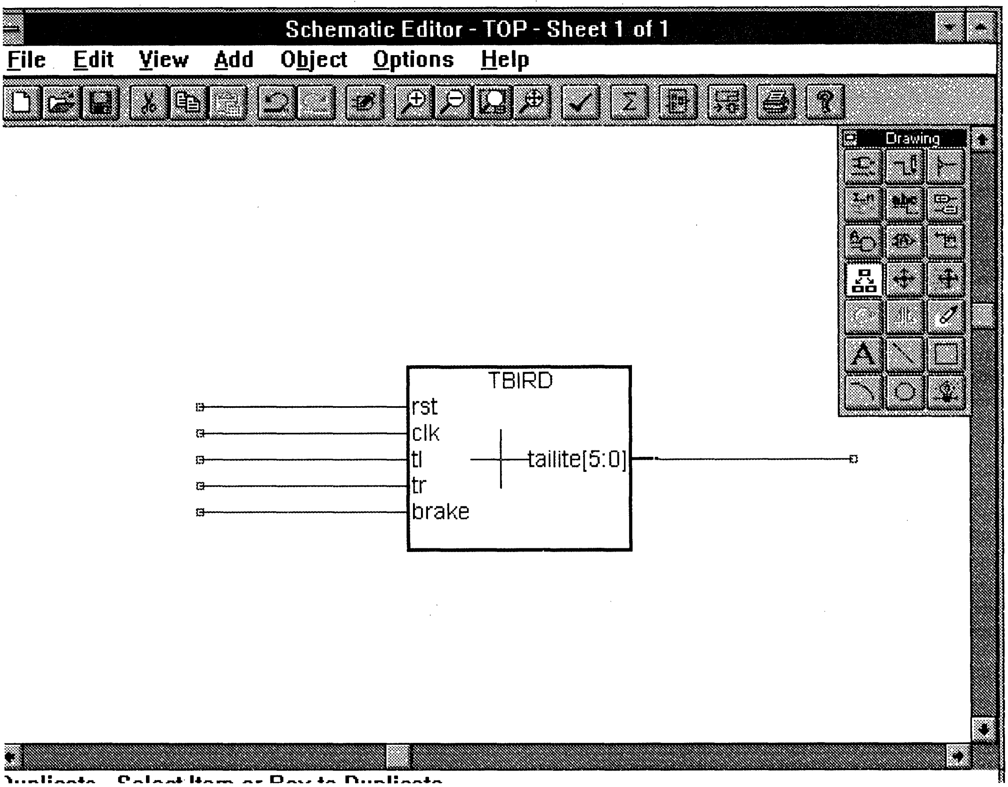
Position the symbol near the center of the schematic and click the left mouse button once to place the object.

Add wires to the symbol by selecting Add/Wire from the Schematic Editor menu It is easier to connect the wires when zoom in on the symbol. To do this, select View/Zoom from the menu to make the cursor turn into a Z. Place the Z in the center of the symbol and click the left mouse button to zoom in around the image. To return to normal view, select Zoom/Full Fit from the menu and click the left button with the cursor anywhere in the schematic.

To draw a wire, position the cursor on the symbol pin and click the left mouse button once.

Drag the wire to the destination and double-click the left button.

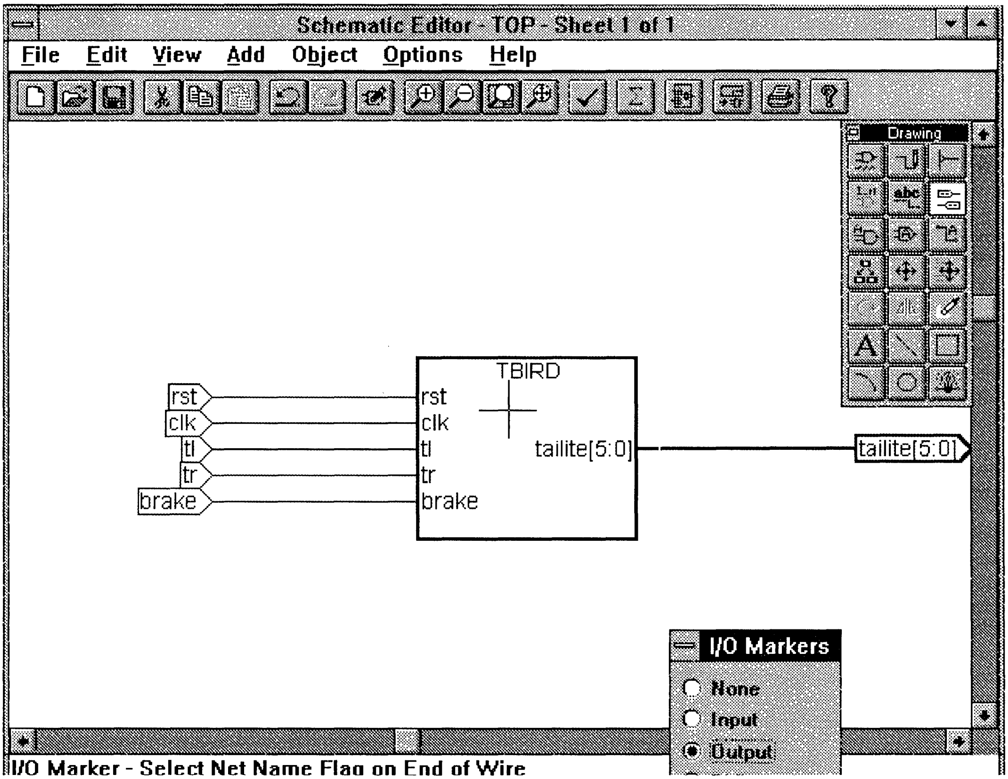
Repeat steps 1 and 2 for each new wire. It is not necessary to select Add/Wire from the menu for each new wire as long as the screen reads "Wire--Click or Drag to Begin Wire" in the bottom left corner. When finished, proceed to the next step.



Duplicate Select Item or Box to Duplicate

Figure 7: Adding wires on a schematic

Label the wires as shown in Figure 8 by selecting Add/Net Name from the Schematic Editor menu: Type in the text to label each wire, position the cursor over the red pad of the wire, and click the left button once. The name is placed near the wire.



I/O Marker - Select Net Name Flag on End of Wire

Figure 8: Labeling nets and adding I/O markers

Repeat this procedure for each wire. As with placing wires, it is not necessary to select Add/Net Name from the menu each time as long as the screen reads “Net Name--Enter Net Name = “ in the bottom left corner.

Add the input and output markers. Select Add I/O Marker from the Schematic Editor menu. A small box listing the marker types will appear on the screen. Verily that “input” is selected in the box.

By holding down the left mouse button, move the mouse to draw a box around all the input wire labels on the left side of the symbol and release the button. Input markers will be added to each wire on the left.

In the dialog box, change the marker type to “output” and draw a box around all the output wire labels on the right. After releasing the button, output markers are added to each wire on the right.

Close the I/O Markers window.

At this point, the schematic should very closely resemble Figure 8. Save the design by selecting File/Save from the Schematic Editor menu. When a schematic is saved, Synario automatically performs an Electrical Rule Check (ERC). An ERC checks that all pins are driven and that all nets have sources and loads. To perform an ERC without saving your work, select File/Consistency Check from the menu.

If no errors are reported, close the Schematic Editor window.

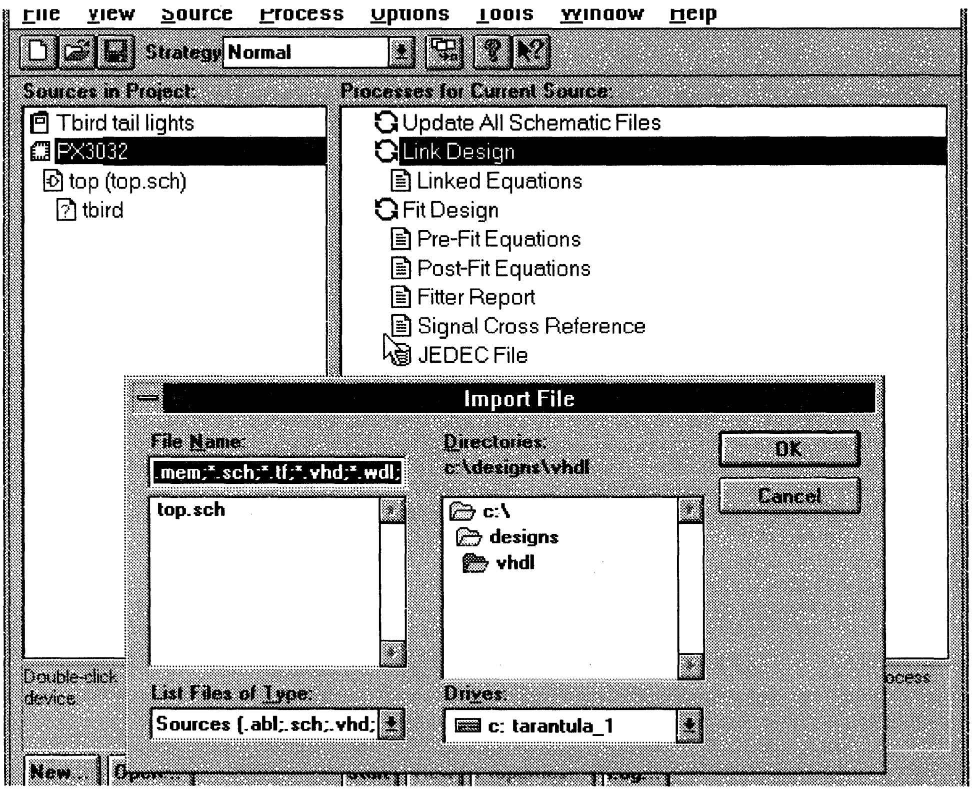


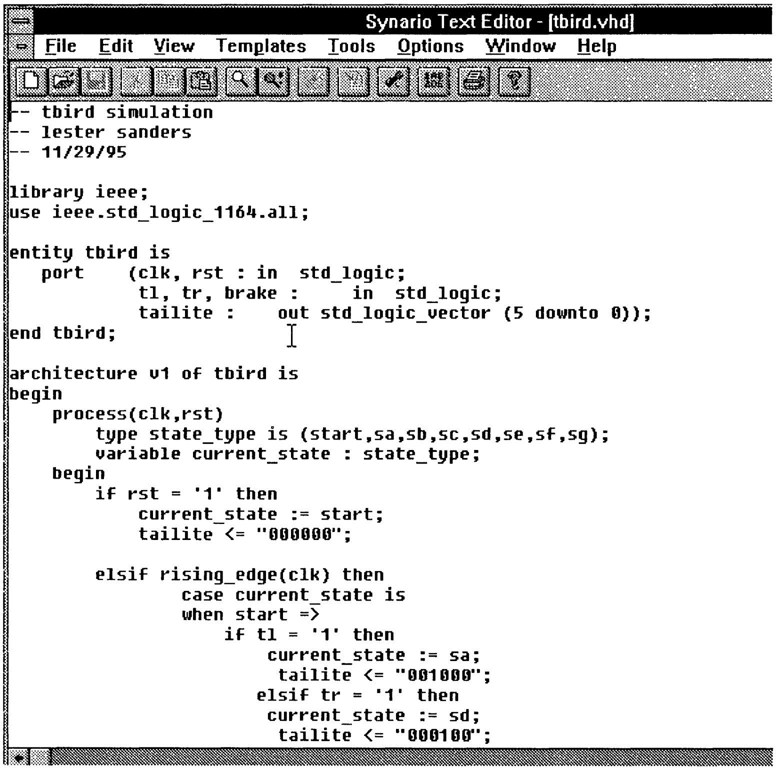
Figure 9: Project Navigator After Schematic and Symbol Creation.

Associate VHDL Code With the Symbol

The Project Navigator window should now look like Figure 9. The sources window (left side) shows the “top.sch” schematic with the tbird source underneath. Notice that the icons to the left of “top.sch” and “tbird” indicate that the schematic is complete but the symbol is not. This is because the functionality of the symbol is not defined. Ordinarily at this point, a designer would either construct a schematic or a textual description of the functionality of the symbol. For this design, an existing VHDL model for the “tbird” symbol is I imported into the design.

From the Project Navigator menu bar, select Source/Import, and use the menus to navigate to the “c:\designs\vhdl” directory. Select the file “tbird.vhd” and press OK. The VHDL text will now be loaded into the tbird symbol, and the icon should now show that the symbol design is complete.

Save the project by selecting File/Save from the Project Navigator menu.


Synario Text Editor - [tbird.vhd]
File Edit View Templates Tools Options Window Help

```
-- tbird simulation
-- lester sanders
-- 11/29/95

library ieee;
use ieee.std_logic_1164.all;

entity tbird is
   port    (clk, rst : in  std_logic;
            tl, tr, brake :    in  std_logic;
            tailite :    out std_logic_vector (5 downto 0));
end tbird;

architecture v1 of tbird is
begin
    process(clk,rst)
        type state_type is (start,sa,sb,sc,sd,se,sf,sg);
        variable current_state : state_type;
    begin
        if rst = '1' then
            current_state := start;
            tailite <= "000000";

        elsif rising_edge(clk) then
                case current_state is
                when start =>
                    if tl = '1' then
                        current_state := sa;
                         tailite <= "001000";
                       elsif tr = '1' then
                        current_state := sd;
                         tailite <= "000100";
```

Figure 10: VHDL Source

The VHDL code may be viewed in either of two different methods. One is to double-click on the icon to the right of the tbird symbol in the source window. A text editor displaying the VHDL code (Figure 10) will appear.

The second method is to invoke the Hierarchical Navigator and display the model by pushing into the tbird symbol. Highlight the top.sch entry in the source window. The process window should show several actions that may be performed on top.sch.

Double-click on “Navigate Hierarchy” in the process window. The Hierarchy Navigator (Figure 8) displays the “tbird” symbol. Select View/Push/Pop from the Hierarchical Navigator menu, position the cross-hairs in the “tbird” symbol, and click the left button once. The VHDL code is displayed.

Note: There are short cut keys that can be used to execute menu functions without having to go to the menu. These keys are displayed to the right of the menu command when the menu is pulled down. Select View from the Hierarchical Navigator and note that pushing F2 would do the same thing as selecting View/Push. To close the menu without taking any action, click on View again.

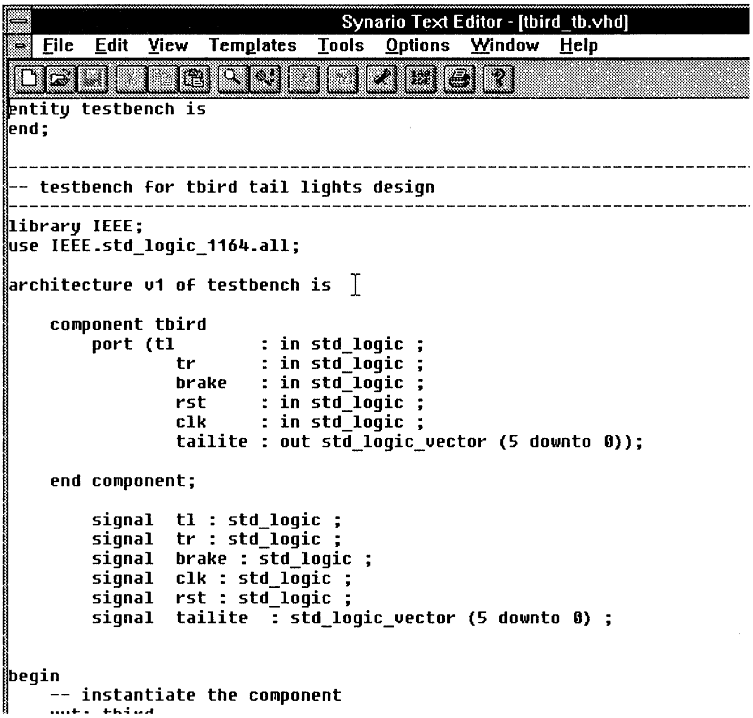
Link the Design

From the Project Navigator, select “PZ3032” in the source window and double-click on “Link Design” in the process window. The software will link the symbol and the VHDL code If the linking is successful, a green check will appear in the process window to the left of “Link Design”. A red “X” will appear if the linking fails

FUNCTIONAL SIMULATION

To verify operation of the design, a functional simulation is done. For this example, the VHDL test bench “tb_sim.vhd” is provided for the simulation. Load the test bench by selecting Source/Import from the Project Navigator window. Navigate to the “C:\designs\vhdl” directory, select the file “tb_sim.vhd” and press OK. An “Associate VHDL Test Bench” window prompting you to associate the test bench with a particular source will appear.

Select “PZ3032” and press OK. The file tb_sim.vhd should now be displayed in the source window underneath PZ3032.

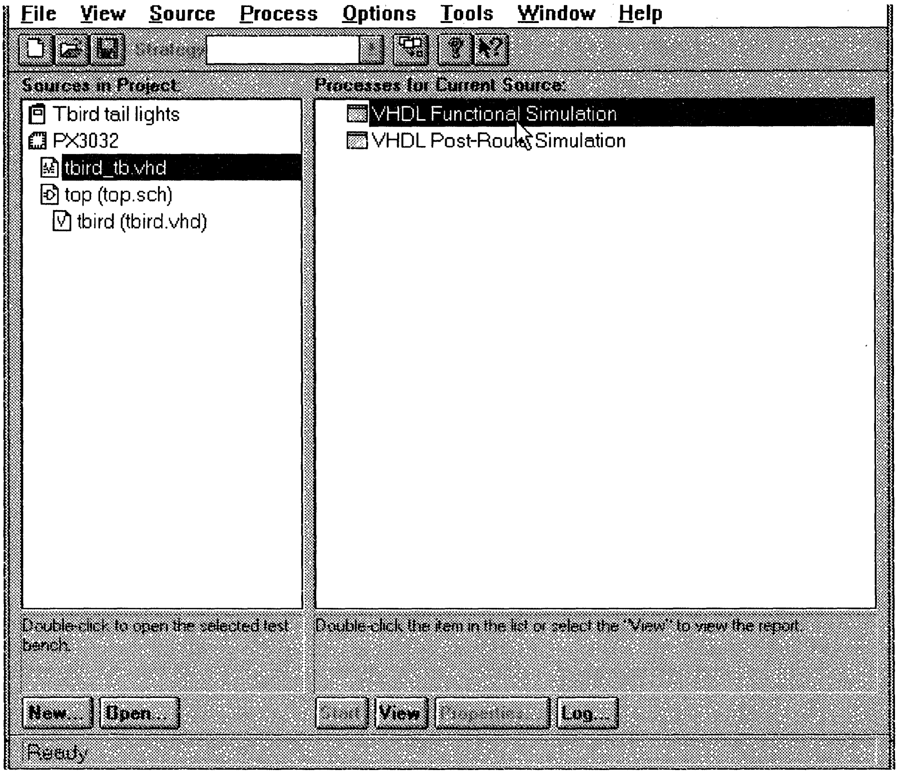


```
entity testbench is
end;

-------------------------------------------------------------------------
-- testbench for tbird tail lights design
-------------------------------------------------------------------------
library IEEE;
use IEEE.std_logic_1164.all;

architecture v1 of testbench is

    component tbird
        port (tl        : in std_logic ;
                tr      : in std_logic ;
                brake   : in std_logic ;
                rst     : in std_logic ;
                clk     : in std_logic ;
                tailite : out std_logic_vector (5 downto 0));

    end component;

        signal  tl : std_logic ;
        signal  tr : std_logic ;
        signal  brake : std_logic ;
        signal  clk : std_logic ;
        signal  rst : std_logic ;
        signal  tailite  : std_logic_vector (5 downto 0) ;


begin
    -- instantiate the component
    uut: tbird
```

Figure 11 VHDL test bench tb_sim.vhd

Highlight source “tb_sim.vhd” and verify that VHDL Functional Simulation and VHDL Post-Route Simulation appear in the process window.

Double-click “VHDL Functional Simulation”. A VHDL Simulation Model is created and Synario invokes the Model Technology simulator (Figure 12).



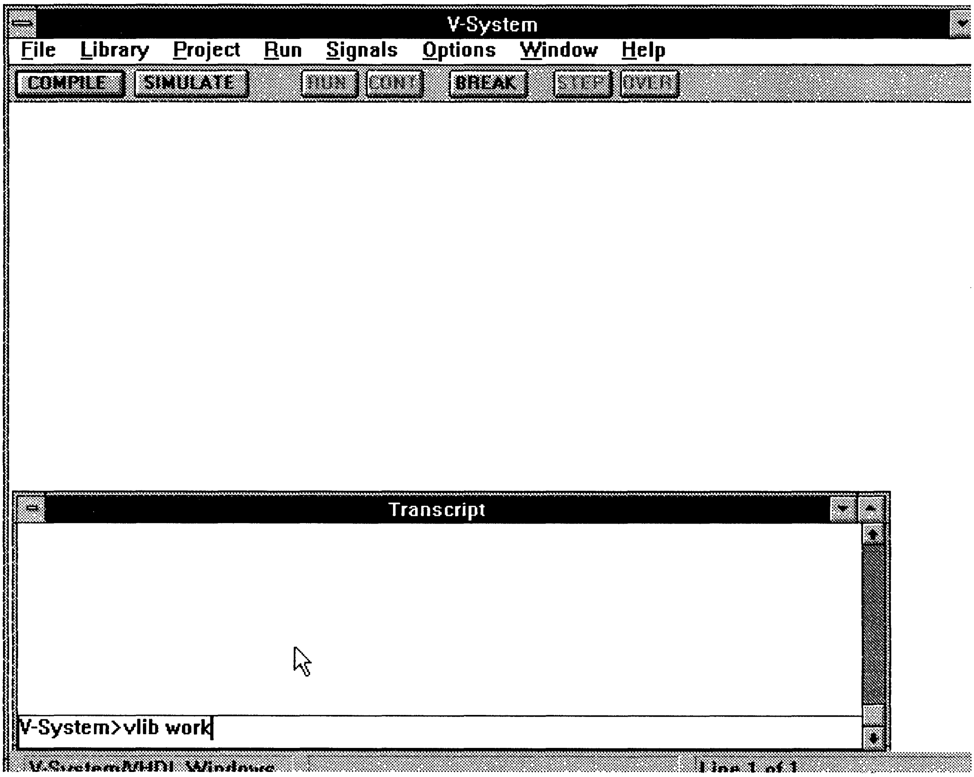
Double-click to open the selected test bench.

Double-click the item in the list or select the "View" to view the report.

Ready

Figure 11: Simulation window

Commands are entered using either the pull down menus from the menu bar, fixed menus, or by entering commands in the transcript window. Using the transcript window shown in Figure 12, change directory to the design directory by entering cd \designs\vhdl\.



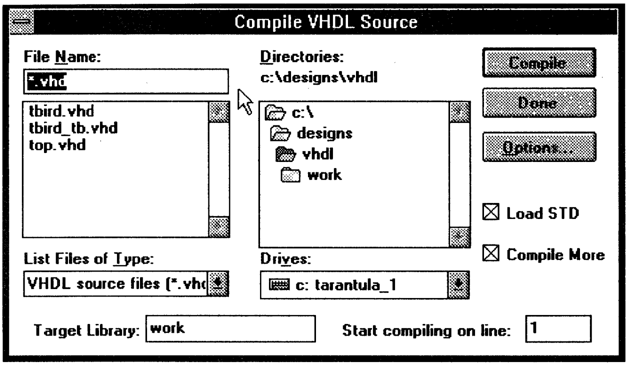
```
V-System>vlib work
```

Figure 12 Entering commands in the transcript window

Create a library by entering

vlib work

Select the Compile fixed menu and ompile tbird.vhd, top.vhd, and tb_sim.vhd design units This can be done through the transcript window or the dialog box. The dialog box is shown in Figure 13.



File Name: *.vhd

tbird.vhd
tbird_tb.vhd
top.vhd

Directories: c:\designs\vhdl

c:\
designs
vhdl
work

☒ Load STD

☒ Compile More

List Files of Type: VHDL source files (*.vh(

Drives: c: tarantula_1

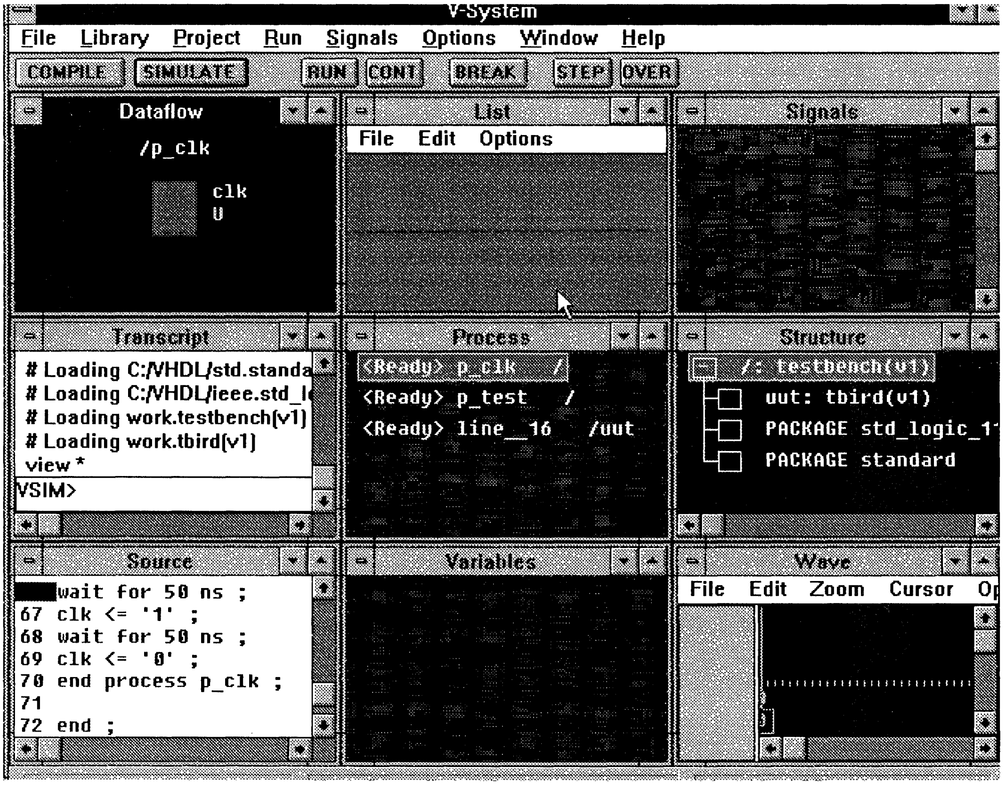
Target Library: work

Start compiling on line: 1

Figure 13 Compiling files

.

After compilation, select Simulate and simulate the design. Then run the design for 1000 ns. To display all nine Model Technology windows shown in Figure 14, select Windows- Restore All followed by Windows-Tile Vertically from the menu bar.



```
# Loading C:/VHDL/std.standa
# Loading C:/VHDL/ieee.std_l
# Loading work.testbench(v1)
# Loading work.tbird(v1)
view *
VSIM>
```

```
<Ready> p_clk      /
<Ready> p_test     /
<Ready> line__16   /uut
```

```
/: testbench(v1)
  uut: tbird(v1)
  PACKAGE std_logic_1
  PACKAGE standard
```

```
wait for 50 ns ;
clk <= '1' ;
wait for 50 ns ;
clk <= '0' ;
end process p_clk ;

end ;
```

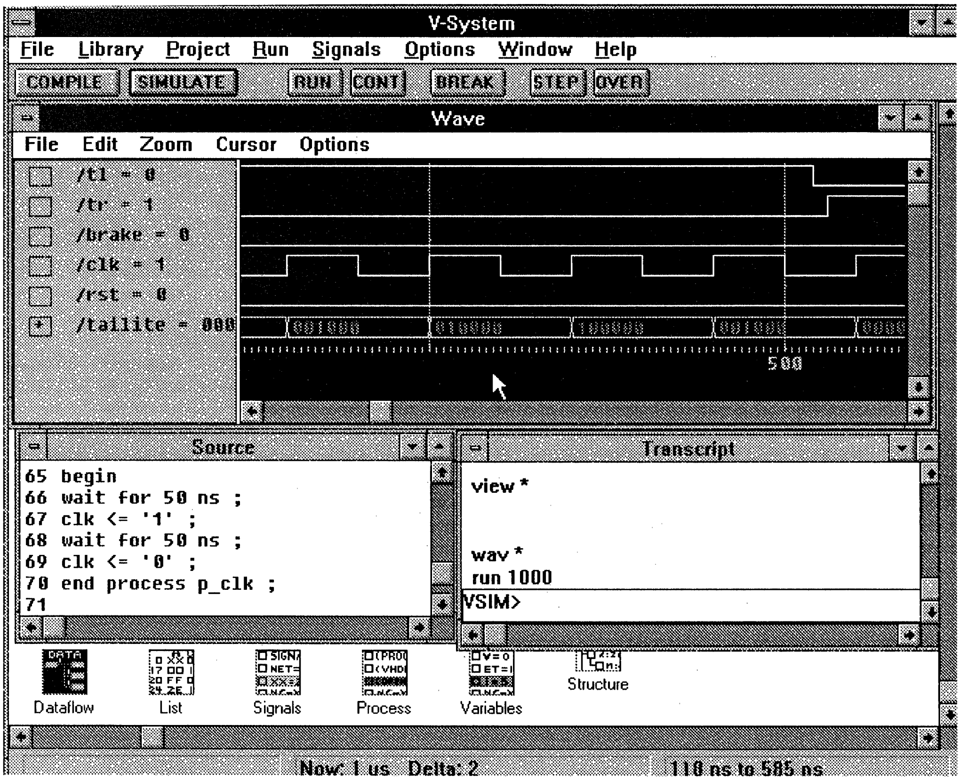
Figure 14 Simulation windows

Minimize all but the Sources, Transcript, and Wave windows. To display all of the signals in the waveform window, enter

wave *

in the transcript window.

View the display as in Figure 15, and move the cursor to various times in the waveform window and verify that the logical values change as expected when different times are viewed. Select a time when brake is high and notice that all the bulbs (outputs) are lit. Do the same for the turn left and turn right signals.


V-System
File Library Project Run Signals Options Window Help
COMPILE SIMULATE RUN CONT BREAK STEP OVER
Wave
File Edit Zoom Cursor Options
001000
010000
100000
001000
0000
500
Source
Transcript
Dataflow
List
Signals
Process
Variables
Structure

/tl = 0
/tr = 1
/brake = 0
/clk = 1
/rst = 0
/tailite = 000

```
begin
wait for 50 ns ;
clk <= '1' ;
wait for 50 ns ;
clk <= '0' ;
end process p_clk ;
```

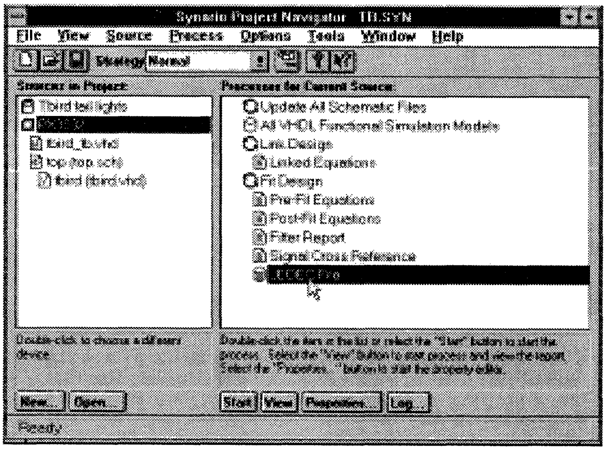
```
view *


wav *
run 1000
VSIM>
```

Now: 1 us Delta: 2   110 ns to 585 ns

Figure 15 Waveform display

COMPILING THE DESIGN AND PROGRAMMING THE PZ3032

Select the PZ3032 device in the source window and double-click on “Jedec File” in the process window (Figure 16). This causes all processes listed in the process window to be executed on the PZ3032 source. Processes that were successfully executed are indicated by a green check mark to the left of the process. Processes that fail are indicated by a red “X”.



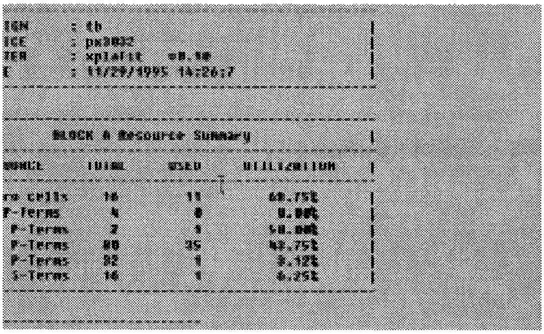
Double-click to choose a different device.

Double-click the item in the list or select the "Start" button to start the process. Select the "View" button to start process and view the report. Select the "Properties..." button to start the property editor.

Figure 16:Compilation the design

As seen in Figure 16, compilation options can be entered by selecting a source and clicking on the Properties fixed menu. Change the maximum P-terms node per node to 21.'Change Generate Clock Enable logic to false.

Three report files are generated upon compilation. The “top.fit” file contains the fitter report (figure 17) which indicates the CPLD utilization and pinout. The “top.tim” file contains timing information. The “top.jed” file can be used with a device programmer such as Data I/O's Unisite/2900/3900 or BP Microsystems to configure the CPLD.



```
IGN      : th                                    |
ICE      : px3032                                |
TER      : xplafit   v0.10                       |
E        : 11/29/1995 14:26:7                    |
--------------------------------------------------

--------------------------------------------------
     BLOCK A Resource Summary                    |
--------------------------------------------------
OURCE      TOTAL     USED     UTILIZATION        |
--------------------------------------------------
ro cells    16        11         68.75%          |
P-Terms      4         0          0.00%          |
 P-Terms     2         1         50.00%          |
 P-Terms    80        35         43.75%          |
 P-Terms    32         1          3.12%          |
 S-Terms    16         1          6.25%          |
--------------------------------------------------

---------------------------
```

Figure 17 Partial display of fitter report

XPLA Designer workstation flow for targeting Philips CPLDs

AN061

INTRODUCTION

The workstation based XPLA Designer provides a cost effective means for designing Philips CPLDs on Sparcand Hewlett-Packard workstations. XPLA Designer provides a graphical user interface (GUI) and can be run from the command line.The GUI uses the Motif window interface. This note provides a script for running XPLA Designer on workstations. It also provides a method for simulation using the Verilog simulator. XPLA Designer uses the Philips Hardware Description Language as the source description of a design. It is used to compile to a jedec file. It also generates Verilog and/or VHDL models from the jedec file. These can be used for timing simulation. This note describes a flow using the Verilog-XL simulator. Some designs have been tested using the Veriwell simulator, which for simulation of CPLDs up to approximately 128 macrocells is available at no charge. For many functions, it is compatible with Verilog-XL and VCS.

The Programmable Logic Group of Philips Semiconductor is developing a family of advanced 3-volt and 5-volt complex programmable logic devices(CPLDs). The XPLA series, designated as the PZ5000 - (5-volt) and PZ3000 (3-volt) series devices, is footprint compatible with the Altera 7000 series devices. The principle advantage of Philips CPLDs over all existing CPLDs is that they consume zero static power. The other advantages are 25% higher logic capacity and a better ability to fit logic with fixed pinouts. The first devices, the 32-macrocell PZ3032 and PZ5032, began sampling in Feb 1996. The PZ3064/PZ5064 is scheduled to sample in November and the PZ5128/PZ3128 is scheduled to sample in December. The PZ5128/PZ3128 are in-system programmable. All devices are all programmable on Data I/O and BP Microsystems programmers.

For additional information, telephone Philips Applications Support at 888-coolpld or browse http://www.coolpld.com.The following documentation is available either through the web server or calling 888 coolpld.

XPLA Designer User's Manual

Philips Hardware Description Language (PHDL) models of commonly used digital functions

PZ5000/PZ3000 Series Data Sheets

DESIGN FLOWS

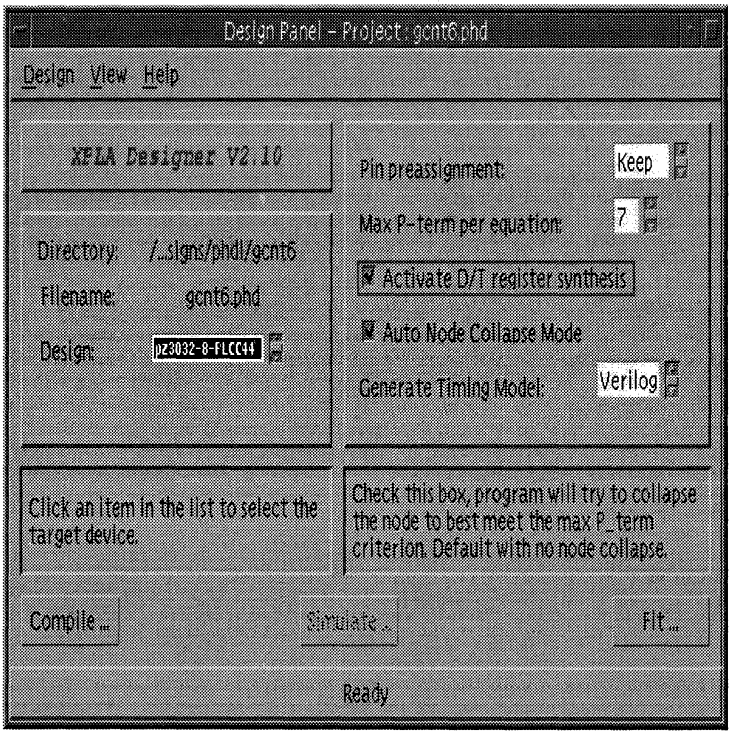
The design flow differs from that of a typical workstation flow using VHDL or Verilog. The steps are as follows:

1. Generate the description of the design using Philips Hardware Description Language (PHDL) using any text editor
2. Compile using the xsh script or GUI.
3. Write the testfixture and simulate

This is different from the standard Verilog/VHDL design methodogy of simulating and then synthesizing, but is not too different from the flow used for many PLD designs. If the testfixture is written at the design outset, it may require significant edits. By using the template in this application note, the edits can be avoided.

After generating the PHDL source, the GUI is invoked by entering

xplayer



Directory: /...signs/phdl/gcnt6

Filename: gcnt6.phd

Design: pz3032-8-PLCC44

Pin preassignment: Keep

Max P-term per equation: 7

☒ Activate D/T register synthesis

☒ Auto Node Collapse Mode

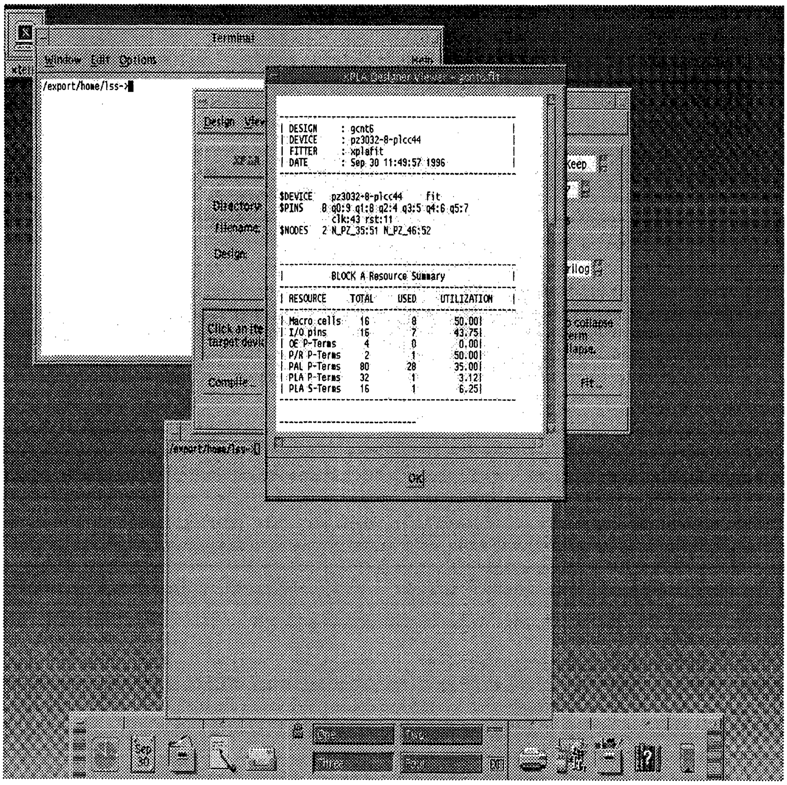
Generate Timing Model: Verilog

Click an item in the list to select the target device.

Check this box, program will try to collapse the node to best meet the max P_term criterion. Default with no node collapse.

Ready

The steps in using the GUI are straightfoware. Open a design, select the part type and compilation ottions, and compile and fit. The results can be viewed as shown below.



```
/export/home/lss->
```

```
-------------------------------------------------------
| DESIGN      : gcnt6                                  |
| DEVICE      : pz3032-8-plcc44                        |
| FITTER      : xplafit                                |
| DATE        : Sep 30 11:49:57 1996                   |
-------------------------------------------------------

$DEVICE    pz3032-8-plcc44     Fit
$PINS     8 q0:9 q1:8 q2:4 q3:5 q4:6 q5:7
            clk:43 rst:11
$NODES    2 N_PZ_35:51 N_PZ_46:52


-------------------------------------------------------
|            BLOCK A Resource Summary                  |
-------------------------------------------------------
| RESOURCE      TOTAL     USED     UTILIZATION         |
-------------------------------------------------------
| Macro cells     16        8         50.00|
| I/O pins        16        7         43.75|
| OE P-Terms       4        0          0.00|
| P/R P-Terms      2        1         50.00|
| PAL P-Terms     80       28         35.00|
| PLA P-Terms     32        1          3.12|
| PLA S-Terms     16        1          6.25|
-------------------------------------------------------


------------------------------
```

A command line script xsh is invoked using either of two methods. The compile and fit functions can be done using xplaopt and xplafit commands respectively. entering teh command with a -help argument provides usage of the command. Alternately, a script such as xsh given below can be used.

xsh $1

where $1 is the design name without the .phd extension.

This will compile to a jedec file in th $1.run1 directory. The device is specified in this script, which is located in <path>/xpla/bin. Entering xsh without any arguments will provide a usage message. The user can edit xsh to change options such as the number product terms used, the device targeted, and ensure that each run is placed in its own run directory.

The contents of xsh are editable and given below.

# This is an editable xsh script for running XPLA designer at the command line

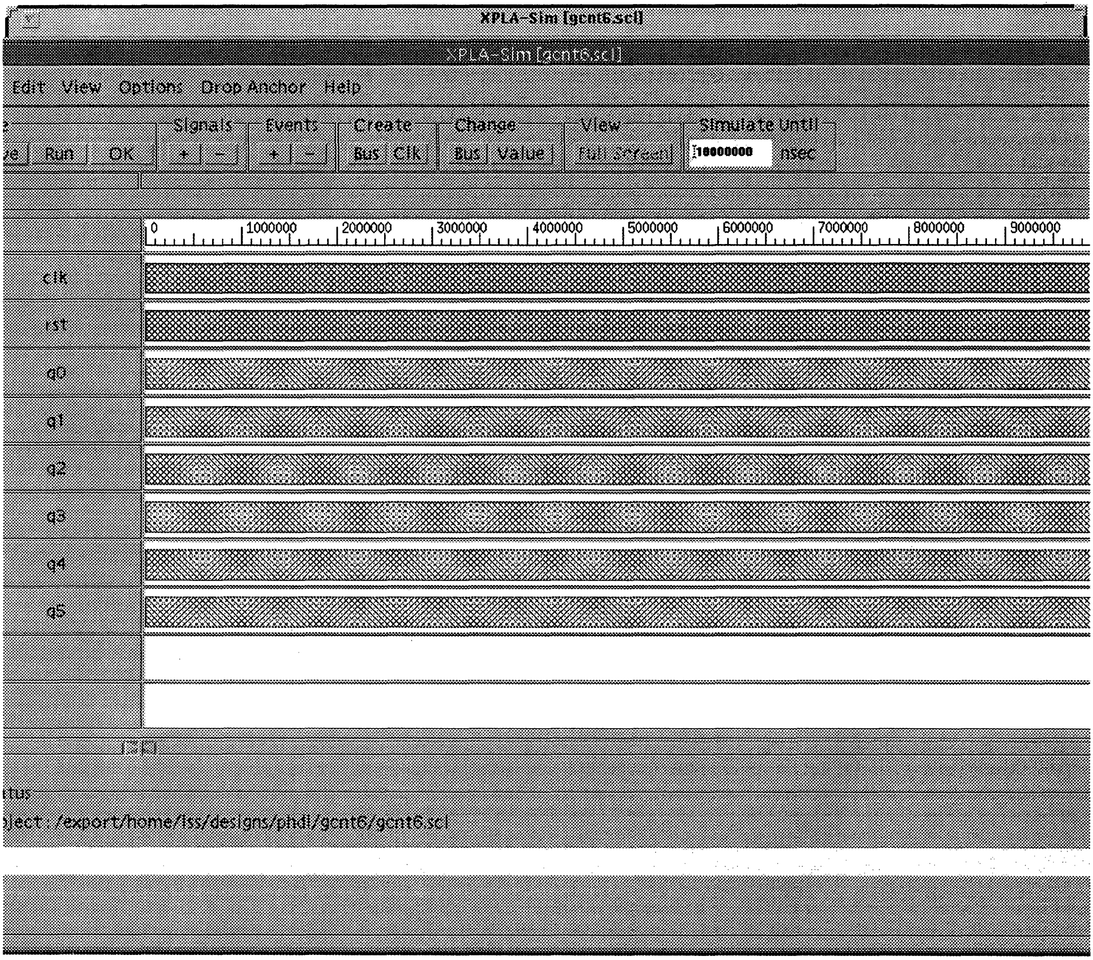
# This is an editable xsh script for running XPLA designer at the command lint

# Sept 12, 1996

```
if [ $# -eq 0 ]
then
echo "Usage - "
echo ""
echo "Compilation and fitting options are assigned in the xsh script."
echo "Valid devices are:"
echo "PS3032-8-plcc44                    PS5032-7-plcc44"
echo "PS3032-8-qfp44                     PS5032-7-qfp44"
echo "PS3032-10-plcc44                   PS5032-7-plcc44"
echo "PS3032-10-qfp44                    PS5032-7-qfp44"
echo "PS3032-7-plcc44                    PS5032-7-plcc44"
echo ""
echo "-vo - generates Verilog model from jedec file
echo "-vhdl - generatei VHDL model from jedec file
echo "-pre try - Directs fitter to attempt to keep pin assignments
echo "-pre ignore - Directs fitter to ignore pin assignments
echo "-pre keep - Directs fitter to keep pin assignments
echo ""
echo "-max # - Directs fitter to use a maximum of # PTs; # ranges from 5 to 37
fi
echo ""
echo "Philips Semiconductor"
echo "XPLA Designer - Version 1.0"
echo ""
echo "Targeting PZ5032-7-plcc44"
mkdir $1.run1
cp $1.phd $1.run1
cd $1.run1
xplaopt -I $1.phd -it phd -o $1.pla -ot tt2 -reg -co none
echo "$?"
xplafit $1.pla -dev pz5032-7-plcc44 -pre try -max 36 -vo
echo "$?"
cd ..
```

After completing design compilation, the user should analyze the <design>.fit to verify that the design fit and that the correct resources were used.

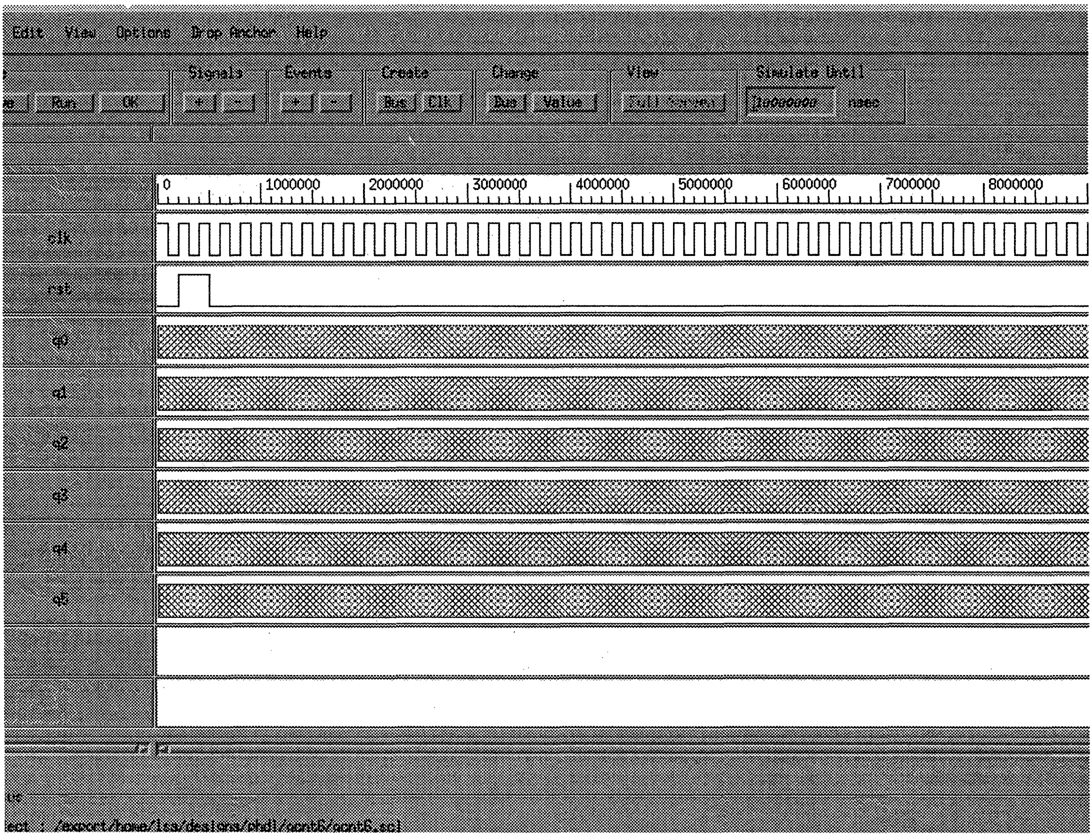
To simulate using the XPLA Designer click on the Functional Simulation radio button.



Simulate Until 10000000 nsec

oject:/export/home/lss/designs/phdl/gcnt6/gcnt6.scl

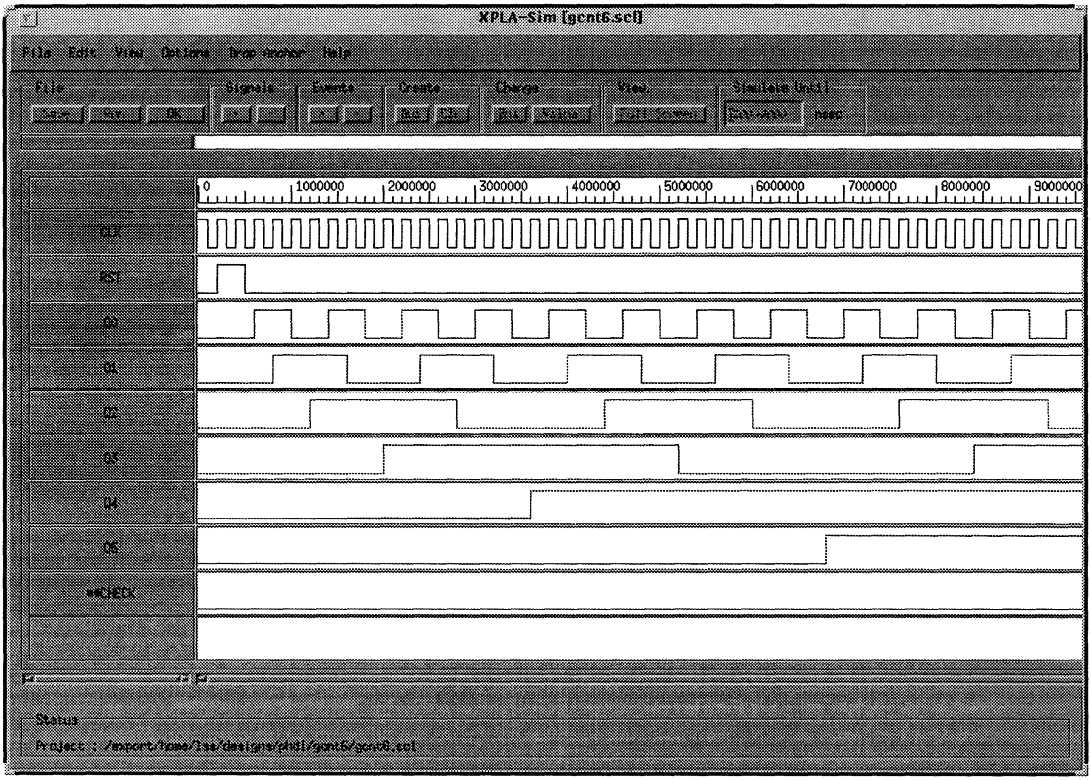
The signals sre displayed. For this gray code counter, the user needs to provide the clock and reset stimuli.



Simulate Until 10000000 nsec

ect : /export/home/lsa/designs/vhdl/gcnt6/gcnt6.scl

Use Change Values and Change Event to input reset and select Run.



Project : /export/home/lss/designs/phdl/gcnt6/gcnt6.scl

The results are displayed.

Simulation from the command line. The steps for simulationg from the command line are given below.

```
pla2net $1.net
simchk $1.scl $1.net
simnet $XPLA_PATH/bin/ $1.net # produces $1.bin and $1.log
simscl $1.bin $1.scl # procuces $1.scr
simrun $1.bin $1.scr # produces $1.res
simprt $1.res # get simprt.txt
~
```

To simulate using Veriwell or another Verilog simulator, the next step is to complete a test fixture from the template listed below.

Verilog testfixture template

```
module $_tf;
reg <input_variables> ;
wire <output_variables>
$1 u1 (<module_ports_from $1.vo>) ;
initial begin
<input_variables> = <initial_value> ;
end
integer $1_chann ;
initial begin
$1_chann= $fopen ("$1r.rpt") ;
end
initial begin
// $shm_open("$1.shm") ; Not supported by Veriwell
// $shm_probe("AS") ; Not supported by Veriwell
$fmonitor ($1_chann,"time %d <variables_to monitor>=format_specifier>", $time, <variables_to_monitor>)
#<time> <input_stimuli> = <value>;

.
$fdisplay ($1_chann,"\nSimulation of comparator is complete.") ;
$finish ;
end
endmodule
```

Design Flow example

Using an 4-bit comparator, a design flow example is as follows.

The source PHDL file is

```
MODULE comparator
TITLE '4-bit equality comparator - high level implementation'
```

```
a3..a0 pin ;
b3..b0 pin ;
equal pin ;
a = [a3..a0] ;
b = [b3..b0] ;
equations
equal = (a==b) ;
end
```

The verilog model output from XPLA Designer is

```
`timescale 1 ns / 100 ps
module comp4(a0, a1, a2, a3, b0, b1, b2, b3, eq);
input a0, a1, a2, a3, b0, b1, b2, b3;
inout eq;
wire  eq_D, eq_OE;
parameter tpd0 = 0.0;
parameter tpd1 =  6.0;
parameter tpd2 =  8.5;
parameter tclk =  0.5;
// Equations:

//----( eq )----------------------------------------------------------------
assign #tpd2 eq_D = !(a1 && !b1
      || !a2 && b2
      || a2 && !b2
      || !a3 && b3
      || a3 && !b3
      || !a0 && b0
      || a0 && !b0
      || !a1 && b1);
assign #tpd0 eq_OE = (1);
pxa_bufif1 eq_buf(eq, eq_D, eq_OE);
endmodule
module pxa_bufif1(out, in, oe);
```

```
  output out;
  input  in, oe;
  bufif1(out, in, oe);
/*  specify
    specparam tbuf =  1.5;
    specparam toe = 12.5;
    (in => out) = tbuf;
    (oe => out) = toe;
  endspecify
*/
endmodule // pxa_bufif1
```

To use the $1.vo with the Veriwell simulator, comment the specify block. This isn't necessary with Verilog-XL.

The test fixture is given below. To simulate with the verilog model generated from XPLA Designer, the bus must be broken into discrete signals. Also, the module port order in the testfixture must match that or comp4.vo, not comp4.v.

```
`timescale 1 ns / 100 ps
module comp4_tf ;
reg  a0 ;
reg  a1 ;
reg  a2 ;
reg  a3 ;
reg  b0 ;
reg  b1 ;
reg  b2 ;
reg  b3 ;
wire eq ;
comp4 u1 (a0,a1,a2,a3,b0,b1,b2,b3,eq) ;
initial begin
a0=1'b0 ; a1=1'b0 ; a2=1'b0 ; a3=1'b0 ;
b0=1'b0 ; b1=1'b0 ; b2=1'b0 ; b3=1'b0 ;
end
integer comp4_chann ;
initial begin
comp4_chann=$fopen("comp4.rpt") ;
end
```

```
initial begin
$fmonitor(comp4_chann,"time=%t,a0=%b,a1=%b,a2=%b,a3=%b,b0=%b,b1=%b,b2=%b,b3=%b,eq=%b",$time,a0,a1,
a2,a3,b0,b1,b2,b3,eq) ;
#10 a0=1'b0; a1=1'b0 ; a2=1'b1 ; a3=1'b1 ;
#10 b0=1'b0; b1=1'b0 ; b2=1'b1 ; b3=1'b1 ;
#10 a0=1'b0; a1=1'b1 ; a2=1'b1 ; a3=1'b1 ;
#10 b0=1'b0; b1=1'b1 ; b2=1'b1 ; b3=1'b1 ;
#10 a0=1'b0; a1=1'b0 ; a2=1'b0 ; a3=1'b1 ;
#10 b0=1'b0; b1=1'b0 ; b2=1'b0 ; b3=1'b1 ;
#10 a0=1'b0; a1=1'b0 ; a2=1'b1 ; a3=1'b1 ;
#10 b0=1'b0; b1=1'b0 ; b2=1'b1 ; b3=1'b1 ;
$fdisplay (comp4_chann,"\nSimulation of comp4 is complete.");
$finish ;
end
endmodule
```

Understanding the hidden costs of using high–power CPLDs

AN062

Author: B. Wade Baker, Senior CPLD Specialist, Philips Semiconductors

Old guys like me can remember the days when, as designers, we didn't have to worry a bit about power consumption. Everything consumed enormous amounts of energy and 'low power' meant something around 100 watts. As we all know, those days are gone! Even if you are building systems that run from the wall socket, you still have to consider power as a part of your design check-off.

CPLDs are enjoying ever-increasing use as the requirement for smaller and smaller packaging rises. It makes sense to use CPLDs in your design because you can combine many of the functions that were once discrete into one package. CPLDs are also the only parts that make sense for things like state machines, address decoders, high speed data path controllers, etc. As packaging size decreases, energy density increases. Thermal issues are now a major concern that every designer must heed.

You would think that the CPLD you chose to fit into your small form factor design will consume less power than the total of the discrete parts it replaced, but this is not always the case. Our competitor's CPLDs consume power in the miliwatts while sitting idly by, doing nothing. This is their 'low power' mode. When you actually start clocking the part, watch out, you just might start consuming watts of power! The obvious thing to do with a part that consumes so much power is attach a heat sink. If you are already experiencing board space budget problems, this will not work. Heat sinks require some movement of air to work properly, and if your design is tight and fully enclosed you will have problems. You could always add a fan but... now you have EMI problems, air flow calculation issues, and a large form factor to deal with. You could get exotic and use heat pipes or a peltier effect heat pump, but you would really be disguising the same problems as mentioned above in 'new' clothes.

Let's say you don't have board space problems, your design is in a big rack. You run from wall power and there is plenty of it. There is a big honking power supply with a fan at the bottom of your rack and it will be there regardless of what you do. So you're thinking, "Why should I go to the trouble of switching from my old reliable CPLD manufacturer to the Philips Semiconductors' CoolRunner™ CPLD? I already know how to use my proprietary tool set to design my parts and everything is great." Just for fun, let's say you have 10 to 20 CPLDs in your system. This is not an uncommon amount. Let's say the mix is 32, 64 and 128 macrocell parts. If the average quiescent current consumption for each CPLD is 150 mA, (a conservative number if you use our competitors' parts), then you are burning 3 Amperes for nothing in return. At 5 volts that comes to 15 watts down the drain, so to speak. Of course, it only gets worse when the system is actually clocked. Fifteen watts of heat load is significant. You will probably need to have a fan at the top of your rack to handle waste heat removal. What about the power supply? You may be able to go with a less robust one if you remove 15 watts from your power requirements. If you can save $2.00 per system in power supply costs and you sell 10,000 systems a year—won't your boss love you for that! You might even get a medal for thinking outside the box! What if you want to sell in the European market? Cooling fans are not a great thing to have when dealing with the very strict European emission standards. Lose the cooling fans and you're talking about approximately $20.00 per system in cost reduction! What about a battery powered portable unit? You cannot even think about it using our competitors' CPLDs.

Maybe your competitor is thinking about it and has decided to go with a truly low power solution... If, in the above example, you had used Philips Semiconductors' CoolRunner™ CPLDs your worst case quiescent current drain would have been 1 mA—that's right 1 mA! Our worst case quiescent current requirement for the entire family is 0.00005 Amperes per device! At idle your CPLDs would be consuming 5 miliwatts! That is 3000 times less current at idle! It boggles the mind! At f_{MAX} we will be, at the very least, 1/3rd of our competitors' current requirements.

In order to stay ahead of the competition, you must manufacturer products of superior quality at a lower cost. Philips Semiconductors' CoolRunner™ CPLDs can help you accomplish your goal. Reducing system current requirements allows you to increase reliability, reduce component count, increase battery life, and shrink your packaging size.

Probing internal nodes using XPLA Designer's graphic simulator

AN063

Author: B. Wade Baker, Philips Semiconductors

Many times, when using a simulator to explore the dynamics of your latest CPLD design, you would like to be able to see what is going on between the inputs and outputs. The Philips XPLA Designer Simulator allows you to do just that by providing a method of displaying internal nodes directly as part of the waveform viewer. This application note describes how to accomplish this task.

XPLA Designer allows you to do both functional and timing simulations. The functional simulation does not include part specific information since it is a generic check of your work and could be applied to any part in the Philips Coolrunner family. The timing simulation contains part specific information based upon the particular device you selected before your design was fitted.

The XPLA Simulator constructs a simulation of your design's logic based upon a binary netlist (.bin) file created by the simulator and an input stimulus (.scl) file created when you manipulate the voltage levels for the input signals displayed on the waveform viewer window. When the simulator is commanded to run in the functional mode it combines the .bin file with the .scl file and creates a net (.net) file. The net file generates the voltage levels of the outputs based upon your logic's design parameters. Figure 1 displays the .net file for the 3 bit counter demo design that comes with XPLA Designer.

As you can see from Figure 1, there are not any internal nodes to check because this is the functional simulation file. Any signal you picked to display in the waveform viewer would already be present, therefore the idea of probing internal nodes does not apply to functional simulations.

For timing simulations the simulator combines the .scl and .bin files with a .mod file. The .mod file combines part specific information such as; package type, voltage level, speed, and density with design specific information such as how many levels of logic, PAL or PAL/PLA delay, and anything else that is applicable in accurately simulating the design. Figure 2 is the .mod file for the Demo design using a PZ3032–8 PLCC44. In Figure 3 we have a schematic representation of the internal workings of the Coolrunner family of parts. This schematic can be used to interpret the .mod file contents.

```
* pla2net.exe Created on:Mon Mar 03 12:54:10 1997
* Input File  : d:\xpla\example\demo\demo.pla
* Output File : d:\xpla\example\demo\demo.net
*
 NETSTART
*
CLOCK_P AND I(CLOCK) O(CLOCK_P)
RESET_P AND I(RESET) O(RESET_P)
bit0_N AND I(bit0_Q) O(bit0_N)
bit1_N AND I(bit1_Q) O(bit1_N)
PT0 AND I(CLOCK_P) O(PT0)
PT1 AND I(RESET_P) O(PT1)
PT2 AND I(bit0_N) O(PT2)
PT3 AND I(bit0_N, bit1_N) O(PT3)
bit0_AR  OR I(PT1) O(bit0_AR)
bit1_AR  OR I(PT1) O(bit1_AR)
bit2_AR  OR I(PT1) O(bit2_AR)
bit0_C  OR I(PT0) O(bit0_C)
bit1_C  OR I(PT0) O(bit1_C)
bit2_C  OR I(PT0) O(bit2_C)
bit0_T NOR I(GND) O(bit0_T)
bit1_T  OR I(PT2) O(bit1_T)
bit2_T  OR I(PT3) O(bit2_T)
bit0_Q TFFSR I(bit0_T, bit0_C, GND, bit0_AR) O(bit0_Q)
bit1_Q TFFSR I(bit1_T, bit1_C, GND, bit1_AR) O(bit1_Q)
bit2_Q TFFSR I(bit2_T, bit2_C, GND, bit2_AR) O(bit2_Q)
bit0 TRIBUF I(VCC, bit0_Q) O(bit0)
bit1 TRIBUF I(VCC, bit1_Q) O(bit1)
bit2 TRIBUF I(VCC, bit2_Q) O(bit2)
*
 NETEND
*
 NETIN VCC, GND, RESET, CLOCK
 NETOUT bit0, bit1, bit2
```

Figure 1. Demo Design Net File

```
 *
 * Created on:Wed Mar 05 10:39:56 1997
 * Input File  : demo.jed
 * Output File : demo.mod
 *
  NETSTART
 *
 XCTA__4       AND          I(XPIN1__B)                            O(XCTA__4)
 bit2          XOUTBUF15    I(VCC, XFF__A0)                        O(bit2)
 XFF__A0       XDFF3032_8   I(XSUM__A0, VCC, XCLK__0 , GND, XCTA__4)  O(XFF__A0)
 XPALA0__0     AND          I(XNODEA2__B, XNODEA1__B)              O(XPALA0__0)
 XSUM__A0      OR           I(XPALA0__0)                           O(XSUM__A0)
 bit1          XOUTBUF15    I(VCC, XFF__A1)                        O(bit1)
 XFF__A1       XDFF3032_8   I(XSUM__A1, VCC, XCLK__0 , GND, XCTA__4)  O(XFF__A1)
 XPALA1__0     AND          I(XNODEA2__B)                          O(XPALA1__0)
 XSUM__A1      OR           I(XPALA1__0)                           O(XSUM__A1)
 bit0          XOUTBUF15    I(VCC, XFF__A2)                        O(bit0)
 XFF__A2       XDFF3032_8   I(XSUM__A2, VCC, XCLK__0 , GND, XCTA__4)  O(XFF__A2)
 XSUM__A2      NOR          I(GND)                                 O(XSUM__A2)
 XPIN1__B      XINBUF45     I(RESET)                               O(XPIN1__B)
 XNODEA1__B    XINBUF45     I(XFF__A1)                             O(XNODEA1__B)
 XNODEA2__B    XINBUF45     I(XFF__A2)                             O(XNODEA2__B)
 XCLK__0       XCKBUF       I(CLOCK)                               O(XCLK__0 )
 *
  NETEND
 *
  NETIN RESET, CLOCK, VCC, GND
  NETOUT       bit2, bit1, bit0
 *
```

Figure 2. Demo Design Mod File

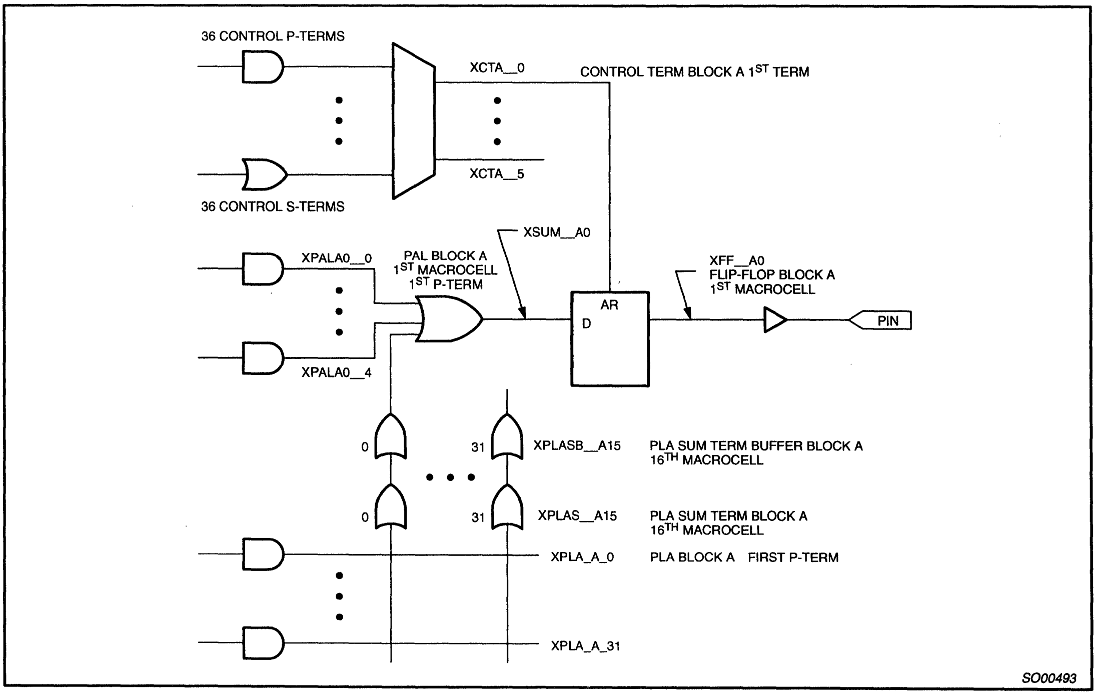
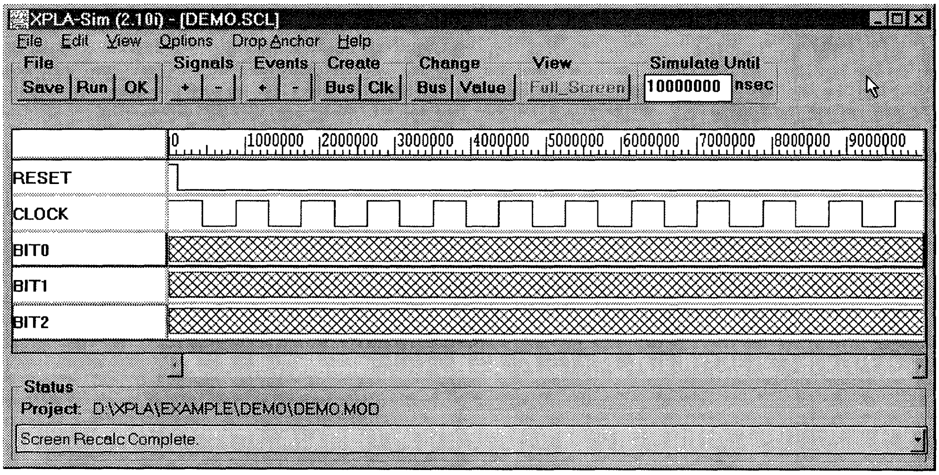


Figure 3. Internal Node Schematic

There are 36 control product terms and 36 control sum terms which feed the 6 Control terms per logic block. It is important to note that control terms can be **ONLY** sum terms **OR** product terms, **NOT BOTH**. For all of the names of internal nodes listed in Figure 3, the X at the beginning of the name represents XPLA. XCTA__0 through XCTA__5 , located at the top right in Figure 3, are those control terms. The CT is for Control Term and A__0 indicates logic block A, first control term. The middle left section of Figure 3 illustrates the 5 dedicated PAL terms available to each macrocell. They are labeled XPALA0__0 through XPALA0__4. The PAL in XPALA0__0 stands for Programmable Array Logic. The A0__0 means logic block A, first macrocell, first product term. XSUM__A0, located in the middle of the page, is the sum of the PAL structures with whatever PLA terms that may have been used by this particular macrocell. XFF__A0, at the middle right of the page, describes the output of the first flip-flop in logic block A. The PLA sum and sum term buffers are the next items down in the internal node schematic. PLA stands for Programmable Logic Array. There are 32 product terms and 32 sum terms available to each logic block. XPLASB__A15 means PLA Sum term Buffer, Block A, connecting to the 16th macrocell in the logic block. These buffers represent nodes that may be probed even though they do not actually exist in hardware. They were added in order to facilitate proper simulator function. XPLAS__A15 is PLA Sum term, Block A, 16th macrocell connection. The last part of the internal node schematic describes the PLA product term structures. XPLA_A_0 stands for PLA, Block A, 1st pterm.

