INTEGRATED CIRCUITS

Complex Programmable Logic Devices

Data Handbook IC27
CD-ROM included
1998



http://www.semiconductors.philips.com

PHILIPS

Let's make things better.

QUALITY ASSURED

Our quality system focuses on the continuing high quality of our components and the best possible service for our customers. We have a three-sided quality strategy: we apply a system of total quality control and assurance; we operate customer-oriented dynamic improvement programmes; and we promote a partnering relationship with our customers and suppliers.

PRODUCT SAFETY

In striving for state-of-the-art perfection, we continuously improve components and processes with respect to environmental demands. Our components offer no hazard to the environment in normal use when operated or stored within the limits specified in the data sheet.

Some components unavoidably contain substances that, if exposed by accident or misuse, are potentially hazardous to health. Users of these components are informed of the danger by warning notices in the data sheets supporting the components. Where necessary the warning notices also indicate safety precautions to be taken and disposal instructions to be followed. Obviously users of these components, in general the set-making industry, assume responsibility towards the consumer with respect to safety matters and environmental demands.

All used or obsolete components should be disposed of according to the regulations applying at the disposal location. Depending on the location, electronic components are considered to be 'chemical', 'special' or sometimes 'industrial' waste. Disposal as domestic waste is usually not permitted.

Complex Programmable Logic Devices (CPLDs)

CONTENTS

		page
	PREFACE	3
SECTION 1	GENERAL INFORMATION	9
SECTION 2	INTRODUCTION	17
SECTION 3	XPLA FAMILY	47
SECTION 4	XPLA ENHANCED FAMILY	123
SECTION 5	XPLA2 FAMILY	217
SECTION 6	RELATED PRODUCTS	283
SECTION 7	APPLICATION NOTES	309
SECTION 8	PACKAGE INFORMATION	627
APPENDIX A	DATA HANDBOOK SYSTEM	643

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. 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 third party tool vendors have been formed to ensure state-of-the-art design solutions that are easily accessible to the designer. Whether it be Synario, Minc, Exemplar Logic, Synplicity, OrCAD, 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. The current families of Philips' CPLDs are the original XPLA, XPLA Enhanced, and XPLA2. The XPLA and XPLA enhanced families accommodate applications demanding industry-standard 32, 64, and 128 macrocell densities, with the enhanced family adding extra features like more clocks and In System Programmability. The XPLA2 family offers densities of 320 to 960 macrocells for faster, larger, more power-efficient applications.

Now you can maximize density without sacrificing speed

Philips' 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 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 7.5 ns at 3V and 6ns at 5V, and standby currents under 100 microamps, Philips' 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®
PAL®
PLDSynthesis II™
Quicksim II
SpeedWave™
Sun MicroSystems SunOS
Sun MicroSystems Solaris
Synario
Synergy™
Synopsys®
Turbo-Bits™
Verilog-XL™
ViewPLD®
ViewDraw®
ViewSynthesis®
ViewSim®
Windows®95
Windows NT®
XPLA™

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

CONTENTS

IC27: Complex Programmable Logic Devices (CPLDs)

Preface 3

Section 1 – General Information

Family Selection Guide 11
Ordering Information 12
FAX-on-DEMAND System 13
CPLD internet and support access 14
Handling MOS devices 15
Quality 16

Section 2 – Introduction

CoolRunner™ architecture overview 19
Fast Zero Power (FZP™) 30
Development software 32
Programming companies 33
In-System Programming (ISP™) 34
ISP download cable specification 46

Section 3 – XPLA Family

PZ3032 32 macrocell CPLD 49
PZ3064 64 macrocell CPLD 59
PZ3128 128 macrocell CPLD 71
PZ5032 32 macrocell CPLD 86
PZ5064 64 macrocell CPLD 96
PZ5128 128 macrocell CPLD 108

Section 4 – XPLA Enhanced Family

PZ3032C 32 macrocell CPLD with enhanced clocking 125
PZ3032A/PZ3032D 32 macrocell CPLD with enhanced clocking 137
PZ3064A/PZ3064D 64 macrocell CPLD with enhanced clocking 150
PZ3128A/PZ3128D 128 macrocell CPLD with enhanced clocking 164
PZ5032C 32 macrocell CPLD with enhanced clocking 177
PZ5064C/PZ5064N 64 macrocell CPLD with enhanced clocking 189
PZ5128C/PZ5128N 128 macrocell CPLD with enhanced clocking 203

Section 5 – XPLA 2 Family

PZ3320C/PZ3320N 320 macrocell SRAM CPLD 219
PZ3960C/PZ3960N 960 macrocell SRAM CPLD 249

Section 6 – Related Products

P3Z22V10 3V zero power, TotalCMOS™, universal PLD device 285
P5Z22V10 5V zero power, TotalCMOS™, universal PLD device 296
PZLCP Low Cost Programmer for Philips CoolRunner devices and related adapters 307
PZVHDLEZ Full VHDL synthesis and automatic optimization software for Philips Semiconductors CoolRunners 308

CONTENTS

Section 7 – Application notes

AN055 Metastability Characteristics for Philips CPLDs ... 311
AN057 Altera (AHDL) to Philips (PHDL) design conversion guidelines ... 313
AN058 Cadence/Synopsys design flows for targeting Philips CPLDs ... 319
AN059 Mentor Graphics Design Flow for targeting Philips Semiconductors CPLDs ... 331
AN060 Using Data I/O-Model Technology VHDL tools to target Philips Semiconductors CPLDs ... 351
AN062 Understanding the hidden costs of using high–power CPLDs ... 369
AN063 Probing internal nodes using XPLA software graphic simulator ... 370
AN064 Using sum of products control terms in Philips CoolRunner™ CPLDs ... 375
AN065 Understanding CoolRunner™ clocking options ... 378
AN066 XPLA Designer™ hierarchical PHDL design support ... 382
AN068 Terminating unused CoolRunner™ I/O pins ... 399
AN069 ISP design considerations for CoolRunner™ CPLDs ... 400
AN070 Verilog implementation of a Manchester Encoder/Decoder in Philips CPLDs ... 405
AN071 OrCAD Express Design Flow for Philips CPLDs ... 440
AN072 Implementing a UART in Philips CPLDs ... 481
AN073 Synplicity/Model Tech design flow for targeting Philips CPLDs ... 489
AN074 OrCAD Capture Schematic/PHDL Design Flow for Philips CPLDs ... 515
AN075 Clock modulation: How to synthesize additional clocks for counters and state machines ... 532
AN076 Using the Philips PZ3960 Evaluation Board ... 536
AN078 VHDL Easy Design Flow for Philips CPLDs ... 568
AN079 Viewlogic Intelliflow Design Flow for Philips CPLDs ... 598

Section 8 – Package information

Soldering ... 628
Package outlines
SO24: plastic small outline package; 24 leads; body width 7.5 mm ... SOT137-1 ... 630
TSSOP24: plastic thin shrink small outline package; 24 leads; body width 4.4 mm ... SOT355-1 ... 631
PLCC28: plastic leaded chip carrer; 28 leads; pedestal ... SOT261-3 ... 632
PLCC44: plastic leaded chip carrier; 44 leads ... SOT187-2 ... 633
TQFP44: plastic thin quad flat package; 44 leads; body 10 x 10 x 1.0 mm ... SOT376-1 ... 634
PLCC68: plastic leaded chip carrier; 68 leads; pedestal ... SOT188-3 ... 635
PLCC84: plastic leaded chip carrier; 84 leads; pedestal ... SOT189-3 ... 636
QFP100: plastic quad flat package; 100 leads (lead length 1.6 mm); body 14 x 20 x 2.8 mm ... SOT382-1 ... 637
QFP160: plastic quad flat package; 160 leads (lead length 1.6 mm); body 28 x 28 x 3.4 mm; high stand-off height ... SOT322-2 ... 638
TQFP100: plastic thin quad flat package; 100 leads; body 14 x 14 x 1.0 mm ... SOT386-1 ... 639
LQFP128: plastic low profile quad flat package; 128 leads; body 14 x 20 x 1.4 mm ... SOT425-1 ... 640
BGA492: plastic ball grid array package; 492 balls; body 35 x 35 x 1.75 mm ... SOT514-1 ... 641

Data Handbook System ... **644**

CONTENTS

ADDITIONAL MATERIAL ON ENCLOSED CD-ROM ONLY:

Application Notes/Design Models

Verilog models of commonly used digital function for targeting Philips CPLDs
VHDL models of commonly used digital functions for targeting Philips CPLDs
Philips hardware description language models of commonly used digital functions
Exemplar/Model tech design flow for targeting Philips CPLDs

Section 1
General Information

CONTENTS

Family Selection Guide 11
Ordering Information 12
FAX-on-DEMAND System 13
CPLD internet and support access 14
Handling MOS devices 15
Quality 16

Family Selection Guide

PHILIPS CoolRunner™ FAMILY SELECTION GUIDE

Base Part Number	Comm'l Speeds (t_{PD} in ns)	Package Designator	Package Description	I/O	Dedicated Inputs	Enhanced Clocking	JTAG	ISP	PCI	5 V Tolerance	Progr. Slew Rate Control	Process
P3Z22V10	10, 15	A	PLCC28	10	12					X		0.5 µ EE
P3Z22V10	10, 15	D	SOL24	10	12					X		0.5 µ EE
P3Z22V10	10, 15	DH	TSSOP24	10	12					X		0.5 µ EE
P5Z22V10	7.5, 10	A	PLCC28	10	12							0.5 µ EE
P5Z22V10	7.5, 10	D	SOL24	10	12							0.5 µ EE
P5Z22V10	7.5, 10	DH	TSSOP24	10	12							0.5 µ EE
PZ3032	8, 10, 12	A44	PLCC44	32	4				X			0.5 µ EE
PZ3032	8, 10, 12	BC	TQFP44	32	4				X			0.5 µ EE
PZ5032	6, 7.5, 10	A44	PLCC44	32	4				X			0.5 µ EE
PZ5032	6, 7.5, 10	BC	TQFP44	32	4				X			0.5 µ EE
PZ3032C	8, 10, 12	A44	PLCC44	32	4	X	compatible	X	X			0.5 µ EE
PZ3032C	8, 10, 12	BC	TQFP44	32	4	X	compatible	X	X			0.5 µ EE
PZ5032C	6, 7.5, 10	A44	PLCC44	32	4	X	compatible	X	X			0.5 µ EE
PZ5032C	6, 7.5, 10	BC	TQFP44	32	4	X	compatible	X	X			0.5 µ EE
PZ3032A	6, 7.5, 10	A44	PLCC44	32	4	X	compatible	X		X		0.35 µ EE
PZ3032A	6, 7.5, 10	BC	TQFP44	32	4	X	compatible	X		X		0.35 µ EE
PZ3064	10, 12	A44	PLCC44	32	4				X			0.5 µ EE
PZ3064	10, 12	BC	TQFP44	32	4				X			0.5 µ EE
PZ3064	10, 12	A68	PLCC68	48	4				X			0.5 µ EE
PZ3064	10, 12	A84	PLCC84	64	4				X			0.5 µ EE
PZ3064	10, 12	BB1	PQFP100	64	4				X			0.5 µ EE
PZ5064	7.5, 10	A44	PLCC44	32	4				X			0.5 µ EE
PZ5064	7.5, 10	BC	TQFP44	32	4				X			0.5 µ EE
PZ5064	7.5, 10	A68	PLCC68	48	4				X			0.5 µ EE
PZ5064	7.5, 10	A84	PLCC84	64	4				X			0.5 µ EE
PZ5064	7.5, 10	BB1	PQFP100	64	4				X			0.5 µ EE
PZ3064A	7.5, 10	A44	PLCC44	32	4	X	compatible	X		X		0.35 µ EE
PZ3064A	7.5, 10	BC	TQFP44	32	4	X	compatible	X		X		0.35 µ EE
PZ3064A	7.5, 10	BP	TQFP100	64	4	X	compatible	X		X		0.35 µ EE
PZ5064C	7.5, 10	A44	PLCC44	32	4	X	compatible	X	X			0.5 µ EE
PZ5064C	7.5, 10	BC	TQFP44	32	4	X	compatible	X	X			0.5 µ EE
PZ5064C	7.5, 10	BP	TQFP100	64	4	X	compatible	X	X			0.5 µ EE
PZ3128	10, 12, 15	A84	PLCC84	64	4		fully compliant	X	X			0.5 µ EE
PZ3128	10, 12, 15	BB1	PQFP100	80	4		fully compliant	X	X			0.5 µ EE
PZ3128	10, 12, 15	BP	TQFP100	80	4		fully compliant	X	X			0.5 µ EE
PZ3128	10, 12, 15	BE	LQFP128	96	4		fully compliant	X	X			0.5 µ EE
PZ3128	10, 12, 15	BB2	PQFP160	96	4		fully compliant	X	X			0.5 µ EE
PZ5128	7.5, 10, 12	A84	PLCC84	64	4		fully compliant	X	X			0.5 µ EE
PZ5128	7.5, 10, 12	BB1	PQFP100	80	4		fully compliant	X	X			0.5 µ EE
PZ5128	7.5, 10, 12	BP	TQFP100	80	4		fully compliant	X	X			0.5 µ EE
PZ5128	7.5, 10, 12	BE	LQFP128	96	4		fully compliant	X	X			0.5 µ EE
PZ5128	7.5, 10, 12	BB2	PQFP160	96	4		fully compliant	X	X			0.5 µ EE
PZ3128A	7.5, 10, 12	BP	TQFP100	80	4	X	compatible	X		X		0.35 µ EE
PZ3128A	7.5, 10, 12	BE	LQFP128	96	4	X	compatible	X		X		0.35 µ EE
PZ5128C	7.5, 10, 12	BP	TQFP100	80	4	X	compatible	X	X			0.5 µ EE
PZ5128C	7.5, 10, 12	BE	LQFP128	96	4	X	compatible	X	X			0.5 µ EE
PZ3320C	tbd	tbd	LQFP160	112		X	fully compliant	X		X	X	0.35 µ SRAM
PZ3320C	tbd	tbd	PBGA256	192		X	fully compliant	X		X	X	0.35 µ SRAM
PZ3960C	7.5	EB	PBGA492	384		X	fully compliant	X	X		X	0.35 µ SRAM

NOTE:
All products available in industrial temperature range—check our website at www.coolpld.com for availability details.

Ordering Information

COOLRUNNER™ CPLD PRODUCTS

Example: PZ x yyy k S zz YYY

PZ = Philips Zero power

x = Supply Voltage
- 3 = 3.3 V
- 5 = 5.0 V

yyy = Macrocell Count

k = Operating Temp/Architecture/Process
- – = Commercial, original architecture
- I = Industrial, original architecture
- C = Commercial, enhanced clocking
- N = Industrial, enhanced clocking
- A = Commercial, enhanced clocking, 0.35 micron process w/5V tol I/Os
- D = Industrial, enhanced clocking 0.35 micron process w/5 V tol I/Os

S = ISP Device (if S is present)

zz = Speed Grade (t_{PD})
- 6 = 6 ns
- 7 = 7.5 ns
- 8 = 8 ns
- 10 = 10 ns
- 12 = 12 ns
- 15 = 15 ns

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)
- EB = 492-pin Plastic Ball Grid Array (PBGA)

COOLRUNNER™ 22V10 PRODUCTS

Example: P x Z 22V10k z YY

PZ = Philips Zero power

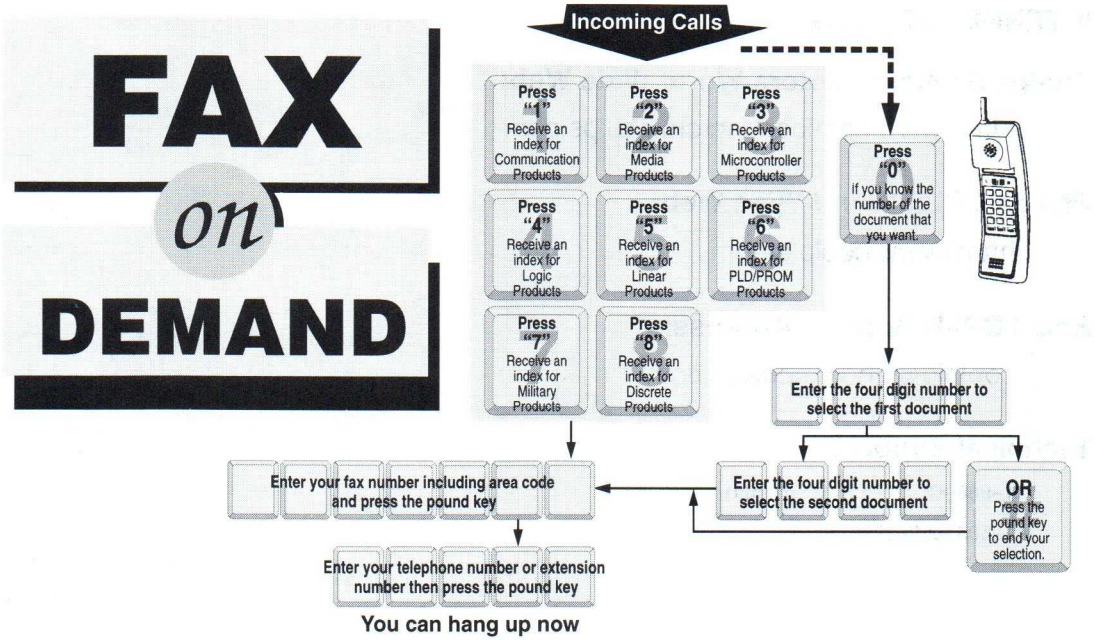
x = Supply Voltage
- 3 = 3.3 V
- 5 = 5 V

k = Operating Temperature
- – = Commercial
- I = Industrial

z = Speed Grade (t_{PD})
- 7 = 7.5 ns
- D = 10 ns
- B = 15 ns

YY = Package Code
- A = 28-pin Plastic Leaded Chip Carrier (PLCC)
- D = 24-pin Small Outline Large (SOL)
- DH = 24-pin Plastic Thin Shrink Small Outline Package (TSSOP)

FAX-on-DEMAND System



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:

United Kingdom, Ireland, Benelux & Scandinavia	+44-181-730-5020
North America	1-800-282-2000
Asia/Pacific (Australia, China/HK, India, Indonesia, Japan, Korea, Malaysia, New Zealand, Philippines, Singapore, Taiwan, and Thailand)	+852 2811 9990

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

Technical Support:

1–888–COOLPLD (USA only)

1–505–858–2996

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

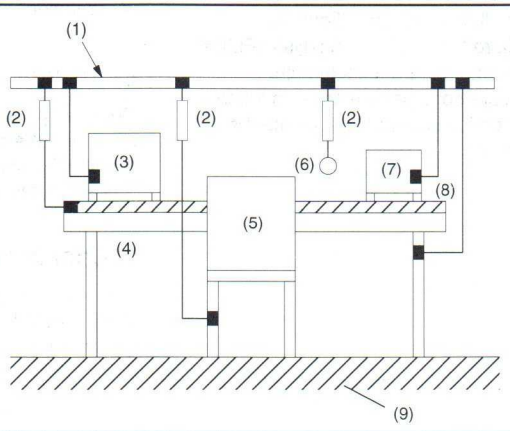
MOS devices must be removed from their protective packing with earthed component pincers or short-circuit clips. Short-circuit clips must remain in place during mounting, soldering and cleansing/drying processes. Do not remove more devices from the storage packing than are needed at any one time.
Production/assembly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

During assembly, ensure 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 as unmounted MOS devices. They should also carry warning 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 Effects Analysis (FMEA) analysis the critical parameters are detected and measures taken to ensure good performance on these parameters. This method is also used to evaluate the capability of process steps.

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 and continuous improvement.

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

CoolRunner™ architecture overview 19

Fast Zero Power (FZP™) 30

Development software 32

Programming companies 33

In-System Programming (ISP™) 34

ISP download cable specification 46

CoolRunner™ architecture overview

INTRODUCTION

There are three basic families of Philips Semiconductors' CoolRunner™ CPLDs: XPLA, XPLA Enhanced, and XPLA2. The architecture is similar for all of these families, but there are important differences for each. XPLA is Philips' first CPLD family with densities between 32 and 128 macrocells. XPLA Enhanced is a modified version of the XPLA family, with extra features like more clocks and In System Programmability. XPLA2 is a family of higher density (320 to 960 macrocells) CPLDs.

XPLA™ AND XPLA ENHANCED ARCHITECTURE

Figure 1 gives a high level block diagram of the XPLA™ and XPLA Enhanced architecture. These architectures 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.

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. In the XPLA Enhanced family, these can also be used as clocks. 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 and XPLA Enhanced architectures are 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 these architectures accommodate 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.

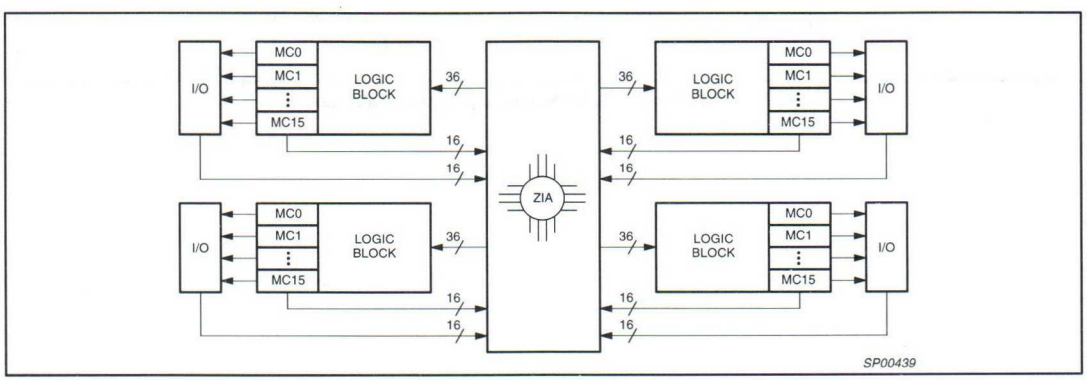


Figure 1. XPLA™ Block Diagram of 64 Macrocell Device

CoolRunner™ architecture overview

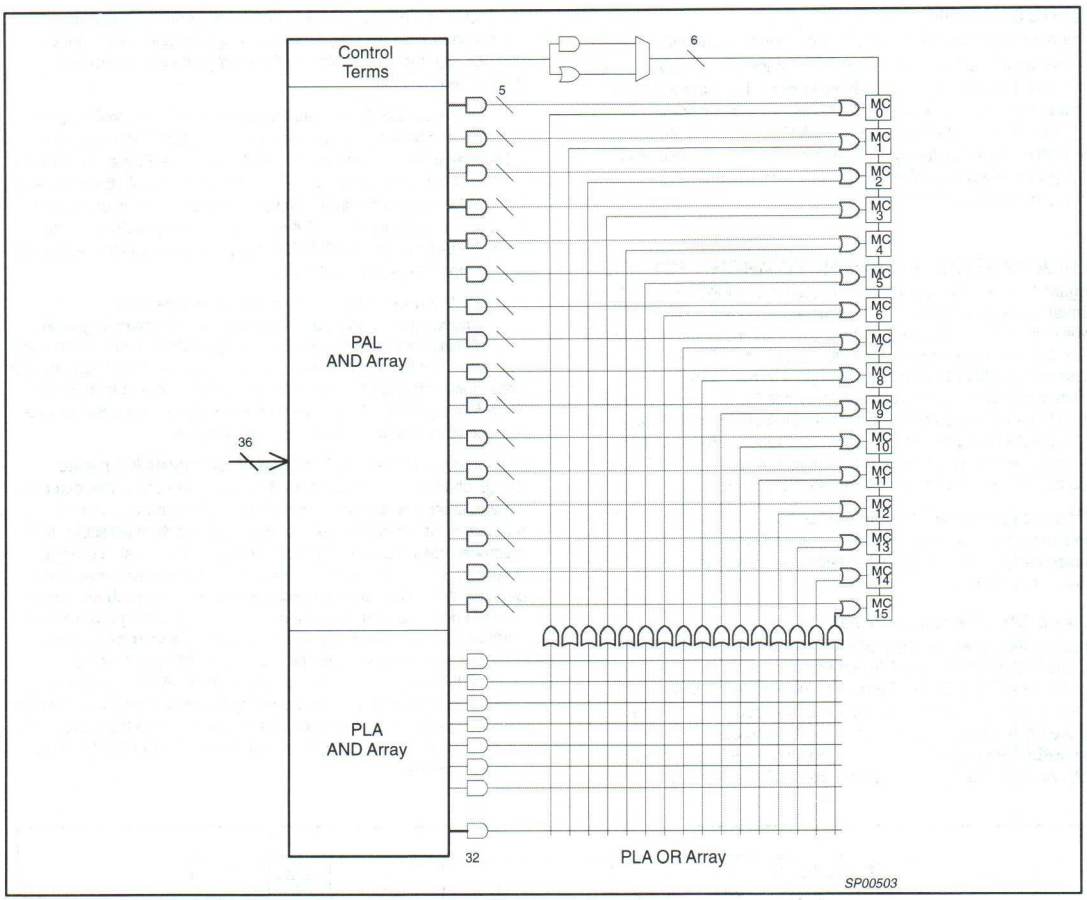


Figure 2. XPLA and XPLA Enhanced logic block architecture

XPLA Macrocell Configuration

Figure 3 shows the architecture of the macrocell used in the XPLA 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 XPLA 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".

For the macrocells in the logic block that are associated with I/O pins, 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 feedback path. When these macrocells are used as outputs, the output buffer is enabled, and either feedback path can be used to feedback the logic implemented in the macrocell. When the I/O pins are used as inputs, the output buffer of these macrocells will be 3-stated and the input signal will be fed into the ZIA via the I/O feedback path. In this case the logic functions implemented in the buried macrocell can be fed back into the ZIA via the macrocell feedback path. For macrocells that are not associated with I/O pins, there is one feedback path to the ZIA. Logic functions implemented in these buried macrocells are fed back into the ZIA via this path.

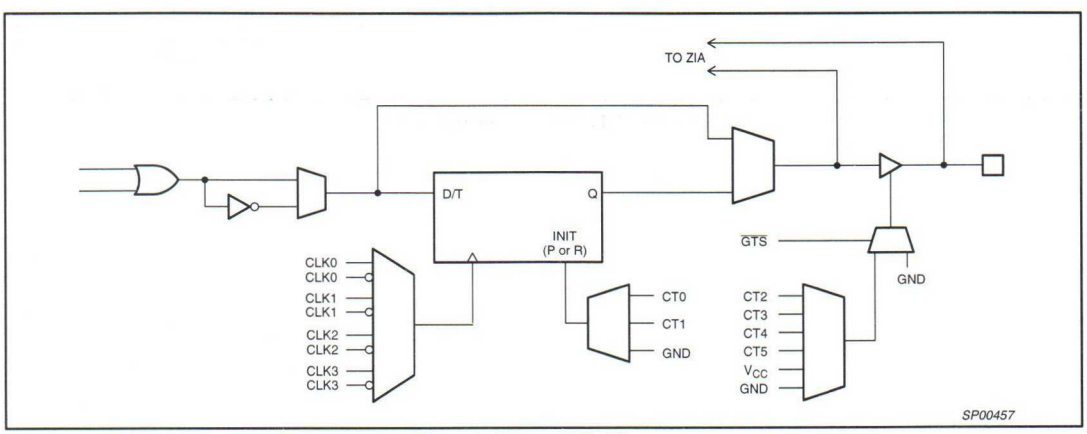


Figure 3. Macrocell Architecture

XPLA Enhanced Macrocell Configuration

Figure 4 shows the architecture of the macrocell used in the XPLA Enhanced family. It is exactly like the macrocell of the XPLA family except that two of the six control terms are shared by the output enable mux and the clock mux. Sharing these control terms provides two extra clocks per logic block in these devices. This means that the Enhanced 32, 64, and 128 macrocell devices can have up to 6, 12, and 20 clocks, respectively. For more information, see the application note Understanding CoolRunner Clocking Options.

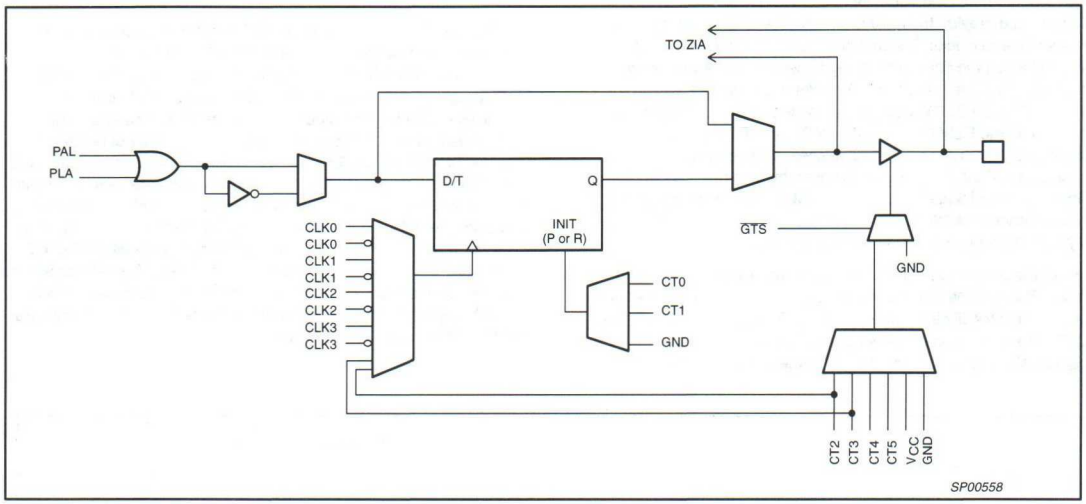


Figure 4. XPLA Macrocell Architecture

XPLA2 ARCHITECTURE

Figure 5 shows a high level block diagram of the XPLA2 architecture. The XPLA2 architecture is a multi-level, modular hierarchy that consists of Fast Modules interconnected by a Global Zero Power Interconnect Array (GZIA). The GZIA is a virtual crosspoint switch that connects the Fast Modules together. Each Fast Module accepts 64 bits from the GZIA and outputs 64 bits to the GZIA. Each Fast Module is essentially an 80 macrocell CPLD with four logic blocks of 20 macrocells each inside. There are eight dedicated, low-skew, global clocks for the device; and each Fast Module has access to any two of these clocks (there are additional asynchronous clocks available in the Fast Modules, see Figure 6). There are also Global Tri-state (gts) and Global Reset (rstn) pins that are common to all Fast Modules. When gts is pulled high, all output buffers in the device will be disabled, causing all I/O pins to be tri-stated. When rstn is pulled low, all flip-flops of the device will be reset.

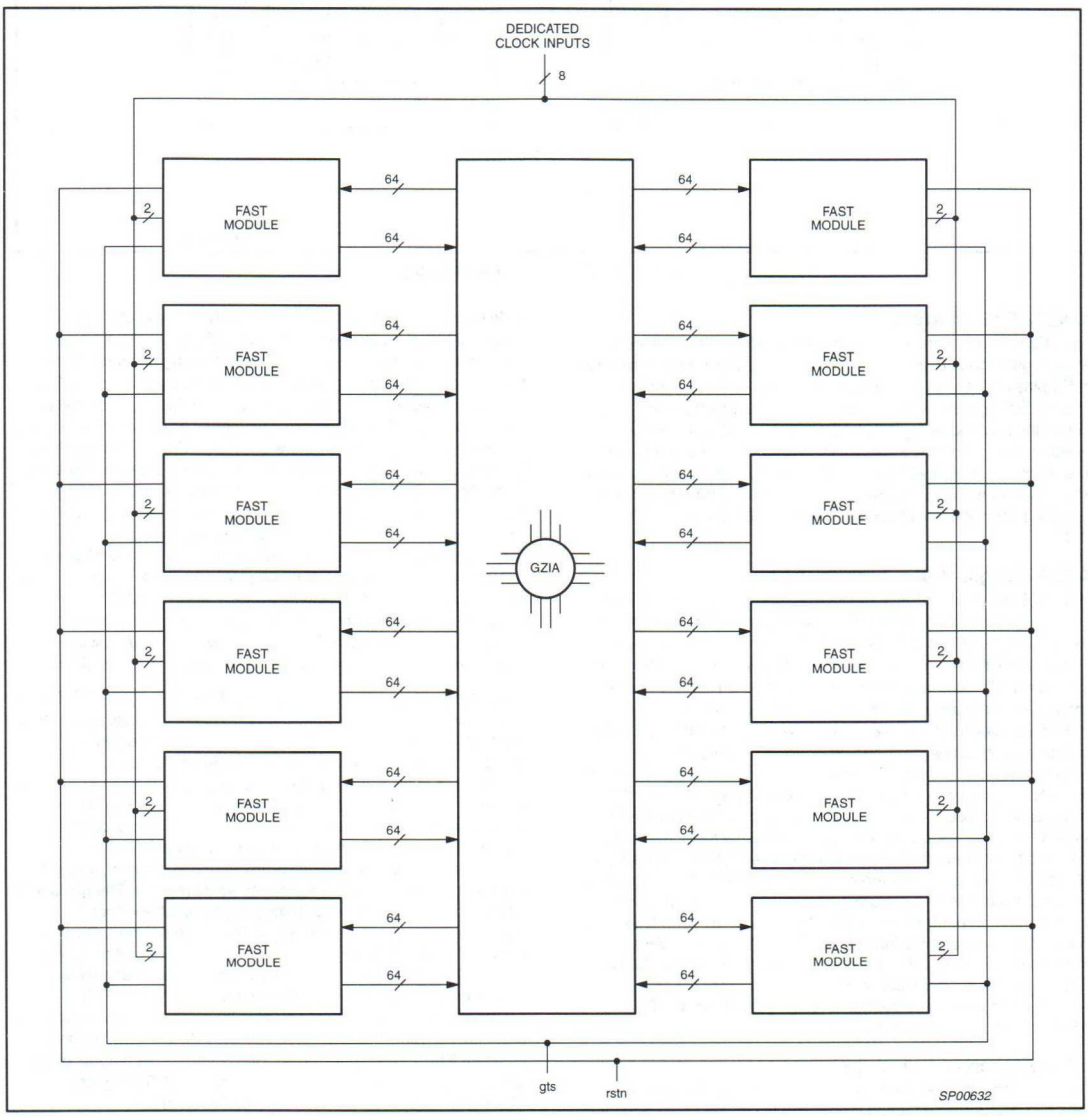


Figure 5. Philips XPLA2 CPLD Architecture

CoolRunner™ architecture overview

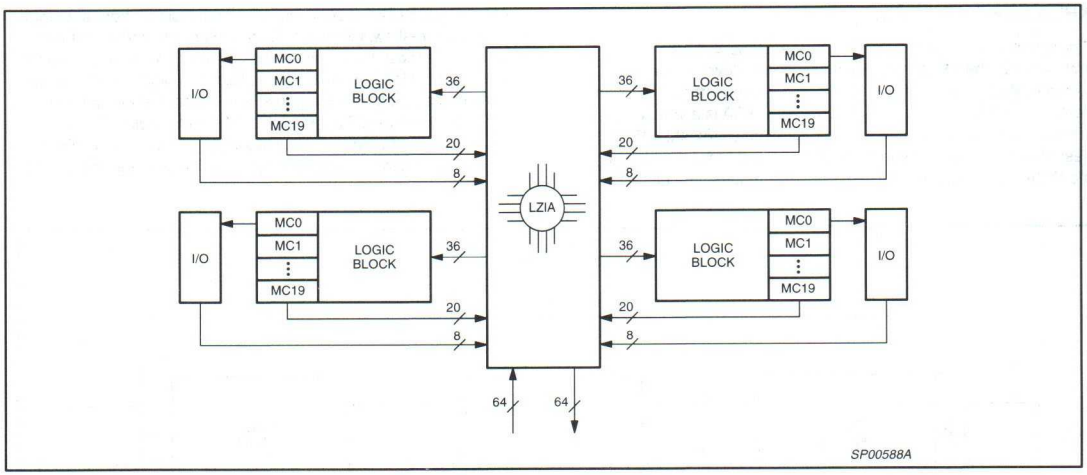


Figure 6. Philips XPLA2 Fast Module

XPLA2 Fast Module

Each Fast Module consists of four Logic Blocks of 20 macrocells each. Eight of the 20 macrocells in each Logic Block are connected to I/O pins and the remaining 12 can be used as buried nodes. These four Logic Blocks are connected together by the Local Zero Power Interconnect Array (LZIA). The LZIA is a virtual crosspoint switch that connects the logic blocks to each other and to the GZIA. The feedback from all 80 macrocells, input from the I/O pins, and the 64 bit input bus from the GZIA are input into the LZIA. The LZIA outputs 36 signals into each Logic Block and 64 signals into the GZIA.

XPLA2 Logic Block Architecture

Figure 7 illustrates the XPLA2 logic block architecture. Each logic block contains 8 control terms, a PAL array, and 20 macrocells. The 36 inputs from the LZIA are available to all control terms and each product term in both the PAL and the PLA array. The 8 control terms can individually be configured as either SUM or PRODUCT terms, and are used to control the asynchronous preset and reset functions of the macrocell registers, the output enables of the 20 macrocells, and for asynchronous clocking. 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. Each macrocell has 4 dedicated product terms from the PAL array. When additional logic is required, each macrocell takes the extra product term(s) from the PLA array. The PLA array consists of 32 extra product terms that are shared between the 20 macrocells of the logic block. The PAL product terms can be connected to the PLA product terms through either an OR gate or an XOR gate. One input to the XOR gate can be connected to all the PLA terms, which provides for extremely efficient logic synthesis. An eight bit XOR function can be implemented in only 20 product terms. Each macrocell can use the output from the OR gate or the XOR gate in either normal or inverted state.

XPLA2 Macrocell Architecture

Figure 8 shows the XPLA2 macrocell architecture. The macrocell can be configured as either a D- or T-type flip-flop or a combinatorial logic function. A D-type flip-flop is generally more useful for implementing state machines and data buffering while a T-type flip-flop is generally more useful in implementing counters. Each of these flip-flops can be clocked from any one of four sources. Two of the clock sources (CLK0 and CLK1) are from the eight dedicated, low-skew, global clock networks designed to preserve the integrity of the clock signal by reducing skew between rising and falling edges. These clocks are designated as "synchronous" clocks and must be driven by an external source. Both CLK0 and CLK1 can clock the macrocell flip-flops on either the rising edge or the falling edge of the clock signal. The other clock sources are designated as "asynchronous" and are connected to two of the eight control terms (CT6 and CT7) provided in each Logic Block. These clocks can be individually configured as any PRODUCT term or SUM term equation created from the 36 signals available inside the Logic Block. Thus, in each Logic Block, there are up to four possible clocks; and in each Fast Module, there are up to 10 possible clocks.

The remaining six control terms of each Logic Block (CT0–CT5) are used to control the asynchronous preset/reset of the flip-flops and the enable/disable of the output buffers in each macrocell. Control terms CT0 and CT1 are used to control the asynchronous preset/reset of the macrocell's flip-flop. Note that the power-on reset leaves all macrocells in the "zero" state when power is properly applied, and that the preset/reset feature for each macrocell can also be disabled. Each macrocell can choose between an asynchronous reset or an asynchronous preset function, but both cannot be simultaneously used on the same register. The global rstn function can always be used, regardless of whether or not asynchronous reset or preset control terms are enabled. Control terms CT2, CT3, CT4 and CT5 are used to enable or disable the macrocell's output buffer. Having four dedicated output enable control terms ensures that the CoolRunner devices are PCI compliant. The output buffers can also be always enabled or always disabled. All CoolRunner devices also provide a Global Tri-State (gts) pin, which, when pulled high, will 3-state all the outputs of the device. This pin is provided to support "In-Circuit Testing" or "Bed-of-Nails" testing used during manufacturing.

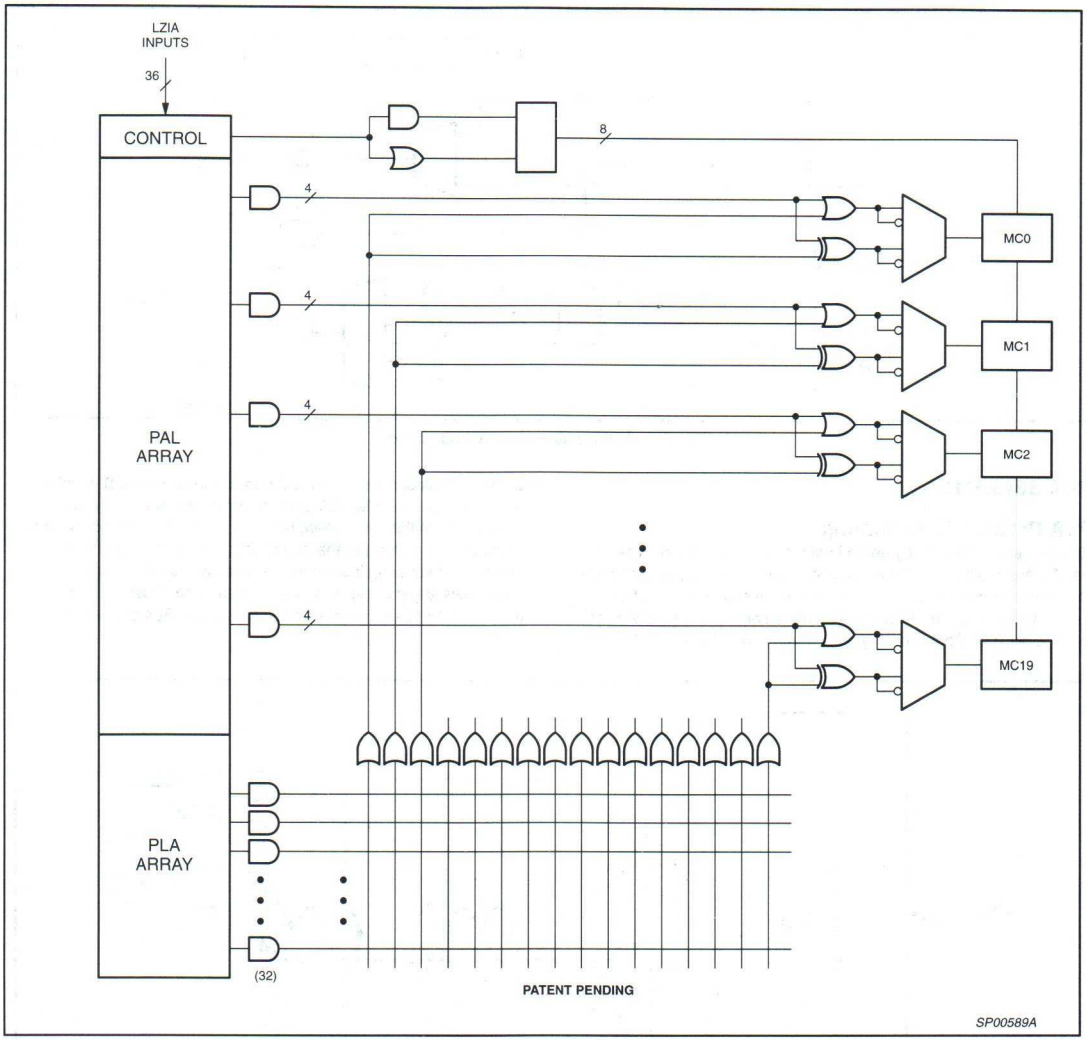


Figure 7. Philips XPLA2 Logic Block Architecture

For the macrocells in the Logic Block that are associated with I/O pins, there are two feedback paths to the LZIA: on from the macrocell, and one from the I/O pin. The LZIA feedback path before the output buffer is the macrocell feedback path, while the LZIA feedback path after the output buffer is the I/O pin feedback path. When these macrocells are used as outputs, the output buffer is enabled, and either feedback path can be used to feedback the logic implemented in the macrocell. When the I/O pins are used as inputs, the output buffer of these macrocells will be 3-stated and the input signal will be fed into the LZIA via the I/O feedback path. In this case the logic functions implemented in the buried macrocell can be fed back into the LZIA via the macrocell feedback path. For macrocells that are not associated with I/O pins, there is one feedback path to the LZIA. Logic functions implemented in these buried macrocells are fed back into the LZIA via this path.

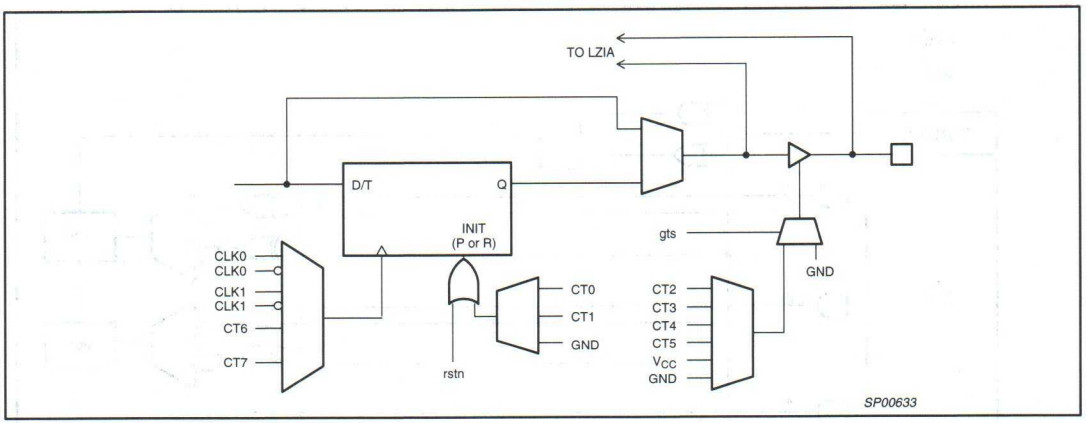


Figure 8. XPLA2 Macrocell Architecture

PLA BENEFITS

PLA Product Term Sharing

Another feature offered by an architecture with a PLA, 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 CoolRunner architectures allow sharing of PLA product terms between macrocells as shown in Figure 9. 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.

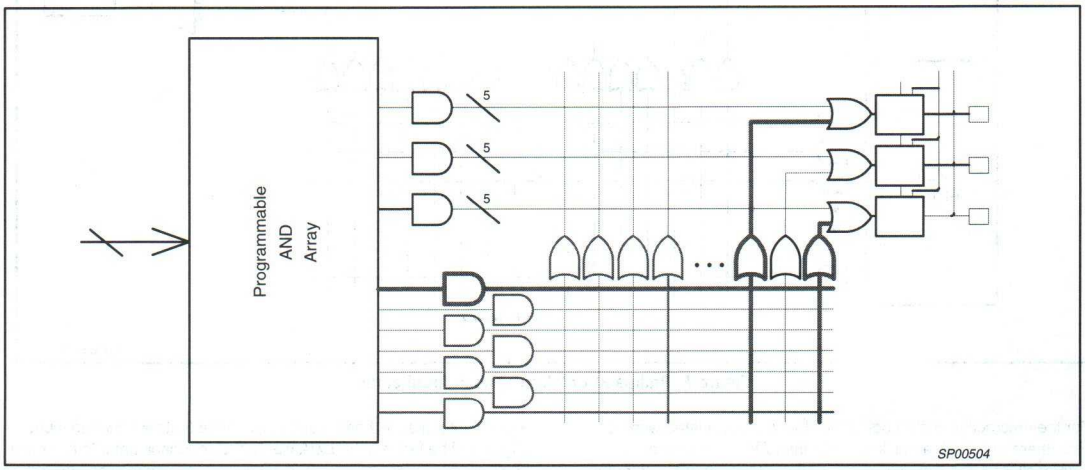


Figure 9.

Simple Timing Model

Figure 10 shows the XPLA and XPLA Enhanced 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}. Figure 11 shows the XPLA2 timing model. Note that it only has one more variable: t_{GZIA}. 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 CoolRunner architectures, 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!

In the PZ3960 for example, a combinatorial logic function of four or fewer product terms constructed from inputs within the same logic block would have a t_{PD} delay of 7.5 ns. If the logic function were more than four product terms wide, the delay would be t_{PD} plus the fixed PLA delay, or 7.5 + 1.5 = 9.0 ns. A function that used the PAL array and inputs from a different Fast Module would have a propagation delay of t_{PD} plus the fixed GZIA delay, or 7.5 + 4.0 = 11.5 ns. Again, designers can easily determine if the device will meet his system requirements, before they even begin the design!

HOW TO USE COOLRUNNER ARCHITECTURE FEATURES

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 (logic high on XPLA2), 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) or with control terms inside the logic blocks. 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, a general purpose I/O, or a control term. 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.

CLK1, CLK2, and 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.

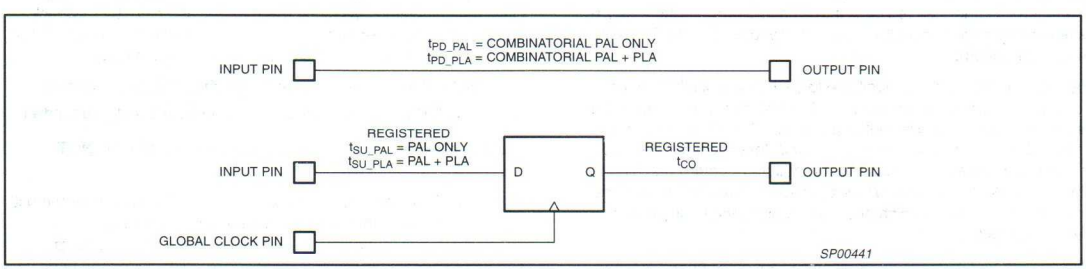


Figure 10. CoolRunner™ Timing Model

CoolRunner™ architecture overview

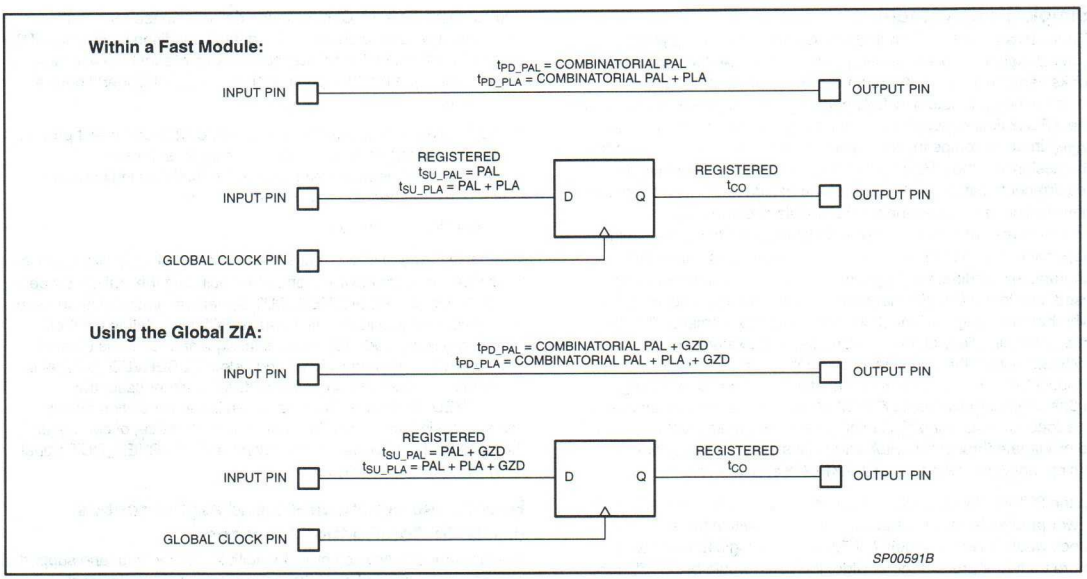


Figure 11.

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.

In the XPLA enhanced and XPLA2 devices, there are two control term clocks in every Logic Block. These clocks can be individually configured as any PRODUCT term or SUM term equation created from the 36 signals available inside the Logic Block. If a SUM-OF-PRODUCTS equation is required, it must be implemented in a macrocell and then fed back into a control term through the ZIA (see the application note Using Sum of Products Control Terms for more information).

Each control term clock is available to all the macrocells within a Logic Block, but it must be duplicated on another control term if the same clock is used in different Logic Blocks. These clocks are not attached to a low-skew clock network and, they must pass through the interconnect array and a single product term before reaching the flip-flop. Unlike the dedicated clock networks, these clocks are not associated with specific pins and may be assigned to any I/O or dedicated input.

Please note that all of the clock signals in a CoolRunner™ device have selectable clock polarity 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 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.

See application note Using Sum of Products Control forms for more information.

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.

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}. 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.

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.

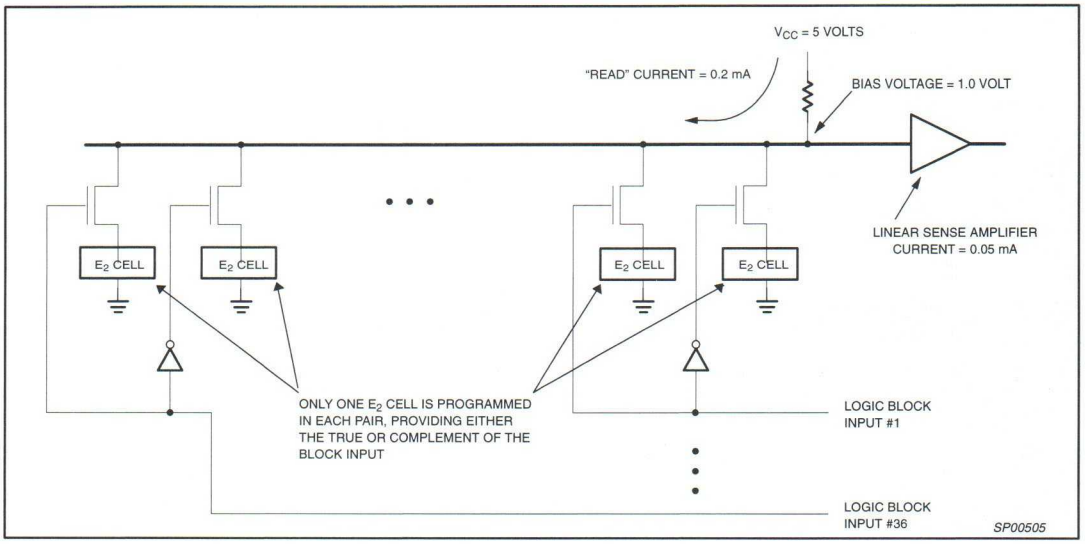


Figure 1. Product Term in Existing CPLDs

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.

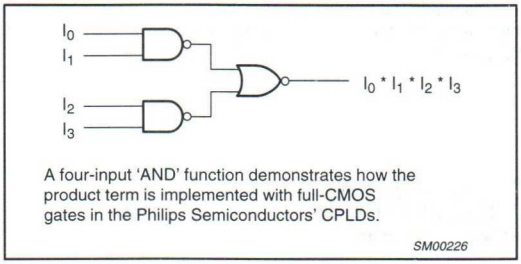


Figure 2. Representation of a Product Term using the Philips FZP™ Design Technique

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

DEVELOPMENT SOFTWARE

Philips' Programmable Logic Business Line (PLBL) software strategy is based on three elements:

- To provide support for all Philips CPLDs in tools and design flows already in place at the customer site.
- To offer cost effective software solutions that enable users easy access to Philips silicon technologies.
- To maintain in-house software expertise that optimizes interaction between device architecture and CAE software.