Now that we have developed an understanding of the nomenclature and structure of the XPLA Designer timing simulator, let's run the simulator and add an internal node. First, start XPLA Designer and open the design **Demo.phd**. Choose the **PZ3032–8PLCC** device, set the *pin assignment* to **keep**, and *max P–term per equation* to **16**. Now fit the design. XPLA Designer will automatically compile the design and then proceed to fit it into the selected device. Now invoke the timing simulator by pressing the **TimSim□** radio button. Once the simulator window opens, move the cursor to the word *file* located in the top left corner of the window as shown in Figure 4. Click on the word *file* and then on the word *open.* A dialog box very much like Figure 5 will open.

Toward the middle of Figure 5, near the bottom, you will see the word *Edit.* Click in the box located just to the left of the word. A check mark will appear in the box. Now move the cursor to the words *demo.mod* located near the top left portion of Figure 5 and click on them. *Demo.mod* will appear in the dialog window located just above the files box and the words *demo.mod* located in the files box will become highlighted. Now click the radio button labeled **Open** located near the bottom left of the dialog box. The contents of Figure 2 will appear on your computer screen. Select the node name **XPALA0__0,** located on the 12th line down from the top, by placing the cursor at the beginning of the word, holding the left mouse button down, and dragging the mouse to the right until the entire word is highlighted. Now, on the computer keyboard, press the **Control** key and the **C** key at the same time. This will copy the highlighted word into the Windows Clipboard. Now go back to the simulator window. Located near the top left portion of the window is the section called **Signals**. Under the word are two radio buttons labeled + and –. Figure 6 depicts the proper cursor location.



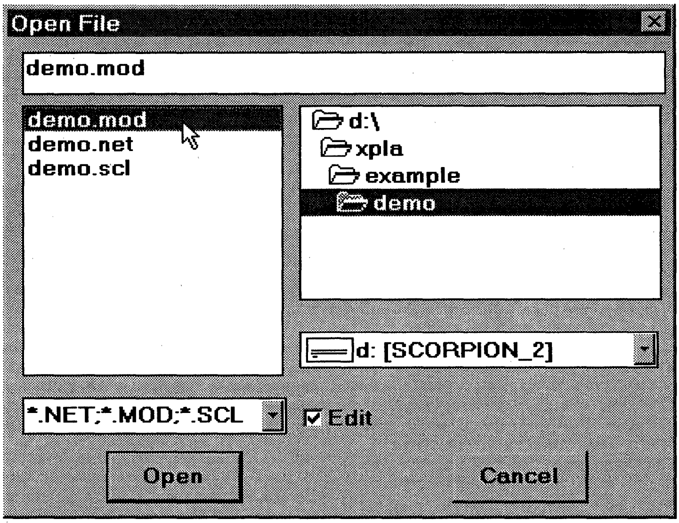
Simulate Until: 10000000 nsec

Project: D:\XPLA\EXAMPLE\DEMO\DEMO.MOD

Screen Recalc Complete.

SP00494

Figure 4. About to Click on File



Open File

demo.mod

demo.mod
demo.net
demo.scl

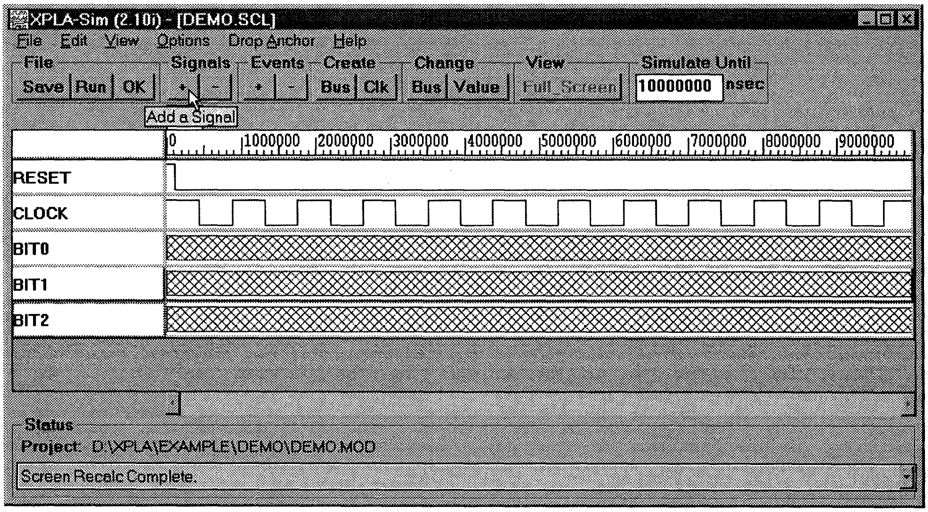
d:\
xpla
example
demo

d: [SCORPION_2]

.NET;.MOD;*.SCL

☑ Edit

Figure 5. About to Edit .MOD File



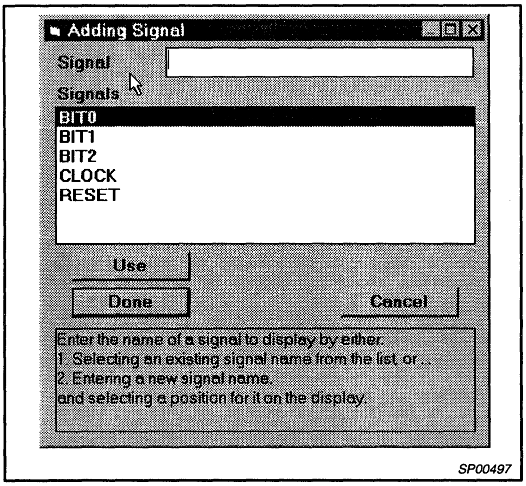
XPLA-Sim (2.10i) - [DEMO.SCL]

Simulate Until: 10000000 nsec

Status
Project: D:\XPLA\EXAMPLE\DEMO\DEMO.MOD
Screen Recalc Complete.

Figure 6. Adding Signals

Click the **Signals** + button. A dialog box will appear that looks like Figure 7.

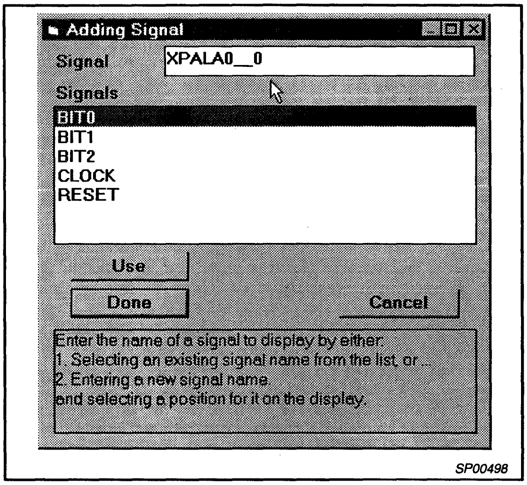


Enter the name of a signal to display by either:
1. Selecting an existing signal name from the list, or...
2. Entering a new signal name.
and selecting a position for it on the display.

SP00497

Figure 7. Adding Signal Dialog Box

Depress the **v** key while depressing and holding the **control** key. The signal **XPALA0__0** will appear in the dialog box as shown in Figure 8.



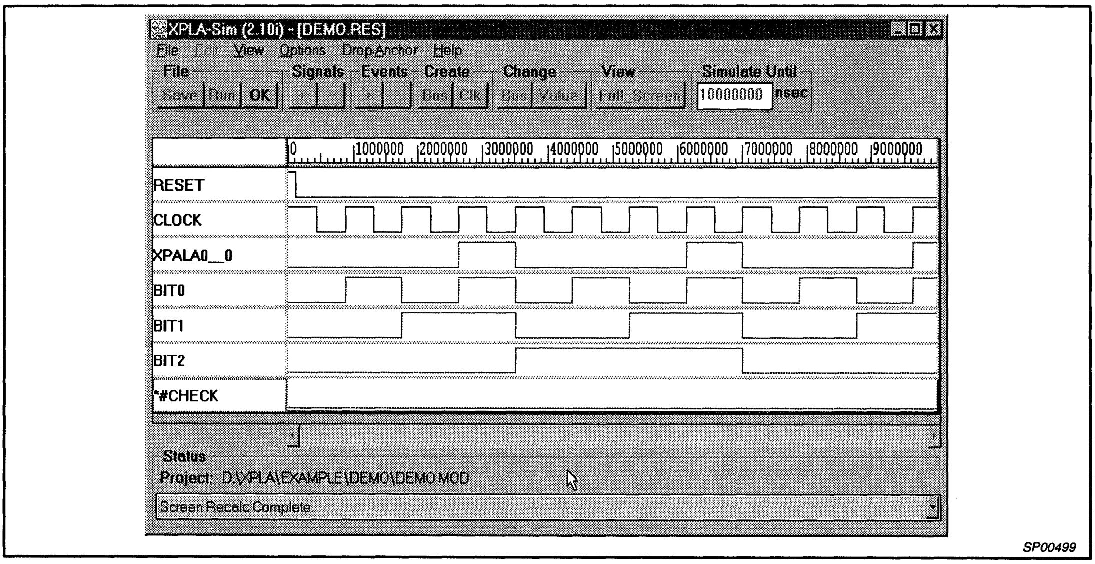
Signal: XPALA0__0

Enter the name of a signal to display by either:
1. Selecting an existing signal name from the list, or...
2. Entering a new signal name.
and selecting a position for it on the display.

SP00498

Figure 8. Adding A Signal

Click the **Done** radio button located near the bottom left of the dialog box. The signal **XPALA0__0** will now appear in the simulator waveform viewer window. You can position this signal anywhere within the simulator window by moving the cursor over the signal name and clicking and holding the left mouse button while you drag the signal to a different location. Click the **OK** radio button and then the **Run** button. You should now see a simulation very similar to the one in Figure 9.



Simulate Until: 10000000 nsec

Project: D:\XPLA\EXAMPLE\DEMO\DEMO.MOD

Screen Recalc Complete.

SP00499

Figure 9. Simulation with Internal Node Signal Added

The internal signal levels over time for **XPALA0__0** are now displayed. In this manner you can add as many internal node signals as you require to fully understand the operation of your design.

Using sum of products control terms in Philips CoolRunner™ CPLDs

AN064

DOCUMENT SCOPE

This document describes how to use sum of products equations to drive control terms in Philips CoolRunner™ CPLDs. The procedure for creating the control term is illustrated, and design considerations such as timing and register use are discussed. The concepts presented here apply to all Philips CoolRunner™ CPLDs.

INTRODUCTION

Under normal use, Philips CoolRunner™ CPLDs provide either direct product term or sum term support for asynchronous resets, asynchronous presets, and output enables. Sum of products control terms are not directly supported. Figure 1 shows a block diagram of the XPLA logic block used in Philips CPLDs.

Note the control terms located above the PAL array. The block diagram shows there are six control terms available to each logic block, and each one can be either a product term (all ANDs) or a sum term (all ORs) comprised of the 36 inputs into the Zero–Power Interconnect Array (ZIA). Unlike the logic available for each macrocell output, there is not an additional OR gate directly in front of the macrocell that allows control terms to support sums of products. For example, control terms like

OUT_EN = A & B & C & D

or

RESET = A # B # C # D

are directly supported in Philips CPLDs, but control terms like

PRESET = (A & B) # (C & D)

are not. The & and # symbols represent logical AND and OR, respectively.

In reality, there is a single product term available for each control term, and OR functions are Demorganized to change them into AND functions. Each control term is actually implemented as a product term that uses either a buffered or inverted path into the macrocell (see Figure 2).

Consider the RESET control term equation. Using Demorgan's Theorem, this will be altered during logic synthesis to

RESET = ! (!A & !B & !C & !D)

which can then be implemented using the product term and the inverted path into the macrocell.

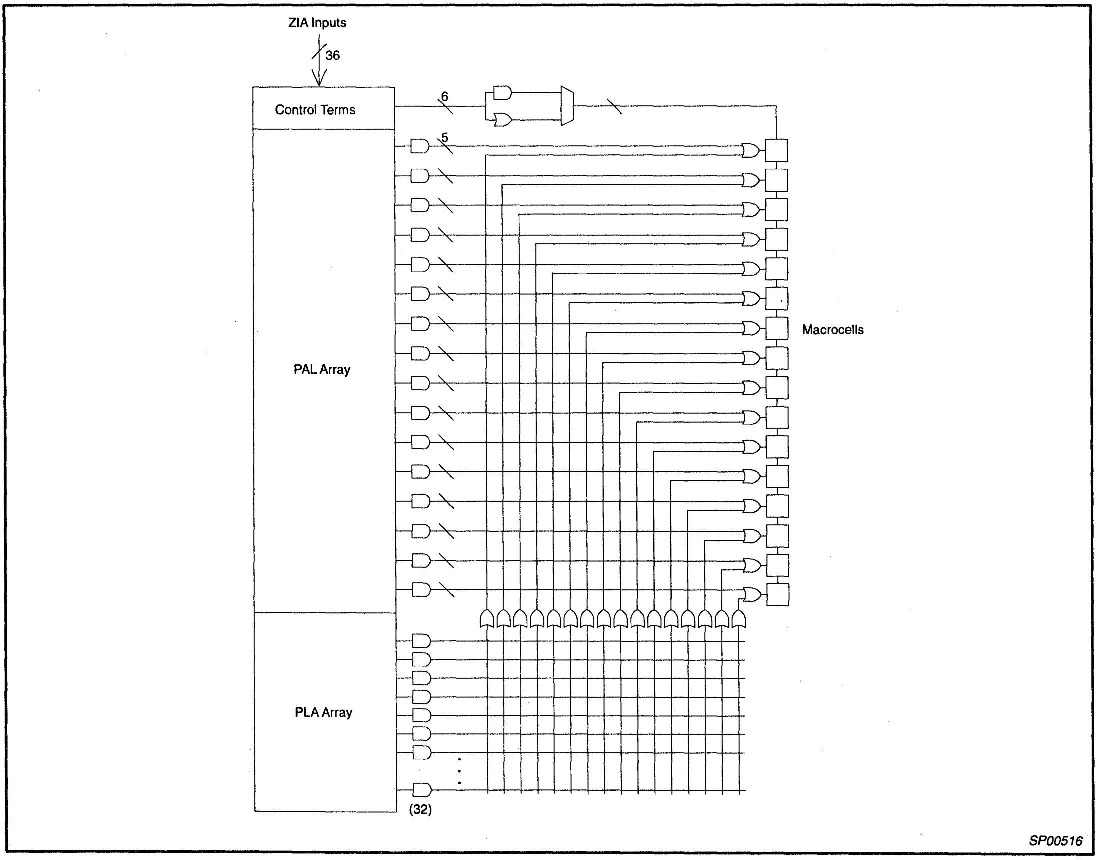


Figure 1 Philips XPLA Logic Block

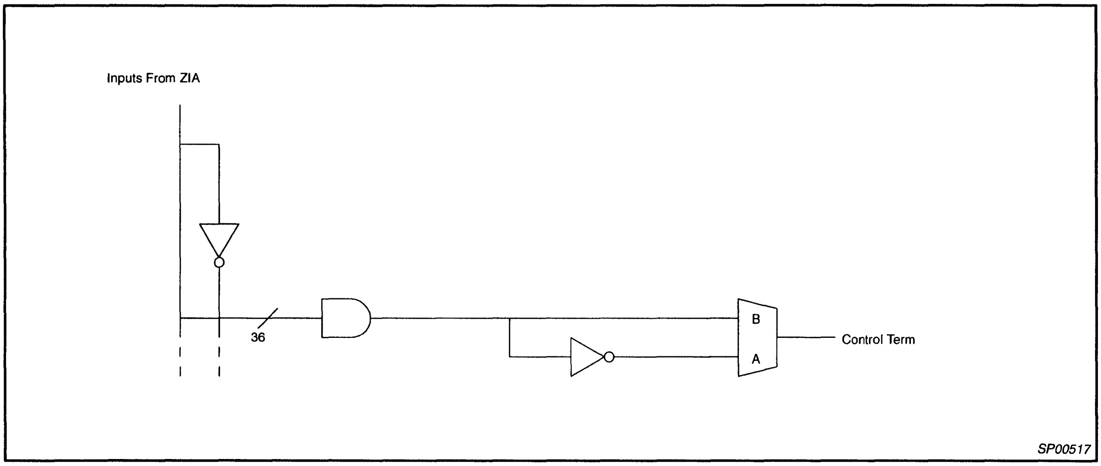


Figure 2. Product Term Implementation for Control Terms in Philips CPLDs.

IMPLEMENTING SUM OF PRODUCT CONTROL TERMS

To use sum of product control terms in Philips CoolRunner™ CPLDs, you must use an intermediate node to generate the sum of products, and then assign this intermediate node to the control term. In this way, the control term appears to the logic compiler and optimizer to be a single input (the intermediate node) to a product term. For example, consider the PRESET control term shown below:

PRESET = (A & B) # (C & D).

This cannot be directly implemented, but if the control term is assigned to an intermediate node like

PRE_NODE = (A & B) # (C & D)

and then assigned to preset like

PRESET = PRE_NODE,

then the sum of products control term could be implemented.

In order to make this work, the intermediate node cannot be collapsed during optimization of the logic. It must be preserved so that the sum of products is resolved at the node and the output of the node is used as the single input to the control term. Otherwise, the intermediate node will be removed from the design and the fitter will attempt to apply a sum of products directly to a control term. This condition will result in a fatal error from the design fitter.

Preventing a node from being collapsed during logic optimization is usually done by attaching an attribute to the node that tells the logic optimizer to preserve this particular node. The procedure for preserving nodes varies depending on the design system you are using. For example, if you are using Philips XPLA Designer and the Philips Hardware Description Language (PHDL), you would assign a 'keep' attribute while declaring the intermediate node. The declaration of the intermediate node and the assignment of the node would look something like

```
module EXAMPLE

declarations

A, B, C, D      pin;

PRE_NODE        node istype 'com, keep';

OUTPUT          pin istype 'reg';

equations

PRE_NODE = (A & B) # (C & D);

OUTPUT.AP = PRE_NODE;
```

These statements assign the sum of products control term equation to the intermediate node, PRE_NODE, and then assign PRE_NODE to the asynchronous preset of the signal OUTPUT. Figure 3 shows schematically how the above logic assignments are implemented inside the device.

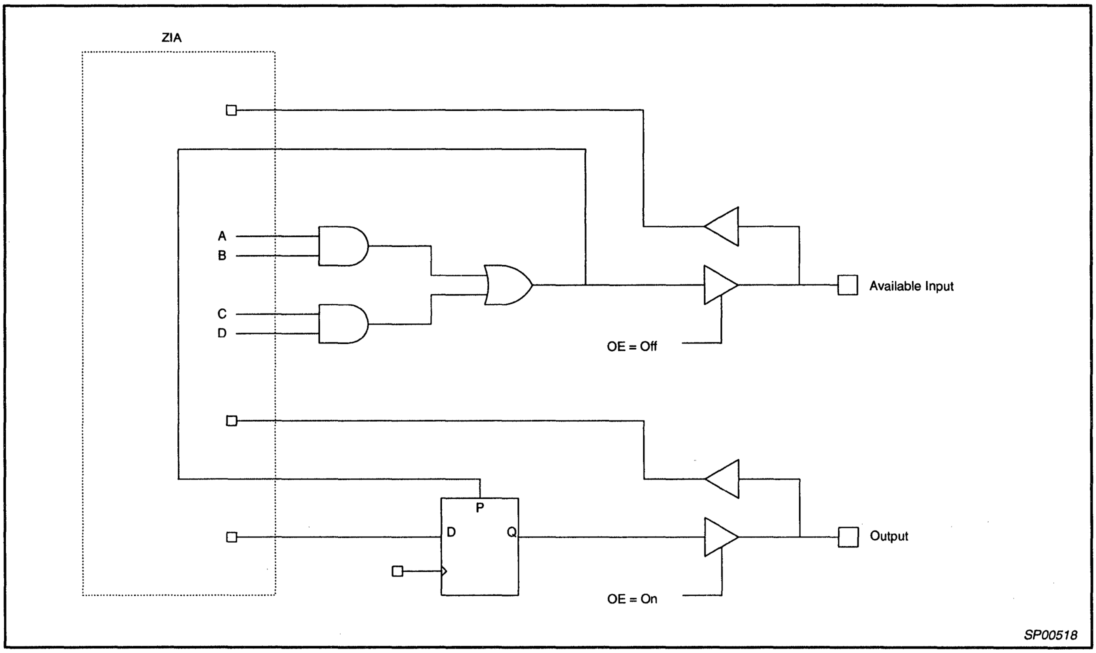


Figure 3 Schematic Representation of Sum of Products Control Term Implemented in Philips CPLD.

As you can see, PRE_NODE is preserved in the design and the sum of products is resolved there. The output of the node then feeds back to the ZIA and is used as a single input to a control term.

DESIGN CONSIDERATIONS

There are design trade–offs that must be considered when implementing sum of product control terms. First, as shown in figure 3, the creation of the internal node uses one of the available macrocells for resolution of the sum of products. This node use results from specifying that the node must not be collapsed, and is necessary to prevent the compiler and fitter from applying the sum of products directly to the control term. Note, however, that the node uses the internal feedback path (before the output buffer) for sending the sum of products result back to the ZIA. Whenever buried nodes are created in this manner, the output buffer is disabled and the node signal is not propagated to the output pin. This allows the pin associated with the macrocell where the intermediate node resides to be used as a dedicated input.

The other consideration is the fact that the time it takes for a sum of products control term to have an affect is longer than the time it takes for a product term or sum term control term to have an affect. Figure three shows that in addition to propagating through the ZIA and the logic array like ordinary control terms, sum of product control terms must first pass from the input to the internal feedback node, and then proceed through the same path as normal control terms. Thus, whenever sum of product control terms are used, the time specified in the data sheet for the asynchronous reset, asynchronous preset, or output enable to take affect should be lengthened by tPDF, the amount of time it takes for a signal to propagate from the input to the internal feedback.

TECHNICAL SUPPORT

This document was authored by Chris Schell, CPLD Applications Engineer. For more information, contact me at 505–858–2813 or call the Philips CPLD Technical Support Line at 1–888–COOLPLD (1–888–2665753) or 505–858–2996; or send email to coolpld@scs.philips.com.

Understanding CoolRunner™ clocking options AN065

The CoolRunner™ family of CPLDs includes versatile clocking options that include both synchronous (external) and asynchronous (internal, equation-based) clocking and selectable clock polarity at every macrocell. This application brief describes in detail these clocking options, and shows how to access these features using Philips XPLA Designer. We also detail how to synthesize 'soft' flip-flops and latches for those instances where these devices can be useful.

XPLA Clocking Architecture

The XPLA Devices have global, low skew clocking networks that are routed to all macrocell clock inputs. In the CoolRunner™32, there are two of these clock networks. In the CoolRunner™64 and CoolRunner™128 there are four clock networks. In all CoolRunner™ devices, at every Macrocell, there are polarity selection muxes that allow the user to generate either rising or falling edge clocking in that macrocell.

The inputs to these clock networks are one of two types. The first type of input to the clocking network are clock pins that are associated with dedicated input pins (Figure 1). These inputs can only generate synchronous clocks to the associated clock network. All of the CoolRunner™ devices have one of these clock inputs that are associated with in Input pin. It is worth noting that the input may be used as both the input to the associated clock network and as an input to the logic array (via the ZIA interconnect) at the same time. Thus this input can be used as both a clock and as a signal in the logic simultaneously.

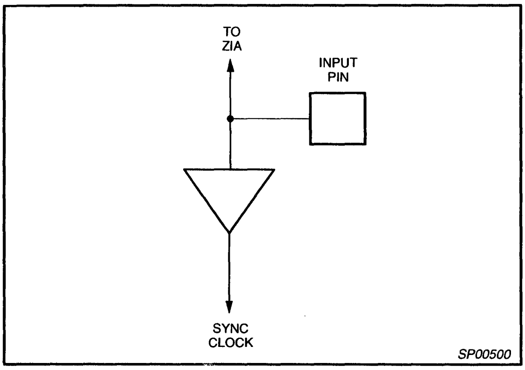


Figure 1.

The second type of input to the clocking network are clock pins that are associated with I/O macrocells. There is one of these input types on the CoolRunner™32, and three of these on the CoolRunner™64 and CoolRunner™128 devices. These clock inputs have more versatility than the dedicated input type. In the synchronous clocking configuration (Figure 2), the output buffer is set to the High-Z state, and the I/O pin is propagated to the associated clock network and the logic array (via the ZIA). This behaves identically to the dedicated input clock. The Macrocell is still usable for internal 'buried' logic in this configuration. Configuration of the macrocell for buried logic and disabling the output buffer is automatically done by the design software.

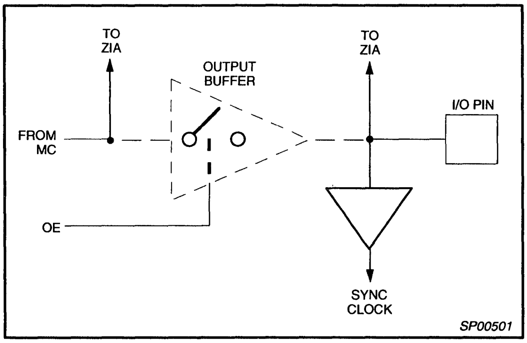


Figure 2.

The generation of asynchronous 'equation-based' clocks is also done using the I/O macrocell based clock input. Figure 3 shows that in this configuration the output buffer is enabled. Therefore, the logic that is generated in the macrocell is propagated to the associated clock network, the I/O pin, and is also fed back into the logic array. Since macrocells in the XPLA architecture can deploy as many as 37 Sum of Product equations, the resulting clocking equation in this configuration may be much more complex than in competing devices that have only a single product term available for asynchronous clocking. It also significant to note that the asynchronous clock that is generated is observable on the associated I/O pin. For this reason, the associated pin should not be terminated by tying to ground or V_{DD}.

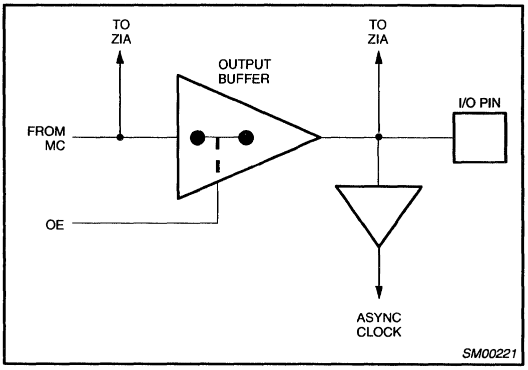


Figure 3.

Using XPLA Designer to generate clocks

XPLA Designer will automatically assign clocks to the correct pins / macrocells based on the context of the clocking desired.
Synchronous clocks are generated by declaring the name you want for the clock, and simply using this by itself in a **.clk** equation, as shown below for the clock signal we've created called CLOCK_1.

```
Module  DEMO
Title   'A simple design:3-bit counter'


CLOCK_1       pin;
bit2..bit0    pin istype 'reg';
count = [bit2..bit0];

equations

count.CLK = CLOCK_1;
count =  count.q + 1;

end;
```

Figure 4.

Asynchronous clocks are also easy to generate. Again, simply declare the variables that will make up the equation that the clock will be based upon. In the example below, we are generating a clock from the variables A, B, and C. Then in the **.clk** equation, we write any expression we want for the clock, and the design software will assign this to a macrocell, enable the output buffer, and generate the corresponding clock equations for you.

```
Module  DEMO2
Title   'A simple design:3-bit asynch counter'


A,B,C         pin;
bit2..bit0    pin istype 'reg';
count = [bit2..bit0];

equations

count.CLK = (A&B) # C;
count = count.q + 1;

end;
```

Figure 5.

Soft Flip-Flops

For the rare cases where there are too few clocks in a CoolRunner® device to implement a large number of input registers (for example), soft D flip-flops or transparent latches may be useful. The following examples illustrate the generation of a transparent latch and D Flip-Flop using only the gates in the logic array.

```
Module  Soft_Latch
Title   'Soft Latch w/ Latch Enable -'

D       pin;
LE      pin;

/* This uses only one macrocell to implement a
transparent latch. It is level (not edge)
triggered - D hold must extend beyond LE's
falling edge */

Q       pin istype 'com,keep,retain';

equations

Q       = (D & LE) # (Q & !LE) # (D & Q);

end
```

Figure 6.

```
Module  Soft_D2
Title   'Soft D Flip Flop -
         Rising Edge triggered -'

/* This edge triggered soft flip-flop uses two
   macrocells. The Input latch opens when the
   clock is low, and closes when the clock is
   high. The output latch (Q) opens when the
   clock is high, and closes when the clock is
   low */

D       pin;
CLK     pin;
IL      node istype 'com,keep,retain';
Q       pin istype 'com,keep,retain';

equations

IL = (D & !CLK) # (IL & CLK) # (D & IL);
Q  = (IL & CLK) # (Q & !CLK) # (IL & Q);end
```

Figure 7.

XPLA Designer™ hierarchical PHDL design support AN066

Author: Reno L. Sanchez

INTRODUCTION

Hierarchical designs are designs which contain multiple levels of circuit descriptions. The upper-level design file usually contains a description of all functional blocks of the design and how these functional blocks are interconnected. The lower-level design file(s) usually contain the circuit details of each of the functional block(s) of the design. Hierarchical designs are very useful especially in two areas:

Parsing very large designs into smaller more manageable functional blocks.

When a design file uses the same functionality in multiple locations of the design.

The XPLA Designer™ supports hierarchical PHDL (Philips Hardware Description Language) designs. This application note provides the necessary information on how to generate hierarchical designs using hierarchical design files. This document first gives the required hierarchy syntax and then gives two design examples using this syntax.

After reading these hierarchical guidelines, you should be well on your way to implementing designs using the hierarchical feature supported by XPLA Designer™ and the PHD Language. This will enable you to take advantage of all the great features the Philips CoolRunner™ CPLDs offer.

Terminology

CoolRunner™	Name of Philips first CPLD family
CPLD	Complex Programmable Logic Device
PHD	Philips Hardware Description
PHDL	Philips Hardware Description Language
PZ5032	Philips 5 Volt CoolRunner 32 Macrocell CPLD

HIERARCHY SYNTAX

Hierarchy declaration is supported in PHDL. An upper-level PHDL module can refer to a lower-level PHDL module. Modules must be defined in files whose names should be identical to the module name with the ".phd" extension. All files for the same design must be contained in the same directory.

To instantiate a PHDL module, some statements have to be written in the following steps:

In the upper-level source:

- ·Declare the lower-level module with a higher-level **Interface** declaration.
- ·Instantiate the module with **Functional_block** declarations.
- ·Specify port mapping in the equations section.

In the lower-level source:

- Identify lower-level I/O ports with a lower-level Interface statement.

A default value for each port can be specified in the interface declaration. However, **only the value specified in the top-level module will be used.** Any other default value in sub-modules will be ignored.

Port Mappings

There are three kinds of port mappings used in the equation section:

```
• instance_name.port_name  =   signal_name;
• signal_name              =   instance_name.port_name;
• instance1_name.port_name =   instance2_name.port_name;
```

Port_name can also be replaced by a group of port_names between [and];

- instance_name.[port1, port2, ...] = [signal_1, signal_2, ...];

However, dot extensions **cannot** be used with port signals and they **cannot** be used in the interface declaration. Port mappings **cannot** be used within an expression as an operand.

A work around for this condition is given in the design example shown in Appendix B. As seen in the equation below, all *.ackn signals are generated by lower-level design files and passed to the upper-design file, CPU_CTL, where these signals are "ORed" together to form the final !ackn signals.

Desired Equation:

```
!ackn = !bus_brm_gen.g_ackn
    # !bus_frm_gen.g_ackn
    # !bus_sif_gen.g_ackn
    # !bus_sio_gen.g_ackn
    # !bus_fld_gen.g_ackn
    # !bus_etc_gen.g_ackn;
```

In order to work around the condition where port mappings cannot be used within an expression as an operand, the upper-level design file, CPU_CTL, uses temporary nodes to create the equation for !ackn as shown below.

PHDL Equation:

```
  !ackn     =  !t_node1 # !t_node2 # !t_node3 # !t_node4 #
               !t_node5 # !t_node6;
```

where:

```
  t_node1   = bus_brm_gen.g_ackn;
  t_node2   = bus_frm_gen.g_ackn;
  t_node3   = bus_sif_gen.g_ackn;
  t_node4   = bus_sio_gen.g_ackn;
  t_node5   = bus_fld_gen.g_ackn;
  t_node6   = bus_etc_gen.g_ackn;
```

Please note that the compiler will re-write the PHDL equation to match the desired equation (i.e., t_node1 through t_node6 will be eliminated if the node collapse feature is enabled). The compiler simply needs these temporary nodes to work around the restriction of not being able to have port mapping in the equation.

HIERARCHICAL DESIGN EXAMPLES

The following subsections contain hierarchical design examples.

Hierarchical Design Example: Two-bit Adder

Appendix A contains a hierarchical design example of a simple two-bit adder. The upper-level design file, adder2.phd, adds two bits by calling a one-bit adder contained in a lower-level design file, adder1.phd. This design example is given to illustrate the PHDL hierarchical syntax and give the reader an easy to understand hierarchical design example.

Hierarchical Design Example: CPU Control Circuitry

Appendix B contains a hierarchical design example of a complex CPU controller circuit. This design has been compiled and fit into a PZ5032 device using the XPLA Designer. This design example is given to illustrate more advanced PHDL hierarchical syntax and usage. Table 2 contains all 9 PHD files which make up the design. When examining this design example, please note how parameters are passed between the upper-level and lower-level design modules and in-between the lower-level design modules.

CLOSING

If you wish to learn more about PHDL, please refer to the XPLA Designer Users Manual. If you have any questions, please contact the Philips CoolRunner Applications Hotline by dialing tollfree 1-888-COOLPLD or 1-505-858-2996.

Table 2. Design Example PHDL Files

Design File	Hierarchy	Description
cpu_ctl	Upper Level	Top level PHDL files which calls all lower-level routines
addrsdec	Lower Level	Address Decode Logic
boot_gen	Lower Level	Boot ROM Single/Throttled Quad Read Cycle Generator
nda_gen	Lower Level	No Device Area Single Read/Write Cycle Generator
load_gen	Lower Level	Flash Loader Area Single Read/Write Cycle Generator
fmem_gen	Lower Level	Flash Memory Single/Throttled Quad Read Cycle Generator
sio1_gen	Lower Level	Serial I/O Single Read/Write Cycle Generator - 1 wait state
sio3_gen	Lower Level	Serial I/O Single Read/Write Cycle Generator – 3 wait states
stat_gen	Lower Level	ROMCONT Status Code Generator

APPENDIX A: HIERARCHY DESIGN — TWO-BIT ADDER

File: adder2.phd — Upper-Level Design File (Two-bit Adder)

```
Module      adder2;
Tile        'Two-bit Adder'
Declarations
     a0, a1, b0, b1, cin, cout, so, s1   pin;
     adder1 interface (a, b, cin -> sum, cout);
     adder1_0 functional_block adder1;
     adder1_1 functional_block adder1;
Equations
     adder1_0.a   = a0;
     adder1_0.b   = b0;
     adder1.cin   = cin;
     s0           = adder1_0.sum;
     adder1_1.cin = adder1_0.cout;
     adder1_1.a   = a1;
     adder1_1.b   = b1;
     s1           = adder1_1.sum;
     cout         = adder1_1.sum;
end
```

File: adder1.phd — Lower-Level Design File (One-bit Adder)

```
Module      adder1
Title       'One-bit Adder'
Declarations
     a, b, cin, cout, sum     pin;
Equations
     sum    = a & b # a & cin # b & cin;
     cout   = a $ b $ cin;
end
```

APPENDIX B: HIERARCHY DESIGN – CPU CONTROLLER

File: cpu_ctl.phd — Upper-Level Design File

```
Module cpu_ctl

Title 'CPU Controller'

/*
*****************************************************************
*                                                               *
*   File Name    - cpu_ctl.phd                                  *
*   Function     - CPU Controller Circuit                       *
*                                                               *
*****************************************************************
*/

Declarations
         ad27..ad21, rdn, wrn, burstn       pin;
         dataenn, sysclk0n, busgntn, resetn pin;

         brom_csn, brom_oen                 pin     istype 'com';
         flsh_csn, flsh_oen                 pin     istype 'com';
         flsh_wen                           pin     istype 'reg_d';
         sio_cs1n, sio_cs2n                 pin     istype 'com';
         sio_cs3n, sio_cs4n                 pin     istype 'com';
         sio_oen                            pin     istype 'com';
         sio_wen                            pin     istype 'reg_d';
         sif_csn, sif_oen                   pin     istype 'com';
         sif_wen                            pin     istype 'reg_d';
         ackn, rdcenn                       pin     istype 'com';

         g_sts3..g_sts0                     node    istype 'reg';
         t_node0                            node    istype 'com,keep';        "temporary nodes
         t_node12..t_node1                  node     istype 'com';             "temporary nodes


"---------------------------------------------"
"        Address Generation                   "
"---------------------------------------------"
         addrsdec interface (addrH3..addrH0, addrL2..addrL0, g_busgntn ->
                             fld_decn, sif_decn, sio_decn, sio_dec1n, sio_dec2n, sio_dec3n,
                             sio_dec4n, frm_decn, brm_decn, nrw_decn);
         addr_dec_gen functional_block addrsdec;


"---------------------------------------------"
"        4bits Status Counter                 "
"---------------------------------------------"
         stat_gen interface (g_clrn, g_clk -> g_sts3..g_sts0);
         bus_statusC functional_block stat_gen;


"---------------------------------------------"
"        SIO   Bus Cycle Gen. (1wait)         "
"---------------------------------------------"
         sio1_gen interface (g_decn, g_clk, g_rdn, g_wrn, g_resetn, g_dataenn, g_c3..g_c0 ->
                             g_csn, g_oen, g_wen, g_ackn, g_rdcenn);
         bus_sif_gen functional_block sio1_gen;
```

```
"---------------------------------------------"
"        SIO   Bus Cycle Gen. (3wait)         "
"---------------------------------------------"
         sio3_gen interface (g_decn, g_dec1n, g_dec2n, g_dec3n, g_dec4n, g_clk, g_rdn,
                                   g_wrn, g_resetn, g_dataenn, g_c3..g_c0 ->
                                   g_cs1n, g_cs2n, g_cs3n, g_cs4n, g_oen, g_wen, g_ackn, g_rdcenn);
         bus_sio_gen functional_block sio3_gen;


"---------------------------------------------"
"        Flash Bus Cycle Gen. (3wait)         "
"---------------------------------------------"
         fmem_gen interface (g_decn, g_clk, g_rdn, g_wrn, g_resetn, g_burstn, g_dataenn,
                                   g_c3..g_c0 -> g_csn, g_oen, g_wen, g_ackn, g_rdcenn);
         bus_frm_gen functional_block fmem_gen;


"---------------------------------------------"
"        Boot  Bus Cycle Gen. (3wait)         "
"---------------------------------------------"
         boot_gen interface (g_decn, g_rdn, g_resetn, g_burstn, g_dataenn, g_c3..g_c0 ->
                                   g_csn, g_oen, g_ackn, g_rdcenn);
         bus_brm_gen functional_block boot_gen;


"---------------------------------------------"
"        Fload Bus Cycle Gen. (3wait)         "
"---------------------------------------------"
         fld_gen interface (g_decn, g_dataenn, g_c3..g_c0 -> g_ackn, g_rdcenn);
         bus_fld_gen functional_block fld_gen;


"---------------------------------------------"
"        No Device Cycle Gen. (1wait)         "
"---------------------------------------------"
         nda_gen interface (g_decn, g_dataenn, g_c3..g_c0 -> g_ackn, g_rdcenn);
         bus_etc_gen functional_block nda_gen;
```

```
Equations

"
"       Set Input Address(addr) of Address Generator
"
        addr_dec_gen.addrH3     = ad27;
        addr_dec_gen.addrH2     = ad26;
        addr_dec_gen.addrH1     = ad25;
        addr_dec_gen.addrH0     = ad24;

        addr_dec_gen.addrL2     = ad23;
        addr_dec_gen.addrL1     = ad22;
        addr_dec_gen.addrL0     = ad21;

        addr_dec_gen.g_busgntn =  busgntn;


"
"       Set 4bits Status Counter
"
        !t_node0                 =  !resetn # (rdn & wrn);
        bus_statusC.g_clrn       =  t_node0;
        bus_statusC.g_clk        =  sysclk0n;


"
"       Create Control Signal(CSn,OEn,WEn) of Serial I/F
"
"       Setup Signals

     bus_sif_gen.g_decn         =  addr_dec_gen.sif_decn;
     bus_sif_gen.g_clk          =  sysclk0n;
     bus_sif_gen.g_rdn          =  rdn;
     bus_sif_gen.g_wrn          =  wrn;
     bus_sif_gen.g_resetn       =  resetn;
     bus_sif_gen.g_dataenn      =  dataenn;

     bus_sif_gen.g_c0           = bus_statusC.g_sts0;
     bus_sif_gen.g_c1           = bus_statusC.g_sts1;
     bus_sif_gen.g_c2           = bus_statusC.g_sts2;
     bus_sif_gen.g_c3           = bus_statusC.g_sts3;


     sif_csn                    =  bus_sif_gen.g_csn;         " CSn
     sif_oen                    =  bus_sif_gen.g_oen;         " OEn
     sif_wen                    =  bus_sif_gen.g_wen;         " WEn


"
"    Create Control Signal(CSn,OEn,WEn) of Serial I/O
"
"    Setup Signals

   bus_sio_gen.g_decn           =  addr_dec_gen.sio_decn;
   bus_sio_gen.g_dec1n          =  addr_dec_gen.sio_dec1n;
   bus_sio_gen.g_dec2n          =  addr_dec_gen.sio_dec2n;
   bus_sio_gen.g_dec3n          =  addr_dec_gen.sio_dec3n;
   bus_sio_gen.g_dec4n          =  addr_dec_gen.sio_dec4n;
   bus_sio_gen.g_clk            =  sysclk0n;
   bus_sio_gen.g_rdn            =  rdn;
   bus_sio_gen.g_wrn            =  wrn;
   bus_sio_gen.g_resetn         =  resetn;
   bus_sio_gen.g_dataenn        =  dataenn;
   bus_sio_gen.g_c0             = bus_statusC.g_sts0;
```

```
    bus_sio_gen.g_c1            = bus_statusC.g_sts1;
    bus_sio_gen.g_c2            = bus_statusC.g_sts2;
    bus_sio_gen.g_c3            = bus_statusC.g_sts3;

    sio_cs1n                    =  bus_sio_gen.g_cs1n;       " CSn
    sio_cs2n                    =  bus_sio_gen.g_cs2n;
    sio_cs3n                    =  bus_sio_gen.g_cs3n;
    sio_cs4n                    =  bus_sio_gen.g_cs4n;
    sio_oen                     =  bus_sio_gen.g_oen;        " OEn
    sio_wen                     =  bus_sio_gen.g_wen;        " WEn


"
"    Create Control Signal(CSn,OEn) of Flash(Apli) ROM
"
"    Setup Signals

    bus_frm_gen.g_decn          = addr_dec_gen.frm_decn;
    bus_frm_gen.g_clk           = sysclk0n;
    bus_frm_gen.g_rdn           = rdn;
    bus_frm_gen.g_wrn           = wrn;
    bus_frm_gen.g_resetn        = resetn;
    bus_frm_gen.g_burstn        = burstn;
    bus_frm_gen.g_dataenn       = dataenn;
    bus_frm_gen.g_c0            = bus_statusC.g_sts0;
    bus_frm_gen.g_c1            = bus_statusC.g_sts1;
    bus_frm_gen.g_c2            = bus_statusC.g_sts2;
    bus_frm_gen.g_c3            = bus_statusC.g_sts3;

    flsh_csn                    = bus_frm_gen.g_csn;         " CSn
    flsh_oen                    = bus_frm_gen.g_oen;         " OEn
    flsh_wen                    = bus_frm_gen.g_wen;         " WEn


"
"    Create Control Signal(CSn,OEn) of Boot ROM
"
"    Setup Signals

    bus_brm_gen.g_decn          = addr_dec_gen.brm_decn;
    bus_brm_gen.g_rdn           = rdn;
    bus_brm_gen.g_resetn        = resetn;
    bus_brm_gen.g_burstn        = burstn;
    bus_brm_gen.g_dataenn       = dataenn;
    bus_brm_gen.g_c0            = bus_statusC.g_sts0;
    bus_brm_gen.g_c1            = bus_statusC.g_sts1;
    bus_brm_gen.g_c2            = bus_statusC.g_sts2;
    bus_brm_gen.g_c3            = bus_statusC.g_sts3;
    brom_csn                    = bus_brm_gen.g_csn;         " CSn
    brom_oen                    = bus_brm_gen.g_oen;         " OEn
"
"    Create Control Signal of Flash Loader Area
"
"    Setup Signals
    bus_fld_gen.g_decn          = addr_dec_gen.fld_decn;
    bus_fld_gen.g_dataenn       = dataenn;
    bus_fld_gen.g_c0            = bus_statusC.g_sts0;
    bus_fld_gen.g_c1            = bus_statusC.g_sts1;
    bus_fld_gen.g_c2            = bus_statusC.g_sts2;
    bus_fld_gen.g_c3            = bus_statusC.g_sts3;
```

```
"
"    Create Control Signal(CSn) of No Device Area
"
"    Setup Signals

   bus_etc_gen.g_decn          = addr_dec_gen.nrw_decn;
   bus_etc_gen.g_dataenn       = dataenn;
   bus_etc_gen.g_c0            = bus_statusC.g_sts0;
   bus_etc_gen.g_c1            = bus_statusC.g_sts1;
   bus_etc_gen.g_c2            = bus_statusC.g_sts2;
   bus_etc_gen.g_c3            = bus_statusC.g_sts3;
"
"    Create Control Signal(ACKn,RDCENn)
"
"
"    ACKn - I had to re-write this using temporary nodes
"
"  !ackn     = !bus_brm_gen.g_ackn # !bus_frm_gen.g_ackn # !bus_sif_gen.g_ackn
"              # !bus_sio_gen.g_ackn # !bus_fld_gen.g_ackn
"              # !bus_etc_gen.g_ackn;

    t_node1       = bus_brm_gen.g_ackn;
    t_node2       = bus_frm_gen.g_ackn;
    t_node3       = bus_sif_gen.g_ackn;
    t_node4       = bus_sio_gen.g_ackn;
    t_node5       = bus_fld_gen.g_ackn;
    t_node6       = bus_etc_gen.g_ackn;

   !ackn          = !t_node1 # !t_node2 # !t_node3 # !t_node4 # !t_node5 # !t_node6;
"
"    RDCENn - I had to re-write this using temporary nodes
"
"  !rdcenn  = !bus_brm_gen.g_rdcenn # !bus_frm_gen.g_rdcenn # !bus_sif_gen.g_rdcenn
"            # !bus_sio_gen.g_rdcenn # !bus_fld_gen.g_rdcenn
"            # !bus_etc_gen.g_rdcenn;

    t_node7       = bus_brm_gen.g_rdcenn;
    t_node8       = bus_frm_gen.g_rdcenn;
    t_node9       = bus_sif_gen.g_rdcenn;
    t_node10      = bus_sio_gen.g_rdcenn;
    t_node11      = bus_fld_gen.g_rdcenn;
    t_node12      = bus_etc_gen.g_rdcenn;

   !rdcenn        = !t_node7 # !t_node8 # !t_node9 # !t_node10 # !t_node11 # !t_node12;

End
```

File: addrsdec.phd — Lower-Level Design File

```
Module addrsdec

Title 'Address Decode Circuitry'

/*
 *******************************************************************
 *                                                                 *
 *      File Name       - addrsdec.phd                             *
 *      Function        - Address Decode  Circuitry                *
 *                                                                 *
 *******************************************************************

    CPU Memory MAP

    Flash Loader        : 0x?180 0000 - 0x?1ff ffff (withCentronix)
    SIF                 : 0x?980 0000 - 0x?9ff ffff
    SIO CN1             : 0x?b00 0000 - 0x?b1f ffff
    SIO CN2(ISR)        : 0x?b20 0000 - 0x?b3f ffff
    SIO CN3             : 0x?b40 0000 - 0x?b5f ffff
    SIO CN4             : 0x?b60 0000 - 0x?b7f ffff
    Flash ROM (Appli)   : 0x?f80 0000 - 0x?fbf ffff
    Boot  ROM           : 0x?fc0 0000 - 0x?fff ffff
    No Device           : 0x?0c0 0000 - 0x?0ff ffff (Read/Write NG)
                          0x?500 0000 - 0x?8ff ffff (Read/Write NG)
                          0x?e00 0000 - 0x?f7f ffff (Read/Write NG)
*/
Declarations
   addrH3..addrH0, addrL2..addrL0, g_busgntn        pin;
   fld_decn, sif_decn, sio_decn                     pin istype 'com';
   sio_dec1n, sio_dec2n, sio_dec3n, sio_dec4n       pin istype 'com';
   frm_decn, brm_decn, nrw_decn                     pin istype 'com';

   addrH      = [addrH3..addrH0];
   addrL      = [addrL2..addrL0];

Equations
   !fld_decn  = g_busgntn & (addrH == ^b0001) & addrL2;
   !sif_decn  = g_busgntn & (addrH == ^b1001) & addrL2;
   !sio_decn  = g_busgntn & (addrH == ^b1011) & !addrL2;
   !sio_dec1n = g_busgntn & (addrH == ^b1011) & (addrL == ^b000);
   !sio_dec2n = g_busgntn & (addrH == ^b1011) & (addrL == ^b001);
   !sio_dec3n = g_busgntn & (addrH == ^b1011) & (addrL == ^b010);
   !sio_dec4n = g_busgntn & (addrH == ^b1011) & (addrL == ^b011);
   !frm_decn  = g_busgntn & (addrH == ^b1111) & addrL2 & !addrL1;
   !brm_decn  = g_busgntn & (addrH == ^b1111) & addrL2 &  addrL1;
   !nrw_decn  = (((addrH == ^b0000) &  addrL2 &  addrL1)
              # ((addrH >= ^b0101) & (addrH <= ^b1000))
              #  (addrH == ^b1110)
              # ((addrH == ^b1111) & !addrL2));
End
```

File: boot_gen.phd — Lower-Level Design File

```
Module boot_gen

Title ' BOOT ROM Single/Throttled Quad Read Cycle Generator'

*/
 ********************************************************************************
 *                                                                              *
 *       File Name       - boot_gen.phd                                         *
 *       Function        - Boot ROM Single/Throttled Quad Read Cycle Generator  *
 *                                                                              *
 ********************************************************************************
*/

Declarations

        g_decn, g_rdn, g_resetn           pin;
        g_burstn, g_dataenn, g_c3..g_c0   pin;
        g_csn, g_oen, g_ackn, g_rdcenn    pin istype 'com';

        g_c       = [g_c3..g_c0];


Equations
/* This Gen. supports between single read and throttled quad read.
   This Gen. supports that Tacc(Access Time) is 100ns + Driver.
        Single bus(read/write) cycle span is 4 cycles (3wait).
        Throttled quad read cycle span is 15 cycles (3wait x 4).
*/


"       Create Control Signal(CSn,OEn)

        !g_csn     = g_resetn & !g_rdn;
        !g_oen     = !g_decn & !g_rdn & !g_dataenn;


"       Create Control Signal(ACKn,RDCENn)

        !g_ackn   = !g_decn & ((g_burstn &  (g_c == ^b0011)) # (!g_burstn & (g_c == ^b1100)));
        !g_rdcenn = !g_decn & !g_dataenn & ((g_c == ^b0011)
                    # (!g_burstn & ((g_c == ^b0111) # (g_c == ^b1011) # (g_c == ^b1111))));

End
```

File: nda_gen.phd — Lower-Level Design File

```
Module nda_gen

Title ' No Device Area Single Read/Write Cycle Generator'

/*
 ****************************************************************************
 *                                                                          *
 *      File Name       - nda_gen.phd                                       *
 *      Function        - No Device Area Single Read/Write Cycle Generator  *
 *                                                                          *
 ****************************************************************************
*/

Declarations

        g_decn                      pin;
        g_dataenn, g_c3..g_c0       pin;
        g_ackn, g_rdcenn            pin istype 'com';

        g_c = [g_c3..g_c0];

Equations
/*
        This Gen. supports single read and single write bus transaction.
                Single read cycle span is 3 cycles (0wait).
                Single write cycle span is 3 cycles (0wait).
*/

"       Create Control Signal(ACKn,RDCENn)

        !g_ackn         = !g_decn & (g_c == ^b0001);
        !g_rdcenn       = !g_decn & !g_dataenn & (g_c == ^b0001);

End
```

File: fld_gen.phd — Lower-Level Design File

```
Module fld_gen

Title ' Flash Loader Area Single Read/Write Cycle Generator'

/*
 *********************************************************************************
 *                                                                               *
 *      File Name      - ld_gen.phd                                              *
 *      Function       - Flash Loader Area Single Read/Write Cycle Generator     *
 *                                                                               *
 *********************************************************************************

*/

Declarations

        g_decn                     pin;
        g_dataenn, g_c3..g_c0      pin;
        g_ackn, g_rdcennpin istype 'com';

        g_c     = [g_c3..g_c0];

Equations

"       This Gen. supports single read and single write bus transaction.
"       Single read cycle span is 6 cycles (3wait).
"       Single write cycle span is 6 cycles (3wait).

"       Create Control Signal(ACKn,RDCENn)

        !g_ackn        = !g_decn & (g_c == ^b0011);
        !g_rdcenn      = !g_decn & !g_dataenn & (g_c == ^b0011);

End
```

File: fmem_gen.phd — Lower-Level Design File

```
Module fmem_gen

Title ' Flash Memory Single/Throttled Quad Read Cycle Generator'

/*
 ********************************************************************************
 *                                                                              *
 *      File Name       - fmem_gen.phd                                          *
 *      Function        - Flash Memory Single/Throttled Quad Read Cycle Generator *
 *                                                                              *
 ********************************************************************************
*/

Declarations
        g_decn, g_clk, g_rdn, g_wrn, g_resetn       pin;
        g_burstn, g_dataenn, g_c3..g_c0             pin;
        g_csn, g_oen                                pin istype 'com';
        g_wen                                       pin istype 'reg_d';
        g_ackn, g_rdcenn                            pin istype 'com';
        g_c     = [g_c3..g_c0];

Equations
/*      This Gen. supports between single read and throttled quad read, and, single write.
        This Gen. supports that Tacc(Access Time) is 120ns.
        Single read bus cycle span is 6 cycles (3wait).
        Throttled quad read cycle span is  19 cycles (3w x 4).
        Single write bus cycle span is 6 cycles (3wait).
*/

"       Create Control Signal(CSn,OEn)

        !g_csn   =  g_resetn & (!g_rdn # !g_wrn);               " CSn
        !g_oen   = !g_decn & !g_rdn & !g_dataenn;               " OEn

"       Create Control Signal(WEn)

         g_wen.ap  = !g_decn & !g_wrn;
         g_wen.clk =  g_clk;
        !g_wen.d   = (g_c <= ^b0010);

"       Create Control Signal(ACKn,RDCENn)

        !g_ackn     = !g_decn & ((g_burstn & (g_c == ^b0011)) # (!g_burstn & (g_c == ^b1100)));
        !g_rdcenn   = !g_decn & !g_dataenn & ((g_c == ^b0011)
           # (!g_burstn & ((g_c == ^b0111) # (g_c == ^b1011) # (g_c == ^b1111))));

End
```

File: sio1_gen.phd — Lower-Level Design File

```
Module sio1_gen

Title ' Serial I/O Single Read/Write Cycle Generator - 1 Wait States'

/*
 ***************************************************************************************
 *                                                                                     *
 *      File Name       - sio1_gen.phd                                                 *
 *      Function        - Serial I/O Single Read/Write Cycle Generator - 1 Wait State *
 *                                                                                     *
 ***************************************************************************************
*/

Declarations
        g_decn, g_clk, g_rdn, g_wrn         pin;
        g_resetn, g_dataenn, g_c3..g_c0     pin;
        g_csn, g_oen                        pin istype 'com';
        g_wen                               pin istype 'reg_d';
        g_ackn, g_rdcenn                    pin istype 'com';

        g_c      = [g_c3..g_c0];

Equations
/*      This Gen. supports single read and single write bus transaction.
        This Gen. supports that Tacc(Access Time) is 42.1ns.
        Single read cycle span is 3 cycles (1wait).
        Single write cycle span is 3 cycles (1wait).
*/

"       Create Control Signal(CSn,OEn)

        !g_csn         =  g_resetn & (!g_rdn # !g_wrn);                " CSn
        !g_oen         = !g_decn & !g_rdn & !g_dataenn;                " OEn

"       Create Control Signal(WEn)

         g_wen.ap       = !g_decn & !g_wrn;
         g_wen.clk      =  g_clk;
        !g_wen.d        = (g_c == ^b0000);                             " WEn

"       Create Control Signal(ACKn,RDCENn)

        !g_ackn         = !g_decn & (g_c == ^b0001);
        !g_rdcenn       = !g_decn & !g_dataenn & (g_c == ^b0001);

End
```

File: sio3_gen.phd — Lower-Level Design File

```
Module sio3_gen

Title ' Serial I/O Single Read/Write Cycle Generator - 3 Wait States'

/*
 ******************************************************************************
 *                                                                            *
 *      File Name      - sio3_gen.phd                                         *
 *      Function       - Serial I/O Single Read/Write Cycle Generator -3 Wait States*
 *                                                                            *
 ******************************************************************************
*/

Declarations
         g_decn, g_dec1n, g_dec2n, g_dec3n, g_dec4n        pin;
         g_clk, g_rdn                                      pin;
         g_wrn, g_resetn, g_dataenn, g_c3..g_c0            pin;
         g_cs1n, g_cs2n, g_cs3n, g_cs4n                    pin istype 'com';
         g_oen                                             pin istype 'com';
         g_wen                                             pin istype 'reg_d';
         g_ackn, g_rdcenn                                  pin istype 'com';

         g_c     =  [g_c3..g_c0];

Equations
/*      This Gen. supports single read and single write bus transaction.
        This Gen. supports that Tacc(Access Time) is 95ns.
        Single read cycle span is 5 cycles (3wait).
        Single write cycle span is 5 cycles (3wait).
*/
"       Create Control Signal(CS1n,CS2n,OEn)

        !g_cs1n =  g_resetn & !g_dec1n & (!g_rdn # !g_wrn);        " CS1n
        !g_cs2n =  g_resetn & !g_dec2n & (!g_rdn # !g_wrn);        " CS2n
        !g_cs3n =  g_resetn & !g_dec3n & (!g_rdn # !g_wrn);        " CS3n
        !g_cs4n =  g_resetn & !g_dec4n & (!g_rdn # !g_wrn);        " CS4n

        !g_oen  = !g_decn & !g_rdn & !g_dataenn;                   " OEn

"       Create Control Signal(WEn)

        g_wen.ap      = !g_decn & !g_wrn;
        g_wen.clk     =  g_clk;
       !g_wen.d       = (g_c <= ^b0010);                          "WEn

"       Create Control Signal(ACKn,RDCENn)

       !g_ackn        = !g_decn & (g_c == ^b0011);
       !g_rdcenn      = !g_decn & !g_dataenn & (g_c == ^b0011);
End
```

File: stat_gen.phd — Lower-Level Design File

```
Module stat_gen

Title 'ROMCONT status code generator'

/*
 ********************************************************************************
 *                                                                              *
 *      File Name           - stat_gen.phd                                      *
 *      Function            - ROMCONT status code generator                     *
 *                                                                              *
 ********************************************************************************
*/

Declarations
   g_clrn, g_clk             pin;
   g_sts3..g_sts0            pin istype 'reg_d';


Equations
  !g_sts0.ar     = !g_clrn;                        " b0
   g_sts0.clk    =  g_clk;
   g_sts0.d      = !g_sts0.q;

  !g_sts1.ar     = !g_clrn;                       " b1
   g_sts1.clk    =  g_clk;
   g_sts1.d      =  g_sts1.q $ g_sts0.q;

  !g_sts2.ar     = !g_clrn;                       " b2
   g_sts2.clk    =  g_clk;
   g_sts2.d      =  g_sts2.q $ (g_sts1.q & g_sts0.q);

  !g_sts3.ar     = !g_clrn;                       " b3
   g_sts3.clk    =  g_clk;
   g_sts3.d      =  g_sts3.q $ (g_sts2.q & g_sts1.q & g_sts0.q);

End
```

JEDEC file equation generator

AN067

INTRODUCTION

Have you ever found yourself in a situation where the only remnants of your design is a JEDEC file and the CPLD you are using has been discontinued? Believe it or not, this situation occurs often, most particularly in government and military applications. This application note describes a Philips methodology and tool, name JED2EQU, which allows the user to reproduce the equations of the old CPLD and fit them into a Philips CoolRunner CPLD. As you may be away, using the Philips CoolRunner CPLD gives the user many industry advantages including: Lowest standby and dynamic power, best re-routability with fixed pins, more logic per equivalent Macrocell count device, etc.

Terminology

ABEL	Data I/O Hardware Description Language
CoolRunner	Name of Philips first CPLD family
JEDEC	Name of industry standard program file format
JED2EQU	JEDEC to Equation conversion tool
PHDL	Philips Hardware Description Language
LanguageSynario	Design tool from Data I/O

CONVERSION METHODOLOGY

The Philips JEDEC to Equation methodology can be summarized by the following steps:

Use the JED2EQU tool to convert the JEDEC file into generic equations. The Philips JED2EQU tool can handle JEDEC files produced by most manufacturers.

Convert these generic equations to ABEL/PHDL equations.

Compile the ABEL / PHDL equations using an industry stand tools such as Synario for the existing device to create a JEDEC file.

Compare old and new JEDEC files to validate the equation conversion.

Compile validate ABEL/PHLD equations for the Philips CoolRunner CPLD.

CLOSING

If you wish to learn more about JED2EQN, please contact the Philips CoolRunner Applications Hotline by
dialing tollfree 1-888-COOLPLD or 1-505-858-2996.

Terminating unused CoolRunner™ I/O pins

AN068

The CoolRunner™ family of CPLDs are the first PLDs to employ a TotalCMOS™ design methodology. Since these devices are fabricated on a 0.5 micron CMOS process technology it is important to consider the options available in terminating unused pins.

CoolRunner™ 32

The CoolRunner™ 32 is only available in packages that bring all of the I/Os to pins. As a result, this device does not have programmable on-chip pull-down circuits. Therefore, we recommend using 10kΩ pull-up resistors on all inputs or I/Os that are not used. This will allow the flexibility of using these pins should late design changes require additional I/O. These unused pins may also be tied directly to V_{DD}, but this will make it more difficult to reclaim the use of the pin should this be needed by a subsequent design revision.

CoolRunner™ 64 and CoolRunner™ 128

The CoolRunner™ 64 and CoolRunner™ 128 devices are offered in packages that do not bond out all of the I/O pins. As a result, these devices have programmable on-chip pull-down circuits on all of the I/O pins. The device fitters in the software automatically activate the on-chip pull-down circuit on all unused I/O pins. No current is consumed by the device when the internal pull-down is connected to the High-Z output buffer. If you connect the pin to V_{DD}, the device would only sink current, not source it, therefore there is no pull-down current incorporated into the I_{DD} specification. The pull-down is very weak, and when the pin is connected to 5 volts, the maximum sink current is less than 10 microamps per I/O. Note that the fitting software considers buried macrocells that do not use the pin for an input as unused, and activates the on-chip pull-down. It is our recommendation that the unused I/O pins be left unconnected on CoolRunner™ 64 and CoolRunner™ 128 designs.

There is no associated pull-down circuit on the dedicated input pins. Therefore, we would again recommend the use of external 10kΩ pull-up resistors on all inputs to maintain maximum design flexibility should any changes be required at a later date.

Philips' XPLA™ Designer CPLD synthesis tool allows the user the ability to disable the pull-down circuit on the CoolRunner™ 64 and CoolRunner™ 128 if desired. This is accomplished via the following property statements. The statement

```
xpla property 'tri-state all';
```

disables the pull-down circuit on all of the unused I/O pins. If disabling the pull-down on a single pin is desired, this is done with the statement

```
xpla property 'dingo:12 tri-state';
```

disables the pull-down on pin12, which has been named dingo. The symbol name 'dingo' need not be declared separately.