This strategy means that the PLBL supports both third party and proprietary design tools. 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. By supporting a high-value, low-cost proprietary solution, Philips can provide solutions to companies that do not own third party design tools but need cost-effective access to Philips silicon solutions.

The above strategy facilitates the need for partnerships, and Philips works 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.

Below are the logos of the companies that currently support design development for Philips CPLDs (XPLA is Philips own proprietary software). There are multiple flows that are possible through many of these design environments, and new flows are constantly being added. To determine the design flow that best meets your needs, or for specific questions regarding any of these flows, contact the CoolPLD Technical Support hotline by one of the methods shown in the General Information section of this book, or your local Philips representative.

MINC INCORPORATED

SYNARIO® Design Automation

CADENCE™

EXEMPLAR LOGIC

Synplicity

Mentor Graphics The Power to Create

XPLA

SYNOPSYS®

OrCAD

VIEWlogic

SP00649

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	(44) 1707 33214
Data I/O	10525 Willows Road N.E. Redmond, WA 98073	800-332-8246

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.

In-System Programming (ISP™)

- 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.

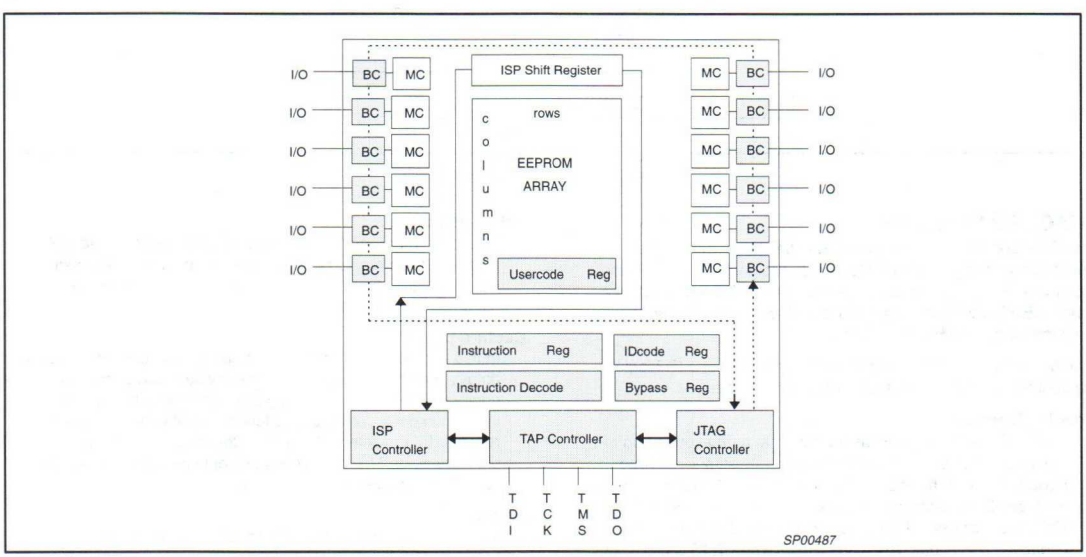


Figure 1. ISP and BST CPLD High Level Architecture

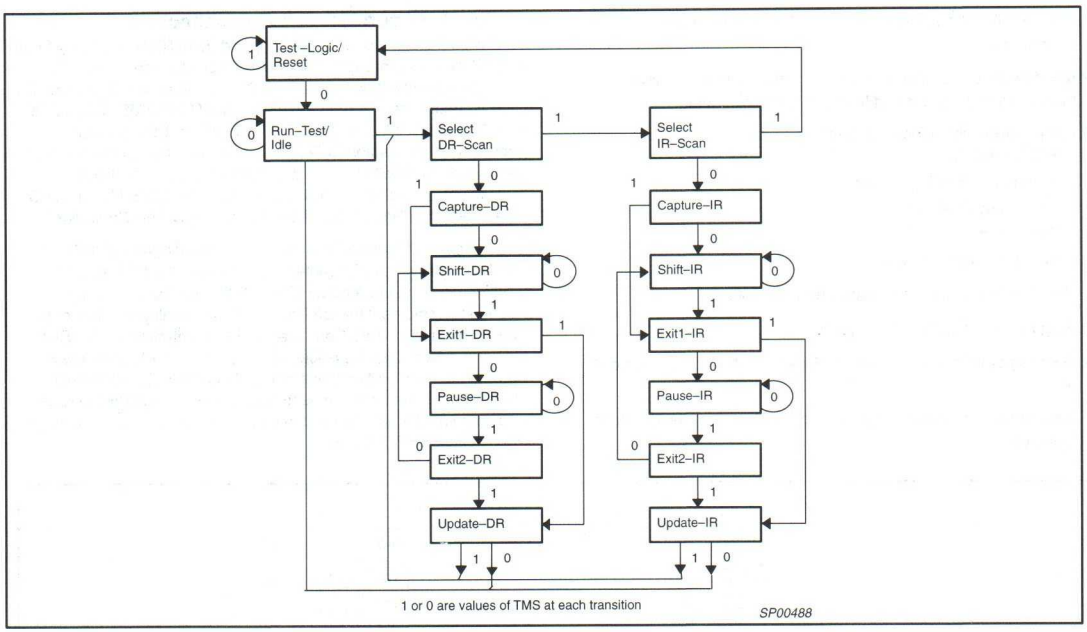


Figure 2. JTAG TAP Controller

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.

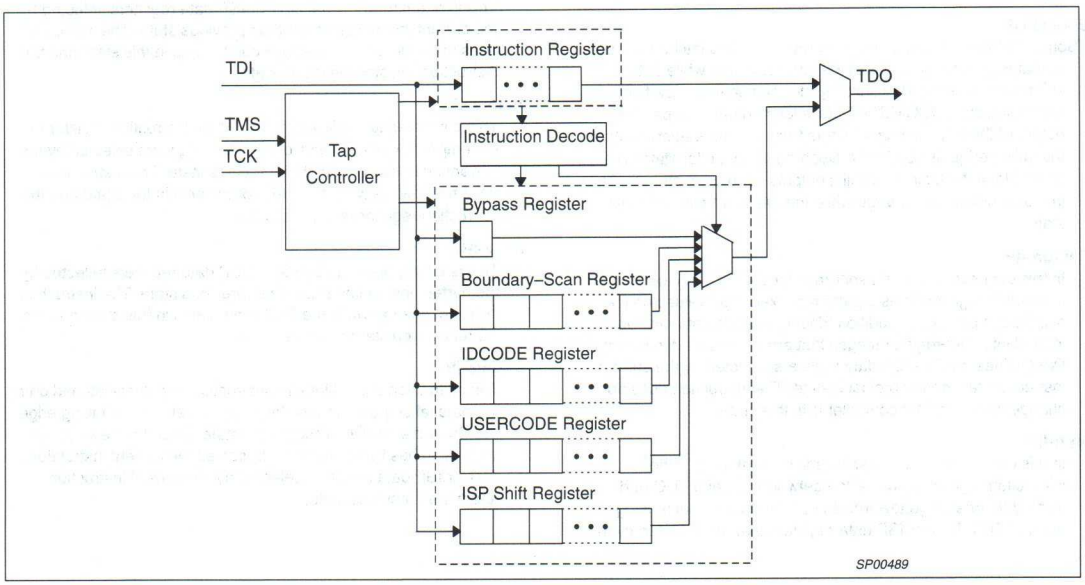


Figure 3. JTAG Registers

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.

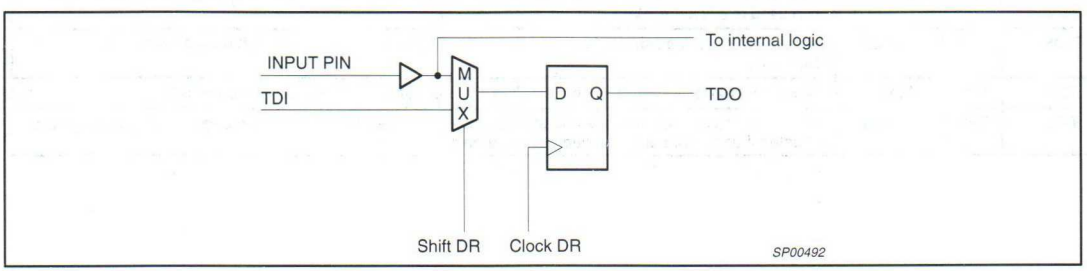


Figure 4. An Observe-Only Dedicated Input BSR Cell

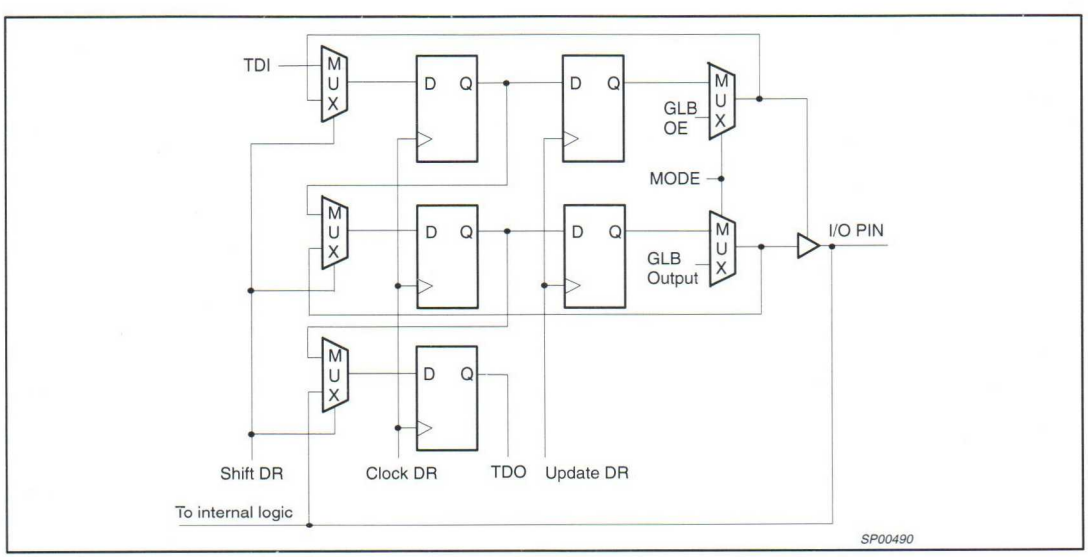


Figure 5. Compliant BSR Cell with an Observe–Only Input for a Bi–directional Pin

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. 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 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.

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.

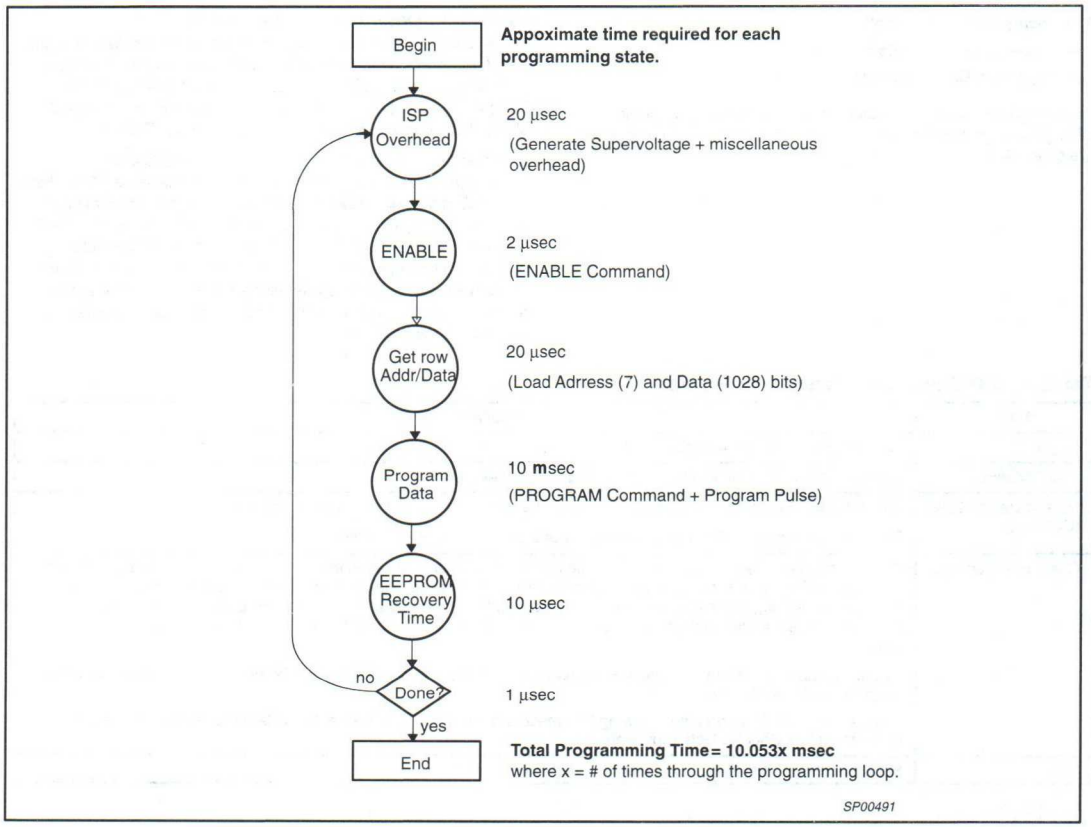


Figure 6. PZ3128/PZ5128 ISP Programming Flow Diagram

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 $(1.22x)^1$ required to simultaneously program twenty devices instead of a single device. Programming twenty devices in series would require twenty times $(20x)^1$ 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

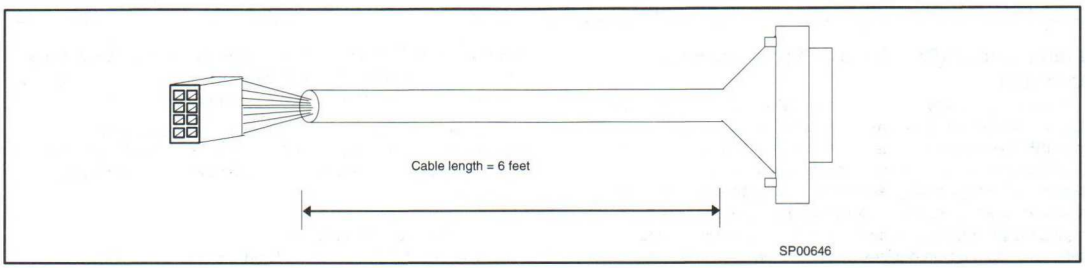
Boundary Scan Description Language (BSDL) descriptions of Philips' CPLDs are also available for use in test program development.

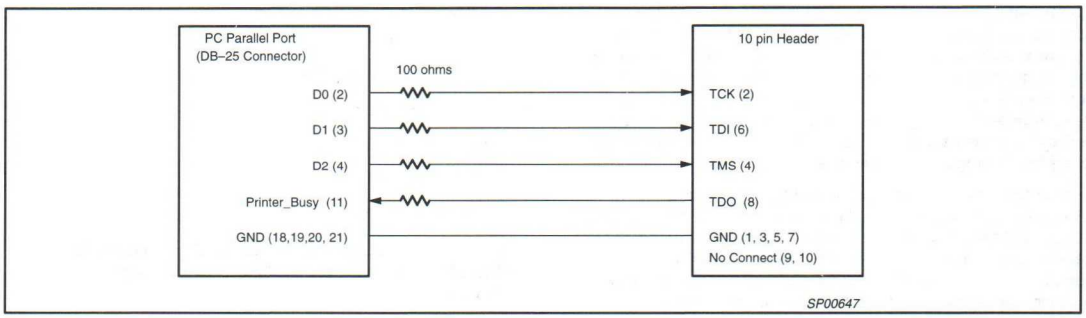
1. Where x is the time required to program 1 device.

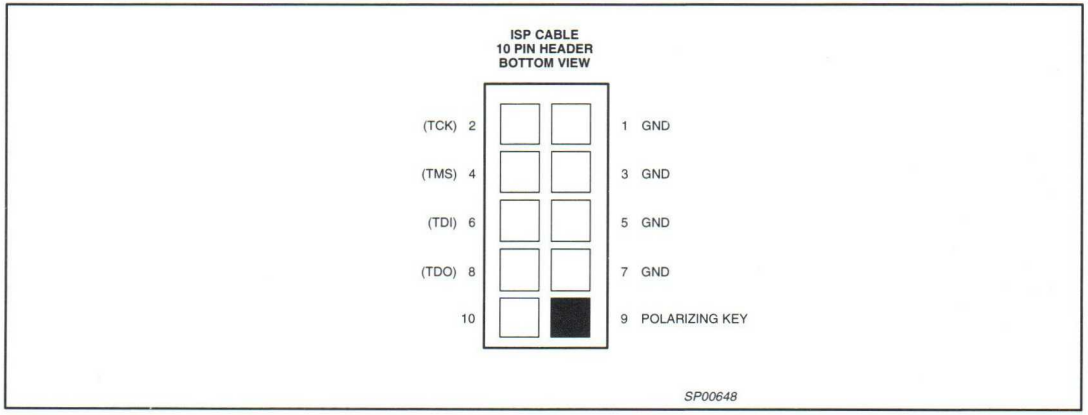
ISP download cable specification

PHILIPS XPLA™ PC–ISP DOWNLOAD CABLE

The Philips PC–ISP download cable connects to a standard PC parallel port and a 10-pin JTAG board header. The header is keyed on the PC–ISP ProtoBoard to prevent improperly connecting the cable. For reference purpose, the cable is implemented as shown in the diagram below. The use of ribbon cable with alternating signal and ground conductors provides a low noise signal path to the board. The use of 100 Ohm resistors in series with the signal lines serves as protection in cases where the voltage potential between the PC and the board differs.







Section 3
XPLA Family

CONTENTS

PZ3032	32 macrocell CPLD	49
PZ3064	64 macrocell CPLD	59
PZ3128	128 macrocell CPLD	71
PZ5032	32 macrocell CPLD	86
PZ5064	64 macrocell CPLD	96
PZ5128	128 macrocell CPLD	108

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 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, Minc, Exemplar Logic, Synplicity, 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).

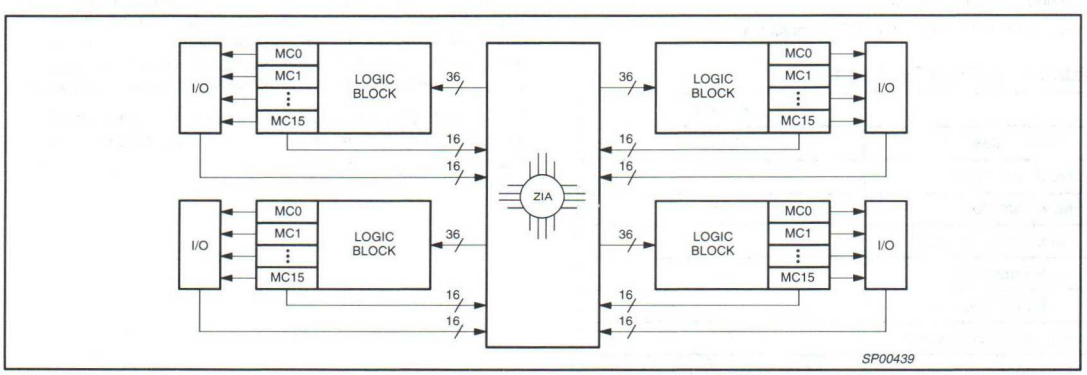


Figure 1. Philips XPLA CPLD Architecture

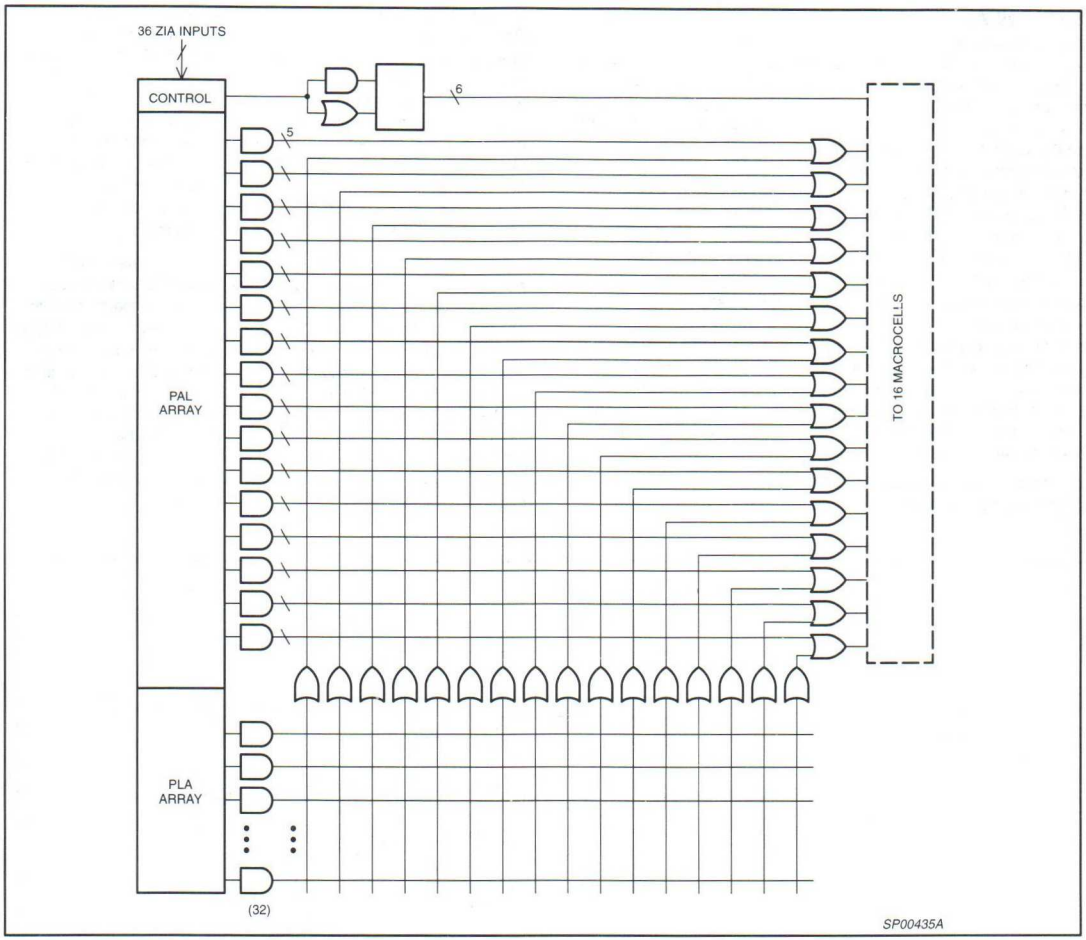


Figure 2. Philips XPLA Logic Block Architecture

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). The timing for asynchronous clocks is different in that the t_{CO} time is extended by the amount of time that it takes for the signal to propagate through the array and reach the clock network, and the t_{SU} time is reduced. Please see the application note titled "Understanding CoolRunner Clocking Options" for more detail.

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 enabled and 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.

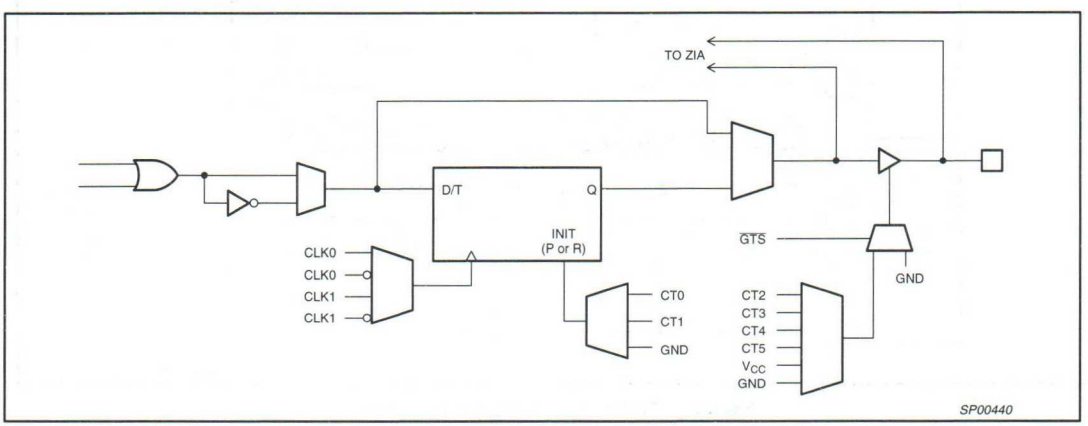


Figure 3. PZ3032 Macrocell Architecture

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.

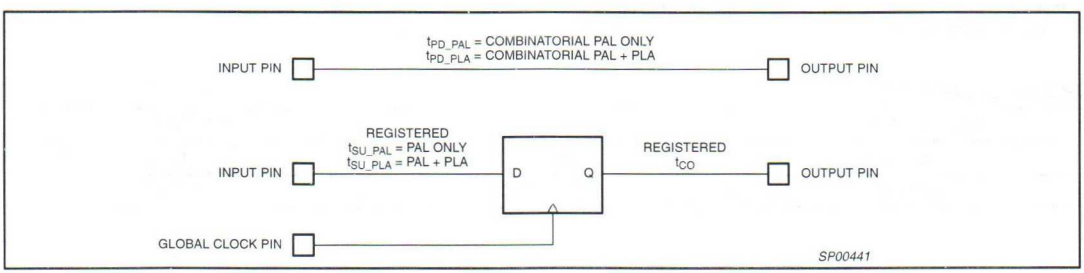


Figure 4. CoolRunner™ Timing Model

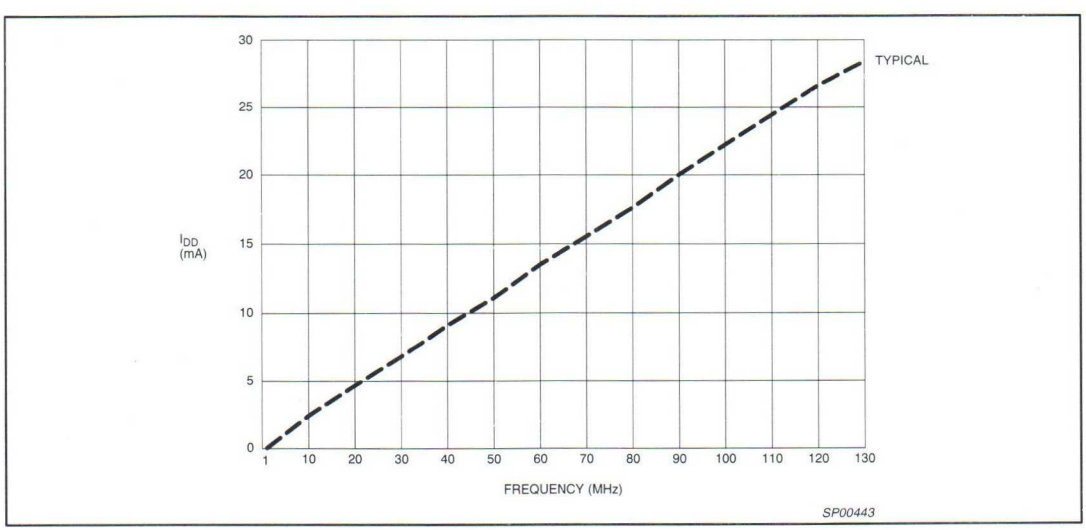


Figure 5. I_{DD} vs. Frequency @ V_{DD} = 3.3V

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[2]	–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.
2. The chip supply voltage must be monotonic.

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: $0°C \le T_{amb} \le +70°C$; $3.0V \le V_{DD} \le 3.6V$

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_{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}[1]	Standby current	V_{DD} = 3.6V, T_{amb} = 0°C		35	µA
I_{DDD}[1, 2]	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[3]	1 pin at a time for no longer than 1 second	–5	–100	mA
C_{IN}	Input pin capacitance[3]	T_{amb} = 25°C, f = 1MHz		8	pF
C_{CLK}	Clock input capacitance[3]	T_{amb} = 25°C, f = 1MHz	5	12	pF
$C_{I/O}$	I/O pin capacitance[3]	T_{amb} = 25°C, f = 1MHz		10	pF

NOTES:

1. See Table 2 on page 53 for typical values.
2. 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.
3. Typical values, not tested.

AC ELECTRICAL CHARACTERISTICS[1] FOR COMMERCIAL GRADE DEVICES

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

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 (global synchronous clock from pin)	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: $-40°C \le T_{amb} \le +85°C$; $3.0V \le V_{DD} \le 3.6V$

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_{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}[1]	Standby current	V_{DD} = 3.6V, T_{amb} = –40°C		45	µA
I_{DDD}[1, 2]	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[3]	1 pin at a time for no longer than 1 second	–5	–120	mA
C_{IN}	Input pin capacitance[3]	T_{amb} = 25°C, f = 1MHz		8	pF
C_{CLK}	Clock input capacitance[3]	T_{amb} = 25°C, f = 1MHz	5	12	pF
$C_{I/O}$	I/O pin capacitance[3]	T_{amb} = 25°C, f = 1MHz		10	pF

NOTES:

1. See Table 2 on page 53 for typical values.
2. 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.
3. Typical values, not tested.

AC ELECTRICAL CHARACTERISTICS[1] FOR INDUSTRIAL GRADE DEVICES

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

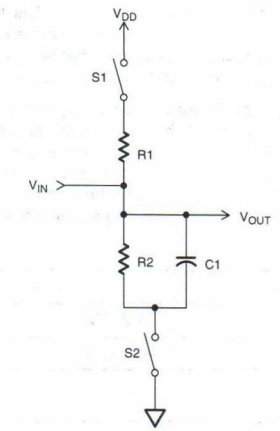
SYMBOL	PARAMETER	10		12		UNIT
		MIN.	MAX.	MIN.	MAX.	
t_{PD_PAL}	Propagation delay time, input (or feedback node) to output through PAL	2	10	2	12	ns
t_{PD_PLA}	Propagation delay time, input (or feedback node) to output through PAL & PLA	3	12.5	3	15	ns
t_{CO}	Clock to out (global synchronous clock from pin)	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

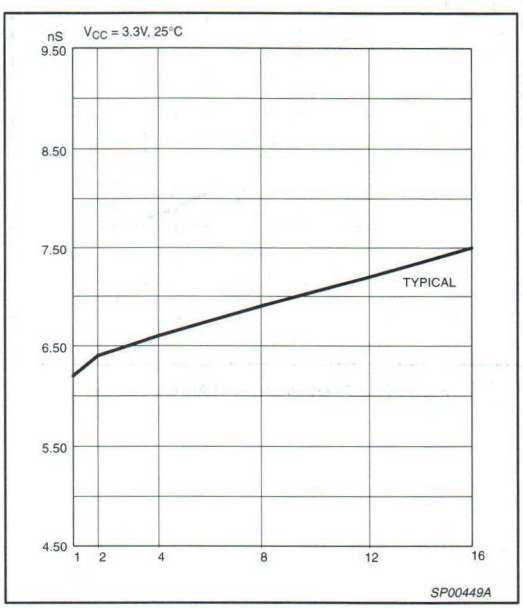
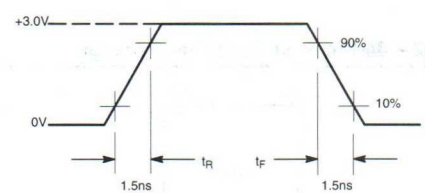


Figure 6. t_{PD_PAL} vs. Outputs switching

VOLTAGE WAVEFORM



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

Input Pulses

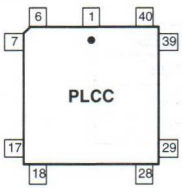
Table 3. t_{PD_PAL} vs. # of Outputs switching

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

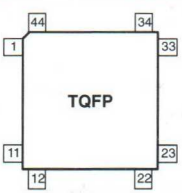
PZ3032 – 44-Pin Plastic Leaded Chip Carrier



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

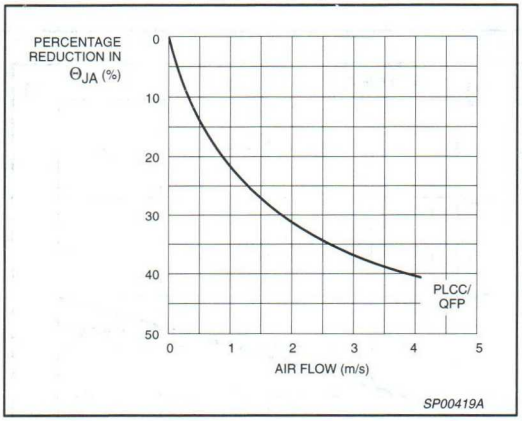


Figure 7. Average Effect of Airflow on Θ_{JA}

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 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, Minc, Exemplar Logic, Synplicity, 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).

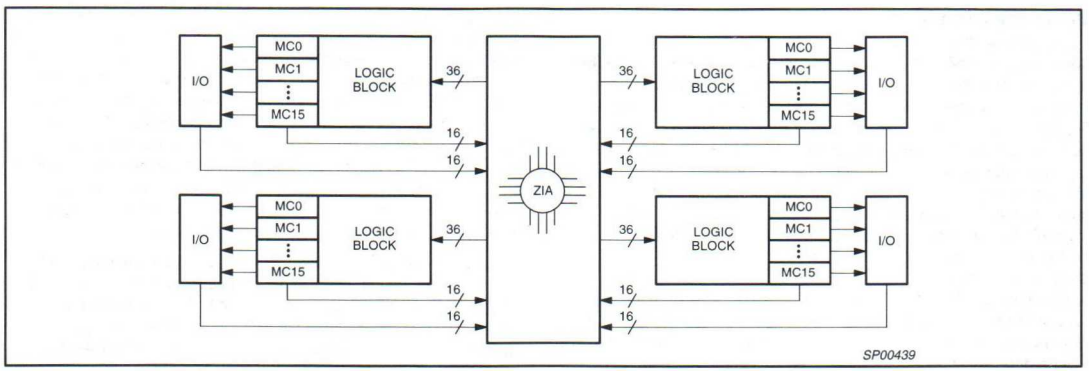


Figure 1. Philips XPLA CPLD Architecture

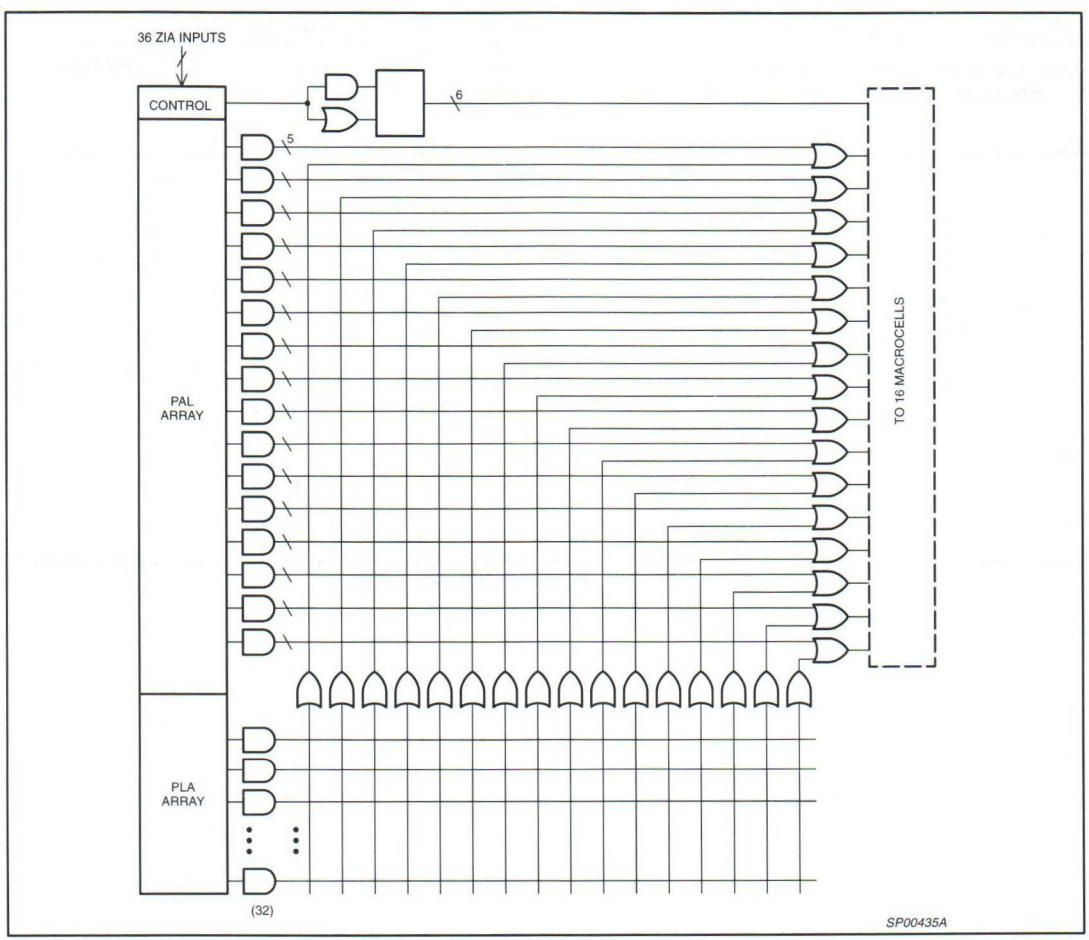


Figure 2. Philips XPLA Logic Block Architecture

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). The timing for asynchronous clocks is different in that the t_{CO} time is extended by the amount of time that it takes for the signal to propagate through the array and reach the clock network, and the t_{SU} time is reduced. Please see the application note titled "Understanding CoolRunner Clocking Options" for more detail.

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 enabled and 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.

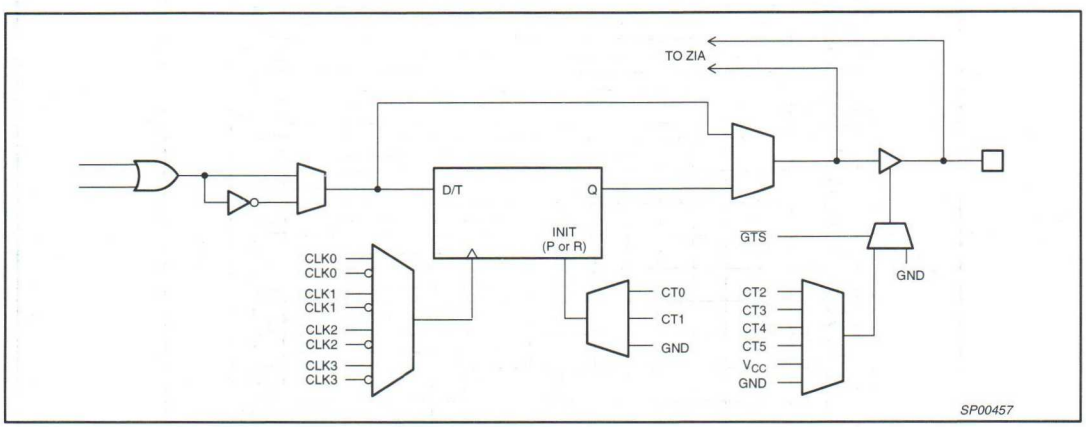


Figure 3. PZ3064 Macrocell Architecture

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.

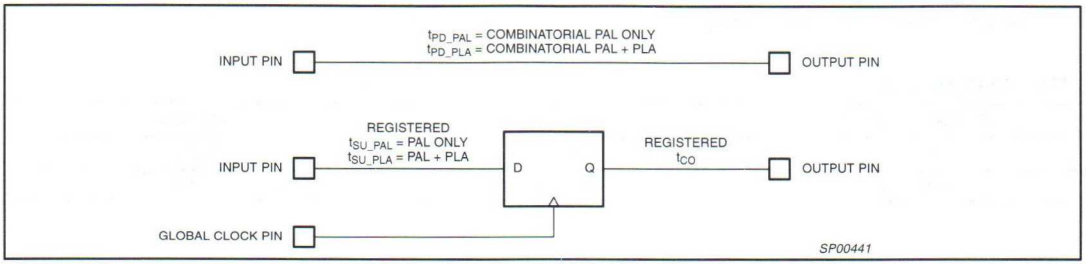


Figure 4. CoolRunner™ Timing Model

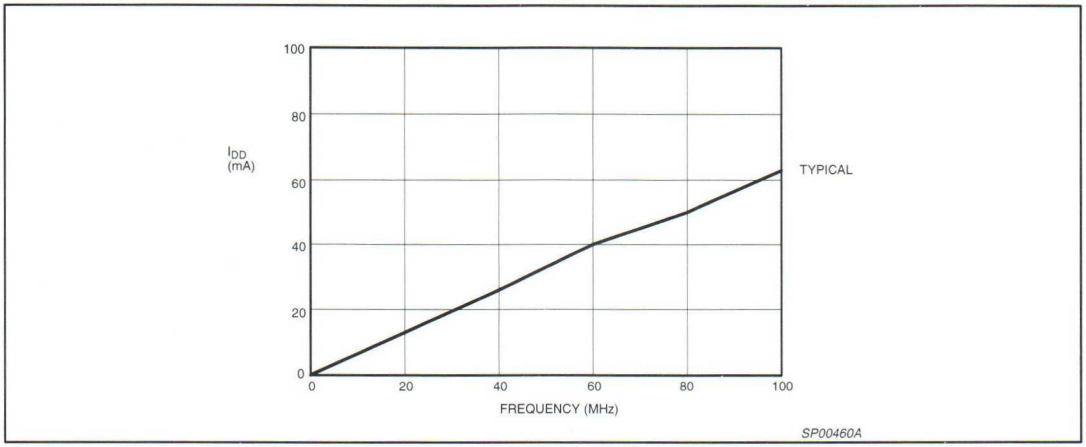


Figure 5. I_{DD} vs. Frequency @ V_{DD} = 3.3V, 25°C

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[2]	–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.
2. The chip supply voltage must rise monotonically.

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: $0°C \le T_{amb} \le +70°C$; $3.0V \le V_{DD} \le 3.6V$

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}[1]	Standby current	V_{DD} = 3.6V, T_{amb} = 0°C		50	µA
I_{DDD}[1, 2]	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[3]	1 pin at a time for no longer than 1 second	–5	–100	mA
C_{IN}	Input pin capacitance[3]	T_{amb} = 25°C, f = 1MHz		8	pF
C_{CLK}	Clock input capacitance[3]	T_{amb} = 25°C, f = 1MHz	5	12	pF
$C_{I/O}$	I/O pin capacitance[3]	T_{amb} = 25°C, f = 1MHz		10	pF

NOTES:

1. See Table 2, page 63 for typical value.
2. 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.
3. Typical values, not tested.

AC ELECTRICAL CHARACTERISTICS[1] FOR COMMERCIAL GRADE DEVICES

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

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 (global synchronous clock from pin)	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: $-40°C \le T_{amb} \le +85°C$; $3.0V \le V_{DD} \le 3.6V$

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}[1]	Standby current	V_{DD} = 3.6V, T_{amb} = –40°C		50	µA
I_{DDD}[1, 2]	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

NOTES:
1. See Table 2, page 63 for typical values.
2. 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.
3. Typical values, not tested.

AC ELECTRICAL CHARACTERISTICS[1] FOR INDUSTRIAL GRADE DEVICES

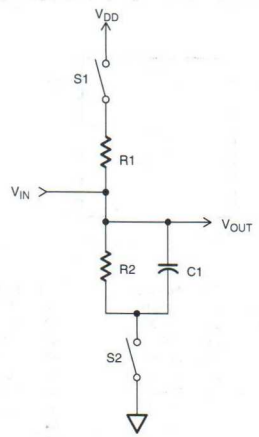
Industrial: $-40°C \le T_{amb} \le +85°C$; $3.0V \le V_{DD} \le 3.6V$

SYMBOL	PARAMETER	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 (global synchronous clock from pin)	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, and 3-State levels are measured 0.5V from steady-state active level.

SP00461B

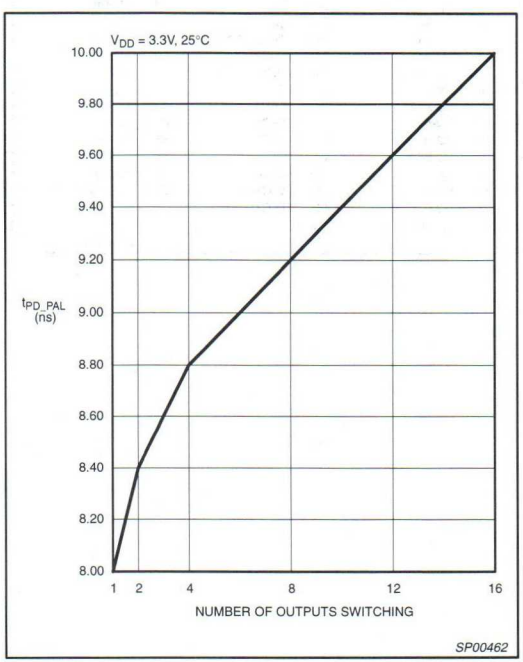
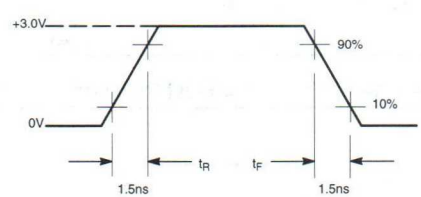


Figure 6. t_{PD_PAL} vs. Outputs Switching

VOLTAGE WAVEFORM



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

Input Pulses

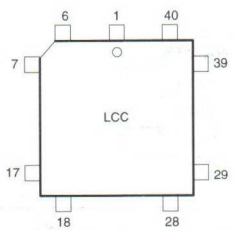
Table 3. t_{PD_PAL} vs. Number of Outputs Switching

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

PIN DESCRIPTIONS

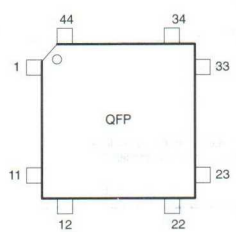
PZ3064 – 44-Pin Plastic Leaded Chip Carrier



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
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	22	GND	37	I/O-D11
8	I/O-A11	23	V_{DD}	38	I/O-D8
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	28	I/O-C7	43	IN0-CK0
14	I/O-B13	29	I/O-C8	44	IN2-gtsn
15	V_{DD}	30	GND		

SP00452B

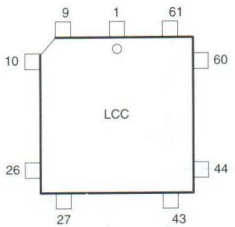
PZ3064 – 44-Pin Thin Quad Flat Package



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
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	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	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		

SP00453A

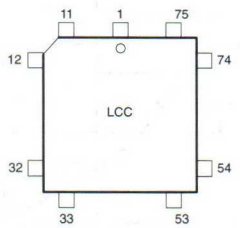
PZ3064 – 68-Pin Plastic Leaded Chip Carrier



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-C13
4	I/O-A0/CK3	27	I/O-B7	50	I/O-C15
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
12	I/O-A8	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	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		

SP00454A

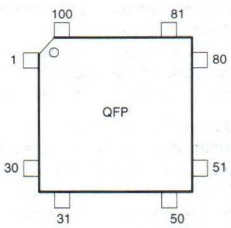
PZ3064 – 84-Pin Plastic Leaded Chip Carrier



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
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	42	GND	70	I/O-D9
15	I/O-A9	43	V_{DD}	71	I/O-D8
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-A15	50	I/O-C5	78	V_{DD}
23	I/O-B15	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

SP00455A

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	40	GND	74	NC
7	NC	41	V_{DD}	75	I/O-D8
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	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	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}		

SP00456A

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

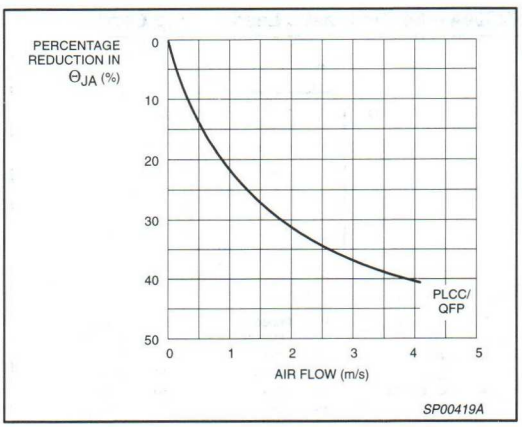


Figure 7. Average Effect of Airflow on Θ_{JA}

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 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 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 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 PZ3128 CPLDs are supported by industry standard CAE tools (Cadence, Mentor, Minc, Exemplar Logic, Synplicity, 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
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
PZ3128IS12A84	84–pin PLCC, 12ns t_{PD}	Industrial 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–S10BB1	100–pin PQFP, 10ns t_{PD}	Commercial temp range, 3.3 volt power supply, ± 10%	SOT382–1
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
PZ3128IS12BB1	100–pin PQFP, 12ns t_{PD}	Industrial 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–S10BP	100–pin TQFP, 10ns t_{PD}	Commercial temp range, 3.3 volt power supply, ± 10%	SOT386–1
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
PZ3128IS12BP	100–pin TQFP, 12ns t_{PD}	Industrial 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–S10BE	128–pin LQFP, 10ns t_{PD}	Commercial temp range, 3.3 volt power supply, ± 10%	SOT425–1
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
PZ3128IS12BE	128–pin LQFP, 12ns t_{PD}	Industrial 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–S10BB2	160–pin PQFP, 10ns t_{PD}	Commercial temp range, 3.3 volt power supply, ± 10%	SOT322–2
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
PZ3128IS12BB2	160–pin PQFP, 12ns t_{PD}	Industrial 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.

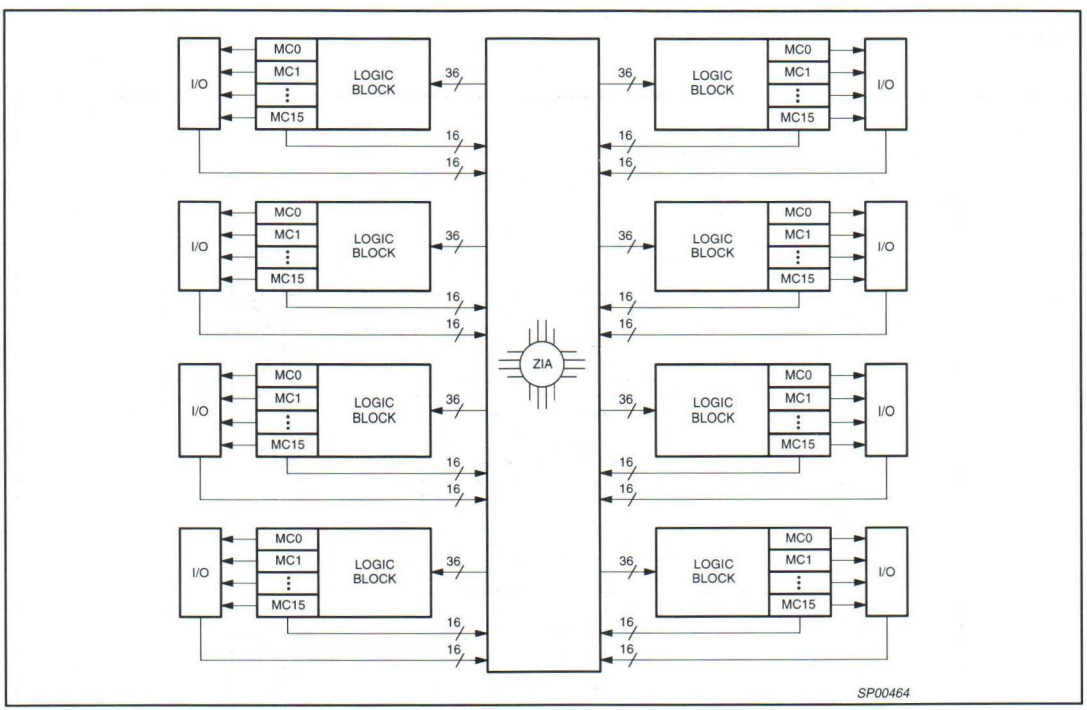


Figure 1. Philips XPLA CPLD Architecture

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 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 PZ3128 using 6 to 37 product terms is 12.5ns (10ns for the PAL + 2.5ns for the PLA).

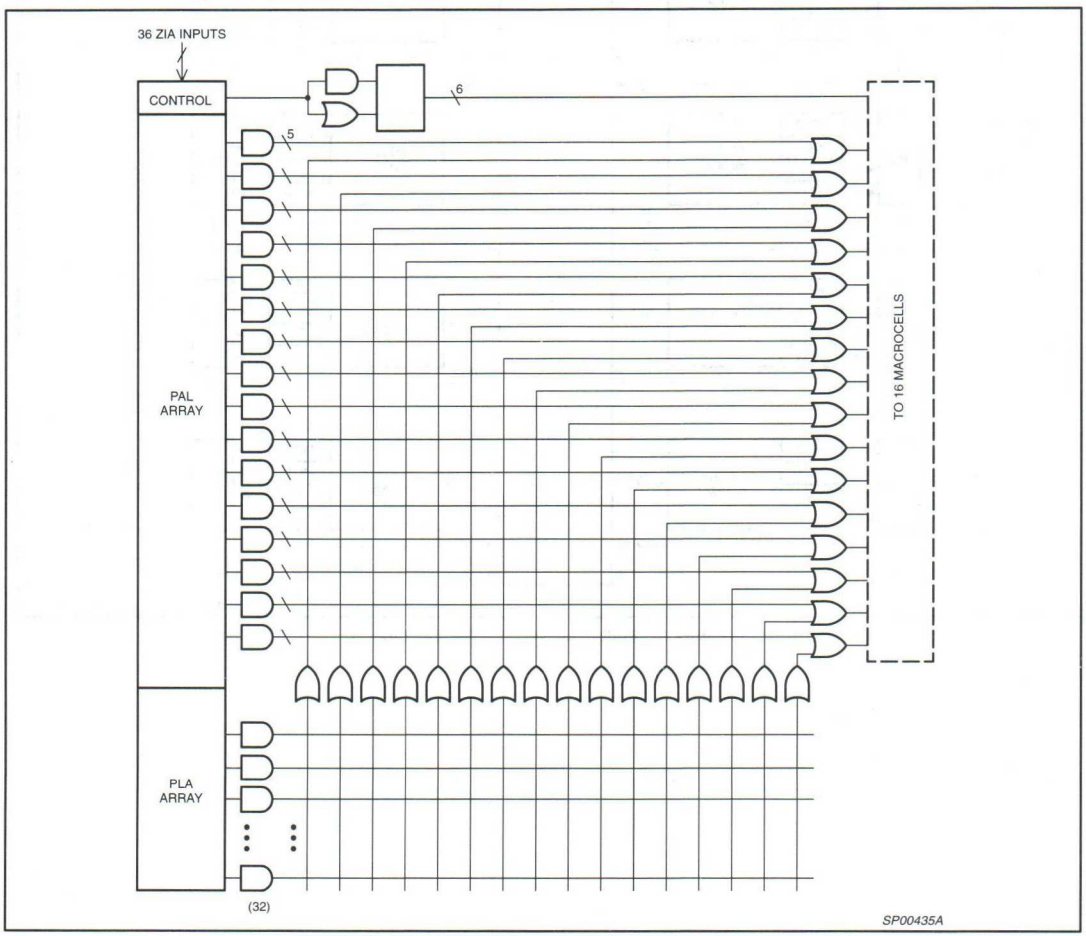


Figure 2. Philips XPLA Logic Block Architecture

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). The timing for asynchronous clocks is different in that the t_{CO} time is extended by the amount of time that it takes for the signal to propagate through the array and reach the clock network, and the t_{SU} time is reduced. Please see the application note titled "Understanding CoolRunner Clocking Options" for more detail.