ISP design considerations for the CoolRunner™ PZx128 CPLD

AN069

Author: B. Wade Baker, Senior CPLD Specialist

ISP DESIGN CONSIDERATIONS FOR THE COOLRUNNER™ PZX128 CPLD

With the introduction of the Coolrunner PZ5128 and PZ3128 CPLDs, Philips Semiconductors enters the realm of ISP, (In System Programming). Using these 128 macrocell devices, it is possible to change your system hardware at will. This application note addresses board design considerations, i.e., signal integrity, power supply decoupling, power supply filtering, and component placement that will allow you to utilize the advantages of Coolrunner ISP, in an actual design environment, without headaches.

SIGNAL INTEGRITY

As with any high speed CMOS device, the Coolrunner ISP CPLDs are susceptible to noise as input signals transition from low to high or high to low. This is because CMOS transistors are voltage controlled devices and their input impedances are **VERY** high, typically 1015 or more. Therefore even small voltages coupled to the true input signal, during the time when the input has driven the output into its linear region, may cause the output to oscillate, spike, or otherwise behave erratically. This may also induce feelings of unease on your part. Especially if you didn't think about this until AFTER the PCB was laid out. The ISP buffers on the PZx128 have protection for this problem in the form of hysteresis and spike rejection; however, no amount of internal device protection can make up for less than adequate board design. If you plan to utilize the ISP capability, it is important to consider the effects of noise pickup and crosstalk. Noise may be introduced into your ISP lines in two main ways:

- Sources external to your board
- Sources on your board

EXTERNAL SOURCES

If you choose to program your Coolrunner ISP CPLD via a download cable, pay close attention to it's construction, (or better still, purchase one from us). The lab, or cubicle, where you work typically contains many sources of radiated electromagnetic energy (read noise). Fluorescent lighting, clock radios, power supplies, test equipment, coffee makers, your computer, etc. can all contribute to a less than benign external environment. Figure 1 illustrates two methods of download cable construction we refer to as 'bad'. As you can see in Figure 1A we have actually built a reasonably good antenna for the reception of unwanted noise. If you build a cable like this you may have problems. The cable design depicted in Figure 1B is a reasonably adequate design for the attenuation of external noise. The problem with this cable, however, is crosstalk. We performed extensive testing on a cable constructed as shown in Figure 1B and discovered that significant amounts of energy could be coupled across individual conductors when they ran parallel to one another. Fast edge rates coupled with sufficient drive capability produced enough energy to induce false clocking in adjacent conductors. The amount of energy coupled could be influenced by the orientation of the cable! Clearly not something you would want to rely upon when trying to download new hardware in the field.

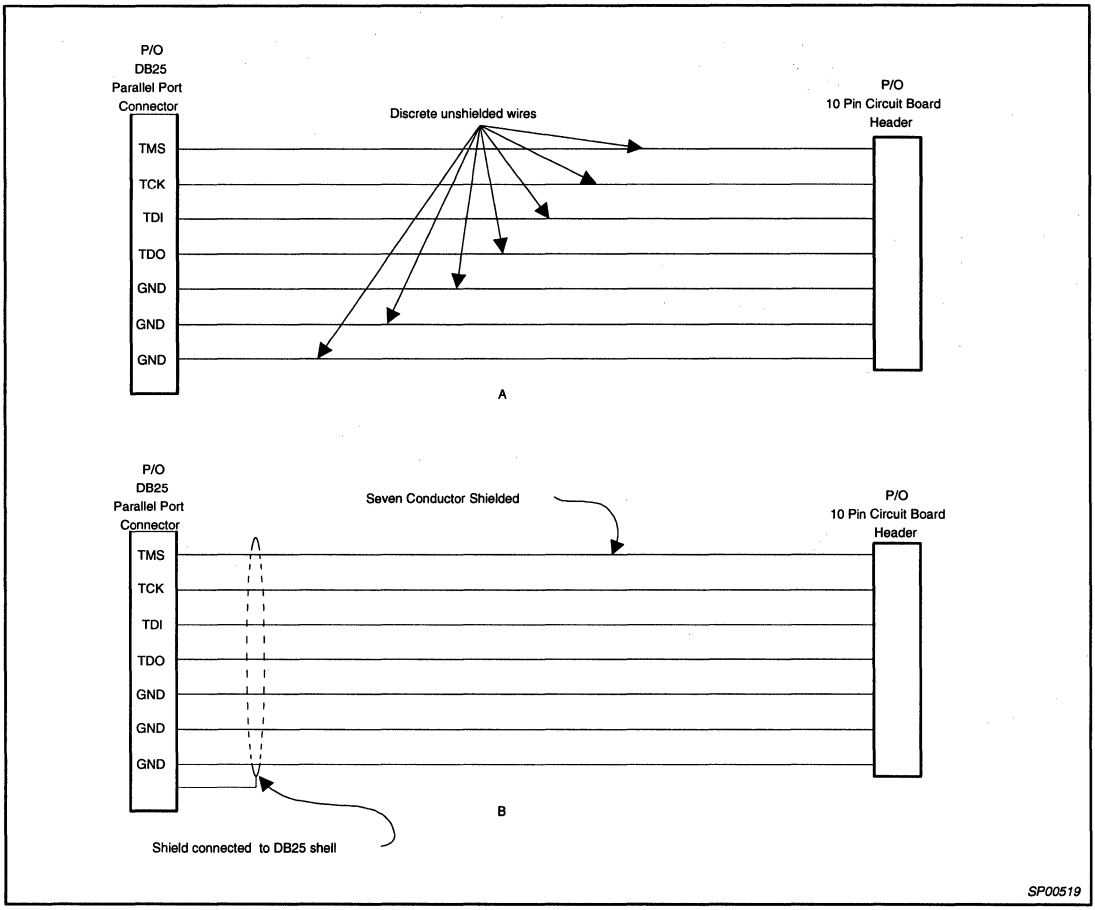


Figure 1. 'Bad' Download Cable Design

Figure 2 depicts a 'good' cable design that has proven itself to be reliable over a wide range of voltage and noise conditions, including many different types and makes of computers.

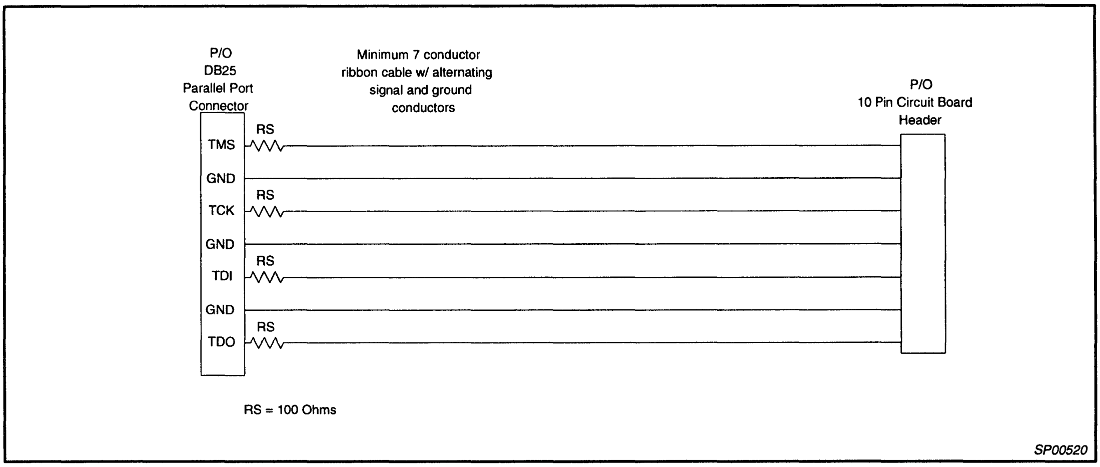


Figure 2. 'Good' Download Cable Design

In this cable the signal conductors are separated from each other by interleaved ground conductors. This arrangement maintains reasonably uniform capacitance throughout the cable length and, to some degree, represents an equivalence to the classical wire over ground transmission line. This combination produces a cable with excellent noise attenuation and crosstalk rejection. The 100 resistors in series with the signal conductors, (located at the computer DB25 connector) serve two purposes. The first is voltage decoupling, the second is series damping. It is unlikely that the output high voltage present on a computer's parallel port data lines and VCC for the PZx128 will be the same value. This is especially true when programming 3 volt parts from a 5 volt computer interface. In this situation, the series resistor acts as a voltage decoupler, preventing potentially damaging current from flowing through the device's input protection diode. Series damping is an arcane method of terminating a transmission line. In this form, a series resistor plus the output impedance of the source combine to match the characteristic impedance of the transmission line. For the Philips download cable, 100 Ohms represents a good compromise between output impedance, cable impedance, and standard resistor values. With the series resistor located as close as possible to the signal source it is possible to reduce the source reflection coefficient to practically zero. This absorbs the wave reflection coming back from the load thus terminating the reflection cycle.

The advantages of this type of termination scheme are:

- Very little power is consumed. This is especially important in low power applications, those for which the Coolrunner is perfect.
- Requires only one resistor per signal line.

The disadvantage of this technique is that for a time, (2tpd) the voltage present along the line is 0.5VOUT. This is not a problem in this application, however, since the signal connections are at the end of the line.

ON BOARD SOURCES

High performance devices that populate most circuit boards today routinely produce output edge rates of 2 to 4 nS. If the trace length, from device to device, is long enough, transmission line effects must be taken into account. One can calculate when to start worrying by using the formula:

$$L = \frac{t_r}{2\,t_{pd}} \times \frac{1}{\sqrt{1 + \frac{C_D Z_O}{t_r}}}$$

Where:

- L = trace length
- tr = rise time
- tpd = calculated propagation delay
- CD = trace capacitance
- ZO = calculated characteristic impedance of the trace

Table 1 was generated using this formula. A line propagation delay of approximately 2.3 nS per foot is assumed for the above calculation. This is a typical delay for well constructed printed circuit boards.

Table 1 When to worry about reflections

tr IN NANOSECONDS	CD IN PICOFARADS	L X 12 TO CONVERT TO INCHES
4	10	9.99
4	20	9.46
4	40	8.63
4	80	7.47
2	10	4.73
2	20	4.32
2	40	3.74
2	80	3.05

If your PCB trace length for high clock speed signals has met the criteria for transmission lines, you must terminate the lines. The exact methods and procedures for line termination are beyond the scope of this application note. Suffice it to say, reflections can be considered noise and should be eliminated or reduced on all PCB traces, not just the ones used for ISP. Crosstalk between adjacent PCB traces is also a noise source to watch for. Try not to run signal lines parallel to one another or, if you must, interleave the signal traces with ground traces. Insure that adequate copper exists to prevent, or greatly reduce, L di/dt rises in ground or dips in VCC. It is also a good idea to have only one point on the board from which ground radiates, this will reduce the possibility of generating ground loops. Please do not tie your analog and digital grounds together, except at the one common point mentioned above. If you don't follow this rule, it will not bother your digital circuitry very much, but your beautiful, elegant, analog designs may not appreciate the interference. I left out power and ground planes because I know I don't even have to mention this requirement, right? If you locate your clocked filter, switching power supply, or any other source of high frequency switching transients near the ISP lines without adequate decoupling you may have trouble. We will talk more about decoupling in the next section. When operating at transmission line frequencies, one must take special care when routing signals on the PCB. The high frequencies that are generated by high slew rate rise times love to launch themselves in space when they encounter right angle turns in your PCB traces. The high impedance inputs on your CMOS devices will be only too happy to include this source of noise as part of the original signal. Make smooth, gradual turns with your traces and you will prosper.

POWER SUPPLY DECOUPLING

Every active device on your PCB should be decoupled from the power supply with at least one capacitor placed across the power and ground pins and located as close as possible to the device. If the device has more than one power/ground pair, then a capacitor(s) will be needed for however many it has. The PZx128 has eight power and ground pairs therefore you should use a minimum of eight capacitors to decouple this device. A tantalum or aluminum electrolytic capacitor, 10F or larger, should also be provided for every 50 or so active devices on the PCB. These capacitors will reduce power supply ripple and, for best results, they should be located near the point at which the power and ground rails enter the PCB. You may be wondering what value of capacitor to use at each active component. The answer depends upon what you are trying to accomplish. Capacitors located across component power and ground pins, depending upon their value, can perform high frequency decoupling or provide a charge reservoir for rapid changes in device current. To calculate a value for a reservoir capacitor, it may be helpful to use the equation:

$$C = i\frac{dt}{dv}$$

Where:

- C= the capacitor value you are valiantly trying to calculate
- i= the total current your device will consume worst case
- dt = lets call this the output slew rate, or rise time
- dv = a reasonable value of allowable voltage droop

The PZx128 has a maximum of 96 I/O lines. If we require every output to turn on at the same time and we set the characteristic impedance of each output trace to 50, the current required will be 96 x (5V/50) = 9.6 amperes! This can safely be considered WORST CASE! With a 100 mV drop in VCC and a 2 nS rise time, the capacitance required will be:

$$\frac{9.6\,\text{A} \times 2\,\text{nS}}{100\ \text{mV}} = 0.192\ \text{mF}$$

A 0.1F capacitor for every power/ground pair will therefore handle the maximum instantaneous current requirements quite easily.

FILTERING THE POWER SUPPLY

At high frequencies the 0.1F capacitor will look like a hippo as far as filtering is concerned. This is because its resonant frequency, (about 100 MHz) is too low to adequately reduce the high frequency harmonics generated in high performance systems. Without going into an in–depth discourse on Fourier transforms of periodic pulses, let's assume that the fundamental frequency relates to period, the first harmonic is a function of pulse width with the same energy as the fundamental, and that the second harmonic is a function of rise time and contains approximately half the energy of the fundamental. The following generic equation can be used to calculate fundamental and harmonic frequencies.

$$F = \frac{1}{p \times T_X}$$

Substitute:

- The period of the clock signal for TX when calculating the fundamental frequency
- The pulse width of the clock signal for TX when calculating the 1st harmonic
- The rise time of the clock signal for TX when calculating the 2nd harmonic

For a system operating at 100MHz with a pulse width of 5nS, the F0 is 32MHz, F1 is 64MHz, and F2 is 159MHz.

Using the formula for resonance,

$$f = \frac{1}{2p\sqrt{LC}}$$

and plugging in the lead inductance of 1.1nH for a 0.001F ceramic capacitor, as determined from it's data sheet, we find that it self resonates at approximately 150 MHz. Attaching this capacitor to a PCB in a most careful fashion, will provide adequate high frequency filtering of your design. Place this capacitor in parallel with the 0.1F reservoir capacitor, as close as possible to the active device, with as little total lead length as possible. Those who still experience trouble may want to use chip capacitors instead of leaded ones. This greatly reduces the series L and therefore increases the resonant frequency, giving better coverage of the higher frequency harmonics.

COMPONENT PLACEMENT

The placement of components on a PCB can have enormous impact on the noise environment. You may be less than pleased if you have arranged your PZx128 and the ISP download header in such a fashion as described in Figure 3. Remember to place the download header as close as possible to the device being programmed. If you are programming our ISP device from an on–board microcontroller, or an edge connector, it is important to follow the rules set forth in the section entitled **On Board Sources** of noise in this app note.

SUMMARY

High performance systems require that close attention by paid to printed circuit board layout, noise sources, and noise coupling. By following a few simple rules it is possible to construct a PCB that will allow you to effectively utilize our ISP devices.

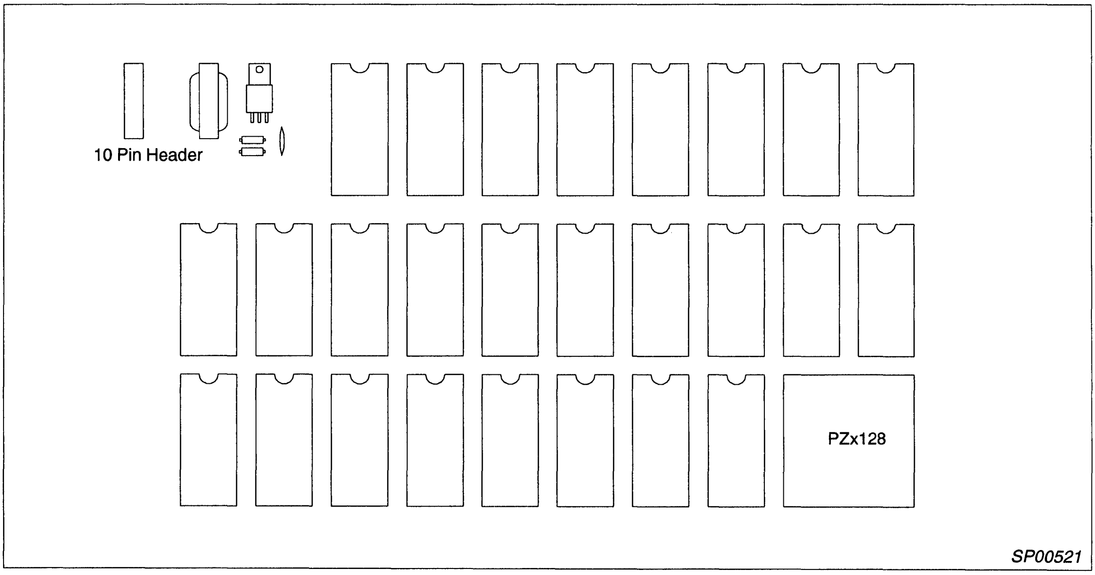


Figure 3. 'Bad' PCB Layout

Verilog implementation of a Manchester Encoder/Decoder in Philips CPLDs

AN070

INTRODUCTION

In this application note, Manchester code is defined, and the advantages relative to Non–Return to Zero code are given. Target applications of Manchester code are discussed. A verilog implementation of the Manchester Encoder–Decoder is given, along with schematics and Philips CPLD utilization results. The decoder and encoder are simulated using Verilog XL. Much of this is given in the context of more familiar serial communication circuits as UARTs, with the intent to illustrate some of the issues in designing serial communication functions in CPLDs.

NRZ and Manchester code defined

Non–return to Zero (NRZ) and Manchester codes are used in digital systems to represent the binary values 1 and 0. Figure 1 defines how NRZ and Manchester code represent binary values. NRZ is the code most often used. In NRZ, a logic 1 is represented as a high level throughout a data cell, and a logic 0 is represented by a low level. Manchester code represents binary values by a transition rather than a level. The transition occurs at mid–bit, with a low to high transition used to represent a logic 0, and a high to low to represent a logic–1. Depending on the data pattern, there may be a transition at the cell boundary (beginning/end). A pattern of consecutive 1s or 0s results in a transition on the cell boundary. When the data pattern alternates between 1 and 0, there is no transition on the cell boundary.

In NRZ, only one level/data cell is required, while in Manchester, two levels are required. A DC component exist in NRZ when contiguous 1s or contiguous 0s are transmitted. When the data pattern alternates between 1s and 0s, the frequency response is equal to 1/2 the clock rate.The frequency response for NRZ then ranges from DC to clock/2. The frequency response of Manchester code ranges from clock/2, occurring when the data pattern is alternating 1s and 0s, to clock, which occurs when the data pattern is consecutive 1s or 0s. The frequency response of Manchester is a single octave versus 5–10 octaves for NRZ.

Relative advantages on NRZ/Manchester code

Two advantages of NRZ are that it doesn't require encoding/decoding, and it makes the most efficient use of a communication channels bandwidth. Manchester requires a modulation rate twice that of NRZ to transmit the same amount of information. This can be important in bandwidth limited communication channels. On the other hand, the receiver of NRZ requires a true DC response. Since, manchester code has no DC component, it can be transformer coupled.

The mid–bit transition in Manchester code provides a self–clocking feature of code. This can be used to improve synchronization over non–self clocking codes as NRZ. The transition also allows additional error detection to be done with relatively little circuitry.

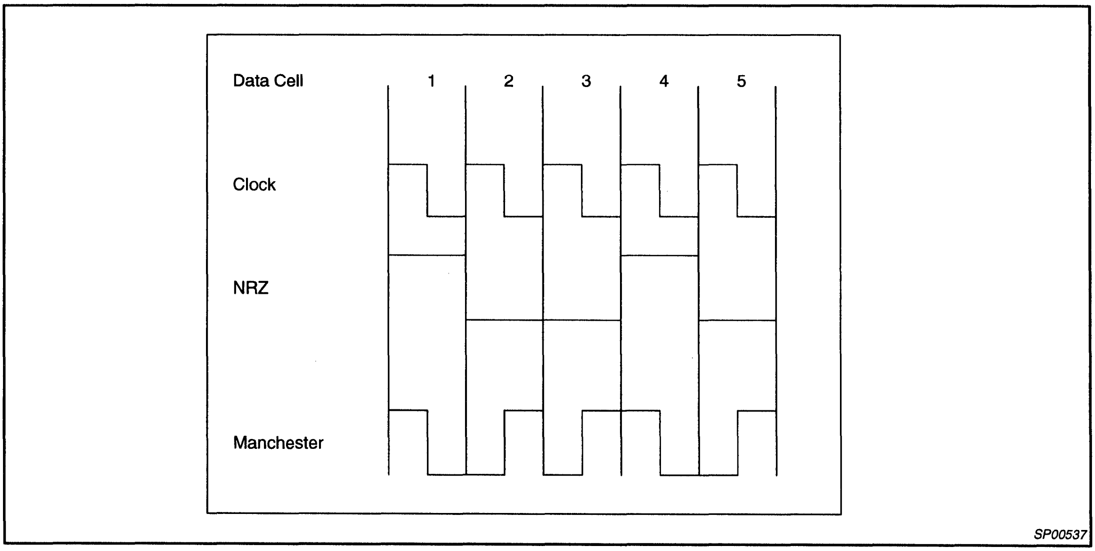


Figure 1. NRZ and Manchester Code Defined

Synchronization

In serial communication, clocks are used to define the size/boundary of a data cell. With a non–self clocking code, since the clock and data are distinct, there can be skew between clock and data. In magnetic media applications, skew may be due to variations in the tape drive speed. In serial communication, skew results from differences in the transit delay between clock and data lines on long serial links.

One design objective in serial communication is to decode data correctly in the presence of noise, or an otherwise degraded signal. Signals have non–zero rise and fall times during which the signal value is indeterminate. In a receiver, sampling, or the decoding of the value of a signal, should occur as far from the signal transition as possible. Sampling at the time furthest from the signal transition is known as center sampling.

A UART is a serial communication circuit which uses NRZ code. To sample at mid–bit of the data cell, the receiver in a UARTs uses a local clock which is 16X the received data rate. The data format of a UART consists of a high idle state, and a character format consisting of a start bit, 5–8 data bits, optional parity, and one or more stop bits. After detecting the edge of a start bit, the receiver validates the start bit by counting the 16X clock to 8 and verifying that it is still low. Subsequent center samples are reached by counting the 16X clock to 16.

In a Manchester decoder, center sampling occurs at points 1/4 and 3/4 through the cell, since transitions occur always at mid–bit and sometimes on the cell boundary. In addition to center sampling, the receiver in a Manchester decoder does clock recovery. Since Manchester has transitions at least once each data cell, the receiver has known references to which it can re–synchronize at each bit. To synchronize to an incoming serial data stream, the receiving ciruitry in a Manchester decoder can use a digital phase lock loop or a counter algorithm. Digital phase lock loops are most often used in networks with a ring topology while counter algorithms are common in point to point links. An example of a counter algorithm which uses a 16x clock is:

1. After receiving the initial transition on manchester data in, count the 16x clock to 4 and sample. The count of 4 is known as end count. At this time, end count is 1/4 through the the data cell.
2. Reset the counter to 0. Begin counting the 16x clock with an end count of 8, and sample. If there is a transition on manchester data, reset the counter and go to (1).

When initialized correctly to the manchester data, this algorithm causes the counter to use an end count equal to 4 when consecutive 1s or 0s are transmitted, and an end count equal to 8 when alternating 1s and 0s.

In summary, UARTs synchronize on a character basis while MEDs synchronize on a bit basis. In a UART, the timing jitter for the each bit in the character is cumulative until the end of the character. MEDs re–synchronize at each bit, or at least once each bit.

Error Detection

The most commonly used error detection schemes in serial communication are parity and cyclic redundancy check codes. When Manchester code is used, a small amount of additional circuitry can detect bit errors. Figure 2 illustrates how the mid bit transitions of Manchester code allow error detection. The figure shows four rows of transmitted data, ith the first row the valid Manchester representation for a logic–1. The three lower rows are waveforms of a corrupted form of the first row, and are erroneously transmitted data. When Manchester data is shifted in serially into a shift register in the decoder, an exclusive OR can monitor for different values on each side of the data cell (since there is a mid–bit transition) With this error detection, an error is undetected only if each half of a data cell transitions from its original state.

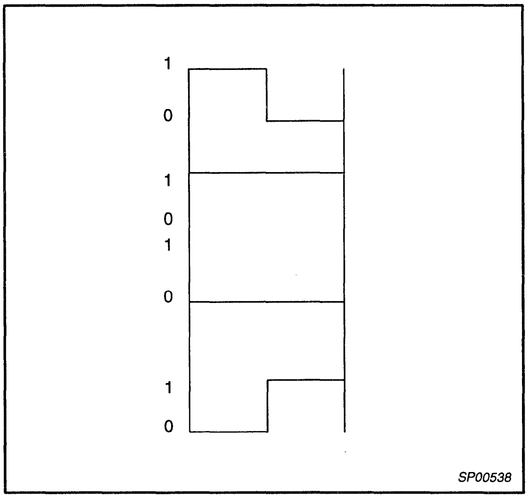


Figure 2. Manchester Error Detection

Manchester Encoder Decoder functionality

The functions of the encoder section of the MED are microprocessor interface, parallel to serial conversion, frame generation, and NRZ to Manchester encoding. This circuitry can run very fast since it does not require a high frequency clock. The frame format used is similar to that of a UART.

The Manchester decoder limits the maximum frequency of operation of the MED, since it uses a high frequency clock. The receiver circuitry is more complex, since clock recovery and center sampling is done. Additional receiver functions are frame detection, decoding of Manchester to NRZ, serial to parallel conversion, and a microprocessor interface.

Manchester Decoder

The mdi1 and mdi2 registers have decodeing circuitry which detects an incoming edge on mdi and activates the clk1x_enable. The clk1x_enable signal is used as a clock enable for various registers clocked by the 1x clock. This significantly reduces power consumption when data is not being received. The no_bits received keeps track of the number of bits received and sequences the decoder through its operations. To vary the word size, change the value of no_bits_rcvd. The decoder does clock recovery using the first variable, clkdiv register, and sample signal.The Manchester data is decoded at the sample signal.

Table 1. Manchester Decoder Pinout functionality

SIGNAL	DIRECTION	FUNCTION
rst	Input	Resets clk1x, no_bits_rcvd, nrz registers
clk16x	Input	16x clock input used as reference for clock recovery, center sampling
mdi	Input	Serial manchester data input
dout[7:0]	Output	Parallel NRZ data bus out
mdi1, mdi2	Internal	Internal registersused to detect edge on mdi
no_bits_rcvd	Internal	Controls word size and sequences decoder through operations
clk1x_enable	Internal	Enables 1x clock upon receipt of a word
clk1x	Internal	Internal 1x clock
sample	Internal	Determines time at which the receiver is to decode data
first	Internal	Variable used by the counter to determine end count
data_ready	Output	Status signal indicating data is present on the data bus dout.
rdn	Input	Control signal initiates a read operation

The test fixture for the Manchester decoder is given in Appendix A.
The text based simulation results is given in Appendix B.

The waveform output is :

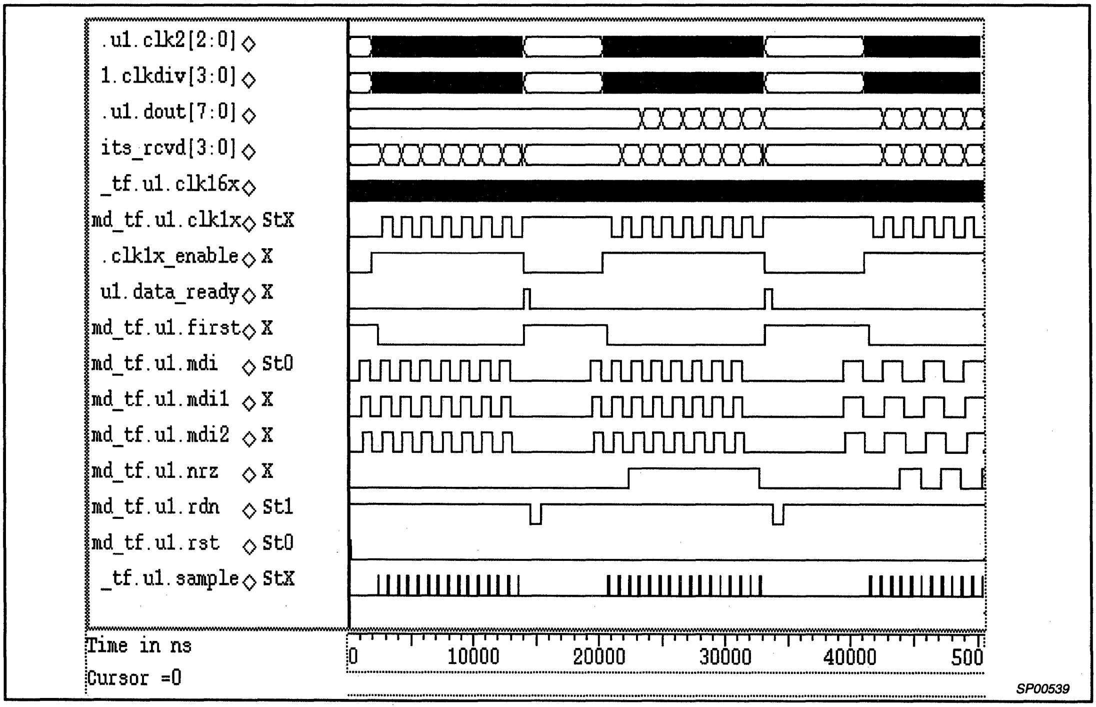


Figure 3. Manchester Simulation Waveform

Athough the resolution does not allow complete verification, the following is evident.

1. The clk1x_enable signal is high upon receipt of data on mdi, and low during the idle state of the line.
2. NRZ ia all 0s for first byte, all 1s second byte, alternating data third byte (not visable)
3. A read operation on rdn resets the data_ready status signal.

The waveform below shows the center sampling done by the manchester docoder shows the sample pulse occuring midway between the transitions on the mdi2 signal.

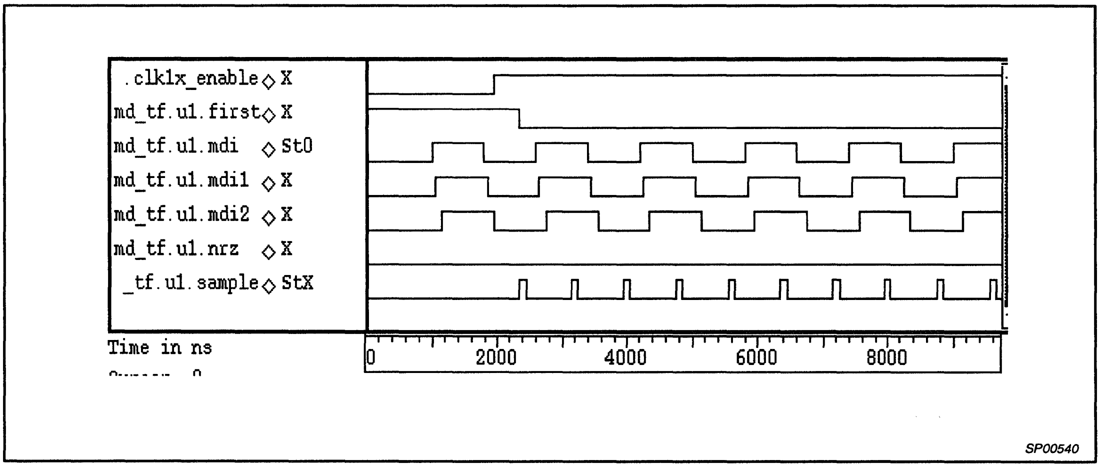


Figure 4. Waveform Showing Timing of Center Sampling in the Manchester Decoder

The verilog source is

```
module md (rst,clk16x,mdi,rdn,dout,data_ready) ;

input rst ;
input clk16x ;
input mdi ;
input rdn ;
output [7:0] dout ;
output data_ready ;

reg clk1x_enable ;
reg mdi1 ;
reg mdi2 ;
reg [7:0] dout ;
reg [3:0] no_bits_rcvd ;
reg [3:0] clkdiv ;
reg [2:0] clk2 ;
reg first ;
reg data_ready ;

wire clk1x ;
reg nrz ;
wire sample ;

always @(posedge clk16x or posedge rst)
begin
if (rst)
begin
mdi1 <= 1'b0 ;
mdi2 <= 1'b0 ;
end
else
```

```
begin
mdi2 <= mdi1 ;
mdi1 <= mdi ;
end
end

// enable the 1x clock if there is an edge on mdi

always @(posedge clk16x or posedge rst)
begin
if (rst)
clk1x_enable <= 1'b0 ;
else if (!mdi1 && mdi2)
clk1x_enable <= 1'b1;
else if (!mdi1 && !mdi2 && no_bits_rcvd == 4'b1000)
clk1x_enable <= 1'b0 ;
end

// generate center sample at 1/4 and 3/4 through the data cell

always @(posedge clk16x or posedge rst or negedge clk1x_enable)
begin
if (rst)
begin
clk2 = 3'b000 ;
first = 1'b1 ;
end
else if (!clk1x_enable)
begin
clk2 = 3'b000 ;
first = 1'b1 ;
end
else if (first && (clk2 < 3'b011))
clk2 = clk2 + 1 ;
else  if (first  && (clk2 == 3'b011))
begin
clk2 <= 3'b000 ;
first <= 1'b0 ;
end

if (!first)
clk2 = clk2 + 1 ;

end

assign sample = first && clk2[2] && !clk2[1] && !clk2[0]
|| !first && !clk2[2] && !clk2[1] && !clk2[0] ;

// decode manchester into nrz

always @(posedge rst or posedge sample)
if (rst)
nrz = 1'b0 ;
else
if (no_bits_rcvd > 0)
nrz = mdi2 ^ clk1x ;

// generate 1x clock
```

```
always @(posedge clk16x or posedge rst)
begin
if (rst)
clkdiv = 4'b0 ;
else if (clk1x_enable)
clkdiv = clkdiv + 1 ;
end

assign clk1x = clkdiv[3] ;

// serial to parallel conversion

always @(posedge clk1x or posedge rst)
if (rst) begin
dout = 8'h0 ;
end

else begin
dout[7:1] <= dout[6:0] ;
dout[0] <= nrz ;
end

// track no of bits rcvd for word size

always @(posedge clk1x or posedge rst or negedge clk1x_enable)
begin
if (rst)
no_bits_rcvd = 4'b0000 ;
else if (!clk1x_enable)
begin
no_bits_rcvd = 4'b0000 ;
end
else no_bits_rcvd = no_bits_rcvd + 1 ;

end

// generate data_ready status signal

always @(negedge clk1x_enable or posedge rst or negedge rdn)
if (rst || !rdn)
data_ready = 1'b0 ;
else
if (!clk1x_enable)
data_ready = 1'b1 ;
endmodule
```

The Manchester decoder schematics are given in Figure 5. A legible copy is given in the seven figures in Appendix C.

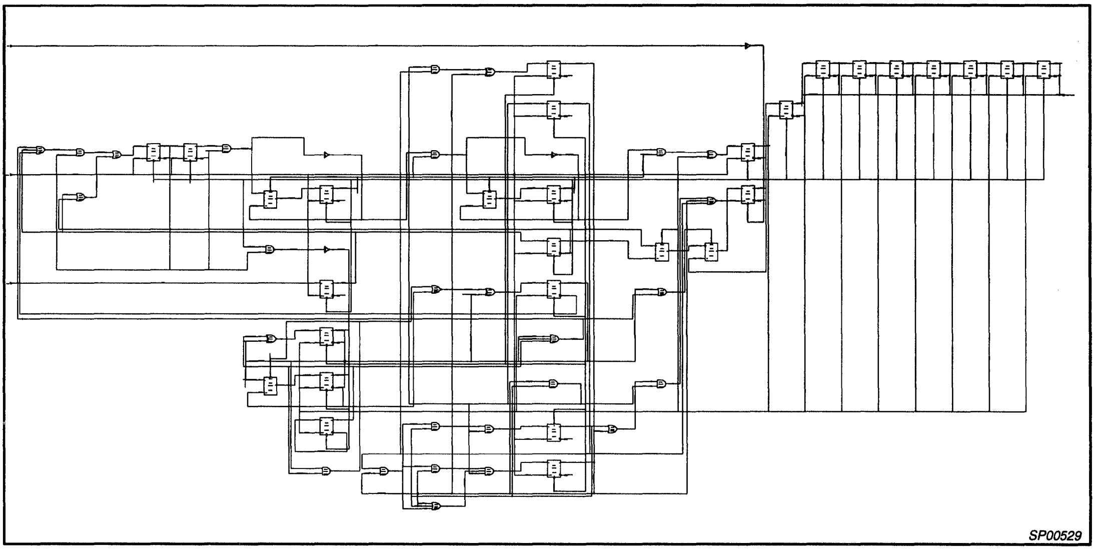


Figure 5. Manchester Decoder Schematic

Table 2. Manchester Encoder Pinout Funcionality Manchester Encoder

PIN/SIGNAL	DIRECTION	FUNCTION
din[7:0]	Input	Input data bus
clk	Input	Clock
wr	Input	Control signal input used to strobe data into the buffer register through din[7:0]
mdo	Output	Serial Manchester data out
rst	Input	Resets mdi1, mdi2, data[7:0], clk1, ready registers
ready	Output	Status signal indicating the encoder can accept data.

The verilog test fixture for the Manchester encoder is given in Appendix D. It writes the results into the md.rpt file given later, and the waveform database file me.shm displayed in Figure 4.

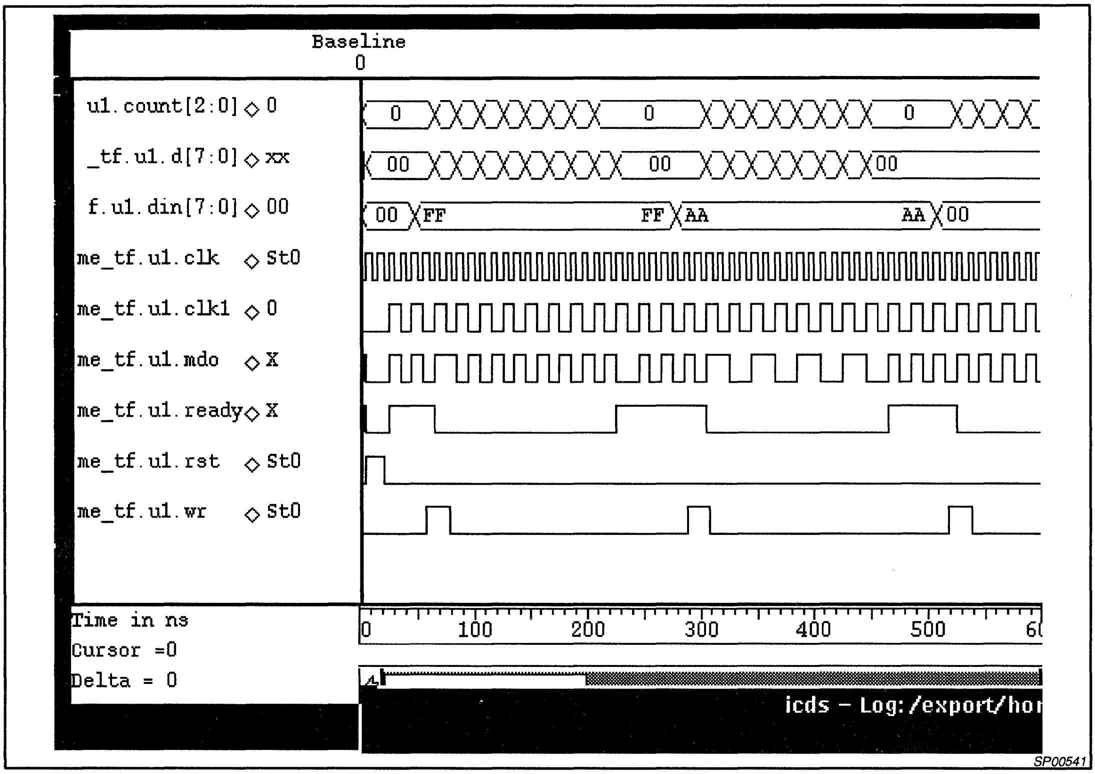


Figure 6. Waveform Results of Manchester Encoder Simulation

In the waveform shown in Figure 6, following can be verified

1. The pattern on manchester data out (mdo) when input data is consecutive 1s (ff), and again when input data is alternating 1s and 0s (aa).
2. The ready status signal goes high after word is transmitted, and is reset by a strobe on wr,

The manchester encoder source is

```
module me (rst,clk,wr,din,ready,mdo);
input rst ;
input clk ;
input wr ;
input [7:0] din;
output ready ;
output mdo ;

reg mdo ;
reg ready ;
reg [2:0] count;
reg  [7:0] d ;
reg clk1;
always @(posedge clk or posedge rst)
if (rst) begin
```

```
clk1 = 1'b0 ;
mdo = 1'b0 ;
end
else begin
clk1 = ~clk1 ;
mdo = d[7] ^ clk1 ;
end
always @(posedge clk1 or posedge rst)
if (rst) begin
count = 3'b0 ;
d = 8'h0 ;
ready = 1'b0 ;
end
else begin
if ((count == 0) & !wr) begin
d[7:1] <= d[6:0];
d[0] <= 1'b0;
ready <= 1'b1;
end else if ((count == 0) & wr ) begin
d  <= din;
count <= count + 1;
ready <= 1'b0;
end else if (count == 7) begin
d[7:1] <= d[6:0];
d[0] <= 1'b0;
count <= 0;
      end else begin
d[7:1] <= d[6:0];
d[0] <= 1'b0;
count <= count + 1;
end
end
endmodule
```

The schematics of the Manchester encoder is given in Figure 7, which consists of four sections.

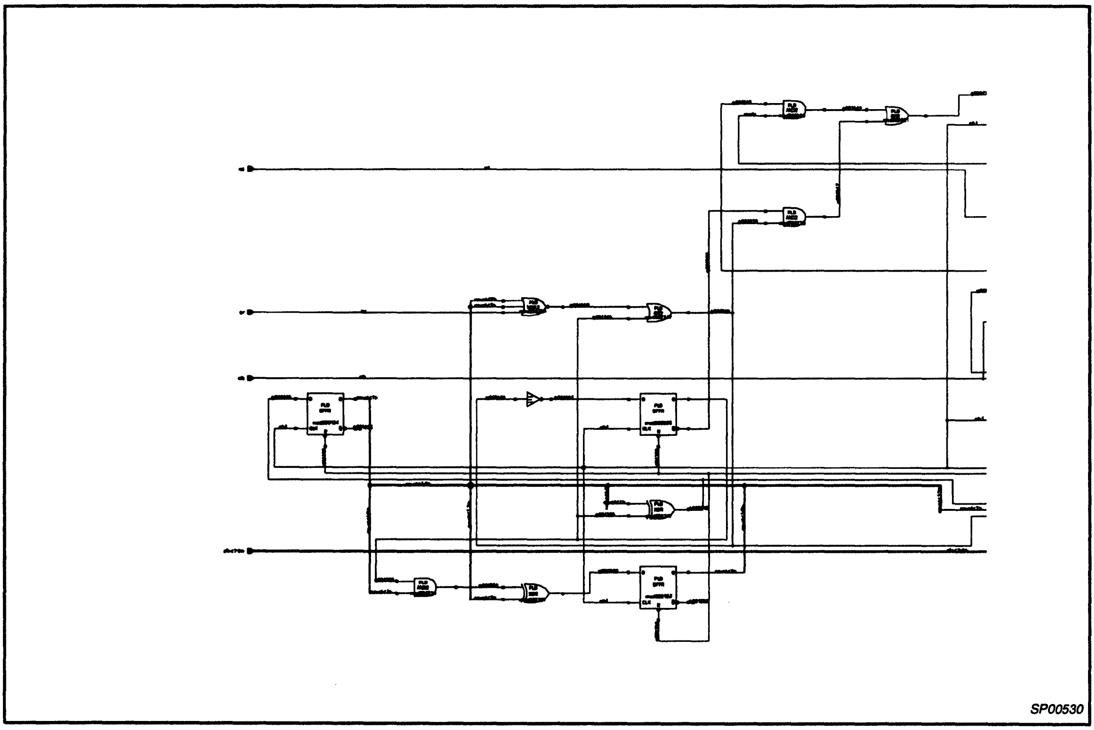


Figure 7. Manchester Encoder

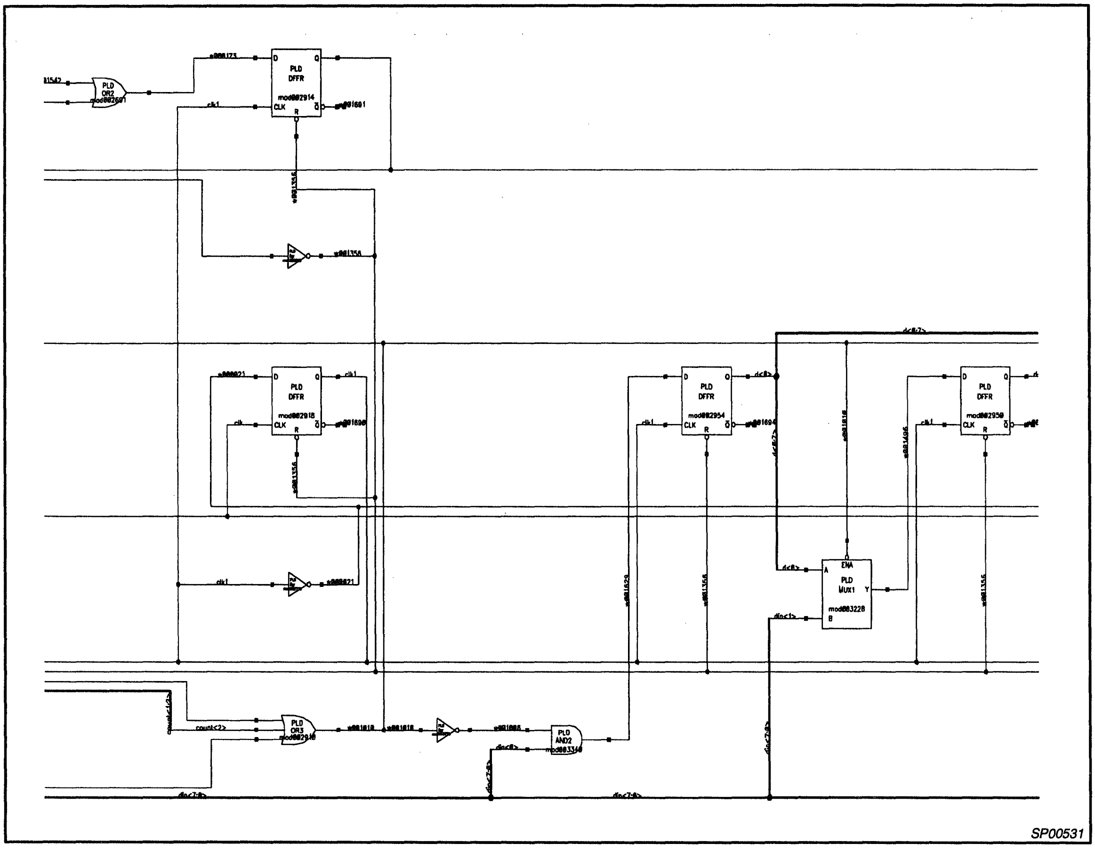
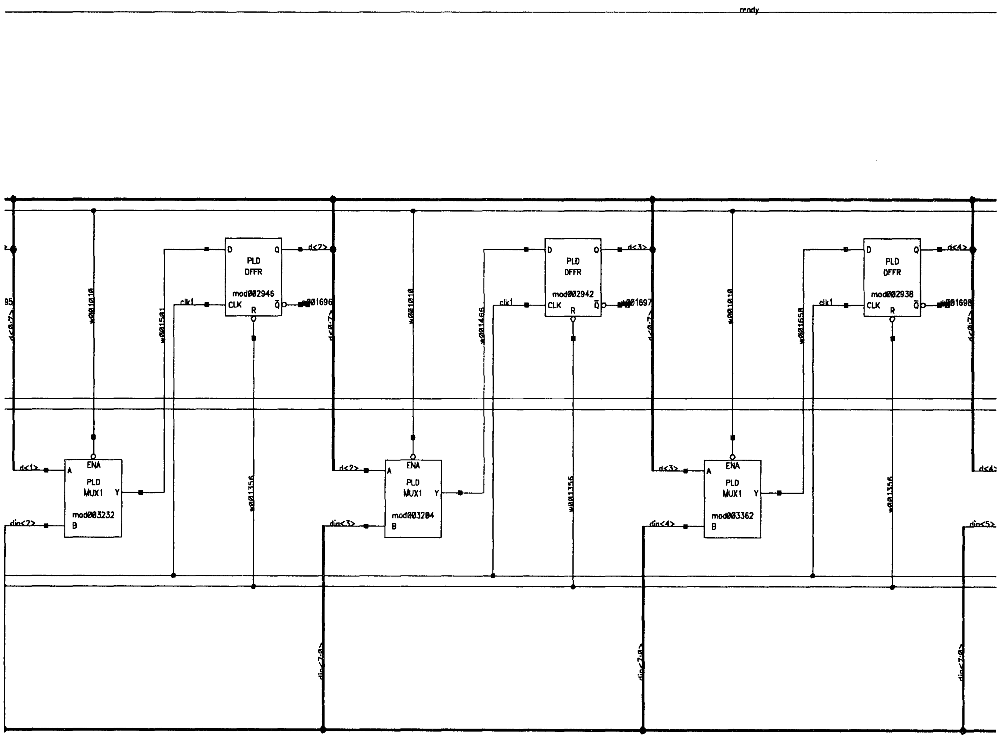


Figure 7A. Manchester Encoder Schematic (Cont.)



SP00532

Figure 7B. Manchester Encoder Schematic (Cont.)

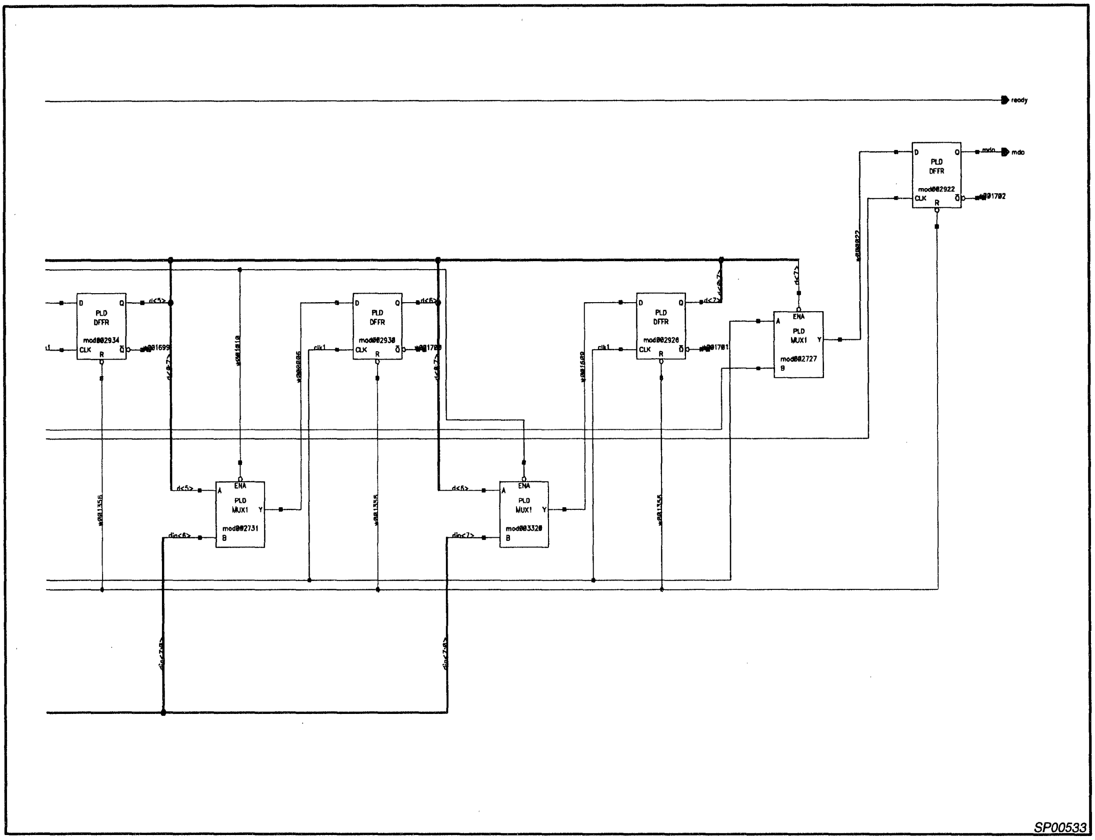


Figure 7C. Manchester Encoder Schematic (Cont.)

Other Manchester Encoder Decoder Functions

The functiions discussed in the remainder of this application note can be used in addition of in place of those described earlier. They include

- Sync pulse and manchester error detection
- Adding a buffer register to the decoder

Sync pulse and Manchester error detection

The design above uses a frame format similar to that of a UART. Additional noise immunity can be provided by substituting a 3–bit wide sync pulse for the 1–bit start bit. The sync pulse used in this section is 1 1/2 bits at one level followed by 1 1/2 bits at the opposite level. This pattern is given in the mdi strip in Figure 8. The reason for allowing either polarity (high/low or low/high) is that it provides a very efficient method for the transmitter and receiver to distinguish between a command and data. Since the 3–bit wide pattern is a relatively long pattern without a valid Manchester, and one which is unlikely to occur randomly, the decoder is unlikely to start erroneously.

Th detect the pattern, a 10–bit register md is used. The manchester data in (mdi) input is routed in serially into the 10–bit md register, which is clocked by a 2X clock clk2x. There is pattern detection circuitry monitoring md[9:0]. Since md is clocked by clk2x, the detection circuitry is actually continuously monitoring 5–bit data patterns. In the scheme given below, the pattern detection asserts the sync_ pulse signal if the 3–bit wide sync pulse is followed by two valid manchester bits. The pulse on sync_pulse can trigger the decode operation.

The waveform results are given in Figure 8.

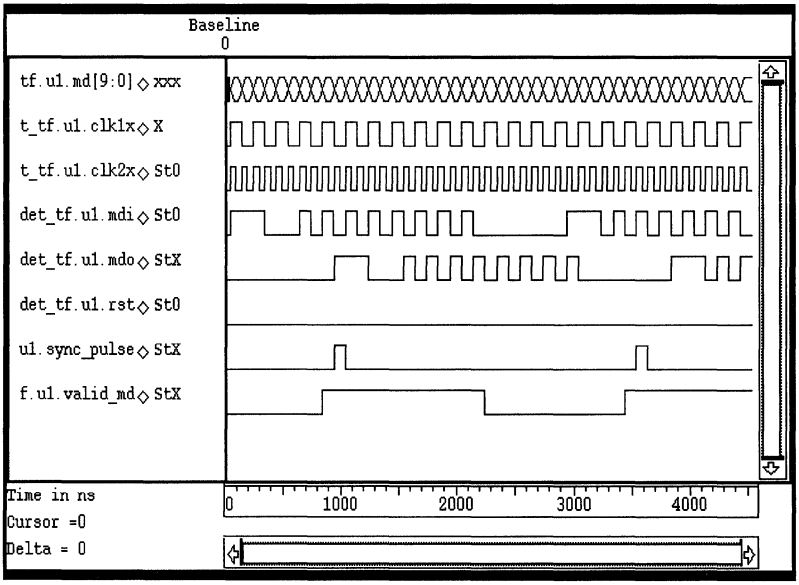


Figure 8. Sync and Manchester DetectorWaveform Results

The points to verify in the figure are that sync pulse goes high after reception of a valid 5–bit pattern, that mdo is the same as mdi 10 clk2x cycles later, and that valid_md is high when valid manchester data is received.

The verilog source is

```
// Philips CPLD Applications
// Sync pulse and valid manchester detector
// Otober 3, 1996
module sync_det (rst,clk2x,mdi,mdo,valid_md,sync_pulse) ;
input rst ;
input clk2x ;
input mdi ;
output mdo ;
output valid_md ;
output sync_pulse ;

reg [9:0] md ;
reg clk1x ;

always @(posedge clk2x  or posedge rst)
if (rst)
begin
clk1x <= 1'b0 ;
md <= 10'b0 ;
end
else
begin
md[9:1] <= md[8:0] ;
md[0] <= mdi ;
clk1x <= ~clk1x ;
end
```

```
assign valid_md = ((md[3] ^ md[2]) && (md[1] ^ md[0])) ;
assign sync_pulse = valid_md && (((md[9] && md[8] && md[7]) && (!md[6] && !md[5] && !md[4])) || ((!md[9] && !md[8] && !md[7]) && (md[6] &&
md[5] && md[4]))) ;

assign mdo = md[9] ;
endmodule
```

The schematics are given in Figure 9.

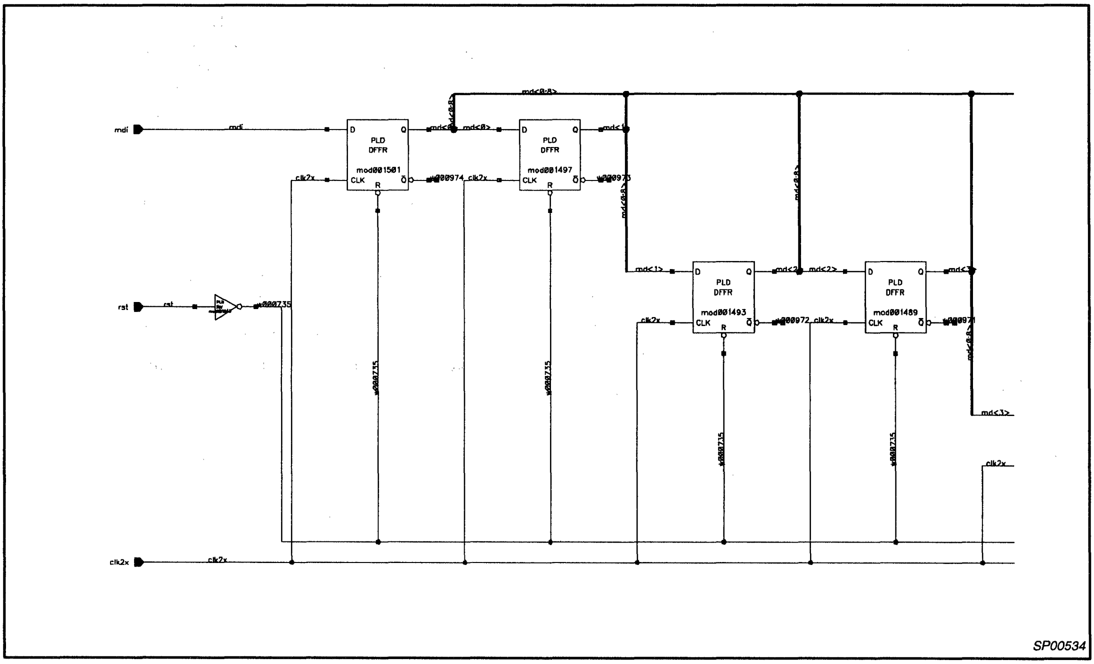
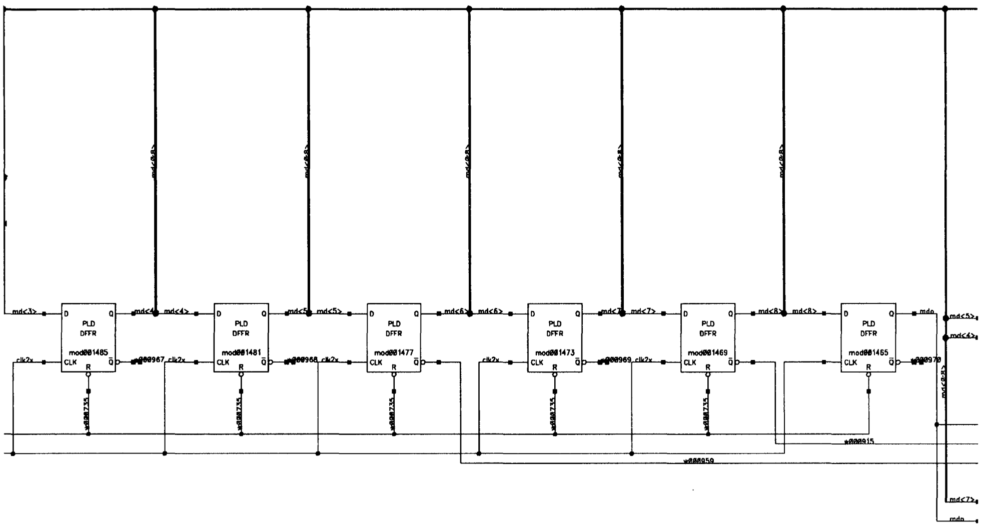


Figure 9. LHS of Sync and Valid Manchester Schematic



SP00535

Figure 9A. LHS of Sync and Valid Manchester Schematic (Cont.)

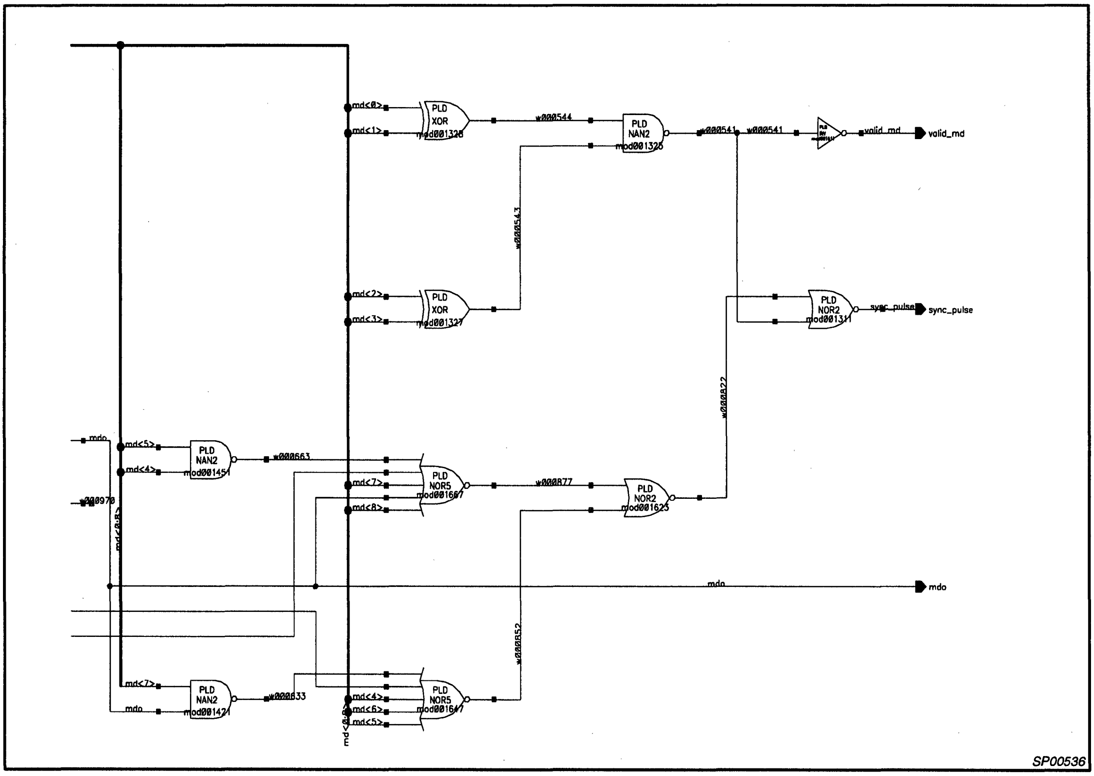


Figure 9B. LHS of Sync and Valid Manchester Schematic (Cont.)