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 enabled and 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.

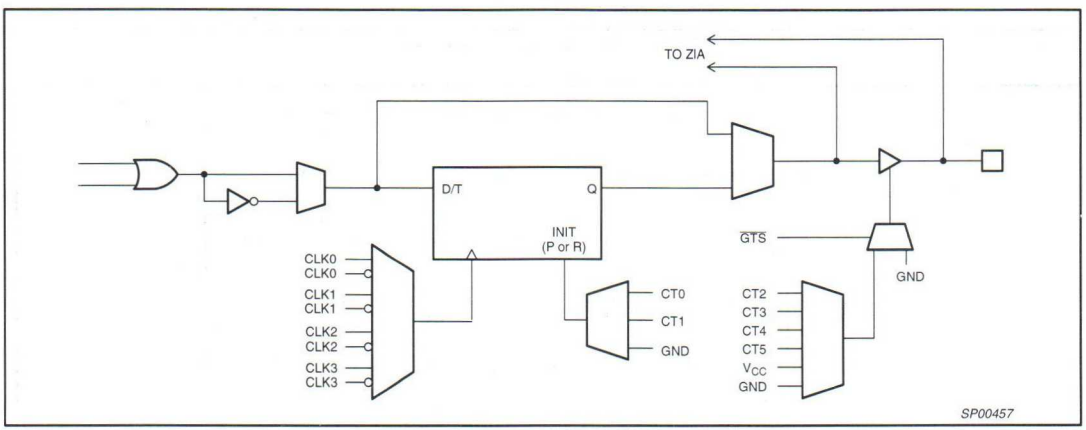


Figure 3. PZ3128 Macrocell Architecture

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 (data taken w/eight up/down, loadable 16 bit counters @ 3.3V, 25°C).

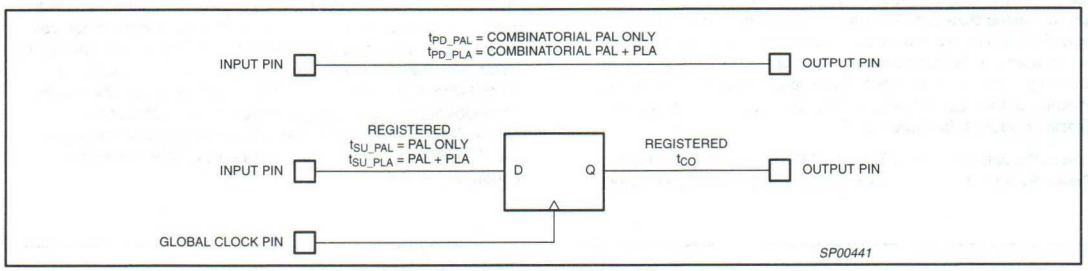


Figure 4. CoolRunner™ Timing Model

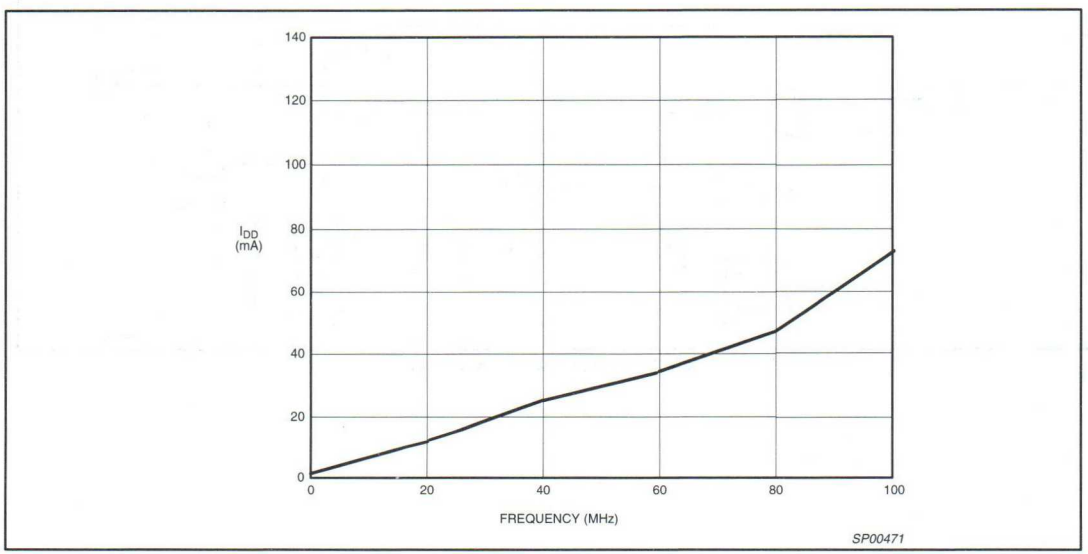


Figure 5. I_{DD} vs. Frequency @ V_{DD} = 3.3V, 25°C

Table 2. I_{DD} vs. Frequency

V_{DD} = 3.3V

FREQUENCY (MHz)	0	1	20	40	60	80	100
Typical I_{DD} (mA)	.03	.06	12	24	35	46	63

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
  - Reconfigurability 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[2]	–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.
2. The chip supply voltage must rise monotonically.

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: $0°C \le T_{amb} \le +70°C$; $3.0V \le V_{DD} \le 3.6V$

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}[1]	Standby current	V_{DD} = 3.6V, T_{amb} = 0°C		60	µA
I_{DDD}[1, 2]	Dynamic current	V_{DD} = 3.6V, T_{amb} = 0°C @ 1MHz		2	mA
		V_{DD} = 3.6V, T_{amb} = 0°C @ 50MHz		50	mA
I_{OS}	Short circuit output current[3]	1 pin at a time for no longer than 1 second	–50	–100	mA
C_{IN}	Input pin capacitance[3]	T_{amb} = 25°C, f = 1MHz		8	pF
C_{CLK}	Clock input capacitance[3]	T_{amb} = 25°C, f = 1MHz	5	12	pF
$C_{I/O}$	I/O pin capacitance[3]	T_{amb} = 25°C, f = 1MHz		10	pF

NOTES:
1. See Table 2 on page 76 for typical values.
2. This parameter measured with a 16-bit, loadable up/down counter loaded into every logic block, with all outputs disabled and unloaded. Inputs are tied to V_{DD} or ground. This parameter guaranteed by design and characterization, not testing.
3. Typical values, not tested.

AC ELECTRICAL CHARACTERISTICS[1] FOR COMMERCIAL GRADE DEVICES

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

SYMBOL	PARAMETER	10		12		15		UNIT
		MIN.	MAX.	MIN.	MAX.	MIN.	MAX.	
t_{PD_PAL}	Propagation delay time, input (or feedback node) to output through PAL	2	10	2	12	2	15	ns
t_{PD_PLA}	Propagation delay time, input (or feedback node) to output through PAL & PLA	3	12.5	3	14.5	3	17.5	ns
t_{CO}	Clock to out (global synchronous clock from pin)	2	7	2	8	2	9	ns
t_{SU_PAL}	Setup time (from input or feedback node) through PAL	6		7		8		ns
t_{SU_PLA}	Setup time (from input or feedback node) through PAL + PLA	8.5		9.5		10.5		ns
t_H	Hold time		0		0		0	ns
t_{CH}	Clock High time	3		4		4		ns
t_{CL}	Clock Low time	3		4		4		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		125		MHz
f_{MAX2}	Maximum internal frequency[2] $1/(t_{SUPAL} + t_{CF})$	87		74		65		MHz
f_{MAX3}	Maximum external frequency[2] $1/(t_{SUPAL} + t_{CO})$	77		66		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	2	0.5	2	10.5	2	13.5	ns
t_{PDF_PLA}	Input (or feedback node) to internal feedback node delay time through PAL+PLA	3	11	3	13	3	16	ns
t_{CF}	Clock to internal feedback node delay time		5.5		6.5		7.5	ns
t_{INIT}	Delay from valid V_{DD} to valid reset		50		50		50	µs
t_{ER}	Input to output disable[3]		12.5		14		17	ns
t_{EA}	Input to output valid		12.5		14		17	ns
t_{RP}	Input to register preset		14		16		19	ns
t_{RR}	Input to register reset		14		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: $-40°C \le T_{amb} \le +85°C$; $3.0V \le V_{DD} \le 3.6V$

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}[1]	Standby current	V_{DD} = 3.6V, T_{amb} = –40°C		75	µA
I_{DDD}[1, 2]	Dynamic current	V_{DD} = 3.6V, T_{amb} = –40°C @ 1MHz		2	mA
		V_{DD} = 3.6V, T_{amb} = –40°C @ 50MHz		50	mA
I_{OS}	Short circuit output current[3]	1 pin at a time for no longer than 1 second	–50	–130	mA
C_{IN}	Input pin capacitance[3]	T_{amb} = 25°C, f = 1MHz		8	pF
C_{CLK}	Clock input capacitance[3]	T_{amb} = 25°C, f = 1MHz	5	12	pF
$C_{I/O}$	I/O pin capacitance[3]	T_{amb} = 25°C, f = 1MHz		10	pF

NOTES:
1. See Table 2 on page 76 for typical values.
2. This parameter measured with a 16–bit, loadable up/down counter loaded into every logic block, with all outputs DISabled and unloaded. Inputs are tied to V_{DD} or ground. This parameter guaranteed by design and characterization, not testing.
3. Typical values, not tested.

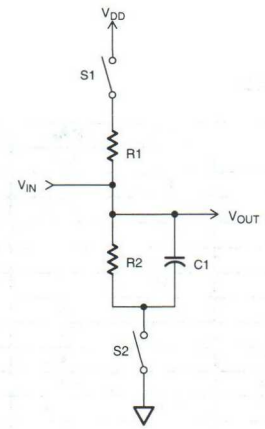
AC ELECTRICAL CHARACTERISTICS[1] FOR INDUSTRIAL GRADE DEVICES

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

SYMBOL	PARAMETER	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 (global synchronous clock from pin)	2	7.5	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	3		4		ns
t_{CL}	Clock Low time	3		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] $1/(t_{CH} + t_{CL})$	167		125		MHz
f_{MAX2}	Maximum internal frequency[2] $1/(t_{SUPAL} + t_{CF})$	77		65		MHz
f_{MAX3}	Maximum external frequency[2] $1/(t_{SUPAL} + t_{CO})$	69		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		7.5	ns
t_{INIT}	Delay from valid V_{DD} to valid reset		50		50	µs
t_{ER}	Input to output disable[3]		13		15.5	ns
t_{EA}	Input to output valid		13		15.5	ns
t_{RP}	Input to register preset		15		17	ns
t_{RR}	Input to register reset		15		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.

SWITCHING CHARACTERISTICS

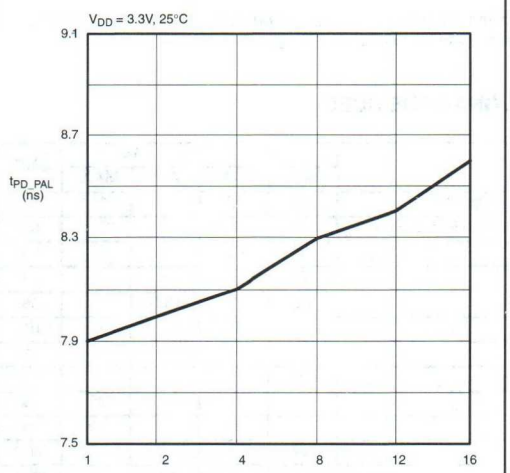


COMPONENT	VALUES
R1	390Ω
R2	390Ω
C1	35pF

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.

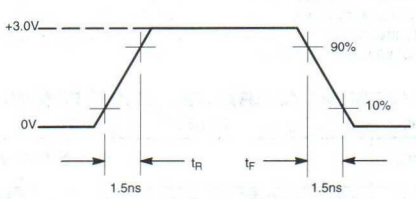
SP00477



SP00466A

Figure 6. t_{PD_PAL} vs. Outputs Switching

VOLTAGE WAVEFORM



SP00368

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

Input Pulses

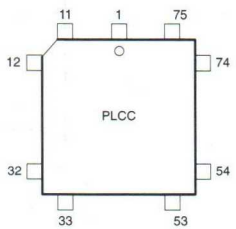
Table 8. t_{PD_PAL} vs. Number of Outputs Switching

V_{DD} = 3.3V

NUMBER OF OUTPUTS	1	2	4	8	12	16
Typical (ns)	7.9	8	8.1	8.3	8.4	8.6

PIN DESCRIPTIONS

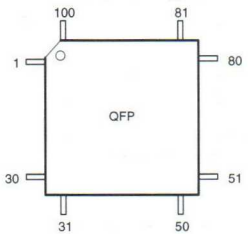
84-Pin Plastic Leaded Chip Carrier



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

SP00467

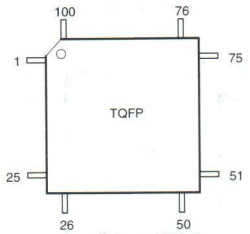
100-Pin Plastic Quad Flat Package



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-D0/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-C4	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}		

SP00468

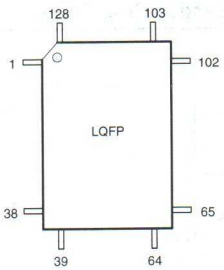
100-Pin Thin Quad Flat Package



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		

SP00485

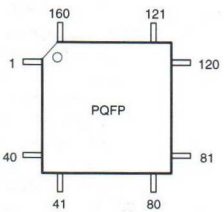
128-Pin Low Profile Quad Flat Package



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		

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

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

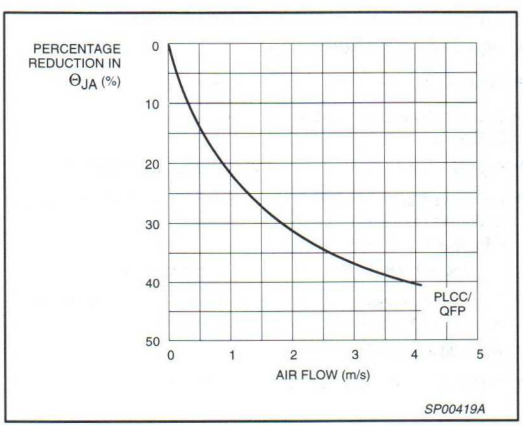


Figure 7. Average Effect of Airflow on Θ_{JA}

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 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, Exemplar Logic, Minc, 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).

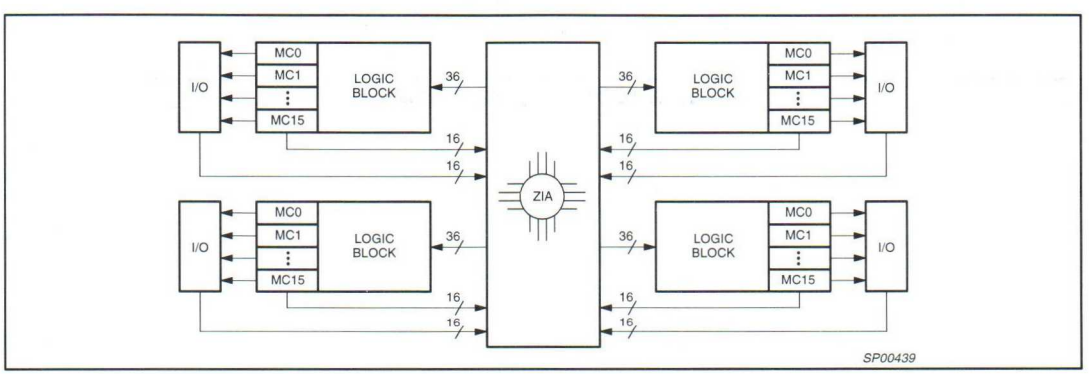


Figure 1. Philips XPLA CPLD Architecture

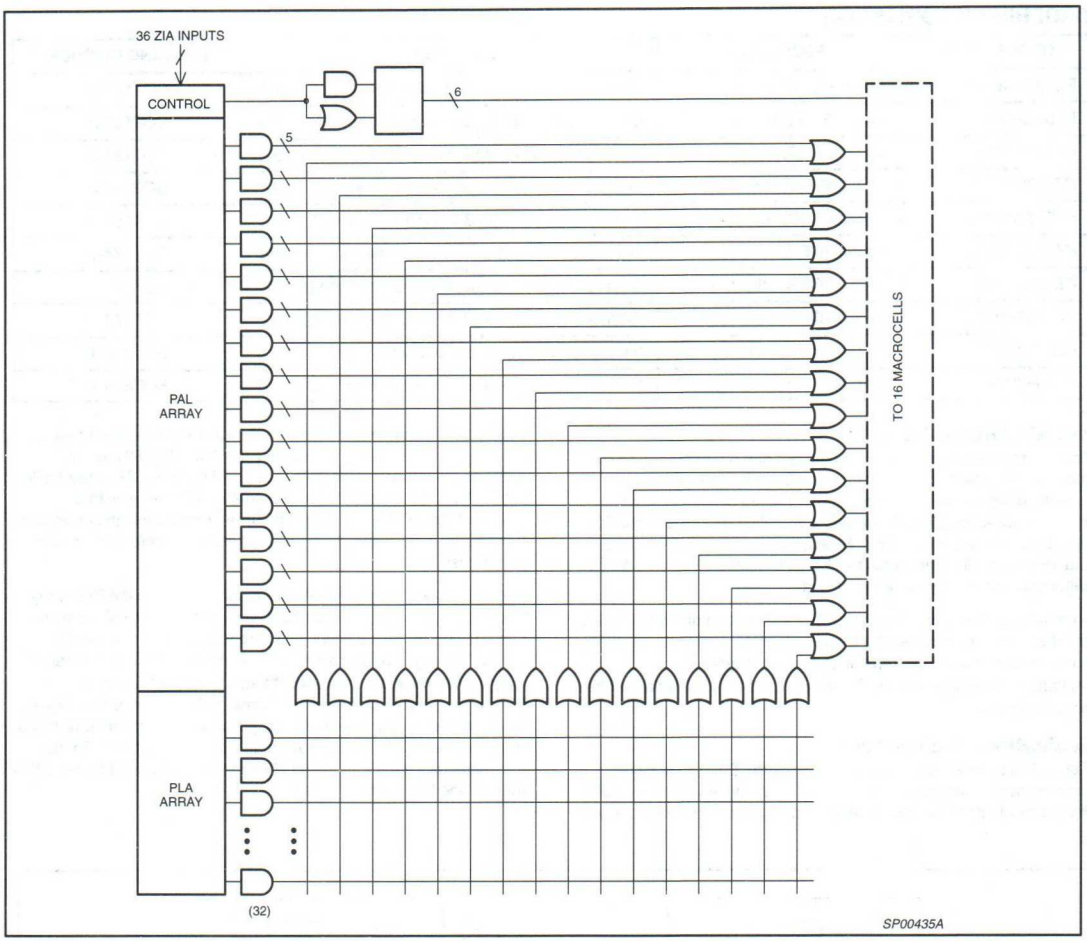


Figure 2. Philips XPLA Logic Block Architecture

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). The timing for asynchronous clocks is different in that the t_{CO} time is extended by the amount of time that it takes for the signal to propagate through the array and reach the clock network, and the t_{SU} time is reduced. Please see the application note titled "Understanding CoolRunner Clocking Options" for more detail.

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 enabled and 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.

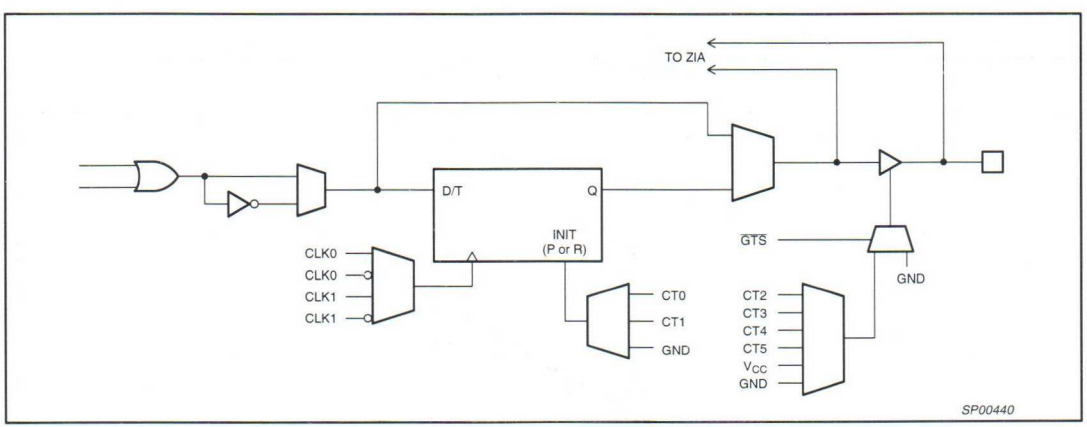


Figure 3. PZ5032 Macrocell Architecture

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.

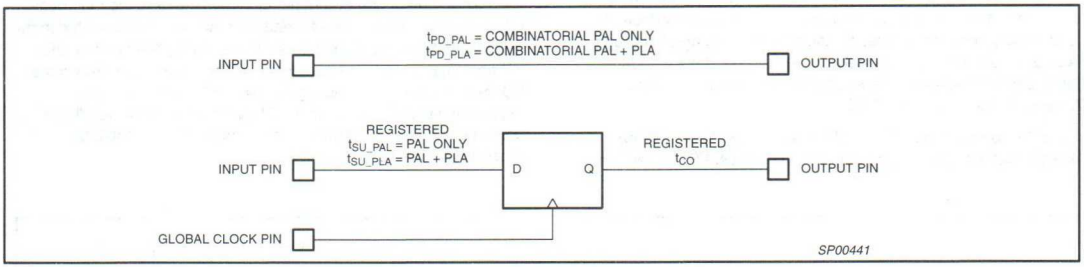


Figure 4. CoolRunner™ Timing Model

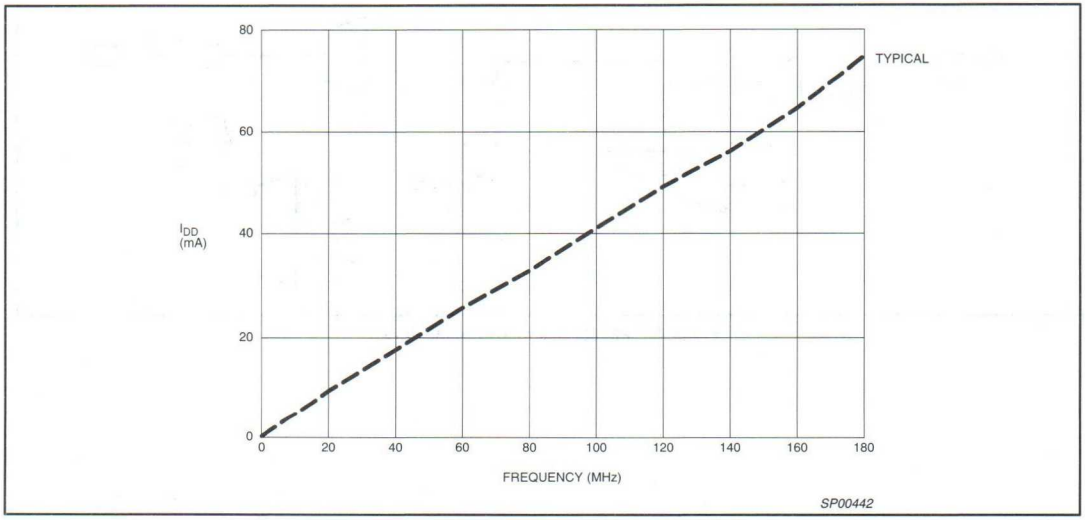


Figure 5. I_{DD} vs. Frequency @ V_{DD} = 5.0V, 25°C

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[2]	–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.
2. The chip supply voltage must rise monotonically.

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: $0°C \le T_{amb} \le +70°C$; $4.75V \le V_{DD} \le 5.25V$

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}[1]	Standby current	V_{DD} = 5.25V, T_{amb} = 0°C		75	µA
I_{DDD}[1, 2]	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[3]	1 pin at a time for no longer than 1 second	–50	–200	mA
C_{IN}	Input pin capacitance[3]	T_{amb} = 25°C, f = 1MHz		8	pF
C_{CLK}	Clock input capacitance[3]	T_{amb} = 25°C, f = 1MHz	5	12	pF
$C_{I/O}$	I/O pin capacitance[3]	T_{amb} = 25°C, f = 1MHz		10	pF

NOTES:
1. See Table 2 on page 90 for typical values.
2. 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.
3. Typical values, not tested.

AC ELECTRICAL CHARACTERISTICS[1] FOR COMMERCIAL GRADE DEVICES

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

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 (global synchronous clock from pin)	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[2, 3]		11		12.5		15	ns
t_{EA}	Input to output valid[2]		11		12.5		15	ns
t_{RP}	Input to register preset[2]		11		12.5		15	ns
t_{RR}	Input to register reset[2]		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}[1]	Standby current	V_{DD} = 5.5V, T_{amb} = –40°C		95	µA
I_{DDD}[1, 2]	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[3]	1 pin at a time for no longer than 1 second	–50	–230	mA
C_{IN}	Input pin capacitance[3]	T_{amb} = 25°C, f = 1MHz		8	pF
C_{CLK}	Clock input capacitance[3]	T_{amb} = 25°C, f = 1MHz	5	12	pF
$C_{I/O}$	I/O pin capacitance[3]	T_{amb} = 25°C, f = 1MHz		10	pF

NOTES:
1. See Table 2 on page 90 for typical values.
2. 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.
3. Typical values, not tested.

AC ELECTRICAL CHARACTERISTICS[1] FOR INDUSTRIAL GRADE DEVICES

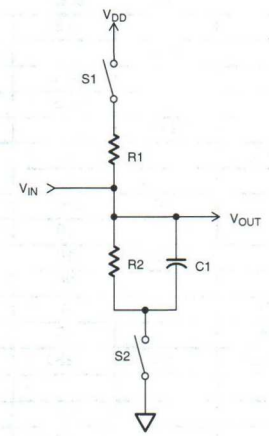
Industrial: $-40°C \le T_{amb} \le +85°C$; $4.5V \le V_{DD} \le 5.5V$

SYMBOL	PARAMETER	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 (global synchronous clock from pin)	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[2, 3]		12		15	ns
t_{EA}	Input to output valid[2]		12		15	ns
t_{RP}	Input to register preset[2]		12		15	ns
t_{RR}	Input to register reset[2]		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.

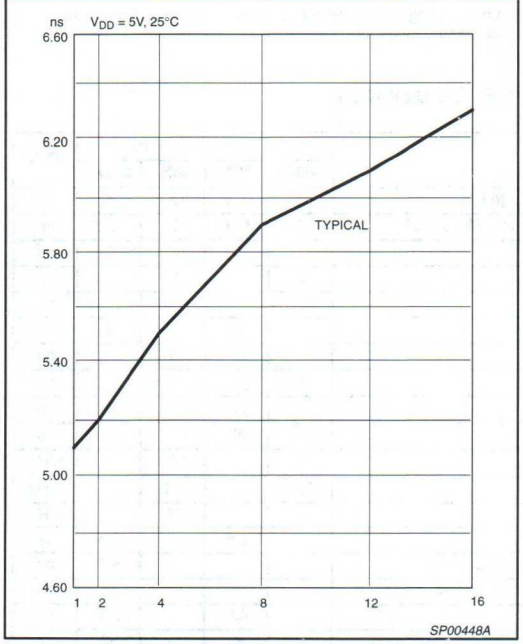


COMPONENT	VALUES
R1	470Ω
R2	250Ω
C1	35pF

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.

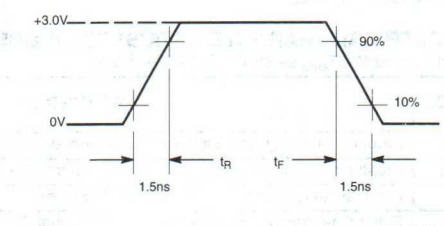
SP00476



SP00448A

Figure 6. t_{PD_PAL} vs Outputs switching

VOLTAGE WAVEFORM



SP00368

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

Input Pulses

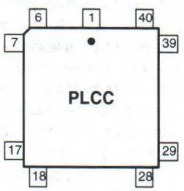
Table 3. t_{PD_PAL} vs # of Outputs switching

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

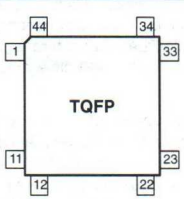
PZ5032 – 44-Pin Plastic Leaded Chip Carrier



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

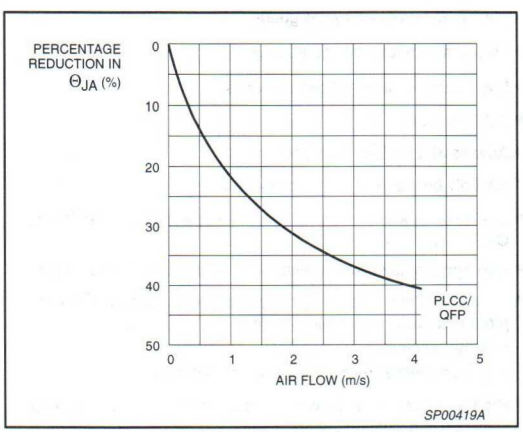


Figure 7. Average Effect of Airflow on Θ_{JA}

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 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, Exemplar Logic, Minc, 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).

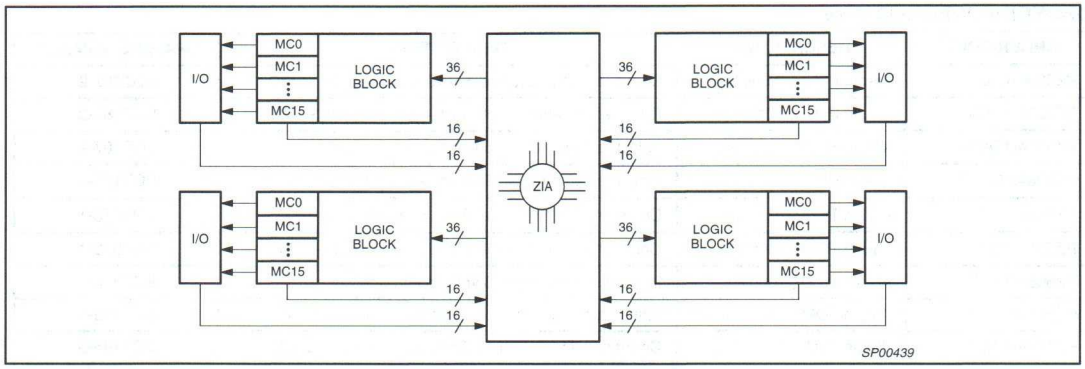


Figure 1. Philips XPLA CPLD Architecture

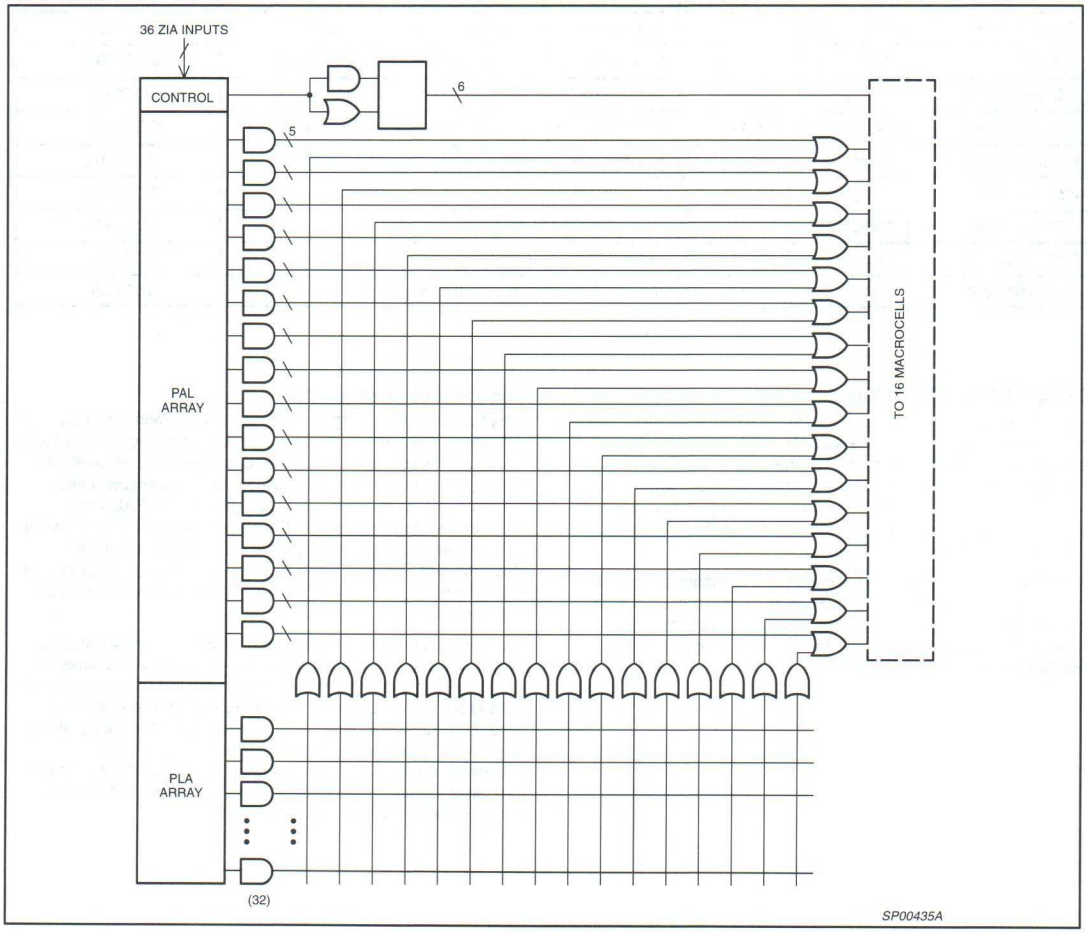


Figure 2. Philips XPLA Logic Block Architecture

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). The timing for asynchronous clocks is different in that the t_{CO} time is extended by the amount of time that it takes for the signal to propagate through the array and reach the clock network, and the t_{SU} time is reduced. Please see the application note titled "Understanding CoolRunner Clocking Options" for more detail.

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 enabled and 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.

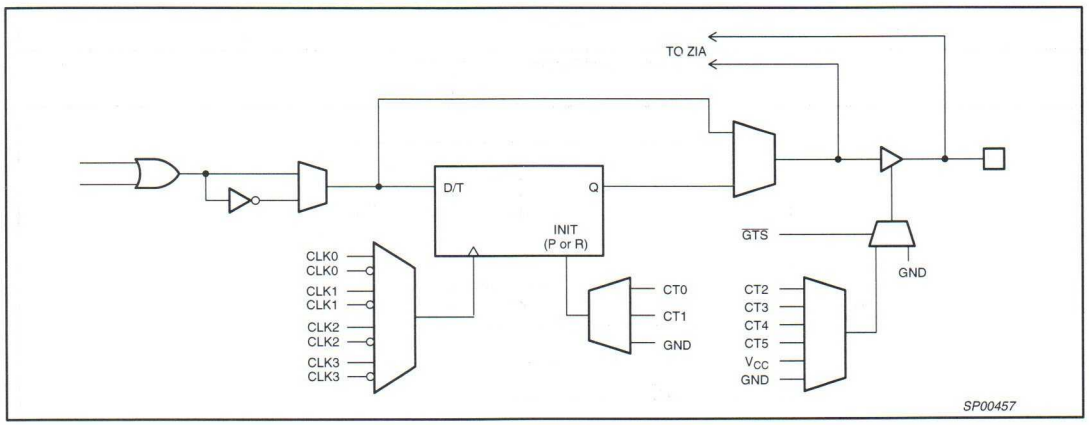


Figure 3. PZ5064 Macrocell Architecture

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.

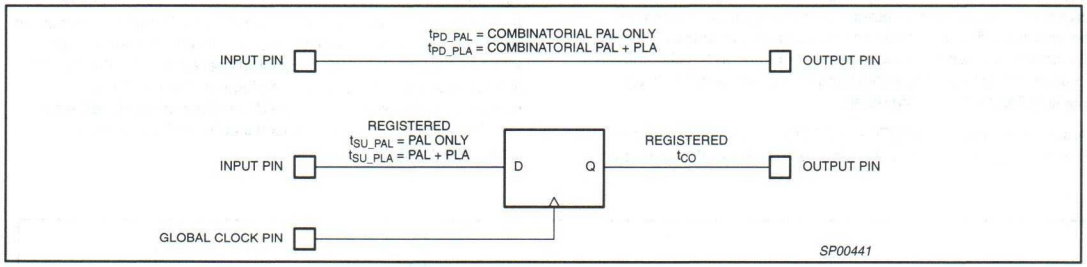


Figure 4. CoolRunner™ Timing Model

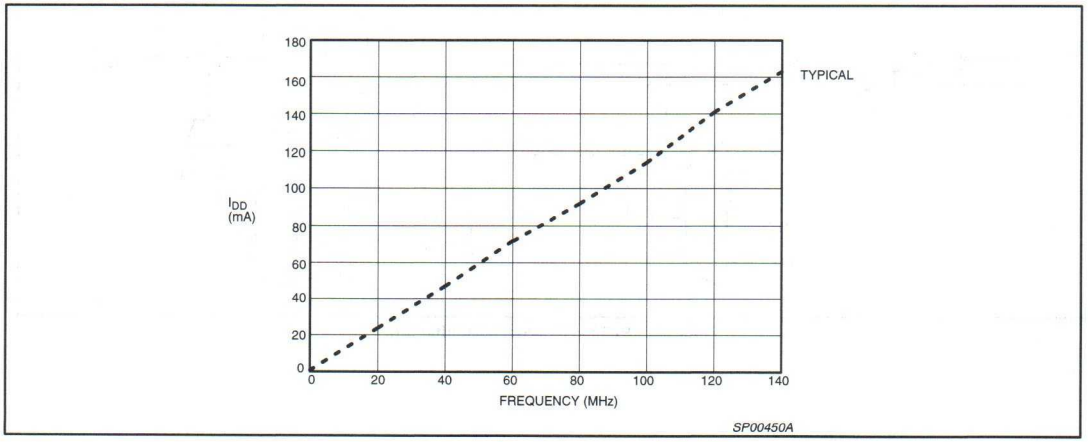


Figure 5. I_{DD} vs. Frequency @ V_{DD} = 5.0V, 25°C

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[2]	–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.
2. The chip supply voltage must rise monotonically.

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: $0°C \le T_{amb} \le +70°C$; $4.75V \le V_{DD} \le 5.25V$

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}[1]	Standby current	V_{DD} = 5.25V, T_{amb} = 0°C		80	µA
I_{DDD}[1, 2]	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[3]	1 pin at a time for no longer than 1 second	–50	–200	mA
C_{IN}	Input pin capacitance[3]	T_{amb} = 25°C, f = 1MHz		8	pF
C_{CLK}	Clock input capacitance[3]	T_{amb} = 25°C, f = 1MHz	5	12	pF
$C_{I/O}$	I/O pin capacitance[3]	T_{amb} = 25°C, f = 1MHz		10	pF

NOTES:
1. See Table 2 on page 100 for typical values.
2. 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.
3. Typical values, not tested.

AC ELECTRICAL CHARACTERISTICS[1] FOR COMMERCIAL GRADE DEVICES

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

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 (global synchronous clock from pin)	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[2, 3]		10		12	ns
t_{EA}	Input to output valid[2]		10		12	ns
t_{RP}	Input to register preset[2]		10		12.5	ns
t_{RR}	Input to register reset[2]		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: $-40°C \le T_{amb} \le +85°C$; $4.5V \le V_{DD} \le 5.5V$

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}[1]	Standby current	V_{DD} = 5.5V, T_{amb} = –40°C		100	µA
I_{DDD}[1, 2]	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[3]	1 pin at a time for no longer than 1 second	–50	–230	mA
C_{IN}	Input pin capacitance[3]	T_{amb} = 25°C, f = 1MHz		8	pF
C_{CLK}	Clock input capacitance[3]	T_{amb} = 25°C, f = 1MHz	5	12	pF
$C_{I/O}$	I/O pin capacitance[3]	T_{amb} = 25°C, f = 1MHz		10	pF

NOTE:
1. See Table 2 on page 100 for typical values.
2. 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.
3. Typical values, not tested.

AC ELECTRICAL CHARACTERISTICS[1] FOR INDUSTRIAL GRADE DEVICES

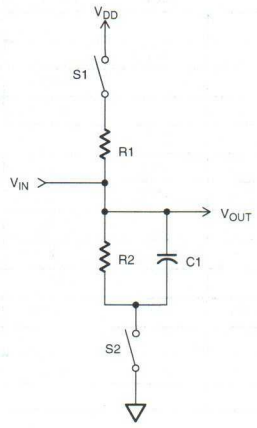
Industrial: $-40°C \le T_{amb} \le +85°C$; $4.5V \le V_{DD} \le 5.5V$

SYMBOL	PARAMETER	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 (global synchronous clock from pin)	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[2, 3]		12		13	ns
t_{EA}	Input to output valid[2]		12		13	ns
t_{RP}	Input to register preset[2]		12.5		13.5	ns
t_{RR}	Input to register reset[2]		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.

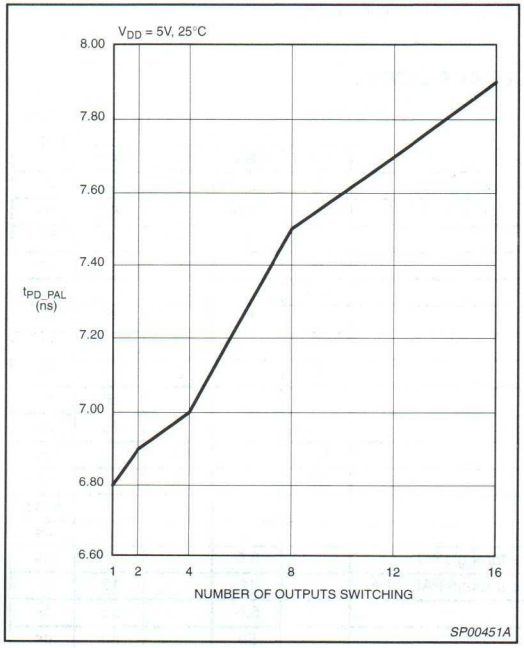


COMPONENT	VALUES
R1	470Ω
R2	250Ω
C1	35pF

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

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

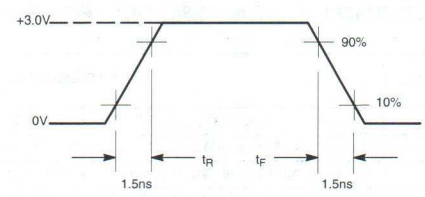
SP00458A



SP00451A

Figure 6. t_{PD_PAL} vs. Outputs Switching

VOLTAGE WAVEFORM



SP00368

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

Input Pulses

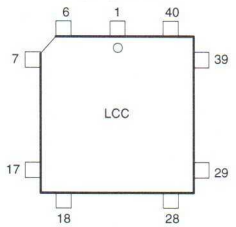
Table 3. t_{PD_PAL} vs. Number of Outputs Switching

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

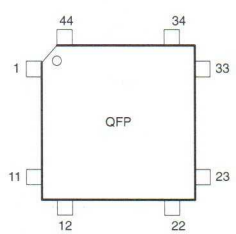
PZ5064 – 44-Pin Plastic Leaded Chip Carrier



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
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	22	GND	37	I/O-D11
8	I/O-A11	23	V_{DD}	38	I/O-D8
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	28	I/O-C7	43	IN0-CK0
14	I/O-B13	29	I/O-C8	44	IN2-gtsn
15	V_{DD}	30	GND		

SP00452B

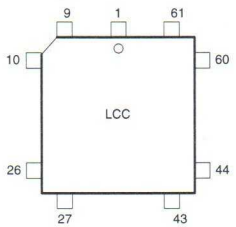
PZ5064 – 44-Pin Thin Quad Flat Package



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
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	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	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		

SP00453A

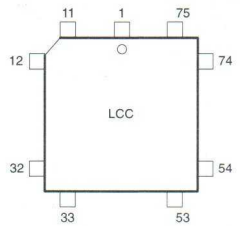
PZ5064 – 68-Pin Plastic Leaded Chip Carrier



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-C13
4	I/O-A0/CK3	27	I/O-B7	50	I/O-C15
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
12	I/O-A8	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	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		

SP00454A

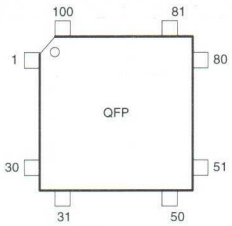
PZ5064 – 84-Pin Plastic Leaded Chip Carrier



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
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	42	GND	70	I/O-D9
15	I/O-A9	43	V_{DD}	71	I/O-D8
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-A15	50	I/O-C5	78	V_{DD}
23	I/O-B15	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

SP00455A

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	40	GND	74	NC
7	NC	41	V_{DD}	75	I/O-D8
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	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	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}		

SP00456A

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

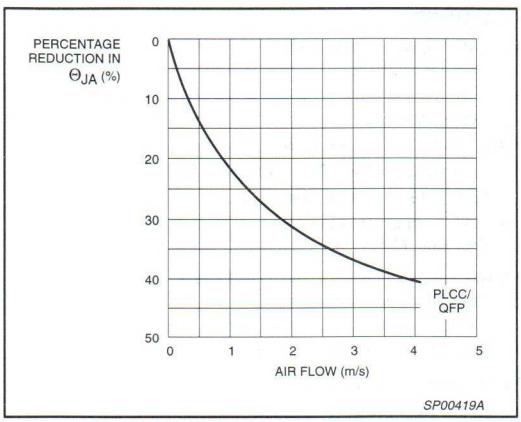


Figure 7. Average Effect of Airflow on Θ_{JA}

128 macrocell CPLD 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 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 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)	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 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 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, Exemplar Logic, Minc, 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
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
PZ528IS10A84	84–pin PLCC, 10ns t_{PD}	Industrial temp range, 5 volt power supply, ± 10%	SOT189–3
PZ5128IS15A84	84-pin PLCC, 15ns 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
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
PZ5128IS10BB1	100–pin PQFP, 10ns t_{PD}	Industrial temprange, 5 volt power supply, ± 10%	SOT382–1
PZ5128IS15BB1	100-pin PQFP, 15ns 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
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
PZ5128IS10BP	100–pin TQFP, 10ns t_{PD}	Industrial temp range, 5 volt power supply, ± 10%	SOT386–1
PZ5128IS15BP	100-pin TQFP, 15ns t_{PD}	Industrial temp range, 5 volt power supply, ± 10%	SOT386-1
PZ5128–S7BE	128–LQFP, 7.5ns t_{PD}	Commercial temp range, 5 volt power supply, ± 5%	SOT425–1
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
PZ5128IS10BE	128–pin LQFP, 10ns t_{PD}	Industrial temp range, 5 volt power supply, ± 10%	SOT425–1
PZ5128IS15BE	128-pin LQFP, 15ns 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
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
PZ5128IS10BB2	160–pin PQFP, 10ns t_{PD}	Industrial temp range, 5 volt poweer supply, ± 10%	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.

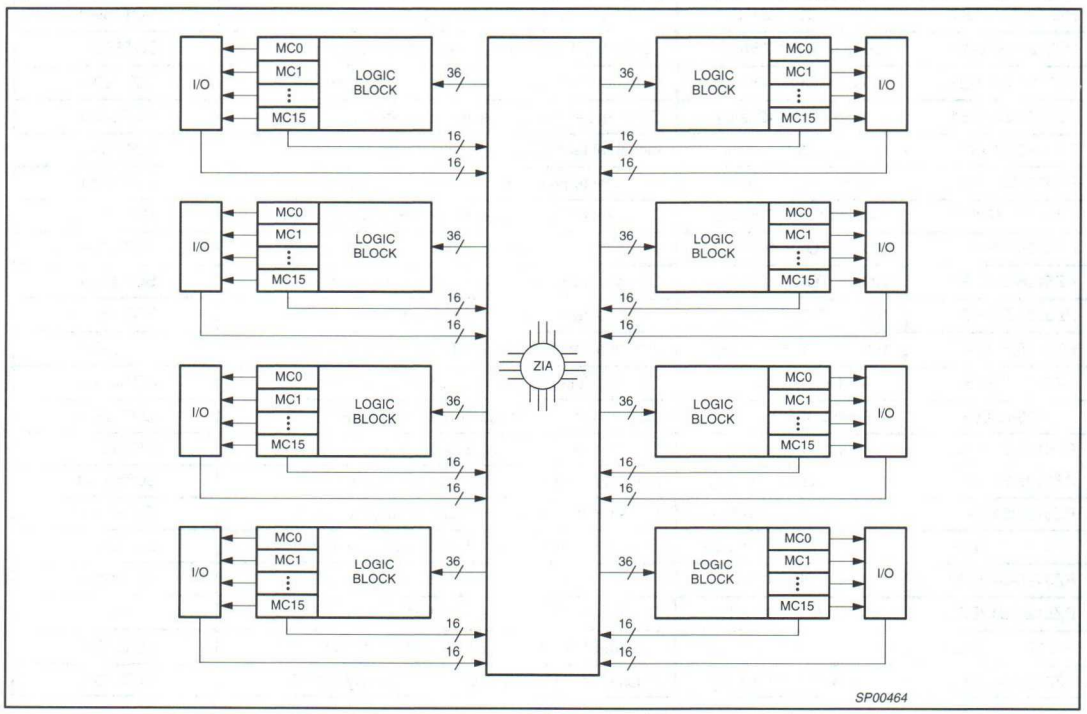


Figure 1. Philips XPLA CPLD Architecture

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 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).

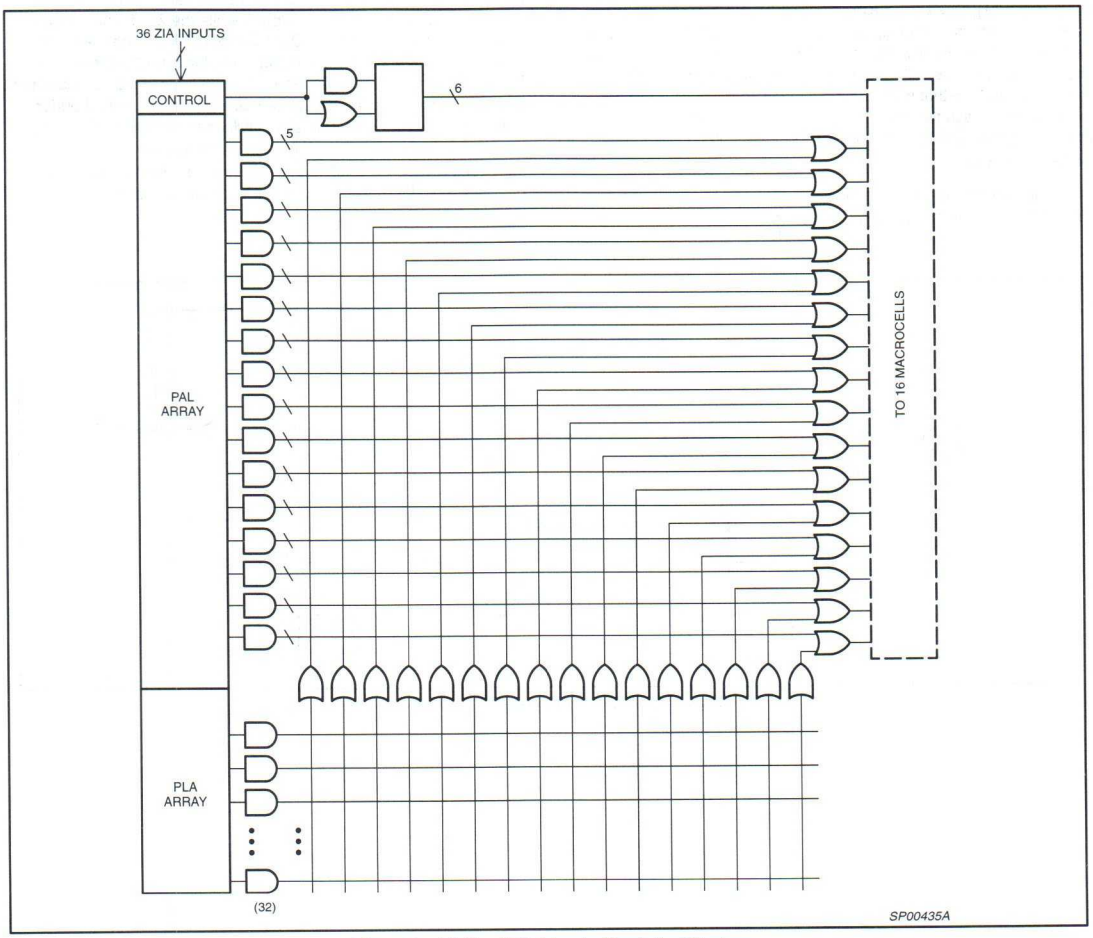


Figure 2. Philips XPLA Logic Block Architecture

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). The timing for asynchronous clocks is different in that the t_{CO} time is extended by the amount of time that it takes for the signal to propagate through the array and reach the clock network, and the t_{SU} time is reduced. Please see the application note titled "Understanding CoolRunner Clocking Options" for more detail.

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 enabled and 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.

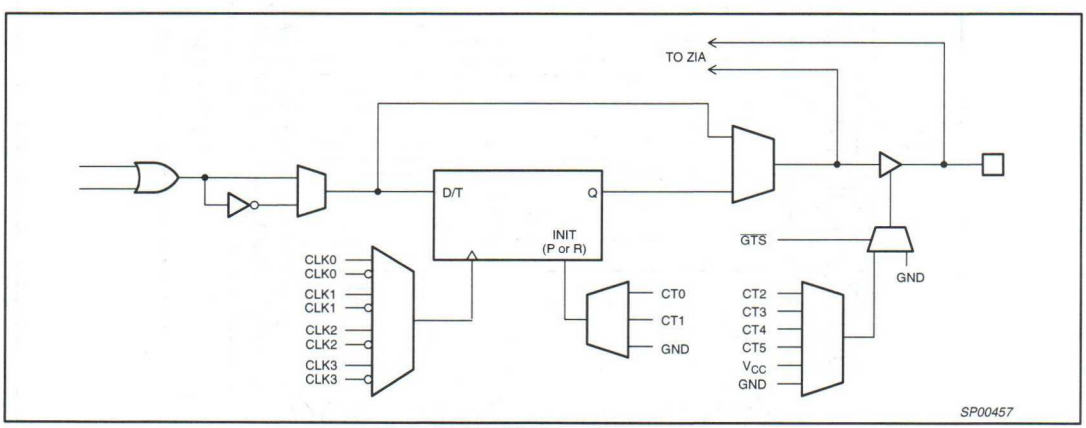


Figure 3. PZ5128 Macrocell Architecture

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 (data taken w/eight up/down, loadable 16 bit counters@5V, 25°C.

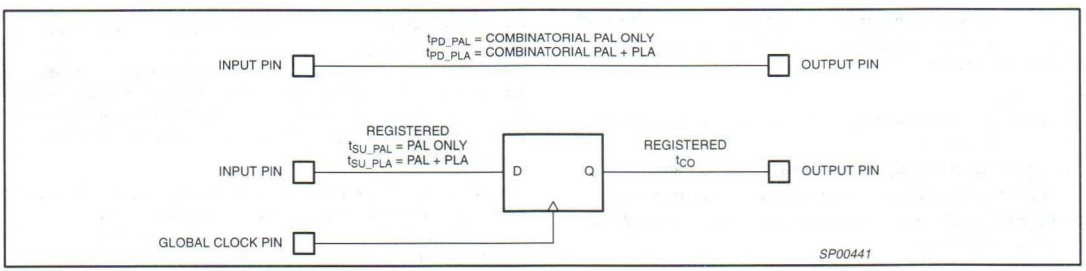


Figure 4. CoolRunner™ Timing Model

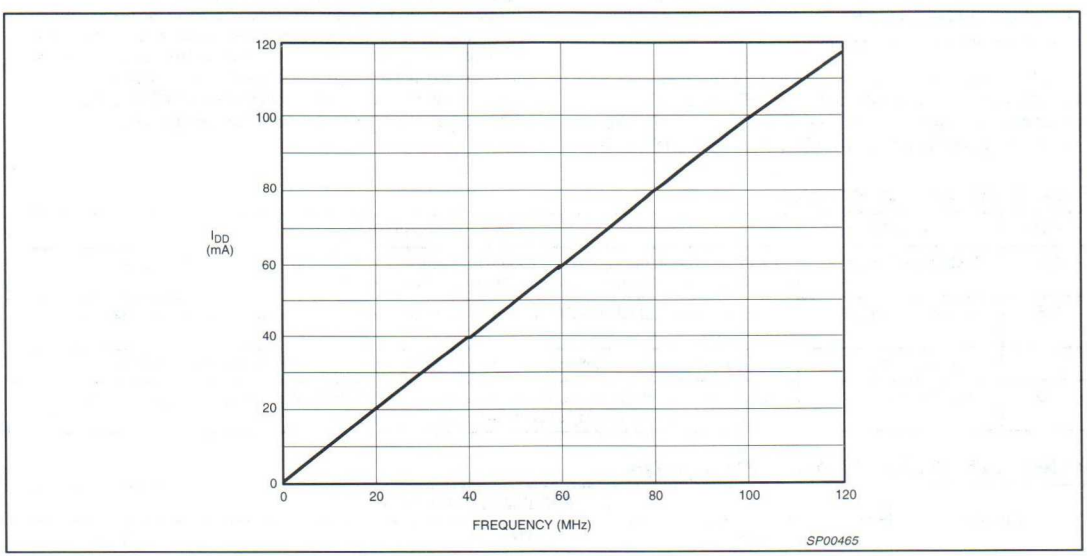


Figure 5. I_{DD} vs. Frequency @ V_{DD} = 5.0V, 25°C

Table 2. I_{DD} vs. Frequency

V_{DD} = 5.00V

FREQUENCY (MHz)	0	1	20	40	60	80	100	120
Typical I_{DD} (mA)	0.5	1	20	40	60	80	99	118

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
  - Reconfigurability 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[2]	–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.
2. The chip supply voltage must rise monotonically.

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: $0°C \le T_{amb} \le +70°C$; $4.75V \le V_{DD} \le 5.25V$

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}[2]	Dynamic current	V_{DD} = 5.25V, T_{amb} = 0°C @ 1MHz		5	mA
		V_{DD} = 5.25V, T_{amb} = 0°C @ 50MHz		75	mA
I_{OS}	Short circuit output current[3]	1 pin at a time for no longer than 1 second	–50	–200	mA
C_{IN}	Input pin capacitance[3]	T_{amb} = 25°C, f = 1MHz		8	pF
C_{CLK}	Clock input capacitance[3]	T_{amb} = 25°C, f = 1MHz	5	12	pF
$C_{I/O}$	I/O pin capacitance[3]	T_{amb} = 25°C, f = 1MHz		10	pF

NOTES:
1. See Table 2 on page 113 for typical values.
2. This parameter measured with a 16-bit, loadable up/down counter loaded into every logic block, with all outputs disabled and unloaded. Inputs are tied to V_{DD} or ground. This parameter guaranteed by design and characterization, not testing.
3. Typical values, not tested.

AC ELECTRICAL CHARACTERISTICS[1] FOR COMMERCIAL GRADE DEVICES

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

SYMBOL	PARAMETER	7		10		12		UNIT
		MIN/	MAX.	MIN.	MAX.	MIN.	MAX.	
t_{PD_PAL}	Propagation delay time, input (or feedback node) to output through PAL	2	7.5	2	10	2	12	ns
t_{PD_PLA}	Propagation delay time, input (or feedback node) to output through PAL & PLA	3	9.5	3	12	3	14.5	ns
t_{CO}	Clock to out (global synchronous clock from pin)	2	6	2	7	2	8	ns
t_{SU_PAL}	Setup time (from input or feedback node) through PAL	4.5		7		8		ns
t_{SU_PLA}	Setup time (from input or feedback node) through PAL + PLA	6.5		9		10.5		ns
t_H	Hold time		0		0		0	ns
t_{CH}	Clock High time	3		4		4		ns
t_{CL}	Clock Low time	3		4		4		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		125		MHz
f_{MAX2}	Maximum internal frequency[2] $1/(t_{SUPAL} + t_{CF})$	111		80		69		MHz
f_{MAX3}	Maximum external frequency[2] $1/(t_{SUPAL} + t_{CO})$	95		71		63		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	2	6	2	8.5	2	10.5	ns
t_{PDF_PLA}	Input (or feedback node) to internal feedback node delay time through PAL+PLA	3	8	3	10.5	3	13	ns
t_{CF}	Clock to internal feedback node delay time		4.5		5.5		6.5	ns
t_{INIT}	Delay from valid V_{DD} to valid reset		50		50		50	µs
t_{ER}	Input to output disable[2, 3]		9		12		15	ns
t_{EA}	Input to output valid[2]		9		12		15	ns
t_{RP}	Input to register preset[2]		11		12.5		15	ns
t_{RR}	Input to register reset[2]		11		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}[1]	Standby current	V_{DD} = 5.5V, T_{amb} = –40°C		125	µA
I_{DDD}[1, 2]	Dynamic current	V_{DD} = 5.5V, T_{amb} = –40°C @ 1MHz		6	mA
		V_{DD} = 5.5V, T_{amb} = –40°C @ 50MHz		90	mA
I_{OS}	Short circuit output current[3]	1 pin at a time for no longer than 1 second	–50	–230	mA
C_{IN}	Input pin capacitance[3]	T_{amb} = 25°C, f = 1MHz		8	pF
C_{CLK}	Clock input capacitance[3]	T_{amb} = 25°C, f = 1MHz	5	12	pF
$C_{I/O}$	I/O pin capacitance[3]	T_{amb} = 25°C, f = 1MHz		10	pF

NOTES:
1. See Table 2 on page 113 for typical values.
2. This parameter measured with a 16–bit, loadable up/down counter loaded into every logic block, with all outputs disabled and unloaded. Inputs are tied to V_{DD} or ground. This parameter guaranteed by design and characterization, not testing.
3. Typical values, not tested.

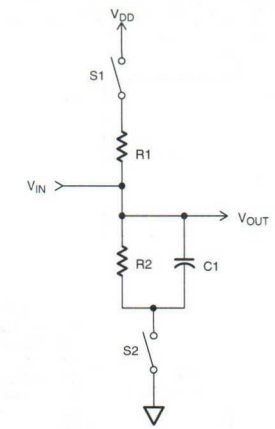
AC ELECTRICAL CHARACTERISTICS FOR INDUSTRIAL GRADE DEVICES

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

SYMBOL	PARAMETER	10		15		UNIT
		MIN.	MAX.	MIN.	MAX.	
t_{PD_PAL}	Propagation delay time, input (or feedback node) to output through PAL	2	10	2	15	ns
t_{PD_PLA}	Propagation delay time, input (or feedback node) to output through PAL & PLA	3	12	3	17.5	ns
t_{CO}	Clock to out (global synchronous clock from pin)	2	7	2	8	ns
t_{SU_PAL}	Setup time (from input or feedback node) through PAL	8		8		ns
t_{SU_PLA}	Setup time (from input or feedback node) through PAL + PLA	10		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] $1/(t_{SUPAL} + t_{CF})$	71		69		MHz
f_{MAX3}	Maximum external frequency[2] $1/(t_{SUPAL} + t_{CO})$	66		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	13.5	ns
t_{PDF_PLA}	Input (or feedback node) to internal feedback node delay time through PAL+PLA	3	10.5	3	16	ns
t_{CF}	Clock to internal feedback node delay time		6		6.5	ns
t_{INIT}	Delay from valid V_{DD} to valid reset		50		50	µs
t_{ER}	Input to output disable[2, 3]		15		15	ns
t_{EA}	Input to output valid[2]		15		15	ns
t_{RP}	Input to register preset[2]		15		17	ns
t_{RR}	Input to register reset[2]		15		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.

SWITCHING CHARACTERISTICS



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

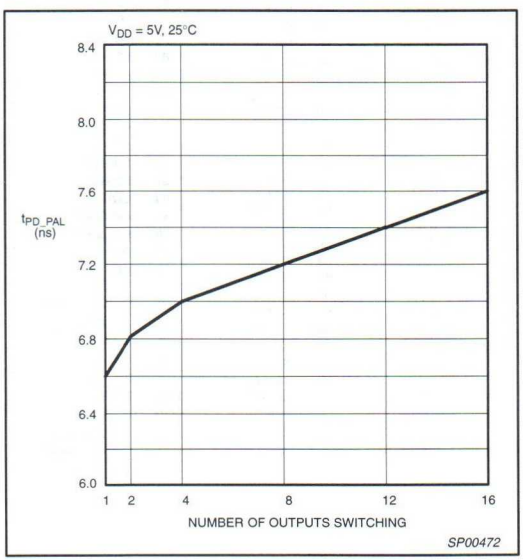
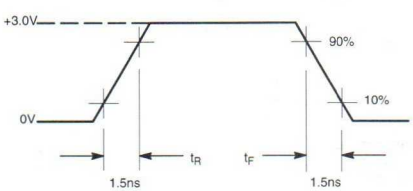


Figure 6. t_{PD_PAL} vs. Outputs Switching

VOLTAGE WAVEFORM



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

Input Pulses

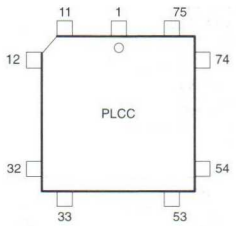
Table 8. t_{PD_PAL} vs. Number of Outputs Switching

V_{DD} = 5.00V

NUMBER OF OUTPUTS	1	2	4	8	12	16
Typical (ns)	6.6	6.8	7.0	7.2	7.4	7.6

PIN DESCRIPTIONS

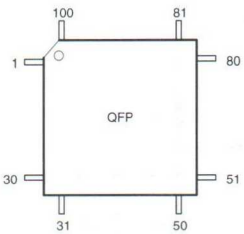
84-Pin Plastic Leaded Chip Carrier



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

SP00467

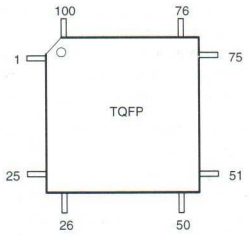
100-Pin Plastic Quad Flat Package



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-D0/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-C4	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}		

SP00468

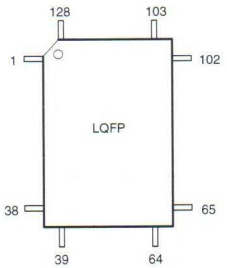
100-Pin Thin Quad Flat Package



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		

SP00485

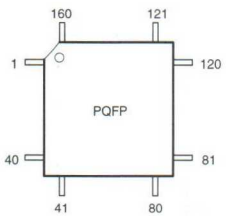
128-Pin Low Profile Quad Flat Package



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		

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

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

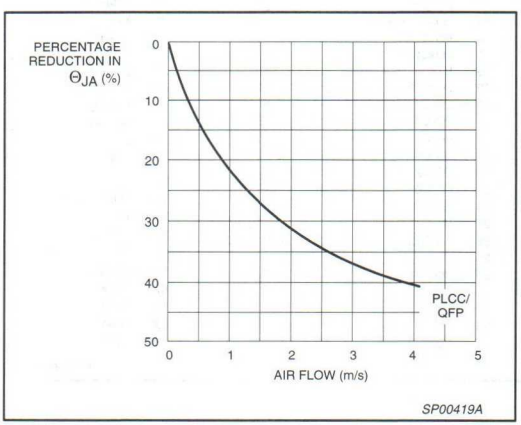


Figure 7. Average Effect of Airflow on Θ_{JA}

Section 4
XPLA Enhanced Family

CONTENTS

PZ3032C	32 macrocell CPLD with enhanced clocking	125
PZ3032A/PZ3032D	32 macrocell CPLD with enhanced clocking	137
PZ3064A/PZ3064D	64 macrocell CPLD with enhanced clocking	150
PZ3128A/PZ3128D	128 macrocell CPLD with enhanced clocking	164
PZ5032C	32 macrocell CPLD with enhanced clocking	177
PZ5064C/PZ5064N	64 macrocell CPLD with enhanced clocking	189
PZ5128C/PZ5128N	128 macrocell CPLD with enhanced clocking	203

32 macrocell CPLD with enhanced clocking PZ3032C

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
- Up to 6 clocks with programmable polarity at every macrocell
- 3.3 Volt, In-System Programmable (ISP) using a JTAG interface
  - On-chip supervoltage generation
  - ISP commands include: Enable, Erase, Program, Verify
  - Supported by multiple ISP programming platforms
  - 4 pin JTAG interface (TCK, TMS, TDI, TDO)
  - JTAG commands include: Bypass, Idcode
- 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
  - Up to 2 asynchronous clocks
- Programmable global 3-State pin facilitates 'bed of nails' testing without using logic resources
- Available in both PLCC and TQFP packages

Table 1. PZ3032C Features

	PZ3032C
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 PZ3032C CPLD (Complex Programmable Logic Device) is a member of the Fast Zero Power (FZP™) family of CPLDs from Philips Semiconductors. These devices combine high speed and zero power in a 32 macrocell CPLD. With the FZP™ design technique, the PZ3032C 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 PZ5032C 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 PZ3032C CPLDs are supported by industry standard CAE tools (Cadence, Exemplar Logic, Minc, 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 PZ3032C CPLD is reprogrammable using industry standard device programmers from vendors such as Data I/O, BP Microsystems, SMS, and others. The PZ3032C also includes an industry-standard, IEEE 1149.1, JTAG interface through which In-System Programming (ISP) and reprogramming of the device are supported.

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

ORDERING INFORMATION

ORDER CODE	DESCRIPTION	DESCRIPTION	DRAWING NUMBER
PZ3032CS8A44	44-pin PLCC, 8ns t_{PD}	Commercial temp range, 3.3 volt power supply, ± 10%	SOT187-2
PZ3032CS10A44	44-pin PLCC, 10ns t_{PD}	Commercial temp range, 3.3 volt power supply, ± 10%	SOT187-2
PZ3032CS12A44	44-pin PLCC, 12ns t_{PD}	Commercial temp range, 3.3 volt power supply, ± 10%	SOT187-2
PZ3032CS8BC	44-pin TQFP, 8ns t_{PD},	Commercial temp range, 3.3 volt power supply, ± 10%	SOT376-1
PZ3032CS10BC	44-pin TQFP, 10ns t_{PD}	Commercial temp range, 3.3 volt power supply, ± 10%	SOT376-1
PZ3032CS12BC	44-pin TQFP, 12ns t_{PD}	Commercial temp range, 3.3 volt power supply, ± 10%	SOT376-1

XPLA™ ARCHITECTURE

Figure 1 shows a high level block diagram of a 32 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. In addition, two of the control terms can be used as clock signals (see Macrocell Architecture section for details). 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 PZ3032C 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 PZ3032C using 6 to 37 product terms is 10.5ns (8ns for the PAL + 2.5ns for the PLA).

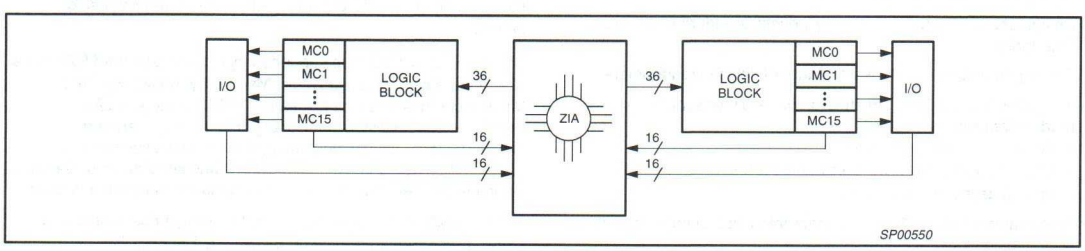


Figure 1. Philips XPLA CPLD Architecture

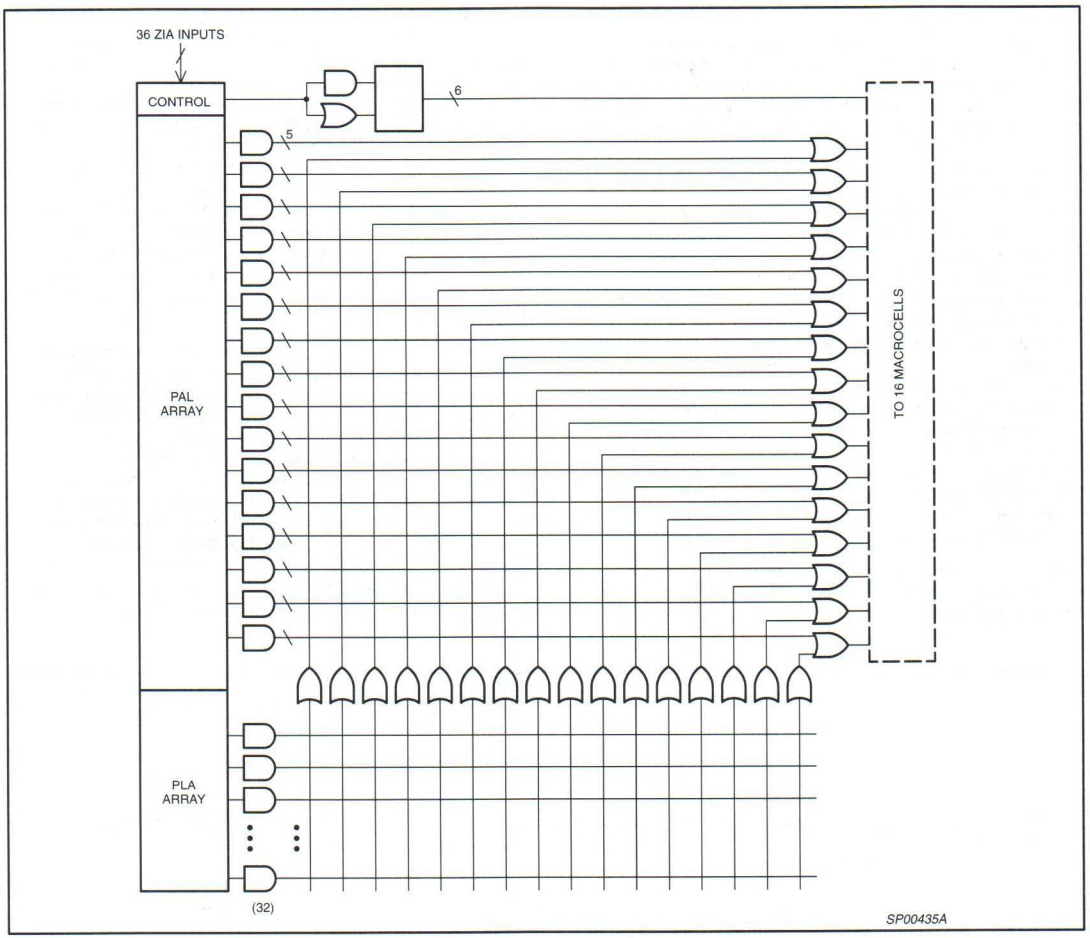


Figure 2. Philips XPLA Logic Block Architecture

Macrocell Architecture

Figure 3 shows the architecture of the macrocell used in the CoolRunner™ PZ3032C. The macrocell can be configured as either a D or T type flip-flop or a combinatorial logic function. A D-type flip-flop is generally more useful for implementing state machines and data buffering while a T-type flip-flop is generally more useful in implementing counters. Each of these flip-flops can be clocked from any one of four sources. Two of the clock sources (CLK0 and CLK1) are connected to low-skew, device-wide clock networks designed to preserve the integrity of the clock signal by reducing skew between rising and falling edges. Clock 0 (CLK0) is designated as a "synchronous" clock and must be driven by an external source. Clock 1 (CLK1) can be used as a "synchronous" clock that is driven by an external source, or as an "asynchronous" clock that is driven by a macrocell equation. Both CLK0 and CLK1 can clock the macrocell flip-flops on either the rising edge or the falling edge of the clock signal. The other clock sources are two of the six control terms (CT2 and CT3) provided in each logic block. These clocks can be individually configured as either a PRODUCT term or SUM term equation created from the 36 signals available inside the logic block. The timing for asynchronous and control term clocks is different in that the Tco time is extended by the amount of time that it takes for the signal to propagate through the array and reach the clock network, and the Tsu time is reduced. Please see the app note titled "Understanding CoolRunner Clocking Options" for more detail.

The six control terms of each logic block are used to control the asynchronous Preset/Reset of the flip-flops and the enable/disable of the output buffers in each macrocell. Control terms CT0 and CT1 are used to control the asynchronous Preset/Reset of the macrocell's flip-flop. Note that the Power-on Reset leaves all macrocells in the "zero" state when power is properly applied, and that the Preset/Reset feature for each macrocell can also be disabled. Control terms CT2 and CT3 can be used as a clock signal to the flip-flops of the macrocells, and as the Output Enable of the macrocell's output buffer. Control terms CT4 and CT5 can be used to control the Output Enable of the macrocell's output buffer. Having four dedicated Output Enable control terms ensures that the CoolRunner™ devices are PCI compliant. The output buffers can also be always enabled or always disabled. All CoolRunner™ devices also provide a Global Tri-State (GTS) pin, which, when enabled and 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 feedback 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 (See the section on terminations in this data sheet and the app note *Terminating Unused CoolRunner™ I/O Pins*).

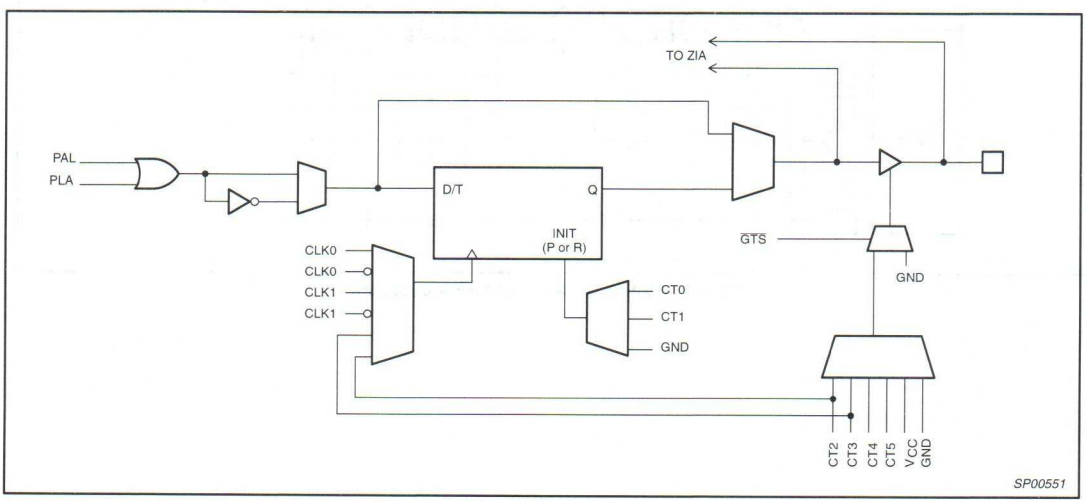


Figure 3. PZ3032C Macrocell Architecture

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 PZ3032C 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 PZ3032C TotalCMOS™ CPLD.

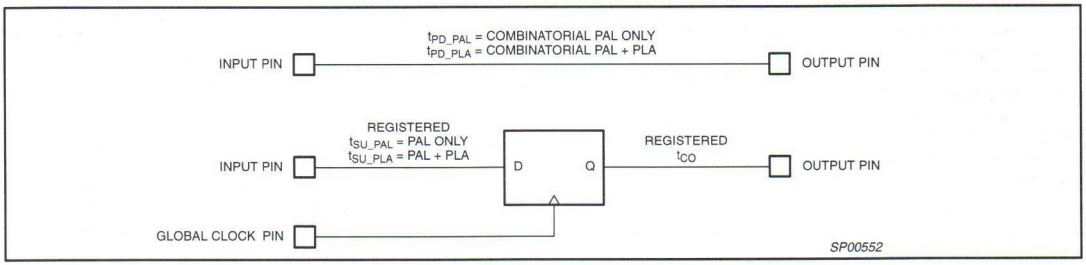


Figure 4. CoolRunner™ Timing Model

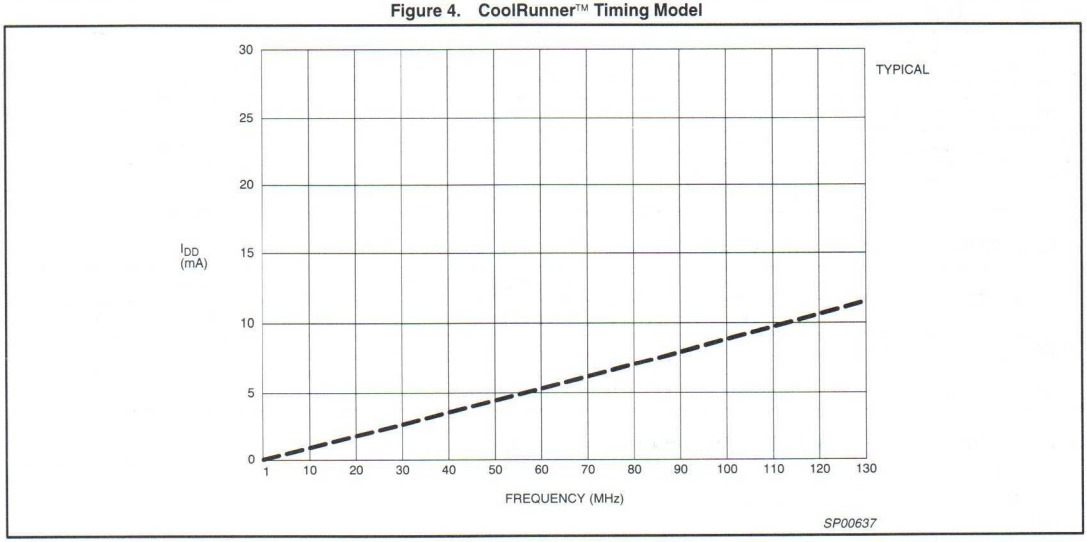


Figure 5. I_{DD} vs. Frequency @ V_{DD} = 3.3V

Table 2. I_{DD} vs Frequency

V_{DD} = 3.3V

FREQ (MHz)	0	1	10	20	30	40	50	60	70	80	90	100	110	120	130
Typical I_{DD} (mA)	0.01	0.10	0.89	1.77	2.63	3.55	4.44	5.3	6.19	7.08	7.96	8.83	9.7	10.6	11.5

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. The Philips PZ3032C devices use the JTAG interface for In–System Programming/Reprogramming. Although only a subset of the full JTAG command set is implemented (see Table 5), the devices are fully capable of sitting in a JTAG scan chain.

The Philips PZ3032C's JTAG interface includes a TAP Port defined by the IEEE 1149.1 JTAG Specification. As implemented in the Philips PZ3032C, 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 PZ3032C saves an I/O pin for general purpose use by not implementing the optional TRST* signal in the JTAG interface. Instead, the Philips PZ3032C 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 TAP Port should connect to an external pull–up resistor to keep the JTAG pins from floating when they are not being used (see section on Terminations).

In the Philips PZ3032C, the four mandatory JTAG pins each require a unique, dedicated pin on the device. The devices come from the factory with these I/O pins set to perform JTAG functions, but through the software, the final function of these pins can be controlled. If the end application will require the device to be reprogrammed at some future time with ISP, then the pins can be left as dedicated JTAG functions, which means they are not available for use as general purpose I/O pins. However, unlike competing CPLDs, the Philips PZ3032C allow the macrocells associated with these pins to be used as buried logic when the JTAG/ISP function is enabled. This is the default state for the software, and no action is required to leave these pins enabled for the JTAG/ISP functions. If, however, JTAG/ISP is not required to leave these pins enabled for the JTAG/ISP functions. If, however, JTAG/ISP is not required in the end application, the software can specify that this function be turned off and that these pins be used as general purpose I/O. Because the devices initially have the JTAG/ISP functions enabled, the JEDEC file can be downloaded into the device once, after which the JTAG/ISP pins will become general purpose I/O. This feature is good for manufacturing because the devices can be programmed during test and assembly of the end product and yet still use all of the I/O pins after the programming is done. It eliminates the need for a costly, separate programming step in the manufacturing process. Of course, if the JTAG/ISP function is never required, this feature can be turned off in the software and the device can be programmed with an industry-standard programmer, leaving the pins available for I/O functions. Table 4 defines the dedicated pins used by the four mandatory JTAG signals for each of the PZ3032C package types.

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.
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. PZ3032C JTAG Pinout by Package Type

DEVICE	(PIN NUMBER / MACROCELL #)			
	TCK	TMS	TDI	TDO
PZ3032C				
44-pin PLCC	32/B8	13/A8	7/A3	38/B3
44-pin TQFP	26/B8	7/A8	1/A3	32/B3

Table 5. PZ3032C Low-Level JTAG Boundary-Scan Commands

INSTRUCTION (Instruction Code) *Register Used*	DESCRIPTION
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.

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
  - Reconfigurability 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 PZ3032C 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 PZ3032C 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 PZ3032C enable this feature. The ISP commands implemented in the Philips PZ3032C 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.

Terminations

The CoolRunner™ PZ3032C CPLDs are TotalCMOS™ devices. As with other CMOS devices, it is important to consider how to properly terminate unused inputs and I/O pins when fabricating a PC board. The PZ3032C devices do not have on-chip termination circuits, so it is recommended that unused inputs and I/O pins be properly terminated. Allowing unused inputs and I/O pins to float can cause the voltage to be in the linear region of the CMOS input structures, which can increase the power consumption of the device. Philips recommends the use of 10KΩ pull-up resistors for the termination. Using pull-up resistors allows 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.

When using the JTAG/ISP functions, it is also recommended that 10KΩ pull-up resistors be used on each of the four mandatory signals. Letting these signals float can cause the voltage on TMS to come close to ground, which could cause the device to enter JTAG/ISP mode at unspecified times. See the application notes *JTAG and ISP in Philips Devices* and *Terminating Unused CoolRunner™ I/O Pins* for more information.

Table 6. Low Level ISP Commands

INSTRUCTION *(Register Used)*	INSTRUCTION CODE	DESCRIPTION
Enable *(ISP Shift Register)*	1001	Enables the Erase, Program, and Verify commands.
Erase *(ISP Shift Register)*	1010	Erases the entire EEPROM array.
Program *(ISP Shift Register)*	1011	Programs the data in the ISP Shift Register into the addressed EEPROM row.
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.

JTAG and ISP Interfacing

A number of industry-established methods exist for JTAG/ISP interfacing with CPLD's and other integrated circuits. The Philips PZ3032C supports the following methods:

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

For more details on JTAG and ISP for the PZ3032C, 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}	TCK 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[2]	–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.
2. The chip supply voltage must rise monotonically.

OPERATING RANGE

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

DC ELECTRICAL CHARACTERISTICS FOR COMMERCIAL GRADE DEVICES

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

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[3]	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}[1]	Standby current	V_{DD} = 3.6V, T_{amb} = 0°C		35	µA
I_{DDD}[1, 2]	Dynamic current	V_{DD} = 3.6V, T_{amb} = 0°C @ 1MHz		0.3	mA
		V_{DD} = 3.6V, T_{amb} = 0°C @ 50MHz		10	mA
I_{OS}	Short circuit output current[3]	1 pin at a time for no longer than 1 second	–5	–100	mA
C_{IN}	Input pin capacitance[3]	T_{amb} = 25°C, f = 1MHz		8	pF
C_{CLK}	Clock input capacitance[3]	T_{amb} = 25°C, f = 1MHz	5	12	pF
$C_{I/O}$	I/O pin capacitance[3]	T_{amb} = 25°C, f = 1MHz		10	pF

NOTES:
1. See Table 2, page 129 for typical values.
2. This parameter measured with a 16–bit, loadable up/down counter loaded into every logic block, with all outputs disabled and unloaded. Inputs are tied to V_{DD} or ground. This parameter guaranteed by design and characterization, not testing.
3. This parameter guaranteed by design and characterization, not by test.

AC ELECTRICAL CHARACTERISTICS[1] FOR COMMERCIAL GRADE DEVICES

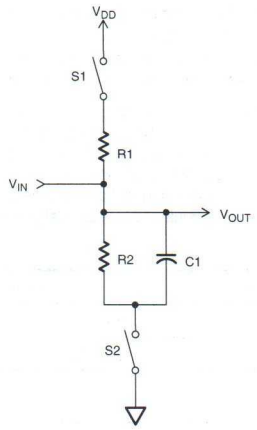
Commercial: $0°C \le T_{amb} \le +70°C$; $3.0V \le V_{DD} \le 3.6V$

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	3	13	3	15	ns
t_{CO}	Clock to out (global synchronous clock from pin)	2	6	2	9	2	11	ns
t_{SU_PAL}	Setup time (from input or feedback node) through PAL	5		8.5		10.5		ns
t_{SU_PLA}	Setup time (from input or feedback node) through PAL + PLA	7		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] $(1/t_{SUPAL} + t_{CF})$	95		63		50		MHz
f_{MAX3}	Maximum external frequency[2] $(1/t_{SUPAL} + t_{CO})$	91		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[2, 3]		11		17		19	ns
t_{EA}	Input to output valid[2]		11		17		19	ns
t_{RP}	Input to register preset[2]		14		18		20	ns
t_{RR}	Input to register reset[2]		14		21		23	ns

NOTES:
1. Specifications measured with one output switching. See Figure 6 and Table 7 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

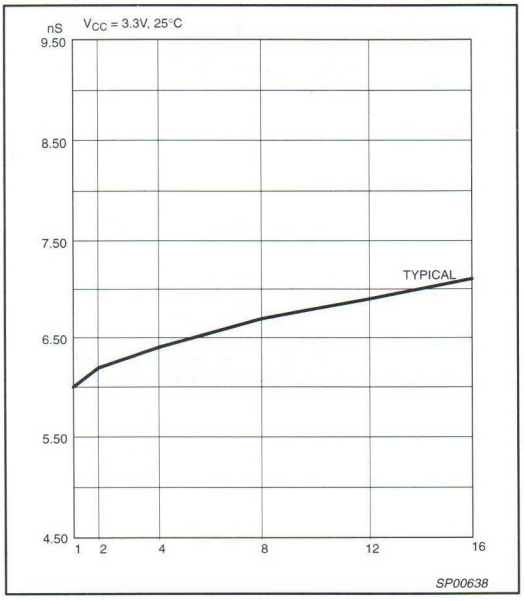
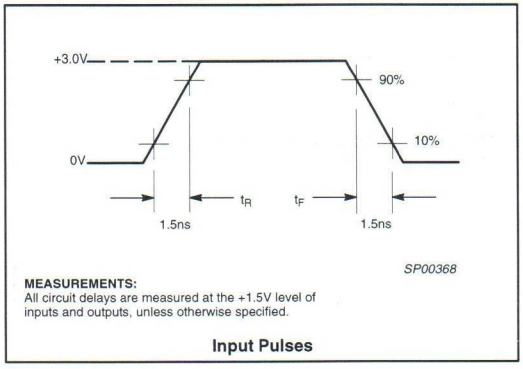


Figure 6. t_{PD_PAL} vs. Outputs switching

VOLTAGE WAVEFORM



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

Input Pulses

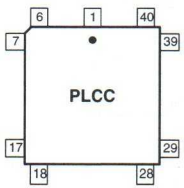
Table 7. t_{PD_PAL} vs. # of Outputs switching

V_{DD} = 3.30V, T = 25°C

# of Outputs	1	2	4	8	12	16
Typical (ns)	6.0	6.2	6.4	6.7	6.9	7.1

PIN DESCRIPTIONS

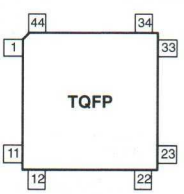
PZ3032C – 44-Pin Plastic Leaded Chip Carrier



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 (TCK)
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 (TDI)	22	GND	37	I/O–B4
8	I/O–A4	23	V_{DD}	38	I/O–B3 (TDO)
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 (TMS)	28	I/O–B11	43	IN0–CK0
14	I/O–A9	29	I/O–B10	44	IN2–gtsn
15	V_{DD}	30	GND		

SP00546

PZ3032C – 44-Pin Thin Quad Flat Package



Pin	Function	Pin	Function	Pin	Function
1	I/O–A3 (TDI)	16	GND	31	I/O–B4
2	I/O–A4	17	V_{DD}	32	I/O–B3 (TDO)
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 (TMS)	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 (TCK)	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		

SP00547

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

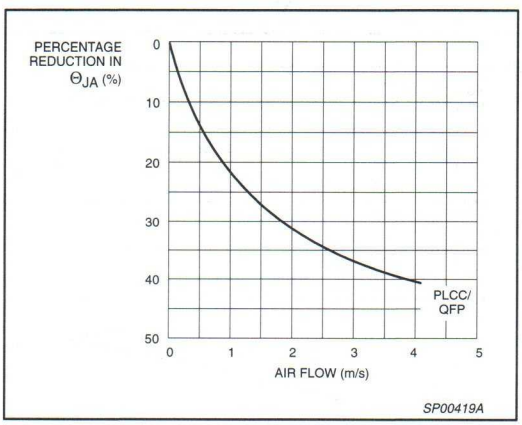


Figure 7. Average Effect of Airflow on Q_{JA}

32 macrocell CPLD with enhanced clocking PZ3032A/PZ3032D

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
- 5 V tolerant I/Os to support mixed voltage systems
- 100% routable with 100% utilization while all pins and all macrocells are fixed
- Deterministic timing model that is extremely simple to use
- Up to 6 clocks with programmable polarity at every macrocell
- 3.3 Volt, In-System Programmable (ISP) using a JTAG interface
  - On-chip supervoltage generation
  - ISP commands include: Enable, Erase, Program, Verify
  - Supported by multiple ISP programming platforms
  - 4 pin JTAG interface (TCK, TMS, TDI, TDO)
  - JTAG commands include: Bypass, Idcode
- 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 0.35µ 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
  - Up to 2 asynchronous clocks
- Programmable global 3-State pin facilitates 'bed of nails' testing without using logic resources
- Available in both PLCC and TQFP packages

Table 1. PZ3032A/PZ3032D Features

	PZ3032A/PZ3032D
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 PZ3032A/PZ3032D CPLD (Complex Programmable Logic Device) is a member of the Fast Zero Power (FZP™) family of CPLDs from Philips Semiconductors. These devices combine high speed and zero power in a 32 macrocell CPLD. With the FZP™ design technique, the PZ3032A/PZ3032D 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 5V applications, Philips also offers the high speed PZ3032A/PZ3032D 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 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

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

multiple outputs via the OR array, effectively increasing design density.

The PZ3032A/PZ3032D CPLDs are supported by industry standard CAE tools (Cadence, Exemplar Logic, Minc, 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 PZ3032A/PZ3032D CPLD is reprogrammable using industry standard device programmers from vendors such as Data I/O, BP Microsystems, SMS, and others. The PZ3032A/PZ3032D also includes an industry-standard, IEEE 1149.1, JTAG interface through which In-System Programming (ISP) and reprogramming of the device are supported.

ORDERING INFORMATION

ORDER CODE	DESCRIPTION	DESCRIPTION	DRAWING NUMBER
PZ3032AS6A44	44-pin PLCC, 6ns t_{PD}	Commercial temp range, 3.3 volt power supply, ±10%	SOT187-2
PZ3032AS7A44	44-pin PLCC, 7.5ns t_{PD}	Commercial temp range, 3.3 volt power supply, ±10%	SOT187-2
PZ3032AS10A44	44-pin PLCC, 10ns t_{PD}	Commercial temp range, 3.3 volt power supply, ±10%	SOT187-2
PZ3032AS6BC	44-pin TQFP, 6ns t_{PD},	Commercial temp range, 3.3 volt power supply, ±10%	SOT376-1
PZ3032AS7BC	44-pin TQFP, 7.5ns t_{PD}	Commercial temp range, 3.3 volt power supply, ±10%	SOT376-1
PZ3032AS10BC	44-pin TQFP, 10ns t_{PD}	Commercial temp range, 3.3 volt power supply, ±10%	SOT376-1
PZ3032DS7A44	44-pin PLCC, 7.5ns t_{PD}	Industrial temp range, 3.3 volt power supply, ±10%	SOT187-2
PZ3032DS10A44	44-pin PLCC, 10ns t_{PD}	Industrial temp range, 3.3 volt power supply, ±10%	SOT187-2
PZ3032DS7BC	44-pin TQFP, 7.5ns t_{PD}	Industrial temp range, 3.3 volt power supply, ±10%	SOT376-1
PZ3032DS10BC	44-pin TQFP, 10ns 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 32 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. In addition, two of the control terms can be used as clock signals (see Macrocell Architecture section for details). 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 PZ3032A/PZ3032D 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 PZ3032A/PZ3032D using 6 to 37 product terms is 8ns (6ns for the PAL + 2ns for the PLA).

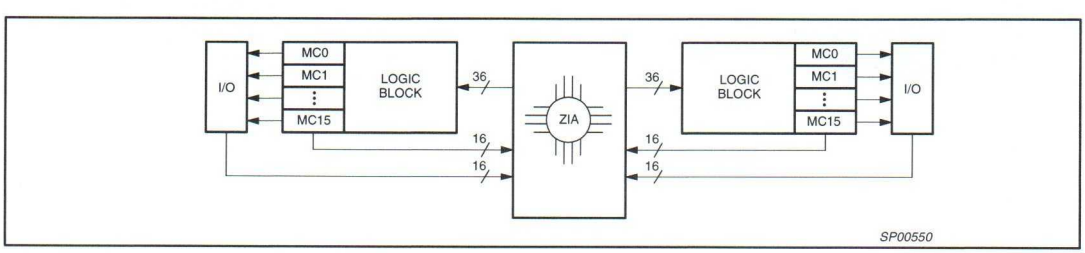


Figure 1. Philips XPLA CPLD Architecture

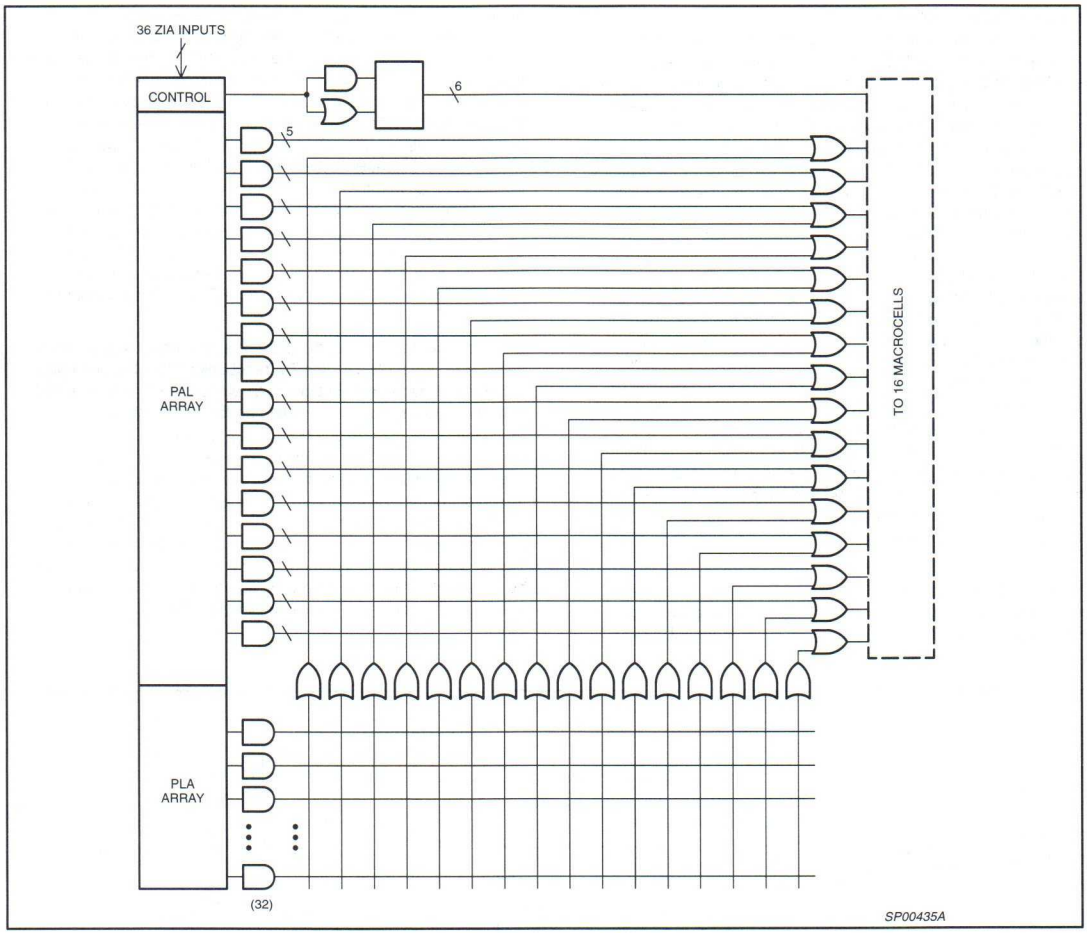


Figure 2. Philips XPLA Logic Block Architecture

Macrocell Architecture

Figure 3 shows the architecture of the macrocell used in the CoolRunner™ PZ3032A/PZ3032D. The macrocell can be configured as either a D or T type flip-flop or a combinatorial logic function. A D-type flip-flop is generally more useful for implementing state machines and data buffering while a T-type flip-flop is generally more useful in implementing counters. Each of these flip-flops can be clocked from any one of four sources. Two of the clock sources (CLK0 and CLK1) are connected to low-skew, device-wide clock networks designed to preserve the integrity of the clock signal by reducing skew between rising and falling edges. Clock 0 (CLK0) is designated as a "synchronous" clock and must be driven by an external source. Clock 1 (CLK1) can be used as a "synchronous" clock that is driven by an external source, or as an "asynchronous" clock that is driven by a macrocell equation. Both CLK0 and CLK1 can clock the macrocell flip-flops on either the rising edge or the falling edge of the clock signal. The other clock sources are two of the six control terms (CT2 and CT3) provided in each logic block. These clocks can be individually configured as either a PRODUCT term or SUM term equation created from the 36 signals available inside the logic block. The timing for asynchronous and control term clocks is different in that the Tco time is extended by the amount of time that it takes for the signal to propagate through the array and reach the clock network, and the Tsu time is reduced. Please see the app note titled "Understanding CoolRunner Clocking Options" for more detail.

The six control terms of each logic block are used to control the asynchronous Preset/Reset of the flip-flops and the enable/disable of the output buffers in each macrocell. Control terms CT0 and CT1 are used to control the asynchronous Preset/Reset of the macrocell's flip-flop. Note that the Power-on Reset leaves all macrocells in the "zero" state when power is properly applied, and that the Preset/Reset feature for each macrocell can also be disabled. Control terms CT2 and CT3 can be used as a clock signal to the flip-flops of the macrocells, and as the Output Enable of the macrocell's output buffer. Control terms CT4 and CT5 can be used to control the Output Enable of the macrocell's output buffer. Having four dedicated Output Enable control terms ensures that the CoolRunner™ devices are PCI compliant. The output buffers can also be always enabled or always disabled. All CoolRunner™ devices also provide a Global Tri-State (GTS) pin, which, when enabled and 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 feedback 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 (See the section on terminations in this data sheet and the app note *Terminating Unused CoolRunner™ I/O Pins*).

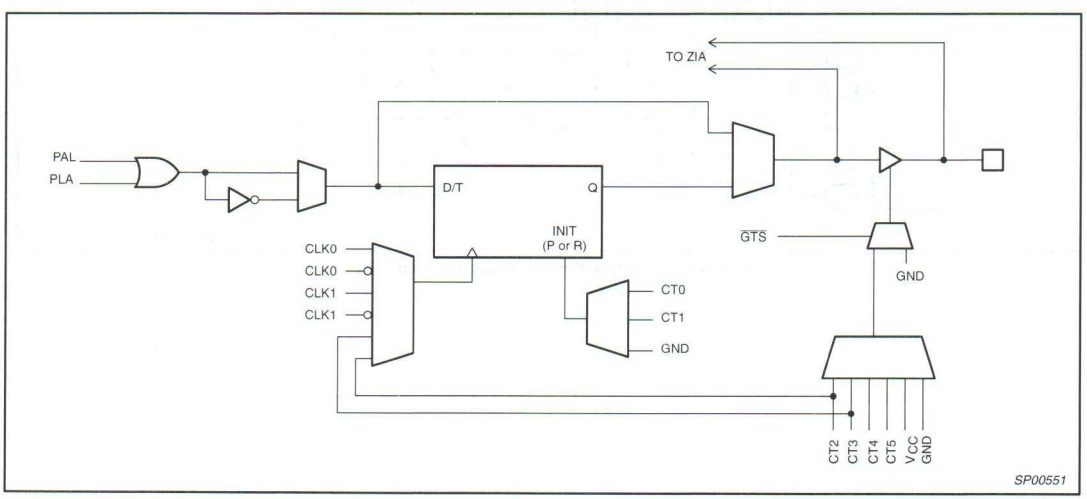


Figure 3. PZ3032A/PZ3032D Macrocell Architecture

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 PZ3032A/PZ3032D device, the user knows up front that if a given output uses 5 product terms or less, the t_{PD} = 6ns, the t_{SU} = 3.5ns, 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 PZ3032A/PZ3032D TotalCMOS™ CPLD.

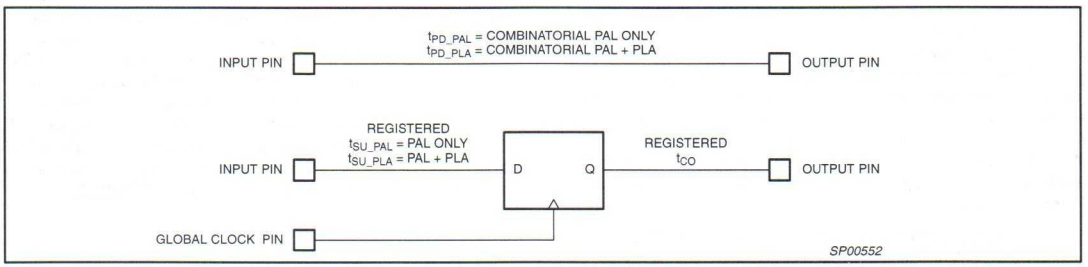


Figure 4. CoolRunner™ Timing Model

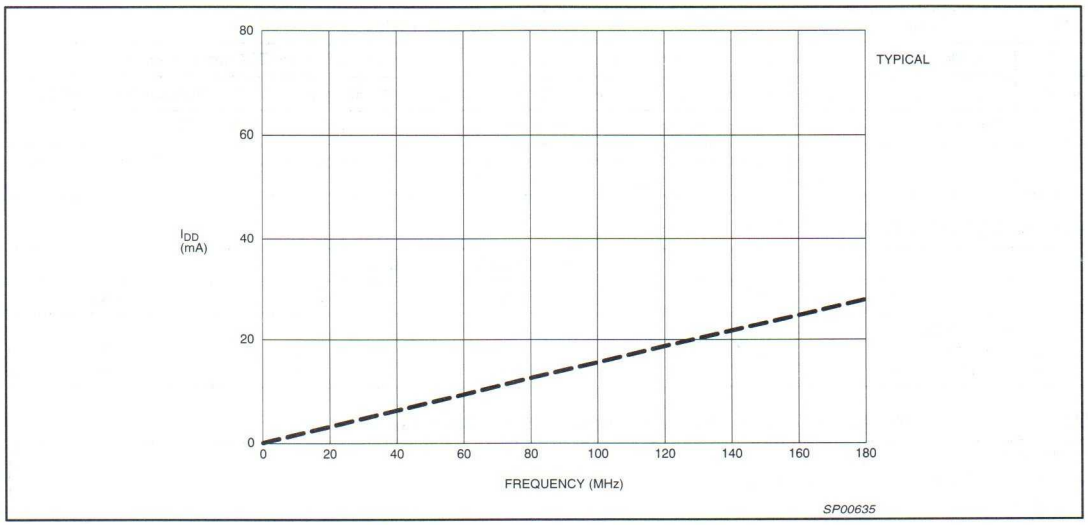


Figure 5. I_{DD} vs. Frequency @ V_{DD} = 3.3 V

Table 2. I_{DD} vs Frequency

V_{DD} = 3.3 V

FREQ (MHz)	0	1	20	40	60	80	100	120	140	160	180
Typical I_{DD} (mA)											

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. The Philips PZ3032A/PZ3032D devices use the JTAG interface for In–System Programming/Reprogramming. Although only a subset of the full JTAG command set is implemented (see Table 5), the devices are fully capable of sitting in a JTAG scan chain.

The Philips PZ3032A/PZ3032D's JTAG interface includes a TAP Port defined by the IEEE 1149.1 JTAG Specification. As implemented in the Philips PZ3032A/PZ3032D, 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 PZ3032A/PZ3032D saves an I/O pin for general purpose use by not implementing the optional TRST* signal in the JTAG interface. Instead, the Philips PZ3032A/PZ3032D 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 TAP Port should connect to an external pull–up resistor to keep the JTAG pins from floating when they are not being used (see section on Terminations).