APPENDIX A: MANCHESTER DECODER TEST FIXTURE

```
‘timescale 1 ns /1 ns
module md_tf ;
reg rst ;
reg clk16x ;
reg mdi ;
reg clk1x_enable ;
reg clk1x ;
reg nrz ;
reg [3:0] no_bits_rcvd ;
reg sample ;
reg rdn ;
wire [7:0] dout ;
md u1 (rst,clk16x,mdi,rdn,dout,data_ready) ;
initial begin
rst = 1’b0 ;
clk16x = 1’b0 ;
mdi = 1’b0 ;
rdn = 1’b1 ;
end
integer md_chann ;
initial begin
md_chann = $fopen(“md.rpt”) ;
$timeformat (–9,,,5) ;
end
parameter clock_period = 100;
always #(clock_period/2) clk16x = ~clk16x ;
initial begin
$fdisplay(md_chann, “Verilog simulation of manchester decoder design\n\n”) ;
$shm_open(“md.shm”) ;
$shm_probe(“AS”) ;

$fmonitor(md_chann,”Time=%t,rst=%b,clk16=%b,clk1x_enable=%b,clk1x=%b,mdi=%b,nrz=%b,no_bits_rcvd=%b,sample=%b,dout=%h,data_re
ady=%b”,$time,rst,clk16x,md.clk1x_enable,md.clk1x,mdi,md.nrz,md.no_bits_rcvd,md.sample,dout,data_ready);

#1 rst = 1’b1 ;
#200 rst = 1’b0 ;

// input 8 0s

#800 mdi = 1’b1 ;
#800 mdi = 1’b0 ;
#800 mdi = 1’b1 ;
#800 mdi = 1’b0 ;
#800 mdi = 1’b1 ;
#800 mdi = 1’b0 ;
#800 mdi = 1’b1 ;
#800 mdi = 1’b0 ;
#800 mdi = 1’b1 ;
#800 mdi = 1’b0 ;
#800 mdi = 1’b1 ;
#800 mdi = 1’b0 ;
#800 mdi = 1’b1 ;
#800 mdi = 1’b0 ;
#800 mdi = 1’b1 ;
#800 mdi = 1’b0 ;
```

```
#1600 rdn = 1’b0 ;
#800 rdn = 1’b1 ;

#3200

// input 8 1s

#800 mdi = 1’b1 ;
#800 mdi = 1’b0 ;
#800 mdi = 1’b1 ;
#800 mdi = 1’b0 ;
#800 mdi = 1’b1 ;
#800 mdi = 1’b0 ;
#800 mdi = 1’b1 ;
#800 mdi = 1’b0 ;
#800 mdi = 1’b1 ;
#800 mdi = 1’b0 ;
#800 mdi = 1’b1 ;
#800 mdi = 1’b0 ;
#800 mdi = 1’b1 ;
#800 mdi = 1’b0 ;
#800 mdi = 1’b1 ;
#800 mdi = 1’b0 ;

#2400 rdn = 1’b0 ;
#800 rdn = 1’b1 ;

#3200
// input alternating 1s, 0s
#1600 mdi = 1’b1 ;
#1600 mdi = 1’b0 ;
#1600 mdi = 1’b1 ;
#1600 mdi = 1’b0 ;
#1600 mdi = 1’b1 ;
#1600 mdi = 1’b0 ;
#1600 mdi = 1’b1 ;
#1600 mdi = 1’b0 ;
#1600 mdi = 1’b1 ;

#1600 rst = 1’b1 ;

$fdisplay (md_chann,"\nSimulation of manchester decoder complete.");

$finish ;
end
endmodule
```

APPENDIX B: MANCHESTER DECODER SIMULATION RESULTS (PARTIAL)

Verilog simulation of manchester decoder design

```
Time=   0,rst=0,clk16=0,clk1x_enable=x,clk1x=x,mdi=0,nrz=x,no_bits_rcvd=xxxx,sample=x,dout=xx,data_ready=x
Time=   1,rst=1,clk16=0,clk1x_enable=0,clk1x=0,mdi=0,nrz=0,no_bits_rcvd=0000,sample=0,dout=00,data_ready=0
Time=  50,rst=1,clk16=1,clk1x_enable=0,clk1x=0,mdi=0,nrz=0,no_bits_rcvd=0000,sample=0,dout=00,data_ready=0
Time= 100,rst=1,clk16=0,clk1x_enable=0,clk1x=0,mdi=0,nrz=0,no_bits_rcvd=0000,sample=0,dout=00,data_ready=0
Time= 150,rst=1,clk16=1,clk1x_enable=0,clk1x=0,mdi=0,nrz=0,no_bits_rcvd=0000,sample=0,dout=00,data_ready=0
Time= 200,rst=1,clk16=0,clk1x_enable=0,clk1x=0,mdi=0,nrz=0,no_bits_rcvd=0000,sample=0,dout=00,data_ready=0
Time= 201,rst=0,clk16=0,clk1x_enable=0,clk1x=0,mdi=0,nrz=0,no_bits_rcvd=0000,sample=0,dout=00,data_ready=0
Time= 250,rst=0,clk16=1,clk1x_enable=0,clk1x=0,mdi=0,nrz=0,no_bits_rcvd=0000,sample=0,dout=00,data_ready=0
Time= 300,rst=0,clk16=0,clk1x_enable=0,clk1x=0,mdi=0,nrz=0,no_bits_rcvd=0000,sample=0,dout=00,data_ready=0
Time= 350,rst=0,clk16=1,clk1x_enable=0,clk1x=0,mdi=0,nrz=0,no_bits_rcvd=0000,sample=0,dout=00,data_ready=0
Time= 400,rst=0,clk16=0,clk1x_enable=0,clk1x=0,mdi=0,nrz=0,no_bits_rcvd=0000,sample=0,dout=00,data_ready=0
Time= 450,rst=0,clk16=1,clk1x_enable=0,clk1x=0,mdi=0,nrz=0,no_bits_rcvd=0000,sample=0,dout=00,data_ready=0
Time= 500,rst=0,clk16=0,clk1x_enable=0,clk1x=0,mdi=0,nrz=0,no_bits_rcvd=0000,sample=0,dout=00,data_ready=0
Time= 550,rst=0,clk16=1,clk1x_enable=0,clk1x=0,mdi=0,nrz=0,no_bits_rcvd=0000,sample=0,dout=00,data_ready=0
Time= 600,rst=0,clk16=0,clk1x_enable=0,clk1x=0,mdi=0,nrz=0,no_bits_rcvd=0000,sample=0,dout=00,data_ready=0
Time= 650,rst=0,clk16=1,clk1x_enable=0,clk1x=0,mdi=0,nrz=0,no_bits_rcvd=0000,sample=0,dout=00,data_ready=0
Time= 700,rst=0,clk16=0,clk1x_enable=0,clk1x=0,mdi=0,nrz=0,no_bits_rcvd=0000,sample=0,dout=00,data_ready=0
Time= 750,rst=0,clk16=1,clk1x_enable=0,clk1x=0,mdi=0,nrz=0,no_bits_rcvd=0000,sample=0,dout=00,data_ready=0
Time= 800,rst=0,clk16=0,clk1x_enable=0,clk1x=0,mdi=0,nrz=0,no_bits_rcvd=0000,sample=0,dout=00,data_ready=0
Time= 850,rst=0,clk16=1,clk1x_enable=0,clk1x=0,mdi=0,nrz=0,no_bits_rcvd=0000,sample=0,dout=00,data_ready=0
Time= 900,rst=0,clk16=0,clk1x_enable=0,clk1x=0,mdi=0,nrz=0,no_bits_rcvd=0000,sample=0,dout=00,data_ready=0
Time= 950,rst=0,clk16=1,clk1x_enable=0,clk1x=0,mdi=0,nrz=0,no_bits_rcvd=0000,sample=0,dout=00,data_ready=0
Time=1000,rst=0,clk16=0,clk1x_enable=0,clk1x=0,mdi=0,nrz=0,no_bits_rcvd=0000,sample=0,dout=00,data_ready=0
Time=1001,rst=0,clk16=0,clk1x_enable=0,clk1x=0,mdi=1,nrz=0,no_bits_rcvd=0000,sample=0,dout=00,data_ready=0
Time=1050,rst=0,clk16=1,clk1x_enable=0,clk1x=0,mdi=1,nrz=0,no_bits_rcvd=0000,sample=0,dout=00,data_ready=0
Time= 1100,rst=0,clk16=0,clk1x_enable=0,clk1x=0,mdi=1,nrz=0,no_bits_rcvd=0000,sample=0,dout=00,data_ready=0
Time= 1150,rst=0,clk16=1,clk1x_enable=0,clk1x=0,mdi=1,nrz=0,no_bits_rcvd=0000,sample=0,dout=00,data_ready=0
Time=1200,rst=0,clk16=0,clk1x_enable=0,clk1x=0,mdi=1,nrz=0,no_bits_rcvd=0000,sample=0,dout=00,data_ready=0
Time=1250,rst=0,clk16=1,clk1x_enable=0,clk1x=0,mdi=1,nrz=0,no_bits_rcvd=0000,sample=0,dout=00,data_ready=0
Time=1300,rst=0,clk16=0,clk1x_enable=0,clk1x=0,mdi=1,nrz=0,no_bits_rcvd=0000,sample=0,dout=00,data_ready=0
Time=1350,rst=0,clk16=1,clk1x_enable=0,clk1x=0,mdi=1,nrz=0,no_bits_rcvd=0000,sample=0,dout=00,data_ready=0
Time=1400,rst=0,clk16=0,clk1x_enable=0,clk1x=0,mdi=1,nrz=0,no_bits_rcvd=0000,sample=0,dout=00,data_ready=0
Time=1450,rst=0,clk16=1,clk1x_enable=0,clk1x=0,mdi=1,nrz=0,no_bits_rcvd=0000,sample=0,dout=00,data_ready=0
Time=1500,rst=0,clk16=0,clk1x_enable=0,clk1x=0,mdi=1,nrz=0,no_bits_rcvd=0000,sample=0,dout=00,data_ready=0
Time=1550,rst=0,clk16=1,clk1x_enable=0,clk1x=0,mdi=1,nrz=0,no_bits_rcvd=0000,sample=0,dout=00,data_ready=0
Time=1600,rst=0,clk16=0,clk1x_enable=0,clk1x=0,mdi=1,nrz=0,no_bits_rcvd=0000,sample=0,dout=00,data_ready=0
Time=1650,rst=0,clk16=1,clk1x_enable=0,clk1x=0,mdi=1,nrz=0,no_bits_rcvd=0000,sample=0,dout=00,data_ready=0
Time=1700,rst=0,clk16=0,clk1x_enable=0,clk1x=0,mdi=1,nrz=0,no_bits_rcvd=0000,sample=0,dout=00,data_ready=0
Time=1750,rst=0,clk16=1,clk1x_enable=0,clk1x=0,mdi=1,nrz=0,no_bits_rcvd=0000,sample=0,dout=00,data_ready=0
Time=1800,rst=0,clk16=0,clk1x_enable=0,clk1x=0,mdi=1,nrz=0,no_bits_rcvd=0000,sample=0,dout=00,data_ready=0
Time=1801,rst=0,clk16=0,clk1x_enable=0,clk1x=0,mdi=0,nrz=0,no_bits_rcvd=0000,sample=0,dout=00,data_ready=0
Time=1850,rst=0,clk16=1,clk1x_enable=0,clk1x=0,mdi=0,nrz=0,no_bits_rcvd=0000,sample=0,dout=00,data_ready=0
Time=1900,rst=0,clk16=0,clk1x_enable=0,clk1x=0,mdi=0,nrz=0,no_bits_rcvd=0000,sample=0,dout=00,data_ready=0
Time=1950,rst=0,clk16=1,clk1x_enable=1,clk1x=0,mdi=0,nrz=0,no_bits_rcvd=0000,sample=0,dout=00,data_ready=0
Time=2000,rst=0,clk16=0,clk1x_enable=1,clk1x=0,mdi=0,nrz=0,no_bits_rcvd=0000,sample=0,dout=00,data_ready=0
Time=2050,rst=0,clk16=1,clk1x_enable=1,clk1x=0,mdi=0,nrz=0,no_bits_rcvd=0000,sample=0,dout=00,data_ready=0
Time=2100,rst=0,clk16=0,clk1x_enable=1,clk1x=0,mdi=0,nrz=0,no_bits_rcvd=0000,sample=0,dout=00,data_ready=0
Time=2150,rst=0,clk16=1,clk1x_enable=1,clk1x=0,mdi=0,nrz=0,no_bits_rcvd=0000,sample=0,dout=00,data_ready=0
Time=2200,rst=0,clk16=0,clk1x_enable=1,clk1x=0,mdi=0,nrz=0,no_bits_rcvd=0000,sample=0,dout=00,data_ready=0
Time=2250,rst=0,clk16=1,clk1x_enable=1,clk1x=0,mdi=0,nrz=0,no_bits_rcvd=0000,sample=0,dout=00,data_ready=0
Time=2300,rst=0,clk16=0,clk1x_enable=1,clk1x=0,mdi=0,nrz=0,no_bits_rcvd=0000,sample=0,dout=00,data_ready=0
Time=2350,rst=0,clk16=1,clk1x_enable=1,clk1x=0,mdi=0,nrz=0,no_bits_rcvd=0000,sample=1,dout=00,data_ready=0
Time=2400,rst=0,clk16=0,clk1x_enable=1,clk1x=0,mdi=0,nrz=0,no_bits_rcvd=0000,sample=1,dout=00,data_ready=0
```

```
Time=2450,rst=0,clk16=1,clk1x_enable=1,clk1x=0,mdi=0,nrz=0,no_bits_rcvd=0000,sample=0,dout=00,data_ready=0
Time=2500,rst=0,clk16=0,clk1x_enable=1,clk1x=0,mdi=0,nrz=0,no_bits_rcvd=0000,sample=0,dout=00,data_ready=0
Time=2550,rst=0,clk16=1,clk1x_enable=1,clk1x=0,mdi=0,nrz=0,no_bits_rcvd=0000,sample=0,dout=00,data_ready=0
Time=2600,rst=0,clk16=0,clk1x_enable=1,clk1x=0,mdi=0,nrz=0,no_bits_rcvd=0000,sample=0,dout=00,data_ready=0
Time=2601,rst=0,clk16=0,clk1x_enable=1,clk1x=0,mdi=1,nrz=0,no_bits_rcvd=0000,sample=0,dout=00,data_ready=0
Time=2650,rst=0,clk16=1,clk1x_enable=1,clk1x=0,mdi=1,nrz=0,no_bits_rcvd=0000,sample=0,dout=00,data_ready=0
Time=2700,rst=0,clk16=0,clk1x_enable=1,clk1x=0,mdi=1,nrz=0,no_bits_rcvd=0000,sample=0,dout=00,data_ready=0
Time=2750,rst=0,clk16=1,clk1x_enable=1,clk1x=1,mdi=1,nrz=0,no_bits_rcvd=0001,sample=0,dout=00,data_ready=0
Time=2800,rst=0,clk16=0,clk1x_enable=1,clk1x=1,mdi=1,nrz=0,no_bits_rcvd=0001,sample=0,dout=00,data_ready=0
Time=2850,rst=0,clk16=1,clk1x_enable=1,clk1x=1,mdi=1,nrz=0,no_bits_rcvd=0001,sample=0,dout=00,data_ready=0
Time=2900,rst=0,clk16=0,clk1x_enable=1,clk1x=1,mdi=1,nrz=0,no_bits_rcvd=0001,sample=0,dout=00,data_ready=0
Time=2950,rst=0,clk16=1,clk1x_enable=1,clk1x=1,mdi=1,nrz=0,no_bits_rcvd=0001,sample=0,dout=00,data_ready=0
Time=3000,rst=0,clk16=0,clk1x_enable=1,clk1x=1,mdi=1,nrz=0,no_bits_rcvd=0001,sample=0,dout=00,data_ready=0
Time=3050,rst=0,clk16=1,clk1x_enable=1,clk1x=1,mdi=1,nrz=0,no_bits_rcvd=0001,sample=0,dout=00,data_ready=0
Time=3100,rst=0,clk16=0,clk1x_enable=1,clk1x=1,mdi=1,nrz=0,no_bits_rcvd=0001,sample=0,dout=00,data_ready=0
Time=3150,rst=0,clk16=1,clk1x_enable=1,clk1x=1,mdi=1,nrz=0,no_bits_rcvd=0001,sample=1,dout=00,data_ready=0
Time=3200,rst=0,clk16=0,clk1x_enable=1,clk1x=1,mdi=1,nrz=0,no_bits_rcvd=0001,sample=1,dout=00,data_ready=0
Time=3250,rst=0,clk16=1,clk1x_enable=1,clk1x=1,mdi=1,nrz=0,no_bits_rcvd=0001,sample=0,dout=00,data_ready=0
Time=3300,rst=0,clk16=0,clk1x_enable=1,clk1x=1,mdi=1,nrz=0,no_bits_rcvd=0001,sample=0,dout=00,data_ready=0
Time=3350,rst=0,clk16=1,clk1x_enable=1,clk1x=1,mdi=1,nrz=0,no_bits_rcvd=0001,sample=0,dout=00,data_ready=0
Time=3400,rst=0,clk16=0,clk1x_enable=1,clk1x=1,mdi=1,nrz=0,no_bits_rcvd=0001,sample=0,dout=00,data_ready=0
Time=3401,rst=0,clk16=0,clk1x_enable=1,clk1x=1,mdi=0,nrz=0,no_bits_rcvd=0001,sample=0,dout=00,data_ready=0
Time=3450,rst=0,clk16=1,clk1x_enable=1,clk1x=1,mdi=0,nrz=0,no_bits_rcvd=0001,sample=0,dout=00,data_ready=0
Time=3500,rst=0,clk16=0,clk1x_enable=1,clk1x=1,mdi=0,nrz=0,no_bits_rcvd=0001,sample=0,dout=00,data_ready=0
Time=3550,rst=0,clk16=1,clk1x_enable=1,clk1x=0,mdi=0,nrz=0,no_bits_rcvd=0001,sample=0,dout=00,data_ready=0
Time=3600,rst=0,clk16=0,clk1x_enable=1,clk1x=0,mdi=0,nrz=0,no_bits_rcvd=0001,sample=0,dout=00,data_ready=0
Time=52450,rst=0,clk16=1,clk1x_enable=1,clk1x=1,mdi=1,nrz=0,no_bits_rcvd=0111,sample=0,dout=2a,data_ready=0
Time=52500,rst=0,clk16=0,clk1x_enable=1,clk1x=1,mdi=1,nrz=0,no_bits_rcvd=0111,sample=0,dout=2a,data_ready=0
Time=52550,rst=0,clk16=1,clk1x_enable=1,clk1x=1,mdi=1,nrz=0,no_bits_rcvd=0111,sample=0,dout=2a,data_ready=0
Time=52600,rst=0,clk16=0,clk1x_enable=1,clk1x=1,mdi=1,nrz=0,no_bits_rcvd=0111,sample=0,dout=2a,data_ready=0
Time=52650,rst=0,clk16=1,clk1x_enable=1,clk1x=1,mdi=1,nrz=0,no_bits_rcvd=0111,sample=0,dout=2a,data_ready=0
Time=52700,rst=0,clk16=0,clk1x_enable=1,clk1x=1,mdi=1,nrz=0,no_bits_rcvd=0111,sample=0,dout=2a,data_ready=0
Time=52750,rst=0,clk16=1,clk1x_enable=1,clk1x=1,mdi=1,nrz=0,no_bits_rcvd=0111,sample=1,dout=2a,data_ready=0
Time=52800,rst=0,clk16=0,clk1x_enable=1,clk1x=1,mdi=1,nrz=0,no_bits_rcvd=0111,sample=1,dout=2a,data_ready=0
Time=52850,rst=0,clk16=1,clk1x_enable=1,clk1x=1,mdi=1,nrz=0,no_bits_rcvd=0111,sample=0,dout=2a,data_ready=0
Time=52900,rst=0,clk16=0,clk1x_enable=1,clk1x=1,mdi=1,nrz=0,no_bits_rcvd=0111,sample=0,dout=2a,data_ready=0
Time=52950,rst=0,clk16=1,clk1x_enable=1,clk1x=1,mdi=1,nrz=0,no_bits_rcvd=0111,sample=0,dout=2a,data_ready=0
Time=53000,rst=0,clk16=0,clk1x_enable=1,clk1x=1,mdi=1,nrz=0,no_bits_rcvd=0111,sample=0,dout=2a,data_ready=0
Time=53050,rst=0,clk16=1,clk1x_enable=1,clk1x=0,mdi=1,nrz=0,no_bits_rcvd=0111,sample=0,dout=2a,data_ready=0
Time=53100,rst=0,clk16=0,clk1x_enable=1,clk1x=0,mdi=1,nrz=0,no_bits_rcvd=0111,sample=0,dout=2a,data_ready=0
Time=53150,rst=0,clk16=1,clk1x_enable=1,clk1x=0,mdi=1,nrz=0,no_bits_rcvd=0111,sample=0,dout=2a,data_ready=0
Time=53200,rst=0,clk16=0,clk1x_enable=1,clk1x=0,mdi=1,nrz=0,no_bits_rcvd=0111,sample=0,dout=2a,data_ready=0
Time=53250,rst=0,clk16=1,clk1x_enable=1,clk1x=0,mdi=1,nrz=0,no_bits_rcvd=0111,sample=0,dout=2a,data_ready=0
Time=53300,rst=0,clk16=0,clk1x_enable=1,clk1x=0,mdi=1,nrz=0,no_bits_rcvd=0111,sample=0,dout=2a,data_ready=0
Time=53350,rst=0,clk16=1,clk1x_enable=1,clk1x=0,mdi=1,nrz=0,no_bits_rcvd=0111,sample=0,dout=2a,data_ready=0

Simulation of manchester decoder complete.
```

APPENDIX C: MANCHESTER DECODER SCHEMATICS

The seven schematics given below correspond to the main schematic given in Figure 4, ordered from left to right and top to bottom.

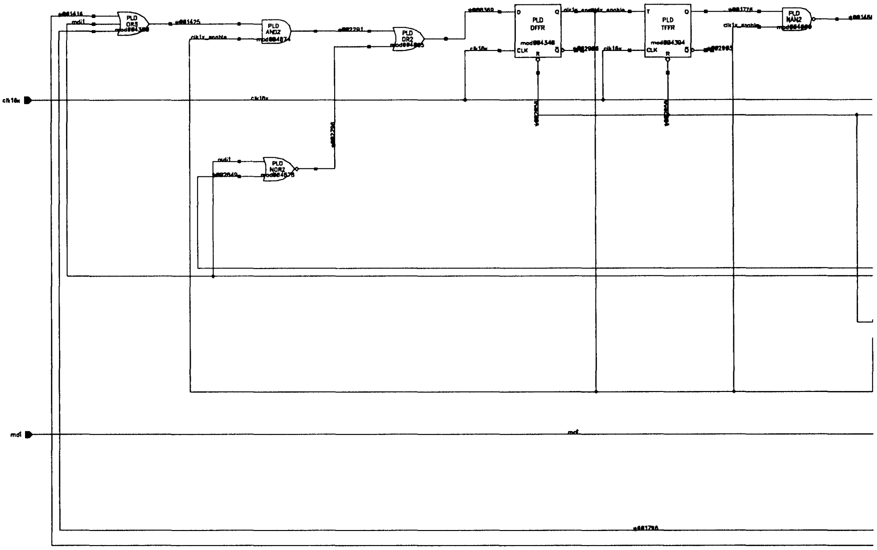


Figure 10. Manchester Decoder Schematic

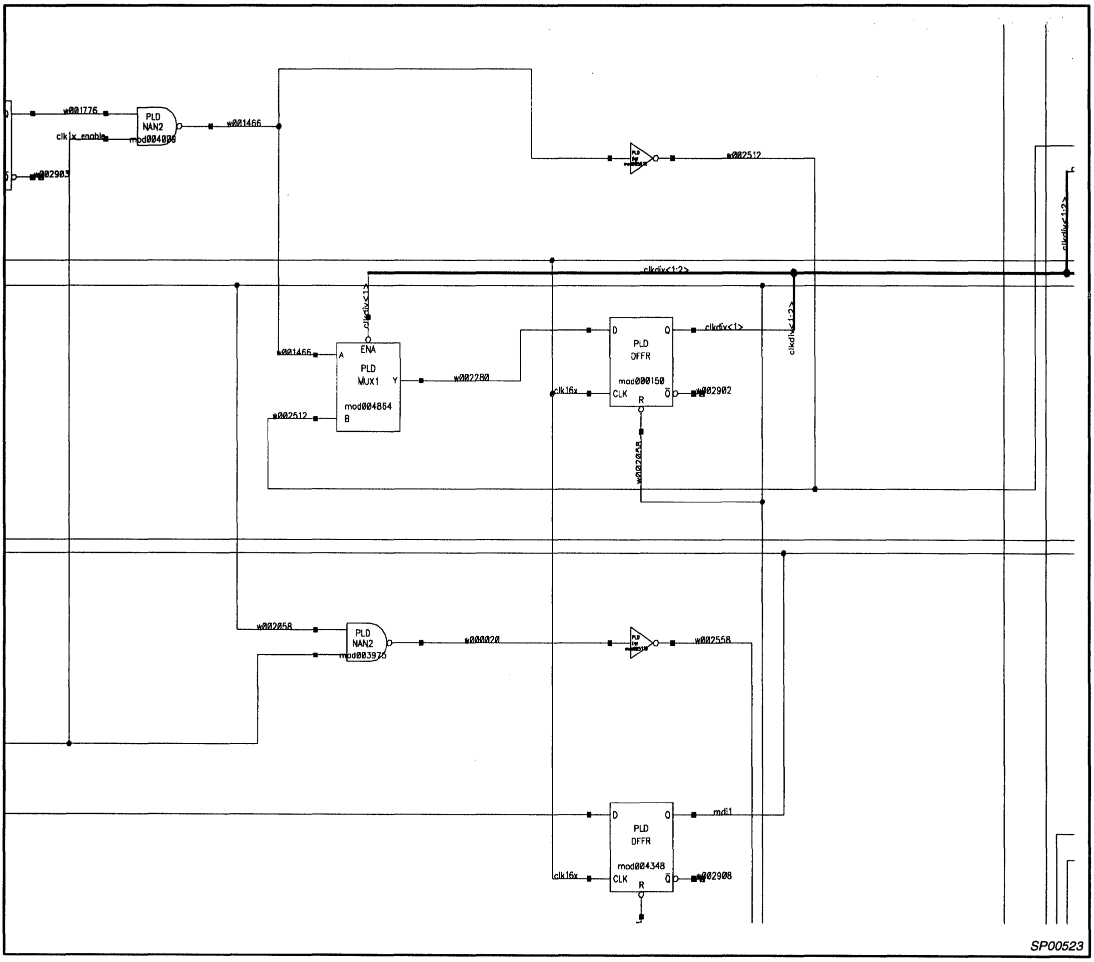


Figure 10A. Manchester Decoder Schematic (Cont.)

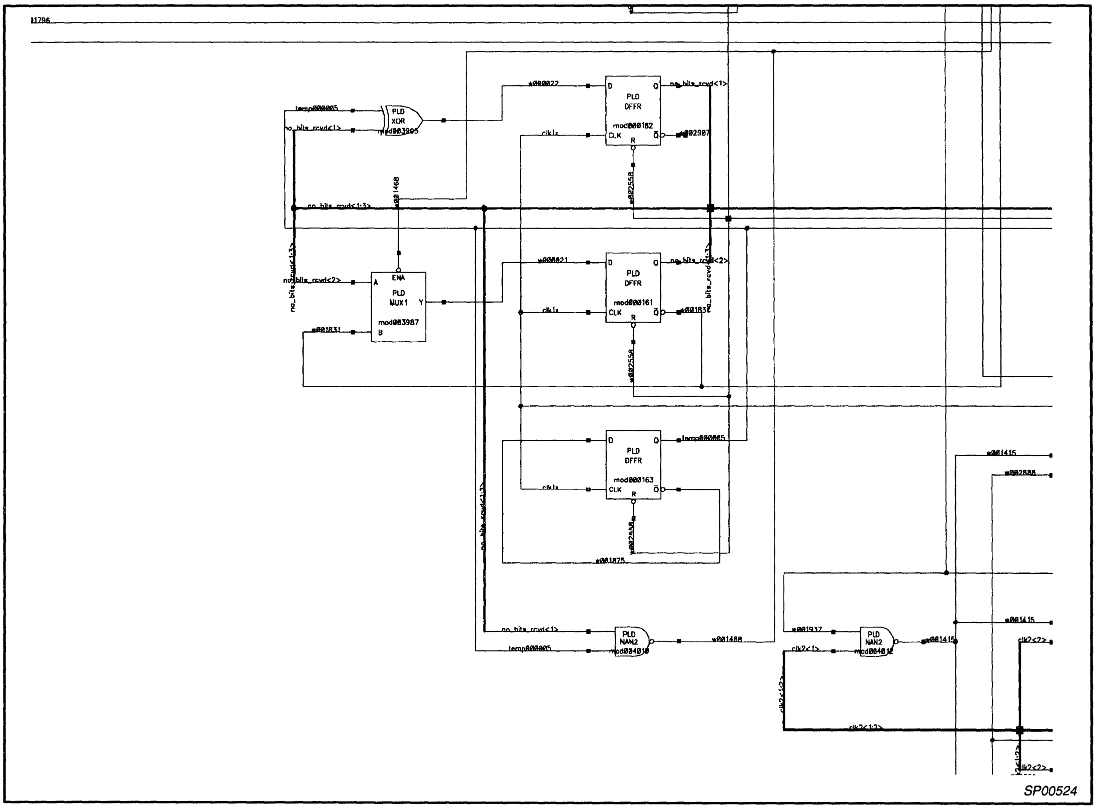


Figure 10B. Manchester Decoder Schematic (Cont.)

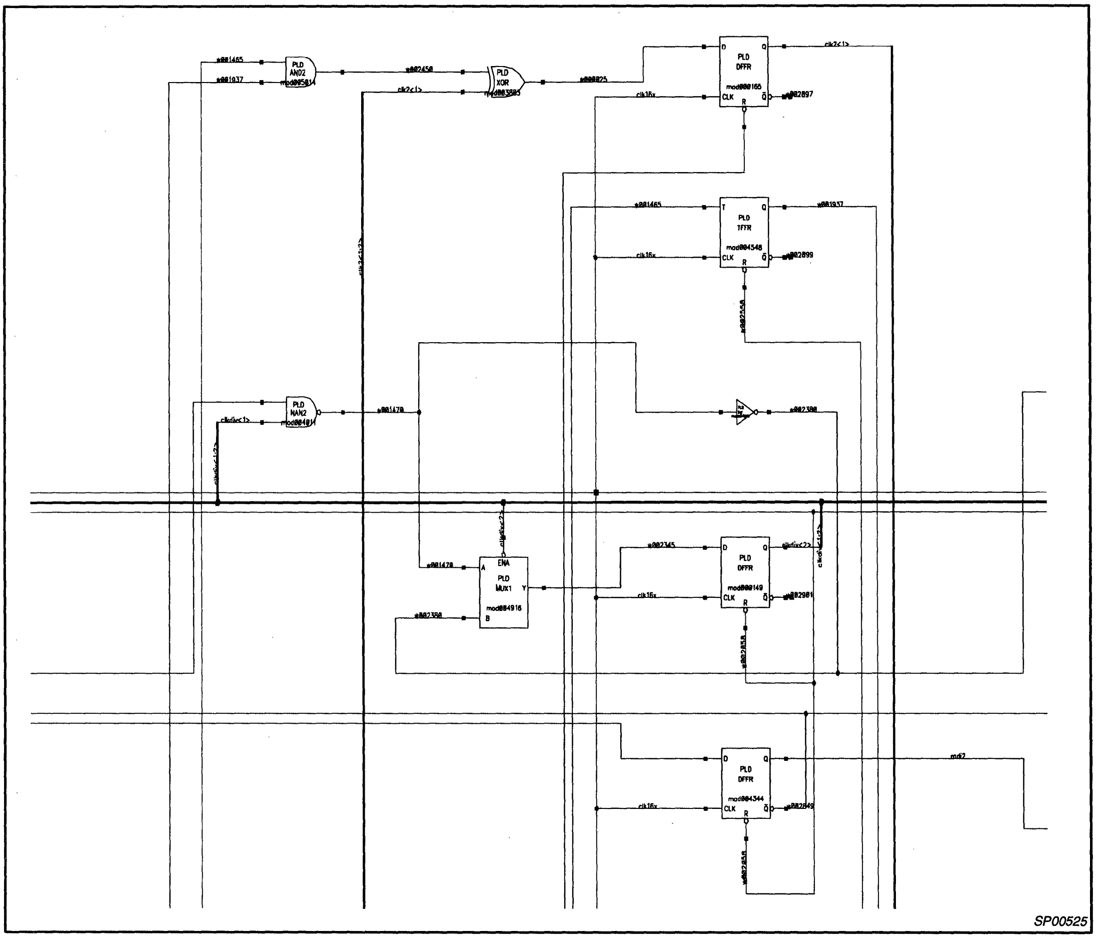


Figure 10C. Manchester Decoder Schematic (Cont.)

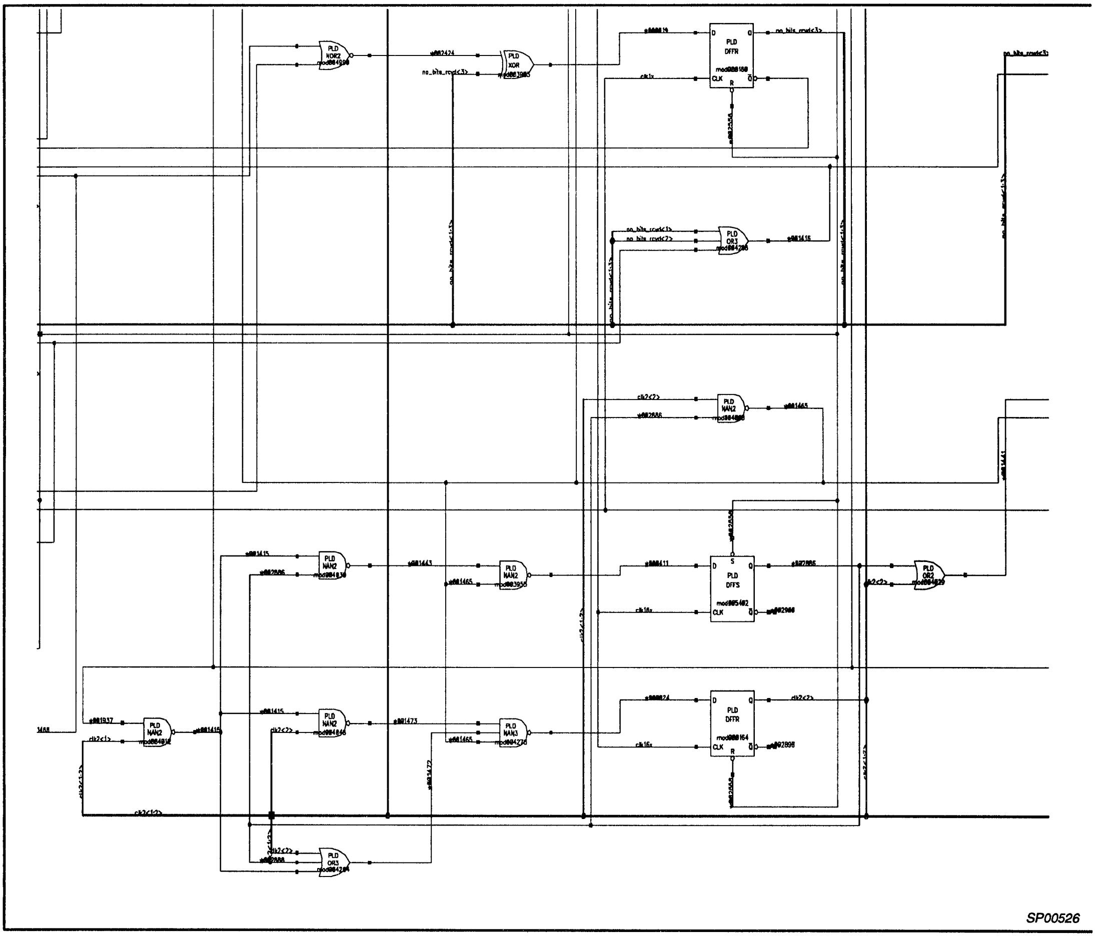


Figure 10D.   Manchester Decoder Schematic (Cont.)

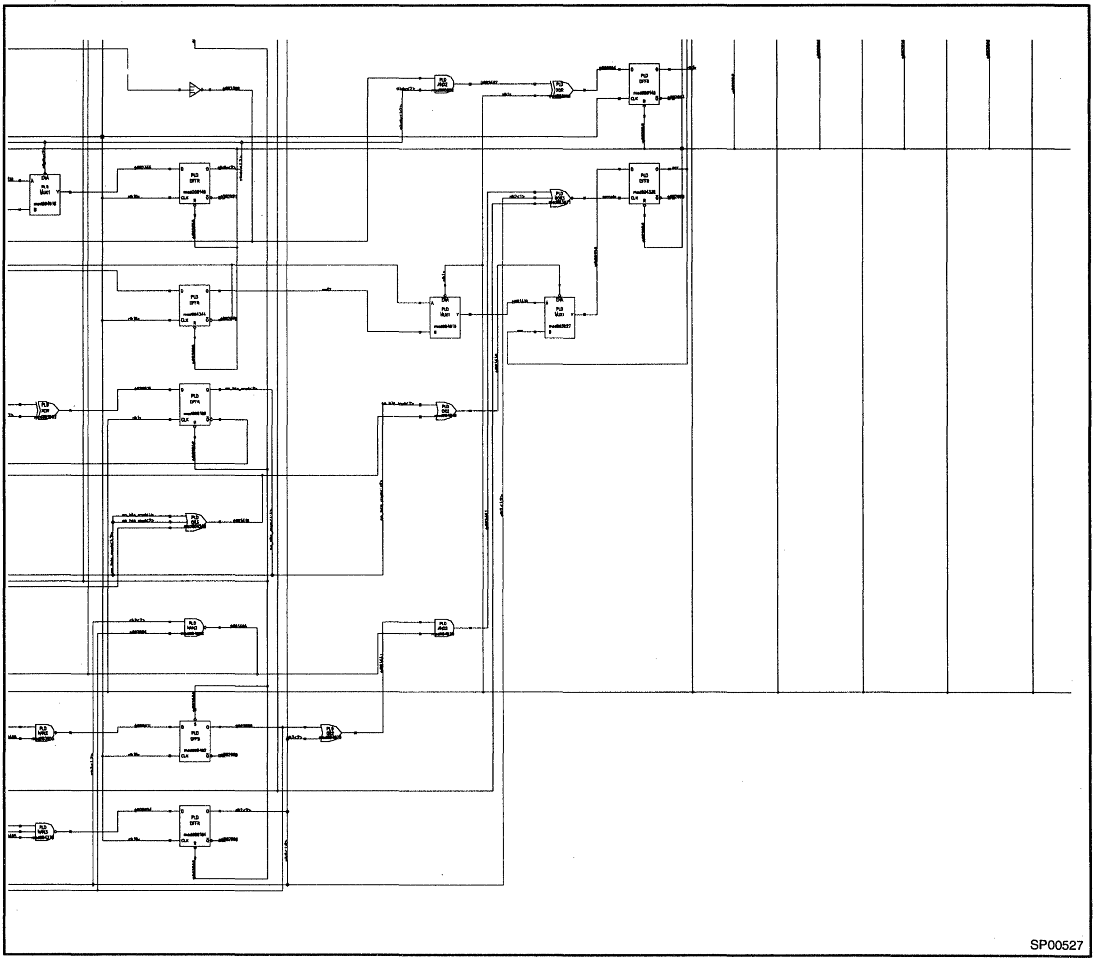


Figure 10E. Manchester Decoder Schematic (Cont.)

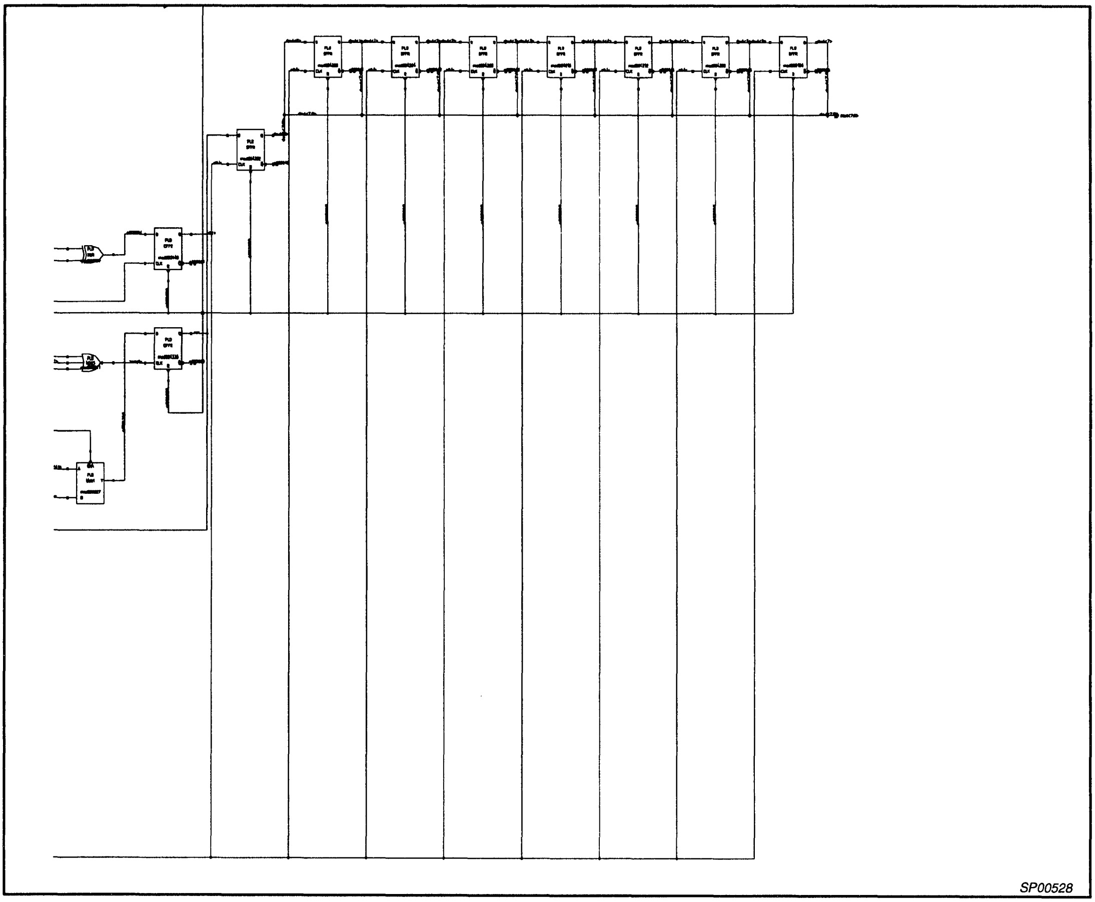


Figure 10F. Manchester Decoder Schematic (Cont.)

APPENDIX D: MANCHESTER ENCODER TEST FIXTURE

```
‘timescale 1 ns /1 ns
module me_tf ;
reg [7:0] din ;
reg rst ;
reg clk ;
reg wr ;
wire mdo ;
wire ready ;
me u1 (rst,clk,wr,din,ready,mdo) ;
initial begin
rst = 1’b0 ;
clk = 1’b0 ;
din = 8’h0 ;
wr = 1’b0 ;
me.clk1 = 1’b0 ;
me.count = 3’b0 ;
end
integer me_chann ;
initial begin
me_chann = $fopen(“me.rpt”) ;
$timeformat (–9,,,5) ;
end
parameter clock_period = 10,
setup_time = clock_period / 4 ;
always #(clock_period / 2) clk = ~clk ;
initial begin
$fdisplay(me_chann, “Verilog simulation of manchester encoder design\n\n”) ;
$shm_open(“me.shm”) ;
$shm_probe(“AS”) ;
$fmonitor(me_chann,”Time=%t,rst=%b,wr=%b,clk=%b,me.clk1=%b,din=%h,me.count=%b,mdo=%b,ready=%b”,$time,rst,wr,clk,me.clk1,din,me.
count,mdo,ready) ;
#5 rst = 1’b1 ;
#15 rst = 1’b0 ;
#(3 * clock_period – setup_time) din = 8’hff ;
#(1 * clock_period) wr = 1’b1 ;
#(2 * clock_period) wr = 1’b0 ;
#(20 * clock_period) din = 8’haa ;
#(1 * clock_period) wr = 1’b1 ;
#(2 * clock_period) wr = 1’b0 ;
#(20 * clock_period) din = 8’h00 ;
#(1 * clock_period) wr = 1’b1 ;
#(2 * clock_period) wr = 1’b0 ;
#(20 * clock_period) din = 8’hf0 ;
#(1 * clock_period) wr = 1’b1 ;
#(2 * clock_period) wr = 1’b0 ;
#(20 * clock_period) din = 8’h0f ;
#(1 * clock_period) wr = 1’b1 ;
#(2 * clock_period) wr = 1’b0 ;
#(100 * clock_period) ;
$fdisplay (me_chann,”\nSimulation of manchester encoder complete.”);
$finish ;

end
endmodule
```

APPENDIX E: MANCHESTER DECODER RESULTS

The intiial lines of the simulation results are given below.

```
Verilog simulation of manchester encoder design
Time=   0,rst=0,wr=0,clk=0,me.clk1=0,din=00,me.count=000,mdo=x,ready=x
Time=   5,rst=1,wr=0,clk=1,me.clk1=0,din=00,me.count=000,mdo=0,ready=0
Time=  10,rst=1,wr=0,clk=0,me.clk1=0,din=00,me.count=000,mdo=0,ready=0
Time=  15,rst=1,wr=0,clk=1,me.clk1=0,din=00,me.count=000,mdo=0,ready=0
Time=  20,rst=0,wr=0,clk=0,me.clk1=0,din=00,me.count=000,mdo=0,ready=0
Time=  25,rst=0,wr=0,clk=1,me.clk1=1,din=00,me.count=000,mdo=1,ready=1
Time=  30,rst=0,wr=0,clk=0,me.clk1=1,din=00,me.count=000,mdo=1,ready=1
Time=  35,rst=0,wr=0,clk=1,me.clk1=0,din=00,me.count=000,mdo=0,ready=1
Time=  40,rst=0,wr=0,clk=0,me.clk1=0,din=00,me.count=000,mdo=0,ready=1
Time=  45,rst=0,wr=0,clk=1,me.clk1=1,din=00,me.count=000,mdo=1,ready=1
Time=  48,rst=0,wr=0,clk=1,me.clk1=1,din=ff,me.count=000,mdo=1,ready=1
Time=  50,rst=0,wr=0,clk=0,me.clk1=1,din=ff,me.count=000,mdo=1,ready=1
Time=  55,rst=0,wr=0,clk=1,me.clk1=0,din=ff,me.count=000,mdo=0,ready=1
Time=  58,rst=0,wr=1,clk=1,me.clk1=0,din=ff,me.count=000,mdo=0,ready=1
Time=  60,rst=0,wr=1,clk=0,me.clk1=0,din=ff,me.count=000,mdo=0,ready=1
Time=  65,rst=0,wr=1,clk=1,me.clk1=1,din=ff,me.count=001,mdo=1,ready=0
Time=  70,rst=0,wr=1,clk=0,me.clk1=1,din=ff,me.count=001,mdo=1,ready=0
Time=  75,rst=0,wr=1,clk=1,me.clk1=0,din=ff,me.count=001,mdo=1,ready=0
Time=  78,rst=0,wr=0,clk=1,me.clk1=0,din=ff,me.count=001,mdo=1,ready=0
Time=  80,rst=0,wr=0,clk=0,me.clk1=0,din=ff,me.count=001,mdo=1,ready=0
Time=  85,rst=0,wr=0,clk=1,me.clk1=1,din=ff,me.count=010,mdo=0,ready=0
Time=  90,rst=0,wr=0,clk=0,me.clk1=1,din=ff,me.count=010,mdo=0,ready=0
Time=  95,rst=0,wr=0,clk=1,me.clk1=0,din=ff,me.count=010,mdo=1,ready=0
Time= 100,rst=0,wr=0,clk=0,me.clk1=0,din=ff,me.count=010,mdo=1,ready=0
Time= 105,rst=0,wr=0,clk=1,me.clk1=1,din=ff,me.count=011,mdo=0,ready=0
Time= 110,rst=0,wr=0,clk=0,me.clk1=1,din=ff,me.count=011,mdo=0,ready=0
Time= 115,rst=0,wr=0,clk=1,me.clk1=0,din=ff,me.count=011,mdo=1,ready=0
Time= 120,rst=0,wr=0,clk=0,me.clk1=0,din=ff,me.count=011,mdo=1,ready=0
Time= 125,rst=0,wr=0,clk=1,me.clk1=1,din=ff,me.count=100,mdo=0,ready=0
Time= 130,rst=0,wr=0,clk=0,me.clk1=1,din=ff,me.count=100,mdo=0,ready=0
Time= 135,rst=0,wr=0,clk=1,me.clk1=0,din=ff,me.count=100,mdo=1,ready=0
Time= 140,rst=0,wr=0,clk=0,me.clk1=0,din=ff,me.count=100,mdo=1,ready=0
Time= 145,rst=0,wr=0,clk=1,me.clk1=1,din=ff,me.count=101,mdo=0,ready=0
Time= 150,rst=0,wr=0,clk=0,me.clk1=1,din=ff,me.count=101,mdo=0,ready=0
Time= 155,rst=0,wr=0,clk=1,me.clk1=0,din=ff,me.count=101,mdo=1,ready=0
Time= 160,rst=0,wr=0,clk=0,me.clk1=0,din=ff,me.count=101,mdo=1,ready=0
Time= 165,rst=0,wr=0,clk=1,me.clk1=1,din=ff,me.count=110,mdo=0,ready=0
Time= 170,rst=0,wr=0,clk=0,me.clk1=1,din=ff,me.count=110,mdo=0,ready=0
Time= 175,rst=0,wr=0,clk=1,me.clk1=0,din=ff,me.count=110,mdo=1,ready=0
Time= 180,rst=0,wr=0,clk=0,me.clk1=0,din=ff,me.count=110,mdo=1,ready=0
Time= 185,rst=0,wr=0,clk=1,me.clk1=1,din=ff,me.count=111,mdo=0,ready=0
Time= 190,rst=0,wr=0,clk=0,me.clk1=1,din=ff,me.count=111,mdo=0,ready=0
Time= 195,rst=0,wr=0,clk=1,me.clk1=0,din=ff,me.count=111,mdo=1,ready=0
Time= 200,rst=0,wr=0,clk=0,me.clk1=0,din=ff,me.count=111,mdo=1,ready=0
Time= 205,rst=0,wr=0,clk=1,me.clk1=1,din=ff,me.count=000,mdo=0,ready=0
Time= 210,rst=0,wr=0,clk=0,me.clk1=1,din=ff,me.count=000,mdo=0,ready=0
Time= 215,rst=0,wr=0,clk=1,me.clk1=0,din=ff,me.count=000,mdo=1,ready=0
Time= 220,rst=0,wr=0,clk=0,me.clk1=0,din=ff,me.count=000,mdo=1,ready=0
```

APPENDIX F: SYNC AND VALID MANCHESTER DETECTOR TEST FIXTURE

```
‘timescale 1 ns /1 ns
module sync_det_tf ;
reg rst ;
reg clk2x ;
reg mdi ;
wire mdo ;
wire sync_pulse ;
wire valid_md ;
sync_det u1 (rst,clk2x,mdi,mdo,valid_md,sync_pulse) ;
initial begin
rst = 1’b0 ;
clk2x = 1’b0 ;
mdi = 1’b0 ;
end
integer sync_det_chann ;
initial begin
sync_det_chann = $fopen(“sync_det.rpt”) ;
$timeformat (–9,,,5) ;
end
parameter clock_period = 100;
always #(clock_period/2) clk2x = ~clk2x ;
initial begin
$fdisplay(sync_det_chann, “Verilog simulation of sync_det design\n\n”) ;
$shm_open(“sync_det.shm”) ;
$shm_probe(“AS”) ;
$fmonitor(sync_det_chann,”Time=%t,rst=%b,clk2=%b,mdi=%b,md=%b,sync_pulse=%b,valid_md=%b”,$time,rst,clk2x,mdi,sync_det.md,sync_p
ulse,valid_md) ;
#1 rst = 1’b1 ;
#10 rst = 1’b0 ;
// start w a valic command sync pulse 8 0s
#38 mdi = 1’b1 ;
#300 mdi = 1’b0 ;
#300 mdi = 1’b1 ;
#100 mdi = 1’b0 ;
#100 mdi = 1’b1 ;
#100 mdi = 1’b0 ;
#100 mdi = 1’b1 ;
#100 mdi = 1’b0 ;
#100 mdi = 1’b1 ;
#100 mdi = 1’b0 ;
#100 mdi = 1’b1 ;
#100 mdi = 1’b0 ;
#100 mdi = 1’b1 ;
#100 mdi = 1’b0 ;
#100 mdi = 1’b1 ;
#100 mdi = 1’b0 ;
#100 mdi = 1’b1 ;
#100 mdi = 1’b0 ;
// no change for 500 ns should cause valid_md to go low
#500
// data sync pulse
#300 mdi = 1’b1 ;
#300 mdi = 1’b0 ;
#100 mdi = 1’b1 ;
```

```
#100 mdi = 1'b0 ;
#100 mdi = 1'b1 ;
#100 mdi = 1'b0 ;
#100 mdi = 1'b1 ;
#100 mdi = 1'b0 ;
#100 mdi = 1'b1 ;
#100 mdi = 1'b0 ;
#100 mdi = 1'b1 ;
#100 mdi = 1'b0 ;
#100 mdi = 1'b1 ;
#100 mdi = 1'b0 ;
$fdisplay (sync_det_chann,"\nSimulation of sync_det complete.");
$finish ;
end
endmodule
```

APPENDIX G: SYNC AND VALID MANCHESTER DETECTOR SIMULATION RESULTS

Verilog simulation of sync_det design

```
Time=    0,rst=0,clk2=0,mdi=0,md=xxxxxxxxxx,sync_pulse=x,valid_md=x
Time=    1,rst=1,clk2=0,mdi=0,md=0000000000,sync_pulse=0,valid_md=0
Time=   11,rst=0,clk2=0,mdi=0,md=0000000000,sync_pulse=0,valid_md=0
Time=   49,rst=0,clk2=0,mdi=1,md=0000000000,sync_pulse=0,valid_md=0
Time=   50,rst=0,clk2=1,mdi=1,md=0000000001,sync_pulse=0,valid_md=0
Time=  100,rst=0,clk2=0,mdi=1,md=0000000001,sync_pulse=0,valid_md=0
Time=  150,rst=0,clk2=1,mdi=1,md=0000000011,sync_pulse=0,valid_md=0
Time=  200,rst=0,clk2=0,mdi=1,md=0000000011,sync_pulse=0,valid_md=0
Time=  250,rst=0,clk2=1,mdi=1,md=0000000111,sync_pulse=0,valid_md=0
Time=  300,rst=0,clk2=0,mdi=1,md=0000000111,sync_pulse=0,valid_md=0
Time=  349,rst=0,clk2=0,mdi=0,md=0000000111,sync_pulse=0,valid_md=0
Time=  350,rst=0,clk2=1,mdi=0,md=0000001110,sync_pulse=0,valid_md=0
Time=  400,rst=0,clk2=0,mdi=0,md=0000001110,sync_pulse=0,valid_md=0
Time=  450,rst=0,clk2=1,mdi=0,md=0000011100,sync_pulse=0,valid_md=0
Time=  500,rst=0,clk2=0,mdi=0,md=0000011100,sync_pulse=0,valid_md=0
Time=  550,rst=0,clk2=1,mdi=0,md=0000111000,sync_pulse=0,valid_md=0
Time=  600,rst=0,clk2=0,mdi=0,md=0000111000,sync_pulse=0,valid_md=0
Time=  649,rst=0,clk2=0,mdi=1,md=0000111000,sync_pulse=0,valid_md=0
Time=  650,rst=0,clk2=1,mdi=1,md=0001110001,sync_pulse=0,valid_md=0
Time=  700,rst=0,clk2=0,mdi=1,md=0001110001,sync_pulse=0,valid_md=0
Time=  749,rst=0,clk2=0,mdi=0,md=0001110001,sync_pulse=0,valid_md=0
Time=  750,rst=0,clk2=1,mdi=0,md=0011100010,sync_pulse=0,valid_md=0
Time=  800,rst=0,clk2=0,mdi=0,md=0011100010,sync_pulse=0,valid_md=0
Time=  849,rst=0,clk2=0,mdi=1,md=0011100010,sync_pulse=0,valid_md=0
Time=  850,rst=0,clk2=1,mdi=1,md=0111000101,sync_pulse=0,valid_md=1
Time=  900,rst=0,clk2=0,mdi=1,md=0111000101,sync_pulse=0,valid_md=1
Time=  949,rst=0,clk2=0,mdi=0,md=0111000101,sync_pulse=0,valid_md=1
Time=  950,rst=0,clk2=1,mdi=0,md=1110001010,sync_pulse=1,valid_md=1
Time= 1000,rst=0,clk2=0,mdi=0,md=1110001010,sync_pulse=1,valid_md=1
Time= 1049,rst=0,clk2=0,mdi=1,md=1110001010,sync_pulse=1,valid_md=1
Time= 1050,rst=0,clk2=1,mdi=1,md=1100010101,sync_pulse=0,valid_md=1
Time= 1100,rst=0,clk2=0,mdi=1,md=1100010101,sync_pulse=0,valid_md=1
Time= 1149,rst=0,clk2=0,mdi=0,md=1100010101,sync_pulse=0,valid_md=1
Time= 1150,rst=0,clk2=1,mdi=0,md=1000101010,sync_pulse=0,valid_md=1
Time= 1200,rst=0,clk2=0,mdi=0,md=1000101010,sync_pulse=0,valid_md=1
Time= 1249,rst=0,clk2=0,mdi=1,md=1000101010,sync_pulse=0,valid_md=1
Time= 1250,rst=0,clk2=1,mdi=1,md=0001010101,sync_pulse=0,valid_md=1
Time= 1300,rst=0,clk2=0,mdi=1,md=0001010101,sync_pulse=0,valid_md=1
Time= 1349,rst=0,clk2=0,mdi=0,md=0001010101,sync_pulse=0,valid_md=1
Time= 1350,rst=0,clk2=1,mdi=0,md=0010101010,sync_pulse=0,valid_md=1
Time= 1400,rst=0,clk2=0,mdi=0,md=0010101010,sync_pulse=0,valid_md=1
Time= 1449,rst=0,clk2=0,mdi=1,md=0010101010,sync_pulse=0,valid_md=1
Time= 1450,rst=0,clk2=1,mdi=1,md=0101010101,sync_pulse=0,valid_md=1
Time= 1500,rst=0,clk2=0,mdi=1,md=0101010101,sync_pulse=0,valid_md=1
Time= 1549,rst=0,clk2=0,mdi=0,md=0101010101,sync_pulse=0,valid_md=1
Time= 1550,rst=0,clk2=1,mdi=0,md=1010101010,sync_pulse=0,valid_md=1
Time= 1600,rst=0,clk2=0,mdi=0,md=1010101010,sync_pulse=0,valid_md=1
Time= 1649,rst=0,clk2=0,mdi=1,md=1010101010,sync_pulse=0,valid_md=1
Time= 1650,rst=0,clk2=1,mdi=1,md=0101010101,sync_pulse=0,valid_md=1
Time= 1700,rst=0,clk2=0,mdi=1,md=0101010101,sync_pulse=0,valid_md=1
Time= 1749,rst=0,clk2=0,mdi=0,md=0101010101,sync_pulse=0,valid_md=1
Time= 1750,rst=0,clk2=1,mdi=0,md=1010101010,sync_pulse=0,valid_md=1
Time= 1800,rst=0,clk2=0,mdi=0,md=1010101010,sync_pulse=0,valid_md=1
```

```
Time= 1849,rst=0,clk2=0,mdi=1,md=1010101010,sync_pulse=0,valid_md=1
Time= 1850,rst=0,clk2=1,mdi=1,md=0101010101,sync_pulse=0,valid_md=1
Time= 1900,rst=0,clk2=0,mdi=1,md=0101010101,sync_pulse=0,valid_md=1
Time= 1949,rst=0,clk2=0,mdi=0,md=0101010101,sync_pulse=0,valid_md=1
Time= 1950,rst=0,clk2=1,mdi=0,md=1010101010,sync_pulse=0,valid_md=1
Time= 2000,rst=0,clk2=0,mdi=0,md=1010101010,sync_pulse=0,valid_md=1
Time= 2049,rst=0,clk2=0,mdi=1,md=1010101010,sync_pulse=0,valid_md=1
Time= 2050,rst=0,clk2=1,mdi=1,md=0101010101,sync_pulse=0,valid_md=1
Time= 2100,rst=0,clk2=0,mdi=1,md=0101010101,sync_pulse=0,valid_md=1
Time= 2149,rst=0,clk2=0,mdi=0,md=0101010101,sync_pulse=0,valid_md=1
Time= 2150,rst=0,clk2=1,mdi=0,md=1010101010,sync_pulse=0,valid_md=1
Time= 2200,rst=0,clk2=0,mdi=0,md=1010101010,sync_pulse=0,valid_md=1
Time= 2250,rst=0,clk2=1,mdi=0,md=0101010100,sync_pulse=0,valid_md=0
Time= 2300,rst=0,clk2=0,mdi=0,md=0101010100,sync_pulse=0,valid_md=0
Time= 2350,rst=0,clk2=1,mdi=0,md=1010101000,sync_pulse=0,valid_md=0
Time= 2400,rst=0,clk2=0,mdi=0,md=1010101000,sync_pulse=0,valid_md=0
Time= 2450,rst=0,clk2=1,mdi=0,md=0101010000,sync_pulse=0,valid_md=0
Time= 2500,rst=0,clk2=0,mdi=0,md=0101010000,sync_pulse=0,valid_md=0
Time= 2550,rst=0,clk2=1,mdi=0,md=1010100000,sync_pulse=0,valid_md=0
Time= 2600,rst=0,clk2=0,mdi=0,md=1010100000,sync_pulse=0,valid_md=0
Time= 2650,rst=0,clk2=1,mdi=0,md=0101000000,sync_pulse=0,valid_md=0
Time= 2700,rst=0,clk2=0,mdi=0,md=0101000000,sync_pulse=0,valid_md=0
Time= 2750,rst=0,clk2=1,mdi=0,md=1010000000,sync_pulse=0,valid_md=0
Time= 2800,rst=0,clk2=0,mdi=0,md=1010000000,sync_pulse=0,valid_md=0
Time= 2850,rst=0,clk2=1,mdi=0,md=0100000000,sync_pulse=0,valid_md=0
Time= 2900,rst=0,clk2=0,mdi=0,md=0100000000,sync_pulse=0,valid_md=0
Time= 2949,rst=0,clk2=0,mdi=1,md=0100000000,sync_pulse=0,valid_md=0
Time= 2950,rst=0,clk2=1,mdi=1,md=1000000001,sync_pulse=0,valid_md=0
Time= 3000,rst=0,clk2=0,mdi=1,md=1000000001,sync_pulse=0,valid_md=0
Time= 3050,rst=0,clk2=1,mdi=1,md=0000000011,sync_pulse=0,valid_md=0
Time= 3100,rst=0,clk2=0,mdi=1,md=0000000011,sync_pulse=0,valid_md=0
Time= 3150,rst=0,clk2=1,mdi=1,md=0000000111,sync_pulse=0,valid_md=0
Time= 3200,rst=0,clk2=0,mdi=1,md=0000000111,sync_pulse=0,valid_md=0
Time= 3249,rst=0,clk2=0,mdi=0,md=0000000111,sync_pulse=0,valid_md=0
Time= 3250,rst=0,clk2=1,mdi=0,md=0000001110,sync_pulse=0,valid_md=0
Time= 3300,rst=0,clk2=0,mdi=0,md=0000001110,sync_pulse=0,valid_md=0
Time= 3349,rst=0,clk2=0,mdi=1,md=0000001110,sync_pulse=0,valid_md=0
Time= 3350,rst=0,clk2=1,mdi=1,md=0000011101,sync_pulse=0,valid_md=0
Time= 3400,rst=0,clk2=0,mdi=1,md=0000011101,sync_pulse=0,valid_md=0
Time= 3449,rst=0,clk2=0,mdi=0,md=0000011101,sync_pulse=0,valid_md=0
Time= 3450,rst=0,clk2=1,mdi=0,md=0000111010,sync_pulse=0,valid_md=1
Time= 3500,rst=0,clk2=0,mdi=0,md=0000111010,sync_pulse=0,valid_md=1
Time= 3549,rst=0,clk2=0,mdi=1,md=0000111010,sync_pulse=0,valid_md=1
Time= 3550,rst=0,clk2=1,mdi=1,md=0001110101,sync_pulse=1,valid_md=1
Time= 3600,rst=0,clk2=0,mdi=1,md=0001110101,sync_pulse=1,valid_md=1
Time= 3649,rst=0,clk2=0,mdi=0,md=0001110101,sync_pulse=1,valid_md=1
Time= 3650,rst=0,clk2=1,mdi=0,md=0011101010,sync_pulse=0,valid_md=1
Time= 3700,rst=0,clk2=0,mdi=0,md=0011101010,sync_pulse=0,valid_md=1
Time= 3749,rst=0,clk2=0,mdi=1,md=0011101010,sync_pulse=0,valid_md=1
Time= 3750,rst=0,clk2=1,mdi=1,md=0111010101,sync_pulse=0,valid_md=1
Time= 3800,rst=0,clk2=0,mdi=1,md=0111010101,sync_pulse=0,valid_md=1
Time= 3849,rst=0,clk2=0,mdi=0,md=0111010101,sync_pulse=0,valid_md=1
Time= 3850,rst=0,clk2=1,mdi=0,md=1110101010,sync_pulse=0,valid_md=1
Time= 3900,rst=0,clk2=0,mdi=0,md=1110101010,sync_pulse=0,valid_md=1
Time= 3949,rst=0,clk2=0,mdi=1,md=1110101010,sync_pulse=0,valid_md=1
```

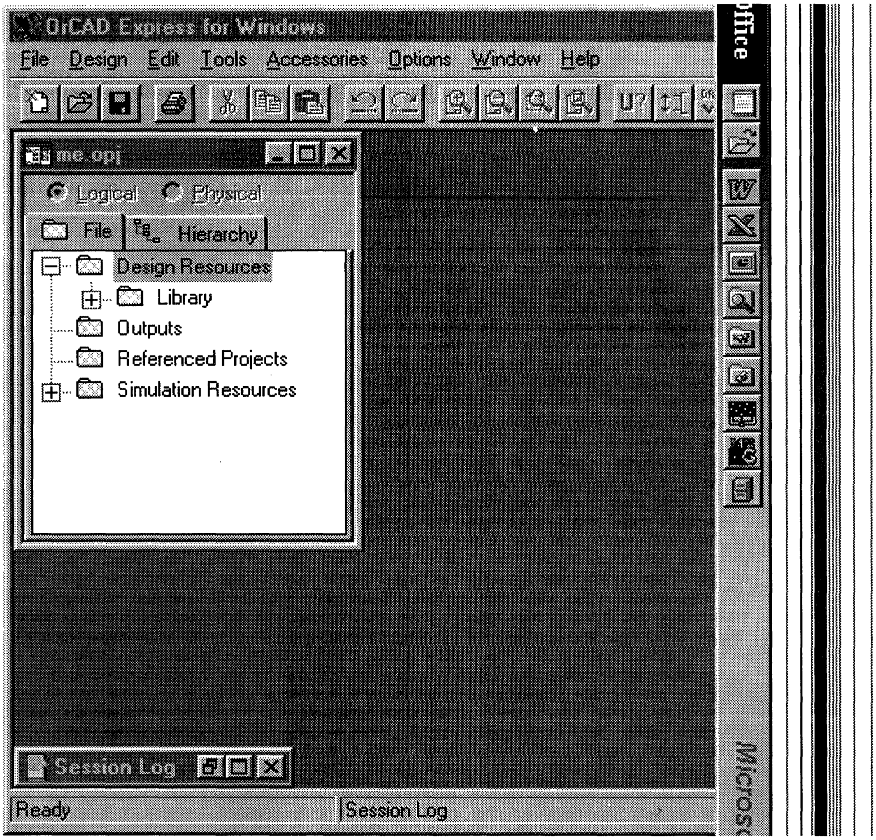
```
Time= 3950,rst=0,clk2=1,mdi=1,md=1101010101,sync_pulse=0,valid_md=1
Time= 4000,rst=0,clk2=0,mdi=1,md=1101010101,sync_pulse=0,valid_md=1
Time= 4049,rst=0,clk2=0,mdi=0,md=1101010101,sync_pulse=0,valid_md=1
Time= 4050,rst=0,clk2=1,mdi=0,md=1010101010,sync_pulse=0,valid_md=1
Time= 4100,rst=0,clk2=0,mdi=0,md=1010101010,sync_pulse=0,valid_md=1
Time= 4149,rst=0,clk2=0,mdi=1,md=1010101010,sync_pulse=0,valid_md=1
Time= 4150,rst=0,clk2=1,mdi=1,md=0101010101,sync_pulse=0,valid_md=1
Time= 4200,rst=0,clk2=0,mdi=1,md=0101010101,sync_pulse=0,valid_md=1
Time= 4249,rst=0,clk2=0,mdi=0,md=0101010101,sync_pulse=0,valid_md=1
Time= 4250,rst=0,clk2=1,mdi=0,md=1010101010,sync_pulse=0,valid_md=1
Time= 4300,rst=0,clk2=0,mdi=0,md=1010101010,sync_pulse=0,valid_md=1
Time= 4349,rst=0,clk2=0,mdi=1,md=1010101010,sync_pulse=0,valid_md=1
Time= 4350,rst=0,clk2=1,mdi=1,md=0101010101,sync_pulse=0,valid_md=1
Time= 4400,rst=0,clk2=0,mdi=1,md=0101010101,sync_pulse=0,valid_md=1
Time= 4449,rst=0,clk2=0,mdi=0,md=0101010101,sync_pulse=0,valid_md=1
Time= 4450,rst=0,clk2=1,mdi=0,md=1010101010,sync_pulse=0,valid_md=1
Time= 4500,rst=0,clk2=0,mdi=0,md=1010101010,sync_pulse=0,valid_md=1
```

```
Simulation of sync_det complete.
```

OrCad express– Minc PLDesigner flow for Philips CPLDs

AN071

Author Lester Sanders, CPLD Applications Engineer



Philips acknowledges the trademarks of the companies mentioned in this document.

INTRODUCTION

This note provides the steps for using the OrCAD Express and Minc PLDesigner tools to simulate and compile a digital design into Philips' Complex Programable Logic Device (PLDs). Philips provides fast zero power CPLDs which are footprint compatible of the Altera 7000 Series of CPLDs. This design is generated using schematic,VHDL synthesis, and simulation tools from OrCAD Express, and compiled to a jedec file using PLDesigner.

This note imports a VHDL source file and creates a schematic text entry. It illustrates one approach of many for targeting Philips CPLDs using OrCAD Express. Please see the OrCAD and Minc documentation to learn more about the capability of these tools

Technical support for the design flow described in this application note is provided by:

Philips Technical Assistance
Telephone no. 888–coolpld
email – support@coolpld.com
web site – http://www.coolpld.com
Fax on Demand – 800 282–2000

Minc Technical Assistance
apps@minc.com 719 590–1155

OrCAD Technical Assistance
techsupport@orcad.com 503 671 9400

References

OrCAD Express for Windows User's Guide

Minc PLDesigner–XL User's Guide

Installation Requirements

This design requires the following PC–based CAE tools:

OrCAD Express v 7.0

PLDesigner v 3.8

OrCad express– Minc PLDesigner flow for Philips CPLDs

Philips CPLD fitters for the PZ3000/PZ5000 Series

This design targets the Philips PZ3032 complex programmable logic device.This design is a manchester encoder which transmits data in a manner similar to that of a UART. See Philips application note *VHDL Implementation of a Manchester Encoder Decoder* for the advantages of Manchester code and for the source code for the Manchester decoder.

To begin, from the Windows Program Manager (Figure 1), double–click on the OrCAD Express icon to invoke the OrCAD.

Select File New and Capture Project from the dialog box. Name the project me. Select the Programmable Logic Wizard and the AMD Mach as a target (Philips CoolRunner).

OrCAD Express supports top down and bottom up design. This example starts with a VHDL description of a manchester encoder (me) and creates a symbol and project with that name. See Chapters 4 and 7 of the OrCAD Express for Windows manual for project manangement using OrCAD.