In the Philips PZ3032A/PZ3032D, the four mandatory JTAG pins each require a unique, dedicated pin on the device. The devices come from the factory with these I/O pins set to perform JTAG functions, but through the software, the final function of these pins can be controlled. If the end application will require the device to be reprogrammed at some future time with ISP, then the pins can be left as dedicated JTAG functions, which means they are not available for use as general purpose I/O pins. However, unlike competing CPLDs, the Philips PZ3032A/PZ3032D allow the macrocells associated with these pins to be used as buried logic when the JTAG/ISP function is enabled. This is the default state for the software, and no action is required to leave these pins enabled for the JTAG/ISP functions. If, however, JTAG/ISP is not required in the end application, the software can specify that this function be turned off and that these pins be used as general purpose I/O. Because the devices initially have the JTAG/ISP functions enabled, the JEDEC file can be down loaded into the device once, after which the JTAG/ISP pins will become general purpose I/O. This feature is good for manufacturing because the devices can be programmed during test and assembly of the end product and yet still use all of the I/O pins after the programming is done. It eliminates the need for a costly, separate programming step in the manufacturing process. Of course, if the JTAG/ISP function is never required, this feature can be turned off in the software and the device can be programmed with an industry-standard programmer, leaving the pins available for I/O functions. Table 4 defines the dedicated pins used by the four mandatory JTAG signals for each of the PZ3032A/PZ3032D package types.

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.
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. PZ3032A/PZ3032D JTAG Pinout by Package Type

DEVICE	(PIN NUMBER / MACROCELL #)			
	TCK	TMS	TDI	TDO
PZ3032A/PZ3032D				
44-pin PLCC	32/B8	13/A8	7/A3	38/B3
44-pin TQFP	26/B8	7/A8	1/A3	32/B3

Table 5. PZ3032A/PZ3032D Low-Level JTAG Boundary-Scan Commands

INSTRUCTION (Instruction Code) *Register Used*	DESCRIPTION
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.

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
  - Reconfigurability 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 PZ3032A/PZ3032D 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 PZ3032A/PZ3032D 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 PZ3032A/PZ3032D enable this feature. The ISP commands implemented in the Philips PZ3032A/PZ3032D 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.

Terminations

The CoolRunner™ PZ3032A/PZ3032D CPLDs are TotalCMOS™ devices. As with other CMOS devices, it is important to consider how to properly terminate unused inputs and I/O pins when fabricating a PC board. The PZ3032A/PZ3032D devices do not have on-chip termination circuits, so it is recommended that unused inputs and I/O pins be properly terminated. Allowing unused inputs and I/O pins to float can cause the voltage to be in the linear region of the CMOS input structures, which can increase the power consumption of the device. Philips recommends the use of 10KΩ pull-up resistors for the termination. Using pull-up resistors allows 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.

When using the JTAG/ISP functions, it is also recommended that 10KΩ pull-up resistors be used on each of the four mandatory signals. Letting these signals float can cause the voltage on TMS to come close to ground, which could cause the device to enter JTAG/ISP mode at unspecified times. See the application notes *JTAG and ISP in Philips Devices* and *Terminating Unused CoolRunner™ I/O Pins* for more information.

Table 6. Low Level ISP Commands

INSTRUCTION *(Register Used)*	INSTRUCTION CODE	DESCRIPTION
Enable *(ISP Shift Register)*	1001	Enables the Erase, Program, and Verify commands.
Erase *(ISP Shift Register)*	1010	Erases the entire EEPROM array.
Program *(ISP Shift Register)*	1011	Programs the data in the ISP Shift Register into the addressed EEPROM row.
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.

JTAG and ISP Interfacing

A number of industry-established methods exist for JTAG/ISP interfacing with CPLD's and other integrated circuits. The Philips PZ3032A/PZ3032D supports the following methods:

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

For more details on JTAG and ISP for the PZ3032A/PZ3032D, 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}	TCK 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[2]	–0.5	4.6	V
V_I	Input voltage	–0.5	5.5	V
V_{OUT}	Output voltage	–0.5	5.5	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.
2. The chip supply voltage must rise monotonically.

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: $0°C \le T_{amb} \le +70°C$; $3.0V \le V_{DD} \le 3.6V$

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 5.25 V	–10	10	µA
I_{OZ}	3-Stated output leakage current	V_{IN} = 0 to 5.25 V	–10	10	µA
I_{DDQ}[1]	Standby current	V_{DD} = 3.6V, T_{amb} = 0°C			µA
I_{DDD}[1, 2]	Dynamic current	V_{DD} = 3.6V, T_{amb} = 0°C @ 1MHz			mA
		V_{DD} = 3.6V, T_{amb} = 0°C @ 50MHz			mA
I_{OS}	Short circuit output current[3]	1 pin at a time for no longer than 1 second	–50	–100	mA
C_{IN}	Input pin capacitance[3]	T_{amb} = 25°C, f = 1MHz		8	pF
C_{CLK}	Clock input capacitance[3]	T_{amb} = 25°C, f = 1MHz	5	12	pF
$C_{I/O}$	I/O pin capacitance[3]	T_{amb} = 25°C, f = 1MHz		10	pF

NOTES:
1. See Table 2 on page 141 for typical values.
2. This parameter measured with a 16-bit, loadable up/down counter loaded into every logic block, with all outputs disabled and unloaded. Inputs are tied to V_{DD} or ground. This parameter guaranteed by design and characterization, not testing.
3. Typical values, not tested.

AC ELECTRICAL CHARACTERISTICS[1] FOR COMMERCIAL GRADE DEVICES

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

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 (global synchronous clock from pin)	2	5.5	2	7	2	9	ns
t_{SU_PAL}	Setup time (from input or feedback node) through PAL	3.5		5.5		8		ns
t_{SU_PLA}	Setup time (from input or feedback node) through PAL + PLA	5.5		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] $(1/t_{SUPAL} + t_{CF})$	133		91		64		MHz
f_{MAX3}	Maximum external frequency[2] $(1/t_{SUPAL} + t_{CO})$	111		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[2, 3]		11		12.5		15	ns
t_{EA}	Input to output valid[2]		11		12.5		15	ns
t_{RP}	Input to register preset[2]		11		12.5		15	ns
t_{RR}	Input to register reset[2]		14		15.5		18	ns

NOTES:
1. Specifications measured with one output switching. See Figure 6 and Table 7 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} = 12mA		0.5	V
V_{OH}	Output voltage high	V_{DD} = 3.0V, I_{OH} = –12mA	2.4		V
I_I	Input leakage current	V_{IN} = 0 to 5.5 V	–10	10	µA
I_{OZ}	3-Stated output leakage current	V_{IN} = 0 to 5.5 V	–10	10	µA
I_{DDQ}[1]	Standby current	V_{DD} = 3.6V, T_{amb} = –40°C			µA
I_{DDD}[1, 2]	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[3]	1 pin at a time for no longer than 1 second	–50	–130	mA
C_{IN}	Input pin capacitance[3]	T_{amb} = 25°C, f = 1MHz		8	pF
C_{CLK}	Clock input capacitance[3]	T_{amb} = 25°C, f = 1MHz	5	12	pF
$C_{I/O}$	I/O pin capacitance[3]	T_{amb} = 25°C, f = 1MHz		10	pF

NOTES:
1. See Table 2 on page 141 for typical values.
2. This parameter measured with a 16-bit, loadable up/down counter loaded into every logic block, with all outputs disabled and unloaded. Inputs are tied to V_{DD} or ground. This parameter guaranteed by design and characterization, not testing.
3. Typical values, not tested.

AC ELECTRICAL CHARACTERISTICS[1] FOR INDUSTRIAL GRADE DEVICES

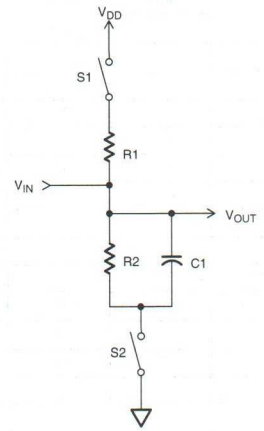
Industrial: $-40°C \le T_{amb} \le +85°C$; $3.0\ V \le V_{DD} \le 3.6\ V$

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 (global synchronous clock from pin)	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	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.

SP00650

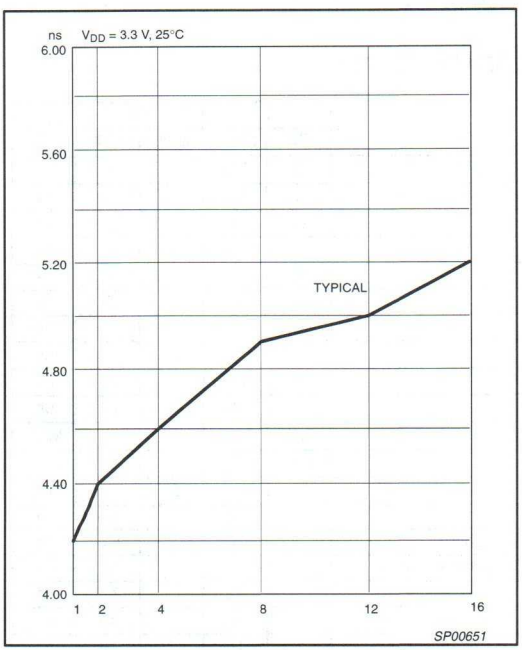
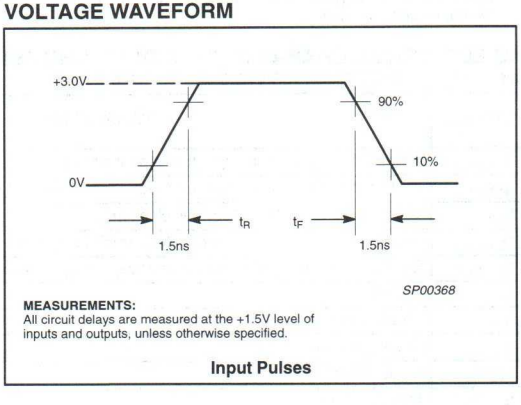


Figure 6. t_{PD_PAL} vs. Outputs switching

VOLTAGE WAVEFORM



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

Input Pulses

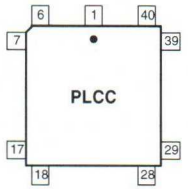
Table 7. t_{PD_PAL} vs # of Outputs switching

V_{DD} = 3.3 V

# of Outputs	1	2	4	8	12	16
Typical (ns)	4.2	4.4	4.6	4.9	5.0	5.2

PIN DESCRIPTIONS

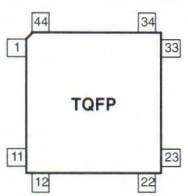
PZ3032A/PZ3032D – 44-Pin Plastic Leaded Chip Carrier



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 (TCK)
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 (TDI)	22	GND	37	I/O–B4
8	I/O–A4	23	V_{DD}	38	I/O–B3 (TDO)
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 (TMS)	28	I/O–B11	43	IN0–CK0
14	I/O–A9	29	I/O–B10	44	IN2–gtsn
15	V_{DD}	30	GND		

SP00546

PZ3032A/PZ3032D – 44-Pin Thin Quad Flat Package



Pin	Function	Pin	Function	Pin	Function
1	I/O–A3 (TDI)	16	GND	31	I/O–B4
2	I/O–A4	17	V_{DD}	32	I/O–B3 (TDO)
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 (TMS)	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 (TCK)	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		

SP00547

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

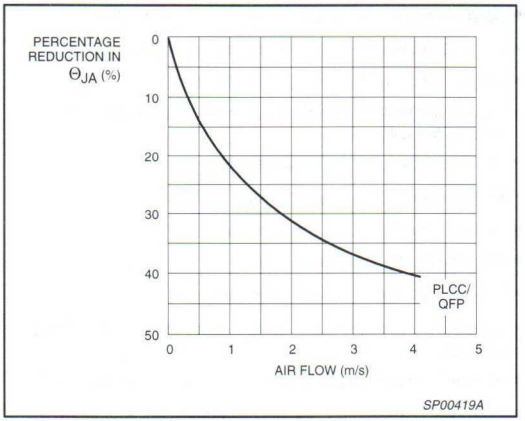


Figure 7. Average Effect of Airflow on Θ_{JA}

64 macrocell CPLD with enhanced clocking PZ3064A/PZ3064D

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
- 3 Volt, In-System Programmable (ISP) using a JTAG interface
  - On-chip supervoltage generation
  - ISP commands include: Enable, Erase, Program, Verify
  - Supported by multiple ISP programming platforms
  - 4 pin JTAG interface (TCK, TMS, TDI, TDO)
  - JTAG commands include: Bypass, Idcode
- 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
- 5 V tolerant I/Os to support mixed voltage systems
- 100% routable with 100% utilization while all pins and all macrocells are fixed
- Deterministic timing model that is extremely simple to use
- Up to 12 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
- Advanced 0.35µ 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
  - Up to 2 asynchronous clocks
- Programmable global 3-State pin facilitates 'bed of nails' testing without using logic resources
- Available in PLCC and TQFP packages
- Available in both Commercial and Industrial grades

Table 1. PZ3064A/PZ3064D Features

	PZ3064A/PZ3064D
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, 100-pin TQFP

DESCRIPTION

The PZ3064A/PZ3064D 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 PZ3064A/PZ3064D 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.

The Philips FZP™ CPLDs introduce the new patented 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 PZ3064A/PZ3064D CPLDs are supported by industry standard CAE tools (Cadence, Exemplar Logic, Mentor, OrCAD, 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 PZ3064A/PZ3064D CPLD is reprogrammable using industry standard device programmers from vendors such as Data I/O, BP Microsystems, SMS, and others. The PZ3064A/PZ3064D also includes an industry–standard, IEEE 1149.1, JTAG interface through which In–System Programming (ISP) and reprogramming of the device are supported.

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

ORDERING INFORMATION

ORDER CODE	DESCRIPTION	DRAWING NUMBER
PZ3064A7A44	44-pin PLCC, 7.5ns t_{PD}, Commercial temp range, 3 volt power supply, ±10%	SOT187-2
PZ3064A10A44	44-pin PLCC, 10ns t_{PD}, Commercial temp range, 3 volt power supply, ±10%	SOT187-2
PZ3064D10A44	44-pin PLCC, 10ns t_{PD}, Industrial temp range, 3 volt power supply, ±10%	SOT187-2
PZ3064D12A44	44-pin PLCC, 12ns t_{PD}, Industrial temp range, 3 volt power supply, ±10%	SOT187-2
PZ3064A7BC	44-pin TQFP, 7.5ns t_{PD}, Commercial temp range, 3 volt power supply, ±10%	SOT376-1
PZ3064A10BC	44-pin TQFP, 10ns t_{PD}, Commercial temp range, 3 volt power supply, ±10%	SOT376-1
PZ3064D10BC	44-pin TQFP, 10ns t_{PD}, Industrial temp range, 3 volt power supply, ±10%	SOT376-1
PZ3064D12BC	44-pin TQFP, 12ns t_{PD}, Industrial temp range, 3 volt power supply, ±10%	SOT376-1
PZ3064A7BP	100-pin TQFP, 7.5ns t_{PD}, Commercial temp range, 3 volt power supply, ±10%	SOT386-1
PZ3064A10BP	100-pin TQFP, 10ns t_{PD}, Commercial temp range, 3 volt power supply, ±10%	SOT386-1
PZ3064D10BP	100-pin TQFP, 10ns t_{PD}, Industrial temp range, 3 volt power supply, ±10%	SOT386-1
PZ3064D12BP	100-pin TQFP, 12ns t_{PD}, Industrial temp range, 3 volt power supply, ±10%	SOT386-1

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.

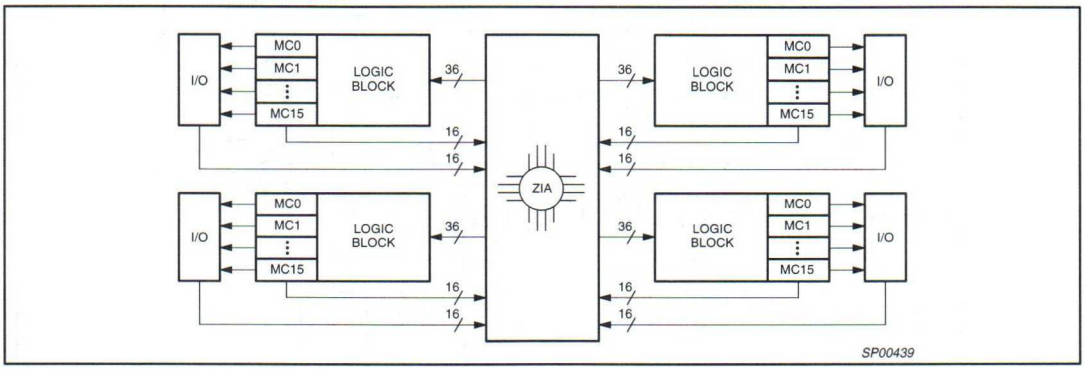


Figure 1. Philips XPLA CPLD Architecture

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. In addition, two of the control terms can be used as clock signals (see Macrocell Architecture Section for details). 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 PZ3064A/PZ3064D 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 PZ3064A/PZ3064D using 6 to 37 product terms is 9.5ns (7.5ns for the PAL + 2.0ns for the PLA).

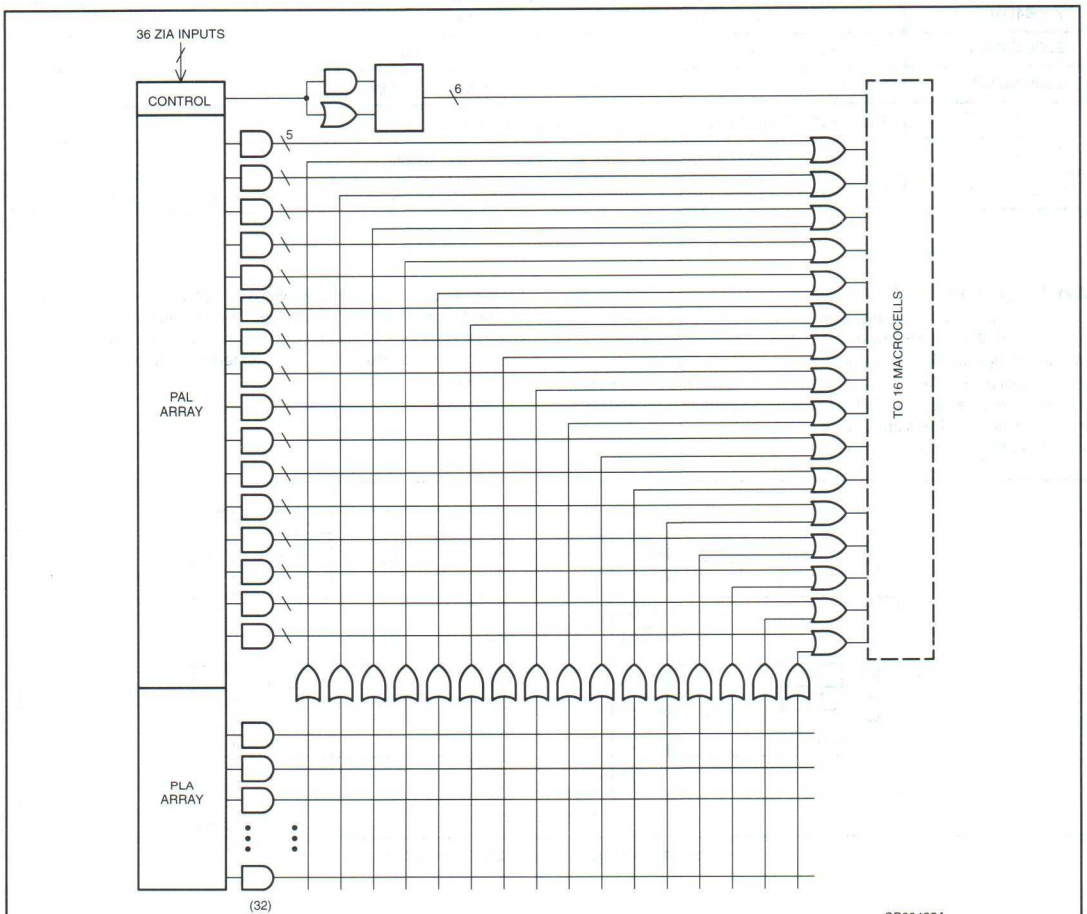


Figure 2. Philips XPLA Logic Block Architecture

Macrocell Architecture

Figure 3 shows the architecture of the macrocell used in the CoolRunner™ PZ3064A/PZ3064D. The macrocell can be configured as either a D or T type flip-flop or a combinatorial logic function. A D-type flip-flop is generally more useful for implementing state machines and data buffering while a T-type flip-flop is generally more useful in implementing counters. Each of these flip-flops can be clocked from any one of six sources. Four of the clock sources (CLK0, CLK1, CLK2, CLK3) are connected to low-skew, device-wide clock networks designed to preserve the integrity of the clock signal by reducing skew between rising and falling edges. Clock 0 (CLK0) is designated as a "synchronous" clock and must be driven by an external source. Clock 1 (CLK1), Clock 2 (CLK2), and Clock 3 (CLK3) can be used as "synchronous" clocks that are driven by an external source, or as "asynchronous" clocks that are driven by a macrocell equation. CLK0, CLK1, CLK2 and CLK3 can clock the macrocell flip-flops on either the rising edge or the falling edge of the clock signal. The other clock sources are two of the six control terms (CT2 and CT3) provided in each logic block. These clocks can be individually configured as either a PRODUCT term or SUM term equation created from the 36 signals available inside the logic block. The timing for asynchronous and control term clocks is different in that the T_{CO} time is extended by the amount of time that it takes for the signal to propagate through the array and reach the clock network, and the T_{SU} time is reduced. Please see the app note "Understanding CoolRunner™ Clocking Options" for more detail.

The six control terms of each logic block are used to control the asynchronous Preset/Reset of the flip-flops and the enable/disable of the output buffers in each macrocell. Control terms CT0 and CT1 are used to control the asynchronous Preset/Reset of the macrocell's flip-flop. Note that the Power-on Reset leaves all macrocells in the "zero" state when power is properly applied, and that the Preset/Reset feature for each macrocell can also be disabled. Control terms CT2 and CT3 can be used as a clock signal to the flip-flops of the macrocells, and as the Output Enable of the macrocell's output buffer. Control terms CT4 and CT5 can be used to control the Output Enable of the macrocell's output buffer. Having four dedicated Output Enable control terms ensures that the CoolRunner™ devices are PCI compliant. The output buffers can also be always enabled or always disabled. All CoolRunner™ devices also provide a Global Tri-State (GTS) pin, which, when enabled and 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 feedback 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 (see the section on Terminations in this data sheet and the application note *Terminating Unused CoolRunner™ I/O Pins*).

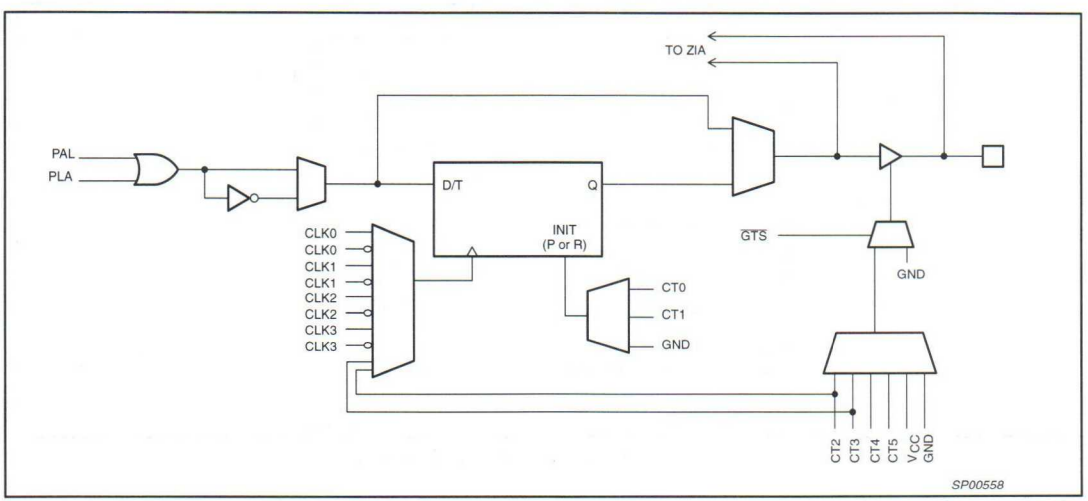


Figure 3. PZ3064A/PZ3064D Macrocell Architecture

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 PZ3064A/PZ3064D 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 PZ3064A/PZ3064D TotalCMOS™ CPLD.

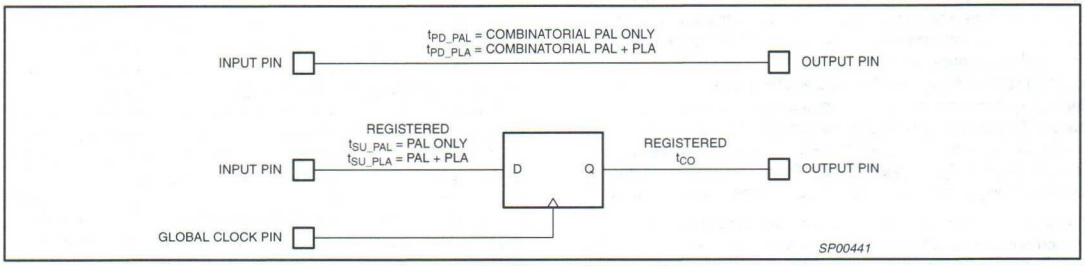


Figure 4. CoolRunner™ Timing Model

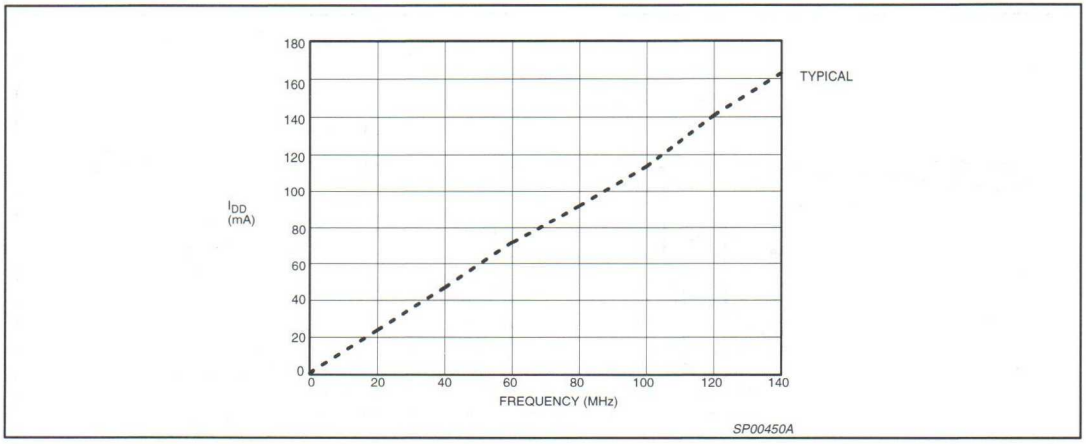


Figure 5. I_{DD} vs. Frequency @ V_{DD} = 3.3 V, 25°C

Table 2. I_{DD} vs. Frequency

V_{DD} = 3.3 V, 25°C

FREQUENCY (MHz)	0	20	40	60	80	100	120	140
Typical I_{DD} (mA)								

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. The Philips PZ3064A/PZ3064D devices use the JTAG interface for In–System Programming/Reprogramming. Although only a subset of the full JTAG command set is implemented (see Table 5), the devices are fully capable of sitting in a JTAG scan chain.

The Philips PZ3064A/PZ3064D's JTAG interface includes a TAP Port defined by the IEEE 1149.1 JTAG Specification. As implemented in the Philips PZ3064A/PZ3064D, 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 PZ3064A/PZ3064D saves an I/O pin for general purpose use by not implementing the optional TRST* signal in the JTAG interface. Instead, the Philips PZ3064A/PZ3064D 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 TAP Port should connect to an external pull–up resistor to keep the JTAG signs from floating when they are not being used.

In the Philips PZ3064A/PZ3064D, the four mandatory JTAG pins each require a unique, dedicated pin on the device. The devices come from the factory with these I/O pins set to perform JTAG functions, but through the software, the final function of these pins can be controlled. If the end application will require the device to be reprogrammed at some future time with ISP, then the pins can be left as dedicated JTAG functions, which means they are not available for use as general purpose I/O pins. However, unlike competing CPLDs, the Philips PZ3064A/PZ3064D allow the macrocells associated with these pins to be used as buried logic when the JTAG/ISP function is enabled. This is the default state for the software, and no action is required to leave these pins enabled for the JTAG/ISP functions. If, however, JTAG/ISP is not required in the end application, the software can specify that this function be turned off and that these pins be used as general purpose I/O. Because the devices initially have the JTAG/ISP functions enabled, the JEDEC file can be downloaded into the device once, after which the JTAG/ISP pins will become general purpose I/O. This feature is good for manufacturing because the devices can be programmed during test and assembly of the end product and yet still use all of the I/O pins after the programming is done. It eliminates the need for a costly, separate programming step in the manufacturing process. Of course, if the JTAG/ISP function is never required, this feature can be turned off in the software and the device can be programmed with an industry-standard programmer, leaving the pins available for I/O functions. Table 4 defines the dedicated pins used by the four mandatory JTAG signals for each of the PZ3064A/PZ3064D package types.

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.
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. PZ3064A/PZ3064D JTAG Pinout by Package Type

DEVICE	(PIN NUMBER / MACROCELL #)			
	TCK	TMS	TDI	TDO
PZ3064A/PZ3064D				
44-pin PLCC	32/C15	13/B15	7/A8	38/D8
44-pin TQFP	26/C15	7/B15	1/A8	32/D8
100–pin TQFP	62/C15	15/B15	4/A8	73/D8

Table 5. PZ3064A/PZ3064D Low-Level JTAG Boundary-Scan Commands

INSTRUCTION (Instruction Code) *Register Used*	DESCRIPTION
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.

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
  - Reconfigurability 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 PZ3064A/PZ3064D 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 PZ3064A/PZ3064D 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 PZ3064A/PZ3064D enable this feature. The ISP commands implemented in the Philips PZ3064A/PZ3064D 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.

Terminations

The CoolRunner™ PZ3064A/PZ3064D CPLDs are TotalCMOS™ devices. As with other CMOS devices, it is important to consider how to properly terminate unused inputs and I/O pins when fabricating a PC board. Allowing unused inputs and I/O pins to float can cause the voltage to be in the linear region of the CMOS input structures, which can increase the power consumption of the device. The PZ3064A/PZ3064D CPLDs have programmable on-chip pull-down resistors on each I/O pin. These pull-downs are automatically activated by the fitter software for all unused I/O pins. Note that an I/O macrocell used as buried logic that does not have the I/O pin used for input is considered to be unused, and the pull-down resistors will be turned on. We recommend that any unused I/O pins on the PZ3064A/PZ3064D device be left unconnected.

There are no on-chip pull-down structures associated with the dedicated input pins. Philips recommends that any unused dedicated inputs be terminated with external 10kΩ pull-up resistors. These pins can be directly connected to V_{CC} or GND, but using the external pull-up resistors maintains maximum design flexibility should one of the unused dedicated inputs be needed due to future design changes.

When using the JTAG/ISP functions, it is also recommended that 10kΩ pull-up resistors be used on each of the pins associated with the four mandatory JTAG signals. Letting these signals float can cause the voltage on TMS to come close to ground, which could cause the device to enter JTAG/ISP mode at unspecified times. See the application notes *JTAG and ISP in Philips Devices* and *Terminating CoolRunner™ I/O Pins* for more information.

Table 6. Low Level ISP Commands

INSTRUCTION *(Register Used)*	INSTRUCTION CODE	DESCRIPTION
Enable *(ISP Shift Register)*	1001	Enables the Erase, Program, and Verify commands.
Erase *(ISP Shift Register)*	1010	Erases the entire EEPROM array.
Program *(ISP Shift Register)*	1011	Programs the data in the ISP Shift Register into the addressed EEPROM row.
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.

JTAG and ISP Interfacing

A number of industry-established methods exist for JTAG/ISP interfacing with CPLD's and other integrated circuits. The Philips PZ3064A/PZ3064D supports the following methods:

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

For more details on JTAG and ISP for the PZ3064A/PZ3064D, 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}	TCK 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[2]	–0.5	4.6	V
V_I	Input voltage	–0.5	5.5	V
V_{OUT}	Output voltage	–0.5	5.5	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.
2. The chip supply voltage must rise monotonically.

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: $0°C \le T_{amb} \le +70°C$; $3.0\ V \le V_{DD} \le 3.6\ V$

SYMBOL	PARAMETER	TEST CONDITIONS	MIN.	MAX.	UNIT
V_{IL}	Input voltage low	V_{DD} = 3.0 V		0.8	V
V_{IH}	Input voltage high	V_{DD} = 3.6 V	2.0		V
V_I	Input clamp voltage[3]	V_{DD} = 3.0 V, I_{IN} = –18mA		–1.2	V
V_{OL}	Output voltage low	V_{DD} = 3.0 V, I_{OL} = 12mA		0.5	V
V_{OH}	Output voltage high	V_{DD} = 3.0 V, I_{OH} = –12mA	2.4		V
I_I	Input leakage current	V_{IN} = 0 to 5.25 V	–10	10	µA
I_{OZ}	3-Stated output leakage current	V_{IN} = 0 to 5.25 V	–10	10	µA
I_{DDQ}[1]	Standby current	V_{DD} = 3.6 V, T_{amb} = 0°C			µA
I_{DDD}[1, 2]	Dynamic current	V_{DD} = 3.6 V, T_{amb} = 0°C @ 1MHz			mA
		V_{DD} = 3.6 V, T_{amb} = 0°C @ 50MHz			mA
I_{OS}	Short circuit output current[3]	1 pin at a time for no longer than 1 second	–50		mA
C_{IN}	Input pin capacitance[3]	T_{amb} = 25°C, f = 1MHz		8	pF
C_{CLK}	Clock input capacitance[3]	T_{amb} = 25°C, f = 1MHz	5	12	pF
$C_{I/O}$	I/O pin capacitance[3]	T_{amb} = 25°C, f = 1MHz		10	pF

NOTES:
1. See Table 2, page 154 for typical values.
2. 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.
3. This parameter guaranteed by design and characterization, not by test.

AC ELECTRICAL CHARACTERISTICS[1] FOR COMMERCIAL GRADE DEVICES

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

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 (global synchronous clock from pin)	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[2, 3]		10		12	ns
t_{EA}	Input to output valid[2]		10		12	ns
t_{RP}	Input to register preset[2]		10		12.5	ns
t_{RR}	Input to register reset[2]		10		12.5	ns

NOTES:
1. Specifications measured with one output switching. See Figure 6 and Table 7 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: $-40°C \le T_{amb} \le +85°C$; $3.0\ V \le V_{DD} \le 3.6\ V$

SYMBOL	PARAMETER	TEST CONDITIONS	MIN.	MAX.	UNIT
V_{IL}	Input voltage low	V_{DD} = 3.0 V		0.8	V
V_{IH}	Input voltage high	V_{DD} = 3.6 V	2.0		V
V_I	Input clamp voltage[3]	V_{DD} = 3.0 V, I_{IN} = –18mA		–1.2	V
V_{OL}	Output voltage low	V_{DD} = 3.0 V, I_{OL} = 12mA		0.5	V
V_{OH}	Output voltage high	V_{DD} = 3.0 V, I_{OH} = –12mA	2.4		V
I_I	Input leakage current	V_{IN} = 0 to 5.5 V	–10	10	µA
I_{OZ}	3-Stated output leakage current	V_{IN} = 0 to 5.5 V	–10	10	µA
I_{DDQ}[1]	Standby current	V_{DD} = 3.6 V, T_{amb} = –40°C			µA
I_{DDD}[1, 2]	Dynamic current	V_{DD} = 3.6 V, T_{amb} = –40°C @ 1MHz			mA
		V_{DD} = 3.6 V, T_{amb} = –40°C @ 50MHz			mA
I_{OS}	Short circuit output current[3]	1 pin at a time for no longer than 1 second	–50	–130	mA
C_{IN}	Input pin capacitance[3]	T_{amb} = 25°C, f = 1MHz		8	pF
C_{CLK}	Clock input capacitance[3]	T_{amb} = 25°C, f = 1MHz	5	12	pF
$C_{I/O}$	I/O pin capacitance[3]	T_{amb} = 25°C, f = 1MHz		10	pF

NOTES:
1. See Table 2, page 154 for typical values.
2. 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.
3. This parameter guaranteed by design and characterization, not by test.

AC ELECTRICAL CHARACTERISTICS[1] FOR INDUSTRIAL GRADE DEVICES

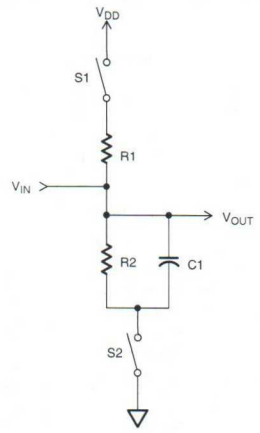
Industrial: $-40°C \le T_{amb} \le +85°C$; $3.0\ V \le V_{DD} \le 3.6\ V$

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 (global synchronous clock from pin)	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[2, 3]		12		13	ns
t_{EA}	Input to output valid[2]		12		13	ns
t_{RP}	Input to register preset[2]		12.5		13.5	ns
t_{RR}	Input to register reset[2]		12.5		13.5	ns

NOTES:
1. Specifications measured with one output switching. See Figure 6 and Table 7 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.

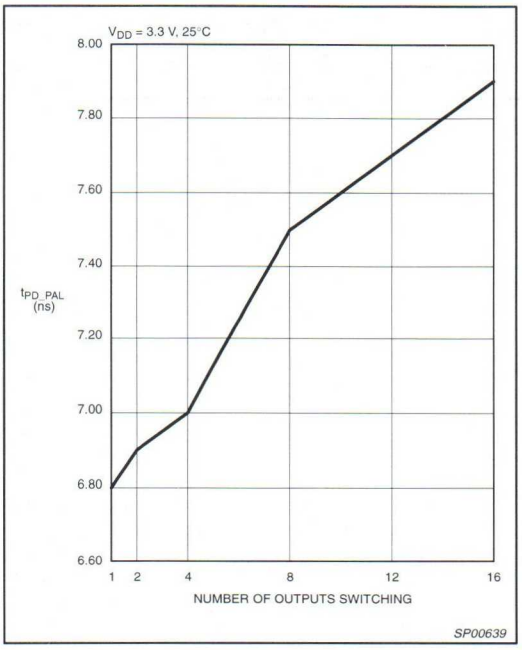


COMPONENT	VALUES
R1	390Ω
R2	390Ω
C1	35pF

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

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

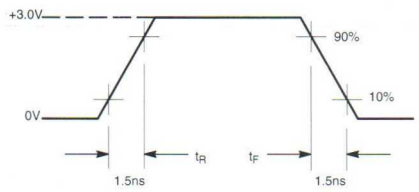
SP00623



SP00639

Figure 6. t_{PD_PAL} vs. Outputs Switching

VOLTAGE WAVEFORM



SP00368

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

Input Pulses

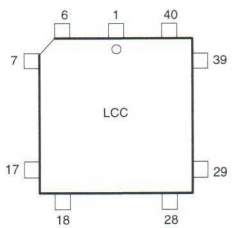
Table 7. t_{PD_PAL} vs. Number of Outputs Switching

V_{DD} = 3.3 V, 25°C

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

PIN DESCRIPTIONS

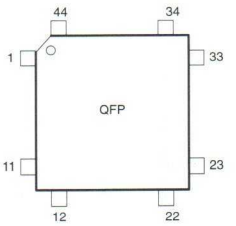
PZ3064 – 44-Pin Plastic Leaded Chip Carrier



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		

SP00554

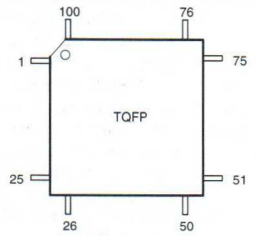
PZ3064 – 44-Pin Thin Quad Flat Package



Pin	Function	Pin	Function	Pin	Function
1	I/O-A8 (TDI)	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		

SP00624

PZ3064 – 100-Pin Thin Quad Flat Package



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

SP00556

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
100-pin TQFP	47.4°C/W

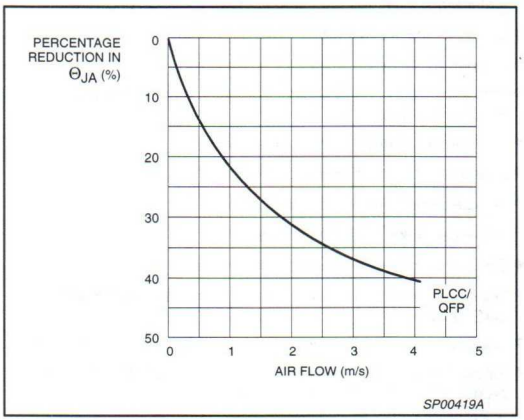


Figure 7. Average Effect of Airflow on Θ_{JA}

128 macrocell CPLD with enhanced clocking PZ3128A/PZ3128D

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
- 3 Volt, In-System Programmable (ISP) using a JTAG interface
  - On-chip supervoltage generation
  - ISP commands include: Enable, Erase, Program, Verify
  - Supported by multiple ISP programming platforms
  - 4 pin JTAG interface (TCK, TMS, TDI, TDO)
  - JTAG commands include: Bypass, Idcode
- 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
- 5V tolerant I/Os to support mixed voltage systems
- 100% routable with 100% utilization while all pins and all macrocells are fixed
- Deterministic timing model that is extremely simple to use
- Up to 20 clocks available
- 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
- Advanced 0.35µ 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
  - up to 2 asynchronous clocks
- Programmable global 3-State pin facilitates 'bed of nails' testing without using logic resources
- Available in TQFP and LQFP packages
- Available in both Commercial and Industrial grades

Table 1. PZ3128A/PZ3128D Features

	PZ3128A/PZ3128D
Usable gates	4000
Maximum inputs	100
Maximum I/Os	96
Number of macrocells	128
Propagation delay (ns)	7.5
Packages	100-pin TQFP, 128-pin LQFP

DESCRIPTION

The PZ3128A/PZ3128D CPLD (Complex Programmable Logic Device) is a member of the Fast Zero Power (FZP™) family of CPLDs from Philips Semiconductors. These devices combine high speed and zero power in a 128 macrocell CPLD. With the FZP™ design technique, the PZ3128A/PZ3128D 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.

The Philips FZP™ CPLDs introduce the new patented 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 PZ3128A/PZ3128D CPLDs are supported by industry standard CAE tools (Cadence, Exemplar Logic, Mentor, OrCAD, 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 PZ3128A/PZ3128D CPLD is electrically reprogrammable using industry standard device programmers from vendors such as Data I/O, BP Microsystems, SMS, and others. The PZ3128A/PZ3128D also includes an industry-standard, IEEE 1149.1, JTAG interface through which In-System Programming (ISP) and reprogramming of the device are supported.

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

ORDERING INFORMATION

ORDER CODE	DESCRIPTION	I/O COUNT	DRAWING NUMBER
PZ3128AS7BP	100-pin TQFP, 7.5ns t_{PD}, Commercial temp range, 3 volt power supply, ±10%	80	SOT386-1
PZ3128AS10BP	100-pin TQFP, 10ns t_{PD}, Commercial temp range, 3 volt power supply, ±10%	80	SOT386-1
PZ3128AS12BP	100-pin TQFP, 12ns t_{PD}, Commercial temp range, 3 volt power supply, ±10%	80	SOT386-1
PZ3128DS10BP	100-pin TQFP, 10ns t_{PD}, Industrial temp range, 3 volt power supply, ±10%	80	SOT386-1
PZ3128DS15BP	100-pin TQFP, 15ns t_{PD}, Industrial temp range, 3 volt power supply, ±10%	80	SOT386-1
PZ3128AS7BE	128-pin LQFP, 7.5ns t_{PD}, Commercial temp range, 3 volt power supply, ±10%	96	SOT425-1
PZ3128AS10BE	128-pin LQFP, 10ns t_{PD}, Commercial temp range, 3 volt power supply, ±10%	96	SOT425-1
PZ3128AS12BE	128-pin LQFP, 12ns t_{PD}, Commercial temp range, 3 volt power supply, ±10%	96	SOT425-1
PZ3128DS10BE	128-pin LQFP, 10ns t_{PD}, Industrial temp range, 3 volt power supply, ±10%	96	SOT425-1
PZ3128DS15BE	128-pin LQFP, 15ns t_{PD}, Industrial temp range, 3 volt power supply, ±10%	96	SOT425-1

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.

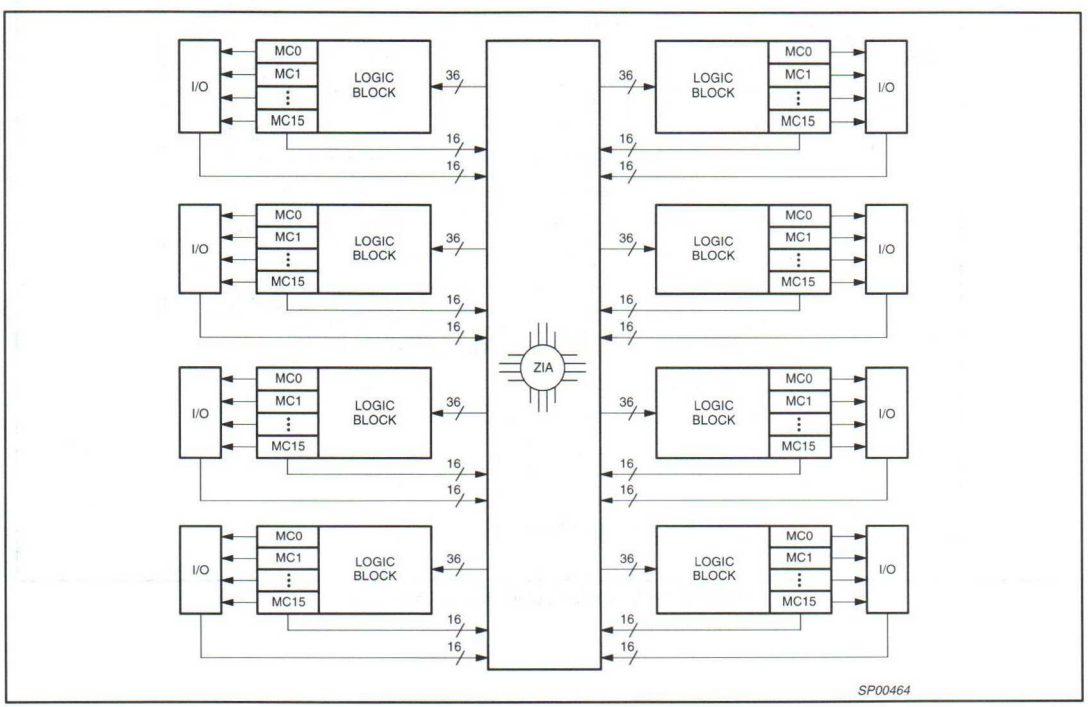


Figure 1. Philips XPLA CPLD Architecture

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. In addition, two of the control terms can be used as clock signals (see Macrocell Architecture section for details). 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 PZ3128A/PZ3128D 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 PZ3128A/PZ3128D using 6 to 37 product terms is 9.5ns (7.5ns for the PAL + 2ns for the PLA).

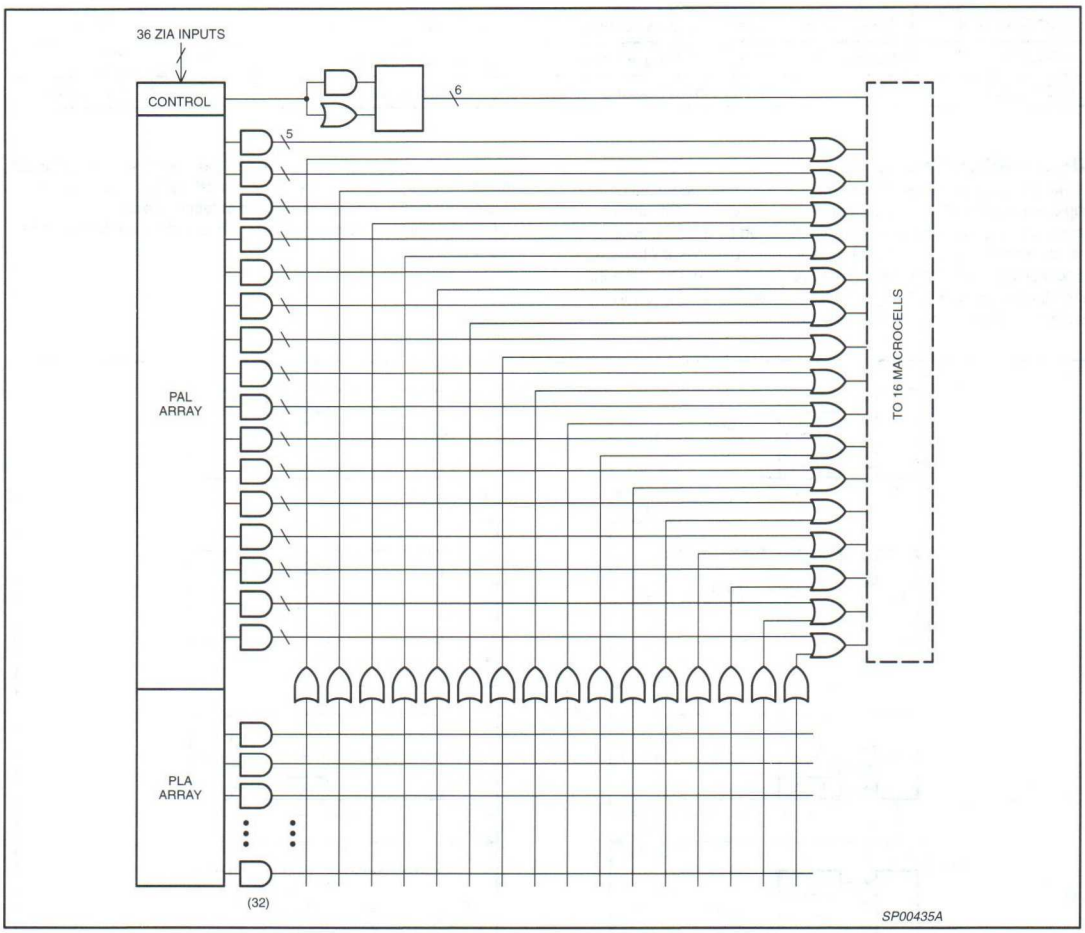


Figure 2. Philips XPLA Logic Block Architecture

Macrocell Architecture

Figure 3 shows the architecture of the macrocell used in the CoolRunner™ PZ3128A/PZ3128D. The macrocell can be configured as either a D or T type flip-flop or a combinatorial logic function. A D-type flip-flop is generally more useful for implementing state machines and data buffering while a T-type flip-flop is generally more useful in implementing counters. Each of these flip-flops can be clocked from any one of six sources. Four of the clock sources (CLK0, CLK1, CLK2, CLK3) are connected to low-skew, device-wide clock networks designed to preserve the integrity of the clock signal by reducing skew between rising and falling edges. Clock 0 (CLK0) is designated as a "synchronous" clock and must be driven by an external source. Clock 1 (CLK1), Clock 2 (CLK2), and Clock 3 (CLK3) can be used as "synchronous" clocks that are driven by an external source, or as "asynchronous" clocks that are driven by a macrocell equation. CLK0, CLK1, CLK2 and CLK3 can clock the macrocell flip-flops on either the rising edge or the falling edge of the clock signal. The other clock sources are two of the six control terms (CT2 and CT3) provided in each logic block. These clocks can be individually configured as either a PRODUCT term or SUM term equation created from the 36 signals available inside the logic block. The timing for asynchronous and control term clocks is different in that the TCO time is extended by the amount of time that it takes for the signal to propagate through the array and reach the clock network, and the TSU time is reduced. Please see the app note titled "Understanding CoolRunner™ Clocking Options" for more detail.

The six control terms of each logic block are used to control the asynchronous Preset/Reset of the flip-flops and the enable/disable of the output buffers in each macrocell. Control terms CT0 and CT1 are used to control the asynchronous Preset/Reset of the macrocell's flip-flop. Note that the Power-on Reset leaves all macrocells in the "zero" state when power is properly applied, and that the Preset/Reset feature for each macrocell can also be disabled. Control terms CT2 and CT3 can be used as a clock signal to the flip-flops of the macrocells, and as the Output Enable of the macrocell's output buffer. Control terms CT4 and CT5 can be used to control the Output Enable of the macrocell's output buffer. Having four dedicated Output Enable control terms ensures that the CoolRunner™ devices are PCI compliant. The output buffers can also be always enabled or always disabled. All CoolRunner™ devices also provide a Global Tri-State ($\overline{GTS}$) pin, which, when enabled and 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 feedback 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 (see the section on Terminations in this data sheet and the Application Note *Terminating Unused CoolRunner™ I/O Pins*).

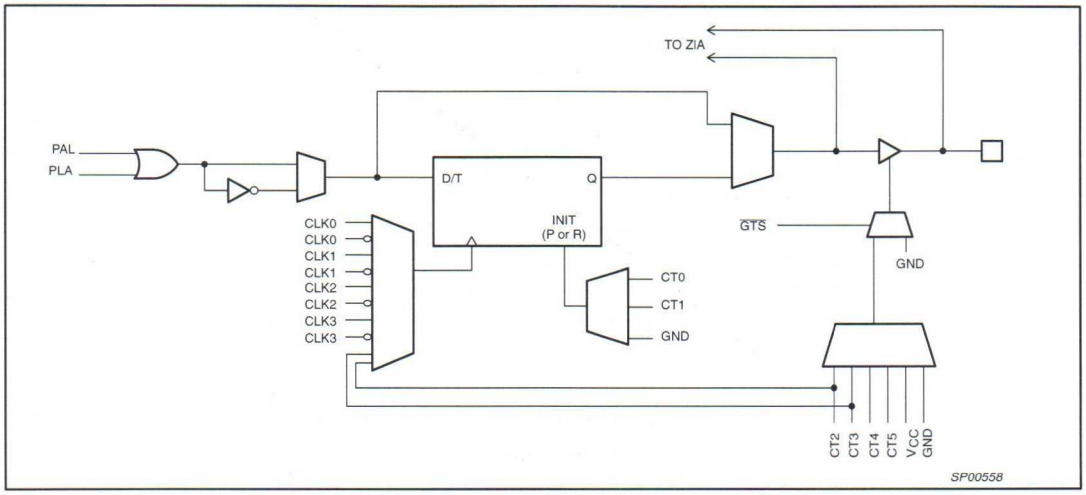


Figure 3. PZ3128A/PZ3128D Macrocell Architecture

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 PZ3128A/PZ3128D TotalCMOS™ CPLD (data taken w/eight up/down, loadable 16 bit counters@3.3V, 25°C).

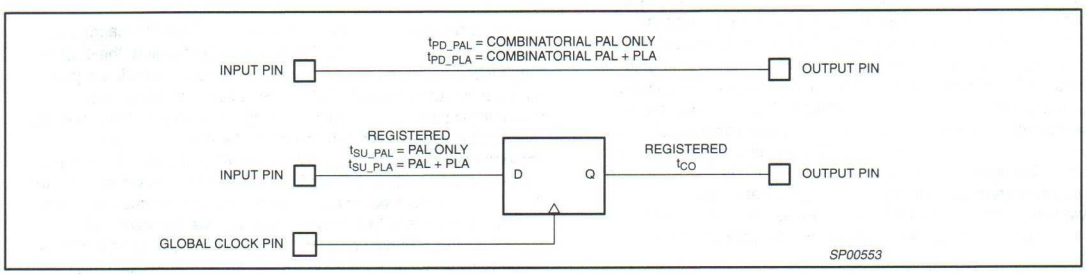


Figure 4. CoolRunner™ Timing Model

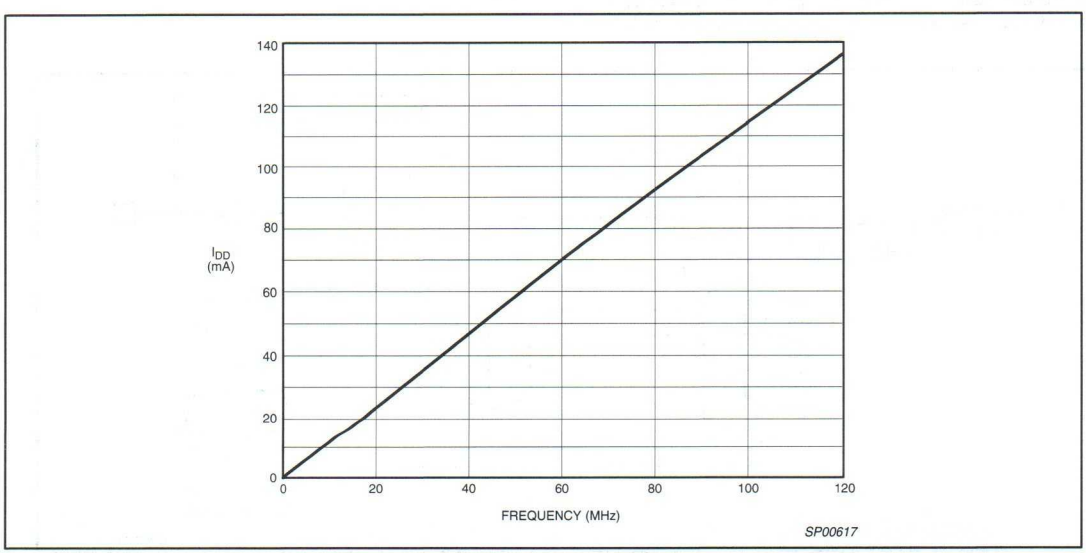


Figure 5. I_{DD} vs. Frequency @ V_{DD} = 3.3V, 25°C

Table 2. I_{DD} vs. Frequency

V_{DD} = 3.3V, 25°C

FREQUENCY (MHz)	0	1	20	40	60	80	100	120
Typical I_{DD} (mA)								

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. The Philips PZ3128A/PZ3128D devices use the JTAG Interface for In–System Programming/Reprogramming. Although only a subset of the full JTAG command set is implemented (see Table 5), the devices are fully capable of sitting in a JTAG scan chain.

The Philips PZ3128A/PZ3128D's JTAG interface includes a TAP Port defined by the IEEE 1149.1 JTAG Specification. As implemented in the Philips PZ3128A/PZ3128D, 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 PZ3128A/PZ3128D saves an I/O pin for general purpose use by not implementing the optional TRST* signal in the JTAG interface. Instead, the Philips PZ3128A/PZ3128D 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 TAP Port should connect to an external pull-up resistor to keep the JTAG signals from floating when they are not being used.

In the Philips PZ3128A/PZ3128D, the four mandatory JTAG pins each require a unique, dedicated pin on the device. The devices come from the factory with these I/O pins set to perform JTAG functions, but through the software, the final function of these pins can be controlled. If the end application will require the device to be reprogrammed at some future time with ISP, then the pins can be left as dedicated JTAG functions, which means they are not available for use as general purpose I/O pins. However, unlike competing CPLDs, the Philips PZ3128A/PZ3128D allow the macrocells associated with these pins to be used as buried logic when the JTAG/ISP function is enabled. This is the default state for the software, and no action is required to leave these pins enabled for the JTAG/ISP functions. If, however, JTAG/ISP is not required in the end application, the software can specify that this function be turned off and that these pins be used as general purpose I/O. Because the devices initially have the JTAG/ISP functions enabled, the JEDEC file can be downloaded into the device once, after which the JTAG/ISP pins will become general purpose I/O. This feature is good for manufacturing because the devices can be programmed during test and assembly of the end product and yet still use all of the I/O pins after the programming is done. It eliminates the need for a costly, separate programming step in the manufacturing process. Of course, if the JTAG/ISP function is never required, this feature can be turned off in the software and the device can be programmed with an industry-standard programmer, leaving the pins available for I/O functions. Table 4 defines the dedicated pins used by the four mandatory JTAG signals for each of the PZ3128A/PZ3128D package types.

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.
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. PZ3128A/PZ3128D JTAG Pinout by Package Type

DEVICE	(PIN NUMBER / MACROCELL #)			
	TCK	TMS	TDI	TDO
PZ3128A/PZ3128D				
100-pin TQFP	62/F15	15/C15	4/B15	73/G15
128-pin LQFP	82/F15	21/C15	8/B15	95/G15

Table 5. PZ3128A/PZ3128D Low-Level JTAG Boundary-Scan Commands

INSTRUCTION (Instruction Code) *Register Used*	DESCRIPTION
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.

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 PZ3128A/PZ3128D 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 PZ3128A/PZ3128D 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 PZ3128A/PZ3128D enable this feature. The ISP commands implemented in the Philips PZ3128A/PZ3128D 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.

Terminations

The CoolRunner™ PZ3128A/PZ3128D CPLDs are TotalCMOS™ devices. As with other CMOS devices, it is important to consider how to properly terminate unused inputs and I/O pins when fabricating a PC board. Allowing unused inputs and I/O pins to float can cause the voltage to be in the linear region of the CMOS input structures, which can increase the power consumption of the device. The PZ3128A/PZ3128D CPLDs have programmable on-chip pull-down resistors on each I/O pin. These pull-downs are automatically activated by the fitter software for all unused I/O pins. Note that an I/O macrocell used as buried logic that does not have the I/O pin used for input is considered to be unused, and the pull-down resistors will be turned on. We recommend that any unused I/O pins on the PZ3128A/PZ3128D device be left unconnected.

There are no on-chip pull-down structures associated with the dedicated input pins. Philips recommends that any unused dedicated inputs be terminated with external 10kΩ pull-up resistors. These pins can be directly connected to V_{CC} or GND, but using the external pull-up resistors maintains maximum design flexibility should one of the unused dedicated inputs be needed due to future design changes.

When using the JTAG/ISP functions, it is also recommended that 10kΩ pull-up resistors be used on each of the pins associated with the four mandatory JTAG signals. Letting these signals float can cause the voltage on TMS to come close to ground, which could cause the device to enter JTAG/ISP mode at unspecified times. See the application notes *ISP Design Considerations for CoolRunner™ CPLDs* and *Terminating CoolRunner™ I/O Pins* for more information.

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 the user.

JTAG and ISP Interfacing

A number of industry-established methods exist for JTAG/ISP interfacing with CPLD's and other integrated circuits. The Philips PZ3128A/PZ3128D supports the following methods:

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

For more details on JTAG and ISP for the PZ3128A/PZ3128D, 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[2]	–0.5	4.6	V
V_I	Input voltage	–0.5	5.5	V
V_{OUT}	Output voltage	–0.5	5.5	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.
2. The chip supply voltage must rise monotonically.

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 5.25 V	–10	10	µA
I_{OZ}	3-Stated output leakage current	V_{IN} = 0 to 5.25 V	–10	10	µA
I_{DDQ}[1]	Standby current	V_{DD} = 3.6V, T_{amb} = 0°C			µA
I_{DDD}[1, 2]	Dynamic current	V_{DD} = 3.6V, T_{amb} = 0°C @ 1MHz			mA
		V_{DD} = 3.6V, T_{amb} = 0°C @ 50MHz			mA
I_{OS}	Short circuit output current[3]	1 pin at a time for no longer than 1 second	–50	–100	mA
C_{IN}	Input pin capacitance[3]	T_{amb} = 25°C, f = 1MHz		8	pF
C_{CLK}	Clock input capacitance[3]	T_{amb} = 25°C, f = 1MHz	5	12	pF
$C_{I/O}$	I/O pin capacitance[3]	T_{amb} = 25°C, f = 1MHz		10	pF

NOTES:
1. See Table 2 on page 168 for typical values.
2. This parameter measured with a 16-bit, loadable up/down counter loaded into every logic block, with all outputs disabled and unloaded. Inputs are tied to V_{DD} or ground. This parameter guaranteed by design and characterization, not testing.
3. 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	7		10		12		UNIT
		MIN.	MAX.	MIN.	MAX.	MIN.	MAX.	
t_{PD_PAL}	Propagation delay time, input (or feedback node) to output through PAL	2	7.5	2	10	2	12	ns
t_{PD_PLA}	Propagation delay time, input (or feedback node) to output through PAL & PLA	3	9.5	3	12	3	14.5	ns
t_{CO}	Clock to out (global synchronous clock from pin)	2	5.5	2	7	2	8	ns
t_{SU_PAL}	Setup time (from input or feedback node) through PAL	4.5		7		8		ns
t_{SU_PLA}	Setup time (from input or feedback node) through PAL + PLA	6.5		9		10.5		ns
t_H	Hold time		0		0		0	ns
t_{CH}	Clock High time	3		4		4		ns
t_{CL}	Clock Low time	3		4		4		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		125		MHz
f_{MAX2}	Maximum internal frequency[2] $1/(t_{SUPAL} + t_{CF})$	111		80		69		MHz
f_{MAX3}	Maximum external frequency[2] $1/(t_{SUPAL} + t_{CO})$	95		71		63		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	2	6	2	8.5	2	10.5	ns
t_{PDF_PLA}	Input (or feedback node) to internal feedback node delay time through PAL+PLA	3	8	3	10.5	3	13	ns
t_{CF}	Clock to internal feedback node delay time		4		5.5		6.5	ns
t_{INIT}	Delay from valid V_{DD} to valid reset		50		50		50	µs
t_{ER}	Input to output disable[2, 3]		9		12		15	ns
t_{EA}	Input to output valid[2]		9		12		15	ns
t_{RP}	Input to register preset[2]		11		12.5		15	ns
t_{RR}	Input to register reset[2]		11		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: $-40°C \le T_{amb} \le +85°C$; $3.0V \le V_{DD} \le 3.6V$

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} = 12mA		0.5	V
V_{OH}	Output voltage high	V_{DD} = 3.0V, I_{OH} = –12mA	2.4		V
I_I	Input leakage current	V_{IN} = 0 to 5.5 V	–10	10	µA
I_{OZ}	3-Stated output leakage current	V_{IN} = 0 to 5.5 V	–10	10	µA
I_{DDQ}[1]	Standby current	V_{DD} = 3.6V, T_{amb} = –40°C			µA
I_{DDD}[1, 2]	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[3]	1 pin at a time for no longer than 1 second	–50	–130	mA
C_{IN}	Input pin capacitance[3]	T_{amb} = 25°C, f = 1MHz		8	pF
C_{CLK}	Clock input capacitance[3]	T_{amb} = 25°C, f = 1MHz	5	12	pF
$C_{I/O}$	I/O pin capacitance[3]	T_{amb} = 25°C, f = 1MHz		10	pF

NOTES:
1. See Table 2 on page 168 for typical values.
2. This parameter measured with a 16-bit, loadable up/down counter loaded into every logic block, with all outputs disabled and unloaded. Inputs are tied to V_{DD} or ground. This parameter guaranteed by design and characterization, not testing.
3. Typical values, not tested.

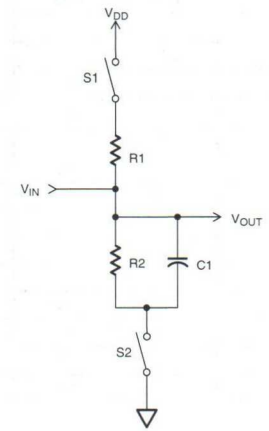
AC ELECTRICAL CHARACTERISTICS[1] FOR INDUSTRIAL GRADE DEVICES

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

SYMBOL	PARAMETER	10		15		UNIT
		MIN.	MAX.	MIN.	MAX.	
t_{PD_PAL}	Propagation delay time, input (or feedback node) to output through PAL	2	10	2	15	ns
t_{PD_PLA}	Propagation delay time, input (or feedback node) to output through PAL & PLA	3	12	3	17.5	ns
t_{CO}	Clock to out (global synchronous clock from pin)	2	7	2	8	ns
t_{SU_PAL}	Setup time (from input or feedback node) through PAL	8		8		ns
t_{SU_PLA}	Setup time (from input or feedback node) through PAL + PLA	10		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] $1/(t_{SUPAL} + t_{CF})$	71		69		MHz
f_{MAX3}	Maximum external frequency[2] $1/(t_{SUPAL} + t_{CO})$	66		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	13.5	ns
t_{PDF_PLA}	Input (or feedback node) to internal feedback node delay time through PAL+PLA	3	10.5	3	16	ns
t_{CF}	Clock to internal feedback node delay time		6		6.5	ns
t_{INIT}	Delay from valid V_{DD} to valid reset		50		50	µs
t_{ER}	Input to output disable[2, 3]		15		15	ns
t_{EA}	Input to output valid[2]		15		15	ns
t_{RP}	Input to register preset[2]		15		17	ns
t_{RR}	Input to register reset[2]		15		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.

SWITCHING CHARACTERISTICS



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.

SP00618

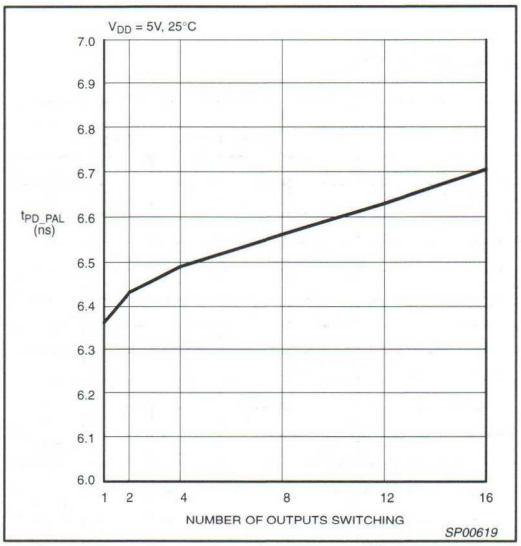
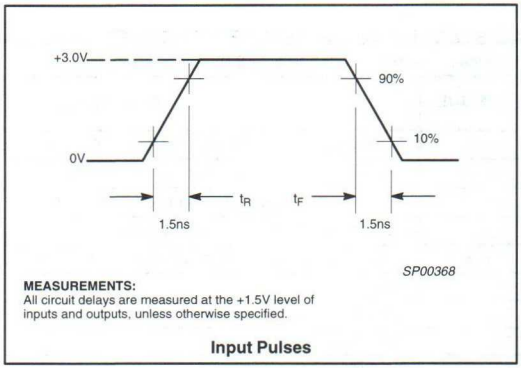


Figure 6. t_{PD_PAL} vs. Outputs Switching

VOLTAGE WAVEFORM



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

Input Pulses

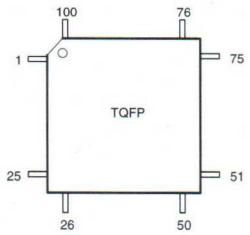
Table 8. t_{PD_PAL} vs. Number of Outputs Switching

V_{DD} = 3.3V, 25°C

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

PIN DESCRIPTIONS

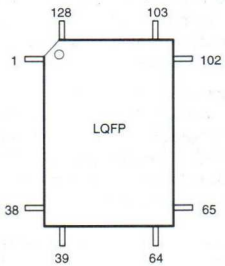
100-Pin Thin Quad Flat Package



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		

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		

SP00469A

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}
100-pin TQFP	47.4 °C/W
128-pin LQFP	45.0 °C/W

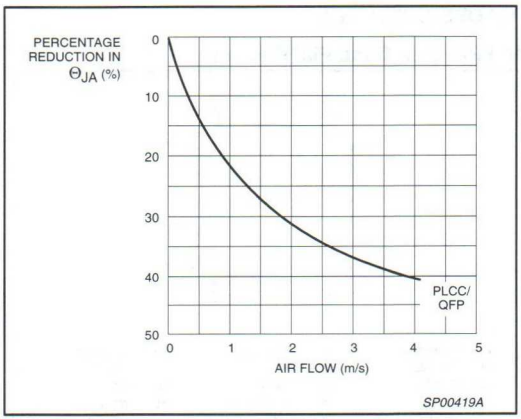


Figure 7. Average Effect of Airflow on Θ_{JA}

32 macrocell CPLD with enhanced clocking PZ5032C

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
- Up to 6 clocks with programmable polarity at every macrocell
- 5 Volt, In-System Programmable (ISP) using a JTAG interface
  - On-chip supervoltage generation
  - ISP commands include: Enable, Erase, Program, Verify
  - Supported by multiple ISP programming platforms
  - 4 pin JTAG interface (TCK, TMS, TDI, TDO)
  - JTAG commands include: Bypass, Idcode
- 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
  - Up to 2 asynchronous clocks
- Programmable global 3-State pin facilitates 'bed of nails' testing without using logic resources
- Available in both PLCC and TQFP packages

Table 1. PZ5032C Features

	PZ5032C
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

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

DESCRIPTION

The PZ5032C CPLD (Complex Programmable Logic Device) is a member of the Fast Zero Power (FZP™) family of CPLDs from Philips Semiconductors. These devices combine high speed and zero power in a 32 macrocell CPLD. With the FZP™ design technique, the PZ5032C 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 PZ3032C CPLD that offers pin-to-pin speeds of 8ns.

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 PZ5032C CPLDs are supported by industry standard CAE tools (Cadence, Exemplar Logic, Minc, 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 PZ5032C CPLD is reprogrammable using industry standard device programmers from vendors such as Data I/O, BP Microsystems, SMS, and others. The PZ5032C also includes an industry-standard, IEEE 1149.1, JTAG interface through which In-System Programming (ISP) and reprogramming of the device are supported.

ORDERING INFORMATION

ORDER CODE	DESCRIPTION	DESCRIPTION	DRAWING NUMBER
PZ5032CS6A44	44-pin PLCC, 6ns t_{PD}	Commercial temp range, 5 volt power supply, ± 5%	SOT187-2
PZ5032CS7A44	44-pin PLCC, 7.5ns t_{PD}	Commercial temp range, 5 volt power supply, ± 5%	SOT187-2
PZ5032CS10A44	44-pin PLCC, 10ns t_{PD}	Commercial temp range, 5 volt power supply, ± 5%	SOT187-2
PZ5032CS6BC	44-pin TQFP, 6ns t_{PD},	Commercial temp range, 5 volt power supply, ± 5%	SOT376-1
PZ5032CS7BC	44-pin TQFP, 7.5ns t_{PD}	Commercial temp range, 5 volt power supply, ± 5%	SOT376-1
PZ5032CS10BC	44-pin TQFP, 10ns t_{PD}	Commercial temp range, 5 volt power supply, ± 5%	SOT376-1

XPLA™ ARCHITECTURE

Figure 1 shows a high level block diagram of a 32 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. In addition, two of the control terms can be used as clock signals (see Macrocell Architecture section for details). 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 PZ5032C 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 PZ5032C using 6 to 37 product terms is 8ns (6ns for the PAL + 2ns for the PLA).

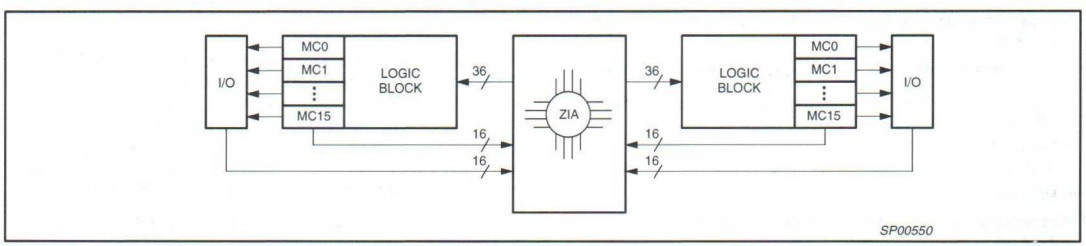


Figure 1. Philips XPLA CPLD Architecture

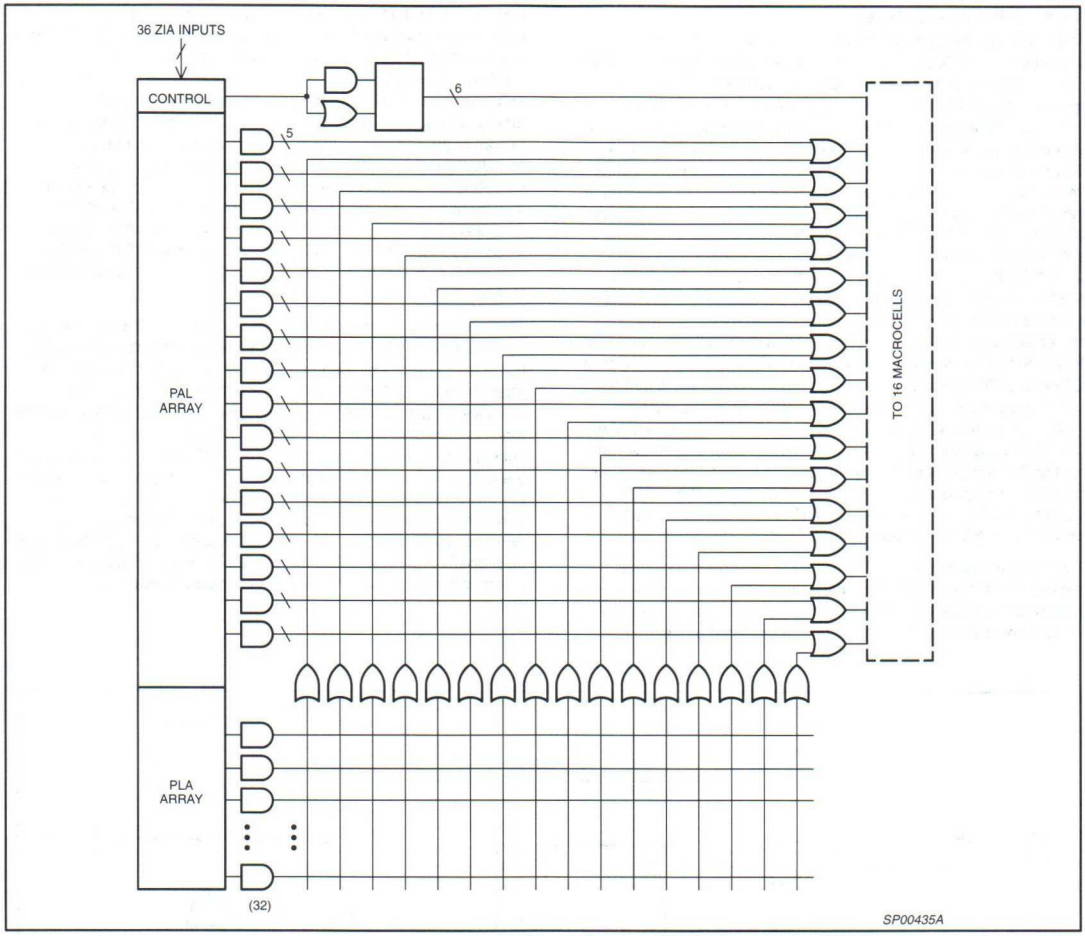


Figure 2. Philips XPLA Logic Block Architecture

Macrocell Architecture

Figure 3 shows the architecture of the macrocell used in the CoolRunner™ PZ5032C. The macrocell can be configured as either a D or T type flip-flop or a combinatorial logic function. A D-type flip-flop is generally more useful for implementing state machines and data buffering while a T-type flip-flop is generally more useful in implementing counters. Each of these flip-flops can be clocked from any one of four sources. Two of the clock sources (CLK0 and CLK1) are connected to low-skew, device-wide clock networks designed to preserve the integrity of the clock signal by reducing skew between rising and falling edges. Clock 0 (CLK0) is designated as a "synchronous" clock and must be driven by an external source. Clock 1 (CLK1) can be used as a "synchronous" clock that is driven by an external source, or as an "asynchronous" clock that is driven by a macrocell equation. Both CLK0 and CLK1 can clock the macrocell flip-flops on either the rising edge or the falling edge of the clock signal. The other clock sources are two of the six control terms (CT2 and CT3) provided in each logic block. These clocks can be individually configured as either a PRODUCT term or SUM term equation created from the 36 signals available inside the logic block. The timing for asynchronous and control term clocks is different in that the Tco time is extended by the amount of time that it takes for the signal to propagate through the array and reach the clock network, and the Tsu time is reduced. Please see the app note titled "Understanding CoolRunner Clocking Options" for more detail.

The six control terms of each logic block are used to control the asynchronous Preset/Reset of the flip-flops and the enable/disable of the output buffers in each macrocell. Control terms CT0 and CT1 are used to control the asynchronous Preset/Reset of the macrocell's flip-flop. Note that the Power-on Reset leaves all macrocells in the "zero" state when power is properly applied, and that the Preset/Reset feature for each macrocell can also be disabled. Control terms CT2 and CT3 can be used as a clock signal to the flip-flops of the macrocells, and as the Output Enable of the macrocell's output buffer. Control terms CT4 and CT5 can be used to control the Output Enable of the macrocell's output buffer. Having four dedicated Output Enable control terms ensures that the CoolRunner™ devices are PCI compliant. The output buffers can also be always enabled or always disabled. All CoolRunner™ devices also provide a Global Tri-State (GTS) pin, which, when enabled and 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 feedback 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 (See the section on terminations in this data sheet and the app note *Terminating Unused CoolRunner™ I/O Pins*).

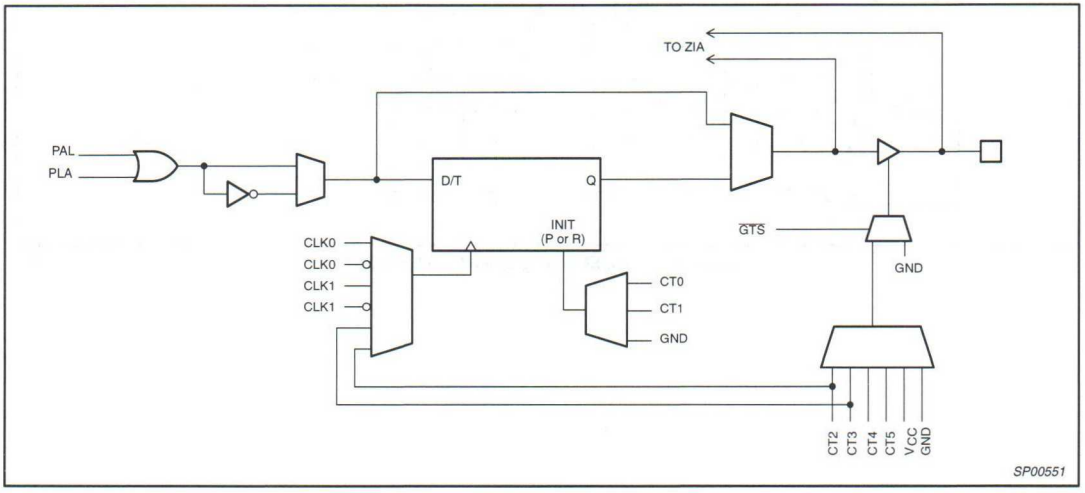


Figure 3. PZ5032C Macrocell Architecture

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 PZ5032C 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 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 PZ5032C TotalCMOS™ CPLD.

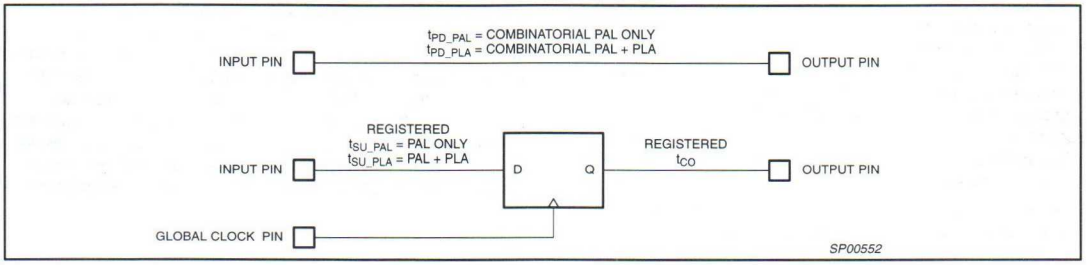


Figure 4. CoolRunner™ Timing Model

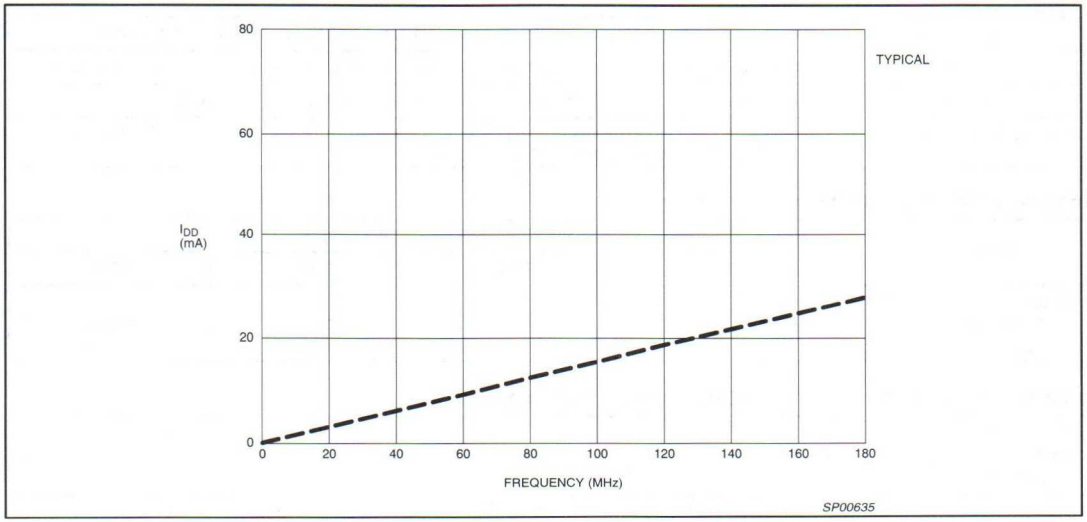


Figure 5. I_{DD} vs. Frequency @ V_{DD} = 5.0V

Table 2. I_{DD} vs Frequency

V_{DD} = 5.00V

FREQ (MHz)	0	1	20	40	60	80	100	120	140	160	180
Typical I_{DD} (mA)	0.04	0.20	3.14	6.25	9.32	12.5	15.5	18.7	21.7	24.7	27.8

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. The Philips PZ5032C devices use the JTAG interface for In–System Programming/Reprogramming. Although only a subset of the full JTAG command set is implemented (see Table 5), the devices are fully capable of sitting in a JTAG scan chain.

The Philips PZ5032C's JTAG interface includes a TAP Port defined by the IEEE 1149.1 JTAG Specification. As implemented in the Philips PZ5032C, 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 PZ5032C saves an I/O pin for general purpose use by not implementing the optional TRST* signal in the JTAG interface. Instead, the Philips PZ5032C 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 TAP Port should connect to an external pull–up resistor to keep the JTAG pins from floating when they are not being used (see section on Terminations).

In the Philips PZ5032C, the four mandatory JTAG pins each require a unique, dedicated pin on the device. The devices come from the factory with these I/O pins set to perform JTAG functions, but through the software, the final function of these pins can be controlled. If the end application will require the device to be reprogrammed at some future time with ISP, then the pins can be left as dedicated JTAG functions, which means they are not available for use as general purpose I/O pins. However, unlike competing CPLDs, the Philips PZ5032C allow the macrocells associated with these pins to be used as buried logic when the JTAG/ISP function is enabled. This is the default state for the software, and no action is required to leave these pins enabled for the JTAG/ISP functions. If, however, JTAG/ISP is not required in the end application, the software can specify that this function be turned off and that these pins be used as general purpose I/O. Because the devices initially have the JTAG/ISP functions enabled, the JEDEC file can be down loaded into the device once, after which the JTAG/ISP pins will become general purpose I/O. This feature is good for manufacturing because the devices can be programmed during test and assembly of the end product and yet still use all of the I/O pins after the programming is done. It eliminates the need for a costly, separate programming step in the manufacturing process. Of course, if the JTAG/ISP function is never required, this feature can be turned off in the software and the device can be programmed with an industry-standard programmer, leaving the pins available for I/O functions. Table 4 defines the dedicated pins used by the four mandatory JTAG signals for each of the PZ5032C package types.

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.
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. PZ5032C JTAG Pinout by Package Type

DEVICE	(PIN NUMBER / MACROCELL #)			
	TCK	TMS	TDI	TDO
PZ5032C				
44-pin PLCC	32/B8	13/A8	7/A3	38/B3
44-pin TQFP	26/B8	7/A8	1/A3	32/B3

Table 5. PZ5032C Low-Level JTAG Boundary-Scan Commands

INSTRUCTION (Instruction Code) *Register Used*	DESCRIPTION
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.

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
  - Reconfigurability 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 PZ5032C 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 PZ5032C 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 PZ5032C enable this feature. The ISP commands implemented in the Philips PZ5032C 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.

Terminations

The CoolRunner™ PZ5032C CPLDs are TotalCMOS™ devices. As with other CMOS devices, it is important to consider how to properly terminate unused inputs and I/O pins when fabricating a PC board. The PZ5032C devices do not have on-chip termination circuits, so it is recommended that unused inputs and I/O pins be properly terminated. Allowing unused inputs and I/O pins to float can cause the voltage to be in the linear region of the CMOS input structures, which can increase the power consumption of the device. Philips recommends the use of 10KΩ pull-up resistors for the termination. Using pull-up resistors allows 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.

When using the JTAG/ISP functions, it is also recommended that 10KΩ pull-up resistors be used on each of the four mandatory signals. Letting these signals float can cause the voltage on TMS to come close to ground, which could cause the device to enter JTAG/ISP mode at unspecified times. See the application notes *JTAG and ISP in Philips Devices* and *Terminating Unused CoolRunner™ I/O Pins* for more information.

Table 6. Low Level ISP Commands

INSTRUCTION *(Register Used)*	INSTRUCTION CODE	DESCRIPTION
Enable *(ISP Shift Register)*	1001	Enables the Erase, Program, and Verify commands.
Erase *(ISP Shift Register)*	1010	Erases the entire EEPROM array.
Program *(ISP Shift Register)*	1011	Programs the data in the ISP Shift Register into the addressed EEPROM row.
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.

JTAG and ISP Interfacing

A number of industry-established methods exist for JTAG/ISP interfacing with CPLD's and other integrated circuits. The Philips PZ5032C supports the following methods:

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

For more details on JTAG and ISP for the PZ5032C, 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)	8		mA
TDO_I_{OH}	Output current (High)	8		mA
AC Parameters				
f_{MAX}	TCK 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[2]	–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.
2. The chip supply voltage must rise monotonically.

OPERATING RANGE

PRODUCT GRADE	TEMPERATURE	VOLTAGE
Commercial	0 to +70°C	5 ±5% V

DC ELECTRICAL CHARACTERISTICS FOR COMMERCIAL GRADE DEVICES

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

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[2]	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		1	mA
		V_{DD} = 5.25V, T_{amb} = 0°C @ 50MHz		15	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	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 disabled and unloaded. Inputs are tied to V_{DD} or ground. This parameter guaranteed by design and characterization, not testing.
2. This parameter guaranteed by design and characterization, not by test.

AC ELECTRICAL CHARACTERISTICS[1] FOR COMMERCIAL GRADE DEVICES

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

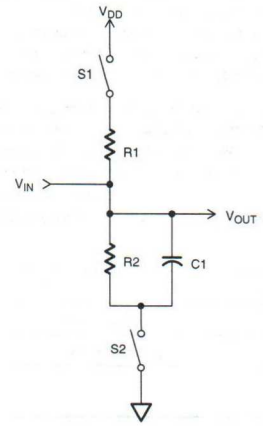
SYMBOL	PARAMETER	–6		–7		–10		UNIT
		MIN.	MAX.	MIN.	MAX.	MIN.	MAX.	
t_{PD_PAL}	Propagation delay time, input (or feedback node) to output through PAL	2	6	2	7.5	2	10	ns
t_{PD_PLA}	Propagation delay time, input (or feedback node) to output through PAL & PLA	3	8	3	10	3	12.5	ns
t_{CO}	Clock to out delay time	2	5.5	2	7	2	9	ns
t_{SU_PAL}	Setup time (from input or feedback node) through PAL	3.5		5.5		8		ns
t_{SU_PLA}	Setup time (from input or feedback node) through PAL + PLA	5.5		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] $(1/t_{SUPAL} + t_{CF})$	133		91		64		MHz
f_{MAX3}	Maximum external frequency[2] $(1/t_{SUPAL} + t_{CO})$	111		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[2, 3]		11		12.5		15	ns
t_{EA}	Input to output valid[2]		11		12.5		15	ns
t_{RP}	Input to register preset[2]		11		12.5		15	ns
t_{RR}	Input to register reset[2]		14		15.5		18	ns

NOTES:

1. Specifications measured with one output switching. See Figure 6 and Table 7 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

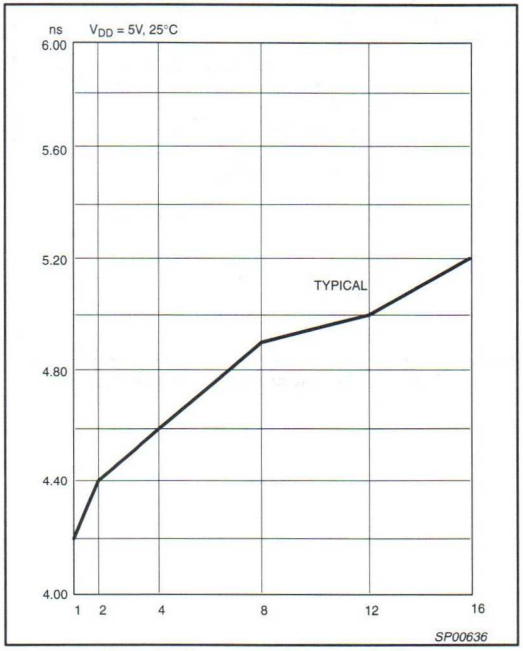
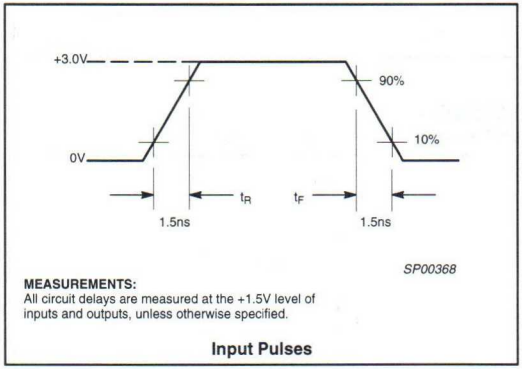


Figure 6. t_{PD_PAL} vs. Outputs switching

VOLTAGE WAVEFORM



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

Input Pulses

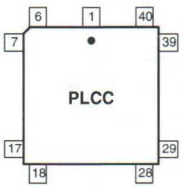
Table 7. t_{PD_PAL} vs # of Outputs switching

V_{DD} = 5.00V

# of Outputs	1	2	4	8	12	16
Typical (ns)	4.2	4.4	4.6	4.9	5.0	5.2

PIN DESCRIPTIONS

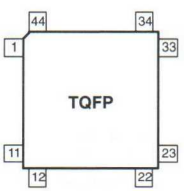
PZ5032C – 44-Pin Plastic Leaded Chip Carrier



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 (TCK)
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 (TDI)	22	GND	37	I/O–B4
8	I/O–A4	23	V_{DD}	38	I/O–B3 (TDO)
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 (TMS)	28	I/O–B11	43	IN0–CK0
14	I/O–A9	29	I/O–B10	44	IN2–gtsn
15	V_{DD}	30	GND		

SP00546

PZ5032C – 44-Pin Thin Quad Flat Package



Pin	Function	Pin	Function	Pin	Function
1	I/O–A3 (TDI)	16	GND	31	I/O–B4
2	I/O–A4	17	V_{DD}	32	I/O–B3 (TDO)
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 (TMS)	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 (TCK)	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		

SP00547

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

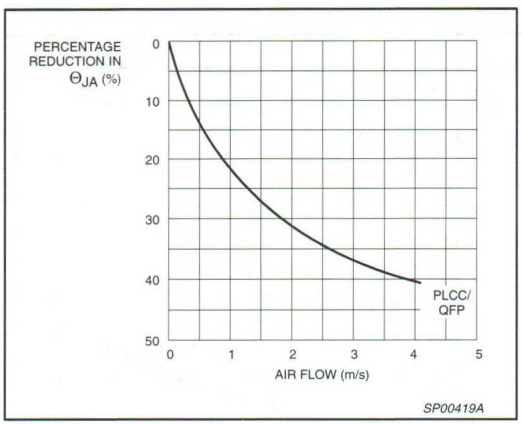


Figure 7. Average Effect of Airflow on Θ_{JA}

64 macrocell CPLD with enhanced clocking PZ5064C/PZ5064N

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
- Up to 12 clocks with programmable polarity at every macrocell
- 5 Volt, In-System Programmable (ISP) using a JTAG interface
  - On-chip supervoltage generation
  - ISP commands include: Enable, Erase, Program, Verify
  - Supported by multiple ISP programming platforms
  - 4 pin JTAG interface (TCK, TMS, TDI, TDO)
  - JTAG commands include: Bypass, Idcode
- 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
  - Up to 2 asynchronous clocks
- Programmable global 3-State pin facilitates 'bed of nails' testing without using logic resources
- Available in PLCC and TQFP packages
- Available in both Commercial and Industrial grades

Table 1. PZ5064C/PZ5064N Features

	PZ5064C/PZ5064N
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, 100-pin TQFP

DESCRIPTION

The PZ5064C/PZ5064N 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 PZ5064C/PZ5064N 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.

The Philips FZP™ CPLDs introduce the new patented 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 PZ5064C/PZ5064N CPLDs are supported by industry standard CAE tools (Cadence, Exemplar Logic, Mentor, OrCAD, 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 PZ5064C/PZ5064N CPLD is reprogrammable using industry standard device programmers from vendors such as Data I/O, BP Microsystems, SMS, and others. The PZ5064C/PZ5064N also includes an industry–standard, IEEE 1149.1, JTAG interface through which In–System Programming (ISP) and reprogramming of the device are supported.

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

ORDERING INFORMATION

ORDER CODE	DESCRIPTION	DRAWING NUMBER
PZ5064C7A44	44-pin PLCC, 7.5ns t_{PD}, Commercial temp range, 5 volt power supply, ± 5%	SOT187-2
PZ5064C10A44	44-pin PLCC, 10ns t_{PD}, Commercial temp range, 5 volt power supply, ± 5%	SOT187-2
PZ5064N10A44	44-pin PLCC, 10ns t_{PD}, Industrial temp range, 5 volt power supply, ± 10%	SOT187-2
PZ5064N12A44	44-pin PLCC, 12ns t_{PD}, Industrial temp range, 5 volt power supply, ± 10%	SOT187-2
PZ5064C7BC	44-pin TQFP, 7.5ns t_{PD}, Commercial temp range, 5 volt power supply, ± 5%	SOT376-1
PZ5064C10BC	44-pin TQFP, 10ns t_{PD}, Commercial temp range, 5 volt power supply, ± 5%	SOT376-1
PZ5064N10BC	44-pin TQFP, 10ns t_{PD}, Industrial temp range, 5 volt power supply, ± 10%	SOT376-1
PZ5064N12BC	44-pin TQFP, 12ns t_{PD}, Industrial temp range, 5 volt power supply, ± 10%	SOT376-1
PZ5064C7BP	100-pin TQFP, 7.5ns t_{PD}, Commercial temp range, 5 volt power supply, ± 5%	SOT386-1
PZ5064C10BP	100-pin TQFP, 10ns t_{PD}, Commercial temp range, 5 volt power supply, ± 5%	SOT386-1
PZ5064N10BP	100-pin TQFP, 10ns t_{PD}, Industrial temp range, 5 volt power supply, ± 10%	SOT386-1
PZ5064N12BP	100-pin TQFP, 12ns t_{PD}, Industrial temp range, 5 volt power supply, ± 10%	SOT386-1

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.

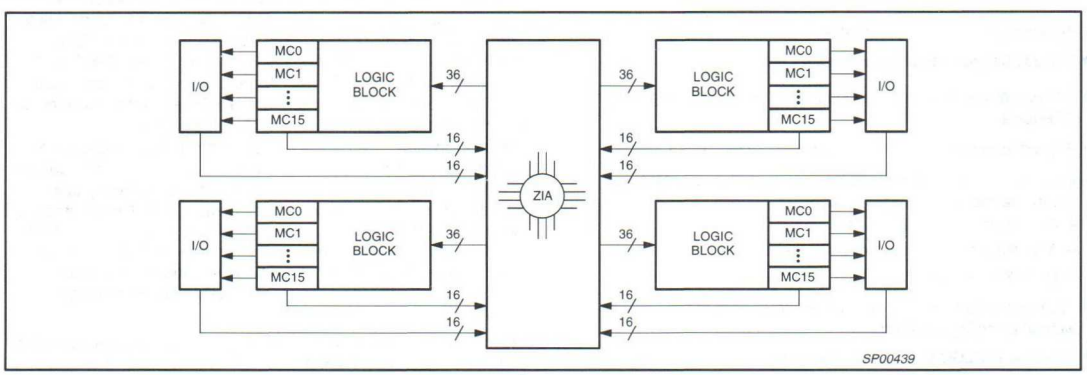


Figure 1. Philips XPLA CPLD Architecture

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. In addition, two of the control terms can be used as clock signals (see Macrocell Architecture Section for details). 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 PZ5064C/PZ5064N 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 PZ5064C/PZ5064N using 6 to 37 product terms is 9.5ns (7.5ns for the PAL + 2.0ns for the PLA).

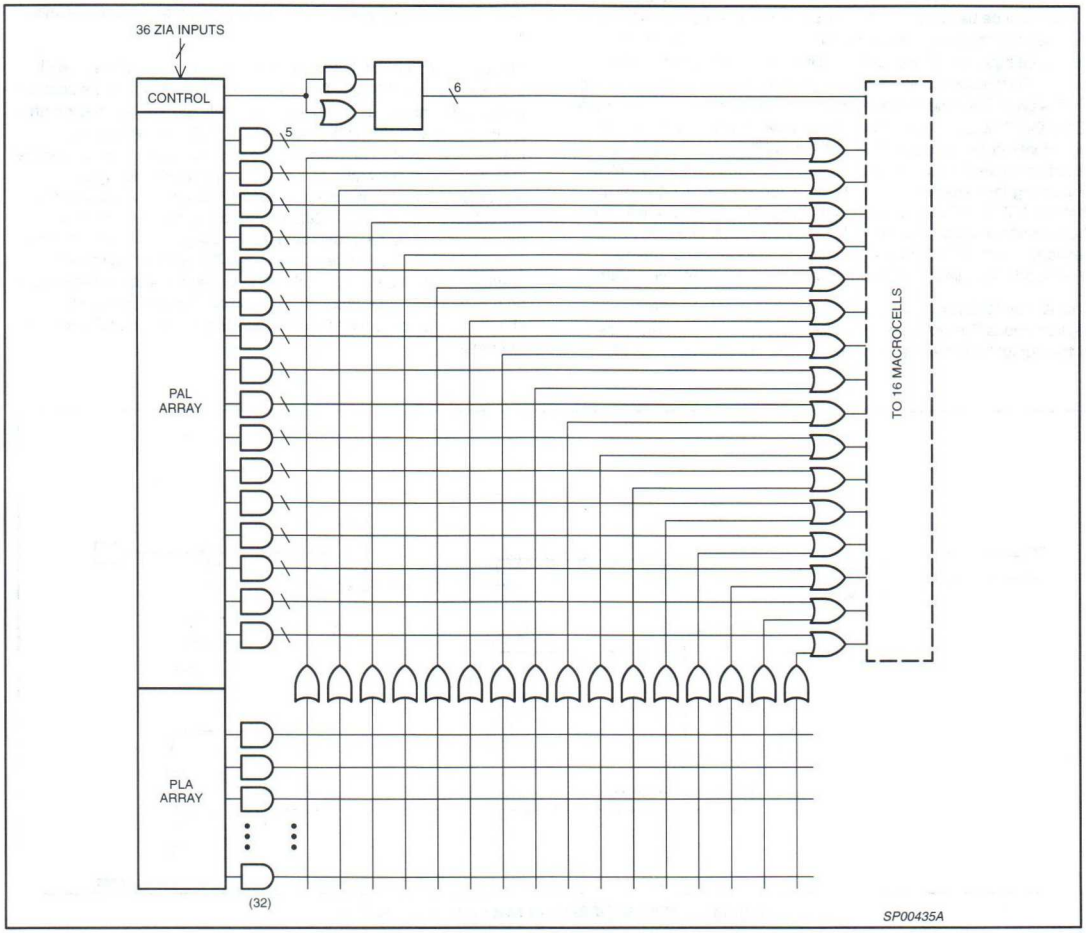


Figure 2. Philips XPLA Logic Block Architecture

Macrocell Architecture

Figure 3 shows the architecture of the macrocell used in the CoolRunner™ PZ5064C/PZ5064N. The macrocell can be configured as either a D or T type flip-flop or a combinatorial logic function. A D-type flip-flop is generally more useful for implementing state machines and data buffering while a T-type flip-flop is generally more useful in implementing counters. Each of these flip-flops can be clocked from any one of six sources. Four of the clock sources (CLK0, CLK1, CLK2, CLK3) are connected to low-skew, device-wide clock networks designed to preserve the integrity of the clock signal by reducing skew between rising and falling edges. Clock 0 (CLK0) is designated as a "synchronous" clock and must be driven by an external source. Clock 1 (CLK1), Clock 2 (CLK2), and Clock 3 (CLK3) can be used as "synchronous" clocks that are driven by an external source, or as "asynchronous" clocks that are driven by a macrocell equation. CLK0, CLK1, CLK2 and CLK3 can clock the macrocell flip-flops on either the rising edge or the falling edge of the clock signal. The other clock sources are two of the six control terms (CT2 and CT3) provided in each logic block. These clocks can be individually configured as either a PRODUCT term or SUM term equation created from the 36 signals available inside the logic block. The timing for asynchronous and control term clocks is different in that the TCO time is extended by the amount of time that it takes for the signal to propagate through the array and reach the clock network, and the TSU time is reduced. Please see the app note titled "Understanding CoolRunner Clocking Options" for more detail.

The six control terms of each logic block are used to control the asynchronous Preset/Reset of the flip-flops and the enable/disable of the output buffers in each macrocell. Control terms CT0 and CT1 are used to control the asynchronous Preset/Reset of the macrocell's flip-flop. Note that the Power-on Reset leaves all macrocells in the "zero" state when power is properly applied, and that the Preset/Reset feature for each macrocell can also be disabled. Control terms CT2 and CT3 can be used as a clock signal to the flip-flops of the macrocells, and as the Output Enable of the macrocell's output buffer. Control terms CT4 and CT5 can be used to control the Output Enable of the macrocell's output buffer. Having four dedicated Output Enable control terms ensures that the CoolRunner™ devices are PCI compliant. The output buffers can also be always enabled or always disabled. All CoolRunner™ devices also provide a Global Tri-State (GTS) pin, which, when enabled and 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 feedback 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 (see the section on Terminations in this data sheet and the application note *Terminating Unused CoolRunner™ I/O Pins*).

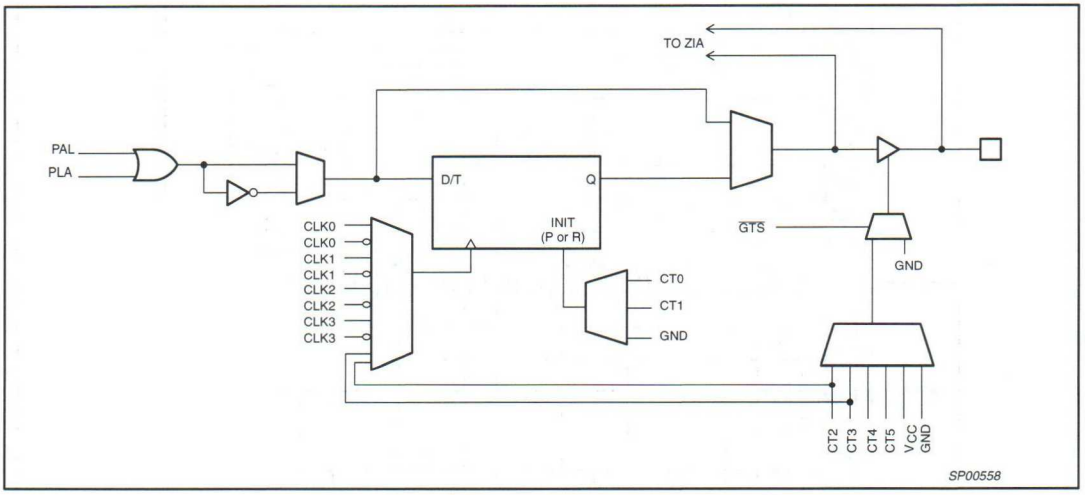


Figure 3. PZ5064C/PZ5064N Macrocell Architecture

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 PZ5064C/PZ5064N 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 PZ5064C/PZ5064N TotalCMOS™ CPLD.