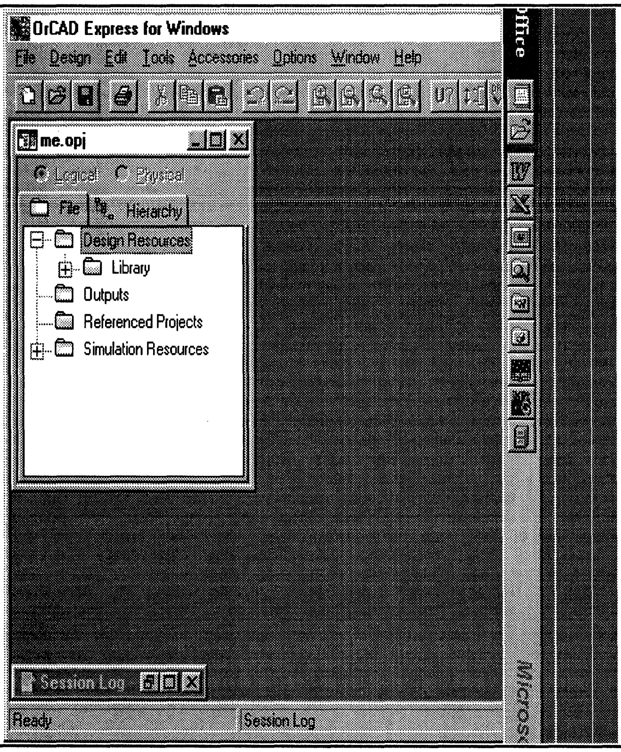
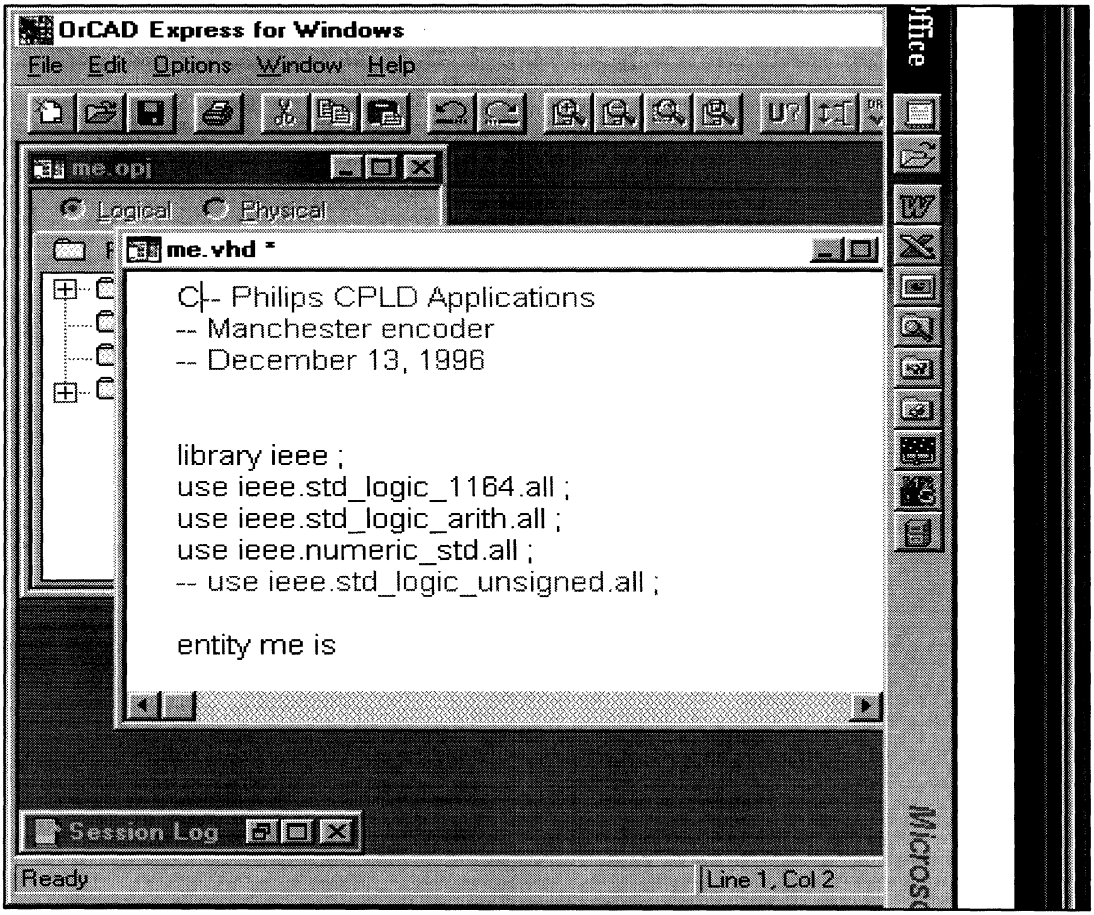


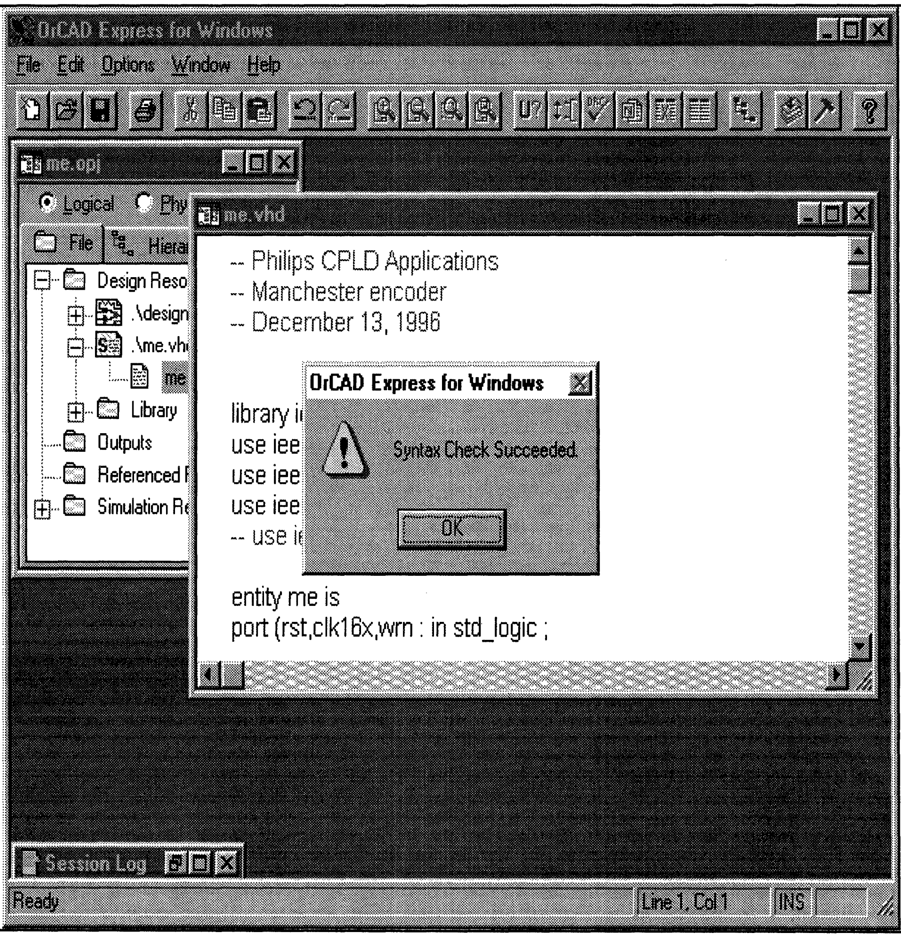
Figure 1. OrCAD Express GUI



```
C-- Philips CPLD Applications
-- Manchester encoder
-- December 13, 1996


library ieee ;
use ieee.std_logic_1164.all ;
use ieee.std_logic_arith.all ;
use ieee.numeric_std.all ;
-- use ieee.std_logic_unsigned.all ;

entity me is
```

Figure 2. Importing Existing VHDL Source

Select File Open and Files of Type VHDL, and browse to the directory containing me.vhd. The code for me.vhd is available on the http://www.coolpld.com website. Notice the syntax error caused by the character C in line 1.



```
-- Philips CPLD Applications
-- Manchester encoder
-- December 13, 1996


library i
use iee
use iee
use iee
-- use i

entity me is
port (rst,clk16x,wrn : in std_logic ;
```

OrCAD Express for Windows

Syntax Check Succeeded.

OK

Figure 3. VHDL Syntax Check

Select Edit VHDL Syntax Check. Correct the error and recheck the syntax.

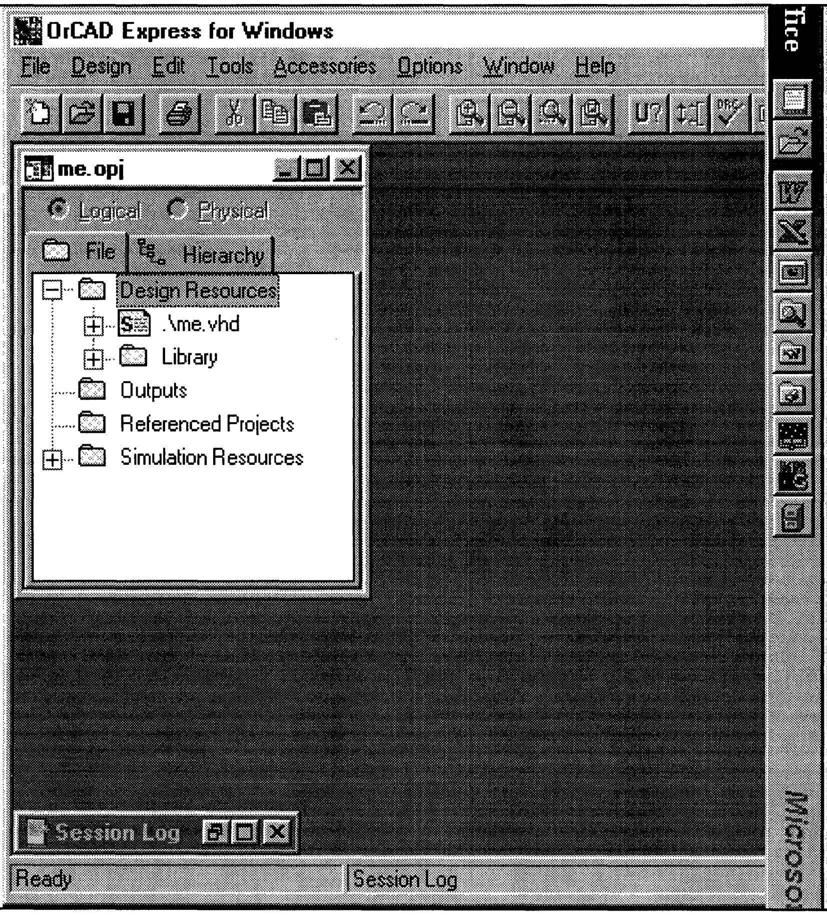


Figure 4. me Project With me.vhd Included as a Design Resource

As shown in Figure 4, the me.vhd source file has been incorporated as a Design Resource in OrCAD Express. The next step is to create a schematic containing the Manchester encoder and package ports.

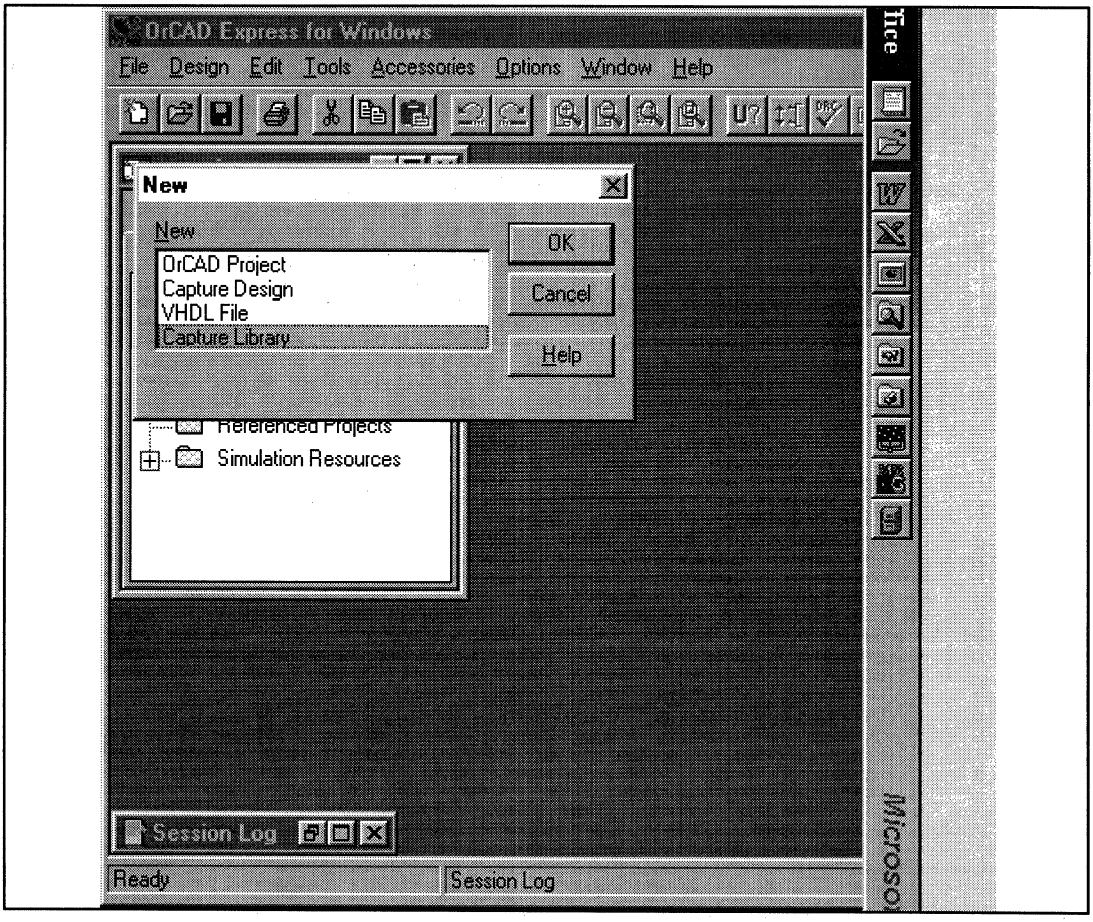


Figure 5. Creating a New Schematic

To add a schematic, select File New, and highlight from the dialog box the Capture Design type, and click OK. Name the schematic me.

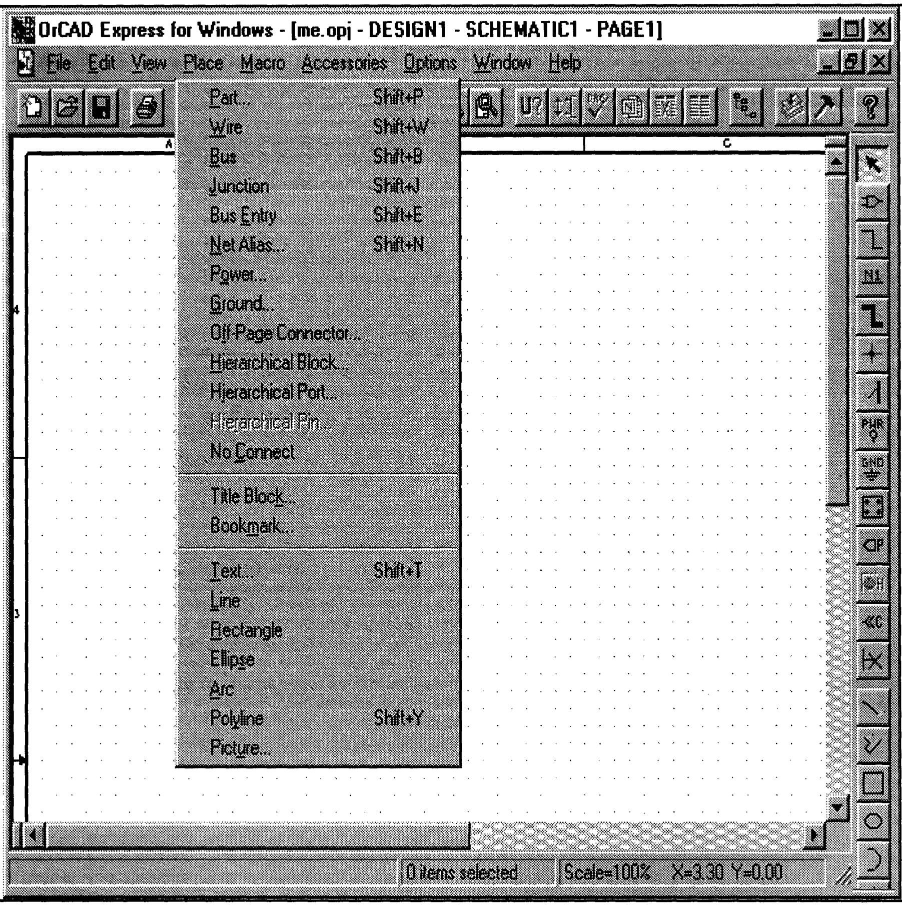
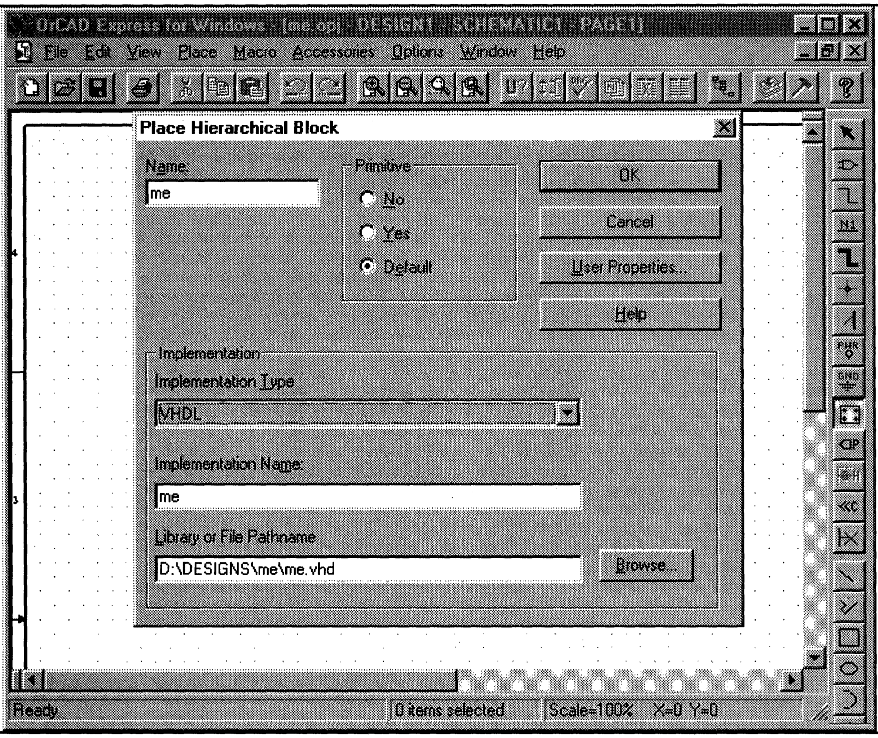


Figure 6. Creating a Symbol

To create a symbol for me, select Place Hierarchical Block.


OrCAD Express for Windows - [me.opj - DESIGN1 - SCHEMATIC1 - PAGE1]
File Edit View Place Macro Accessories Options Window Help
Place Hierarchical Block
OK
Cancel
User Properties...
Help
Browse...
Ready
0 items selected
Scale=100% X=0 Y=0

Name: me

Primitive
○ No
○ Yes
● Default

Implementation
Implementation Type: VHDL
Implementation Name: me
Library or File Pathname: D:\DESIGNS\me\me.vhd

Figure 7. Placing the me Hierarchical Block

Fill in the fields in the dialog box with me in the name box, VHDL as the implementation type in the Implementation Type list box, me as the entity name for the model in the Implementation name text box, and me.vhd in the File pathname text box. Use the tab key or the mouse to move between fields. Select OK.

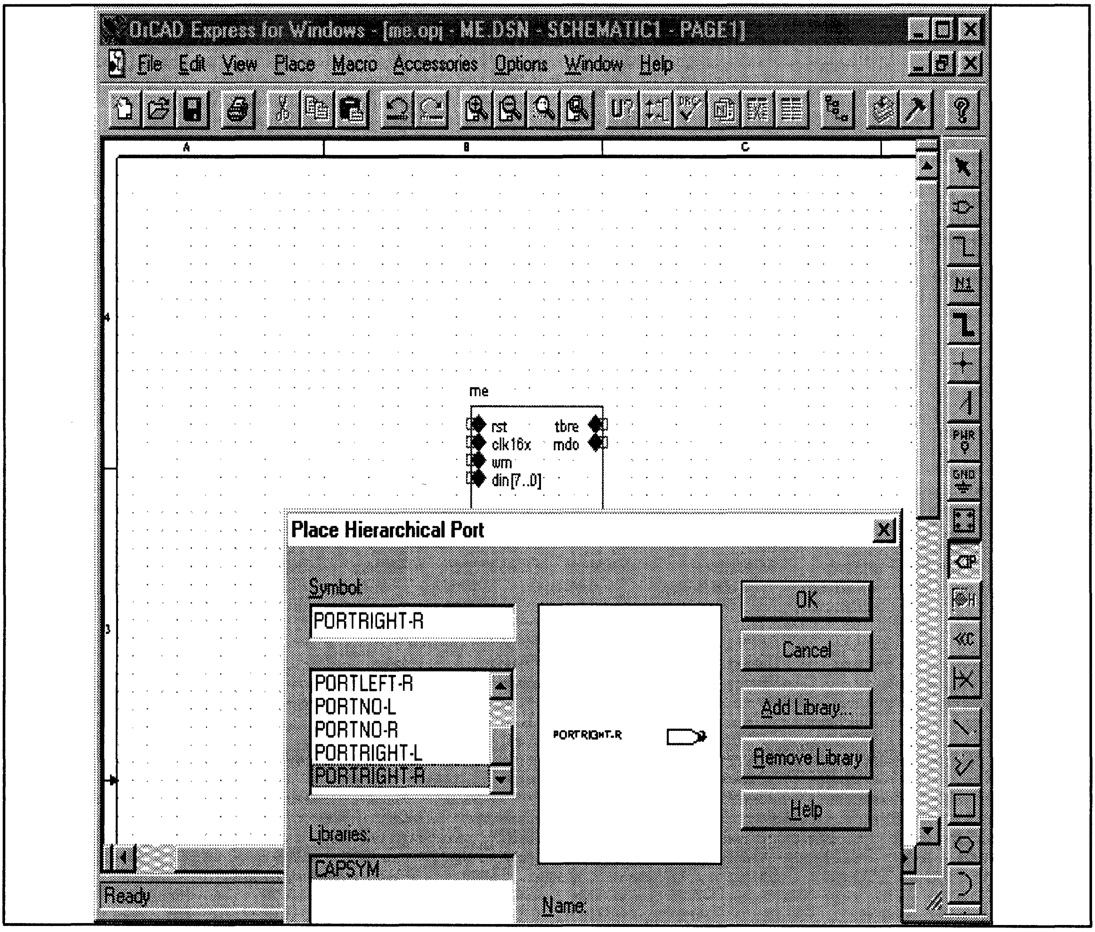
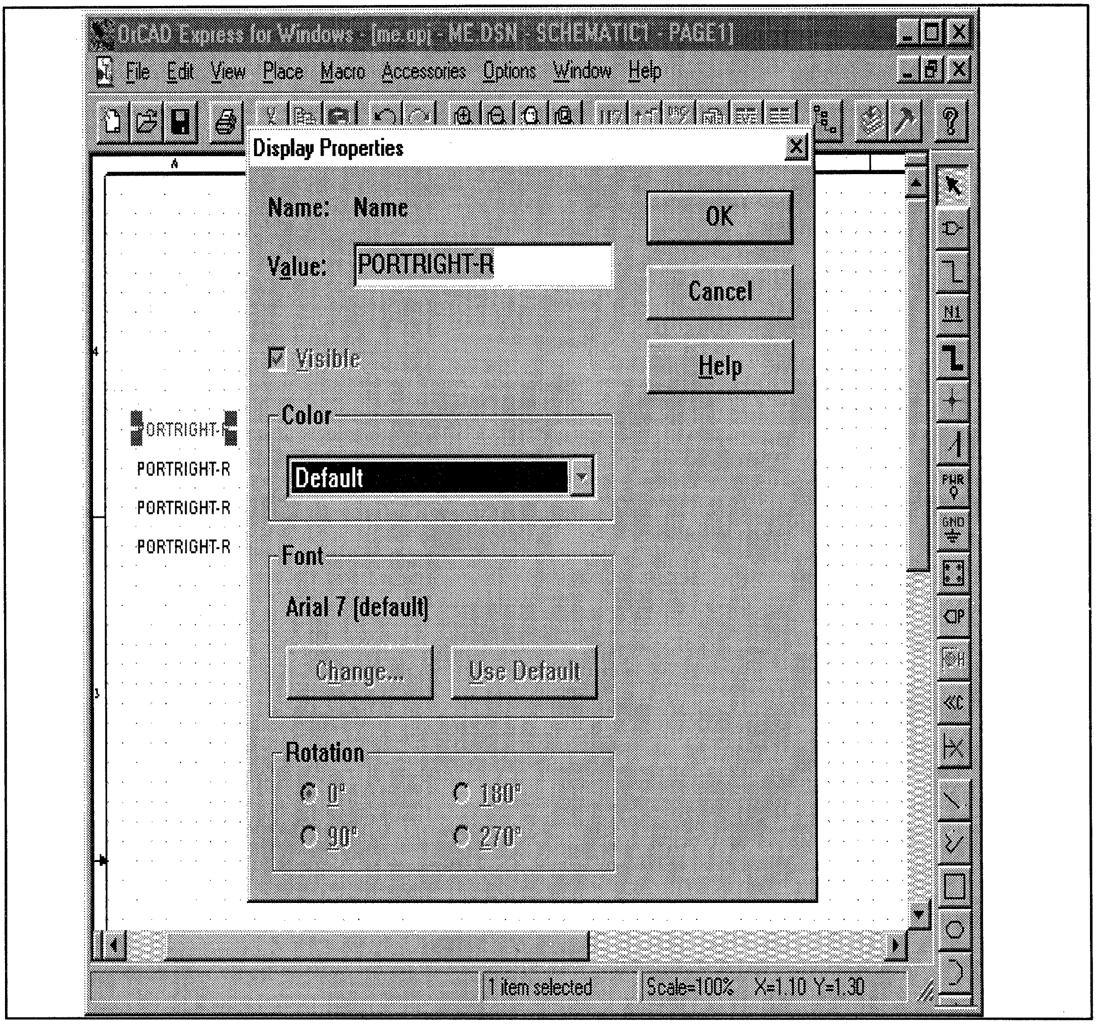


Figure 8. Adding Hierarchical Ports

Once the symbol is generated, place it in the center of the schematic page. From the fixed menu on the right hand side, select Place Hierarchical port and add the ports.



Name: Name

Value: PORTRIGHT-R

☑ Visible

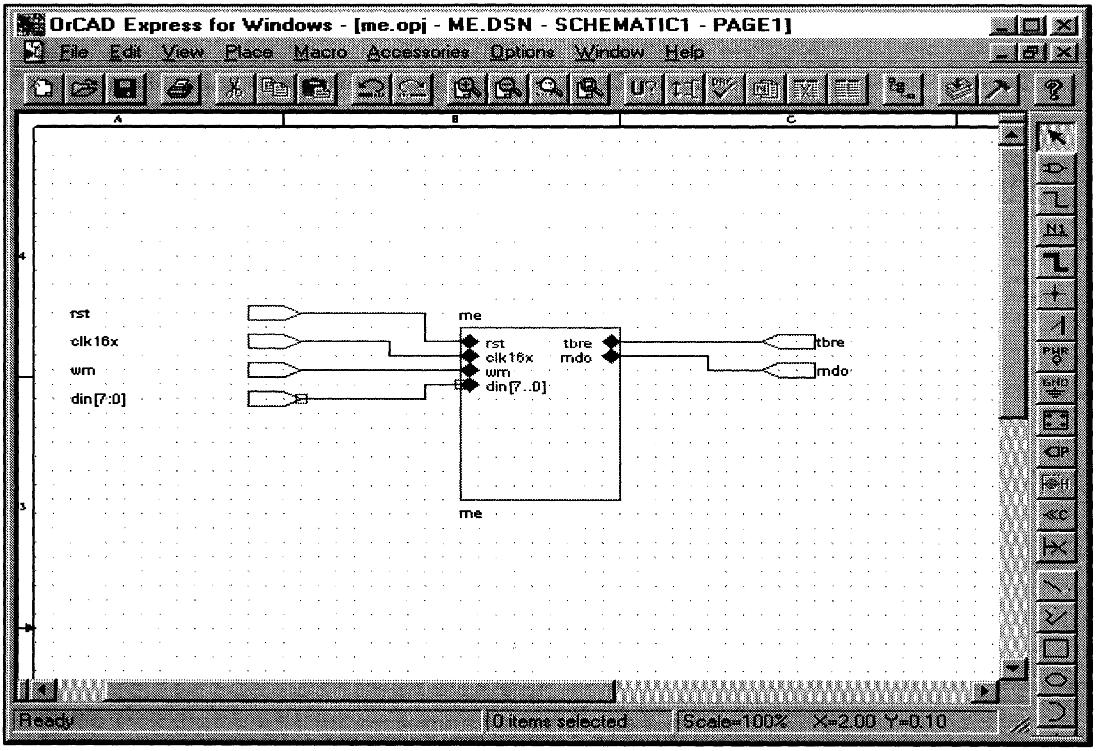
Color: Default

Font: Arial 7 (default)

Rotation: ● 0° ○ 180° ○ 90° ○ 270°

Figure 9. Labeling Ports

Double click on the hierarchical port name and label the ports.


OrCAD Express for Windows - [me.opj - ME.DSN - SCHEMATIC1 - PAGE1]

Ready | 0 items selected | Scale=100% X=2.00 Y=0.10

Figure 10. Adding Wires

Select Place Wire and draw the wires. Once done, save the schematic as me.dsn.

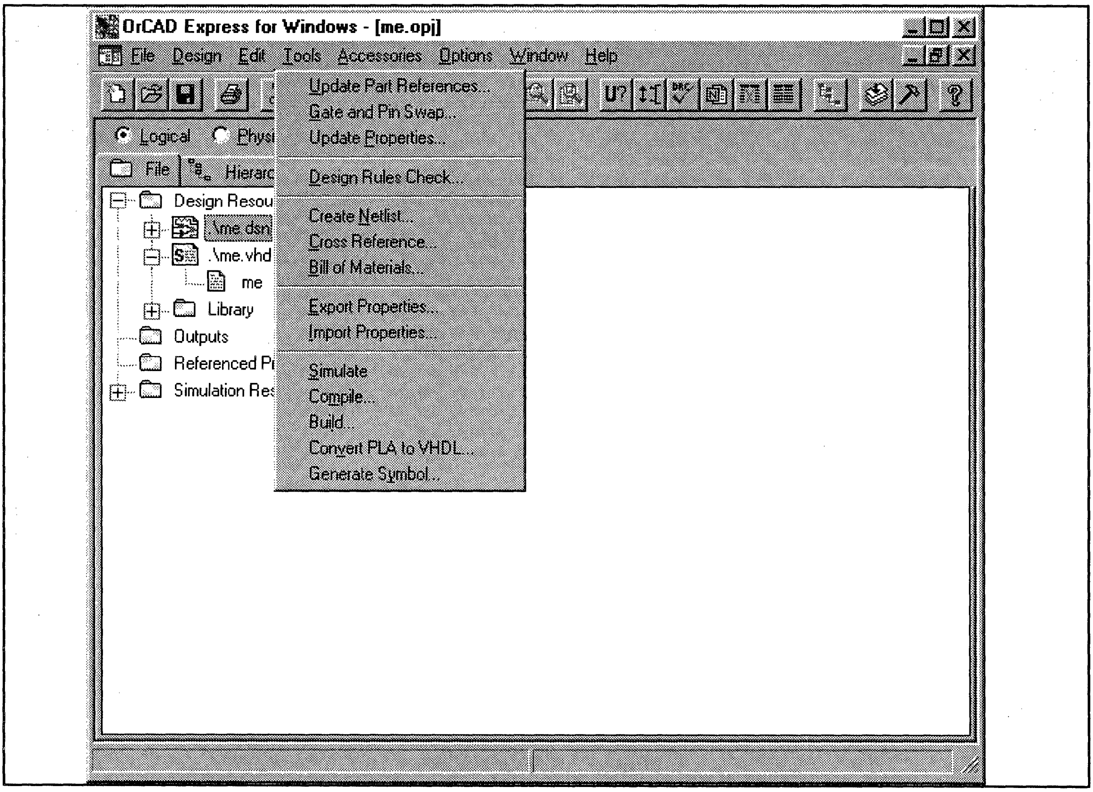
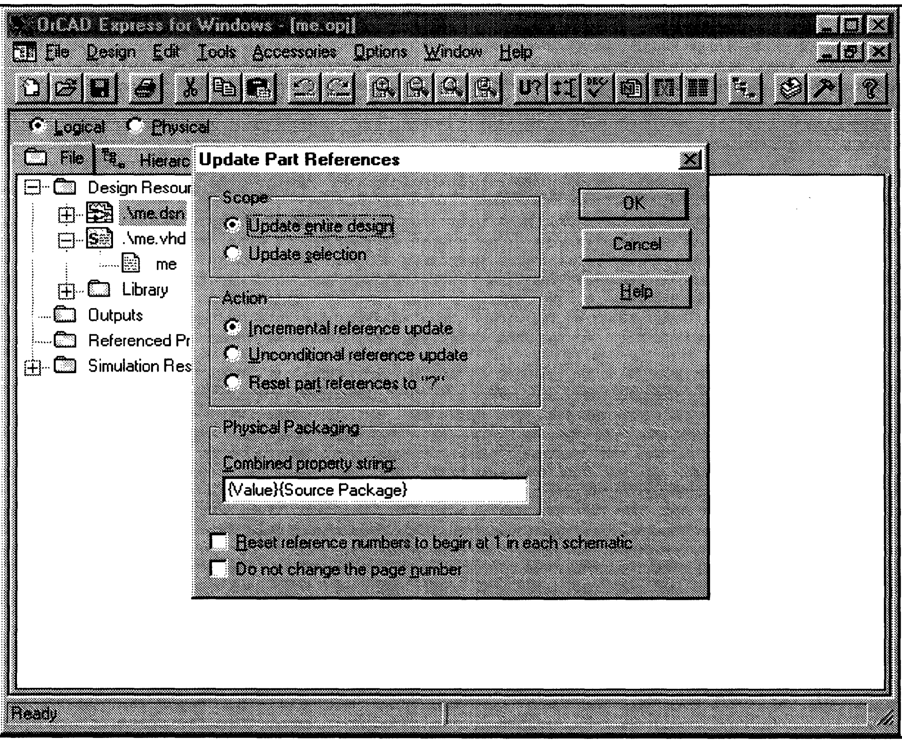


Figure 11. Processing the Design

At this point, invoke the pull down menu from Tools. Select Update Part References.



Update Part References

Scope

● Update entire design

○ Update selection

Action

● Incremental reference update

○ Unconditional reference update

○ Reset part references to "?"

Physical Packaging

Combined property string:
{Value}{Source Package}

□ Reset reference numbers to begin at 1 in each schematic

□ Do not change the page number

Figure 12. Updating Part References

Accept the defaults and click OK.

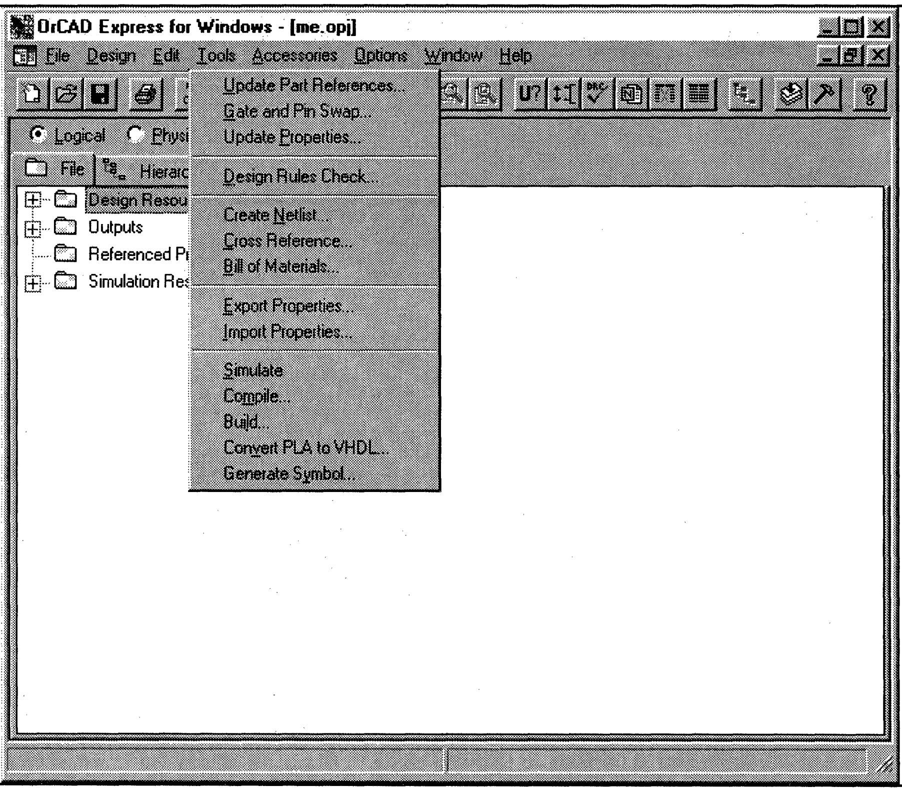
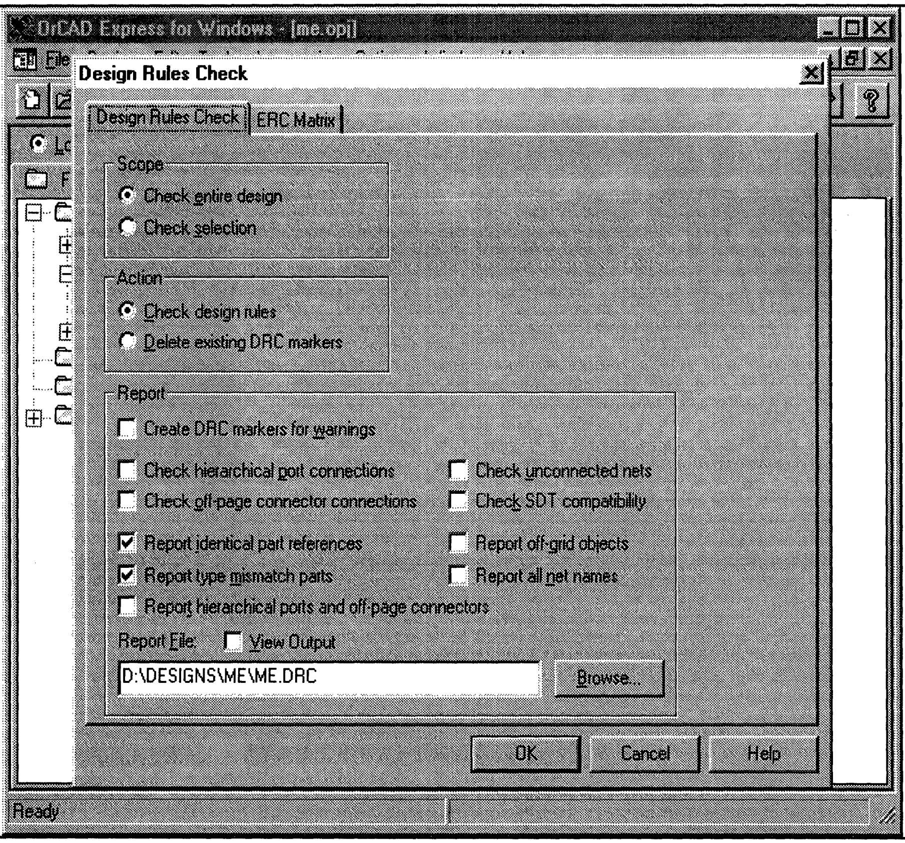


Figure 13. Invoking Design Rule Check

From the Tools Pull Down menu, select Design Rule Check.



Design Rules Check

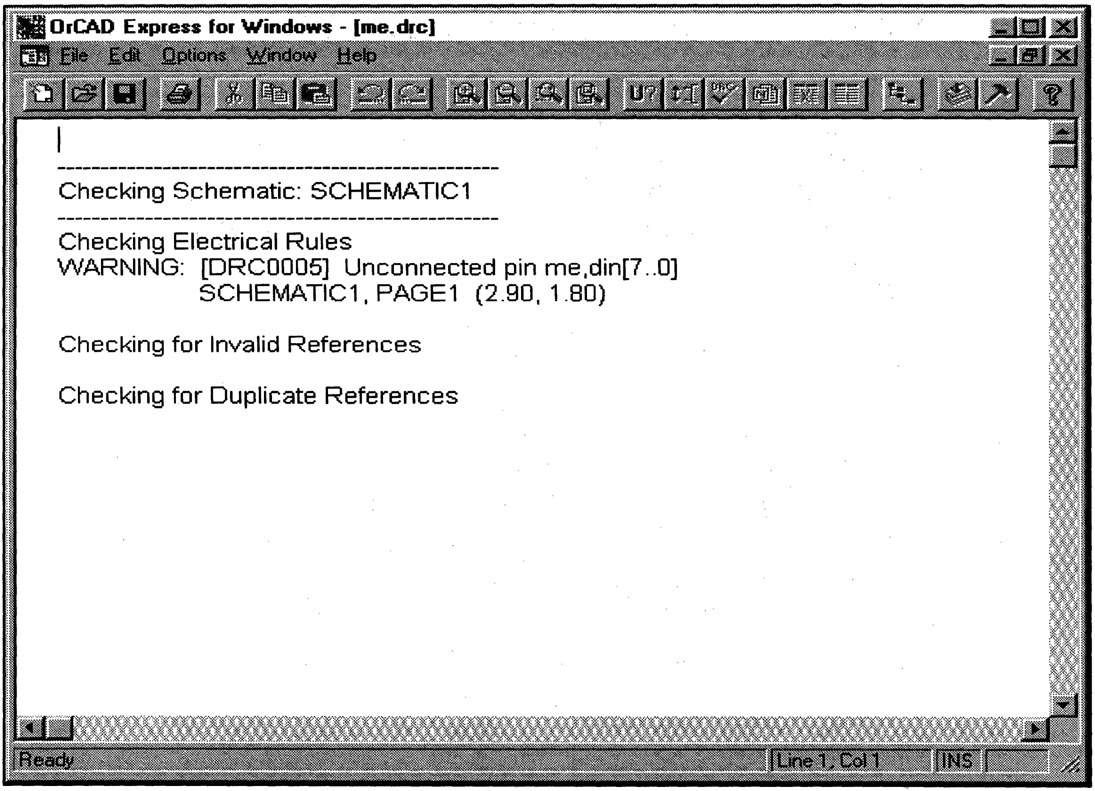
Scope
● Check entire design
○ Check selection

Action
● Check design rules
○ Delete existing DRC markers

Report
□ Create DRC markers for warnings
□ Check hierarchical port connections □ Check unconnected nets
□ Check off-page connector connections □ Check SDT compatibility
☑ Report identical part references □ Report off-grid objects
☑ Report type mismatch parts □ Report all net names
□ Report hierarchical ports and off-page connectors
Report File: □ View Output
D:\DESIGNS\ME\ME.DRC

Figure 14. Design Rule Check Menu

Click OK to invoke the design rule checker.


OrCAD Express for Windows - [me.drc]
File Edit Options Window Help
Ready
Line 1, Col 1
INS

```
----------------------------------------------
Checking Schematic: SCHEMATIC1
----------------------------------------------
Checking Electrical Rules
WARNING:  [DRC0005]  Unconnected pin me,din[7..0]
          SCHEMATIC1, PAGE1  (2.90, 1.80)

Checking for Invalid References

Checking for Duplicate References
```

Figure 15. Design Rule Error

In the schematic editor, Place Wire was used for connecting din[7:0].
Redo the schematic using Place Bus.

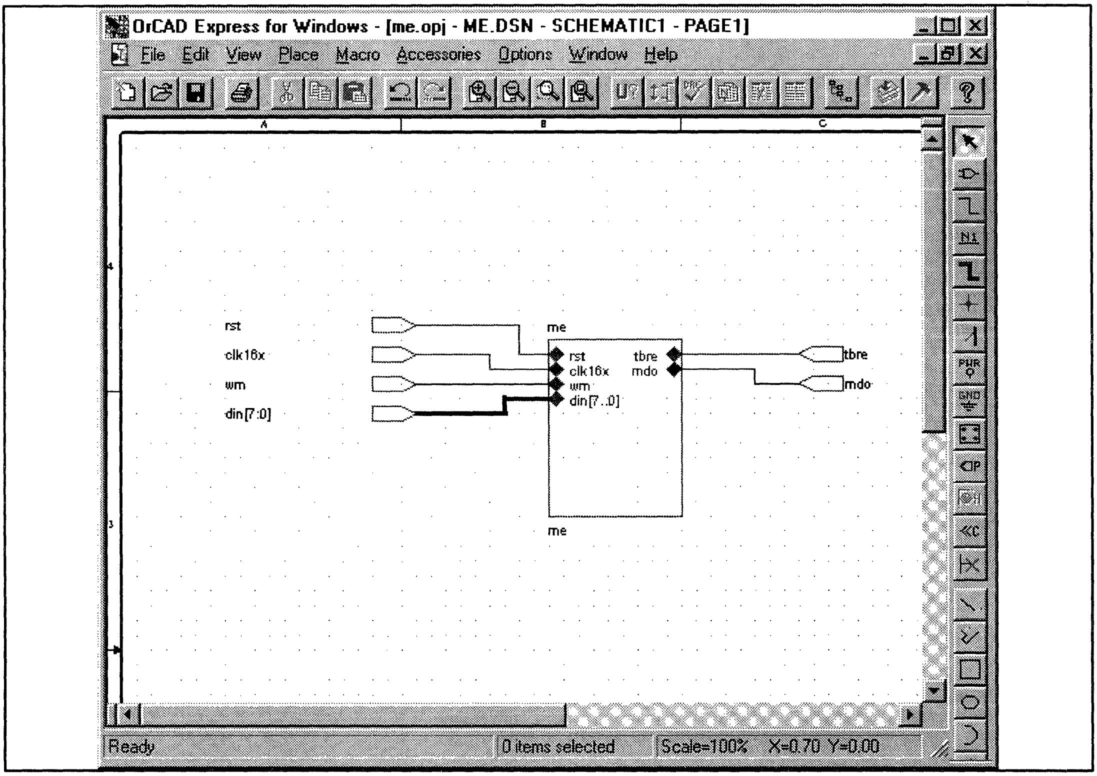
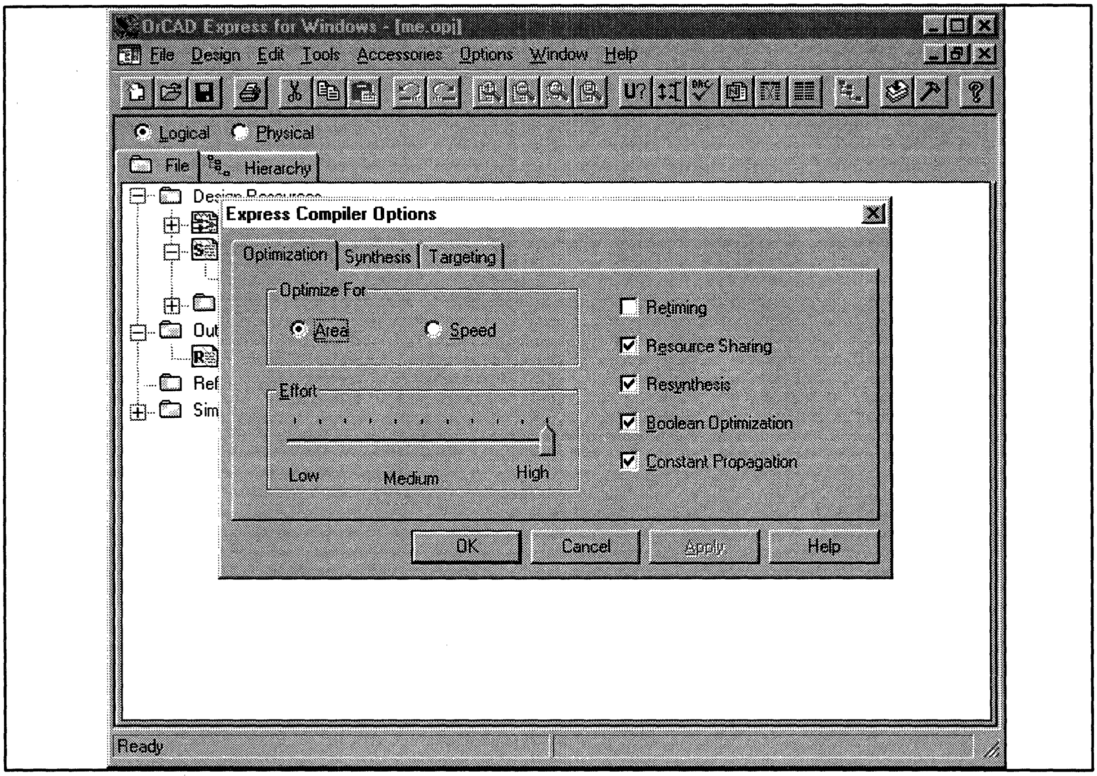

OrCAD Express for Windows - [me.opj - ME.DSN - SCHEMATIC1 - PAGE1]
File Edit View Place Macro Accessories Options Window Help
A
B
C
4
3
rst
clk16x
wm
din[7:0]
me
rst
clk16x
wm
din[7..0]
tbre
mdo
me
tbre
mdo
Ready
0 items selected
Scale=100% X=0.70 Y=0.00

Figure 16. Corrected Schematic

This schematic has no design rule violations.


OrCAD Express for Windows - [me.opj]
File Design Edit Tools Accessories Options Window Help
File
Hierarchy
Design Resources
Out
Ref
Sim
Ready

● Logical ○ Physical

Express Compiler Options

Optimization | Synthesis | Targeting

Optimize For

● Area ○ Speed

Effort

Low Medium High

□ Retiming

☑ Resource Sharing

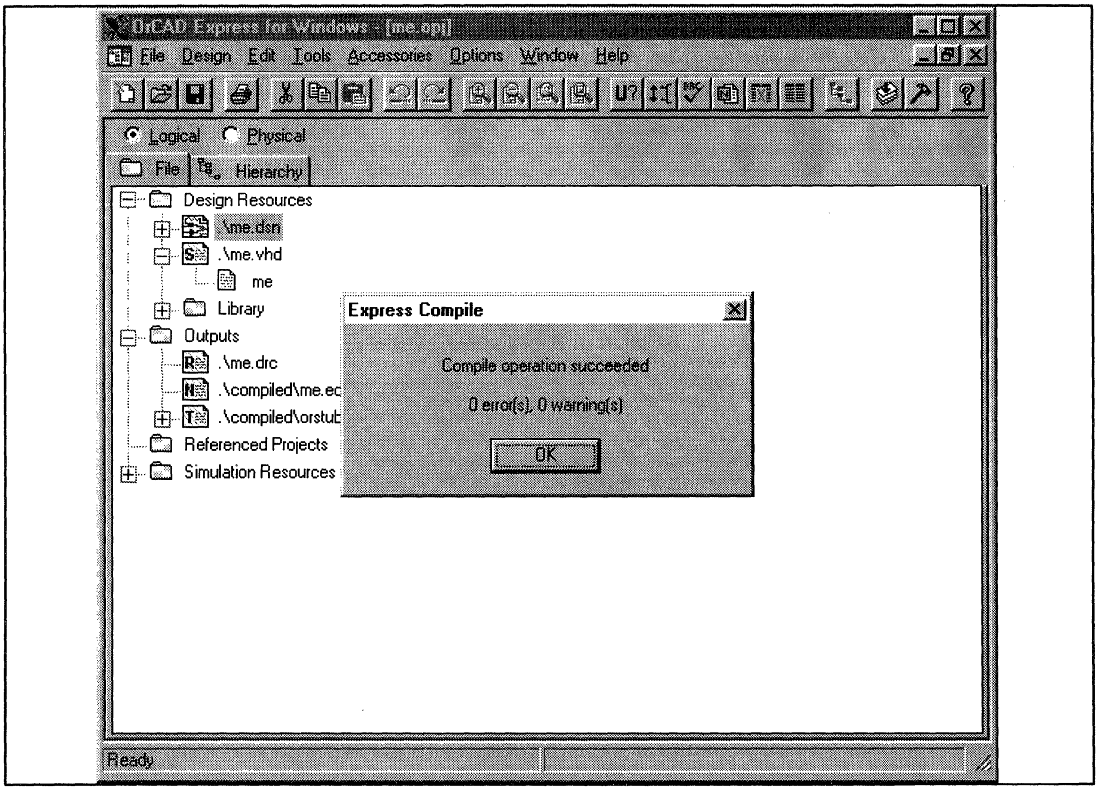
☑ Resynthesis

☑ Boolean Optimization

☑ Constant Propagation

Figure 17. Compilation

When the schematic is correct, select Compile from the Tools pull down menu. There are three folders – Optimization, Synthesis, and Targeting. Click OK to select the defaults.



Compile operation succeeded

0 error(s), 0 warning(s)

Figure 18. Successful Compilation

After a successful compilation, the me.edf file is included in the Outputs folder.

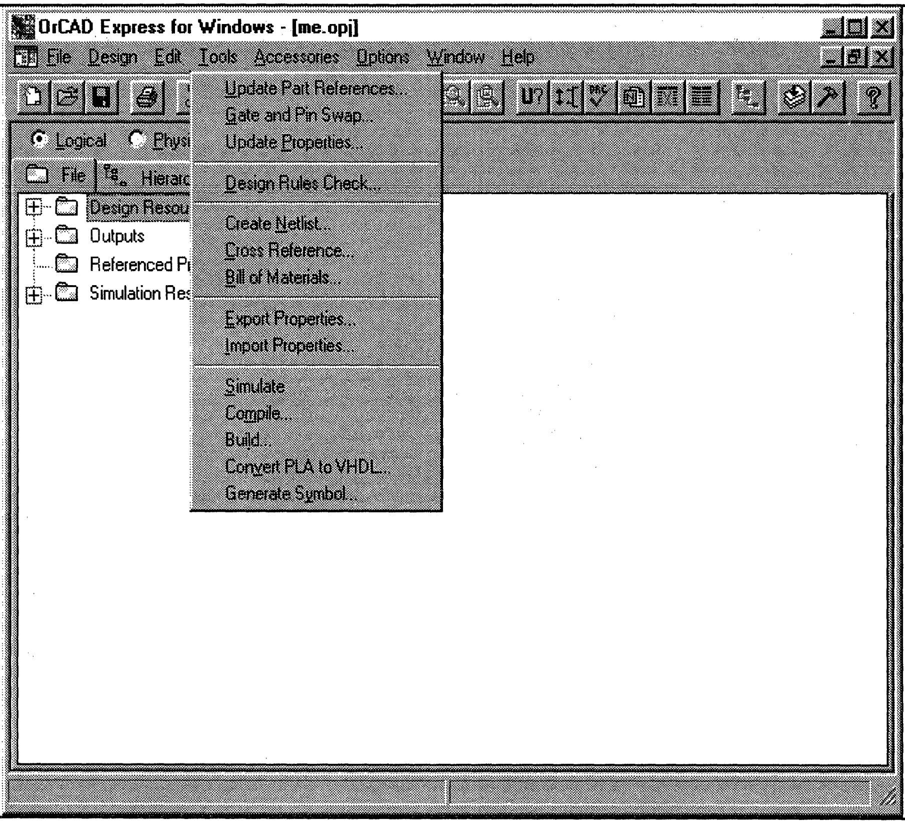
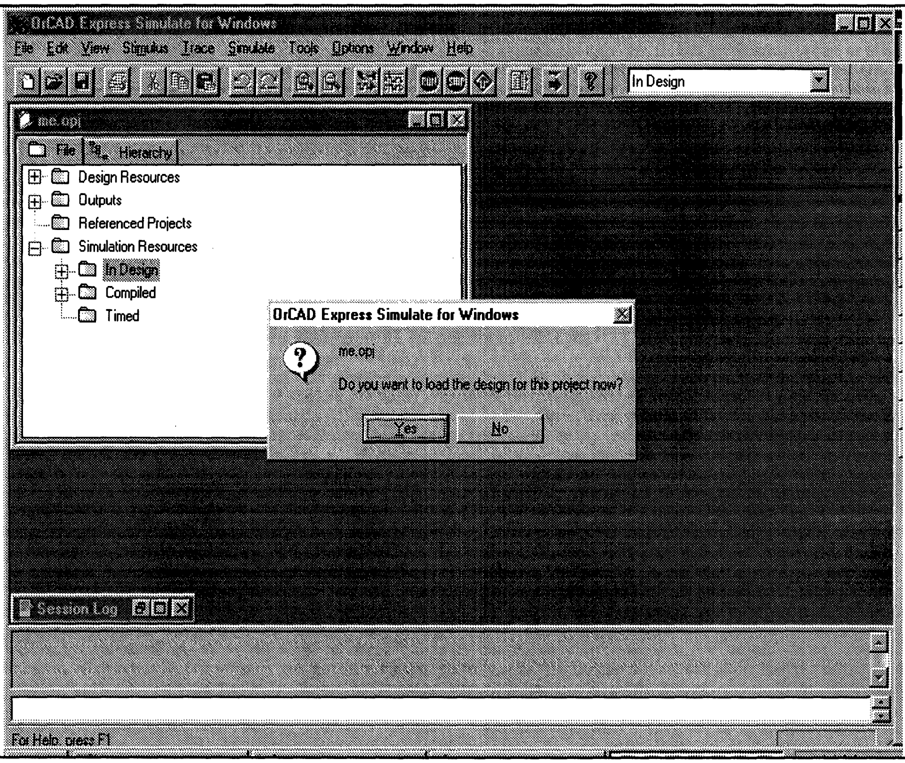


Figure 19. Invoking the Simulator

Select Simulate from the Tools pull down menu to invoke the simulator.



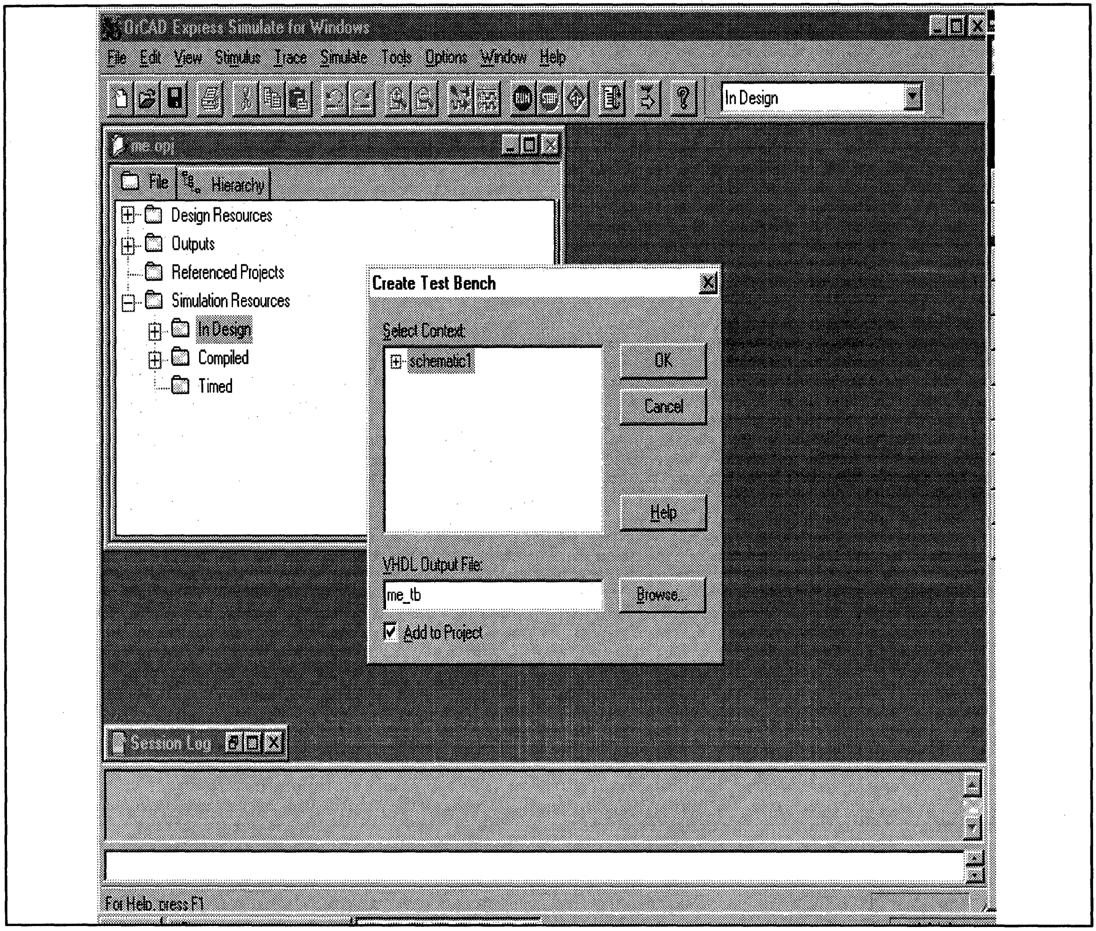
OrCAD Express Simulate for Windows

me.opj

Do you want to load the design for this project now?

Figure 20. Functional Simulation

OrCAD Express allows simulation of the source design, a compiled design, and/or back annotated timing simulation. The schematic is annotated with the simulation results and is viewable simultaneously with the simulation outputs. For this application note, a basic functional simulation is done.



Create Test Bench

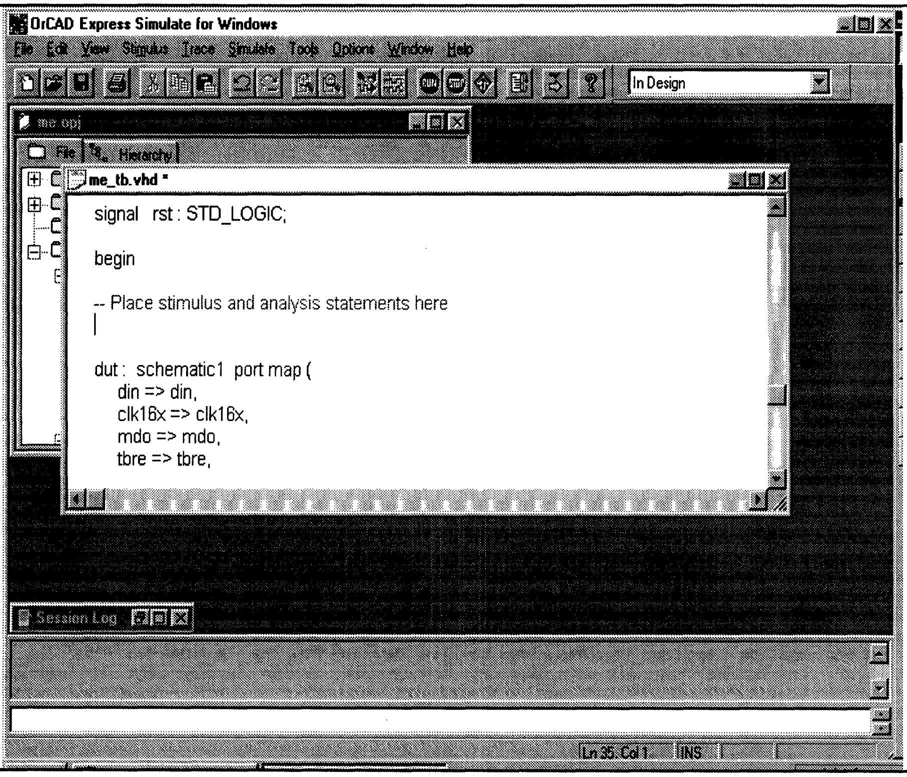
Select Context: schematic1

VHDL Output File: me_tb

☑ Add to Project

Figure 21. VHDL Test Bench Creation

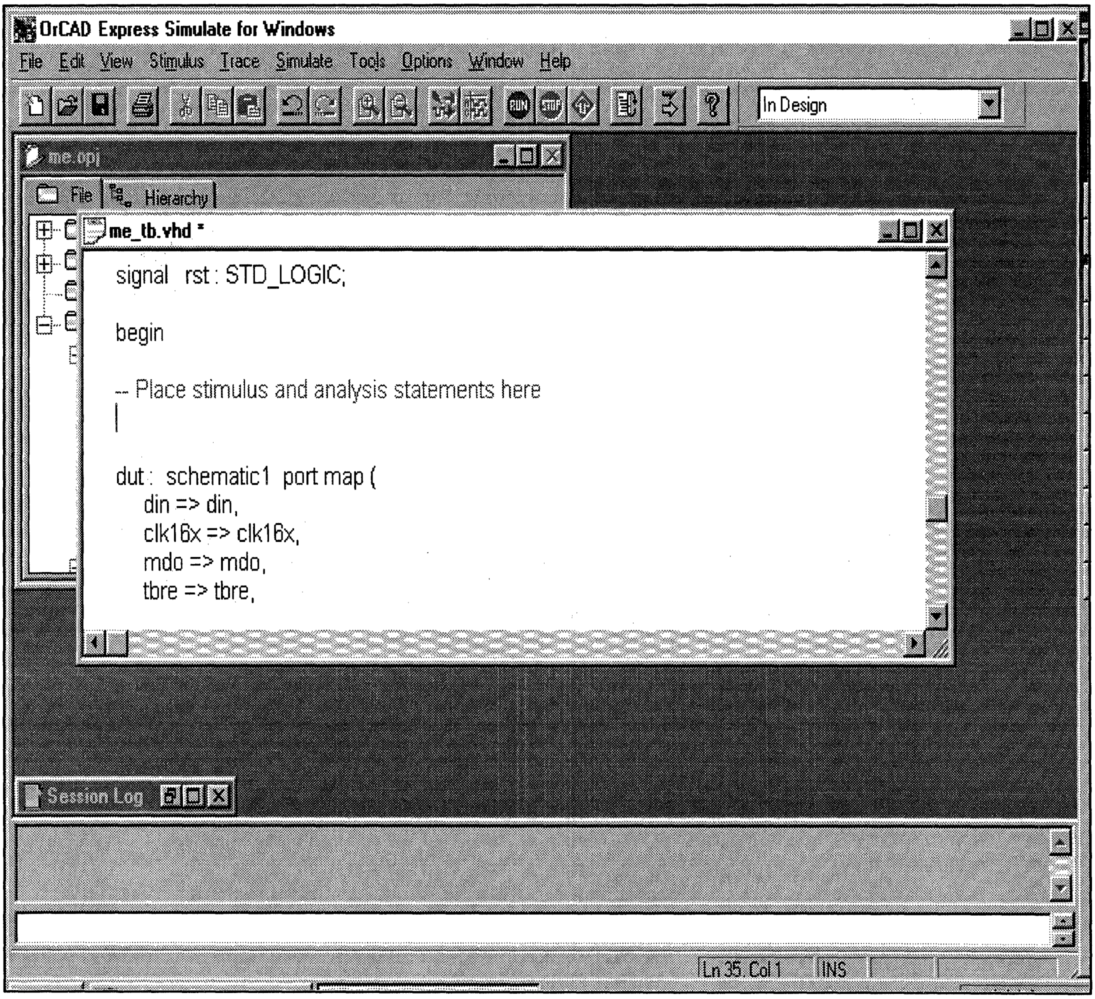
Select Stimulus – Create Testbench to generate a testbench template for the me design. Click OK in the dialog box.