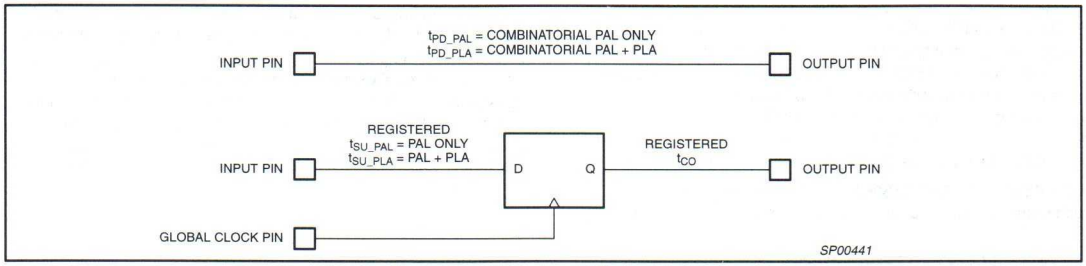


Figure 4. CoolRunner™ Timing Model

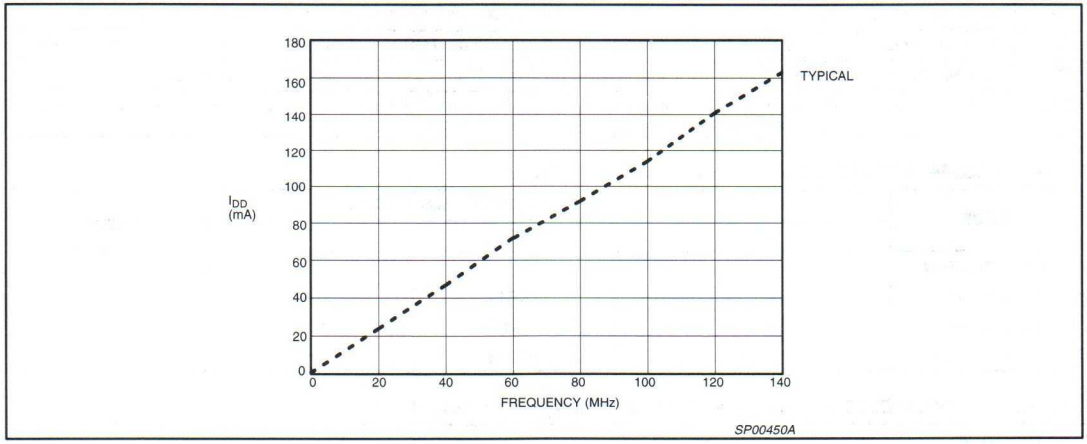


Figure 5. I_{DD} vs. Frequency @ V_{DD} = 5.0V, 25°C

Table 2. I_{DD} vs. Frequency

V_{DD} = 5.00V

FREQUENCY (MHz)	0	20	40	60	80	100	120	140
Typical I_{DD} (mA)								

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. The Philips PZ5064C/PZ5064N devices use the JTAG interface for In–System Programming/Reprogramming. Although only a subset of the full JTAG command set is implemented (see Table 5), the devices are fully capable of sitting in a JTAG scan chain.

The Philips PZ5064C/PZ5064N's JTAG interface includes a TAP Port defined by the IEEE 1149.1 JTAG Specification. As implemented in the Philips PZ5064C/PZ5064N, 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 PZ5064C/PZ5064N saves an I/O pin for general purpose use by not implementing the optional TRST* signal in the JTAG interface. Instead, the Philips PZ5064C/PZ5064N 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 TAP Port should connect to an external pull–up resistor to keep the JTAG signs from floating when they are not being used.

In the Philips PZ5064C/PZ5064N, the four mandatory JTAG pins each require a unique, dedicated pin on the device. The devices come from the factory with these I/O pins set to perform JTAG functions, but through the software, the final function of these pins can be controlled. If the end application will require the device to be reprogrammed at some future time with ISP, then the pins can be left as dedicated JTAG functions, which means they are not available for use as general purpose I/O pins. However, unlike competing CPLDs, the Philips PZ5064C/PZ5064N allow the macrocells associated with these pins to be used as buried logic when the JTAG/ISP function is enabled. This is the default state for the software, and no action is required to leave these pins enabled for the JTAG/ISP functions. If, however, JTAG/ISP is not required in the end application, the software can specify that this function be turned off and that these pins be used as general purpose I/O. Because the devices initially have the JTAG/ISP functions enabled, the JEDEC file can be downloaded into the device once, after which the JTAG/ISP pins will become general purpose I/O. This feature is good for manufacturing because the devices can be programmed during test and assembly of the end product and yet still use all of the I/O pins after the programming is done. It eliminates the need for a costly, separate programming step in the manufacturing process. Of course, if the JTAG/ISP function is never required, this feature can be turned off in the software and the device can be programmed with an industry-standard programmer, leaving the pins available for I/O functions. Table 4 defines the dedicated pins used by the four mandatory JTAG signals for each of the PZ5064C/PZ5064N package types.

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.
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. PZ5064C/PZ5064N JTAG Pinout by Package Type

DEVICE	(PIN NUMBER / MACROCELL #)			
	TCK	TMS	TDI	TDO
PZ5064C/PZ5064N				
44-pin PLCC	32/C15	13/B15	7/A8	38/D8
44-pin TQFP	26/C15	7/B15	1/A8	32/D8
100–pin TQFP	62/C15	15/B15	4/A8	73/D8

Table 5. PZ5064C/PZ5064N Low-Level JTAG Boundary-Scan Commands

INSTRUCTION (Instruction Code) *Register Used*	DESCRIPTION
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.

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
  - Reconfigurability 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 PZ5064C/PZ5064N 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 PZ5064C/PZ5064N 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 PZ5064C/PZ5064N enable this feature. The ISP commands implemented in the Philips PZ5064C/PZ5064N 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.

Terminations

The CoolRunner™ PZ5064C/PZ5064N CPLDs are TotalCMOS™ devices. As with other CMOS devices, it is important to consider how to properly terminate unused inputs and I/O pins when fabricating a PC board. Allowing unused inputs and I/O pins to float can cause the voltage to be in the linear region of the CMOS input structures, which can increase the power consumption of the device. The PZ5064C/PZ5064N CPLDs have programmable on-chip pull-down resistors on each I/O pin. These pull-downs are automatically activated by the fitter software for all unused I/O pins. Note that an I/O macrocell used as buried logic that does not have the I/O pin used for input is considered to be unused, and the pull-down resistors will be turned on. We recommend that any unused I/O pins on the PZ5064C/PZ5064N device be left unconnected.

There are no on-chip pull-down structures associated with the dedicated input pins. Philips recommends that any unused dedicated inputs be terminated with external 10kΩ pull-up resistors. These pins can be directly connected to V_{CC} or GND, but using the external pull-up resistors maintains maximum design flexibility should one of the unused dedicated inputs be needed due to future design changes.

When using the JTAG/ISP functions, it is also recommended that 10kΩ pull-up resistors be used on each of the pins associated with the four mandatory JTAG signals. Letting these signals float can cause the voltage on TMS to come close to ground, which could cause the device to enter JTAG/ISP mode at unspecified times. See the application notes *JTAG and ISP in Philips Devices* and *Terminating CoolRunner™ I/O Pins* for more information.

Table 6. Low Level ISP Commands

INSTRUCTION *(Register Used)*	INSTRUCTION CODE	DESCRIPTION
Enable *(ISP Shift Register)*	1001	Enables the Erase, Program, and Verify commands.
Erase *(ISP Shift Register)*	1010	Erases the entire EEPROM array.
Program *(ISP Shift Register)*	1011	Programs the data in the ISP Shift Register into the addressed EEPROM row.
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.

JTAG and ISP Interfacing

A number of industry-established methods exist for JTAG/ISP interfacing with CPLD's and other integrated circuits. The Philips PZ5064C/PZ5064N supports the following methods:

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

For more details on JTAG and ISP for the PZ5064C/PZ5064N, 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)	8		mA
TDO_I_{OH}	Output current (High)	8		mA
AC Parameters				
f_{MAX}	TCK 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[2]	–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.
2. The chip supply voltage must rise monotonically.

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: $0°C \le T_{amb} \le +70°C$; $4.75V \le V_{DD} \le 5.25V$

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[3]	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}[1]	Standby current	V_{DD} = 5.25V, T_{amb} = 0°C		80	µA
I_{DDD}[1, 2]	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[3]	1 pin at a time for no longer than 1 second	–50	–200	mA
C_{IN}	Input pin capacitance[3]	T_{amb} = 25°C, f = 1MHz		8	pF
C_{CLK}	Clock input capacitance[3]	T_{amb} = 25°C, f = 1MHz	5	12	pF
$C_{I/O}$	I/O pin capacitance[3]	T_{amb} = 25°C, f = 1MHz		10	pF

NOTES:
1. See Table 2, page 154 for typical values.
2. 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.
3. This parameter guaranteed by design and characterization, not by test.

AC ELECTRICAL CHARACTERISTICS[1] FOR COMMERCIAL GRADE DEVICES

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

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 (global synchronous clock from pin)	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[2, 3]		10		12	ns
t_{EA}	Input to output valid[2]		10		12	ns
t_{RP}	Input to register preset[2]		10		12.5	ns
t_{RR}	Input to register reset[2]		10		12.5	ns

NOTES:
1. Specifications measured with one output switching. See Figure 6 and Table 7 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[3]	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}[1]	Standby current	V_{DD} = 5.5V, T_{amb} = −40°C		100	µA
I_{DDD}[1, 2]	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[3]	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

NOTES:
1. See Table 2, page 154 for typical values.
2. 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.
3. This parameter guaranteed by design and characterization, not by test.

AC ELECTRICAL CHARACTERISTICS[1] FOR INDUSTRIAL GRADE DEVICES

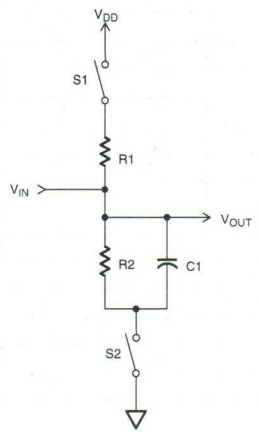
Industrial: $-40°C \le T_{amb} \le +85°C$; $4.5V \le V_{DD} \le 5.5V$

SYMBOL	PARAMETER	I10		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 (global synchronous clock from pin)	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[2, 3]		12		13	ns
t_{EA}	Input to output valid[2]		12		13	ns
t_{RP}	Input to register preset[2]		12.5		13.5	ns
t_{RR}	Input to register reset[2]		12.5		13.5	ns

NOTES:
1. Specifications measured with one output switching. See Figure 6 and Table 7 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.

SP00618

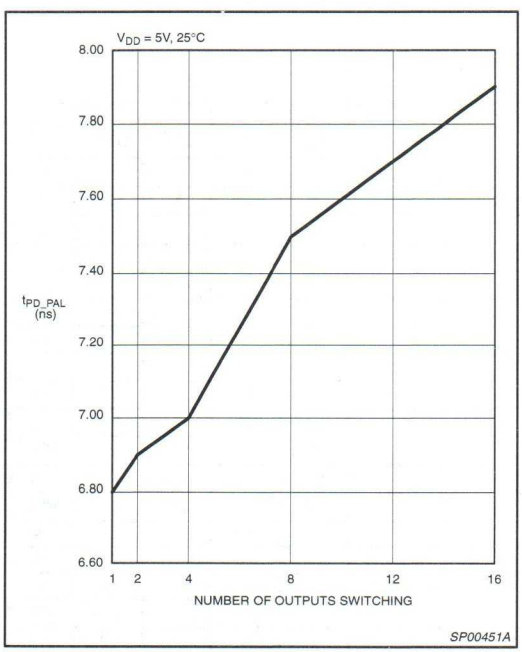
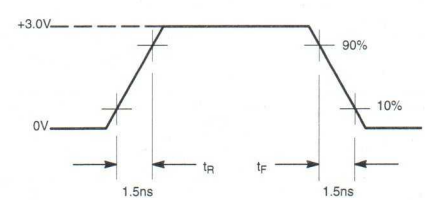


Figure 6. t_{PD_PAL} vs. Outputs Switching

VOLTAGE WAVEFORM



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

Input Pulses

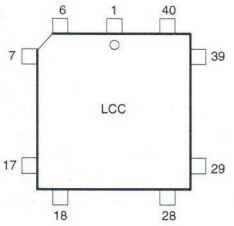
Table 7. t_{PD_PAL} vs. Number of Outputs Switching

V_{DD} = 5.00V

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

PIN DESCRIPTIONS

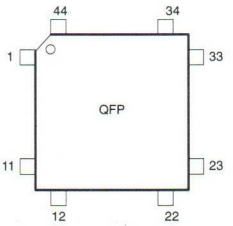
PZ5064 – 44-Pin Plastic Leaded Chip Carrier



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		

SP00554

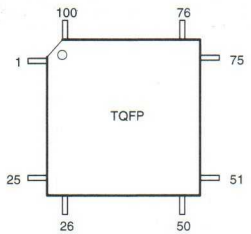
PZ5064 – 44-Pin Thin Quad Flat Package



Pin	Function	Pin	Function	Pin	Function
1	I/O-A8 (TDI)	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		

SP00555

PZ5064 – 100-Pin Thin Quad Flat Package



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

SP00556

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
100-pin TQFP	47.4°C/W

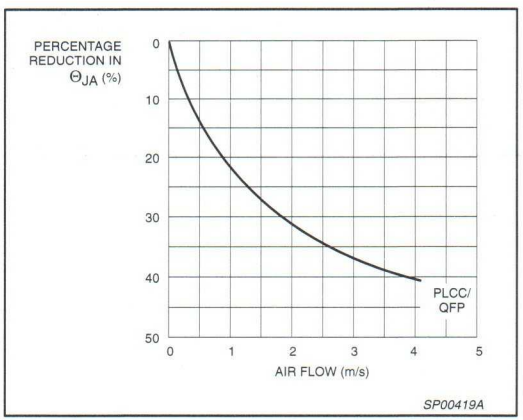


Figure 7. Average Effect of Airflow on Θ_{JA}

128 macrocell CPLD with enhanced clocking PZ5128C/PZ5128N

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
- 5 Volt, In-System Programmable (ISP) using a JTAG interface
  - On-chip supervoltage generation
  - ISP commands include: Enable, Erase, Program, Verify
  - Supported by multiple ISP programming platforms
  - 4 pin JTAG interface (TCK, TMS, TDI, TDO)
  - JTAG commands include: Bypass, Idcode
- 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
- Up to 20 clocks available
- 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
  - up to 2 asynchronous clocks
- Programmable global 3-State pin facilitates 'bed of nails' testing without using logic resources
- Available in TQFP and LQFP packages
- Available in both Commercial and Industrial grades

Table 1. PZ5128C/PZ5128N Features

	PZ5128C/PZ5128N
Usable gates	4000
Maximum inputs	100
Maximum I/Os	96
Number of macrocells	128
Propagation delay (ns)	7.5
Packages	100-pin TQFP 128-pin LQFP

DESCRIPTION

The PZ5128C/PZ5128N CPLD (Complex Programmable Logic Device) is a member of the Fast Zero Power (FZP™) family of CPLDs from Philips Semiconductors. These devices combine high speed and zero power in a 128 macrocell CPLD. With the FZP™ design technique, the PZ5128C/PZ5128N 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.

The Philips FZP™ CPLDs introduce the new patented 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 PZ5128C/PZ5128N CPLDs are supported by industry standard CAE tools (Cadence, Exemplar Logic, Mentor, OrCAD, 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 PZ5128C/PZ5128N CPLD is electrically reprogrammable using industry standard device programmers from vendors such as Data I/O, BP Microsystems, SMS, and others. The PZ5128C/PZ5128N also includes an industry-standard, IEEE 1149.1, JTAG interface through which In-System Programming (ISP) and reprogramming of the device are supported.

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

ORDERING INFORMATION

ORDER CODE	DESCRIPTION	I/O COUNT	DRAWING NUMBER
PZ5128CS7BP	100-pin TQFP, 7.5ns t_{PD}, Commercial temp range, 5 volt power supply, ± 5%	80	SOT386-1
PZ5128CS10BP	100-pin TQFP, 10ns t_{PD}, Commercial temp range, 5 volt power supply, ± 5%	80	SOT386-1
PZ5128CS12BP	100-pin TQFP, 12ns t_{PD}, Commercial temp range, 5 volt power supply, ± 5%	80	SOT386-1
PZ5128NS10BP	100-pin TQFP, 10ns t_{PD}, Industrial temp range, 5 volt power supply, ± 10%	80	SOT386-1
PZ5128NS15BP	100-pin TQFP, 15ns t_{PD}, Industrial temp range, 5 volt power supply, ± 10%	80	SOT386-1
PZ5128CS7BE	128-pin LQFP, 7.5ns t_{PD}, Commercial temp range, 5 volt power supply, ± 5%	96	SOT425-1
PZ5128CS10BE	128-pin LQFP, 10ns t_{PD}, Commercial temp range, 5 volt power supply, ± 5%	96	SOT425-1
PZ5128CS12BE	128-pin LQFP, 12ns t_{PD}, Commercial temp range, 5 volt power supply, ± 5%	96	SOT425-1
PZ5128NS10BE	128-pin LQFP, 10ns t_{PD}, Industrial temp range, 5 volt power supply, ± 10%	96	SOT425-1
PZ5128NS15BE	128-pin LQFP, 15ns t_{PD}, Industrial temp range, 5 volt power supply, ± 10%	96	SOT425-1

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.

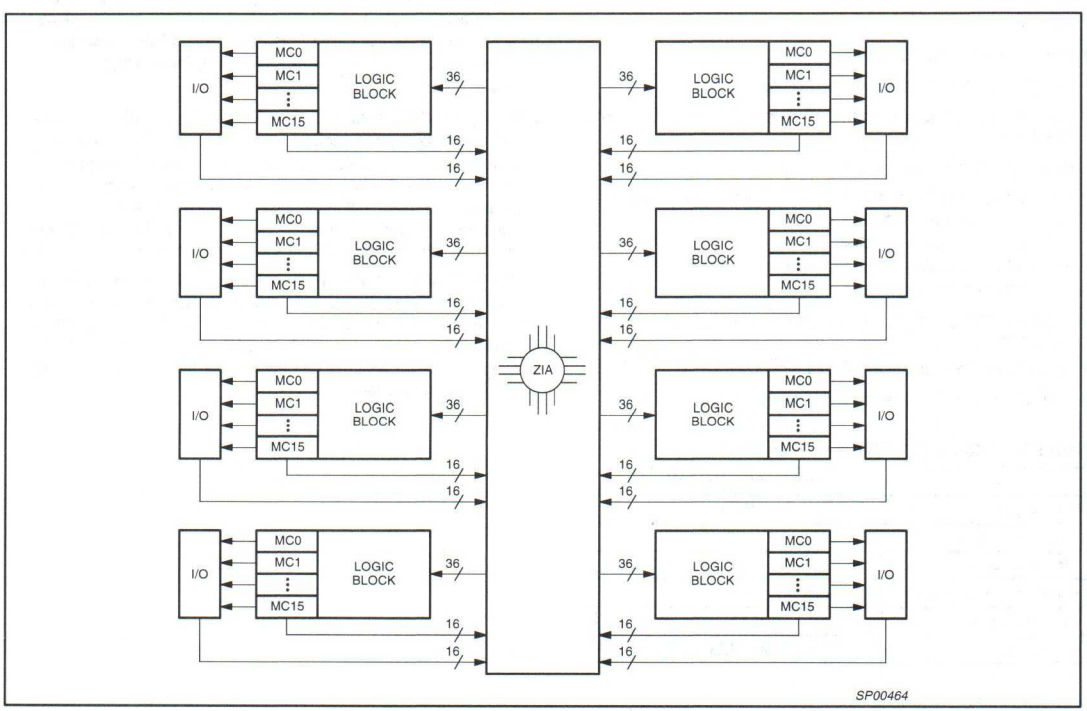


Figure 1. Philips XPLA CPLD Architecture

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. In addition, two of the control terms can be used as clock signals (see Macrocell Architecture section for details). 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 PZ5128C/PZ5128N 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 PZ5128C/PZ5128N using 6 to 37 product terms is 9.5ns (7.5ns for the PAL + 2ns for the PLA).

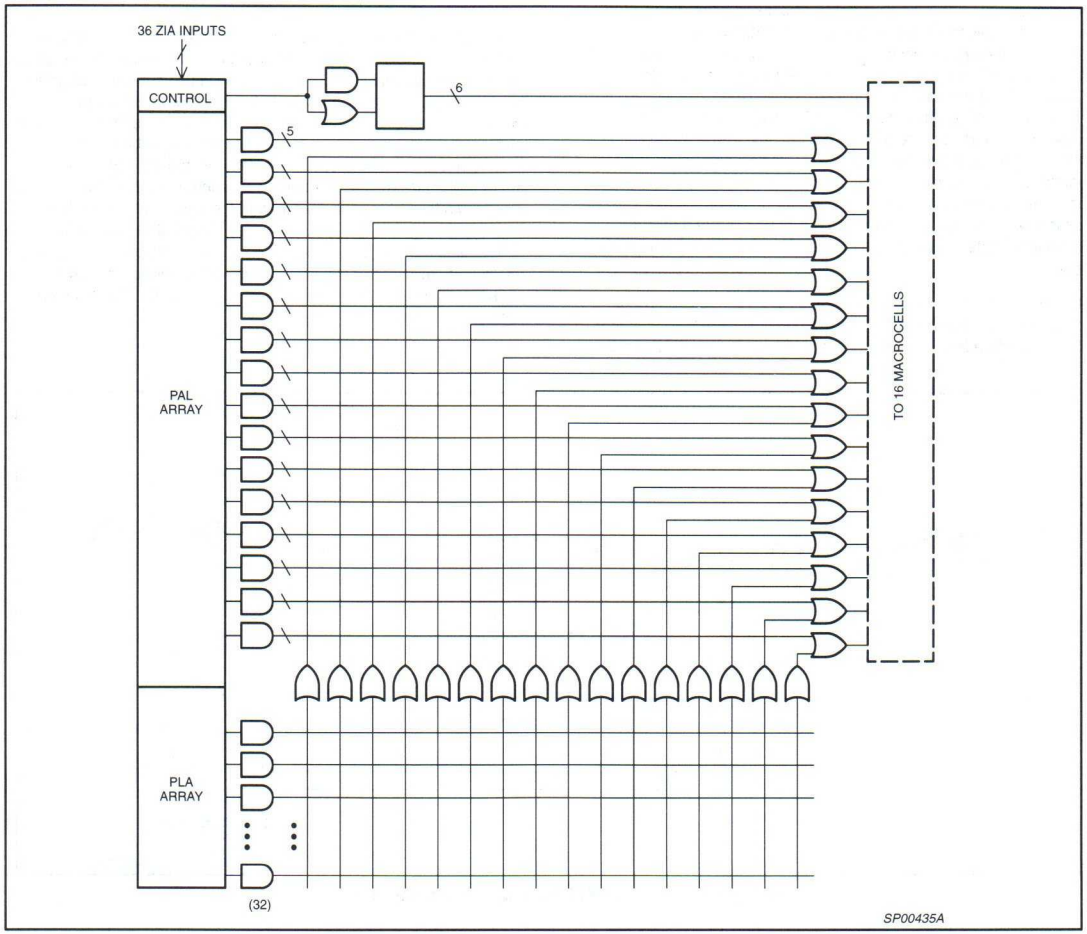


Figure 2. Philips XPLA Logic Block Architecture

Macrocell Architecture

Figure 3 shows the architecture of the macrocell used in the CoolRunner™ PZ5128C/PZ5128N. The macrocell can be configured as either a D or T type flip-flop or a combinatorial logic function. A D-type flip-flop is generally more useful for implementing state machines and data buffering while a T-type flip-flop is generally more useful in implementing counters. Each of these flip-flops can be clocked from any one of six sources. Four of the clock sources (CLK0, CLK1, CLK2, CLK3) are connected to low-skew, device-wide clock networks designed to preserve the integrity of the clock signal by reducing skew between rising and falling edges. Clock 0 (CLK0) is designated as a "synchronous" clock and must be driven by an external source. Clock 1 (CLK1), Clock 2 (CLK2), and Clock 3 (CLK3) can be used as "synchronous" clocks that are driven by an external source, or as "asynchronous" clocks that are driven by a macrocell equation. CLK0, CLK1, CLK2 and CLK3 can clock the macrocell flip-flops on either the rising edge or the falling edge of the clock signal. The other clock sources are two of the six control terms (CT2 and CT3) provided in each logic block. These clocks can be individually configured as either a PRODUCT term or SUM term equation created from the 36 signals available inside the logic block. The timing for asynchronous and control term clocks is different in that the t_{CO} time is extended by the amount of time that it takes for the signal to propagate through the array and reach the clock network, and the t_{SU} time is reduced. Please see the application note titled "Understanding CoolRunner Clocking Options" for more detail.

The six control terms of each logic block are used to control the asynchronous Preset/Reset of the flip-flops and the enable/disable of the output buffers in each macrocell. Control terms CT0 and CT1 are used to control the asynchronous Preset/Reset of the macrocell's flip-flop. Note that the Power-on Reset leaves all macrocells in the "zero" state when power is properly applied, and that the Preset/Reset feature for each macrocell can also be disabled. Control terms CT2 and CT3 can be used as a clock signal to the flip-flops of the macrocells, and as the Output Enable of the macrocell's output buffer. Control terms CT4 and CT5 can be used to control the Output Enable of the macrocell's output buffer. Having four dedicated Output Enable control terms ensures that the CoolRunner™ devices are PCI compliant. The output buffers can also be always enabled or always disabled. All CoolRunner™ devices also provide a Global Tri-State ($\overline{GTS}$) pin, which, when enabled and 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 feedback 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 (see the section on Terminations in this data sheet and the Application Note *Terminating Unused CoolRunner™ I/O Pins*).

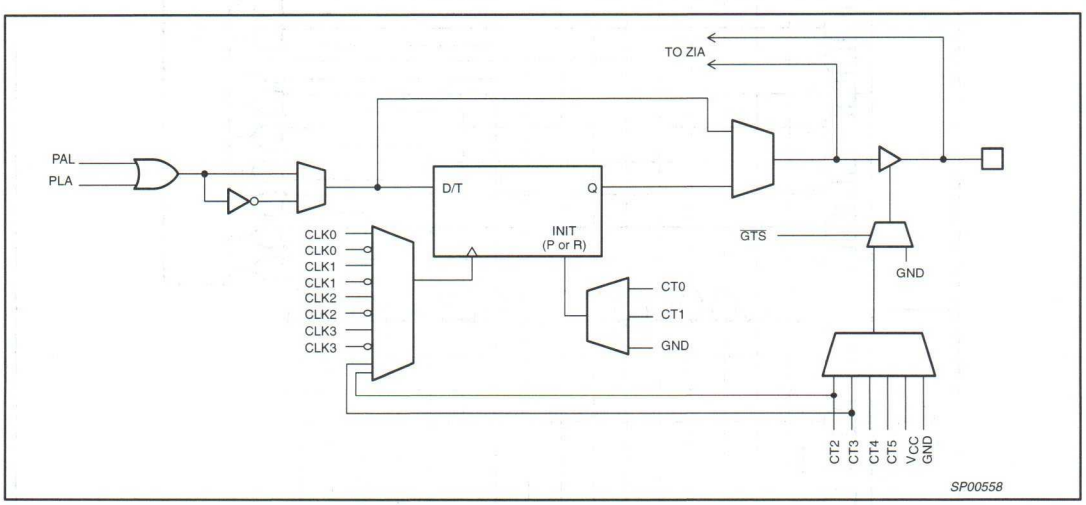


Figure 3. PZ5128C/PZ5128N Macrocell Architecture

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 PZ5128C/PZ5128N TotalCMOS™ CPLD (data taken w/eight up/down, loadable 16 bit counters@5V, 25°C).

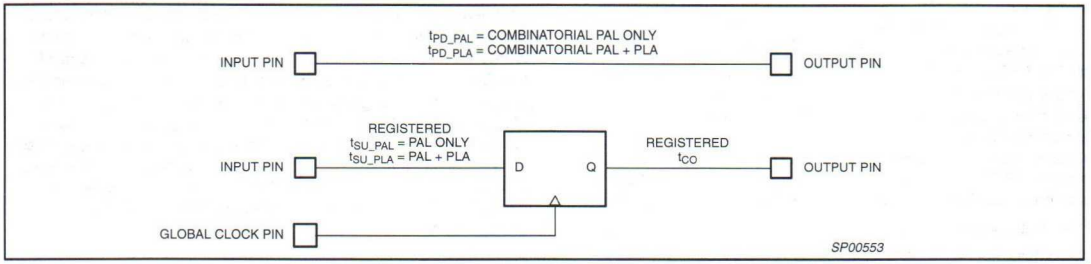


Figure 4. CoolRunner™ Timing Model

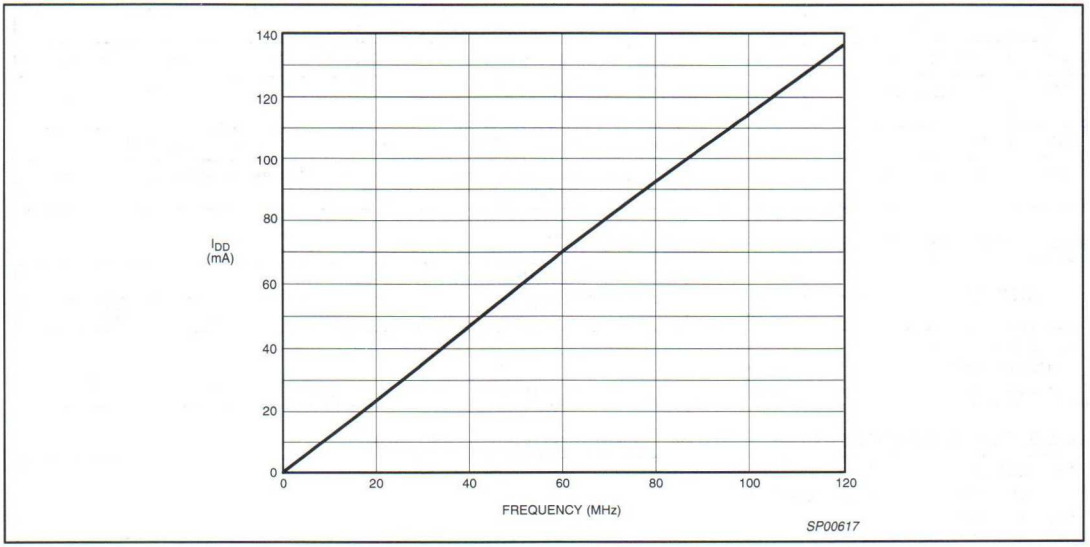


Figure 5. I_{DD} vs. Frequency @ V_{DD} = 5.0V, 25°C

Table 2. I_{DD} vs. Frequency

V_{DD} = 5.00V

FREQUENCY (MHz)	0	1	20	40	60	80	100	120
Typical I_{DD} (mA)	0.048	1.281	23.55	46.93	70.05	92.45	114.4	136.2

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. The Philips PZ5128C/PZ5128N devices use the JTAG Interface for In–System Programming/Reprogramming. Although only a subset of the full JTAG command set is implemented (see Table 5), the devices are fully capable of sitting in a JTAG scan chain.

The Philips PZ5128C/PZ5128N's JTAG interface includes a TAP Port defined by the IEEE 1149.1 JTAG Specification. As implemented in the Philips PZ5128C/PZ5128N, 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 PZ5128C/PZ5128N saves an I/O pin for general purpose use by not implementing the optional TRST* signal in the JTAG interface. Instead, the Philips PZ5128C/PZ5128N 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 TAP Port should connect to an external pull-up resistor to keep the JTAG signals from floating when they are not being used.

In the Philips PZ5128C/PZ5128N, the four mandatory JTAG pins each require a unique, dedicated pin on the device. The devices come from the factory with these I/O pins set to perform JTAG functions, but through the software, the final function of these pins can be controlled. If the end application will require the device to be reprogrammed at some future time with ISP, then the pins can be left as dedicated JTAG functions, which means they are not available for use as general purpose I/O pins. However, unlike competing CPLDs, the Philips PZ5128C/PZ5128N allow the macrocells associated with these pins to be used as buried logic when the JTAG/ISP function is enabled. This is the default state for the software, and no action is required to leave these pins enabled for the JTAG/ISP functions. If, however, JTAG/ISP is not required in the end application, the software can specify that this function be turned off and that these pins be used as general purpose I/O. Because the devices initially have the JTAG/ISP functions enabled, the JEDEC file can be downloaded into the device once, after which the JTAG/ISP pins will become general purpose I/O. This feature is good for manufacturing because the devices can be programmed during test and assembly of the end product and yet still use all of the I/O pins after the programming is done. It eliminates the need for a costly, separate programming step in the manufacturing process. Of course, if the JTAG/ISP function is never required, this feature can be turned off in the software and the device can be programmed with an industry-standard programmer, leaving the pins available for I/O functions. Table 4 defines the dedicated pins used by the four mandatory JTAG signals for each of the PZ5128C/PZ5128N package types.

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.
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. PZ5128C/PZ5128N JTAG Pinout by Package Type

DEVICE	(PIN NUMBER / MACROCELL #)			
	TCK	TMS	TDI	TDO
PZ5128C/PZ5128N				
100-pin TQFP	62/F15	15/C15	4/B15	73/G15
128-pin LQFP	82/F15	21/C15	8/B15	95/G15

Table 5. PZ5128C/PZ5128N Low-Level JTAG Boundary-Scan Commands

INSTRUCTION (Instruction Code) *Register Used*	DESCRIPTION
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.

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
  - Reconfigurability 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 PZ5128C/PZ5128N 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 PZ5128C/PZ5128N 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 PZ5128C/PZ5128N enable this feature. The ISP commands implemented in the Philips PZ5128C/PZ5128N 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.

Terminations

The CoolRunner™ PZ5128C/PZ5128N CPLDs are TotalCMOS™ devices. As with other CMOS devices, it is important to consider how to properly terminate unused inputs and I/O pins when fabricating a PC board. Allowing unused inputs and I/O pins to float can cause the voltage to be in the linear region of the CMOS input structures, which can increase the power consumption of the device. The PZ5128C/PZ5128N CPLDs have programmable on-chip pull-down resistors on each I/O pin. These pull-downs are automatically activated by the fitter software for all unused I/O pins. Note that an I/O macrocell used as buried logic that does not have the I/O pin used for input is considered to be unused, and the pull-down resistors will be turned on. We recommend that any unused I/O pins on the PZ5128C/PZ5128N device be left unconnected.

There are no on-chip pull-down structures associated with the dedicated input pins. Philips recommends that any unused dedicated inputs be terminated with external 10kΩ pull-up resistors. These pins can be directly connected to V_{CC} or GND, but using the external pull-up resistors maintains maximum design flexibility should one of the unused dedicated inputs be needed due to future design changes.

When using the JTAG/ISP functions, it is also recommended that 10kΩ pull-up resistors be used on each of the pins associated with the four mandatory JTAG signals. Letting these signals float can cause the voltage on TMS to come close to ground, which could cause the device to enter JTAG/ISP mode at unspecified times. See the application notes *JTAG and ISP in Philips Devices* and *Terminating CoolRunner™ I/O Pins* for more information.

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 the user.

JTAG and ISP Interfacing

A number of industry-established methods exist for JTAG/ISP interfacing with CPLD's and other integrated circuits. The Philips PZ5128C/PZ5128N supports the following methods:

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

For more details on JTAG and ISP for the PZ5128C/PZ5128N, 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}	TCK 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[2]	–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.
2. The chip supply voltage must rise monotonically.

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: $0°C \le T_{amb} \le +70°C$; $4.75V \le V_{DD} \le 5.25V$

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}[1]	Standby current	V_{DD} = 5.25V, T_{amb} = 0°C		100	µA
I_{DDD}[1, 2]	Dynamic current	V_{DD} = 5.25V, T_{amb} = 0°C @ 1MHz		3	mA
		V_{DD} = 5.25V, T_{amb} = 0°C @ 50MHz		75	mA
I_{OS}	Short circuit output current[3]	1 pin at a time for no longer than 1 second	–50	–200	mA
C_{IN}	Input pin capacitance[3]	T_{amb} = 25°C, f = 1MHz		8	pF
C_{CLK}	Clock input capacitance[3]	T_{amb} = 25°C, f = 1MHz	5	12	pF
$C_{I/O}$	I/O pin capacitance[3]	T_{amb} = 25°C, f = 1MHz		10	pF

NOTES:
1. See Table 2 on page 168 for typical values.
2. This parameter measured with a 16-bit, loadable up/down counter loaded into every logic block, with all outputs disabled and unloaded. Inputs are tied to V_{DD} or ground. This parameter guaranteed by design and characterization, not testing.
3. Typical values, not tested.

AC ELECTRICAL CHARACTERISTICS[1] FOR COMMERCIAL GRADE DEVICES

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

SYMBOL	PARAMETER	7		10		12		UNIT
		MIN.	MAX.	MIN.	MAX.	MIN.	MAX.	
t_{PD_PAL}	Propagation delay time, input (or feedback node) to output through PAL	2	7.5	2	10	2	12	ns
t_{PD_PLA}	Propagation delay time, input (or feedback node) to output through PAL & PLA	3	9.5	3	12	3	14.5	ns
t_{CO}	Clock to out (global synchronous clock from pin)	2	5.5	2	7	2	8	ns
t_{SU_PAL}	Setup time (from input or feedback node) through PAL	4.5		7		8		ns
t_{SU_PLA}	Setup time (from input or feedback node) through PAL + PLA	6.5		9		10.5		ns
t_H	Hold time		0		0		0	ns
t_{CH}	Clock High time	3		4		4		ns
t_{CL}	Clock Low time	3		4		4		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		125		MHz
f_{MAX2}	Maximum internal frequency[2] $1/(t_{SUPAL} + t_{CF})$	111		80		69		MHz
f_{MAX3}	Maximum external frequency[2] $1/(t_{SUPAL} + t_{CO})$	95		71		63		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	2	6	2	8.5	2	10.5	ns
t_{PDF_PLA}	Input (or feedback node) to internal feedback node delay time through PAL+PLA	3	8	3	10.5	3	13	ns
t_{CF}	Clock to internal feedback node delay time		4		5.5		6.5	ns
t_{INIT}	Delay from valid V_{DD} to valid reset		50		50		50	µs
t_{ER}	Input to output disable[2, 3]		9		12		15	ns
t_{EA}	Input to output valid[2]		9		12		15	ns
t_{RP}	Input to register preset[2]		11		12.5		15	ns
t_{RR}	Input to register reset[2]		11		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}[1]	Standby current	V_{DD} = 5.5V, T_{amb} = –40°C		125	µA
I_{DDD}[1, 2]	Dynamic current	V_{DD} = 5.5V, T_{amb} = –40°C @ 1MHz		4	mA
		V_{DD} = 5.5V, T_{amb} = –40°C @ 50MHz		80	mA
I_{OS}	Short circuit output current[3]	1 pin at a time for no longer than 1 second	–50	–230	mA
C_{IN}	Input pin capacitance[3]	T_{amb} = 25°C, f = 1MHz		8	pF
C_{CLK}	Clock input capacitance[3]	T_{amb} = 25°C, f = 1MHz	5	12	pF
$C_{I/O}$	I/O pin capacitance[3]	T_{amb} = 25°C, f = 1MHz		10	pF

NOTES:
1. See Table 2 on page 168 for typical values.
2. This parameter measured with a 16-bit, loadable up/down counter loaded into every logic block, with all outputs disabled and unloaded. Inputs are tied to V_{DD} or ground. This parameter guaranteed by design and characterization, not testing.
3. Typical values, not tested.

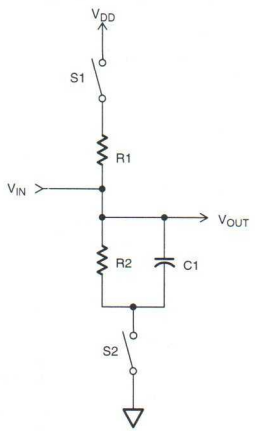
AC ELECTRICAL CHARACTERISTICS[1] FOR INDUSTRIAL GRADE DEVICES

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

SYMBOL	PARAMETER	10		15		UNIT
		MIN.	MAX.	MIN.	MAX.	
t_{PD_PAL}	Propagation delay time, input (or feedback node) to output through PAL	2	10	2	15	ns
t_{PD_PLA}	Propagation delay time, input (or feedback node) to output through PAL & PLA	3	12	3	17.5	ns
t_{CO}	Clock to out (global synchronous clock from pin)	2	7	2	8	ns
t_{SU_PAL}	Setup time (from input or feedback node) through PAL	8		8		ns
t_{SU_PLA}	Setup time (from input or feedback node) through PAL + PLA	10		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] $1/(t_{SUPAL} + t_{CF})$	71		69		MHz
f_{MAX3}	Maximum external frequency[2] $1/(t_{SUPAL} + t_{CO})$	66		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	13.5	ns
t_{PDF_PLA}	Input (or feedback node) to internal feedback node delay time through PAL+PLA	3	10.5	3	16	ns
t_{CF}	Clock to internal feedback node delay time		6		6.5	ns
t_{INIT}	Delay from valid V_{DD} to valid reset		50		50	µs
t_{ER}	Input to output disable[2, 3]		15		15	ns
t_{EA}	Input to output valid[2]		15		15	ns
t_{RP}	Input to register preset[2]		15		17	ns
t_{RR}	Input to register reset[2]		15		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.

SWITCHING CHARACTERISTICS



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.

SP00618

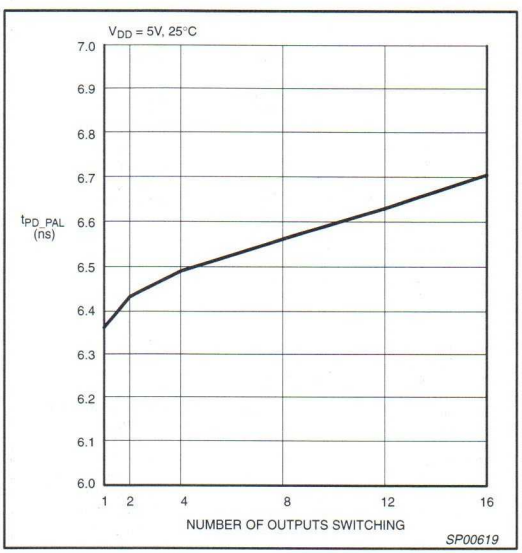
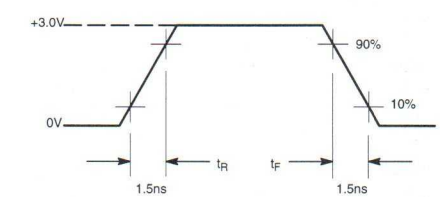


Figure 6. t_{PD_PAL} vs. Outputs Switching

VOLTAGE WAVEFORM



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

Input Pulses

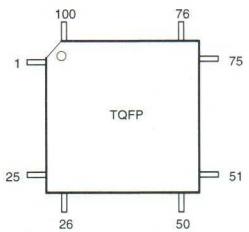
Table 8. t_{PD_PAL} vs. Number of Outputs Switching

V_{DD} = 5.00V, 25°C

NUMBER OF OUTPUTS	1	2	4	8	12	16
Typical (ns)	6.362	6.432	6.49	6.562	6.63	6.705

PIN DESCRIPTIONS

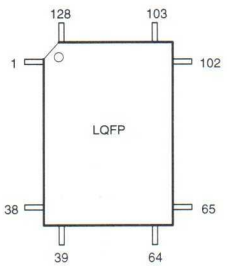
100-Pin Thin Quad Flat Package



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		

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		

SP00469A

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}
100-pin TQFP	47.4 °C/W
128-pin LQFP	45.0 °C/W

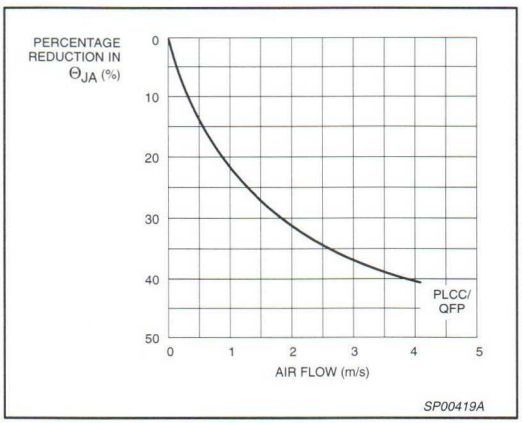


Figure 7. Average Effect of Airflow on Θ_{JA}

Section 5
XPLA2 Family

CONTENTS

PZ3320C/PZ3320N	320 macrocell SRAM CPLD	219
PZ3960C/PZ3960N	960 macrocell SRAM CPLD	249

320 macrocell SRAM CPLD PZ3320C/PZ3320N

FEATURES

- 320 macrocell SRAM based CPLD
- Multiple power-up configuration modes
  - Master serial
  - Slave serial
  - Master parallel-up
  - Master parallel-down
  - Slave parallel
  - Synchronous peripheral
  - Other modes available, contact Philips at 1–888–CoolPLD
- Configuration times of under 1.0 seconds
- IEEE 1149.1 compliant JTAG testing capability
  - 5 pin JTAG interface
  - IEEE 1149.1 TAP controller
- 3.3 volt device
- 5 V tolerant I/O
- Innovative XPLA2 Architecture combines extreme flexibility and high speeds
- 8 synchronous clock networks with programmable polarity at every macrocell
- Up to 32 asynchronous clocks support complex clocking needs
- Innovative XOR structure at every macrocell provides excellent logic reduction capability
- Logic expandable to 36 product terms on a single macrocell
- PCI compliant (except for clamp diode to V_{CC} rail due to 5 V tolerance)
- Advanced 0.35µ SRAM process
- Design entry and verification using industry standard and Philips CAE tools
- Innovative Control Term structure provides either sum terms of product terms in each logic block for:
  - 3-State buffer control
  - Asynchronous macrocell register reset/preset
- Global 3-State pin facilitates 'bed of nails' testing without sacrificing logic resources
- Programmable slew rate control
- Small form factor packages with high I/O counts
- Available in commercial and industrial temperature ranges

Table 1. PZ3320C/PZ3320N Features

	PZ3320C/PZ3320N
Usable gates	10,000
Maximum inputs	192
Maximum I/Os	192
Number of macrocells	320
Propagation delay (ns)	7.5
Packages	160 pin LQFP 256 pin PBGA

DESCRIPTION

The PZ3320 device is a member of the CoolRunner™ family of high-density SRAM-based CPLDs (Complex Programmable Logic Device) from Philips Semiconductors. This device combines high speed and deterministic pin-to-pin timing with high density. The PZ3320 uses the patented Fast Zero Power (FZP) design technique that combines high speed and low power for the first time ever in a CPLD. FZP allows the PZ3320 to have true pin-to-pin timing delays of 7.5ns, and standby currents of 100 microamps 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 since the bipolar era) with a cascaded chain of pure CMOS gates, both standby and dynamic power are dramatically reduced when compared to other CPLDs. The FZP design technique is also what allows Philips to offer a true CPLD architecture in a high density device.

The Philips PZ3320C/PZ3320N devices use the new patent-pending XPLA2™ (eXtended Programmable Logic Array) architecture. This architecture combines the best features of both PAL- and PLA-type logic structures to deliver high speed and flexible logic allocation that results in superior ability to make design changes with fixed pinouts. The XPLA2™ architecture is constructed from 80 macrocell Fast Modules that are connected together by an interconnect array. Within each Fast Module are four Logic Blocks of 20 macrocells each. Each Logic Block contains a PAL structure with four dedicated product terms for each macrocell. In addition, each Logic Block has 32 additional product terms in a PLA structure that can be shared through a fully programmable OR array to any of the 20 macrocells. This combination efficiently allocates logic throughout the Logic Block, which increases device density and allows for design changes without re-defining the pinout or changing the system timing. The PZ3320 offers pin-to-pin propagation delays of 7.5ns through the PAL array of a Fast Module; and if the PLA array is used, an additional 1.5ns is added to the delay, no matter how many PLA product terms are used. If the interconnect array between Fast Modules is used, there is a second fixed addition to the propagation delay of 4.0ns. This means that the worst case pin-to-pin propagation delay within a fast module is 7.5 + 1.5 = 9.0 ns, and the delay from any pin to any other pin across the entire chip is 7.5 + 4.0 = 11.5ns if only the PAL array is used, and 7.5 + 1.5 + 4.0 = 13.0ns if the PLA array is used. This deterministic timing allows you to establish system timing before the logic design is even started.

Each macrocell also has a two input XOR gate with the dedicated PAL product terms on one input and the PLA product terms on the other input. This patent-pending Versatile XOR structure allows for very efficient logic optimization compared to competing XOR structures that have only one product term as the second input to the XOR gate. The Versatile XOR allows an 8 bit XOR function to be implemented in only 20 product terms, compared to 65 product terms for the traditional XOR approach.

The PZ3320 is SRAM-based, which means that it is configured at power up by one of many different methods. The device may be reconfigured any number of times. See the configuration section of this data sheet for more information. The device supports the full JTAG specification (IEEE 1149.1) through an industry standard JTAG interface.

Software support for the PZ3320 is through industry standard CAE tools (Cadence, Mentor, Synopsys, Synario, Viewlogic, MINC, Exemplar Logic, and Orcad) as well as Philips' own XPLA Designer. Entry methods include both text (ABEL, PHDL, VHDL, Verilog) and/or schematic. Design verification uses industry standard simulators for functional and timing simulation, and development tools are supported on personal computer, SPARC, and HP Workstation platforms. Device fitting uses either MINC or Philips Semiconductors developed tools.

ORDERING INFORMATION

ORDER CODE	PACKAGE, PROPAGATION DELAY	DESCRIPTION	DRAWING NUMBER
PZ3320C7xx	160-pin LQFP, 7.5 ns t_{PD}	Commercial temp. range, 3.3 volt power supply ±10%	
PZ3320C7yy	256-pin PBGA, 7.5 ns t_{PD}	Commercial temp. range, 3.3 volt power supply ±10%	
PZ3320N8xx	160-pin LQFP, 7.5 ns t_{PD}	Industrial temp. range, 3.3 volt power supply ±10%	
PZ3320N8yy	256-pin PBGA, 7.5 ns t_{PD}	Industrial temp. range, 3.3 volt power supply ±10%	

XPLA2 ARCHITECTURE

Figure 1 shows a high level block diagram of the PZ3320 implementing the XPLA2 architecture. The XPLA2 architecture is a multi-level, modular hierarchy that consists of Fast Modules interconnected by a Global Zero Power Interconnect Array (GZIA). The GZIA is a virtual crosspoint switch that connects the Fast Modules together. Each Fast Module accepts 64 bits from the GZIA and outputs 64 bits to the GZIA. Each Fast Module is essentially an 80 macrocell CPLD with four logic blocks of 20 macrocells each inside. There are eight dedicated, low-skew, global clocks for the device; and each Fast Module has access to any two of these clocks (there are additional asynchronous clocks available in the Fast Modules, see Figure 3). There are also Global 3-state (gts) and Global Reset (rstn) pins that are common to all Fast Modules. When gts is pulled high, all output buffers in the device will be disabled, causing all I/O pins to be tri-stated. When rstn is pulled low, all flip-flops of the device will be reset.

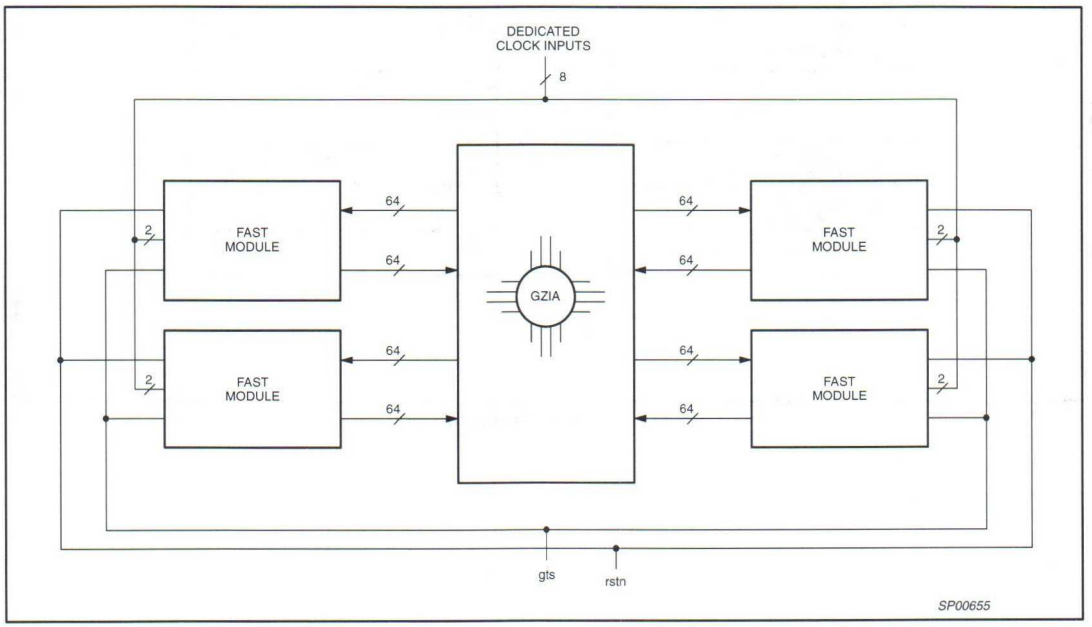


Figure 1. Philips XPLA2 CPLD Architecture

XPLA2 Fast Module

Each Fast Module consists of four Logic Blocks of 20 macrocells each. Depending on the package, either 8 or 12 of the 20 macrocells in each Logic Block are connected to I/O pins, and the remaining macrocells are used as buried nodes. These four Logic Blocks are connected together by the Local Zero Power Interconnect Array (LZIA). The LZIA is a virtual crosspoint switch that connects the Logic Blocks to each other and to the GZIA. The feedback from all 80 macrocells, input from the I/O pins, and the 64 bit input bus from the GZIA are input into the LZIA. The LZIA outputs 36 signals into each Logic Block and 64 signals into the GZIA.

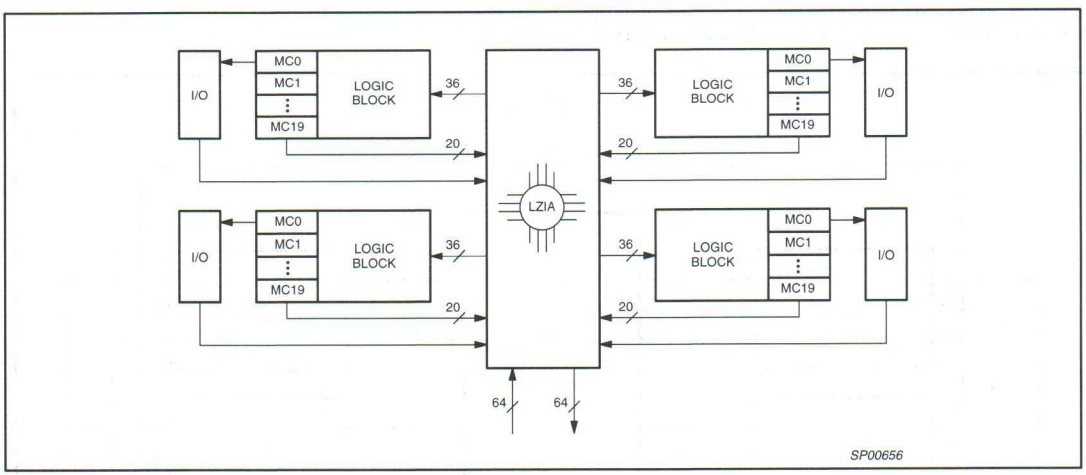


Figure 2. Philips XPLA2 Fast Module

XPLA2 Logic Block Architecture

Figure 3 illustrates the XPLA2 Logic Block architecture. Each Logic Block contains 8 control terms, a PAL array, a PLA array, and 20 macrocells. The 36 inputs from the LZIA are available to all control terms and to each product term in both the PAL and the PLA array. The 8 control terms can individually be configured as either SUM or PRODUCT terms, and are used to control the asynchronous preset and reset functions of the macrocell registers, the output enables of the 20 macrocells, and for asynchronous clocking. 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.