OrCAD Express Simulate for Windows
File Edit View Stimulus Trace Simulate Tools Options Window Help
In Design
me.opj
File
Hierarchy
me_tb.vhd *
Session Log
Ln 35, Col 1
INS

```
signal   rst : STD_LOGIC;

begin

-- Place stimulus and analysis statements here


dut :  schematic1  port map (
    din => din,
    clk16x => clk16x,
    mdo => mdo,
    tbre => tbre,
```

Figure 22. Editing the Testbench

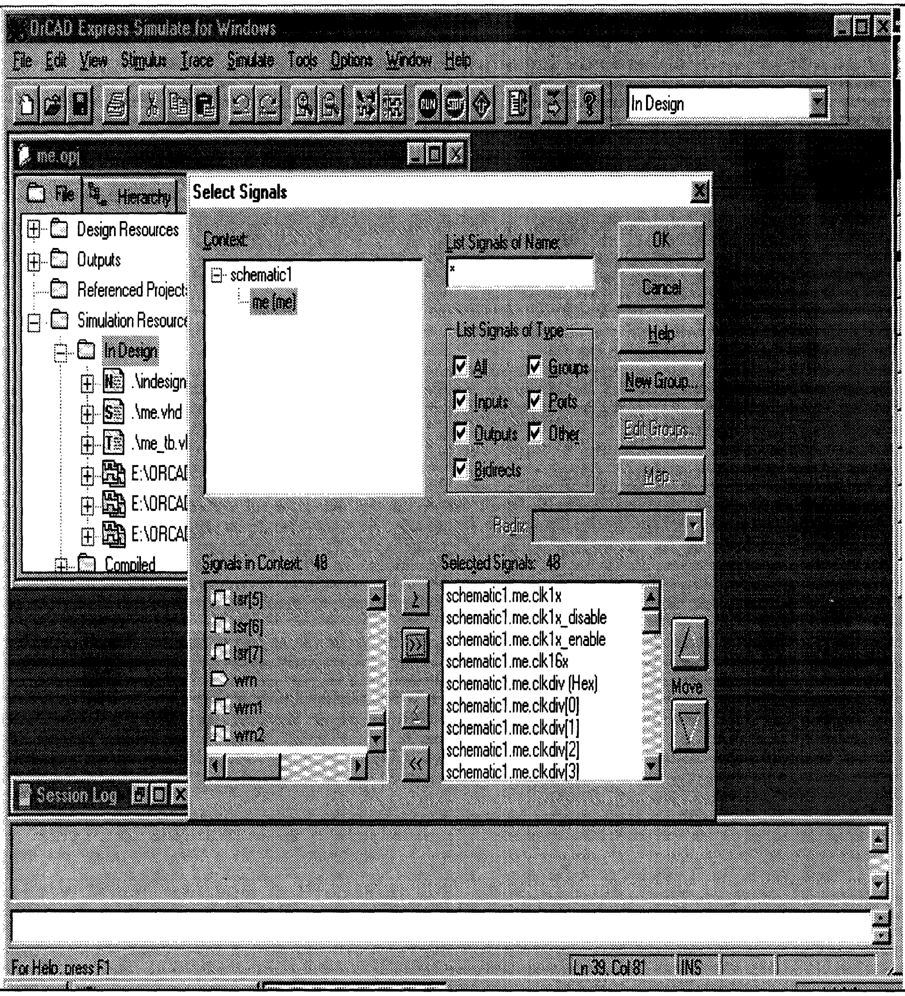
Move the cursor to the position in the testbench in which to place the stimuli.


OrCAD Express Simulate for Windows
File Edit View Stimulus Trace Simulate Tools Options Window Help
In Design
me.opj
File
Hierarchy
me_tb.vhd *
Session Log
Ln 35, Col 1
INS

```
signal   rst : STD_LOGIC;

begin

-- Place stimulus and analysis statements here


dut :  schematic1  port map (
    din => din,
    clk16x => clk16x,
    mdo => mdo,
    tbre => tbre,
```

Figure 23. me_tb.vhd

Enter the stimuli as shown.



Select Signals

Context:
- schematic1
  - me (me)

List Signals of Name: *

List Signals of Type:

☑ All ☑ Groups
☑ Inputs ☑ Ports
☑ Outputs ☑ Other
☑ Bidirects

Signals in Context: 48

tsr[5]
tsr[6]
tsr[7]
wrn
wrn1
wrn2

Selected Signals: 48

schematic1.me.clk1x
schematic1.me.clk1x_disable
schematic1.me.clk1x_enable
schematic1.me.clk16x
schematic1.me.clkdiv (Hex)
schematic1.me.clkdiv[0]
schematic1.me.clkdiv[1]
schematic1.me.clkdiv[2]
schematic1.me.clkdiv[3]

Figure 24. Adding Traces

Traces can be added to a wave and/or a list window.

Select me as the scope so that internal signals can be probed. Add signals to trace by highlighting signals in the Signals in Context and clicking on the arrow to copy them to the Selected Signals window.

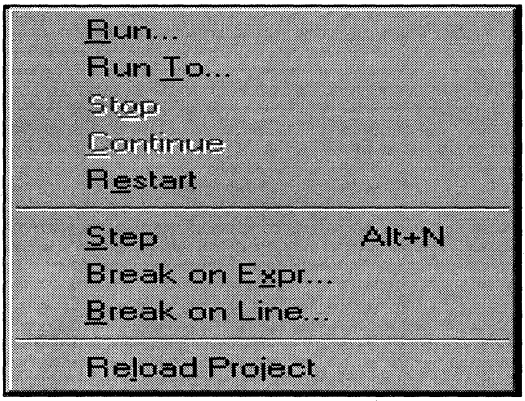
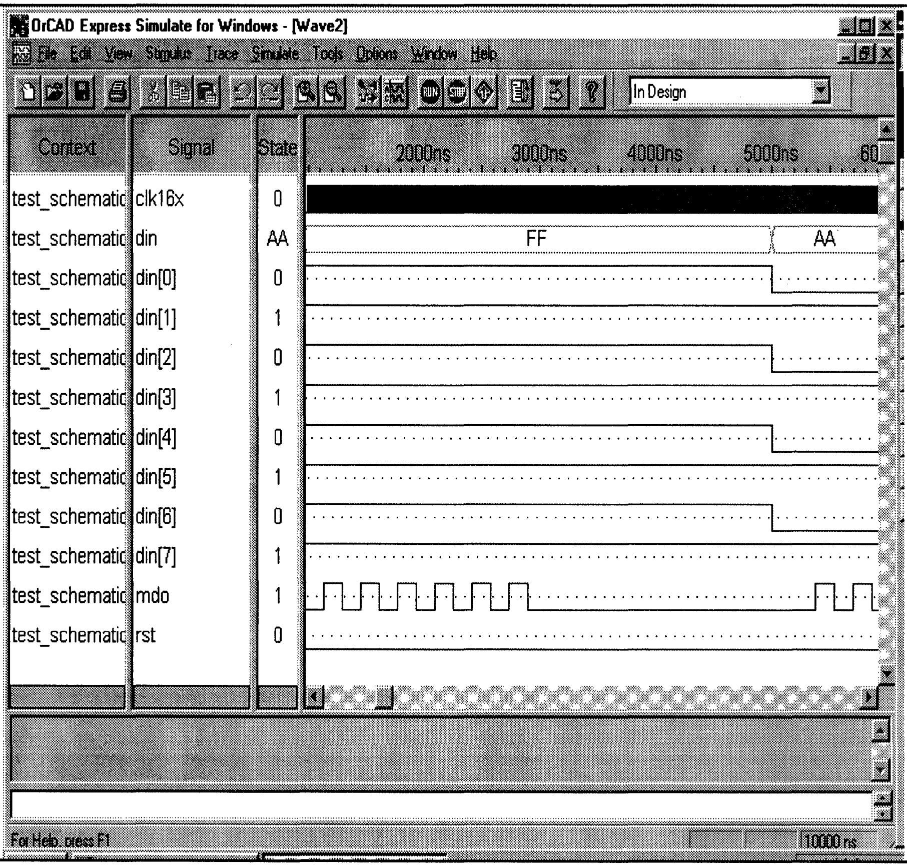


Figure 25. Running a Simulation

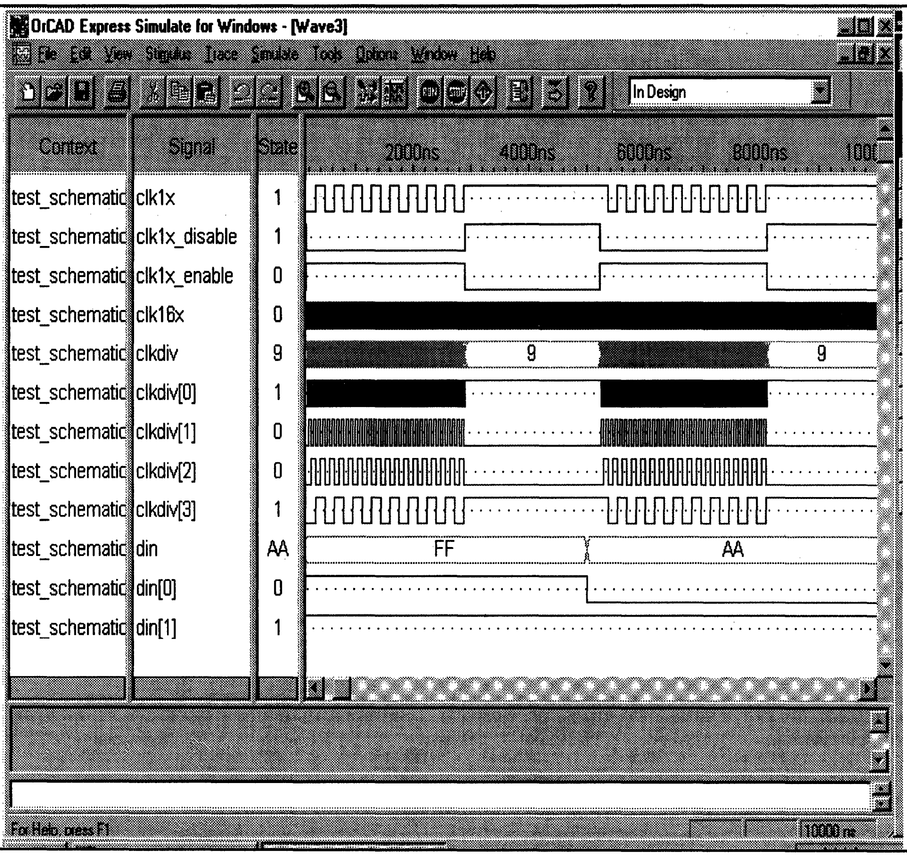
From the Simulate pull down, select Run To and enter 10000 ns in the dialog box.


OrCAD Express Simulate for Windows - [Wave2]
File Edit View Stimulus Trace Simulate Tools Options Window Help
In Design
2000ns
3000ns
4000ns
5000ns
60
FF
AA
For Help, press F1
10000 ns

Context	Signal	State
test_schematic	clk16x	0
test_schematic	din	AA
test_schematic	din[0]	0
test_schematic	din[1]	1
test_schematic	din[2]	0
test_schematic	din[3]	1
test_schematic	din[4]	0
test_schematic	din[5]	1
test_schematic	din[6]	0
test_schematic	din[7]	1
test_schematic	mdo	1
test_schematic	rst	0

Figure 26. Simulation Wave Window

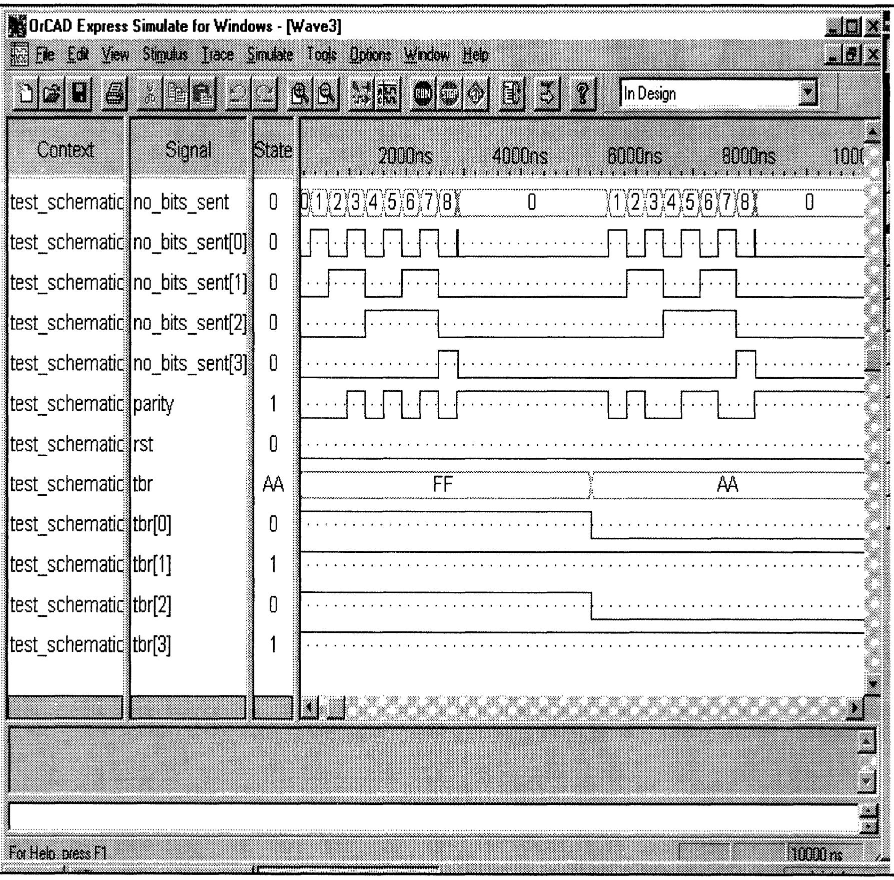
This shows signals from the me external ports.


OrCAD Express Simulate for Windows - [Wave3]
File Edit View Stimulus Trace Simulate Tools Options Window Help
In Design
2000ns
4000ns
6000ns
8000ns
100(
9
9
FF
AA
For Help, press F1
10000 ns

Context	Signal	State
test_schematic	clk1x	1
test_schematic	clk1x_disable	1
test_schematic	clk1x_enable	0
test_schematic	clk16x	0
test_schematic	clkdiv	9
test_schematic	clkdiv[0]	1
test_schematic	clkdiv[1]	0
test_schematic	clkdiv[2]	0
test_schematic	clkdiv[3]	1
test_schematic	din	AA
test_schematic	din[0]	0
test_schematic	din[1]	1

Figure 27. Displaying Clock Divider Signals

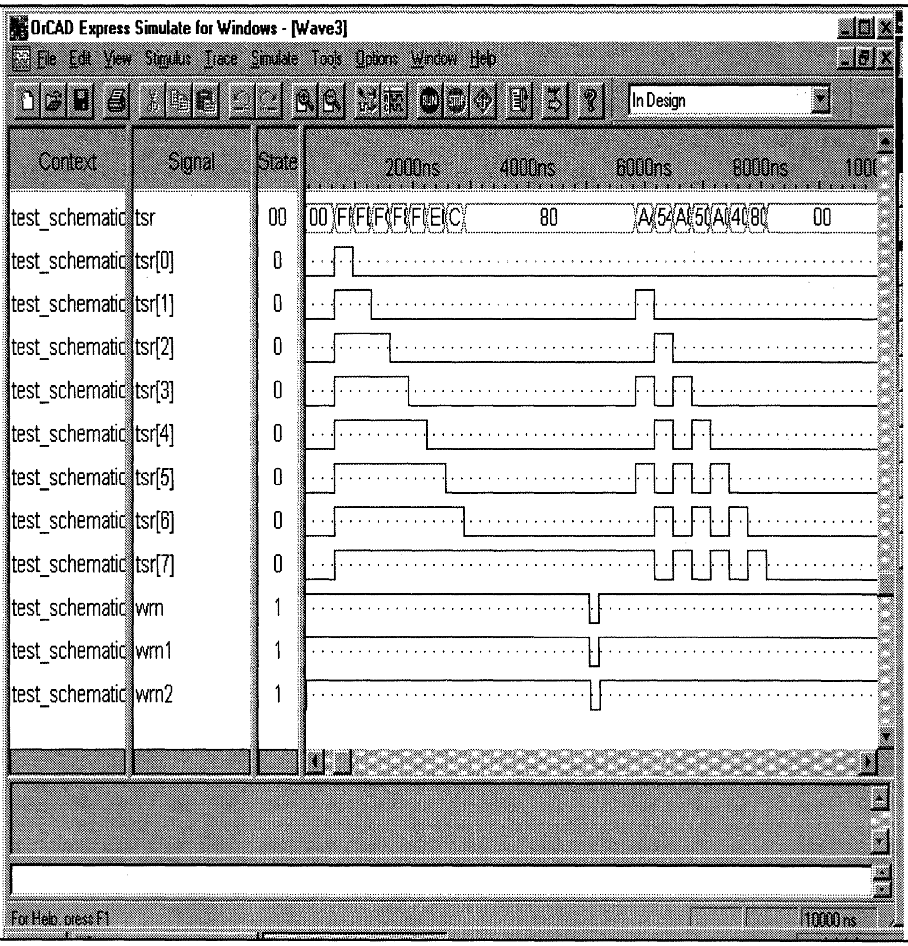
Internal nodes are shown. Since the me design funcions similar to a UART, the internal clk1x is diabled to minimize power consumption when a word is not transmitted.


OrCAD Express Simulate for Windows - [Wave3]
File Edit View Stimulus Trace Simulate Tools Options Window Help
In Design
2000ns
4000ns
6000ns
8000ns
100(
0 1 2 3 4 5 6 7 8 0 1 2 3 4 5 6 7 8 0
FF
AA
For Help, press F1
10000 ns

Context	Signal	State
test_schematic	no_bits_sent	0
test_schematic	no_bits_sent[0]	0
test_schematic	no_bits_sent[1]	0
test_schematic	no_bits_sent[2]	0
test_schematic	no_bits_sent[3]	0
test_schematic	parity	1
test_schematic	rst	0
test_schematic	tbr	AA
test_schematic	tbr[0]	0
test_schematic	tbr[1]	1
test_schematic	tbr[2]	0
test_schematic	tbr[3]	1

Figure 28. Simulation Waveform Viewer

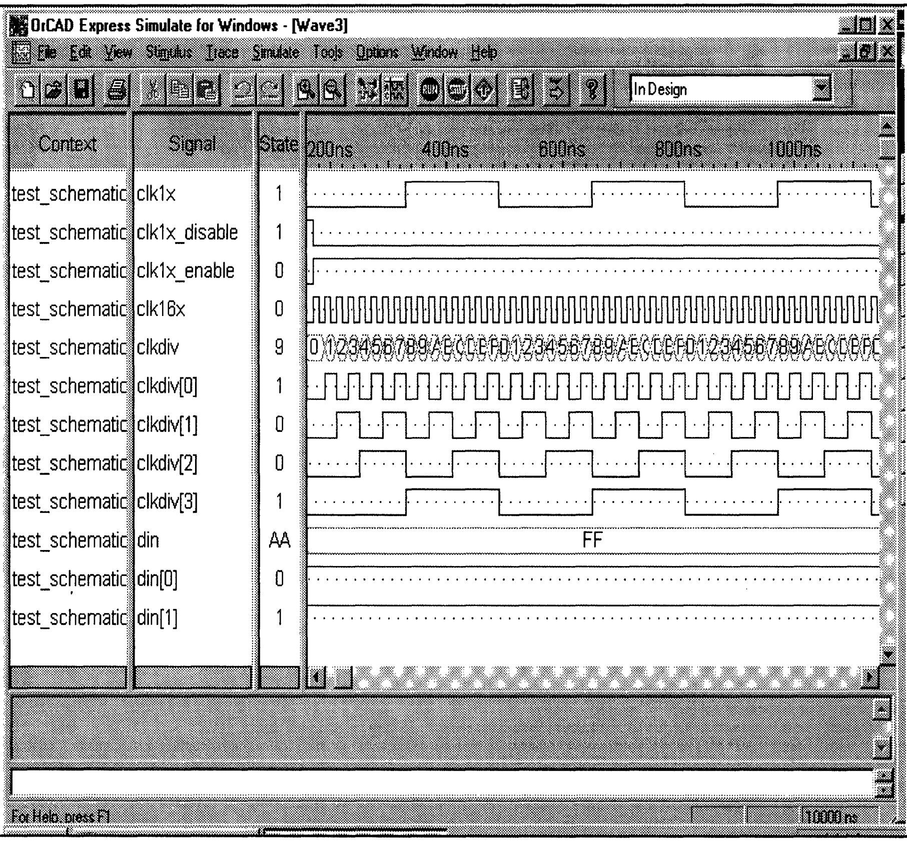
This shows the no_bits_sent process used to sequence the start bits, data, etc, and to control the word size.


OrCAD Express Simulate for Windows - [Wave3]
File Edit View Stimulus Trace Simulate Tools Options Window Help
In Design
2000ns
4000ns
6000ns
8000ns
100(
00
F(
F(
F(
F(
F(
E(
C(
80
A(
5(
A(
5(
A(
4(
8(
00
For Help, press F1
10000 ns

Context	Signal	State
test_schematic	tsr	00
test_schematic	tsr[0]	0
test_schematic	tsr[1]	0
test_schematic	tsr[2]	0
test_schematic	tsr[3]	0
test_schematic	tsr[4]	0
test_schematic	tsr[5]	0
test_schematic	tsr[6]	0
test_schematic	tsr[7]	0
test_schematic	wrn	1
test_schematic	wrn1	1
test_schematic	wrn2	1

Figure 29. Transmitter Shift Register

This shows the write signal and the operation of the transmitter shift register.


OrCAD Express Simulate for Windows - [Wave3]
File Edit View Stimulus Trace Simulate Tools Options Window Help
In Design
200ns
400ns
600ns
800ns
1000ns
0 1 2 3 4 5 6 7 8 9 A B C D E F 0 1 2 3 4 5 6 7 8 9 A B C D E F 0 1 2 3 4 5 6 7 8 9 A B C D E F
FF
For Help, press F1
10000 ns

Context	Signal	State
test_schematic	clk1x	1
test_schematic	clk1x_disable	1
test_schematic	clk1x_enable	0
test_schematic	clk16x	0
test_schematic	clkdiv	9
test_schematic	clkdiv[0]	1
test_schematic	clkdiv[1]	0
test_schematic	clkdiv[2]	0
test_schematic	clkdiv[3]	1
test_schematic	din	AA
test_schematic	din[0]	0
test_schematic	din[1]	1

Figure 30. Clock Divider

This zooms into the clock divider.

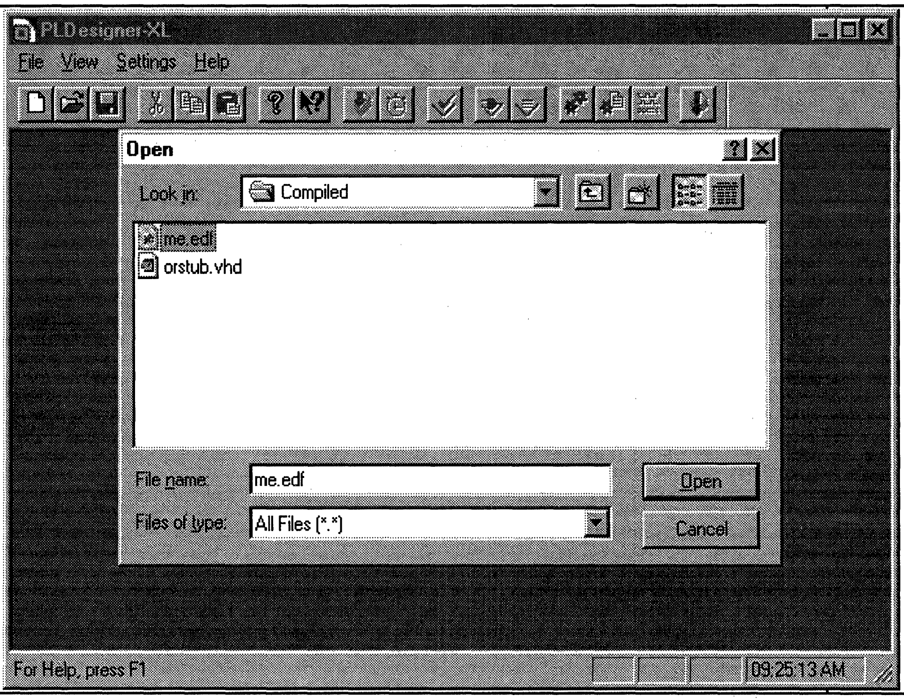
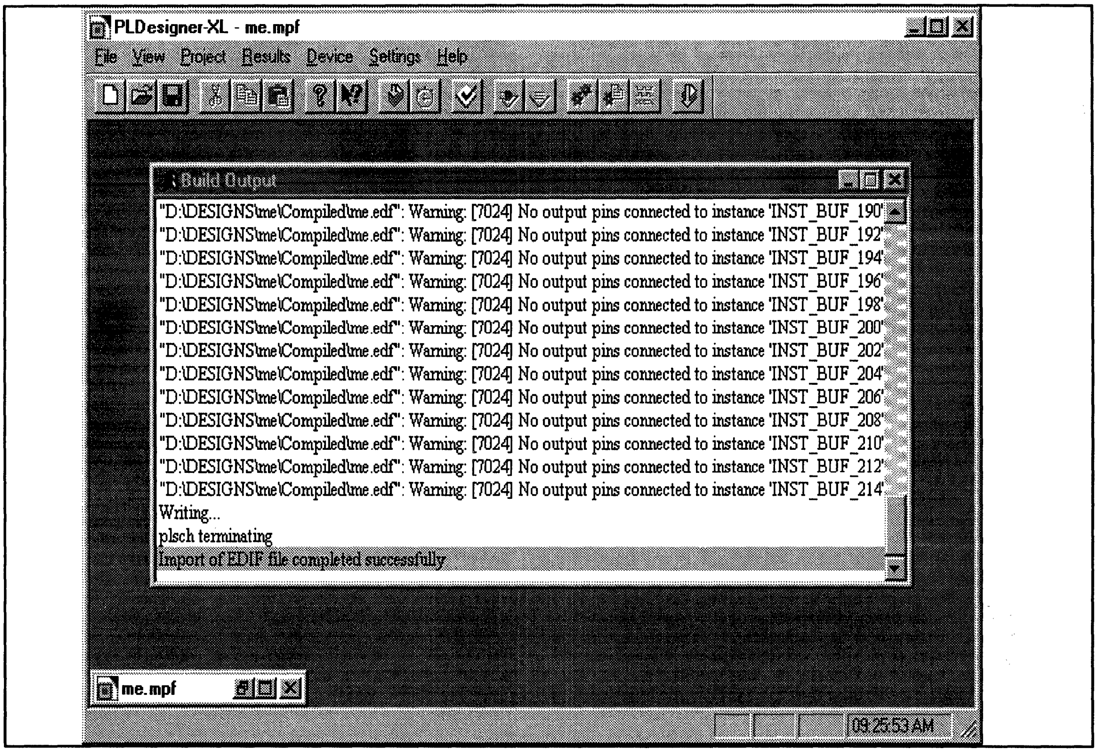


Figure 31. Importing the Design Into PLDesigner–XL

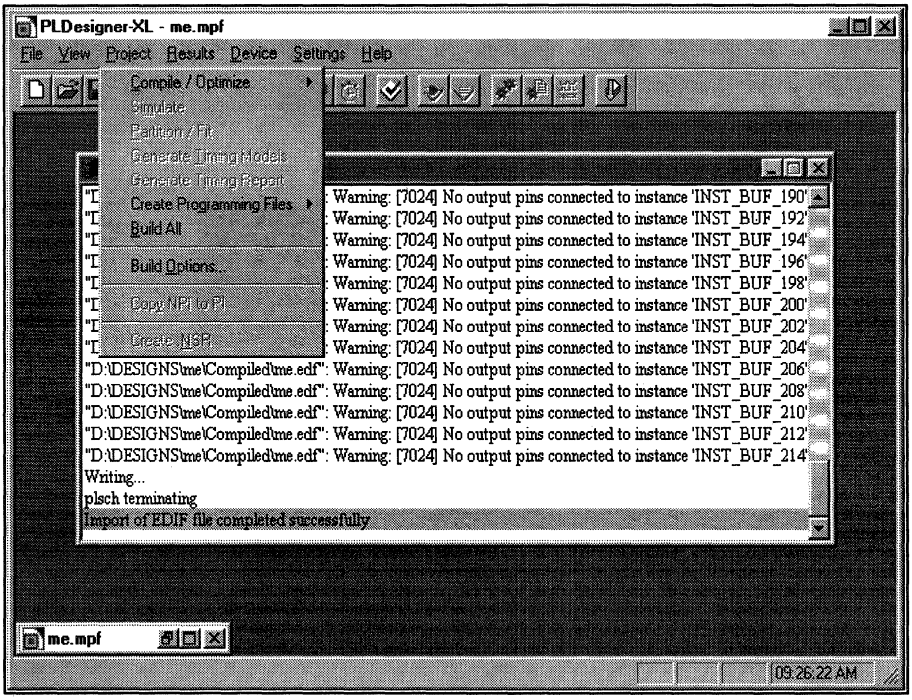
Once simulation is complete, save the waveforms and exit OrCAD Express. Double click on the PLDesigner–XL icon, select File Open, and Open when the dialog box appears.



```
"D:\DESIGNS\me\Compiled\me.edf": Warning: [7024] No output pins connected to instance 'INST_BUF_190'
"D:\DESIGNS\me\Compiled\me.edf": Warning: [7024] No output pins connected to instance 'INST_BUF_192'
"D:\DESIGNS\me\Compiled\me.edf": Warning: [7024] No output pins connected to instance 'INST_BUF_194'
"D:\DESIGNS\me\Compiled\me.edf": Warning: [7024] No output pins connected to instance 'INST_BUF_196'
"D:\DESIGNS\me\Compiled\me.edf": Warning: [7024] No output pins connected to instance 'INST_BUF_198'
"D:\DESIGNS\me\Compiled\me.edf": Warning: [7024] No output pins connected to instance 'INST_BUF_200'
"D:\DESIGNS\me\Compiled\me.edf": Warning: [7024] No output pins connected to instance 'INST_BUF_202'
"D:\DESIGNS\me\Compiled\me.edf": Warning: [7024] No output pins connected to instance 'INST_BUF_204'
"D:\DESIGNS\me\Compiled\me.edf": Warning: [7024] No output pins connected to instance 'INST_BUF_206'
"D:\DESIGNS\me\Compiled\me.edf": Warning: [7024] No output pins connected to instance 'INST_BUF_208'
"D:\DESIGNS\me\Compiled\me.edf": Warning: [7024] No output pins connected to instance 'INST_BUF_210'
"D:\DESIGNS\me\Compiled\me.edf": Warning: [7024] No output pins connected to instance 'INST_BUF_212'
"D:\DESIGNS\me\Compiled\me.edf": Warning: [7024] No output pins connected to instance 'INST_BUF_214'
Writing...
plsch terminating
Import of EDIF file completed successfully
```

Figure 32. Edif Import

At this stage, the me.edf file from OrCAD Express and the me.src file generated by PLDesigner–XL are viewable in PLDesigner's text editor.

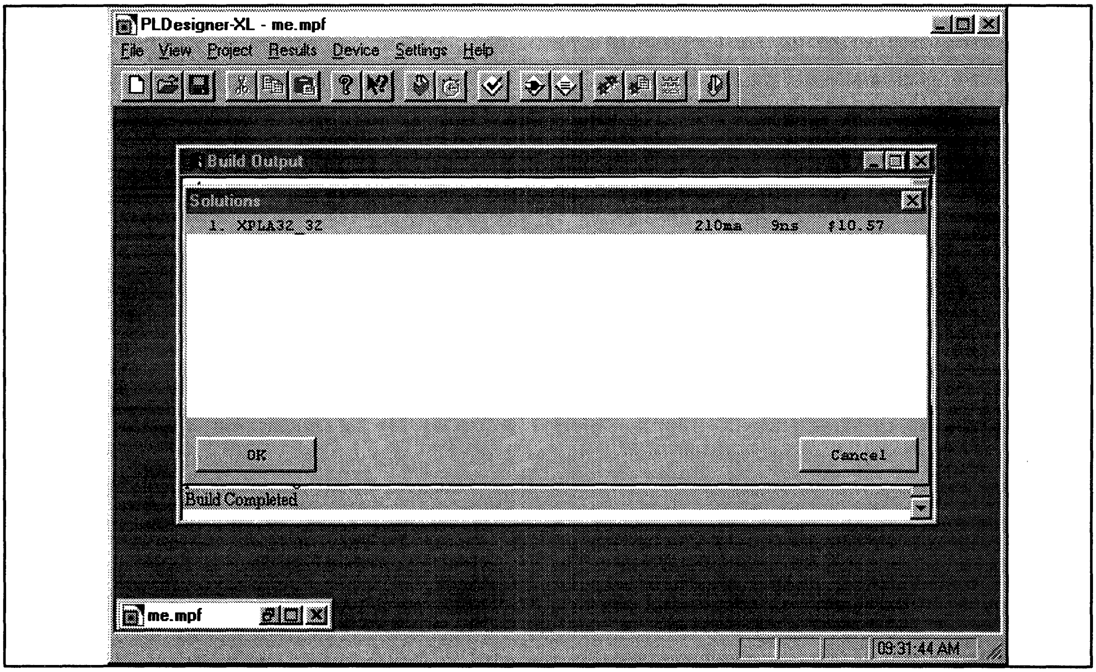


```
"I                                : Warning: [7024] No output pins connected to instance 'INST_BUF_190'
"I                                : Warning: [7024] No output pins connected to instance 'INST_BUF_192'
"I                                : Warning: [7024] No output pins connected to instance 'INST_BUF_194'
"I                                : Warning: [7024] No output pins connected to instance 'INST_BUF_196'
"I                                : Warning: [7024] No output pins connected to instance 'INST_BUF_198'
"I                                : Warning: [7024] No output pins connected to instance 'INST_BUF_200'
"I                                : Warning: [7024] No output pins connected to instance 'INST_BUF_202'
"I                                : Warning: [7024] No output pins connected to instance 'INST_BUF_204'
"D:\DESIGNS\me\Compiled\me.edf": Warning: [7024] No output pins connected to instance 'INST_BUF_206'
"D:\DESIGNS\me\Compiled\me.edf": Warning: [7024] No output pins connected to instance 'INST_BUF_208'
"D:\DESIGNS\me\Compiled\me.edf": Warning: [7024] No output pins connected to instance 'INST_BUF_210'
"D:\DESIGNS\me\Compiled\me.edf": Warning: [7024] No output pins connected to instance 'INST_BUF_212'
"D:\DESIGNS\me\Compiled\me.edf": Warning: [7024] No output pins connected to instance 'INST_BUF_214'
Writing...
plsch terminating
Import of EDIF file completed successfully
```

Figure 33. Compile Operation

In the project directory, add the me.cst and me.pi files to target the Philips PZ3032–06A44 device.

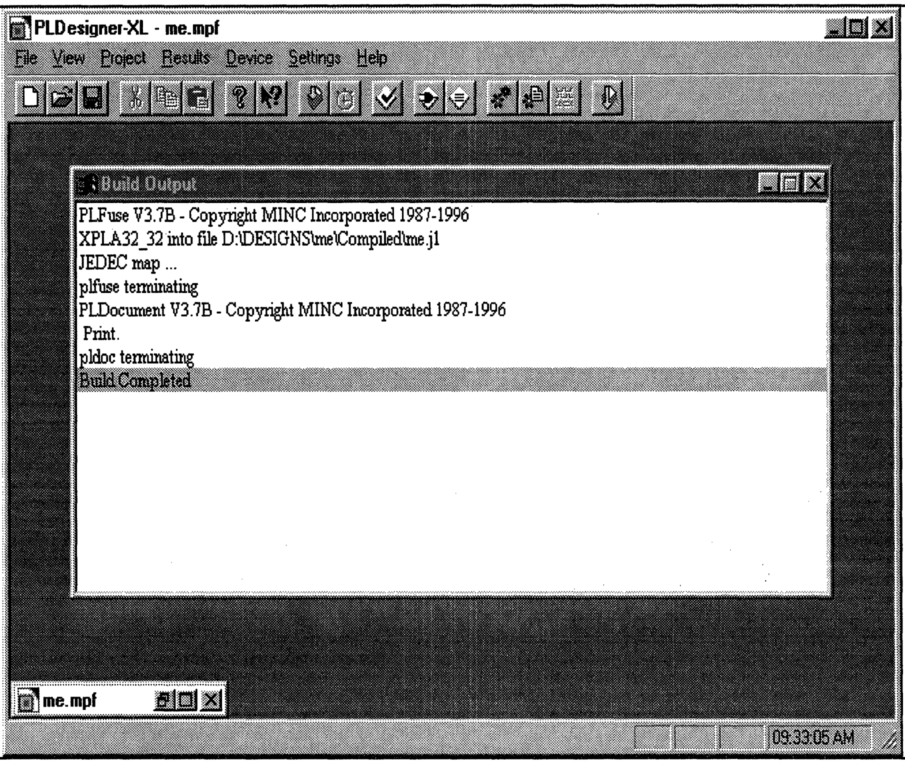
Select the Project pull down menu and Compile/Optimize. Then select Partition/Fit. As mentioned earlier, see the tool manufacturer's documentation for alternative methods of processing a design.



```
1. XPLA32_32                         210ma   9ns   $10.57
```

Figure 34. Solutions

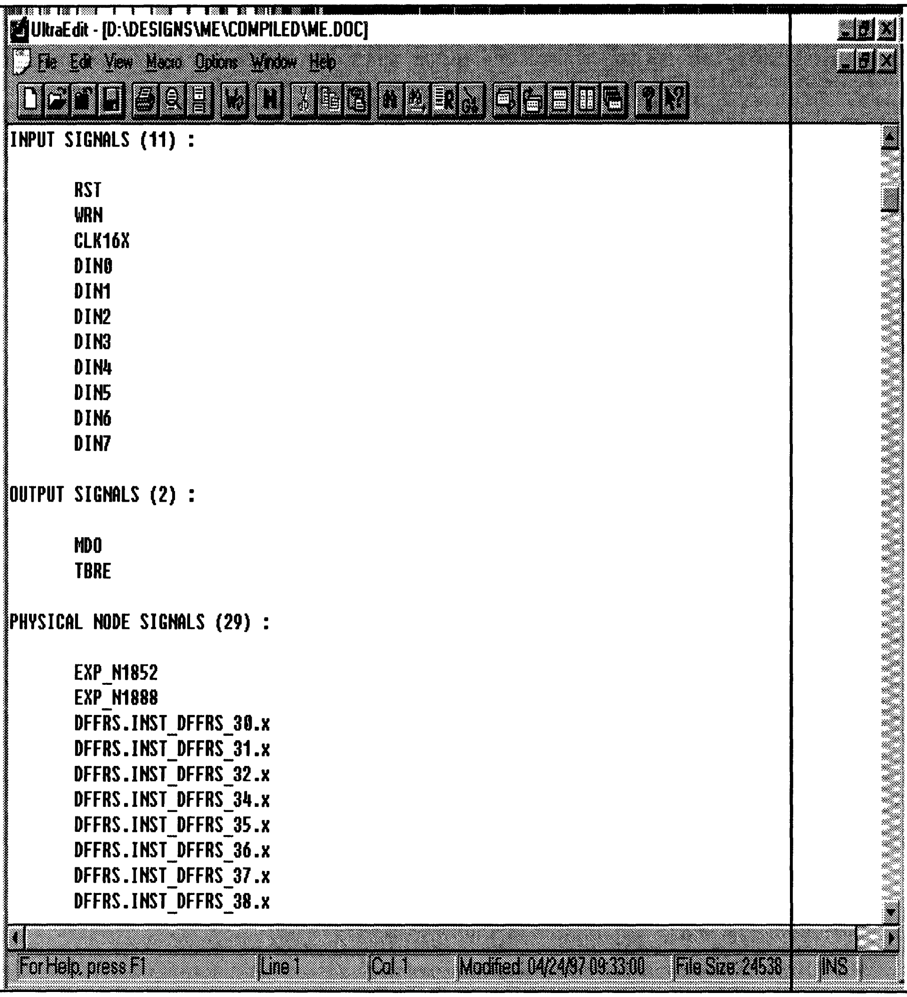
This shows the solutions provided by the fitter. If other devices were included in the cost and physical information files, theymay have been listed as a solution. The actual static IDD for a Philips PZ5032 is under 100 microamps.



```
PLFuse V3.7B - Copyright MINC Incorporated 1987-1996
XPLA32_32 into file D:\DESIGNS\me\Compiled\me.j1
JEDEC map ...
plfuse terminating
PLDocument V3.7B - Copyright MINC Incorporated 1987-1996
 Print.
pldoc terminating
Build Completed
```

Figure 35. Generate Fuse Map

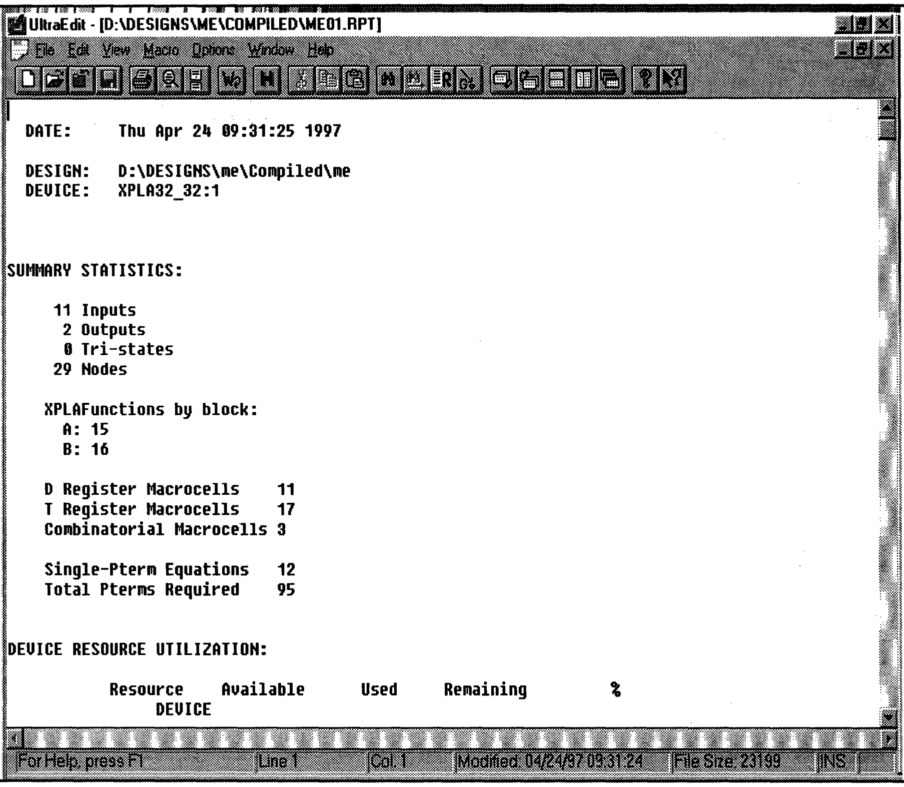
Select Project Generate Fuse Map to generate fuse map for this design.


UltraEdit - [D:\DESIGNS\ME\COMPILED\ME.DOC]
File Edit View Macro Options Window Help
For Help, press F1
Line 1
Col 1
Modified: 04/24/97 09:33:00
File Size: 24538
INS

```
INPUT SIGNALS (11) :

      RST
      WRN
      CLK16X
      DIN0
      DIN1
      DIN2
      DIN3
      DIN4
      DIN5
      DIN6
      DIN7

OUTPUT SIGNALS (2) :

      MDO
      TBRE

PHYSICAL NODE SIGNALS (29) :

      EXP_N1852
      EXP_N1888
      DFFRS.INST_DFFRS_30.x
      DFFRS.INST_DFFRS_31.x
      DFFRS.INST_DFFRS_32.x
      DFFRS.INST_DFFRS_34.x
      DFFRS.INST_DFFRS_35.x
      DFFRS.INST_DFFRS_36.x
      DFFRS.INST_DFFRS_37.x
      DFFRS.INST_DFFRS_38.x
```

Figure 36. me.doc

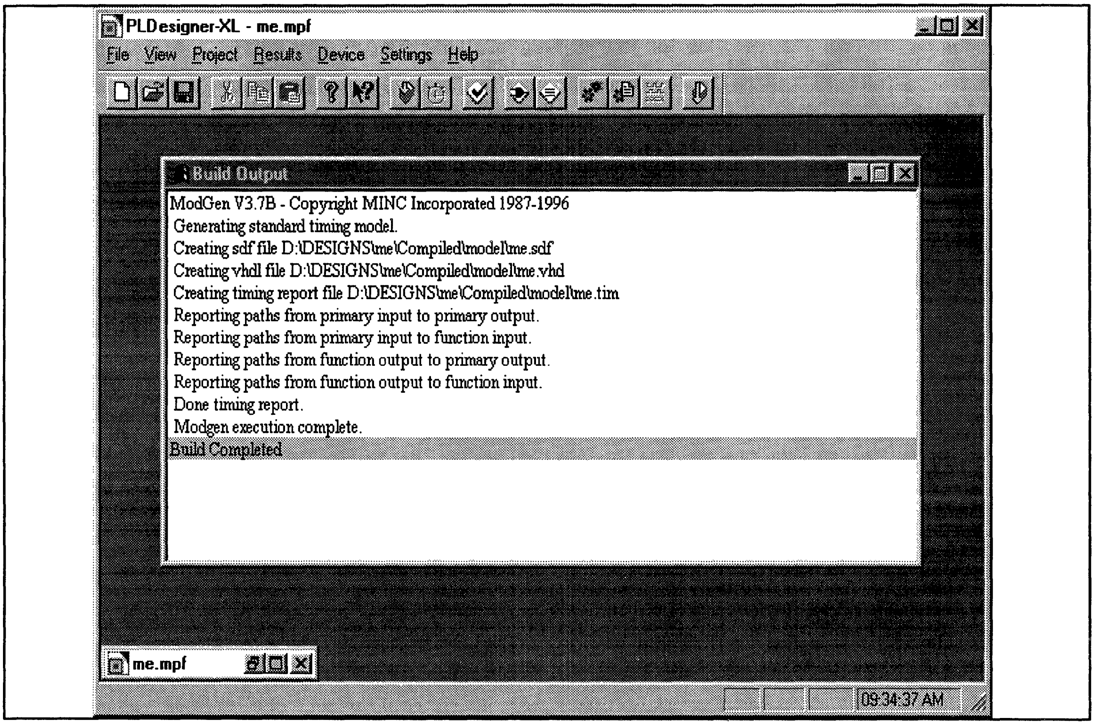
Select Results View doc. This shows the first few lines of the documentation for the me design.


UltraEdit - [D:\DESIGNS\ME\COMPILED\ME01.RPT]
File Edit View Macro Options Window Help
For Help, press F1
Line 1
Col. 1
Modified: 04/24/97 09:31:24
File Size: 23199
INS

```
 DATE:     Thu Apr 24 09:31:25 1997

 DESIGN:   D:\DESIGNS\me\Compiled\me
 DEVICE:   XPLA32_32:1



SUMMARY STATISTICS:

    11 Inputs
     2 Outputs
     0 Tri-states
    29 Nodes

   XPLAFunctions by block:
     A: 15
     B: 16

   D Register Macrocells    11
   T Register Macrocells    17
   Combinatorial Macrocells 3

   Single-Pterm Equations   12
   Total Pterms Required    95


DEVICE RESOURCE UTILIZATION:

          Resource    Available      Used     Remaining        %
               DEVICE
```

Figure 37. me01.rpt File

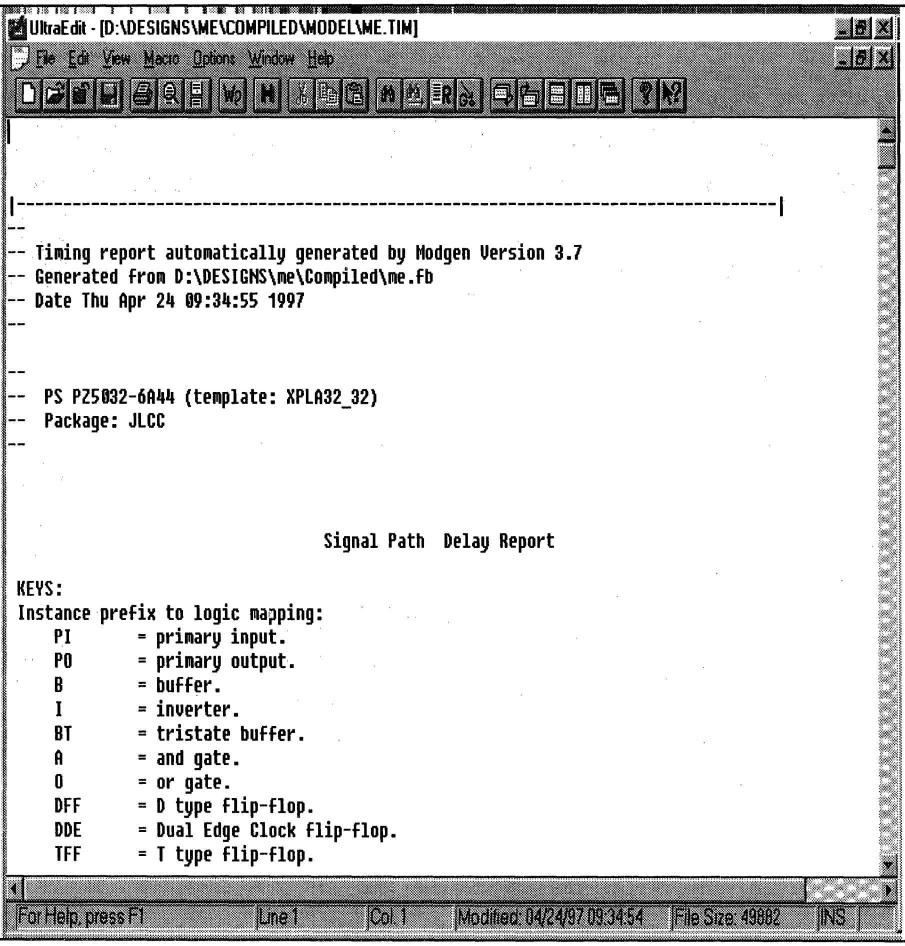
Select Results View rpt file. To review the report file. This shows the detail of the fitting process. A report file is provided if a single device is targeted in the pi file.



```
ModGen V3.7B - Copyright MINC Incorporated 1987-1996
 Generating standard timing model.
 Creating sdf file D:\DESIGNS\me\Compiled\model\me.sdf
 Creating vhdl file D:\DESIGNS\me\Compiled\model\me.vhd
 Creating timing report file D:\DESIGNS\me\Compiled\model\me.tim
 Reporting paths from primary input to primary output.
 Reporting paths from primary input to function input.
 Reporting paths from function output to primary output.
 Reporting paths from function output to function input.
 Done timing report.
 Modgen execution complete.
Build Completed
```

Figure 38. Generating Static Timing Analysis Reports

To do a static timing analysis, select Project Generate Timing Analysis. This will produce me.tim in the project directory.


UltraEdit - [D:\DESIGNS\ME\COMPILED\MODEL\ME.TIM]
File Edit View Macro Options Window Help
For Help, press F1
Line 1
Col. 1
Modified: 04/24/97 09:34:54
File Size: 49882
INS

```
|-----------------------------------------------------------------------------|
--
-- Timing report automatically generated by Modgen Version 3.7
-- Generated from D:\DESIGNS\me\Compiled\me.fb
-- Date Thu Apr 24 09:34:55 1997
--


--
--  PS PZ5032-6A44 (template: XPLA32_32)
--  Package: JLCC
--



                                   Signal Path  Delay Report

 KEYS:
 Instance prefix to logic mapping:
     PI       = primary input.
     PO       = primary output.
     B        = buffer.
     I        = inverter.
     BT       = tristate buffer.
     A        = and gate.
     O        = or gate.
     DFF      = D type flip-flop.
     DDE      = Dual Edge Clock flip-flop.
     TFF      = T type flip-flop.
```

Figure 39. me.tim Timing Report

This shows the first few lines of the timing report.

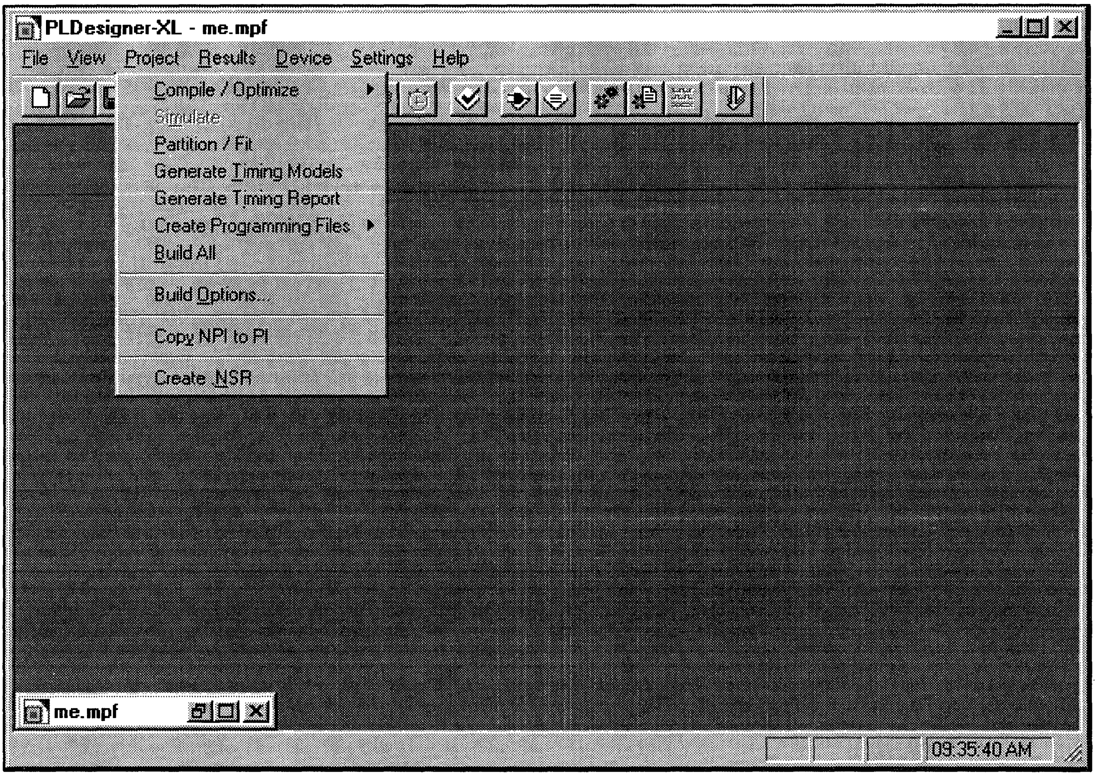


Figure 40. Generating VHDL and sdf Models

PLDesigner also generates a VHDL model and me.sdf from the design for timing simulation. Optionally a new testbench can be generated.

CONCLUSION

This outlines one flow for using OrCAD design entry and simulation tools to target Philips CPLDs.

Implementing a UART in Philips CPLDs

AN072

Author Lester Sanders, CPLD Applicatins Engineer

INTRODUCTION

The Universal Asynchronous Receiver Transmitter (UART) has been the most widely used serial data communication circuit ever. They allow full duplex communication over serial communication links as RS232. This application note implements a UART in Philips CPLDs. UARTs are available as inexpensive standard products from many semiconductor suppliers, including Philips, making it unlikely that this specific design is useful It is intended to illustrate sample Verilog code which simulates correctly and fits into a programmable logic device with restricted resources.

The basic functions of a UART are a microprocessor interface, double buffering of tranmitter data, frame generation, parity generation, parallel to serial conversion, double buffering of receiver data, parity checking, serial to parallel conversion. The frame format of used by UARTs is a low start bit, 5–8 data bits, optional parity bit, and 1 or 2 stop bits. Some UARTs include modem interface signals. These are pass–through signals which are not done in this design.

The Programmable Logic Group of Philips Semiconductor is developing a family of advanced 3–volt and 5–volt complex programmable logic devices(CPLDs). The XPLA series, designated as the PZ5000 – (5–volt) and PZ3000 (3–volt) series devices, is footprint compatible with the Altera 7000 series devices. The principle advantage of Philips CPLDs over all existing CPLDs is that they consume zero static power. The other advantages are 25% higher logic capacity and a better ability to fit logic with fixed pinouts. The PZ5128/PZ3128 are in–system programmable. All devices are all programmable on Data I/O and BP

The organization of this application note is to provide a section on the receiver and then the transmitter. The sections each provide the test fixture and simulation results, followed by the code used in the compilation to a jedec file.

The frame format for data transmitted/received by a UART is given if Figure 1. It consists of a high idle state of the line A character is from 5 – 8 data bits. The start bit is low and the single stop bit is high.

RECEIVER

The receiver interfaces to the data bus d[7:0] with the rdn signal. The controller can generate a rdn strobe if data_ready is true. The receiver is double buffered, allowing data to be held in the buffer register rbr[7:0] while data is shifted in serially into the receiver shift register rst[7:0]. This provides the controller flexibility with bus read operations.

The receiver detects the character frame and strips the start and stop bits. The no_bits_rcvd variable controls the word size.

The clkdiv[3:0] register is used to control the time at which the data is decoded. The receiver uses the 16x local clock and decodes the value of start, data, and stop nits in the center of the data cells. To do this, the start bit initializes a count operation using clkdiv[3:0]. After detecting the low going edge on the start bit, the receiver counts the 16x clock to 8 and decodes, or samples the value of the signal. The clkdiv[3:0] register is then reset to 0, and subsequently counts the 16x c.lock to 16. This provides center sampling for the data and stop bits.

Three error detection signals are provided but not implemented in the verilog source. Parity indicates whether an even or odd number of 1s are present in a data work. Overrun error indicates whether the receive buffer register is overwritten by the receive shift register prior to the controller reading the receiver buffer register. Framing error indicates if the stop bit is not high.

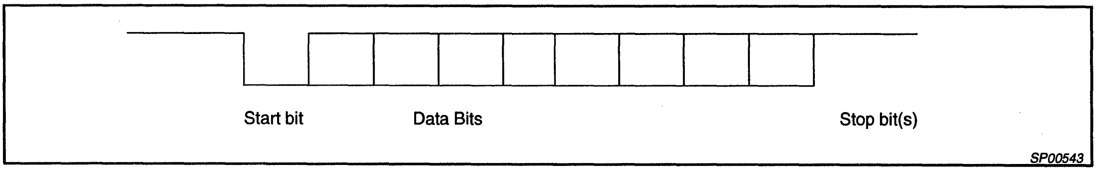


Figure 1. X UART Frame Format

SIGNAL	DIRECTION	FUNCTION
rst	Input	Resets
clk16x	Input	16x input clock
rdn	Input	Read strobe
data[7:0]	Output	Output data bus
fe	Output	Framing error status signal
oe	Output	Overun error status signal
pe	Output	Parity error status signal
rbr[7:0]	Internal	Receiver buffer registe – accepts data from data[7:0] and transfers it to rsr[7:0]
rsr[7:0]	Internal	Receiver shift register – accepts data from rbr[7:0] and transfers it to sdo
no_bite_rcvd	Internal	Tracks character size and sequences receiver operation
clk1x_enable	Internable	Enable signal for registers clocked by clk1x.
clk1x	Internal	1x clock used for internal operations

The receiver testfixture is

```
‘tscale 1 ns /1 ns
module receiver_tf;
reg rst;
reg clk16x;
reg clk1x;
reg rxd ;
reg rdn  ;
wire [7:0] data ;
wire oe ;
wire fe ;
wire pe ;

reg [3:0] no_bits_rcvd ;
reg clk1x_enable ;
reg [3:0] clkdiv ;
reg [7:0] rbr ;
reg [7:0] rsr ;
receiver u1 (data,data_ready,overrun_error,framing_error,parity_error,rxd,clk16x,rst,rdn) ;
initial begin
rst = 1’b0 ;
clk16x = 1’b0 ;
clk1x = 1’b0 ;
rxd = 1’b1 ;
no_bits_rcvd = 4’b0000 ;
clk1x_enable = 1’b0 ;
clkdiv = 4’b0 ;
rbr = 8’b0 ;
rsr = 8’b0 ;
rdn = 1’b1 ;
end
integer receiver_chann ;
initial begin
receiver_chann = $fopen(“receiver.rpt”) ;
$tformat (–9,,,5) ;
end
parameter clock_period = 10;
always #(clock_period/2) clk16x = ~clk16x ;
initial begin
$fdisplay(receiver_chann, “Verilog simulation of receiver design.\n”) ;
$shm_open(“receiver.shm”) ;
$shm_probe(“AS”) ;
$fdisplay (receiver_chann,”Verify reset.\n”) ;

$fmonitor(receiver_chann,”T=%t,rst=%b,rxd=%b,rxd1=%b,rxd2=%b,clk16x=%b,clk1x_enable=%b,clk1x=%b,no_bits_rcvd=%b,rsr=%h,rbr=%h,data=%h”,$t,rst,rxd,receiver.rxd1,receiver.rxd2,clk16x,receiver.clk1x_enable,receiver.clk1x,receiver.no_bits_rcvd,receiver.rsr,receiver.rbr,data) ;

#1 rst = 1’b1 ;
#10 rst = 1’b0 ;
#10 rxd = 1’b1 ;
#30 rxd = 1’b0 ;
#160 rxd = 1’b1 ;
```

```
#160 rxd = 1’b0 ;
#160 rxd = 1’b1 ;
#160 rxd = 1’b0 ;
#160 rxd = 1’b1 ;
#160 rxd = 1’b0 ;
#160 rxd = 1’b1 ;
#160 rxd = 1’b0 ;
#160 rxd = 1’b1 ;
#160 rxd = 1’b1 ;
#160 rxd = 1’b1 ;
#160 rdn = 1’b0 ;
#160 rdn = 1’b1 ;
#480
$fdisplay (receiver_chann,"\nSimulation of receiver complete.");
$finish ;
end
endmodule
```

The verilog soure is :

```
module receiver
(data,data_ready,overrun_error,framing_error,parity_error,rxd,clk16x
,rst,rdn) ;
input rxd ;
input clk16x ;
input rst ;
input rdn ;
output [7:0] data ;
output data_ready ;
output overrun_error ;
output framing_error ;
output parity_error ;
reg rxd1 ;
reg rxd2 ;
reg clk1x_enable ;
reg [3:0] clkdiv ;
reg [7:0] rsr ;
reg [7:0] rbr ;
reg [3:0] no_bits_rcvd ;
reg data_ready ;
reg parity ;
reg parity_error ;
reg framing_error ;
wire clk1x ;
wire rd ;
assign data = !rdn ? rbr : 8’bz ;
always @(posedge clk16x or posedge rst)
begin
if (rst)
begin
rxd1 = 1’b1 ;
rxd2 = 1’b1 ;
clk1x_enable = 1’b0;
end
else if (!rxd1 && rxd2)
clk1x_enable <= 1’b1 ;
if (no_bits_rcvd == 4’b1100)
begin
clk1x_enable = 1’b0 ;
end
rxd2 = rxd1 ;
rxd1 = rxd ;
end
always @(posedge clk16x or posedge rst or negedge rdn)
begin
if (rst)
data_ready = 1’b0 ;
else if (!rdn)
data_ready = 1’b0 ;
else
if (no_bits_rcvd == 4’b1011)
data_ready = 1’b1 ;
end
always @(posedge clk16x or posedge rst)
begin
if (rst)
clkdiv = 4’b0000 ;
else if (clk1x_enable)
clkdiv = clkdiv +1 ;
end
assign clk1x = clkdiv[3] ;
always @(negedge clk1x or posedge rst)
if (rst)
begin
```

```
rsr = 8'b0 ;
rbr = 8'b0 ;
parity = 1'b1 ;
framing_error = 1'b0 ;
parity_error = 1'b0 ;
end
else
begin
if (no_bits_rcvd >= 4'b0001 && no_bits_rcvd <= 4'b1001)
begin
rsr[0] = rxd2 ;
rsr[7:1] = rsr[6:0] ;
parity = parity ^ rsr[7] ;
end
else if (no_bits_rcvd == 4'b1010)
begin
rbr = rsr ;
end
else if (!parity)
parity_error = 1'b1 ;
else if ((no_bits_rcvd == 4'b1011) && (rxd2 != 1'b1))
framing_error = 1'b1 ;
else
framing_error = 1'b0 ;
end
always @(posedge clk1x or posedge rst or negedge clk1x_enable)
if (rst)
no_bits_rcvd = 4'b0000;
else
if (!clk1x_enable)
no_bits_rcvd = 4'b0000 ;
else
no_bits_rcvd = no_bits_rcvd + 1 ;
endmodule
```

The receiver simulation results and schematics are available from Philips CPLD Applications.

TRANSMITTER

The transmitter interfaces to the data bus with the transmitter buffer register empty (tbre) and the wrn signals. The controller can generate a wrn strobe if tbre is true. The transmitter is double buffered, allowing data to be held in the buffer register tbr[7:0] while data is being shifted out of the shift register tsr[7:0]. The transmitter generates a frame which consists of the idle state (high on sdo), low start bit, 8 data bits, and a stop bit. Parity is not generated. The no_bits_sent controls the word size and sequences the transmitter operations. To change the word size, change the value of no_bits_sent in the verilog source.

SIGNAL	DIRECTION	FUNCTION
rst	Input	Restes wrn1,wrn2,no_bits_sent, clkdiv[3:0],tbr[3:0],tsr[3:0]
clk16x	Input	Local reference clock 16X the data rate
wrn	Input	Control signal which strobes data from d[7:0] to tbr[7:0]
sdo	Output	Serial data output
tbre	Output	Status signal indication that the transmiter buffer register is empty
tsre	Internal	Status signal indication that the transmitter shift register is empty
no_bits_sent	Internal	Controls word_size and sequences transmitter operation
clk1x_enable	Internal	Enables internal clock clk1x.
tbr[7:0]	Internal	Accepts data from d[&:0] and transfers data to tsr[7:0]
tsr[7:0]	Internal	Receives data frem tbr[7:0] and shifts to sdo
load_tsr	Internal	Enables transfer from tbr[7:0] to tsr[7:0]
clkdiv[3:0]	Internal	Used in generation of internal clock

The transmitter testfixture is:

```
'tscale 1 ns /1 ns
module transmitter_tf;
reg rst;
reg clk16x ;
reg clk1x ;
reg wrn ;
reg [2:0] word_size ;
reg [7:0] data ;
wire tbre ;
wire tsre ;
wire sdo ;
reg [3:0] no_bits_sent ;
reg load_tsr ;
reg clk1x_enable ;
reg [3:0] clkdiv ;
reg [7:0] tbr ;
reg [7:0] tsr ;
transmitter u1 (data,tbre,tsre,rst,clk16x,wrn,sdo) ;
initial begin
rst = 1'b0 ;
clk16x = 1'b0 ;
clk1x = 1'b0 ;
wrn = 1'b1 ;
no_bits_sent = 4'b0000 ;
clk1x_enable = 1'b0 ;
clkdiv = 4'b0 ;
tbr = 8'b0 ;
tsr = 8'b0 ;
end
integer transmitter_chann ;
initial begin
transmitter_chann = $fopen(“transmitter.rpt”) ;
$tformat (-9,,,5) ;
end
parameter clock_period = 10;
always #(clock_period/2) clk16x = ~clk16x ;
initial begin
$fdisplay(transmitter_chann, “Verilog simulation of transmitter
design.\n”) ;
$shm_open(“transmit.shm”) ;
$shm_probe(“AS”) ;
$fdisplay (transmitter_chann,”Verify reset.\n”) ;
$fmonitor(transmitter_chann,”T=%t,rst=%b,wrn=%b,wrn1=%b,wrn2=
%b,tbre=%b,tsre=%b,clk1x_enable=%b,no_bits_sent=%b,clk1x=%b
,tbr=%h,tsr=%h,parity=%b,sdo=%b”,$t,rst,wrn,transmitter.wrn1,trans
mitter.wrn2,tbre,tsre,transmitter.clk1x_enable,transmitter.no_bits_se
nt,transmitter.clk1x,transmitter.tbr,transmitter.tsr,transmitter.parity,sd
o) ;
#1 rst = 1'b1 ;
#100 rst = 1'b0 ;
#10 data = 8'haa ;
#(1 * clock_period) wrn = 1'b0 ;
#(1 * clock_period) wrn = 1'b1 ;
#2500
#10 data = 8'hf0 ;
#(1 * clock_period) wrn = 1'b0 ;
#(1 * clock_period) wrn = 1'b1 ;
#2500
$fdisplay (transmitter_chann,”\nSimulation of transmitter complete.”);
$finish ;
end
endmodule
```

Verilog simulation of transmitter design.

```
Verify reset.

T=0,wrn=1,wrn1=x,wrn2=x,tbre=x,tsre=x,clk1x_enable=x,no_bits_se
nt=xxxx,clk1x=x,tbr=xx,tsr=xx,parity=x,sdo=x

T=1,rst=1,wrn=1,wrn1=1,wrn2=1,tbre=1,tsre=1,clk1x_enable=0,no_
bits_sent=0000,clk1x=0,tbr=xx,tsr=xx,parity=1,sd1

T=
101,wrn=1,wrn1=1,wrn2=1,tbre=1,tsre=1,clk1x_enable=0,no_bits_s
ent=0000,clk1x=0,tbr=xx,tsr=xx,parity=1,sdo=1

T=
121,wrn=0,wrn1=1,wrn2=1,tbre=1,tsre=1,clk1x_enable=0,no_bits_s
ent=0000,clk1x=0,tbr=aa,tsr=xx,parity=1,sdo=1

T=
125,wrn=0,wrn1=0,wrn2=1,tbre=1,tsre=1,clk1x_enable=0,no_bits_s
ent=0000,clk1x=0,tbr=aa,tsr=xx,parity=1,sdo=1

T=
131,wrn=1,wrn1=0,wrn2=1,tbre=1,tsre=1,clk1x_enable=0,no_bits_s
ent=0000,clk1x=0,tbr=aa,tsr=xx,parity=1,sdo=1

T=
135,wrn=1,wrn1=1,wrn2=0,tbre=0,tsre=1,clk1x_enable=1,no_bits_s
ent=0000,clk1x=0,tbr=aa,tsr=xx,parity=1,sdo=1

T=
145,wrn=1,wrn1=1,wrn2=1,tbre=0,tsre=1,clk1x_enable=1,no_bits_s
ent=0000,clk1x=0,tbr=aa,tsr=xx,parity=1,sdo=1
```

```
T= 215,wrn=1,wrn1=1,wrn2=1,tbre=0,tsre=1,clk1x_enable=1,no_bits_sent=0001,clk1x=1,tbr=aa,tsr=xx,parity=1,sdo=1
T= 295,wrn=1,wrn1=1,wrn2=1,tbre=0,tsre=0,clk1x_enable=1,no_bits_sent=0001,clk1x=0,tbr=aa,tsr=aa,parity=1,sdo=1
T= 375,wrn=1,wrn1=1,wrn2=1,tbre=0,tsre=0,clk1x_enable=1,no_bits_sent=0010,clk1x=1,tbr=aa,tsr=aa,parity=1,sdo=1
T= 385,wrn=1,wrn1=1,wrn2=1,tbre=1,tsre=0,clk1x_enable=1,no_bits_sent=0010,clk1x=1,tbr=aa,tsr=aa,parity=1,sdo=1
T= 455,wrn=1,wrn1=1,wrn2=1,tbre=1,tsre=0,clk1x_enable=1,no_bits_sent=0010,clk1x=0,tbr=aa,tsr=aa,parity=1,sdo=0
T= 535,wrn=1,wrn1=1,wrn2=1,tbre=1,tsre=0,clk1x_enable=1,no_bits_sent=0011,clk1x=1,tbr=aa,tsr=aa,parity=1,sdo=0
T= 615,wrn=1,wrn1=1,wrn2=1,tbre=1,tsre=0,clk1x_enable=1,no_bits_sent=0011,clk1x=0,tbr=aa,tsr=54,parity=1,sdo=0
T= 695,wrn=1,wrn1=1,wrn2=1,tbre=1,tsre=0,clk1x_enable=1,no_bits_sent=0100,clk1x=1,tbr=aa,tsr=54,parity=1,sdo=0
T= 775,wrn=1,wrn1=1,wrn2=1,tbre=1,tsre=0,clk1x_enable=1,no_bits_sent=0100,clk1x=0,tbr=aa,tsr=a8,parity=0,sdo=1
T= 855,wrn=1,wrn1=1,wrn2=1,tbre=1,tsre=0,clk1x_enable=1,no_bits_sent=0101,clk1x=1,tbr=aa,tsr=a8,parity=0,sdo=1
T= 935,wrn=1,wrn1=1,wrn2=1,tbre=1,tsre=0,clk1x_enable=1,no_bits_sent=0101,clk1x=0,tbr=aa,tsr=50,parity=0,sdo=0
T=1015,wrn=1,wrn1=1,wrn2=1,tbre=1,tsre=0,clk1x_enable=1,no_bits_sent=0110,clk1x=1,tbr=aa,tsr=50,parity=0,sdo0
T=1095,wrn=1,wrn1=1,wrn2=1,tbre=1,tsre=0,clk1x_enable=1,no_bits_sent=0110,clk1x=0,tbr=aa,tsr=a0,parity=1,sdo1
T=1175,wrn=1,wrn1=1,wrn2=1,tbre=1,tsre=0,clk1x_enable=1,no_bits_sent=0111,clk1x=1,tbr=aa,tsr=a0,parity=1,sdo1
T=1255,wrn=1,wrn1=1,wrn2=1,tbre=1,tsre=0,clk1x_enable=1,no_bits_sent=0111,clk1x=0,tbr=aa,tsr=40,parity=1,sdo0
T=1335,wrn=1,wrn1=1,wrn2=1,tbre=1,tsre=0,clk1x_enable=1,no_bits_sent=1000,clk1x=1,tbr=aa,tsr=40,parity=1,sdo0
T=1415,wrn=1,wrn1=1,wrn2=1,tbre=1,tsre=0,clk1x_enable=1,no_bits_sent=1000,clk1x=0,tbr=aa,tsr=80,parity=0,sdo1
T=1495,wrn=1,wrn1=1,wrn2=1,tbre=1,tsre=0,clk1x_enable=1,no_bits_sent=1001,clk1x=1,tbr=aa,tsr=80,parity=0,sdo1
T=1575,wrn=1,wrn1=1,wrn2=1,tbre=1,tsre=0,clk1x_enable=1,no_bits_sent=1001,clk1x=0,tbr=aa,tsr=00,parity=0,sdo0
T=1655,wrn=1,wrn1=1,wrn2=1,tbre=1,tsre=0,clk1x_enable=1,no_bits_sent=1010,clk1x=1,tbr=aa,tsr=00,parity=0,sdo0
T=1735,wrn=1,wrn1=1,wrn2=1,tbre=1,tsre=0,clk1x_enable=1,no_bits_sent=1010,clk1x=0,tbr=aa,tsr=00,parity=0,sdo0
T=1815,wrn=1,wrn1=1,wrn2=1,tbre=1,tsre=0,clk1x_enable=1,no_bits_sent=1011,clk1x=1,tbr=aa,tsr=00,parity=0,sdo0
T=1895,wrn=1,wrn1=1,wrn2=1,tbre=1,tsre=0,clk1x_enable=1,no_bits_sent=1011,clk1x=0,tbr=aa,tsr=00,parity=0,sdo0
T=1975,wrn=1,wrn1=1,wrn2=1,tbre=1,tsre=0,clk1x_enable=1,no_bits_sent=1100,clk1x=1,tbr=aa,tsr=00,parity=0,sdo=0
T=2055,wrn=1,wrn1=1,wrn2=1,tbre=1,tsre=1,clk1x_enable=1,no_bits_sent=1100,clk1x=0,tbr=aa,tsr=00,parity=0,sdo=1
T=2135,wrn=1,wrn1=1,wrn2=1,tbre=1,tsre=1,clk1x_enable=1,no_bits_sent=1101,clk1x=1,tbr=aa,tsr=00,parity=0,sdo=1
T=2145,wrn=1,wrn1=1,wrn2=1,tbre=1,tsre=1,clk1x_enable=0,no_bits_sent=0000,clk1x=1,tbr=aa,tsr=00,parity=0,sdo=1
T=2651,wrn=0,wrn1=1,wrn2=1,tbre=1,tsre=1,clk1x_enable=0,no_bits_sent=0000,clk1x=1,tbr=f0,tsr=00,parity=0,sdo=1
T=2655,wrn=0,wrn1=0,wrn2=1,tbre=1,tsre=1,clk1x_enable=0,no_bits_sent=0000,clk1x=1,tbr=f0,tsr=00,parity=0,sdo=1
T=2661,wrn=1,wrn1=0,wrn2=1,tbre=1,tsre=1,clk1x_enable=0,no_bits_sent=0000,clk1x=1,tbr=f0,tsr=00,parity=0,sdo=1
T=2665,wrn=1,wrn1=1,wrn2=0,tbre=0,tsre=1,clk1x_enable=1,no_bits_sent=0000,clk1x=1,tbr=f0,tsr=00,parity=0,sdo=1
T=2675,wrn=1,wrn1=1,wrn2=1,tbre=0,tsre=1,clk1x_enable=1,no_bits_sent=0000,clk1x=1,tbr=f0,tsr=00,parity=0,sdo=1
T=2735,wrn=1,wrn1=1,wrn2=1,tbre=0,tsre=1,clk1x_enable=1,no_bits_sent=0000,clk1x=0,tbr=f0,tsr=00,parity=0,sdo=1
T=2815,wrn=1,wrn1=1,wrn2=1,tbre=0,tsre=1,clk1x_enable=1,no_bits_sent=0001,clk1x=1,tbr=f0,tsr=00,parity=0,sdo=1
T= 2895,wrn=1,wrn1=1,wrn2=1,tbre=0,tsre=0,clk1x_enable=1,no_bits_sent=0001,clk1x=0,tbr=f0,tsr=f0,parity=0,sdo=1
T= 2975,wrn=1,wrn1=1,wrn2=1,tbre=0,tsre=0,clk1x_enable=1,no_bits_sent=0010,clk1x=1,tbr=f0,tsr=f0,parity=0,sdo=1
T= 2985,wrn=1,wrn1=1,wrn2=1,tbre=1,tsre=0,clk1x_enable=1,no_bits_sent=0010,clk1x=1,tbr=f0,tsr=f0,parity=0,sdo=1
T= 3055,wrn=1,wrn1=1,wrn2=1,tbre=1,tsre=0,clk1x_enable=1,no_bits_sent=0010,clk1x=0,tbr=f0,tsr=f0,parity=0,sdo=0
T= 3135,wrn=1,wrn1=1,wrn2=1,tbre=1,tsre=0,clk1x_enable=1,no_bits_sent=0011,clk1x=1,tbr=f0,tsr=f0,parity=0,sdo=0
T=3215,wrn=1,wrn1=1,wrn2=1,tbre=1,tsre=0,clk1x_enable=1,no_bits_sent=0011,clk1x=0,tbr=f0,tsr=e0,parity=1,sdo=1
T=3295,wrn=1,wrn1=1,wrn2=1,tbre=1,tsre=0,clk1x_enable=1,no_bits_sent=0100,clk1x=1,tbr=f0,tsr=e0,parity=1,sdo=1
T=3375,wrn=1,wrn1=1,wrn2=1,tbre=1,tsre=0,clk1x_enable=1,no_bits_sent=0100,clk1x=0,tbr=f0,tsr=c0,parity=0,sdo=1
T=3455,wrn=1,wrn1=1,wrn2=1,tbre=1,tsre=0,clk1x_enable=1,no_bits_sent=0101,clk1x=1,tbr=f0,tsr=c0,parity=0,sdo=1
T=3535,wrn=1,wrn1=1,wrn2=1,tbre=1,tsre=0,clk1x_enable=1,no_bits_sent=0101,clk1x=0,tbr=f0,tsr=80,parity=1,sdo=1
```

```
T=3615,wrn=1,wrn1=1,wrn2=1,tbre=1,tsre=0,clk1x_enable=1,no_bit
s_sent=0110,clk1x=1,tbr=f0,tsr=80,parity=1,sdo=1
T=3695,wrn=1,wrn1=1,wrn2=1,tbre=1,tsre=0,clk1x_enable=1,no_bit
s_sent=0110,clk1x=0,tbr=f0,tsr=00,parity=1,sdo=0
T=3775,wrn=1,wrn1=1,wrn2=1,tbre=1,tsre=0,clk1x_enable=1,no_bit
s_sent=0111,clk1x=1,tbr=f0,tsr=00,parity=1,sdo=0
T=3855,wrn=1,wrn1=1,wrn2=1,tbre=1,tsre=0,clk1x_enable=1,no_bit
s_sent=0111,clk1x=0,tbr=f0,tsr=00,parity=1,sdo=0
T=3935,wrn=1,wrn1=1,wrn2=1,tbre=1,tsre=0,clk1x_enable=1,no_bit
s_sent=1000,clk1x=1,tbr=f0,tsr=00,parity=1,sdo=0
T=4015,wrn=1,wrn1=1,wrn2=1,tbre=1,tsre=0,clk1x_enable=1,no_bit
s_sent=1000,clk1x=0,tbr=f0,tsr=00,parity=1,sdo=0
T=4095,wrn=1,wrn1=1,wrn2=1,tbre=1,tsre=0,clk1x_enable=1,no_bit
s_sent=1001,clk1x=1,tbr=f0,tsr=00,parity=1,sdo=0
T=4175,wrn=1,wrn1=1,wrn2=1,tbre=1,tsre=0,clk1x_enable=1,no_bit
s_sent=1001,clk1x=0,tbr=f0,tsr=00,parity=1,sdo=0
T=4255,wrn=1,wrn1=1,wrn2=1,tbre=1,tsre=0,clk1x_enable=1,no_bit
s_sent=1010,clk1x=1,tbr=f0,tsr=00,parity=1,sdo=0
T=4335,wrn=1,wrn1=1,wrn2=1,tbre=1,tsre=0,clk1x_enable=1,no_bit
s_sent=1010,clk1x=0,tbr=f0,tsr=00,parity=1,sdo=0
T=4415,wrn=1,wrn1=1,wrn2=1,tbre=1,tsre=0,clk1x_enable=1,no_bit
s_sent=1011,clk1x=1,tbr=f0,tsr=00,parity=1,sdo=0
T=4495,wrn=1,wrn1=1,wrn2=1,tbre=1,tsre=0,clk1x_enable=1,no_bit
s_sent=1011,clk1x=0,tbr=f0,tsr=00,parity=1,sdo=1
T=4575,wrn=1,wrn1=1,wrn2=1,tbre=1,tsre=0,clk1x_enable=1,no_bit
s_sent=1100,clk1x=1,tbr=f0,tsr=00,parity=1,sdo=1
T=4655,wrn=1,wrn1=1,wrn2=1,tbre=1,tsre=1,clk1x_enable=1,no_bit
s_sent=1100,clk1x=0,tbr=f0,tsr=00,parity=1,sdo=1
T=4735,wrn=1,wrn1=1,wrn2=1,tbre=1,tsre=1,clk1x_enable=1,no_bit
s_sent=1101,clk1x=1,tbr=f0,tsr=00,parity=1,sdo=1
T=4745,wrn=1,wrn1=1,wrn2=1,tbre=1,tsre=1,clk1x_enable=0,no_bit
s_sent=0000,clk1x=1,tbr=f0,tsr=00,parity=1,sdo=1
Simulation of transmitter complete.
```

The verilog source for the transmitter is :

```
'tscale 1 ns / 1 ns
module transmitter (data,tbre,tsre,rst,clk16x,wrn,sdo) ;
output tbre ;
output tsre ;
output sdo ;
input [7:0] data ;
input rst ;
input clk16x ;
input wrn ;
reg tbre ;
reg tsre ;
reg clk1x_enable ;
reg [7:0] tsr ;
reg [7:0] tbr ;
reg parity ;
reg[3:0] clkdiv ;
wire clk1x ;
reg sdo ;
reg [3:0] no_bits_sent ;
reg wrn1 ;
reg wrn2 ;
always @(posedge clk16x or posedge rst)
begin
if (rst)
begin
wrn1 = 1'b1 ;
wrn2 = 1'b1 ;
tbre = 1'b1 ;
clk1x_enable = 1'b0 ;
end
else if (!wrn1 && wrn2)
begin
tbre <= 1'b0 ;
clk1x_enable <= 1'b1 ;
end
if (no_bits_sent == 4'b0010)
tbre = 1'b1 ;
if (no_bits_sent == 4'b1101)
clk1x_enable = 1'b0 ;
wrn2 = wrn1 ;
wrn1 = wrn ;
end
always @(negedge wrn)
tbr = data ;
always @(posedge clk16x or posedge rst)
begin
if (rst)
clkdiv = 4'b0 ;
else if (clk1x_enable)
clkdiv = clkdiv + 1 ;
end
assign clk1x = clkdiv[3] ;
always @(negedge clk1x or posedge rst)
if (rst)
```

```
begin
sdo = 1'b1 ;
tsre = 1'b1 ;
parity = 1'b1 ;
end
else
begin
if (no_bits_sent == 4'b0001)
begin
tsr = tbr ;
tsre = 1'b0 ;
end
else if (no_bits_sent == 4'b0010)
begin
sdo = 1'b0 ;
end
else
if ((no_bits_sent >= 4'b0011) && (no_bits_sent <= 4'b1010))
begin
tsr[7:1] = tsr[6:0] ;
tsr[0] = 1'b0 ;
sdo = tsr[7] ;
parity = parity ^ tsr[7] ;
end
else if (no_bits_sent == 4'b1011)
begin
sdo = parity ;
end
else if (no_bits_sent == 4'b1100)
begin
sdo = 1'b1 ;
tsre = 1'b1 ;
end
end
always @(posedge clk1x or posedge rst or negedge clk1x_enable)
if (rst)
no_bits_sent = 4'b0000 ;
else if (!clk1x_enable)
no_bits_sent = 4'b0000 ;
else
no_bits_sent = no_bits_sent + 1 ;
endmodule
```

The transmitter simulation results and schematics are avalilable from Philips CPLD Applications.

Synplicity/Model Tech design flow for targeting Philips CPLDs

AN073

INTRODUCTION

Philips Semiconductor has developed a family of advanced 3-volt and 5-volt complex programmable logic devices(CPLDs). The XPLA series, designated as the PZ5000 - (5-volt) and PZ3000 (3-volt) series devices, is footprint compatible with the Altera 7000 series devices. The principle advantage of Philips CPLDs over all existing CPLDs is that they consume zero static power. The other advantages are 25% higher logic capacity and a better ability to fit logic with fixed pinouts. The PZ5128/PZ3128 are in-system programmable. All devices are all programmable on Data I/O and BP Microsystems programmers.

Minc Inc has developed fitters for the PZ5000/PZ3000 series for up to 128 macrocells. This Minc fitter allows users to target Philips CPLDs in both PC and workstation environments. The software is capable of automatically partitioning across multiple CPLDs. VHDL models are generated for timing simulation and post fit board-level simulation.

This note provides scripts for using this capability with Synplicity synthesis and Model Technology simulation tools. This flow can be used with minor edits for Verilog synthesis.

For additional information, telephone Philips Applications Support at 888-coolpld or browse http://www.coolpld.com.The following documentation is available either through the web server or telephoning 888 coolpld.

PLDesigner-XL User's Guide

Synpliciy Reference Manual

Vsystem VHDL User's Manual

PZ5000/PZ3000 Series Data Sheets

DESIGN FLOW

The sýnthesis software used in this note, Synplify, is available from Synplicity. The simulation software is available from Model Technology. The fitter software is from Minc Inc. The software required depends on the design flow. Most of this software also runs on PCs. Designs can be processed using the GUI provided by the tool vendor or using scripts. When scripts are used in the steps listed below, $1 is used to represent the design, and $_tb the testbench. Generally, there are a number of different methods to design using these tools , and scripts may vary based on user preferences..

This note provides a design flow for using Synplicity's Synplify synthesis and Model Tech's QuickVHDL simulators.

Flow

The steps given below do the following:

1. Setup environment
2. Create a testbench and simulate with QuickVHDL
3. Synthesize using Synplify.
4. Use Minc fitter to compile the src f file to a jedec file
5. Simulate with delays using Quickvhdl.

<u>Model Technology Functional Simulation</u>

The basic steps to do a functional simulation are:

qhlib work

qhmap work "<design_path>/work"

```
qhmap minc_vhd "$MINC_PATH/modellib/minc_vhd
```

```
qvhcom <design>.vhd
```

```
qvhcom <design_tb>.vhd
```

```
qhsim testbench v1
```

Assuming testbench is the entity and v1 is the architecture in <design_tb>, the QuickVHDL command window is invoked. The baseline steps then are to invoke the signals window, and from Signals select Wave/Signals in Region, and Run. See the Model Tech documentation for details on simulation techniques.

Synplify steps

To start Synpliciy, enter synplify. Using the Synplicity GUI, select the design source and CPLD target. Select Run to generate a src file as given in the example.

Using the Minc fitter

```
minc.script $1
```

The contents of minc.script are

```
plcomp $1.src
#plsim $1.stm
plopt $1.afb
plscan $1
plfit $1
plfuse $1
pldoc $1
modgen $1 vhd std noconf tb
```

This produces a jedec file ($1.j1) , $101.rpt , and $1.doc files. It also generates a vhdl model and sdf file in the <project> model directory. The jedec file can be used to program a PZ3000 or PZ5000 series device[1]. The model,sdf and testbench files written to the <project_path>/model directory can be read into Quickvhdl fo r timing simulation.

Selecting a Philips CPLD

The Philips CPLD used is specified in the <design>.pi and <design>.cst files. This allows a user to direct PLDesigner to either target a specific device as the PZ3032 or to scan all devices and provide multiple solutions. The use of these files is described in detail in Chapters 14-16 of the PLDesigner-XL User's Guide. To target the PZ3032, the following can be used .

<design>.cst

```
TEMPLATE =  XPLA32_32 ;
```

The basic pi file is given below. The part number for the Philips devices is listed in $MINC_PATH/minclib.avl

<design>.pi

```
DEVICE

TARGET 'TEMPLATE XPLA32_32 TQFP-44-P32';
```

[1] Depending on the .pi file, PLDesigner can partition a design across multiple devices, so that $1.j1, $1,j2,... are produced.

```
default ;
END DEVICE;
```

DESIGN FLOW EXAMPLE

An example of a flow using QuickVHDL, Synplify, and PLDesigner is given below. The example is an 8-bit comparator.

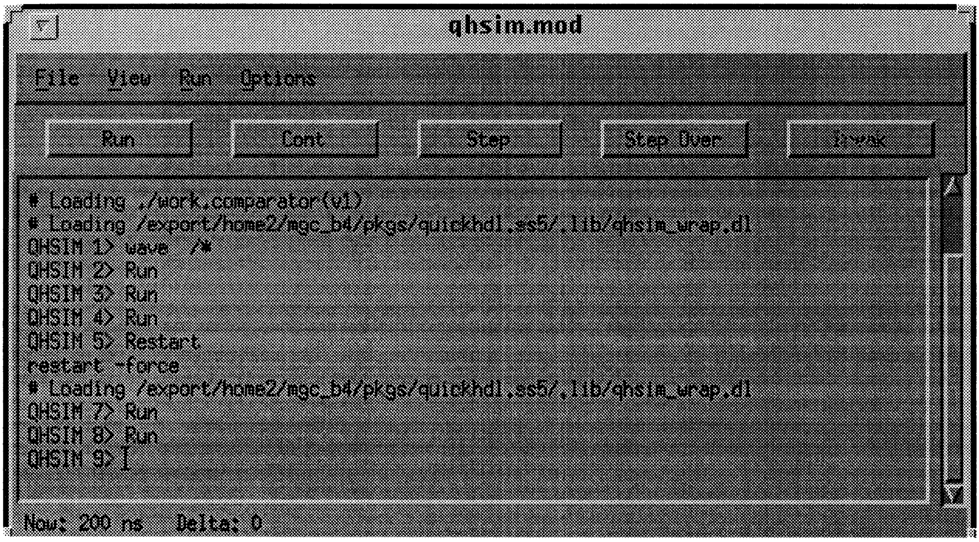
The vhdl testbench is

```
-- Philips CPLD Applications
-- comparator_tb.vhd
-- Mar 8 96
entity testbench is end ;
library ieee ;
use ieee.std_logic_1164.all ;
architecture v1 of testbench is
component comparator
port (a,b: in std_logic_vector (7 downto 0) := “00000000”;
aeqb : out std_logic
);
end component ;
signal a : std_logic_vector (7 downto 0) ;
signal b : std_logic_vector (7 downto 0) ;
signal aeqb : std_logic ;
type test_record is record
a : std_logic_vector (7 downto 0) ;
b : std_logic_vector (7 downto 0) ;
aeqb : std_logic ;
end record ;
type test_array is array(positive range<>) of test_record ;
constant test_vectors : test_array := (
--  a, b, aeqb
(“00000000”,”00000000”,’1’),
(“10000000”,”00000000”,’0’),
(“01000010”,”01000010”,’1’),
(“00011100”,”00000000”,’0’)
);
-- instantiate the component
begin
```

```
dut : comparator port map (
a => a,
b => b,
aeqb => aeqb) ;
-- provide stimulus and check results
process
variable vector : test_record ;
begin
for index in test_vectors’range loop
vector := test_vectors(index);
a <= vector.a ;
b <= vector.b ;
wait for 20 ns ;
assert aeqb = test_vectors(index).aeqb
report “Output aeqb is incorrect.” severity warning ;
end loop ;
wait ;
end process ;
end ;
```

The vhdl source is

```
-- Philips CPLD Applications
-- 8-bit comparator
-- Nov 28, 1996
library ieee;
use ieee.std_logic_1164.all;
entity comparator is
  port(a,b:  in   std_logic_vector(7 downto 0) := “00000000”;
      aeqb: out  std_logic);
end comparator;
architecture v1 of comparator is
begin
aeqb <= ‘1’ when (a = b) else ‘0’;
end v1;
```



```
# Loading ./work.comparator(v1)
# Loading /export/home2/mgc_b4/pkgs/quickhdl.ss5/.lib/qhsim_wrap.dl
QHSIM 1> wave /*
QHSIM 2> Run
QHSIM 3> Run
QHSIM 4> Run
QHSIM 5> Restart
restart -force
# Loading /export/home2/mgc_b4/pkgs/quickhdl.ss5/.lib/qhsim_wrap.dl
QHSIM 7> Run
QHSIM 8> Run
QHSIM 9>
```

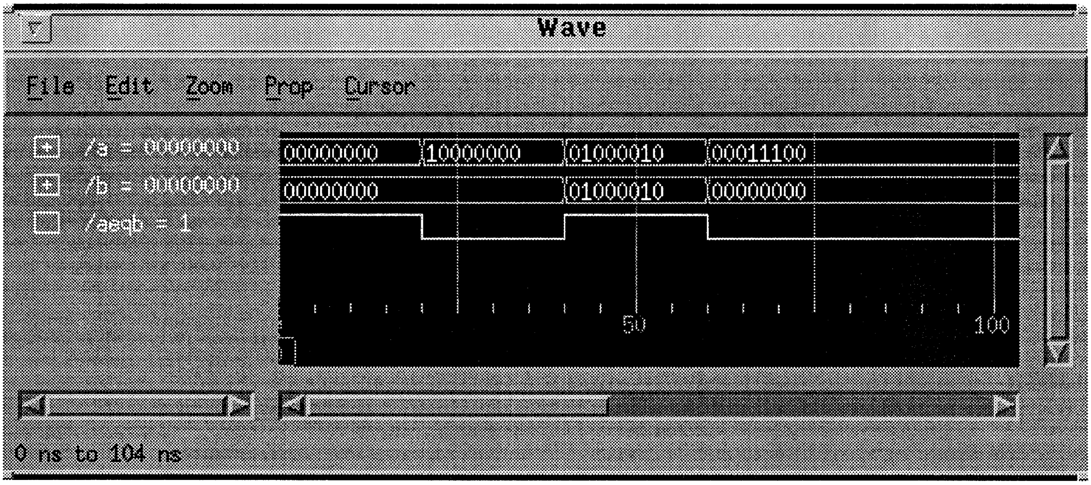
```
Now: 200 ns   Delta: 0
```

To do a funcional simulation,

```
qvlib work
qvmap work "/export/home/lss/designs/synplicity/vhdl/comparator/work"
qvcom comparator.vhd
qvcom comparator_tb.vhd
qvsim testbench v1
```

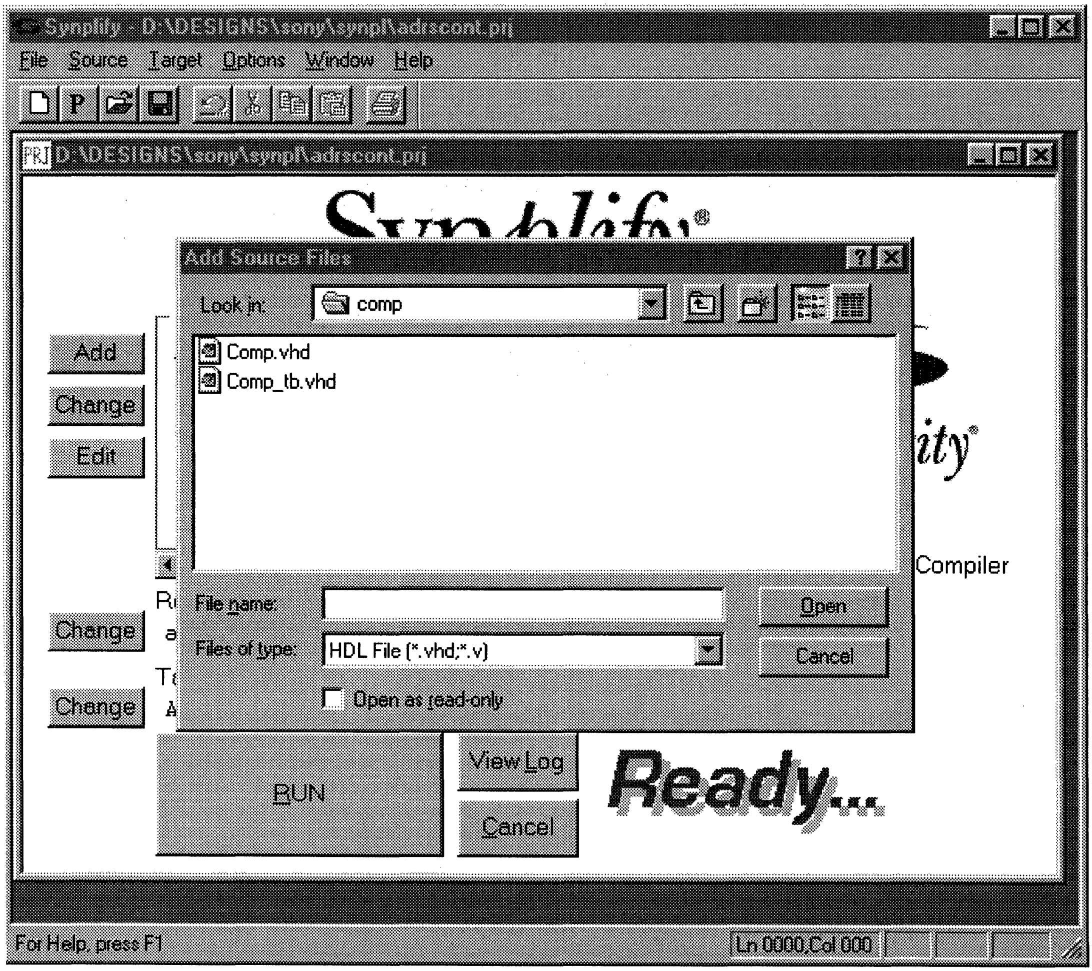
The command window is invoked as shown above.

Now invoke the wave window and run the simulation for 100 ns.



Synthesis

To invoke the Synplicity GUI, enter synplify and select the project.



Synplify - D:\DESIGNS\sony\synpl\adrscont.prj

D:\DESIGNS\sony\synpl\adrscont.prj

Add Source Files

Look in: comp

Comp.vhd
Comp_tb.vhd

File name:

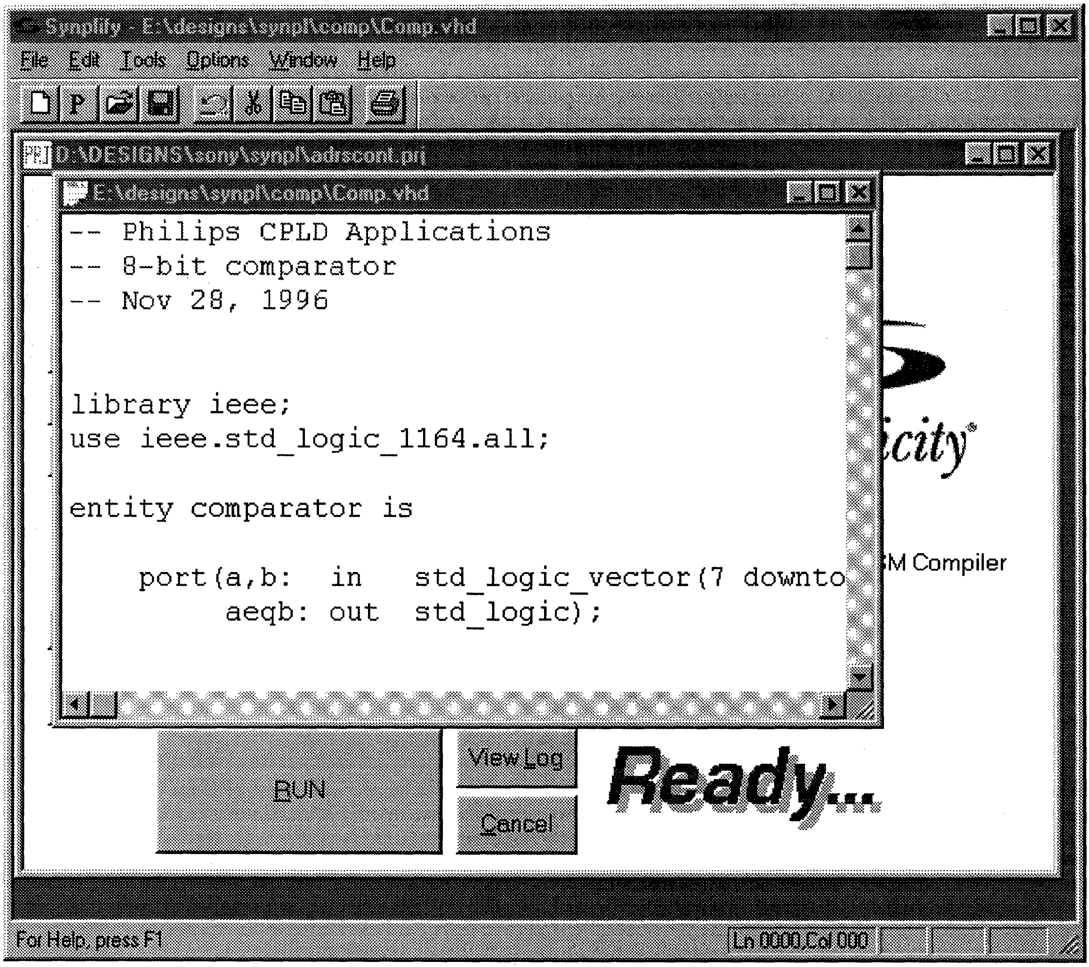
Files of type: HDL File (*.vhd;*.v)

□ Open as read-only

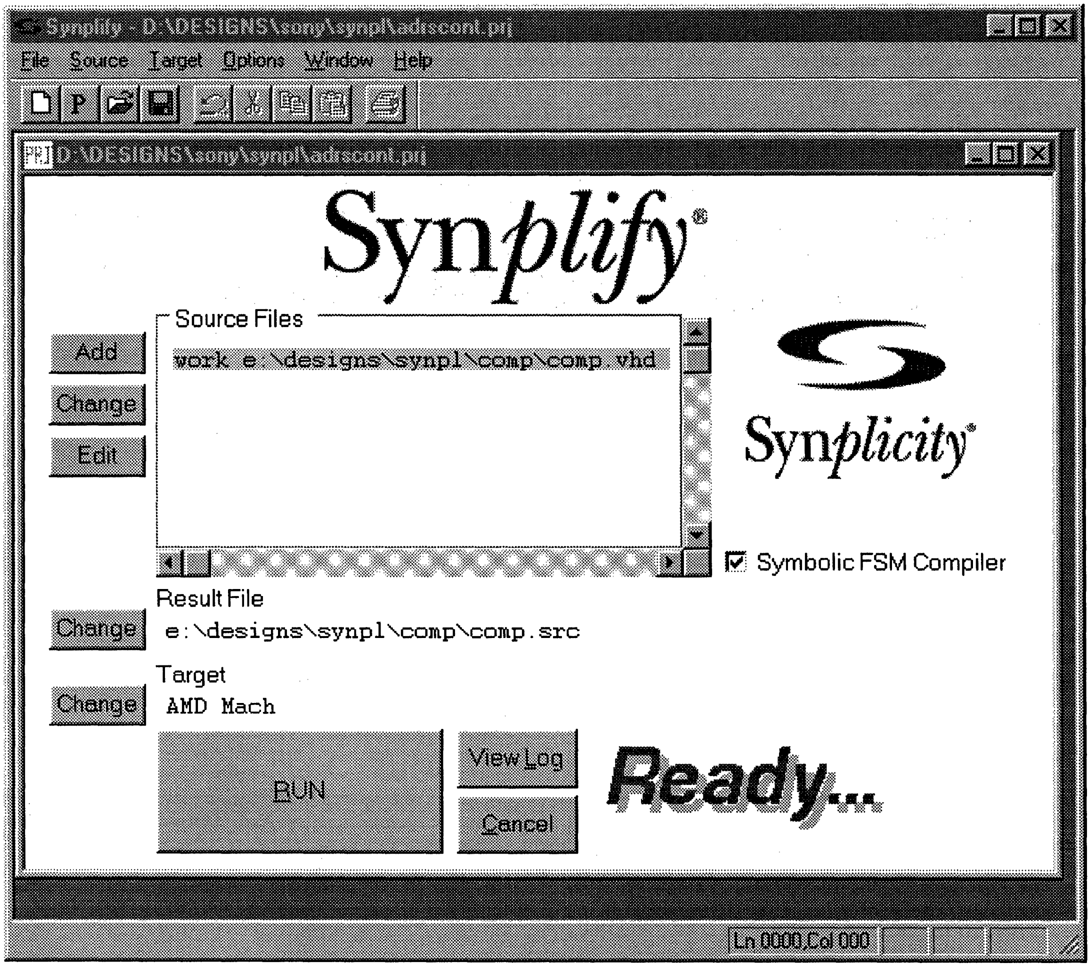
For Help, press F1

Ln 0000,Col 000

The VHDL source can be viewed from Synpicity.


Synplify - E:\designs\synpl\comp\Comp.vhd
File Edit Tools Options Window Help
D:\DESIGNS\sony\synpl\adrscont.prj
E:\designs\synpl\comp\Comp.vhd
icity
M Compiler
RUN
View Log
Cancel
Ready...
For Help, press F1
Ln 0000,Col 000

```
-- Philips CPLD Applications
-- 8-bit comparator
-- Nov 28, 1996



library ieee;
use ieee.std_logic_1164.all;

entity comparator is

    port(a,b:  in   std_logic_vector(7 downto
         aeqb: out  std_logic);
```



Source Files

```
work e:\designs\synpl\comp\comp.vhd
```

☑ Symbolic FSM Compiler

Result File

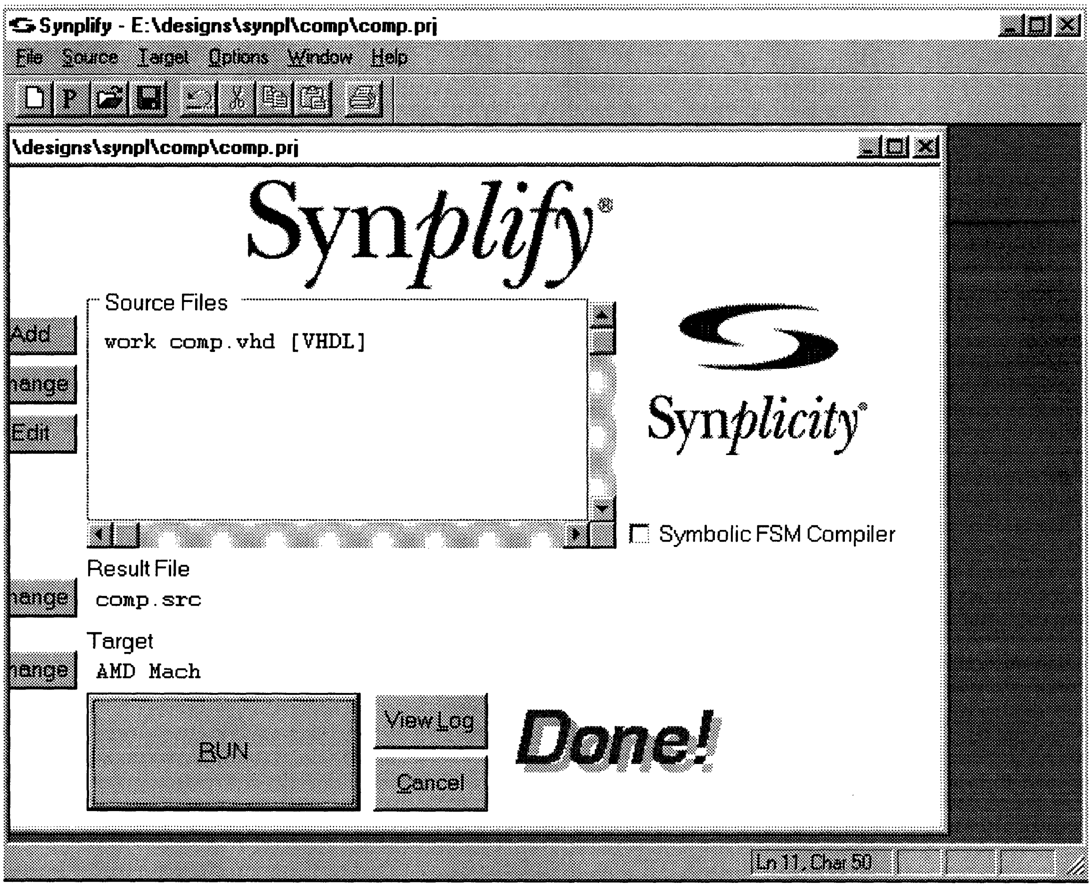
```
e:\designs\synpl\comp\comp.src
```

Target

```
AMD Mach
```

Ln 0000,Col 000

The targe is selected - currently as AMD, soon as Philips Coolrunner:



Synplify - E:\designs\synpl\comp\comp.prj

\designs\synpl\comp\comp.prj

Source Files

```
work comp.vhd [VHDL]
```

□ Symbolic FSM Compiler

Result File

```
comp.src
```

Target

```
AMD Mach
```

Done!

Selecting run caues a src file to be created.

Using PLDesigner-XL

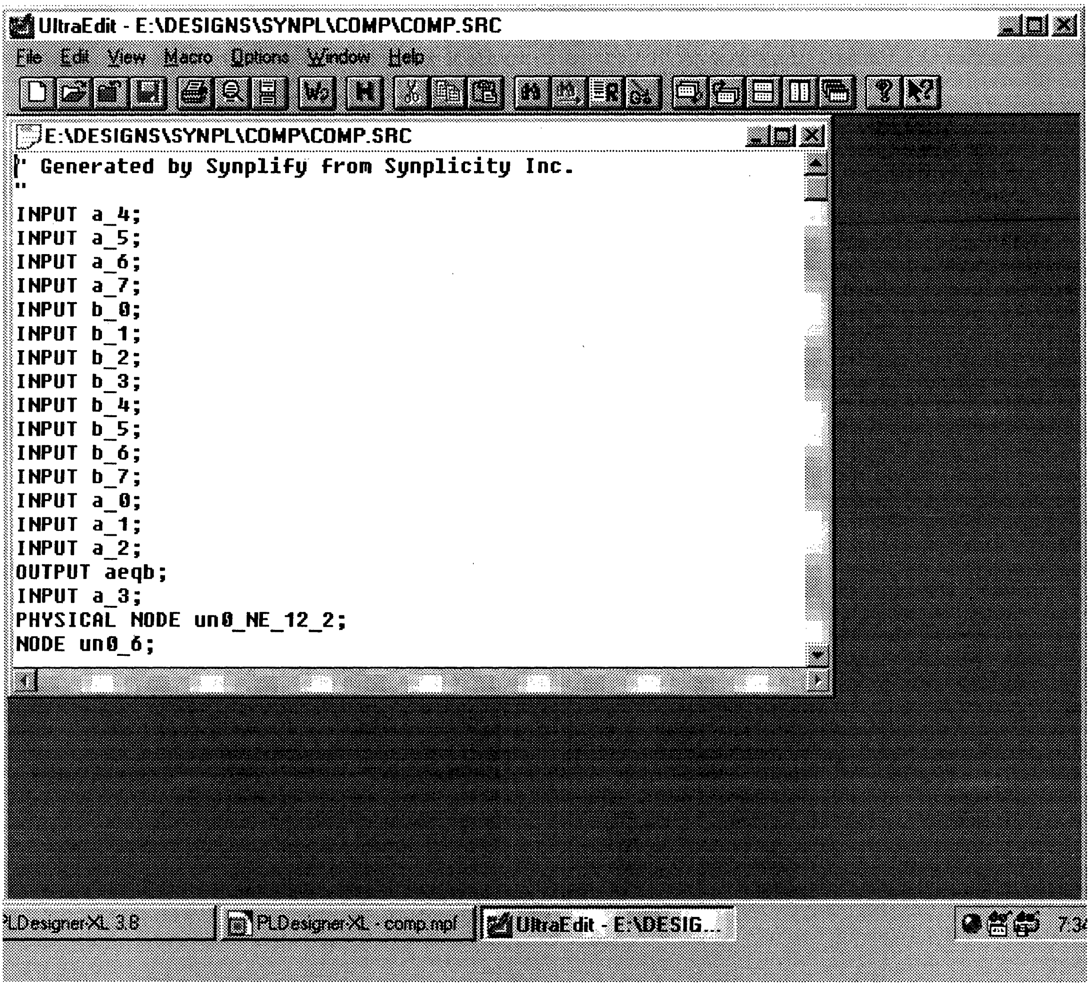
PLDesigner-XL can be used form the command line or its fraphical user interface (GUI).To use from the command line, enter

minc.script comparator

This produces comparator.doc, comparator01.rpt, comparator,npi, and comparator.j1 files. The comparator.j1 file is used by the programmer to program the CPLD.

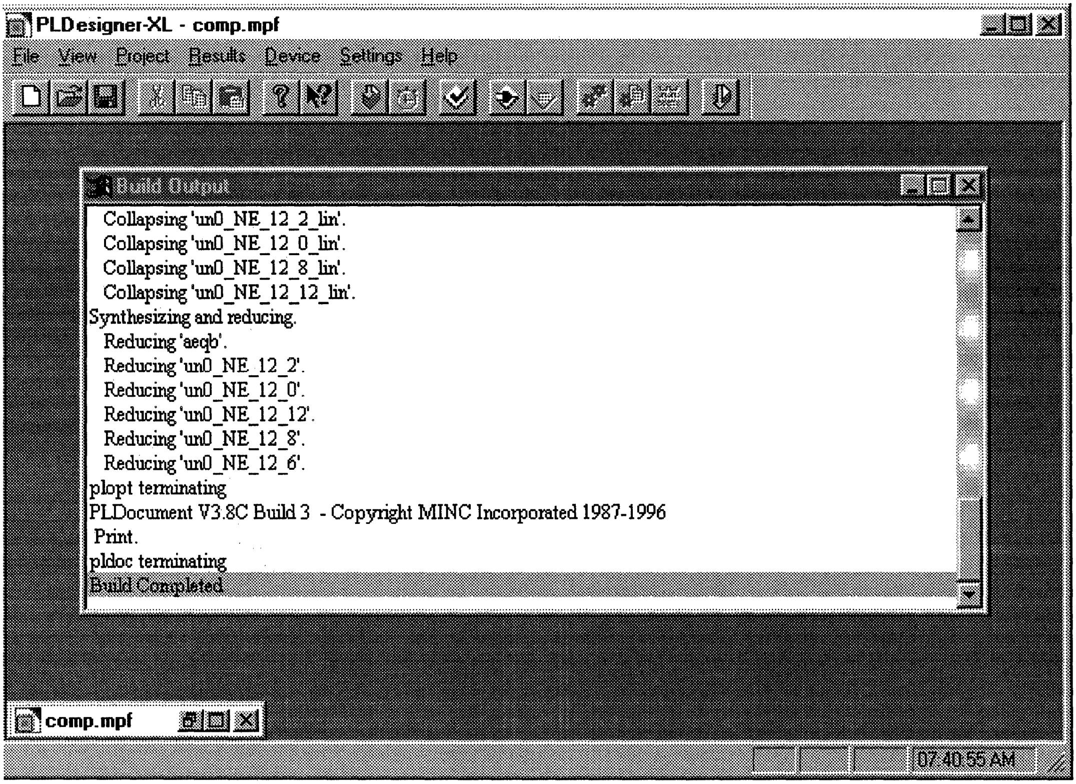
Graphical User Interface

The following provides the steps in using PLDesigner's GUI. It uses the src file generated by Synplicity.


UltraEdit - E:\DESIGNS\SYNPL\COMP\COMP.SRC
File Edit View Macro Options Window Help
E:\DESIGNS\SYNPL\COMP\COMP.SRC
PLDesignerXL 3.8
PLDesignerXL - comp.mpf
UltraEdit - E:\DESIG...

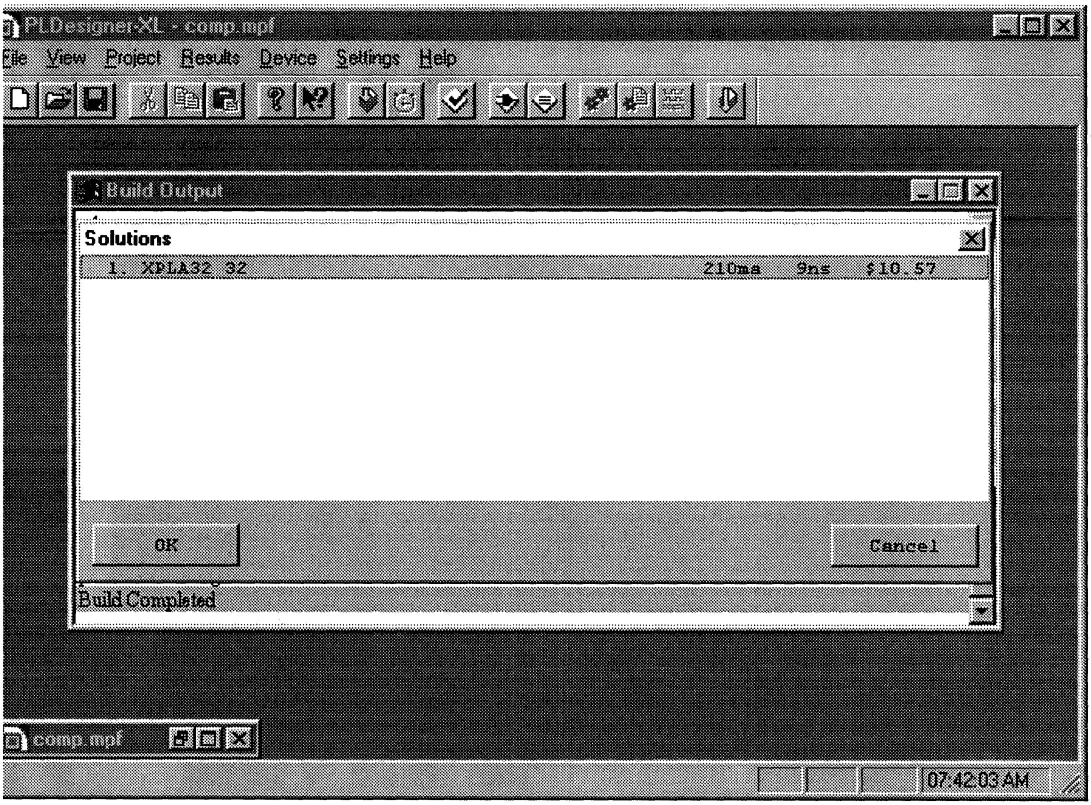
```
" Generated by Synplify from Synplicity Inc.
"
INPUT a_4;
INPUT a_5;
INPUT a_6;
INPUT a_7;
INPUT b_0;
INPUT b_1;
INPUT b_2;
INPUT b_3;
INPUT b_4;
INPUT b_5;
INPUT b_6;
INPUT b_7;
INPUT a_0;
INPUT a_1;
INPUT a_2;
OUTPUT aeqb;
INPUT a_3;
PHYSICAL NODE un0_NE_12_2;
NODE un0_6;
```

Selecting File Open comp allows the user to view/edit the source file.



```
  Collapsing 'un0_NE_12_2_lin'.
  Collapsing 'un0_NE_12_0_lin'.
  Collapsing 'un0_NE_12_8_lin'.
  Collapsing 'un0_NE_12_12_lin'.
Synthesizing and reducing.
  Reducing 'aeqb'.
  Reducing 'un0_NE_12_2'.
  Reducing 'un0_NE_12_0'.
  Reducing 'un0_NE_12_12'.
  Reducing 'un0_NE_12_8'.
  Reducing 'un0_NE_12_6'.
plopt terminating
PLDocument V3.8C Build 3  - Copyright MINC Incorporated 1987-1996
 Print.
pldoc terminating
Build Completed
```

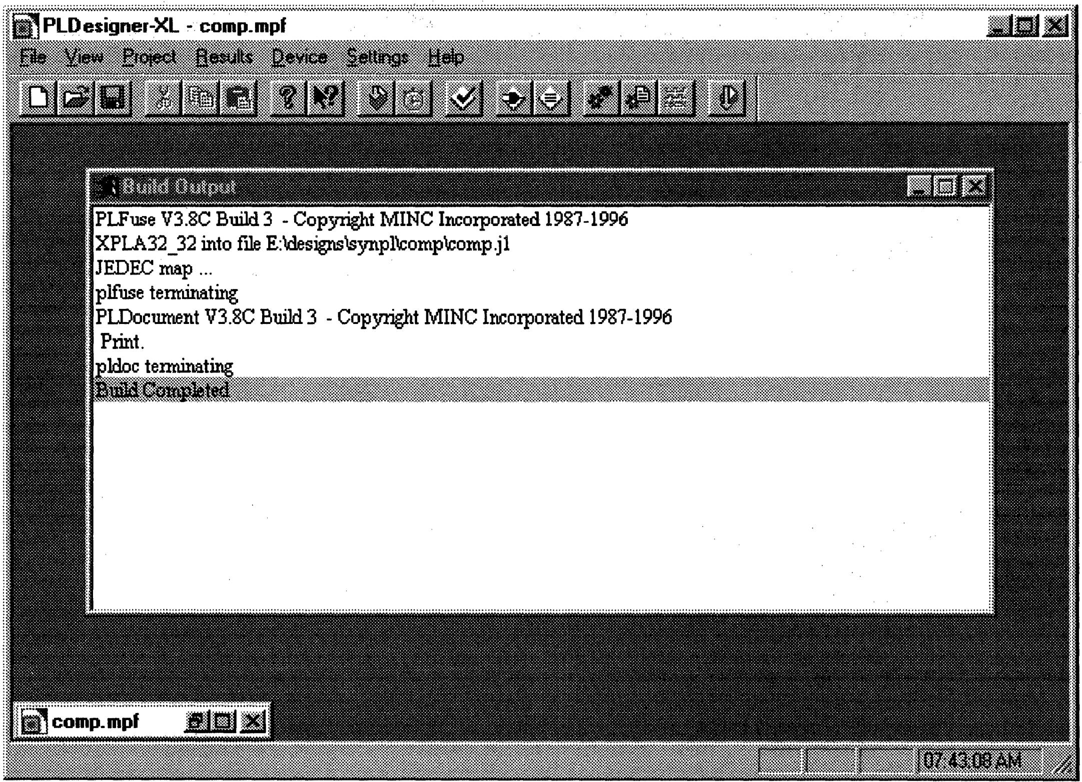
The user can Build All or Compile/Partition. This shows the results of the Build All step.



```
1. XPLA32 32                                  210ma    9ns    $10.57
```

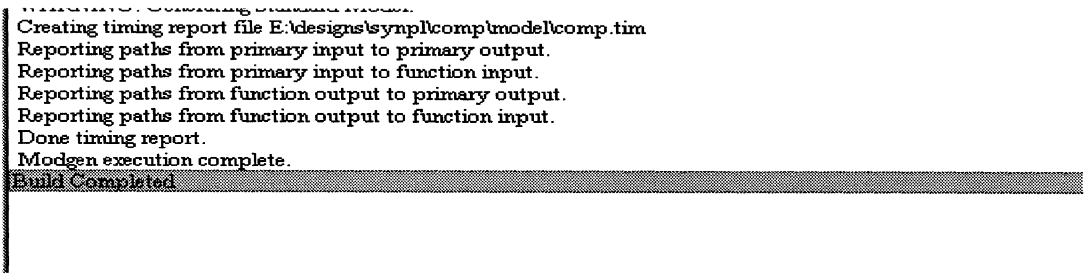
```
Build Completed
```

In this case the PZ3032 is a solution. The actual static power dissipation is under 70 microamps.



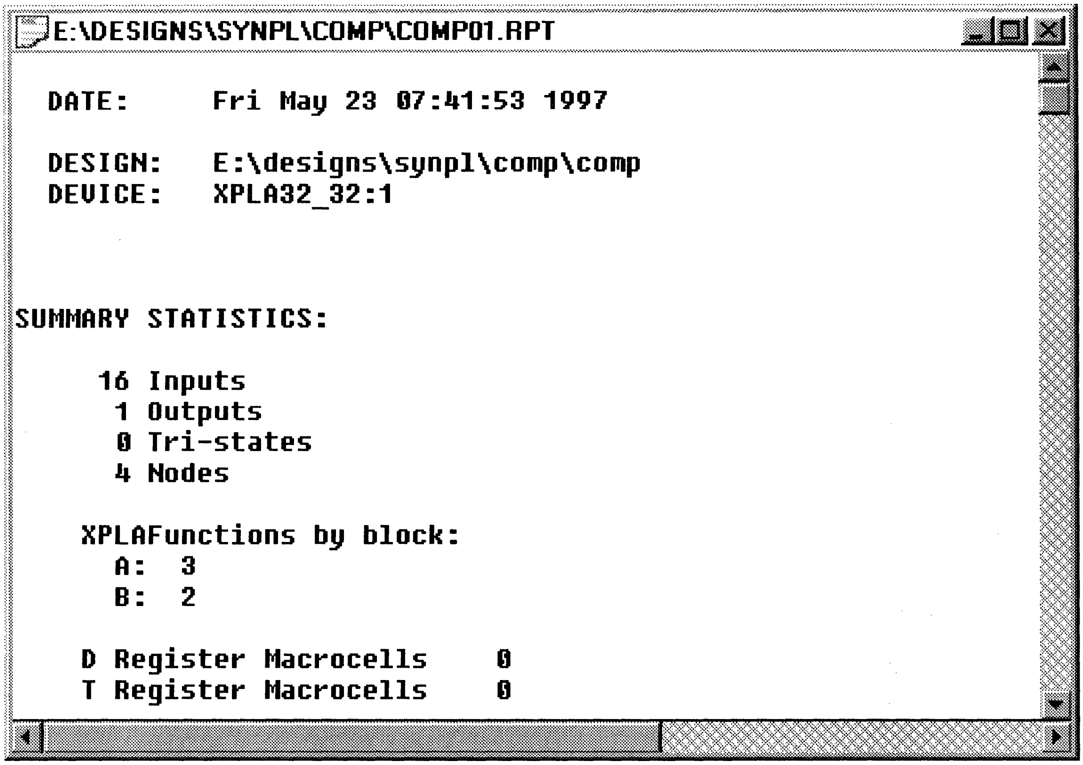
```
PLFuse V3.8C Build 3  - Copyright MINC Incorporated 1987-1996
XPLA32_32 into file E:\designs\synpl\comp\comp.j1
JEDEC map ...
plfuse terminating
PLDocument V3.8C Build 3  - Copyright MINC Incorporated 1987-1996
 Print.
pldoc terminating
Build Completed
```

After Build and selecting a solution, generate the fusemap, timing model, and timing report.



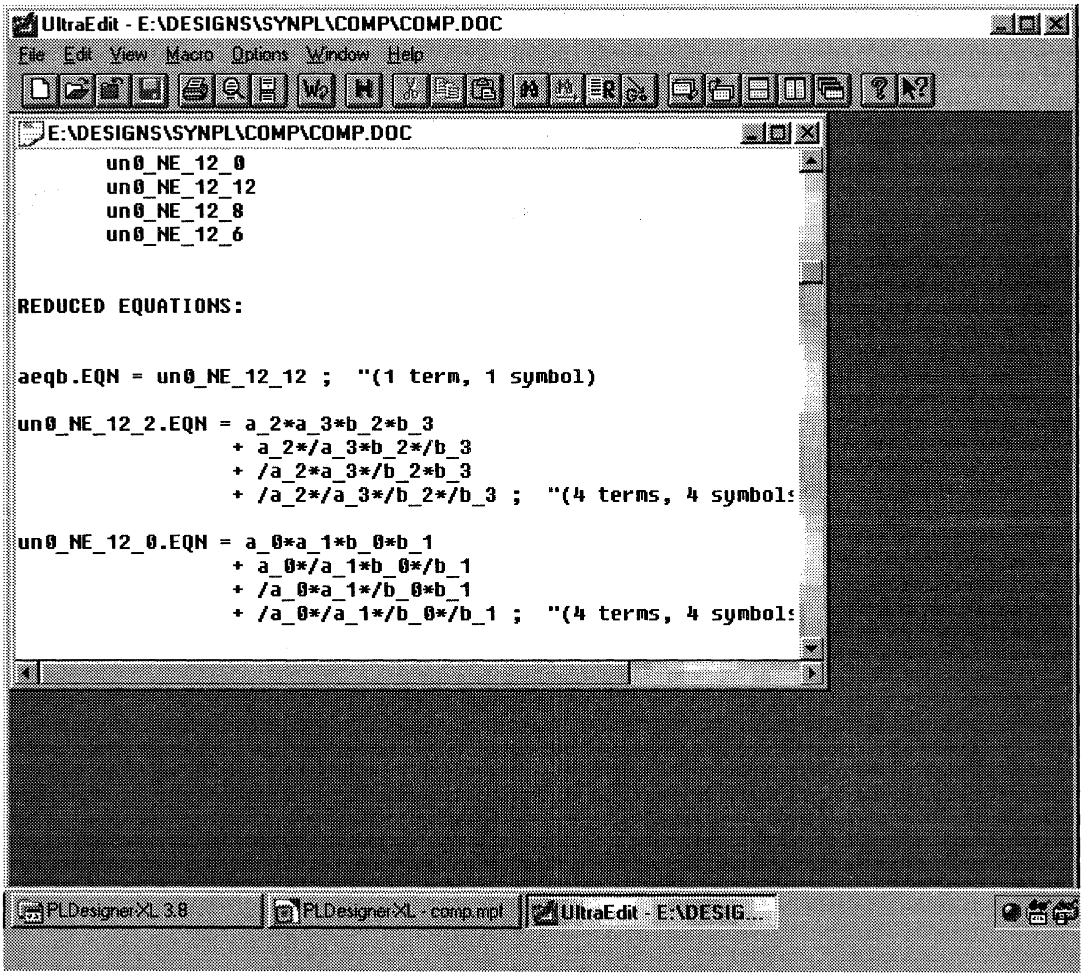
```
WARNING: Generating Standard Model.
Creating timing report file E:\designs\synpl\comp\model\comp.tim
Reporting paths from primary input to primary output.
Reporting paths from primary input to function input.
Reporting paths from function output to primary output.
Reporting paths from function output to function input.
Done timing report.
Modgen execution complete.
Build Completed
```

This shows the beginning of the report file detailing the device resources used.



E:\DESIGNS\SYNPL\COMP\COMP01.RPT

```
  DATE:      Fri May 23 07:41:53 1997

  DESIGN:    E:\designs\synpl\comp\comp
  DEVICE:    XPLA32_32:1



SUMMARY STATISTICS:

     16 Inputs
      1 Outputs
      0 Tri-states
      4 Nodes

    XPLAFunctions by block:
      A:  3
      B:  2

    D Register Macrocells    0
    T Register Macrocells    0
```



UltraEdit - E:\DESIGNS\SYNPL\COMP\COMP.DOC
File Edit View Macro Options Window Help
E:\DESIGNS\SYNPL\COMP\COMP.DOC
PLDesignerXL 3.8
PLDesignerXL - comp.mpf
UltraEdit - E:\DESIG...

```
       un0_NE_12_0
       un0_NE_12_12
       un0_NE_12_8
       un0_NE_12_6


REDUCED EQUATIONS:


aeqb.EQN = un0_NE_12_12 ;  "(1 term, 1 symbol)

un0_NE_12_2.EQN = a_2*a_3*b_2*b_3
                  + a_2*/a_3*b_2*/b_3
                  + /a_2*a_3*/b_2*b_3
                  + /a_2*/a_3*/b_2*/b_3 ;  "(4 terms, 4 symbol

un0_NE_12_0.EQN = a_0*a_1*b_0*b_1
                  + a_0*/a_1*b_0*/b_1
                  + /a_0*a_1*/b_0*b_1
                  + /a_0*/a_1*/b_0*/b_1 ;  "(4 terms, 4 symbol
```

The comp.doc report shows part of the equations used in the design.

Timing simulation

The minc script caused the creation of a directory called model in the project directory. This directory provides the static timing report, and several files for VHDL simulation with timing. The VHDL model is comparator.vhd. Delays are provided in comparator.sdf. Testbenches are given in comparator.tb, which can be renamed to comparator_tb.vhd. Essentially the same steps used in functional simulation are used in timing simulation. The major differences are that the stimulus needs to be re-input into the comparator_tb.vhd file, and some edits to the comparator.sdf file may be necessary to bind the correct instances.

The comparator_tb.vhd provides two entity/architecture pairs, using <design> and design>00 nomenclature.

```
--

-- Model automatically generated by Modgen Version 3.8

-- Generated from comparator

-- Date Fri Mar 28 07:36:28 1997

--


--

--  PS PZ5032-6A44 (template: XPLA32_32)

--  Package: JLCC

--




LIBRARY IEEE;

LIBRARY MINC_VHD;

USE IEEE.STD_LOGIC_1164.ALL;

USE MINC_VHD.PKG_VHD.ALL;

USE MINC_VHD.PKG_GATE.ALL;



ENTITY comparator00 IS
  GENERIC (
    tPD  : TIME := 6.00 ns;
    tCO  : TIME := 5.50 ns;
    tS   : TIME := 4.00 ns);
  PORT (
    AEQB : OUT STD_LOGIC;
    A_6, A_4, A_3, A_2, A_1, A_0, B_7, B_6, B_5, B_4, B_3,
    B_2, B_1, B_0, A_7, A_5 : IN STD_LOGIC);
END comparator00;

ARCHITECTURE comparator00_arch OF comparator00 IS
  SIGNAL pin_1, pin_2, pin_4, pin_5, pin_6, pin_7, pin_8, pin_9,
    pin_11, pin_12, pin_13, pin_14, pin_16, pin_17, pin_18,
    pin_43, pin_44, tmp34, tmp35, tmp36, tmp37, tmp38, tmp39,
    tmp40, tmp41, tmp42, tmp43, tmp44, tmp45, tmp46, tmp47,
    tmp48, tmp49, tmp50, tmp51, tmp52, tmp53, tmp54, tmp55,
    tmp56, tmp57, tmp58, tmp59, tmp60, tmp61, tmp62, tmp63,
    tmp64, tmp65, tmp66, tmp67 : STD_LOGIC := 'X';
BEGIN
PI00001: portin PORT MAP (pin_1, A_6);
PI00002: portin PORT MAP (pin_2, A_4);
PO00001: portout PORT MAP (AEQB, pin_4);
PI00003: portin PORT MAP (pin_5, A_3);
PI00004: portin PORT MAP (pin_6, A_2);
PI00005: portin PORT MAP (pin_7, A_1);
PI00006: portin PORT MAP (pin_8, A_0);
PI00007: portin PORT MAP (pin_9, B_7);
PI00008: portin PORT MAP (pin_11, B_6);
PI00009: portin PORT MAP (pin_12, B_5);
PI00010: portin PORT MAP (pin_13, B_4);
PI00011: portin PORT MAP (pin_14, B_3);
PI00012: portin PORT MAP (pin_16, B_2);
PI00013: portin PORT MAP (pin_17, B_1);
PI00014: portin PORT MAP (pin_18, B_0);
PI00015: portin PORT MAP (pin_43, A_7);
PI00016: portin PORT MAP (pin_44, A_5);
I00001: inv PORT MAP (tmp34, tmp35);
I00002: inv PORT MAP (tmp37, pin_18);
A00001: and2 PORT MAP (tmp36, pin_8, tmp37);
I00003: inv PORT MAP (tmp39, pin_8);
A00002: and2 PORT MAP (tmp38, tmp39, pin_18);
I00004: inv PORT MAP (tmp41, pin_17);
A00003: and2 PORT MAP (tmp40, pin_7, tmp41);
I00005: inv PORT MAP (tmp43, pin_7);
A00004: and2 PORT MAP (tmp42, tmp43, pin_17);
I00006: inv PORT MAP (tmp45, pin_16);
A00005: and2 PORT MAP (tmp44, pin_6, tmp45);
```

```
I00007: inv PORT MAP (tmp47, pin_6);
A00006: and2 PORT MAP (tmp46, tmp47, pin_16);
I00008: inv PORT MAP (tmp49, pin_14);
A00007: and2 PORT MAP (tmp48, pin_5, tmp49);
I00009: inv PORT MAP (tmp51, pin_5);
A00008: and2 PORT MAP (tmp50, tmp51, pin_14);
I00010: inv PORT MAP (tmp53, pin_13);
A00009: and2 PORT MAP (tmp52, pin_2, tmp53);
I00011: inv PORT MAP (tmp55, pin_2);
A00010: and2 PORT MAP (tmp54, tmp55, pin_13);
I00012: inv PORT MAP (tmp57, pin_12);
A00011: and2 PORT MAP (tmp56, pin_44, tmp57);
I00013: inv PORT MAP (tmp59, pin_44);
A00012: and2 PORT MAP (tmp58, tmp59, pin_12);
I00014: inv PORT MAP (tmp61, pin_11);
A00013: and2 PORT MAP (tmp60, pin_1, tmp61);
I00015: inv PORT MAP (tmp63, pin_1);
A00014: and2 PORT MAP (tmp62, tmp63, pin_11);
I00016: inv PORT MAP (tmp65, pin_9);
A00015: and2 PORT MAP (tmp64, pin_43, tmp65);
I00017: inv PORT MAP (tmp67, pin_43);
A00016: and2 PORT MAP (tmp66, tmp67, pin_9);
O00001: or16 PORT MAP (tmp35, tmp36, tmp38, tmp40, tmp42, tmp44,
  tmp46, tmp48, tmp50, tmp52, tmp54, tmp56, tmp58, tmp60, tmp62,
  tmp64, tmp66);
B00001: mbuf
  GENERIC MAP (tpd_INP_OUTP=>tPD)
  PORT MAP (pin_4, tmp34);
END comparator00_arch;


LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;

PACKAGE comparator00_pkg IS
```

```
  COMPONENT comparator00
    GENERIC (
      tPD  : TIME := 6.00 ns;
      tCO  : TIME := 5.50 ns;
      tS   : TIME := 4.00 ns);
    PORT (
      AEQB : OUT STD_LOGIC;
      A_6, A_4, A_3, A_2, A_1, A_0, B_7, B_6, B_5, B_4,
      B_3, B_2, B_1, B_0, A_7, A_5 : IN STD_LOGIC);
  END COMPONENT;
END comparator00_pkg;



--
-- Top Level for design in comparator
--




LIBRARY IEEE;
LIBRARY MINC_VHD;
USE IEEE.STD_LOGIC_1164.ALL;
USE MINC_VHD.PKG_VHD.ALL;
USE MINC_VHD.PKG_GATE.ALL;
USE WORK.comparator00_pkg.all;


ENTITY comparator IS
  PORT (
    AEQB : OUT STD_LOGIC;
    A_6, A_4, A_3, A_2, A_1, A_0, B_7, B_6, B_5, B_4, B_3,
    B_2, B_1, B_0, A_7, A_5 : IN STD_LOGIC);
END comparator;

ARCHITECTURE comparator_arch OF comparator IS
```

```
BEGIN
U00001: comparator00 PORT MAP (AEQB=>AEQB, A_6=>A_6, A_4=>A_4,
  A_3=>A_3, A_2=>A_2, A_1=>A_1, A_0=>A_0, B_7=>B_7, B_6=>B_6,
  B_5=>B_5, B_4=>B_4, B_3=>B_3, B_2=>B_2, B_1=>B_1, B_0=>B_0,
  A_7=>A_7, A_5=>A_5);
END comparator_arch;



LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;

PACKAGE comparator_pkg IS
  COMPONENT comparator
    PORT (
      AEQB : OUT STD_LOGIC;
      A_6, A_4, A_3, A_2, A_1, A_0, B_7, B_6, B_5, B_4,
      B_3, B_2, B_1, B_0, A_7, A_5 : IN STD_LOGIC);
  END COMPONENT;
END comparator_pkg;
```

The revised testbench generated by PLDesigner is given below.

```
--
-- *** VHDL TEST BENCH ***
-- Model automatically generated by Modgen Version 3.8
-- Generated from comparator
-- Date: Fri Mar 28 07:36:28 1997
--

library ieee;
use ieee.std_logic_1164.all;
use work.comparator00_pkg.all;

entity comparator00_tb is
end comparator00_tb;
```

```
architecture comparator00_arch_tb of comparator00_tb is
  --outputs:
  signal AEQB : STD_LOGIC;
  --inputs:
  signal A_6, A_4, A_3, A_2, A_1, A_0, B_7, B_6, B_5, B_4,
    B_3, B_2, B_1, B_0, A_7, A_5 : STD_LOGIC;
  constant period : TIME := 100 ns;
  begin

  --PS PZ5032-6A44 (template: XPLA32_32)
  U00 : comparator00 port map (AEQB=>AEQB, A_6=>A_6, A_4=>
    A_4, A_3=>A_3, A_2=>A_2, A_1=>A_1, A_0=>A_0, B_7=>B_7, B_6=>
    B_6, B_5=>B_5, B_4=>B_4, B_3=>B_3, B_2=>B_2, B_1=>B_1, B_0=>
    B_0, A_7=>A_7, A_5=>A_5);

  process
  begin
    --Initialization (Time=0)
    A_6<=’0’;A_4<=’0’;A_3<=’0’;A_2<=’0’;A_1<=’0’;A_0<=’0’;
    B_7<=’0’;B_6<=’0’;B_5<=’0’;B_4<=’0’;B_3<=’0’;B_2<=’0’;
    B_1<=’0’;B_0<=’0’;A_7<=’0’;A_5<=’0’;

    --Place stimulus below (Ex: signal1<=’1’; wait for period;)

A_6 <= ‘1’ ; wait for 100 ns ;
B_6 <= ‘1’ ; wait for 100 ns ;
A_4 <= ‘1’ ; wait for 100 ns ;
B_4 <= ‘1’ ; wait for 100 ns ;
A_2 <= ‘1’ ; wait for 100 ns ;
B_2 <= ‘1’ ; wait for 100 ns ;
A_1 <= ‘1’ ; wait for 100 ns ;
B_1 <= ‘1’ ; wait for 100 ns ;
A_5 <= ‘1’ ; wait for 100 ns ;
B_5 <= ‘1’ ; wait for 100 ns ;
```

```
A_7 <= ‘1’ ; wait for 100 ns ;
B_7 <= ‘1’ ; wait for 100 ns ;

    wait;
  end process;
end comparator00_arch_tb;



library ieee;
use ieee.std_logic_1164.all;
use work.comparator_pkg.all;

entity comparator_tb is
end comparator_tb;

architecture comparator_arch_tb of comparator_tb is
  --outputs:
  signal AEQB : STD_LOGIC;
  --inputs:
  signal A_6, A_4, A_3, A_2, A_1, A_0, B_7, B_6, B_5, B_4,
    B_3, B_2, B_1, B_0, A_7, A_5 : STD_LOGIC;
  constant period : TIME := 100 ns;
  begin

  --Top Level Design
  U0 : comparator port map (AEQB=>AEQB, A_6=>A_6, A_4=>A_4,
    A_3=>A_3, A_2=>A_2, A_1=>A_1, A_0=>A_0, B_7=>B_7, B_6=>
    B_6, B_5=>B_5, B_4=>B_4, B_3=>B_3, B_2=>B_2, B_1=>B_1, B_0=>
    B_0, A_7=>A_7, A_5=>A_5);

  process
  begin
    --Initialization (Time=0)
    A_6<=’0’;A_4<=’0’;A_3<=’0’;A_2<=’0’;A_1<=’0’;A_0<=’0’;
    B_7<=’0’;B_6<=’0’;B_5<=’0’;B_4<=’0’;B_3<=’0’;B_2<=’0’;
    B_1<='0';B_0<='0';A_7<='0';A_5<='0';

    --Place stimulus below (Ex: signal1<='1'; wait for period;)



    wait;
  end process;
end comparator_arch_tb;
```

The delay file is given below:

```
//Model automatically generated by Modgen
//Generated from comparator



(DELAYFILE
(SDFVERSION “OVI 2.1”)
(DESIGN “comparator”)
(DATE “Fri Mar 28 07:36:28 1997”)
(VENDOR “Minc, Inc.”)
(PROGRAM “Modgen”)
(VERSION “Version 3.8”)
(DIVIDER /)
(TIMESCALE 1ns)




//Manufacturer:  PS
//Part:          PZ5032-6A44
(CELL
(CELLTYPE “mbuf”)
(INSTANCE U00001/B00001)
(DELAY
(ABSOLUTE
//tPD
(IOPATH INP OUTP (6.00:6.00:6.00))
)
)
)
)
```

The prefix in the INSTANCE must agree with the unit instantiated unit invoked in the entity/architecure pair from the testbench

Philips recognizes the trademarks of Synplicity, Inc. and Minc, Inc.

Section 7
Package information

CONTENTS

INTRODUCTION

There is no soldering method that is ideal for all IC packages. Wave soldering is often preferred when though-hole and surface mounted components are mixed on one printed-circuit board. However, wave soldering is not always suitable for surface mounted ICs, or for printed-circuits with high population densities. In these situations reflow soldering is often used.

This text gives a very brief insight to a complex technology. A more in-depth account of soldering ICs can be found in our *"IC Package Databook"* (order code 9398 652 90011).

THROUGH-HOLE MOUNTED PACKAGES

Table 1. Types of through-hole mounted packages

TYPE	DESCRIPTION
DIP	plastic dual in-line package
SDIP	plastic shrink dual in-line package
HDIP	plastic heat-dissipating dual in-line package
DBS	plastic dual in-line bent from a single in-line package
SIL	plastic single in-line package

Soldering by dipping or wave

The maximum permissible temperature of the solder is 260°C; solder at this temperature must not be in contact with the joint for more than 5 seconds. The total contact time of successive solder waves must not exceed 5 seconds.

The device may be mounted to the seating plane, but the temperature of the plastic body must not exceed the specified maximum storage temperature ($T_{stg\ max}$). If the printed-circuit board has been pre-heated, forced cooling may be necessary immediately after soldering to keep the temperature within the permissible limit.

Repairing soldered joints

Apply a low voltage soldering iron (less than 24V) to the lead(s) of the package, below the seating plane or not more than 2mm above it. If the temperature of the soldering iron bit is less than 300°C it may remain in contact for up to 10 seconds. If the bit temperature is between 300 and 400°C, contact may be up to 5 seconds.

SURFACE MOUNTED PACKAGES

Table 2. Types of surface mounted packages

TYPE	DESCRIPTION
SO	plastic small outline package
SSOP	plastic shrink small outline package
TSSOP	plastic thin shrink small outline package
VSO	plastic very small outline package
QFP	plastic quad flat package
LQFP	plastic low profile quad flat package
SQFP	plastic shrink quad flat package
TQFP	plastic thin quad flat package
PLCC	plastic leaded chip carrier

Reflow soldering

Reflow soldering techniques are suitable for all SMD packages, ease of soldering varies with the type of package as indicated in Table 3.

The choice of heating method may be indluenced by larger plastic packages (QFP or PLCC with 44 leads, or more). If infrared or vapor phase heating is used and the large packages are not absolutely dry (less than 0.1% moisture content by weight), vaporization of the small amount of moisture in them can cause cracking of the plastic body. For more information on moisture prevention, refer to the Drypack chapter in our *"Quality Reference Manual"* (order code 9398 510 63011).

Reflow soldering requires solder paste (a suspension of fine solder particles, flux and binding agent) to be applied to the printed-circuit board by screen printing, stenciling or pressure-syringe dispensing before package placement.

Several techniques exist for reflowing; for example, thermal conduction by heated belt. Dwell times vary between 50 and 300 seconds depending on heating method. Typical reflow temperatures range from 215 to 250°C.

Preheating is necessary to dry the paste and evaporate the binding agent. Preheating duration: 45 minutes at 45°C.

Table 3. Suitability of surface mounted packages for various soldering methods

Rating from 'a' to 'd': **'a' indicates most suitable** (soldering is not difficult); **'d' indicates least suitable** (soldering is achievable with difficulty).

TYPE	REFLOW METHOD					DOUBLE WAVE METHOD
	INFRARED	HOT BELT	HOT GAS	VAPOR PHASE	RESISTANCE	
SO	a	a	a	a	d	a
SSOP	a	a	a	c	d	c
TSSOP	b	b	b	c	d	d
VSO	b	b	a	b	a	b
QFP	b	b	a	c	a	c
LQFP	b	b	a	c	d	d
SQFP	b	b	a	c	d	d
TQFP	b	b	a	c	d	d
PLCC	c	b	b	d	d	b

Wave soldering

Wave soldering is **not** recommended for SSOP, TSSOP, QFP, LQFP, SQFP or TQFP packages. This is because of the likelihood of solder bridging due to closely-spaced leads and the possibility of incomplete solder penetration in multi-lead devices.

If wave soldering cannot be avoided, the following conditions must be observed:

- A double-wave (a turbulent wave with high upward pressure followed by a smooth laminar wave) soldering technique should be used.
- For SSOP, TSSOP and VSO packages, the longitudinal axis of the package footprint must be parallel to the solder flow **and** must incorporate solder theives at the downstream end.
- For QFP, LQFP and TQFP packages, the footprint must be at and angle of 45° to the board direction **and** must incorporate solder thieves downstream and at the side corners.

Even with these conditions, only consider wave soldering for the following package types:

- SO
- VSO
- PLCC
- SSOP **only with body width 4.4mm**, e.g., SSOP16 (SOT369-1) or SSOP20 (SOT266-1).
- QFP **except** QFP52 (SOT379-1), QFP100 (SOT317-1, SOT317-2 and SOT382-1) and QFP160 (SOT322-1); these are **not** suitable for wave soldering.
- LQFP **except** LQFP32 (SOT401-1), LQFP48 (SOT313-1, SOT313-2), LQFP64 (SOT314-2), LQFP80 (SOT315-1); these are **not** suitable for wave soldering.
- TQFP **except** TQFP64 (SOT357-1), TQFP80 (SOT375-1) and TQFP100 (SOT386-1); these are **not** suitable for wave soldering.

SQFP are **not** suitable for wave soldering.

During placement and before soldering, the package must be fixed with a droplet of adhesive. The adhesive can be applied by screen printing, pin transfer or syringe dispensing. The package can be soldered after the adhesive is cured.

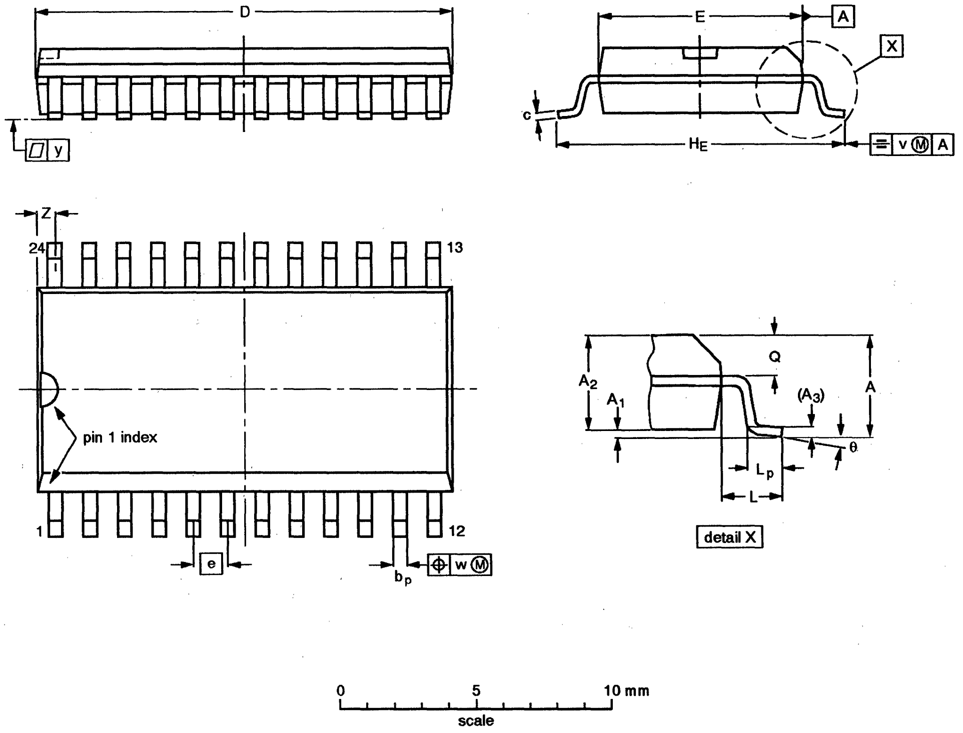
Maximum permissible solder temperature is 260°C, and maximum duration of package immersion in solder is 10 seconds, if cooled to less than 150°C within 6 seconds. Typical dwell time is 4 seconds at 250°C.

A mildly-activated flux will eliminate the need for removal of corrosive residues in most applications.

Repairing soldered joints

Fix the component by first soldering two diagonally-opposite end leads. Use only a low voltage soldering iron (less than 24V) applied to the flat part of the lead. Contact time must be limited to 10 seconds at up to 300°C. When using a dedicated tool, all other leads can be soldered in one operation within 2 to 5 seconds at between 270 and 320°C.

SO24: plastic small outline package; 24 leads; body width 7.5 mm SOT137-1



DIMENSIONS (inch dimensions are derived from the original mm dimensions)

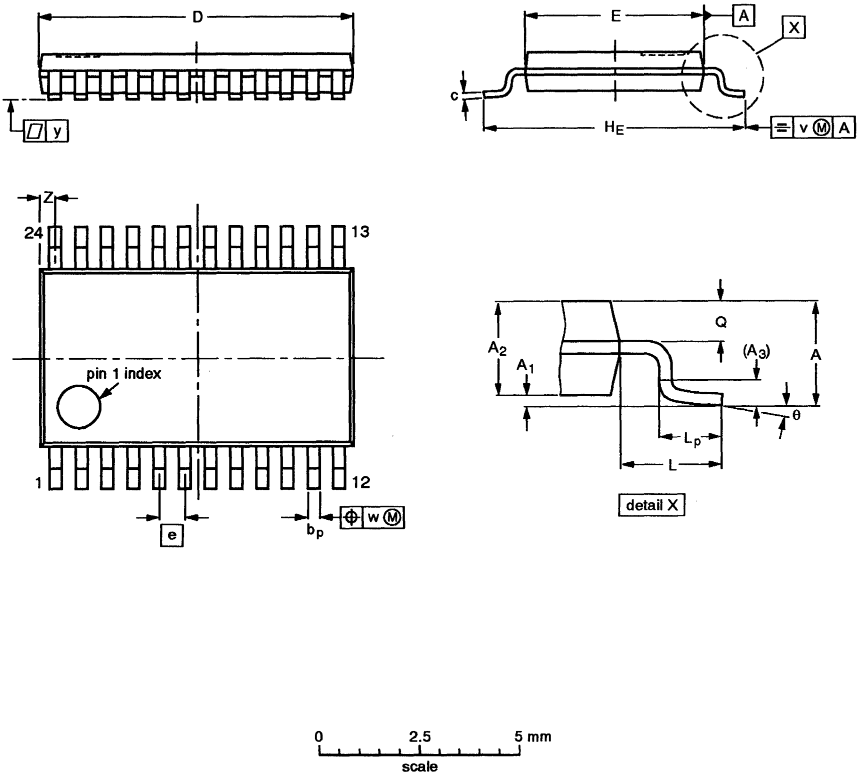
UNIT	A max.	A_1	A_2	A_3	b_p	c	D[1]	E[1]	e	H_E	L	L_p	Q	v	w	y	Z[1]	θ
mm	2.65	0.30 0.10	2.45 2.25	0.25	0.49 0.36	0.32 0.23	15.6 15.2	7.6 7.4	1.27	10.65 10.00	1.4	1.1 0.4	1.1 1.0	0.25	0.25	0.1	0.9 0.4	8° 0°
inches	0.10	0.012 0.004	0.096 0.089	0.01	0.019 0.014	0.013 0.009	0.61 0.60	0.30 0.29	0.050	0.42 0.39	0.055	0.043 0.016	0.043 0.039	0.01	0.01	0.004	0.035 0.016	

Note

1. Plastic or metal protrusions of 0.15 mm maximum per side are not included.

OUTLINE VERSION	REFERENCES				EUROPEAN PROJECTION	ISSUE DATE
	IEC	JEDEC	EIAJ			
SOT137-1	075E05	MS-013AD				~~92-11-17~~ 95-01-24

TSSOP24: plastic thin shrink small outline package; 24 leads; body width 4.4 mm SOT355-1



DIMENSIONS (mm are the original dimensions)

UNIT	A max.	A_1	A_2	A_3	b_p	c	$D^{(1)}$	$E^{(2)}$	e	H_E	L	L_p	Q	v	w	y	$Z^{(1)}$	θ
mm	1.10	0.15 0.05	0.95 0.80	0.25	0.30 0.19	0.2 0.1	7.9 7.7	4.5 4.3	0.65	6.6 6.2	1.0	0.75 0.50	0.4 0.3	0.2	0.13	0.1	0.5 0.2	8° 0°

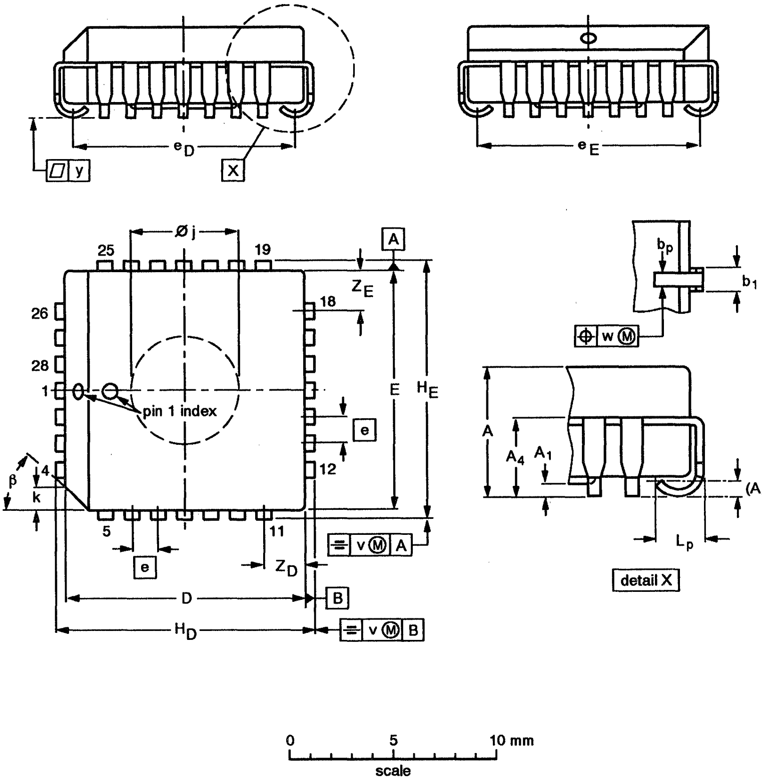
Notes

1. Plastic or metal protrusions of 0.15 mm maximum per side are not included.
2. Plastic interlead protrusions of 0.25 mm maximum per side are not included.

OUTLINE VERSION	REFERENCES				EUROPEAN PROJECTION	ISSUE DATE
	IEC	JEDEC	EIAJ			
SOT355-1		MO-153AD				~~93-06-16~~ 95-02-04

Package outlines

PLCC28: plastic leaded chip carrer; 28 leads; pedestal SOT261-3



DIMENSIONS (millimetre dimensions are derived from the original inch dimensions)

UNIT	A	A_1 min.	A_3	A_4 max.	b_p	b_1	$D^{(1)}$	$E^{(1)}$	e	e_D	e_E	H_D	H_E	k	Øj	L_p	v	w	y	$Z_D^{(1)}$ max.	$Z_E^{(1)}$ max.	β
mm	4.57 4.19	0.13	0.25	3.05	0.53 0.33	0.81 0.66	11.58 11.43	11.58 11.43	1.27	10.92 9.91	10.92 9.91	12.57 12.32	12.57 12.32	1.22 1.07	5.69 5.54	1.44 1.02	0.18	0.18	0.10	2.06	2.06	45°
inches	0.180 0.165	0.005	0.01	0.12	0.021 0.013	0.032 0.026	0.456 0.450	0.456 0.450	0.05	0.430 0.390	0.430 0.390	0.495 0.485	0.495 0.485	0.048 0.042	0.224 0.218	0.057 0.040	0.007	0.007	0.004	0.081	0.081	

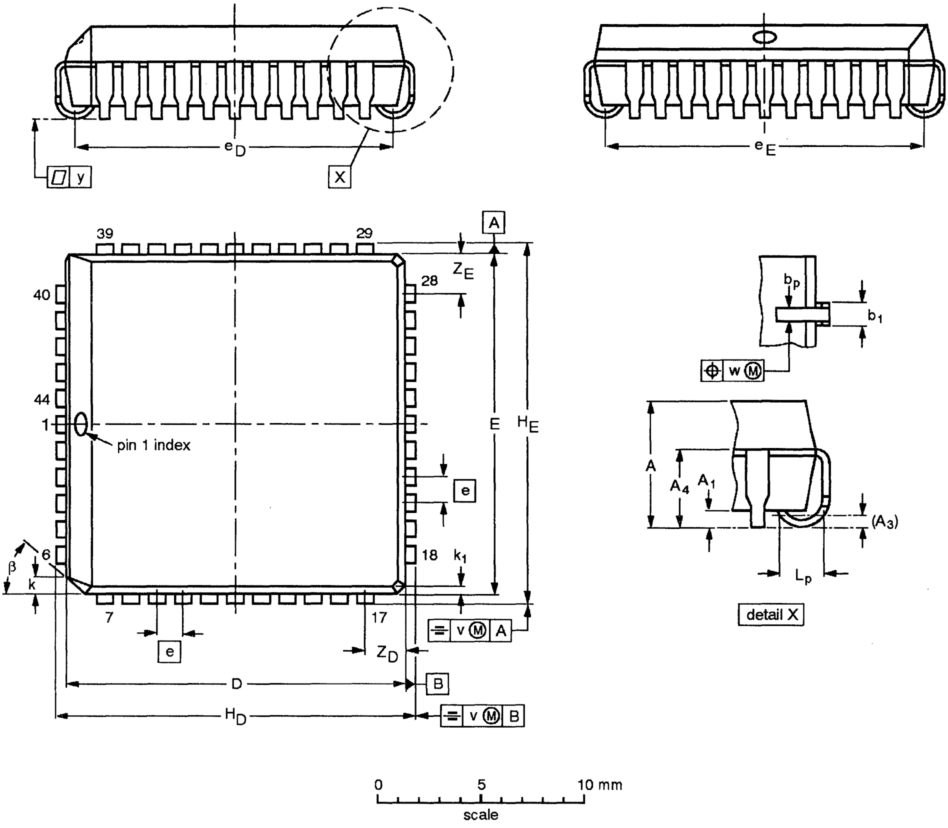
Note

1. Plastic or metal protrusions of 0.01 inches maximum per side are not included.

OUTLINE VERSION	REFERENCES				EUROPEAN PROJECTION	ISSUE DATE
	IEC	JEDEC	EIAJ			
SOT261-3		MO-047AB				~~92-11-17~~ 95-02-25

Package outlines

PLCC44: plastic leaded chip carrier; 44 leads SOT187-2

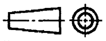


DIMENSIONS (millimetre dimensions are derived from the original inch dimensions)

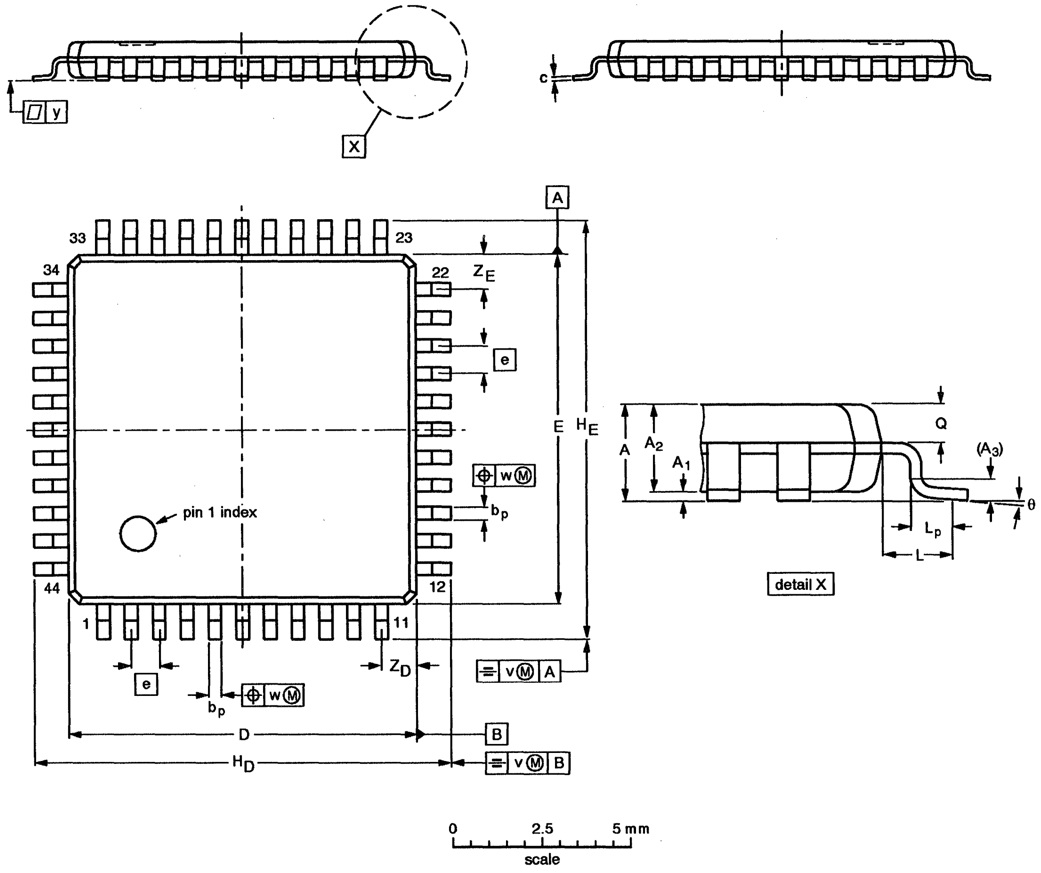
UNIT	A	A_1 min.	A_3	A_4 max.	b_p	b_1	$D^{(1)}$	$E^{(1)}$	e	e_D	e_E	H_D	H_E	k	k_1 max.	L_p	v	w	y	$Z_D^{(1)}$ max.	$Z_E^{(1)}$ max.	β
mm	4.57 4.19	0.51	0.25	3.05	0.53 0.33	0.81 0.66	16.66 16.51	16.66 16.51	1.27	16.00 14.99	16.00 14.99	17.65 17.40	17.65 17.40	1.22 1.07	0.51	1.44 1.02	0.18	0.18	0.10	2.16	2.16	45^o
inches	0.180 0.165	0.020	0.01	0.12	0.021 0.013	0.032 0.026	0.656 0.650	0.656 0.650	0.05	0.630 0.590	0.630 0.590	0.695 0.685	0.695 0.685	0.048 0.042	0.020	0.057 0.040	0.007	0.007	0.004	0.085	0.085	

Note

1. Plastic or metal protrusions of 0.01 inches maximum per side are not included.

OUTLINE VERSION	REFERENCES				EUROPEAN PROJECTION	ISSUE DATE
	IEC	JEDEC	EIAJ			
SOT187-2	112E10	MO-047AC				~~92-11-17~~ 95-02-25

TQFP44: plastic thin quad flat package; 44 leads; body 10 x 10 x 1.0 mm SOT376-1



DIMENSIONS (mm are the original dimensions)

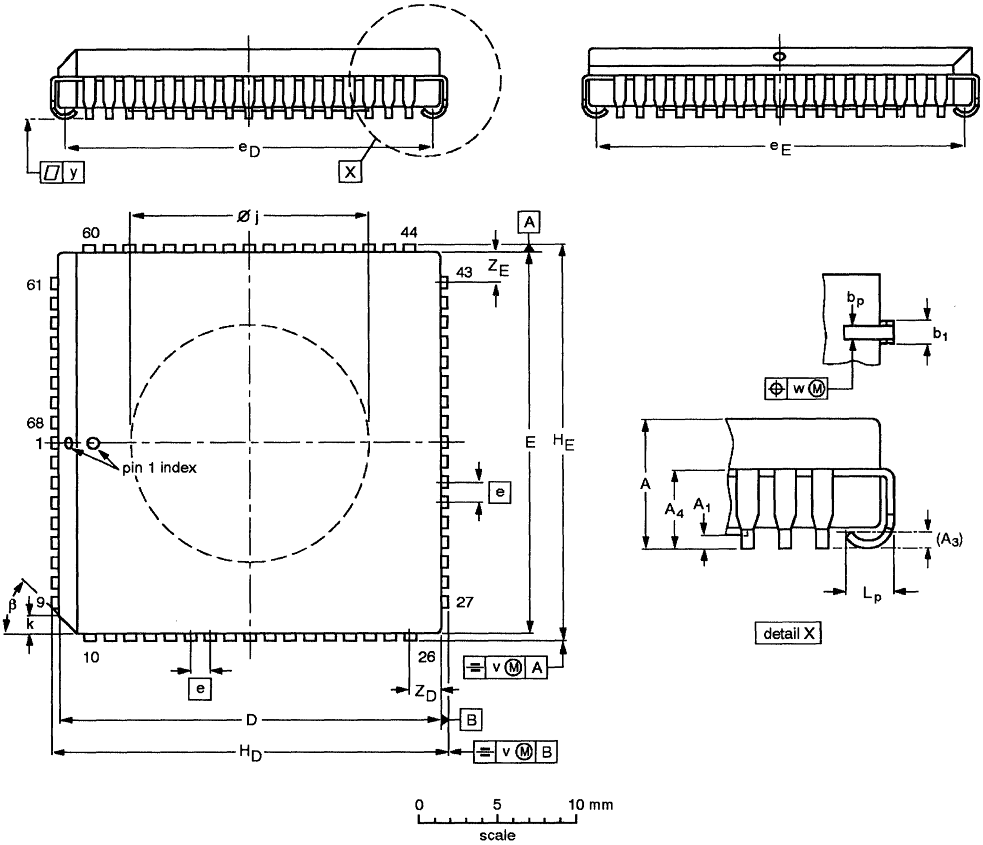
UNIT	A max.	A_1	A_2	A_3	b_p	c	$D^{(1)}$	$E^{(1)}$	e	H_D	H_E	L	L_p	Q	v	w	y	$Z_D^{(1)}$	$Z_E^{(1)}$	θ
mm	1.2	0.15 0.05	1.05 0.95	0.25	0.45 0.30	0.18 0.12	10.1 9.9	10.1 9.9	0.8	12.15 11.85	12.15 11.85	1.0	0.75 0.45	0.50 0.36	0.2	0.2	0.1	1.2 0.8	1.2 0.8	7° 0°

Note

1. Plastic or metal protrusions of 0.25 mm maximum per side are not included.

OUTLINE VERSION	REFERENCES				EUROPEAN PROJECTION	ISSUE DATE
	IEC	JEDEC	EIAJ			
SOT376-1						~~95-05-22~~ 96-04-02

PLCC68: plastic leaded chip carrier; 68 leads; pedestal SOT188-3



DIMENSIONS (millimetre dimensions are derived from the original inch dimensions)

UNIT	A	A_1 min.	A_3	A_4 max.	b_p	b_1	$D^{(1)}$	$E^{(1)}$	e	e_D	e_E	H_D	H_E	k	$Ø_j$	L_p	v	w	y	$Z_D^{(1)}$ max.	$Z_E^{(1)}$ max.	β
mm	4.57 4.19	0.13	0.25	3.05	0.53 0.33	0.81 0.66	24.33 24.13	24.33 24.13	1.27	23.62 22.61	23.62 22.61	25.27 25.02	25.27 25.02	1.22 1.07	15.34 15.19	1.44 1.02	0.18	0.18	0.10	2.06	2.06	45°
inches	0.180 0.165	0.005	0.01	0.12	0.021 0.013	0.032 0.026	0.958 0.950	0.958 0.950	0.05	0.930 0.890	0.930 0.890	0.995 0.985	0.995 0.985	0.048 0.042	0.604 0.598	0.057 0.040	0.007	0.007	0.004	0.081	0.081	

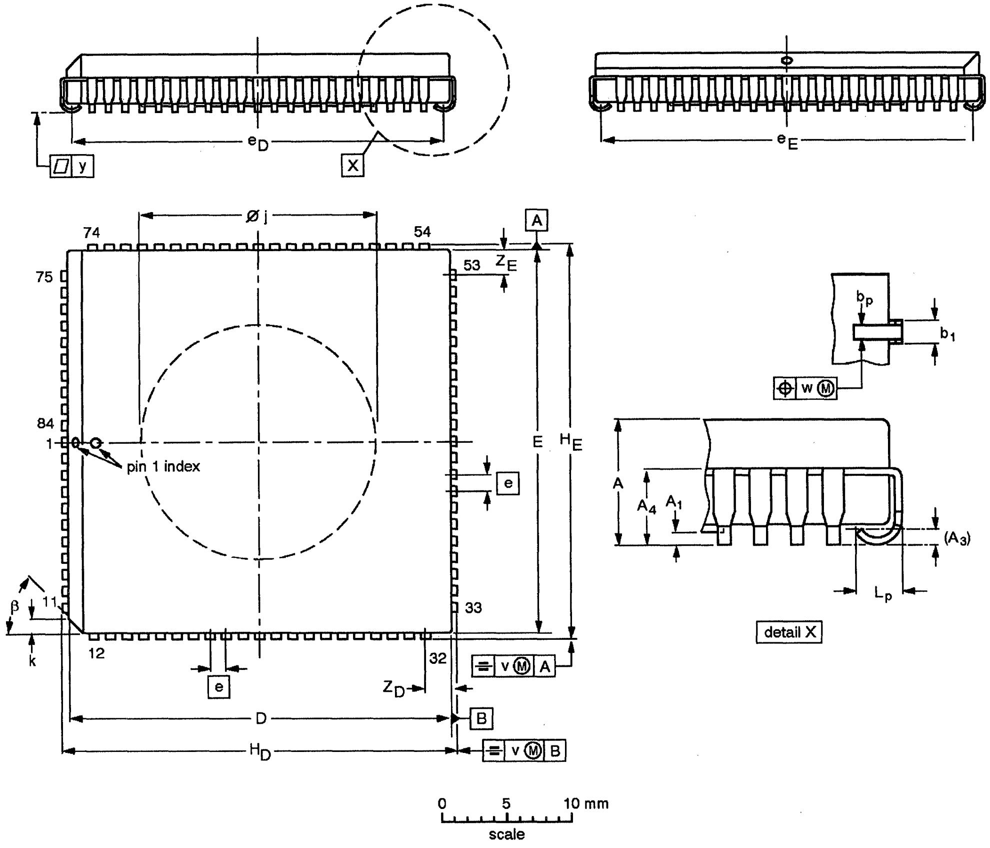
Note

1. Plastic or metal protrusions of 0.01 inches maximum per side are not included.

OUTLINE VERSION	REFERENCES				EUROPEAN PROJECTION	ISSUE DATE
	IEC	JEDEC	EIAJ			
SOT188-3	112E10	MO-047AE				~~92-11-17~~ 95-02-25

Package outlines

PLCC84: plastic leaded chip carrier; 84 leads; pedestal SOT189-3



DIMENSIONS (millimetre dimensions are derived from the original inch dimensions)

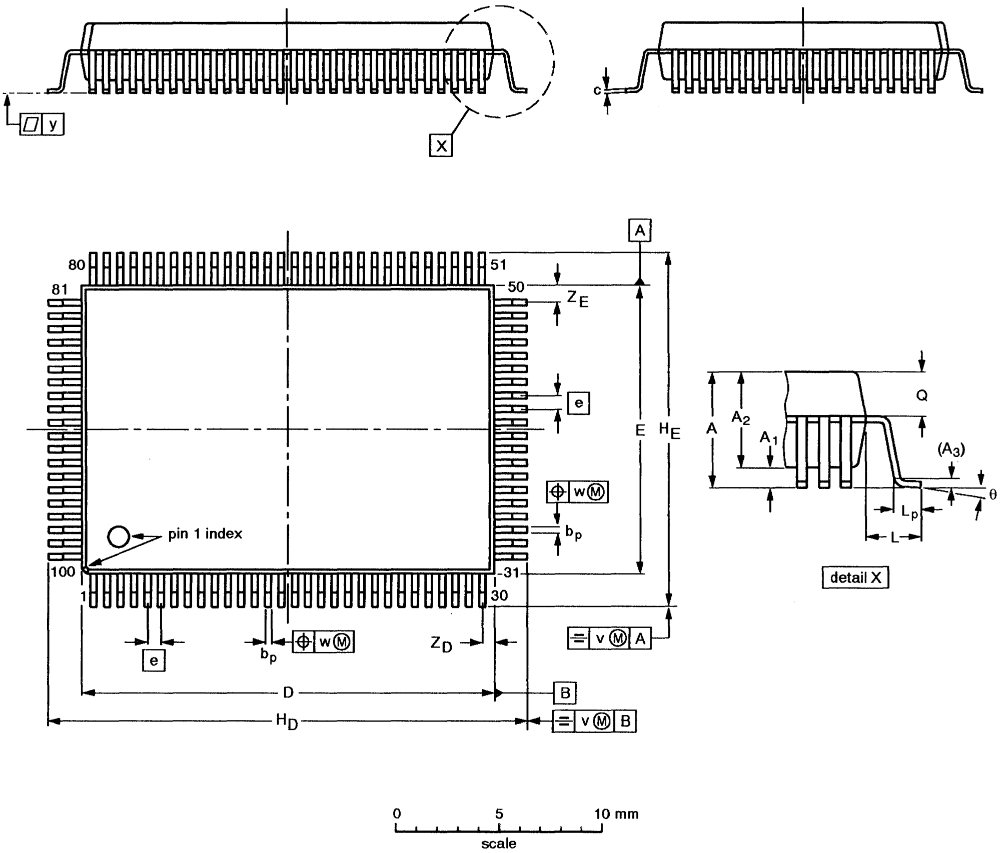
UNIT	A	A_1 min.	A_3	A_4 max.	b_p	b_1	$D^{(1)}$	$E^{(1)}$	e	e_D	e_E	H_D	H_E	k	Øj	L_p	v	w	y	$Z_D^{(1)}$ max.	$Z_E^{(1)}$ max.	β
mm	4.57 4.19	0.13	0.25	3.05	0.53 0.33	0.81 0.66	29.41 29.21	29.41 29.21	1.27	28.70 27.69	28.70 27.69	30.35 30.10	30.35 30.10	1.22 1.07	15.34 15.19	1.44 1.02	0.18	0.18	0.10	2.06	2.06	45°
inches	0.180 0.165	0.005	0.01	0.12	0.021 0.013	0.032 0.026	1.158 1.150	1.158 1.150	0.05	1.130 1.090	1.130 1.090	1.195 1.185	1.195 1.185	0.048 0.042	0.057 0.040	0.057 0.040	0.007	0.007	0.004	0.081	0.081	

Note

1. Plastic or metal protrusions of 0.01 inches maximum per side are not included.

OUTLINE VERSION	REFERENCES				EUROPEAN PROJECTION	ISSUE DATE
	IEC	JEDEC	EIAJ			
SOT189-3		MO-047AF				~~92-11-17~~ 95-02-25

QFP100: plastic quad flat package; 100 leads (lead length 1.6 mm); body 14 x 20 x 2.8 mm SOT382-1



DIMENSIONS (mm are the original dimensions)

UNIT	A max.	A_1	A_2	A_3	b_p	c	$D^{(1)}$	$E^{(1)}$	e	H_D	H_E	L	L_p	Q	v	w	y	$Z_D^{(1)}$	$Z_E^{(1)}$	θ
mm	3.40	0.60 0.25	3.05 2.55	0.25	0.38 0.22	0.23 0.13	20.1 19.1	14.1 13.9	0.65	23.45 22.95	17.45 16.95	1.60	1.03 0.73	1.4 1.2	0.20	0.12	0.10	0.68 0.45	0.68 0.45	7° 0°

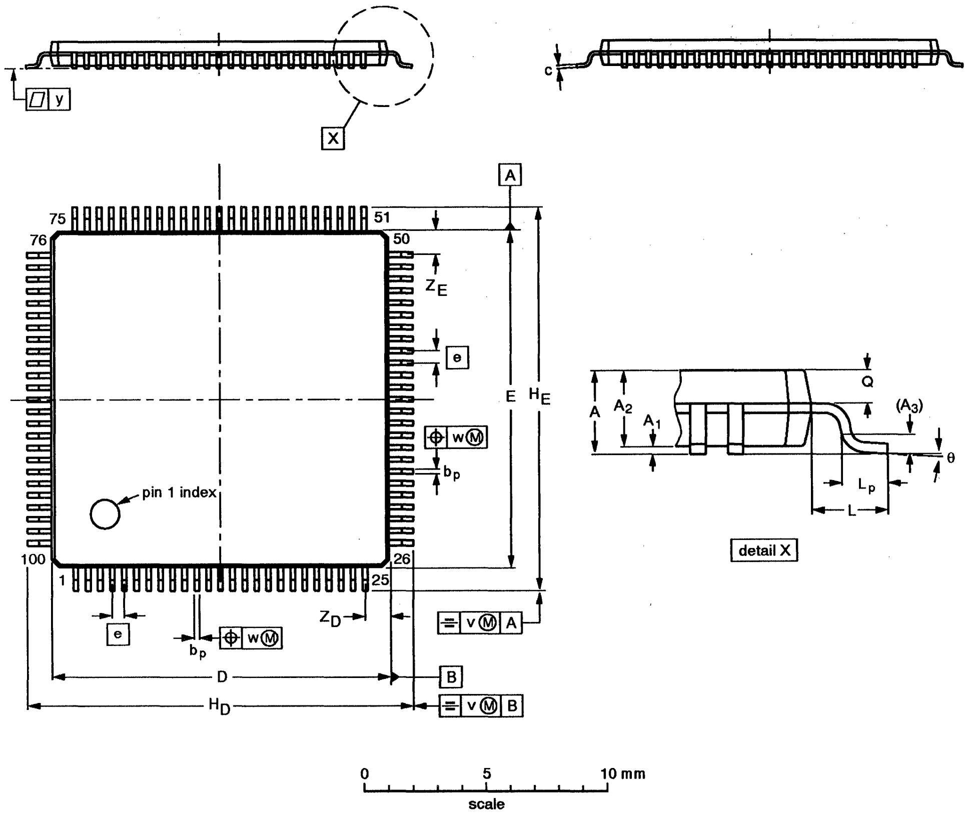
Note

1. Plastic or metal protrusions of 0.25 mm maximum per side are not included.

OUTLINE VERSION	REFERENCES				EUROPEAN PROJECTION	ISSUE DATE
	IEC	JEDEC	EIAJ			
SOT382-1		MO-108CC-1				~~94-12-12~~ 95-02-04

Package outlines

TQFP100: plastic thin quad flat package; 100 leads; body 14 x 14 x 1.0 mm SOT386-1



DIMENSIONS (mm are the original dimensions)

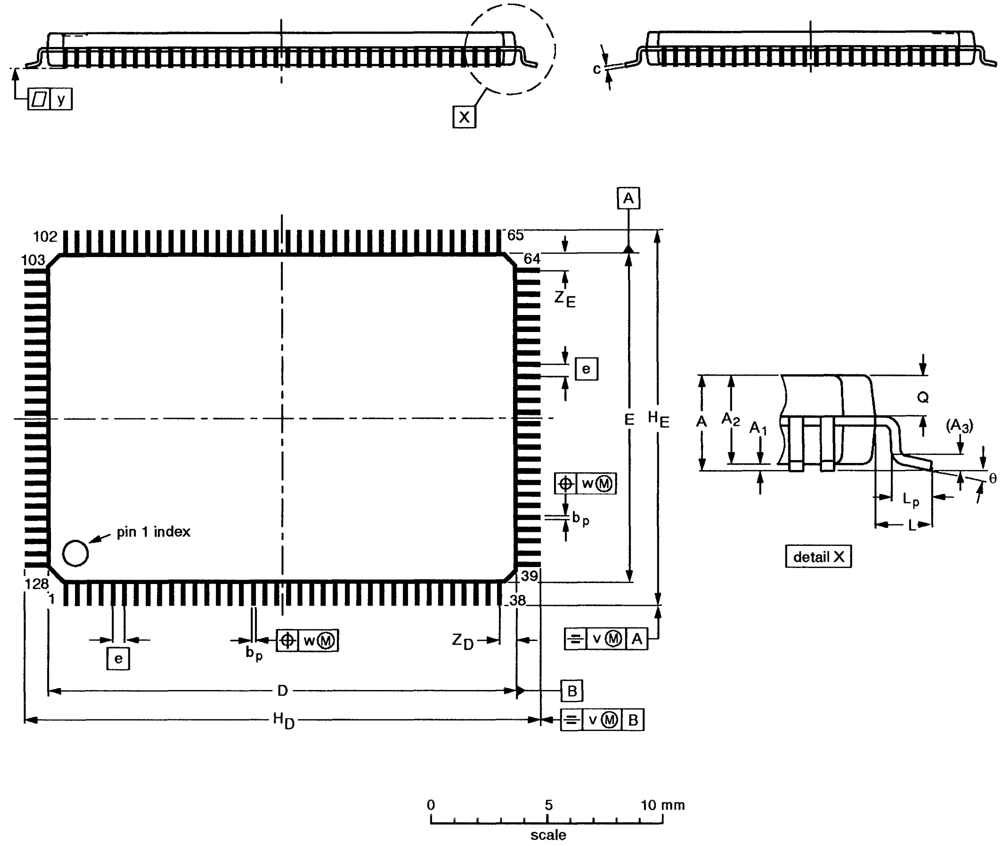
UNIT	A max.	A_1	A_2	A_3	b_p	c	$D^{(1)}$	$E^{(1)}$	e	H_D	H_E	L	L_p	Q	v	w	y	$Z_D^{(1)}$	$Z_E^{(1)}$	θ
mm	1.2	0.15 0.05	1.05 0.95	0.25	0.27 0.17	0.18 0.12	14.1 13.9	14.1 13.9	0.5	16.15 15.85	16.15 15.85	1.0	0.75 0.45	0.50 0.36	0.2	0.08	0.1	1.15 0.85	1.15 0.85	7° 0°

Note

1. Plastic or metal protrusions of 0.25 mm maximum per side are not included.

OUTLINE VERSION	REFERENCES				EUROPEAN PROJECTION	ISSUE DATE
	IEC	JEDEC	EIAJ			
SOT386-1						~~95-05-23~~ 96-04-02

LQFP128: plastic low profile quad flat package; 128 leads; body 14 x 20 x 1.4 mm SOT425-1



DIMENSIONS (mm are the original dimensions)

UNIT	A max.	A_1	A_2	A_3	b_p	c	$D^{(1)}$	$E^{(1)}$	e	H_D	H_E	L	L_p	Q	v	w	y	$Z_D^{(1)}$	$Z_E^{(1)}$	θ
mm	1.6	0.15 0.05	1.45 1.35	0.25	0.27 0.17	0.20 0.09	20.1 19.9	14.1 13.9	0.5	22.15 21.85	16.15 15.85	1.0	0.75 0.45	0.70 0.58	0.2	0.12	0.1	0.81 0.59	0.81 0.59	7° 0°

Note

1. Plastic or metal protrusions of 0.25 mm maximum per side are not included.

OUTLINE VERSION	REFERENCES				EUROPEAN PROJECTION	ISSUE DATE
	IEC	JEDEC	EIAJ			
SOT425-1						96-04-02

Section 8
Data handbook system

DATA HANDBOOK SYSTEM

Philips Semiconductors data handbooks contain all pertinent data available at the time of publication and each is revised and reissued regularly.

Loose data sheets are sent to subscribers to keep them up-to-date on additions or alterations made during the lifetime of a data handbook.

Catalogs are available for selected product ranges (some catalogs are also on floppy discs).

Our data handbook titles are listed here.

Integrated Circuits

Book	*Title*
IC01	Semiconductors for Radio and Audio Systems
IC02	Semiconductors for Television and Video Systems
IC03	Semiconductors for Wired Telecom Systems
IC04	HE4000B Logic Family CMOS
IC05	Advanced Low-power Schottky (ALS) Logic
IC06	High-speed CMOS Logic Family
IC11	General-purpose/Linear ICs
IC12	I^{2}C Peripherals
IC13	Programmable Logic Devices (PLD)
IC14	8048-based 8-bit Microcontrollers
IC15	FAST TTL Logic Series
IC16	CMOS ICs for Clocks and Watches
IC17	Semiconductors for Wireless Communications
IC18	Semiconductors for In-Car Electronics
IC19	ICs for Data Communications
IC20	80C51-based 8-bit Microcontrollers
IC22	Multimedia ICs
IC23	BiCMOS Bus Interface Logic
IC24	Low Voltage CMOS & BiCMOS Logic
IC25	16-bit 80C51XA Microcontrollers (eXtended Architecture)
IC26	IC Package Databook
IC27	Complex Programmable Logic Devices

Discrete Semiconductors

Book	*Title*
SC01	Small-signal and Medium-power Diodes
SC02	Power Diodes
SC03	Thyristors and Triacs
SC04	Small-signal Transistors
SC05	Video Transistors and Modules for Monitors
SC06	High-voltage and Switching NPN Power Transistors
SC07	Small-signal Field-effect Transistors
SC08a	RF Power Transistors for HF and VHF
SC08b	RF Power Transistors for UHF
SC09	RF Power Modules and Transistors for Mobile Phones
SC13a	Power MOS Transistors including TOPFETs and IGBTs
SC13b	Small-signal and Medium-power MOS Transistors
SC14	RF Wideband Transistors
SC15	Microwave Transistors (new version planned)
SC16	Wideband Hybrid IC Modules
SC17	Semiconductor Sensors

Professional Components

PC06	Circulators and Isolators

MORE INFORMATION FROM PHILIPS SEMICONDUCTORS?
For more information about Philips Semiconductors data handbooks, catalogs and subscriptions, contact your nearest Philips Semiconductors national organization, select from the **address list on the back cover of this handbook.** Product specialists are at your service and inquiries are answered promptly.

OVERVIEW OF PHILIPS COMPONENTS DATA HANDBOOKS

Our sister product division, Philips Components, also has a comprehensive data handbook system to support their products. Their data handbook titles are listed here.

Display Components

Book	*Title*
DC01	Colour Television Tubes
DC02	Monochrome Monitor Tubes and Deflection Units
DC03	Television Tuners, Coaxial Aerial Input Assemblies
DC04	Colour Monitor Tubes
DC05	Flyback Transformers, Mains Transformers and General-purpose FXC Assemblies

Magnetic Products

MA01	Soft Ferrites
MA03	Piezoelectric Ceramics Specialty Ferrites
MA04	Dry-reed Switches

Passive Components

PA01	Electrolytic Capacitors
PA02	Varistors, Thermistors and Sensors
PA03	Potentiometers
PA04	Variable Capacitors
PA05	Film Capacitors
PA06	Ceramic Capacitors
PA08	Fixed Resistors
PA10	Quartz Crystals
PA11	Quartz Oscillators

MORE INFORMATION FROM PHILIPS COMPONENTS?

For more information contact your nearest Philips Components national organizaiton shown in the following list.

Argentina: BUENOS AIRES, Tel. (01) 786 7635, Fax. (01) 786 9367.
Australia: NORTH RYDE, Tel. (02) 9805 4455, Fax. (02) 9805 4466.
Austria: WIEN, Tel. (01) 601 01 12 41, Fax. (01) 60 101 12 11.
Belarus: MINSK, Tel. (5172) 200 915, Fax. (5172) 200 773.
Benelux: EINDHOVEN, Tel. (+31 40) 2783 749, Fax. (+31 40) 2788 399.
Brazil: SÃO PAULO, Tel. (011) 821 2333, Fax (011) 829 1849.
Canada: SCARBOROUGH, Tel. (0416) 292 5161, Fax. (0416) 754 6248.
China: SHANGHAI, Tel. (021) 6485 0600, Fax. (021) 6485 5615.
Columbia: BOGOTA, Tel. (01) 345 8713, Fax (01) 345 8712.
Denmark: COPENHAGEN, Tel. (32) 883 333, Fax. (31) 571 949.
Finland: ESPOO, Tel. 9 (0)-615 800, Fax. 9 (0)-615 80510.
France: SURESNES, Tel. (01) 4099 6161, Fax, (01) 4099 6427.
Germany: HAMBURG, Tel. (040) 2489-0, Fax. (040) 2489 1400.
Greece: TAVROS, Tel. (01) 4894 339/(01) 4894 239, Fax. (01) 4814 240.
Hong Kong: KOWLOON, Tel. 2784 3000, Fax. 2784 3003.
India: BOMBAY, Tel. (022) 4938 541, Fax. (022) 4938 722.
Indonesia: JAKARTA, Tel. (021) 520 1122, Fax. (021) 520 5189.
Ireland: DUBLIN, Tel. (01) 76 40 203, Fax. (01) 76 40 210.
Israel: TEL AVIV, Tel (03) 6450 444, Fax. (03) 6491 007.
Italy: MILANO, Tel. (02) 6752 2531, Fax. (02) 6752 2557.
Japan: TOKYO, Tel. (03) 3740 5028, Fax. (03) 3740 0580.
Korea (Republic of): SEOUL, Tel. (02) 709 1472, Fax. (02) 709 1480.
Malaysia: PULAU PINANG, Tel. (04) 657 0055, Fax. (04) 656 5951.
Mexico: EL PASO, Tel. (915) 772 4020, Fax. (915) 772 4332.
New Zealand: AUKLAND, Tel. (09) 849 4160, Fax. (09) 849 7811.
Norway: OSLO, Tel. (22) 74 8000, Fax (22) 74 8341.
Pakistan: KARACHI, Tel. (021) 587 4641-49, Fax. (021) 577 035/(021) 587 4546.
Philippines: MANILA, Tel. (02) 816 6380, Fax. (02) 817 3474.
Poland: WARSZAWA, Tel. (022) 612 2594, Fax. (022) 612 2327.
Portugal: LINDA-A-VELHA, Tel. (01) 416 3160/416 3333,
Fax. (01) 416 3174/416 3366.
Russia: MOSCOW, Tel (095) 247 9124, Fax. (095) 247 9132.
Singapore: SINGAPORE, Tel. 350 2000, Fax. 355 1758.
South Africa: JOHANNESBURG, Tel. (011) 470 5911, Fax. (011) 470 5494.
Spain: BARCELONA, Tel. (93) 301 63 12, Fax. (93) 301 42 43.
Sweden: STOCKHOLM, Tel. (+46) 8 632 2000, Fax. (+46) 8 632 2745.
Switzerland: ZÜRICH, Tel. (01) 488 22 11, Fax. (01) 481 77 30.
Taiwan: TAIPEI, Tel. (02) 388 7666, Fax. (02) 382 4382.
Thailand: BANGKOK, Tel. (02) 745 4090, Fax. (02) 398 0793.
Turkey: ISTANBUL, Tel. (0212) 279 2770, Fax. (0212) 282 6707.
Ukraine: KIEV, Tel (044) 268 7327, Fax. (044) 268 6323.
United Kingdom: DORKING, Tel. (01306) 512 000, Fax. (01306) 512 345.
United States:

- JUPITER, FL, Tel. (561) 745 3300, Fax. (561) 745 3600.
- ANN ARBOR, MI, Tel. (313) 996 9400, Fax. (313) 761 2776.
- SAUGERTIES, NY, Tel. (914) 246 2811, Fax (914) 246 0487.

Uruguay: MONTEVIDEO, Tel. (02) 704 044, Fax (02) 920 601.

For all other countries apply to:
Philips Components.
Marketing Communications,
P.O. Box 218,
5600 MD, EINDHOVEN, The Netherlands
Fax. +31-40-2724547.

Philips Semiconductors – a worldwide company

Argentina: see South America
Australia: 34 Waterloo Road, NORTH RYDE, NSW 2113,
Tel. +61 2 9805 4455, Fax. +61 2 9805 4466
Austria: Computerstr. 6, A-1101 WIEN, P.O. Box 213,
Tel. +43 1 60 101, Fax. +43 1 60 101 1210
Belarus: Hotel Minsk Business Center, Bld. 3, r. 1211, Volodarski Str. 6,
220050 MINSK, Tel. +375 172 200 733, Fax. +375 172 200 773
Belgium: see The Netherlands
Brazil: see South America
Bulgaria: Philips Bulgaria Ltd., Energoproject, 15th floor,
51 James Bourchier Blvd., 1407 SOFIA,
Tel. +359 2 689 211, Fax. +359 2 689 102
Canada: PHILIPS SEMICONDUCTORS/COMPONENTS,
Tel. +1 800 234 7381
China/Hong Kong: 501 Hong Kong Industrial Technology Centre,
72 Tat Chee Avenue, Kowloon Tong, HONG KONG,
Tel. +852 2319 7888, Fax. +852 2319 7700
Colombia: see South America
Czech Republic: see Austria
Denmark: Prags Boulevard 80, PB 1919, DK-2300 COPENHAGEN S,
Tel. +45 32 88 2636, Fax. +45 31 57 0044
Finland: Sinikalliontie 3, FIN-02630 ESPOO,
Tel. +358 9 615800, Fax. +358 9 61580920
France: 4 Rue du Port-aux-Vins, BP317, 92156 SURESNES Cedex,
Tel. +33 1 40 99 6161, Fax. +33 1 40 99 6427
Germany: Hammerbrookstraße 69, D-20097 HAMBURG,
Tel. +49 40 23 53 60, Fax. +49 40 23 536 300
Greece: No. 15, 25th March Street, GR 17778 TAVROS/ATHENS,
Tel. +30 1 4894 339/239, Fax. +30 1 4814 240
Hungary: see Austria
India: Philips INDIA Ltd., Shivsagar Estate, A Block, Dr. Annie Besant Rd.,
Worli, MUMBAI 400 018, Tel. +91 22 4938 541, Fax. +91 22 4938 722
Indonesia: see Singapore
Ireland: Newstead, Clonskeagh, DUBLIN 14,
Tel. +353 1 7640 000, Fax. +353 1 7640 200
Israel: RAPAC Electronics, 7 Kehilat Saloniki St, PO Box 18053,
TEL AVIV 61180, Tel. +972 3 645 0444, Fax. +972 3 649 1007
Italy: PHILIPS SEMICONDUCTORS, Piazza IV Novembre 3,
20124 MILANO, Tel. +39 2 6752 2531, Fax. +39 2 6752 2557
Japan: Philips Bldg. 13-37, Kohnan 2-chome, Minato-ku, TOKYO 108,
Tel. +81 3 3740 5130, Fax. +81 3 3740 5077
Korea: Philips House, 260-199 Itaewon-dong, Yongsan-ku, SEOUL,
Tel. +82 2 709 1412, Fax. +82 2 709 1415
Malaysia: No. 76 Jalan Universiti, 46200 PETALING JAYA, SELANGOR,
Tel. +60 3 750 5214, Fax. +60 3 757 4880
Mexico: 5900 Gateway East, Suite 200, EL PASO, TEXAS 79905,
Tel. +9-5 800 234 7381
Middle East: see Italy
Netherlands: Postbus 90050, 5600 PB EINDHOVEN, Bldg. VB,
Tel. +31 40 27 82785, Fax. +31 40 27 88399
New Zealand: 2 Wagener Place, C.P.O. Box 1041, AUCKLAND,
Tel. +64 9 849 4160, Fax. +64 9 849 7811
Norway: Box 1, Manglerud 0612, OSLO,
Tel. +47 22 74 8000, Fax. +47 22 74 8341
Philippines: Philips Semiconductors Philippines Inc.,
106 Valero St. Salcedo Village, P.O. Box 2108 MCC, MAKATI,
Metro MANILA, Tel. +63 2 816 6380, Fax. +63 2 817 3474
Poland: Ul. Lukiska 10, PL 04-123 WARSZAWA,
Tel. +48 22 612 2831, Fax. +48 22 612 2327
Portugal: see Spain
Romania: see Italy
Russia: Philips Russia, Ul. Usatcheva 35A, 119048 MOSCOW,
Tel. +7 095 755 6918, Fax +7 095 755 6919
Singapore: Lorong 1, Toa Payoh, SINGAPORE 1231,
Tel. +65 350 2538, Fax. +65 251 6500
Slovakia: see Austria
Slovenia: see Italy
South Africa: S.A. PHILIPS Pty Ltd., 195-215 Main Road Martindale,
2092 JOHANNESBURG, P.O. Box 7430, Johannesburg 2000,
Tel. +27 11 470 5911, Fax. +27 11 470 5494
South America: Rua do Rocio 220, 5th Floor, Suite 51,
04552-903 São Paulo, SÃO PAULO-SP, Brazil,
Tel. +55 11 821 2333, Fax. +55 11 829 1849
Spain: Balmes 22, 08007 BARCELONA,
Tel. +34 3 301 6312, Fax. +34 3 301 4107
Sweden: Kottbygatan 7, Akalla. S-16485 STOCKHOLM,
Tel. +46 8 632 2000, Fax. +46 8 632 2745
Switzerland: Allmendstrasse 140, CH-8027 ZÜRICH,
Tel. +41 1 488 2686, Fax. +41 1 481 7730
Taiwan: Philips Semiconductors, 6F, No. 96, Chien Kuo N. Rd., Sec. 1,
TAIPEI, Taiwan, Tel. +886 2 2134 2865, Fax. +886 2 2134 2874
Thailand: PHILIPS ELECTRONICS (THAILAND) Ltd.,
209/2 Sanpavuth-Bangna Road Prakanong, BANGKOK 10260,
Tel. +66 2 745 4090, Fax. +66 2 398 0793
Turkey: Talatpasa Cad. No. 5, 80640 GÜLTEPE/ISTANBUL,
Tel. +90 212 279 2770, Fax. +90 212 282 6707
Ukraine: PHILIPS UKRAINE, 4 Patrice Lumumba str., Building B, Floor 7,
252042 KIEV, Tel. +380 44 264 2776, Fax. +380 44 268 0461
United Kingdom: Philips Semiconductors Ltd., 276 Bath Road, Hayes,
MIDDLESEX UB3 5BX, Tel. +44 181 730 5000, Fax. +44 181 754 8421
United States: 811 East Arques Avenue, SUNNYVALE, CA 94088-3409,
Tel. +1 800 234 7381
Uruguay: see South America
Vietnam: see Singapore
Yugoslavia: PHILIPS, Trg N. Pasica 5/v, 11000 BEOGRAD,
Tel. +381 11 625 344, Fax. +381 11 635 777

For all other countries apply to: Philips Semiconductors, Marketing and Sales Communications,
Building BE-p, P.O. Box 218, 5600 MD EINDHOVEN, The Netherlands, Fax . +31 40 27 24825

Internet: http://www.semiconductors.philips.com

SCH54

The information presented in this document does not form part of any quotation or contract, is believed to be accurate and reliable and may be changed without notice. No liability will be accepted by the publisher for any consequence of its use. Publication thereof does not convey nor imply any license under patent- or industrial or intellectual property rights.

Printed in the USA 217037/27.5M/FP/pp432 Date of release: June 1997 Document order number: 9397 750 01783

Let's make things better.

Philips
Semiconductors

PHILIPS

PHILIPS