Each macrocell has 4 dedicated product terms from the PAL array. When additional logic is required, each macrocell takes the extra product terms from the PLA array. The PLA array consists of 32 extra product terms that are shared between the 20 macrocells of the Logic Block. The PAL product terms can be connected to the PLA product terms through either an OR gate or an XOR gate. One input to the XOR gate can be connected to all the PLA terms, which provides for extremely efficient logic synthesis. An eight bit XOR function can be implemented in only 20 product terms. Each macrocell can use the output from the OR gate or the XOR gate in either normal or inverted state.

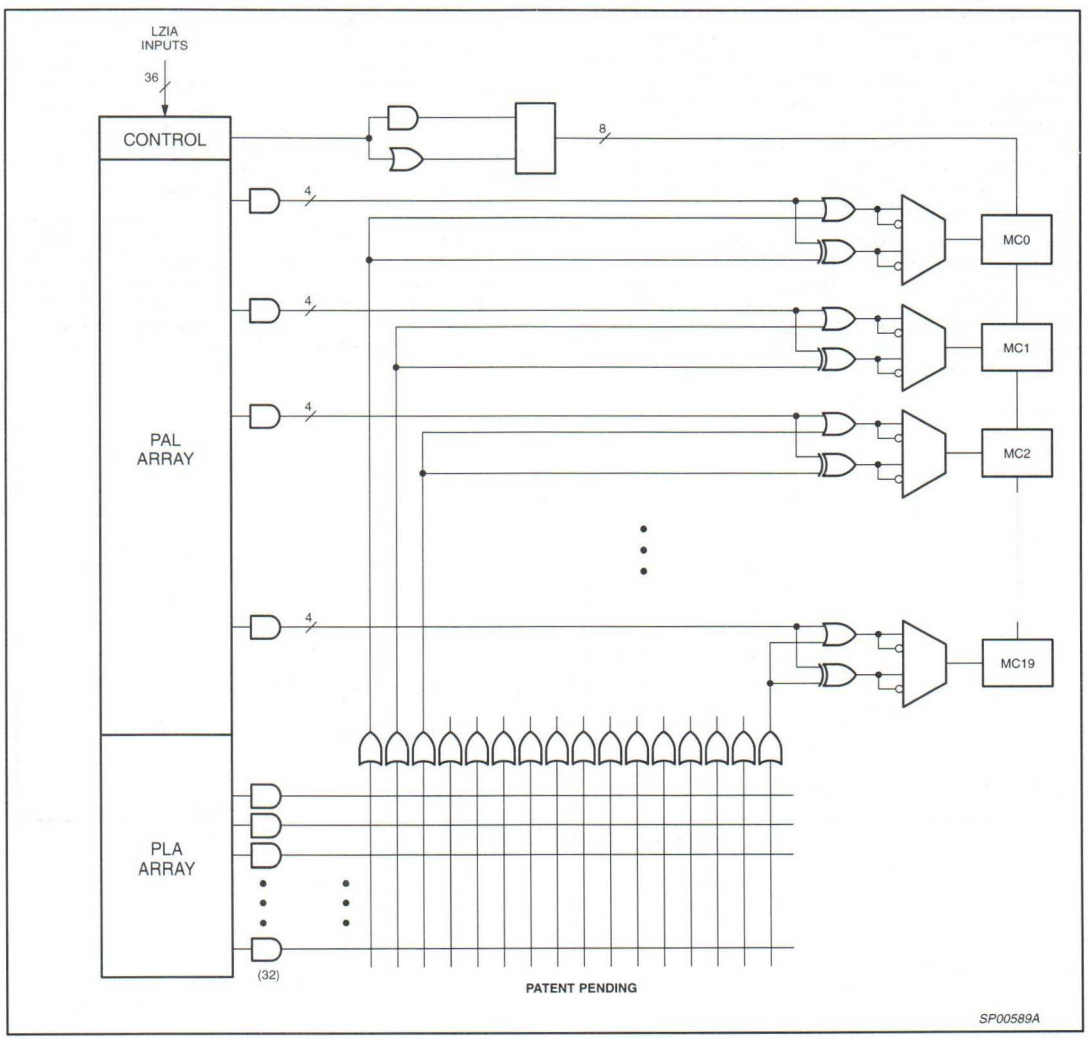


Figure 3. Philips XPLA2 Logic Block Architecture

XPLA2 Macrocell Architecture

Figure 4 shows the XPLA2 macrocell architecture used in the PZ3320. The macrocell can be configured as either a D- or T-type flip-flop or a combinatorial logic function. A D-type flip-flop is generally more useful for implementing state machines and data buffering while a T-type flip-flop is generally more useful in implementing counters. Each of these flip-flops can be clocked from any one of four sources. Two of the clock sources (CLK0 and CLK1) are from the eight dedicated, low-skew, global clock networks designed to preserve the integrity of the clock signal by reducing skew between rising and falling edges. These clocks are designated as a "synchronous" clocks and must be driven by an external source. Both CLK0 and CLK1 can clock the macrocell flip-flops on either the rising edge or the falling edge of the clock signal. The other clock sources are designated as "asynchronous" and are connected to two of the eight control terms (CT6 and CT7) provided in each logic block. These clocks can be individually configured as any PRODUCT term or SUM term equation created from the 36 signals available inside the logic block. Thus, in each Logic Block, there are up to four possible clocks; and in each Fast Module, there are up to 10 possible clocks. Throughout the entire device, there are up to 40 possible clocks—eight from the dedicated, low-skew, global clocks, and two for each of the 16 logic blocks.

The remaining six control terms of each logic block (CT0–CT5) are used to control the asynchronous preset/reset of the flip-flops and the enable/disable of the output buffers in each macrocell. Control terms CT0 and CT1 are used to control the asynchronous preset/reset of the macrocell's flip-flop. Note that the power-on reset leaves all macrocells in the "zero" state when power is properly applied, and that the preset/reset feature for each macrocell can also be disabled. Each macrocell can choose between an asynchronous reset or an asynchronous preset function, but both cannot be simultaneously used on the same register. The global rstn function can always be used, regardless of whether or not asynchronous reset or preset control terms are enabled. Control terms CT2, CT3, CT4 and CT5 are used to enable or disable the macrocell's output buffer. Having four dedicated output enable control terms ensures that the CoolRunner™ devices are PCI compliant. The output buffers can also be always enabled or always disabled. All CoolRunner™ devices also provide a Global 3-State (gts) pin, which, when pulled high, will 3-State all the outputs of the device. This pin is provided to support "In-Circuit Testing" or "Bed-of-Nails" testing used during manufacturing.

For the macrocells in the Logic Block that are associated with I/O pins, there are two feedback paths to the LZIA: one from the macrocell, and one from the I/O pin. The LZIA feedback path before the output buffer is the macrocell feedback path, while the LZIA feedback path after the output buffer is the I/O pin feedback path. When these macrocells are used as outputs, the output buffer is enabled, and either feedback path can be used to feedback the logic implemented in the macrocell. When the I/O pins are used as inputs, the output buffer of these macrocells will be 3-Stated and the input signal will be fed into the LZIA via the I/O feedback path. In this case the logic functions implemented in the buried macrocell can be fed back into the LZIA via the macrocell feedback path. For macrocells that are not associated with I/O pins, there is one feedback path to the LZIA. Logic functions implemented in these buried macrocells are fed back into the LZIA via this path. All unused inputs and I/O pins should be properly terminated. Please refer to the section on terminations.

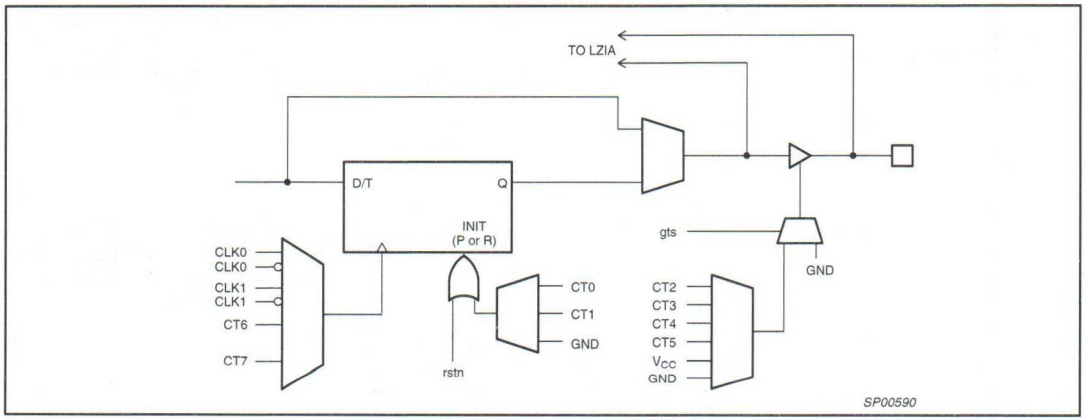


Figure 4. PZ3320 Macrocell Architecture

Simple Timing Model

Figure 5 shows the PZ3320 timing model. The PZ3320 timing model is very simple compared to the models of competing architectures. There are three main timing parameters: the pin-to-pin delay for combinatorial logic functions (t_{PD}), the input pin to register set up time (t_{SU}), and the register clock to valid output time (t_{CO}). As the model shows, timing is only dependent on whether or not you use the PLA array, and whether or not the logic function is created within a single Fast Module or uses the GZIA. The timing starts with a set time for t_{PD} and t_{SU} through the PAL array in a Fast Module, and there are fixed delays added for use of the PLA array or the GZIA. The t_{CO} timing specification never changes. For example, a combinatorial logic function of four or fewer product terms constructed from inputs within the same logic block would have a t_{PD} delay of 7.5ns. If the logic function were more than four product terms wide, the delay would be t_{PD} plus the fixed PLA delay, or 7.5 + 1.5 = 9.0ns. A function that used the PAL array and inputs from a different Fast Module would have a propagation delay of t_{PD} plus the fixed GZIA delay, or 7.5 + 4.0 = 11.5ns.

This simple timing model allows designers to determine whether or not the device will meet system timing specifications up front. In competing devices, the user is unable to determine if the design will meet system timing requirements 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, the fan-out of a signal, the varying number of X and Y routing channels used, etc. The simplicity of the PZ3320 timing model gives you pin-to-pin delay information before the design is set. Further, the timing in the PZ3320 device will not vary with place and route iterations caused by design changes. This allows the PZ3320 device to meet your timing requirements even when you make changes to the design.

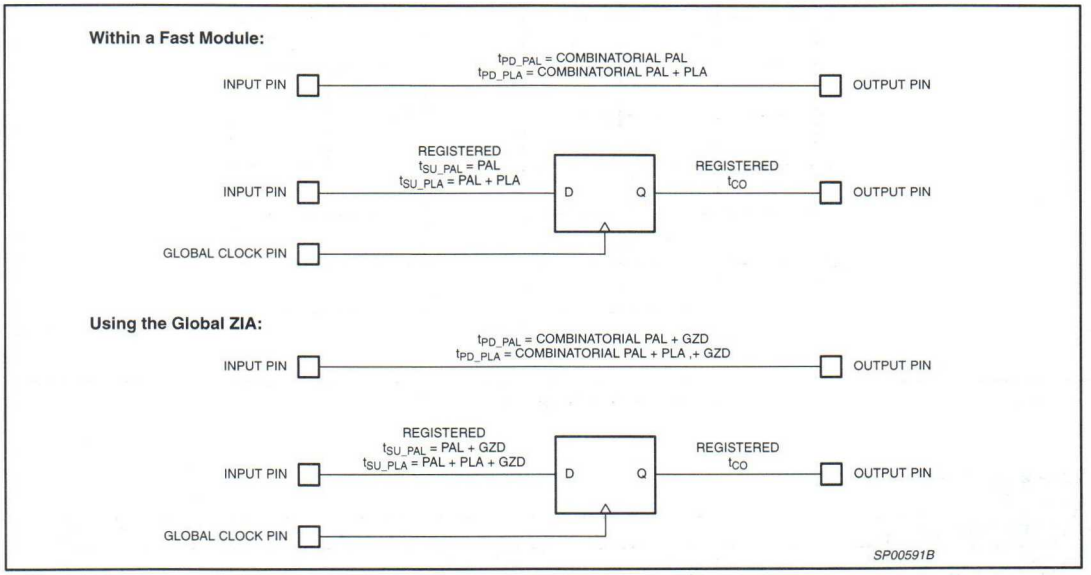


Figure 5. PZ3320 Timing Model

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 product terms 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. This also makes it possible to manufacture high density CPLDs like the PZ3320 that consume a fraction of the power of competing devices. Refer to Figure 6 and Table 2 showing the I_{DD} vs. Frequency of the PZ3320 TotalCMOS™ CPLD (data estimated with 20 16-bit counters @ 3.3V, 25°C).

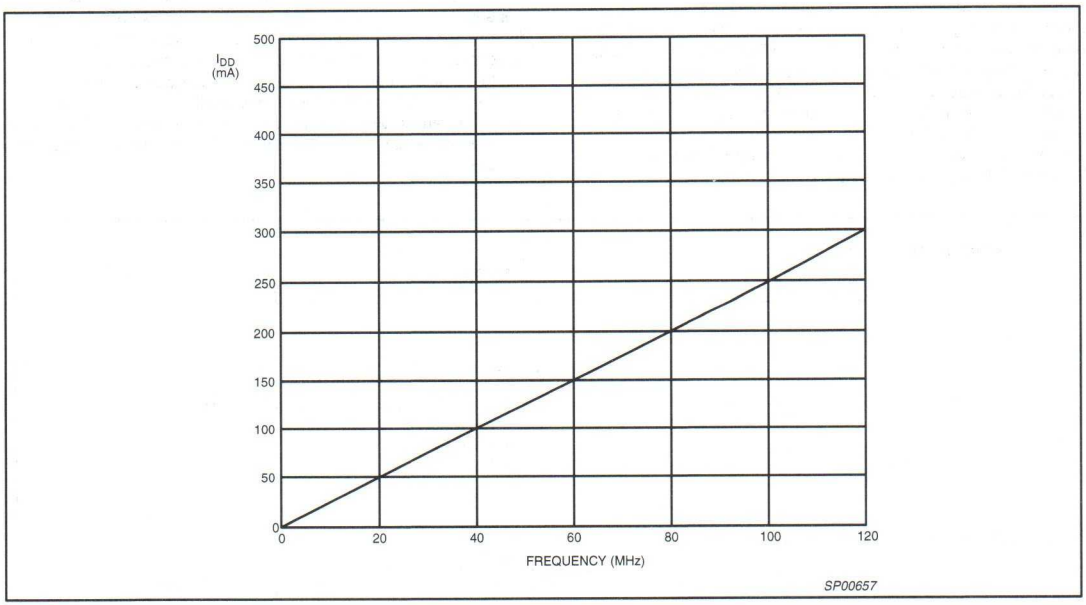


Figure 6. I_{DD} vs. Frequency @ V_{DD} = 3.3V, 25°C

Table 2. I_{DD} vs. Frequency

V_{DD} = 3.3V

FREQUENCY (MHz)	0	1	20	40	60	80	100	120
Typical I_{DD} (mA)	0.1	4.1	50	100	150	200	250	300

Terminations

The CoolRunner™ PZ3320C/PZ3320N CPLDs are TotalCMOS™ devices. As with other CMOS devices, it is important to consider how to properly terminate unused inputs and I/O pins when fabricating a PC board. Allowing unused inputs and I/O pins to float can cause the voltage to be in the linear region of the CMOS input structures, which can increase the power consumption of the device. It can also cause the voltage on a configuration pin to float to an unwanted voltage level, interrupting device operation.

The PZ3320C/PZ3320N CPLDs have programmable on-chip pull-down resistors on each I/O pin. These pull-downs are automatically activated by the fitter software for all unused I/O pins. Note that an I/O macrocell used as buried logic that does not have the I/O pin used for input is considered to be unused, and the pull-down resistors will be turned on. We recommend that any unused I/O pins on the PZ3320C/PZ3320N device be left unconnected.

There are no on-chip pull-down structures associated with dedicated pins used for device configuration or special device functions like global reset and global 3-state. Philips recommends that these pins be terminated consistent with the description given in Table 9. Philips recommends the use of weak pull-up and pull-down resistors for terminating these pins. These pins can be directly connected to V_{CC} or GND, but using the external pull-up resistors maintains maximum design flexibility.

When using the JTAG Boundary Scan functions, it is recommended that 10k pull-up resistors be used on the tdi, tdo, tck, and trstn pins. The tdo signal pin can be left floating unless it is connected to the tdi of another device. Letting these signals float can cause the voltage on tms to come close to ground, which could cause the device to enter JTAG/ISP mode at unspecified times.

CONFIGURATION INTRODUCTION

The Philips CoolRunner™ series are available in technologies which use non-volatile (EEPROM-based) and volatile (SRAM based) configuration memory. The functionality of the XPLA2 family of the CoolRunner™ series is defined by on-chip SRAM. The devices are configured in a manner similar to that of most FPGAs. This section describes the configuration of the PZ3320, and applies to all similarly configured devices to be produced by Philips.

Either the Philips or Minc fitter, XPLA Designer and PL-Designer, respectively, is used to generate a JEDEC file. The JEDEC file contains the configuration data, which is loaded into the PZ3320 configuration memory to control the PZ3320 functionality. This is done at power-up and/or with configure command. This section provides some of the trade-offs in selecting a configuration mode, and provides debug hints for configuration problems.

There are several different methods of configuring the PZ3320. The mode used is selected using the mode select pins. There are three basic configuration methods: master, slave, and peripheral. The configuration data can be transmitted to the PZ3320 serially or in parallel bytes. As a master, the PZ3320 generates the clock and control signals to strobe configuration data into the PZ3320. As a slave device, a clock is generated externally, and provided into the PZ3320's cclk pin. In the peripheral mode, the PZ3320 interfaces as a microprocessor peripheral. Table 3 lists the configuration modes.

Design Flow Overview

Figure 7 is a diagram of the steps used in configuring the PZ3320. The development system is used to generate configuration data in the JEDEC file. Using the <design>.jed file, there are two general methods of configuring the PZ3320. The utility **download** can load the configuration data from a PC or workstation hard disk into the PZ3320. This is one of the methods used on the PZ3320 evaluation board. Alternately, the PZ3320 can be loaded from non-volatile ICs such as serial or parallel EEPROMs.

Table 3. Configuration Modes

M2	M1	M0	cclk	CONFIGURATION MODE	DATA FORMAT
			Output	Master serial	Serial
			Input	Slave parallel	Parallel
			Reserved		
			Input	Synchronous peripheral	Parallel
			Output	Master parallel – up	Parallel
			Reserved		
			Output	Master parallel – down	Parallel
			Input	Slave serial	Serial

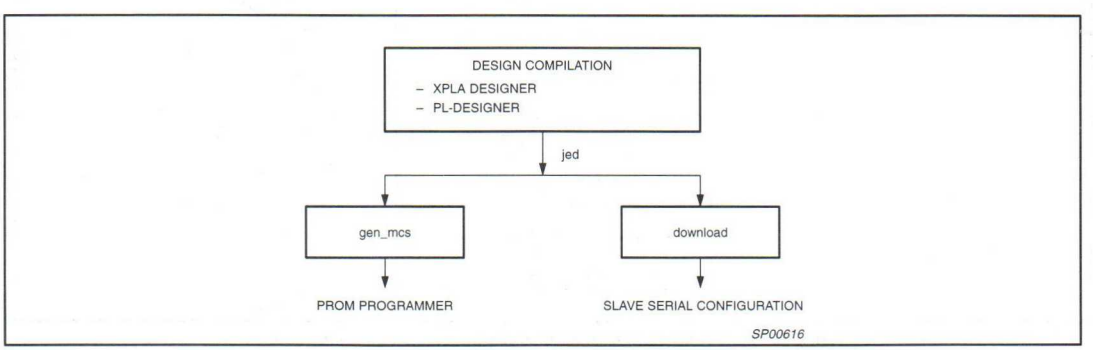


Figure 7. Design flow

PZ3320 STATES OF OPERATION

Prior to becoming operational, the PZ3320 goes through a sequence of states, including initialization, configuration, and start-up. This section discusses these three states. In the master configuration modes, the PZ3320 is the source of configuration clock (cclk). In this mode, the Initialization state is extended to ensure that, in daisy-chain operation, all daisy-chained slave devices are ready.

When configuration is initiated, a counter in the PZ3320 is set to 0 and begins to count configuration clock cycles applied to the PZ3320. As each configuration data frame is supplied to the PZ3320, it is internally assembled into data words. Each data word is loaded into the internal configuration memory. The configuration loading process is complete when the internal length count equals the loaded length count in the length count field, and the required end of configuration frame is written.

All configuration I/Os used as inputs operate with TTL-level input thresholds during configuration. All I/Os that are not used during the configuration process are 3-Stated with internal pull-downs. During configuration, registers are reset. The combinatorial logic begins to function as the PZ3320 is configured. Figure 8 shows the flow between the initialization, configuration, and start-up states. Figure 9 gives the general timing information for configuring the device.

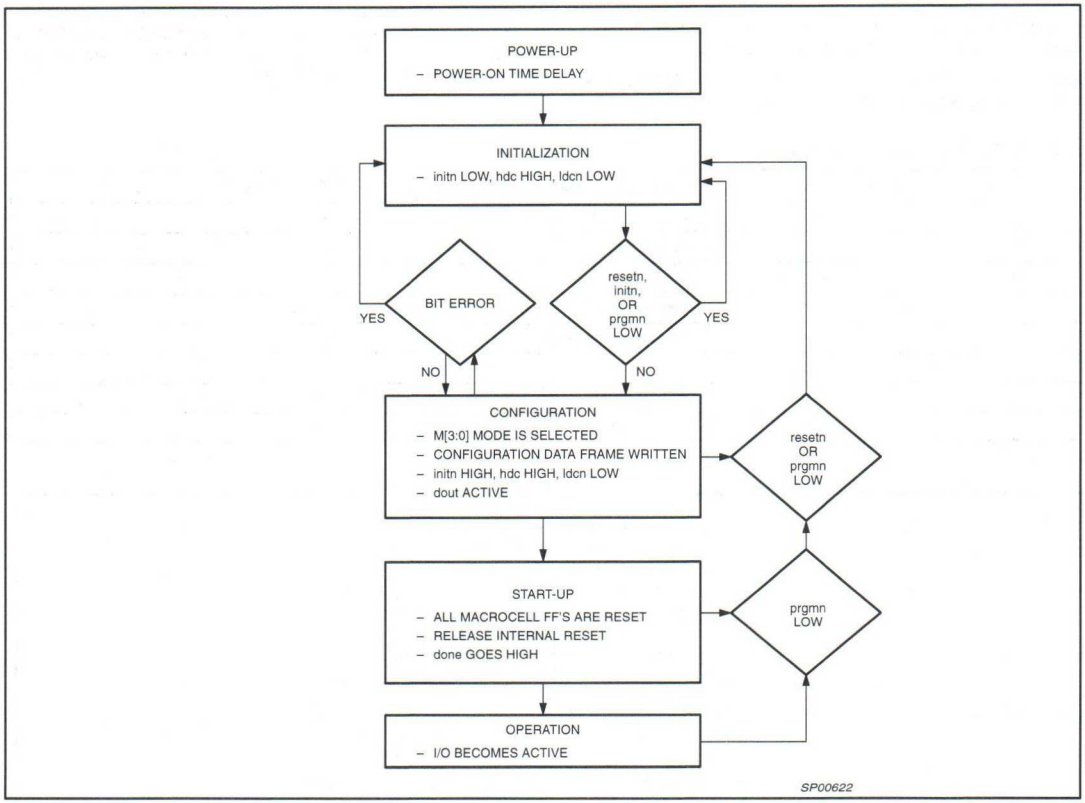


Figure 8. Flow chart of initialization, configuration, and operating states

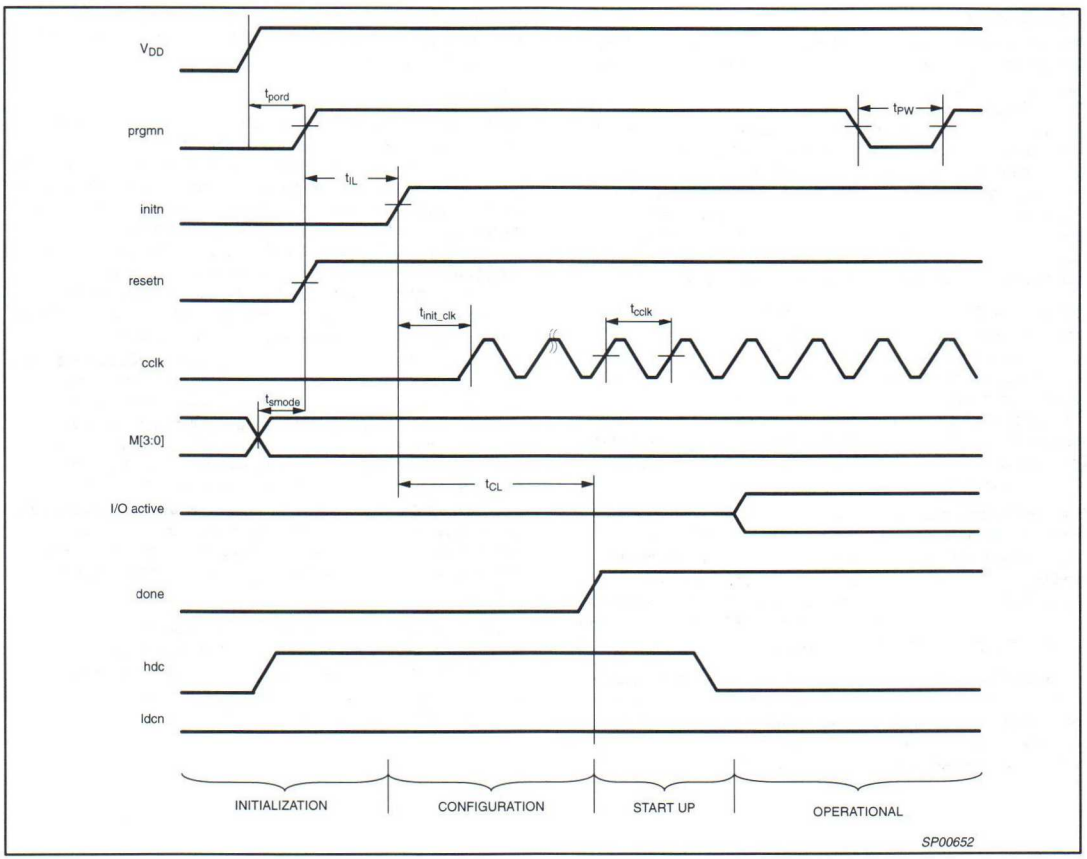


Figure 9. General configuration mode timing diagram

Initialization

Upon power-up, the device goes through an initialization process. First, an internal power-on-reset circuit is triggered when power is applied. When V_{DD} reaches the voltage at which portions of the PZ3320 begin to operate (1.5V), the configuration pins are set to be inputs or outputs based on the configuration mode, as determined by the mode select inputs M[2:0]. A time-out delay is initiated when V_{DD} reaches between 1.0V and 2.0V to allow the power supply voltage to stabilize. The initn and done outputs are low. At power-up, if the power supply does not rise from 1.0V to V_{DD} in less than 25ms, the user should delay configuration by inputting a low into prgmn, initn, or resetn until V_{DD} is greater than the recommended minimum operating voltage (2.75V for commercial devices).

When initialization is complete, the active-low initialization signal initn is released and must be pulled high by an external resistor. To synchronize the configuration of multiple PZ3320s, one or more initn pins should be wire-ANDed. If initn is held low by one or more PZ3320s or an external device (the PZ3320 remains in the initialization state), initn can be used to signal that the PZ3320s are not yet initialized. After initn goes high for two internal clock cycles, the mode select lines are sampled and the PZ3320 enters the configuration state.

The High During Configuration (hdc), Low During Configuration (ldcn), and done signals are active outputs in the PZ3320's initialization and configuration states. hdc, ldcn, and done can be used to provide control of external logic signals such as reset, bus enable, or EEPROM enable during configuration. For master parallel configuration modes, these signals provide EEPROM enable control and allow the data pins to be shared with user logic signals.

If configuration has begun, an assertion of resetn or prgmn initiates an abort, returning the PZ3320 to the initialization state. The resetn and/or prgmn pins must be pulled back high before the PZ3320 will enter the configuration state. During the start-up and operating states, only the assertion of prgmn causes a re-configuration.

During initialization and configuration, all I/O's are 3-stated and the internal weak pull-downs are active. See the section on terminations for more information.

Start-up

After configuration, the PZ3320 enters the start-up phase. This phase is the transition between the configuration and operational states. This transition occurs within three cclk cycles of the done pin going high (it is acceptable to have additional cclk cycles beyond the three required). The system design task in the start-up phase is to ensure that multi-function pins (see pin function on page 34) transition from configuration signals to user definable I/Os without inadvertently activating devices in the system or causing bus contention. The done signal goes high at the beginning of the start up phase, which allows configuration sources to be disconnected so that there is no bus contention when the I/Os become active. In addition to controlling the PZ3320 during start-up, additional start-up techniques to avoid contention include using isolation devices between the PZ3320 and other circuits in the system, re-assigning I/O locations, and keeping I/Os 3-stated until contentions are resolved. For example, Figure 10 shows how to use the global tri-state (gts) signal to avoid signal contention when the mode select pins (M3...M0) are used as I/O after configuration is finished. Holding gts high until after the mode pins are disconnected from the driving source allows pins M3 through M0 to transition from configuration pins to user definable I/O without signal contention. In this case, the I/O become active a t_{gtsr} delay after the gts pin is pulled low.

The flip-flops are reset one cycle after done goes high so that operation begins in a known state. The done outputs from multiple PZ3320s can be wire ANDed and used as an active-high ready signal, to disable PROMs with active-low enable(s), or to reset to other parts of the system (see Figure 28).

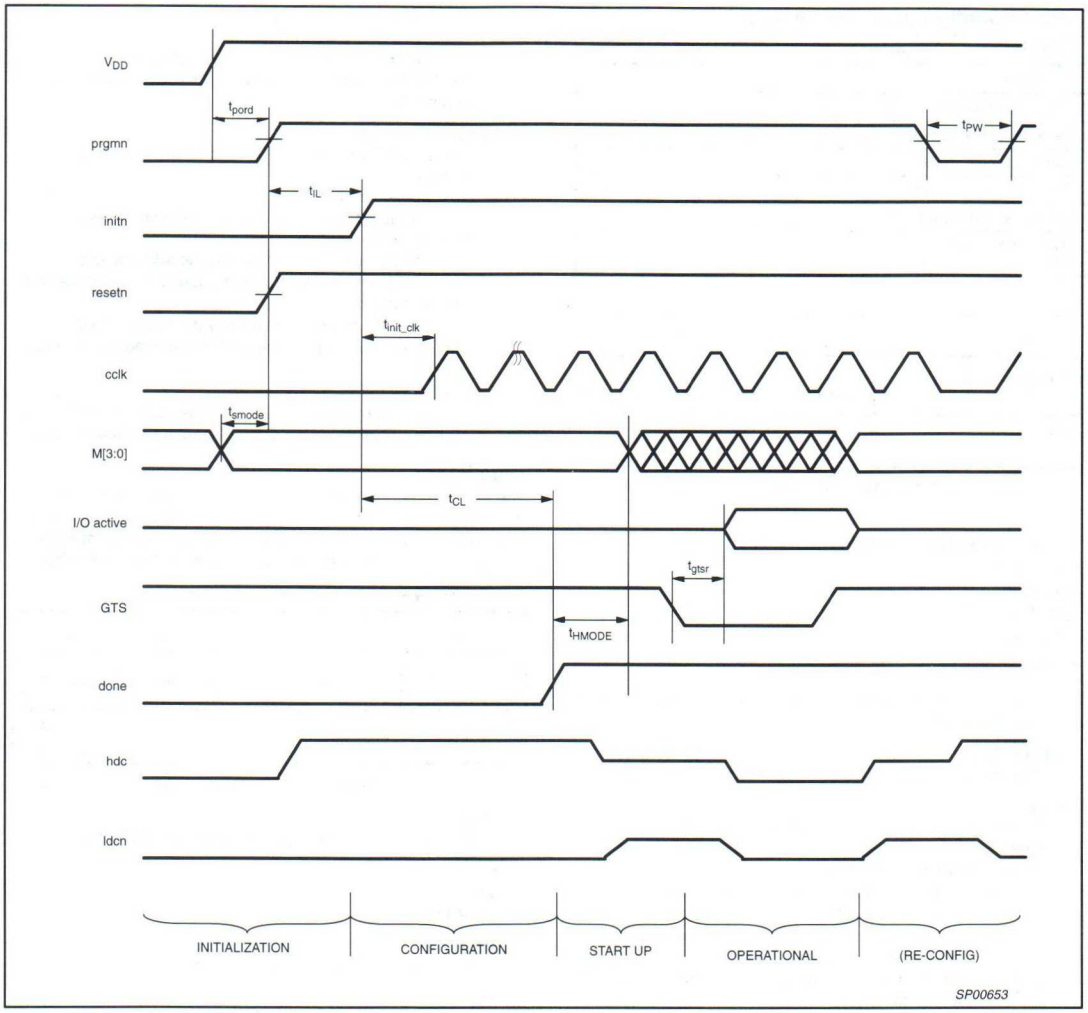


Figure 10. Using gts signal with power up to avoid signal contention with mode select pins

CONFIGURATION DATA FORMAT OVERVIEW

The PZ3320 functionality is determined by the state of internal configuration RAM. This section discusses the configuration data format, and the function of each field in configuration data packets.

Configuration Data Packets

Configuration of the PZ3320 is done using configuration packets. The configuration packet is shown in Figure 11. The data packet consists of a header and a data frame. There are five type of data frames. The header is shifted into the device first, followed by one data frame. Configuration of a single PZ3320 requires 1010 data packets, one for each address. All preceding data must contain only 1s. Once a device is configured, it re-transmits data of any polarity. Before and during configuration, all data re-transmitted out the daisy-chain port (dout) are 1s.

The ordering of the data packets may be random, but they cannot be mixed with other devices' data packets. Alignment bits are not required between data packets. If used, alignment bits must be included in the length count, and they must be at least 2 bits long.

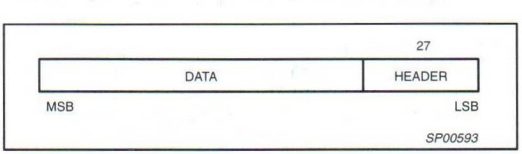


Figure 11. Data Packet

Table 4. Configuration Frame Size

DEVICE	PZ3320
Number of frames	
Data bits/standard frame	
Data bits/compressed frame	
Data bits/user_code frame	
Data bits/isc_code frame	
Data bits/security frame	
Maximum configuration data— # bits/frame × # frames	

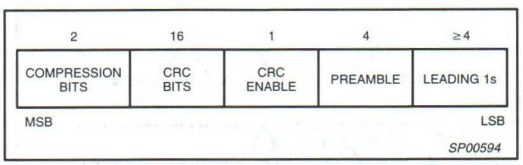


Figure 12. 27-bit Header

The header is fixed and consists of five fields:
- Leading 1s,
- Preamble,
- CRC Enable,
- CRC Bits,
- Compression Bits.

The leading 1s enter the device first. The following is a description of each field in the header.

Leading 1s:
This is a four or greater bit field consisting of 1s.

Preamble:
This is a four bit field which indicates the start of a frame when the least significant bit of the preamble is a 0.
There are two valid preambles:
0010 – indicates that the data packet configures the device receiving the 0010 preamble)
0100 – indicates end of configuration of the device receiving the 0100 preamble
All other values of the preamble field force configuration of the entire system to restart.

The segments CRC Enable, CRC Bits, and Compression Bits are valid only if the Preamble field is 0010.

Cyclic Redundancy Check (CRC) Enable:
In this single bit field, a 0 disables CRC checking of the data stream. If the CRC is disabled the 16 bit CRC field must be the default described below. A 1 enables CRC error checking of the data stream.

CRC Error Checking:
The CRC field is a 16 bit field. The default value is 1010_1010_1010_1010. The calculated value is from data, address, stop bit, and first alignment bit (starting with crc_reg[15:0] = [0]). Using verilog operators, the crc is calculated as:

```
crc_reg[14:2] <= cr_reg[14:2] << 1;
cr_reg[2] <= cr_reg[15]^din^cr_reg[1];
cr_reg[1] <= cr_reg[0];
cr_reg[0] < cr_reg[15]^din;
cr_reg[15] <= cr_reg[15]^din^cr_reg[14];
```

If a CRC error is detected, configuration is halted and must be restarted.

Compression Bits:
This 2-bit field defines the use of compression of the data packets.
00 – Standard mode:
The data packet contains both address and data
01 – Reset mode:
The data packet contains only the address field.
This pattern causes the configuration register to be reset.
10 – Hold mode:
The data packet contains only the address field.
This pattern causes the configuration register to hold its value.
11 – Set mode:
The data packet contains only the address field.
This pattern causes the configuration register to be set.

Data Frames

The five types of data frames are standard, compressed, user_code, isc_code, and security. All fields must be completely filled, with 1s used to fill unused bits. The security frame must be the last frame sent to a device. The definition of each frame is described below:

Standard frame

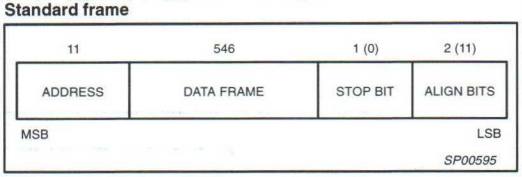


Figure 13. Standard Frame

Address:
This is an 11 bit filed for providing 1011 (1008 SRAM plus 3 user) addresses.

Data:
546 bit field.

Stop bit:
This is a one bit field which must be 0.

Align bit:
This is a two bit field which must be 11.

Compressed frame

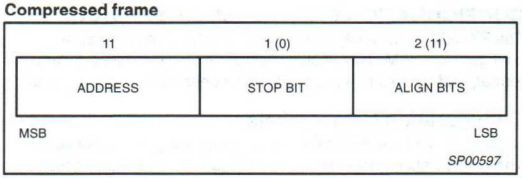


Figure 14. Compressed Frame

The compressed frame contains no data.

User code frame
The user code is located at address 1008D.

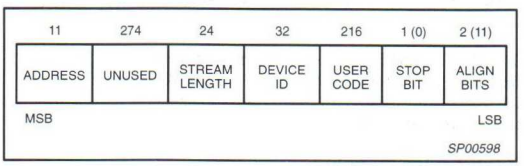


Figure 15. User code Frame

Stream length:
This is a 24 bit field containing the length of the data stream transmitted to configure all of the devices in the daisy chain. This field is only used by a PZ3320 if it is in the master mode.

Device ID:
This is a 32-bit field containing PZ3320 device ID:
492 SBGA: 0000_001_001_101000_1_000_00000010101_1

User code:
This is a 216 bit field reserved for user information.

ISC code frame
The isc_code address is 1009.

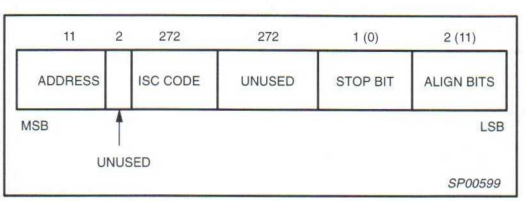


Figure 16. ISC Frame

The ISC frame allows the user to write an ISC code to the device.

Security frame

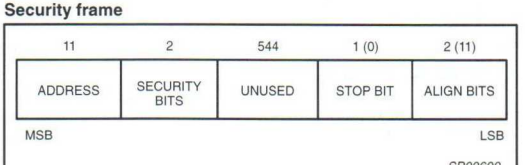


Figure 17. Security Frame

Security bits:
This is a two bit field specifying the level of security.
00 – Unlimited readback allowed.
01 – Readback operation allowed once.
10 – Readback operation allowed once.
11 – Readback operation is disabled.

Re-configuration

To reconfigure the PZ3320 when the device is operating in the system, a low pulse is input into prgmn. The configuration data in the PZ3320 is cleared, and the I/Os not used for configuration are 3-Stated. The PZ3320 then samples the mode select inputs and begins re-configuration. When configuration is compete, done is released, allowing it to be pulled high.

Bit Stream Error Checking

There are three different types of bit stream error checking in the PZ3320:
- ID frame,
- Frame alignment, and
- CRC checking.

An optional ID data frame can be sent to a specified address in the PZ3320. This ID Frame contains a unique code which is compared with the value in the PZ3320 ID register. Any differences are flagged as an ID error.

CRC checking is done on each frame if enabled by setting the CRCen bit in the header. If there is an error, a CRC error is flagged. When an error occurs, the PZ3320 is forced into the initialization state, forcing initn low. The PZ3320 remains in this state until either the resetn or prgmn pins is asserted.

PZ3320 CONFIGURATION MODES

The method for configuring the PZ3320 is selected by the M0, M1, and M2 inputs. The M3 input is used to select the frequency of the internal oscillator, which is the source for cclk in master configuration modes. The nominal frequencies of the internal oscillator are 1.25MHz and 10MHz. The 1.25MHz frequency is selected when the M3 input is unconnected or driven to a high state.

Master Serial Mode

In the master serial mode, the PZ3320 loads the configuration data from an external serial ROM. The configuration data is either loaded automatically at start-up or on a command to reconfigure. Serial EEPROMs from Altera, Atmel, Lucent, Microchip, and Xilinx can be used to configure the PZ3320 in the master serial mode. This provides a simple four-pin interface in an eight-pin package. Serial EEPROMs are available in 32K, 64K, 128K, 256K, and 1M bit densities.

Configuration in the master serial mode can be done at power-up and/or upon a configure command. The system or the PZ3320 must activate the serial EEPROM's $\overline{\text{RESET}}$/OE and $\overline{\text{CE}}$ inputs. At power-up, the PZ3320 and serial EEPROM each contain internal power-on reset circuitry which allows the PZ3320 to be configured without the system providing an external signal. The power-on reset circuitry causes the serial EEPROMs' internal address pointer to be reset. After power-up, the PZ3320 automatically enters its initialization phase.

The serial EEPROM/PZ3320 interface used depends on such factors as the availability of a system reset pulse, availability of an intelligent host to generate a configure command, whether a single serial EEPROM is used or multiple serial ROMs are cascaded, whether the serial EEPROM contains a single or multiple configuration programs, etc.

Data is read into the PZ3320 sequentially from the serial ROM. The DATA output from the serial EEPROM is connected directly into the din input of the PZ3320. The cclk output from the PZ3320 is connected to the CLOCK input of the serial EEPROM. During the configuration process, cclk clocks one data bit into the PZ3320 on each rising edge.

Since the data and clock are direct connects, the PZ3320/serial EEPROM interface task is to use the system or PZ3320 to enable the $\overline{\text{RESET}}$/OE and $\overline{\text{CE}}$ of the serial EEPROM(s). There are several methods for enabling the serial ROM's $\overline{\text{RESET}}$/OE and $\overline{\text{CE}}$ inputs. The serial EEPROM's $\overline{\text{RESET}}$/OE is programmable to function with

resetn active-high and $\overline{OE}$ active-low, or resetn active-low and OE active-high.

In Figure 18, three serial EEPROMs are cascaded to configure a PZ3320. The host generates a 500ns low pulse into the PZ3320's prgmn input and into the serial EEPROMs' $\overline{RESET}$/OE input, which has been programmed to function with resetn active-low and OE active-high. The PZ3320 done is routed to the $\overline{CE}$ pin. The low on done enables the serial EEPROMs. At the completion of configuration, the high on the PZ3320's done disables the EEPROM(s).

When configuration data requirements exceed the capacity of a single serial EEPROM, multiple serial EEPROMs can be cascaded to support the configuration of a single (or multiple) PZ3320(s). After the last bit from the first serial ROM is read, the serial ROM outputs $\overline{CEO}$ low and 3-States the DATA output. The next serial ROM recognizes the low on $\overline{CE}$ input and outputs configuration data on the DATA output. After configuration is complete, the PZ3320's done output into $\overline{CE}$ disables the serial EEPROMs.

In applications in which a serial EEPROM stores multiple configuration programs, the subsequent configuration program(s) are stored in EEPROM locations that follow the last address for the previous configuration program. The user must ensure that a high output on the PZ3320 done signal does not reset the serial EEPROM address pointer, causing the first configuration to be reloaded.

Contention on the PZ3320's din pin must be avoided. During configuration, din receives configuration data. After configuration, it is a user I/O at start-up. An alternative is to use ldcn to drive the serial EEPROMs' $\overline{CE}$ pin.

Master Parallel Mode

The master parallel configuration mode is generally used to interface to industry-standard byte-wide memory such as 256K and larger EEPROMs. Figure 20 provides the interface for master parallel mode. The PZ3320 outputs a 20-bit address on A[19:0] to memory and reads one byte of configuration data on the rising edge of rclk. The parallel bytes are internally serialized starting with the least significant bit, D0. There are two parallel master modes: master up, and master down. In master up, the starting memory address is 00000 Hex and the PZ3320 increments the address for each byte loaded. In master down, the starting memory address is FFFFFH and the PZ3320 decrements the address.

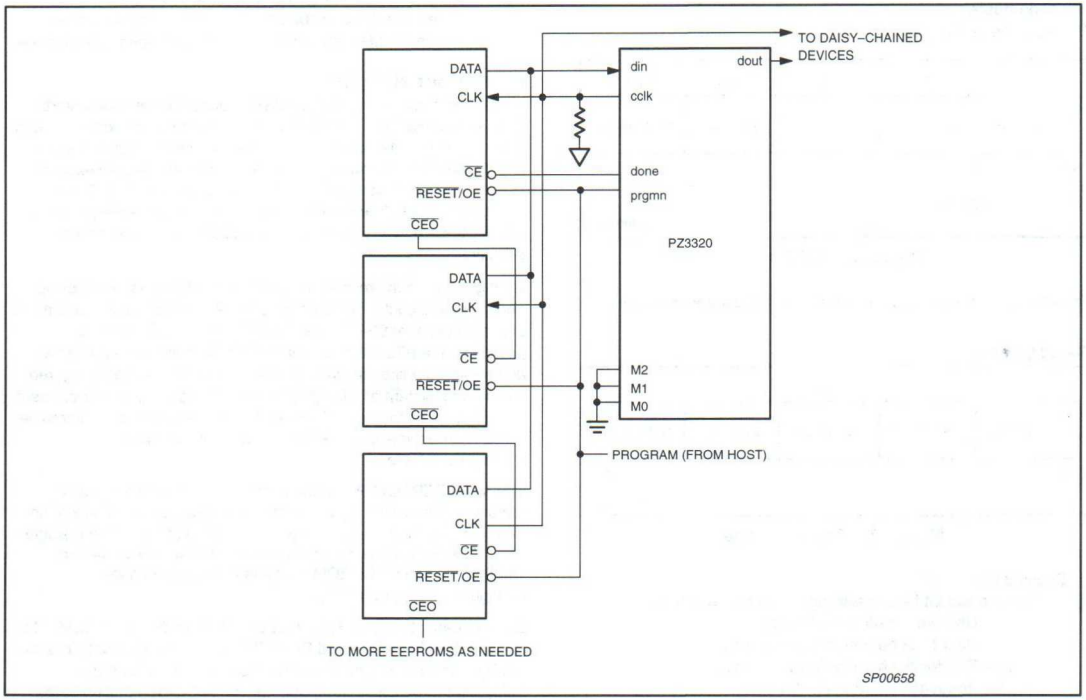


Figure 18. Master Serial Configuration

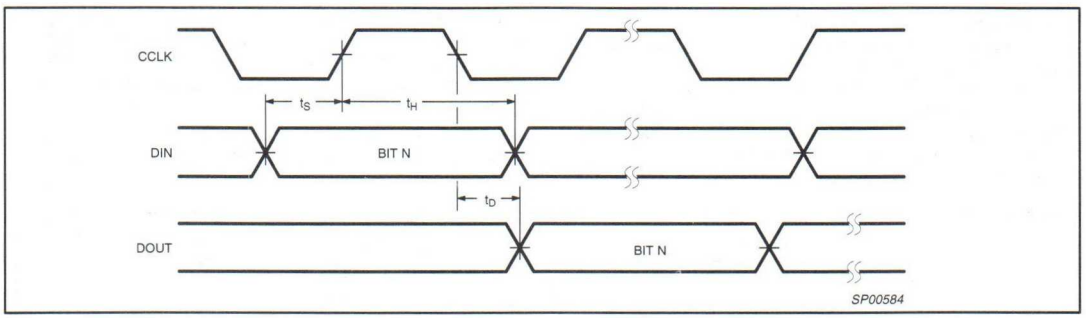


Figure 19. Master Serial Configuration Mode Timing Diagram

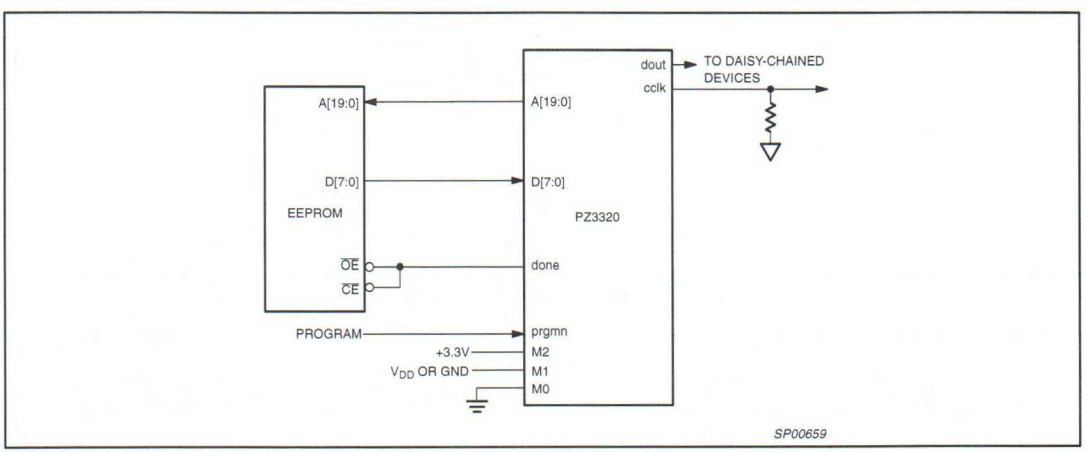


Figure 20. Master Parallel Configuration

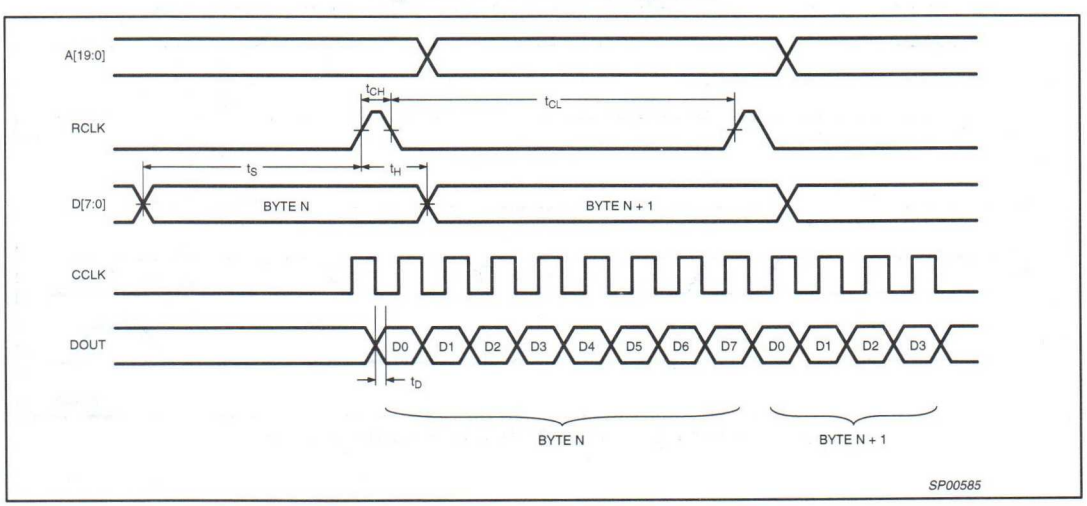


Figure 21. Master Parallel Configuration Mode Timing Diagram

Synchronous Peripheral Mode

In the synchronous peripheral mode, byte-wide data is input into D[7:0] on the rising edge of the cclk input. The first data byte is clocked in on the second cclk after initn goes high. Subsequent data bytes are clocked in on every eighth rising edge of cclk. The rdy_busyn signal is an output which acts as an acknowledge. rdy_busyn goes high one cclk after a byte of data is clocked in on D[7:0] and returns low one cclk cycle later. The process repeats until all of the data is loaded into the PZ3320. The serial data begins shifting out on dout 0.5 cycles after the parallel data was loaded. It requires additional cclks after the last byte is loaded to complete the shifting. Figure 22 shows the interface for synchronous peripheral mode.

As with master modes, the peripheral modes can be used as the lead PZ3320 for daisy-chained PZ3320s.

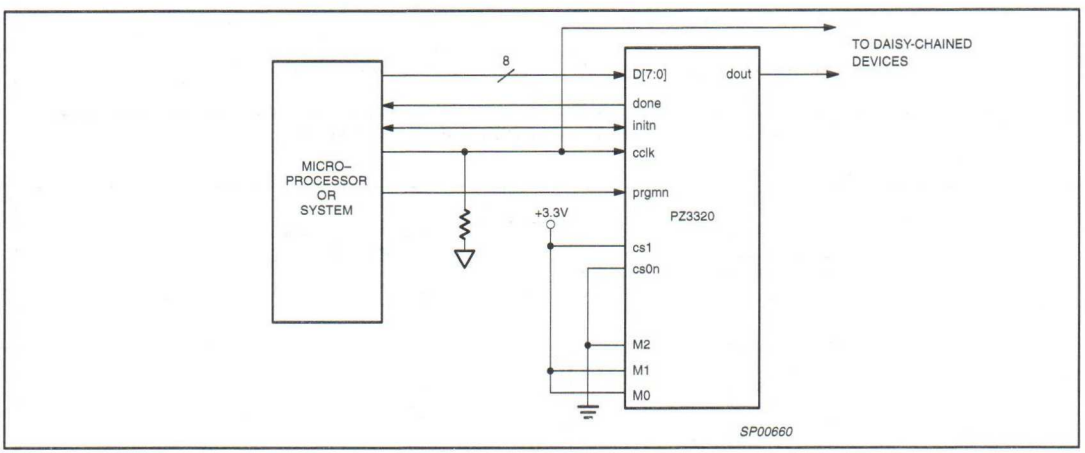


Figure 22. Synchronous Peripheral Configuration

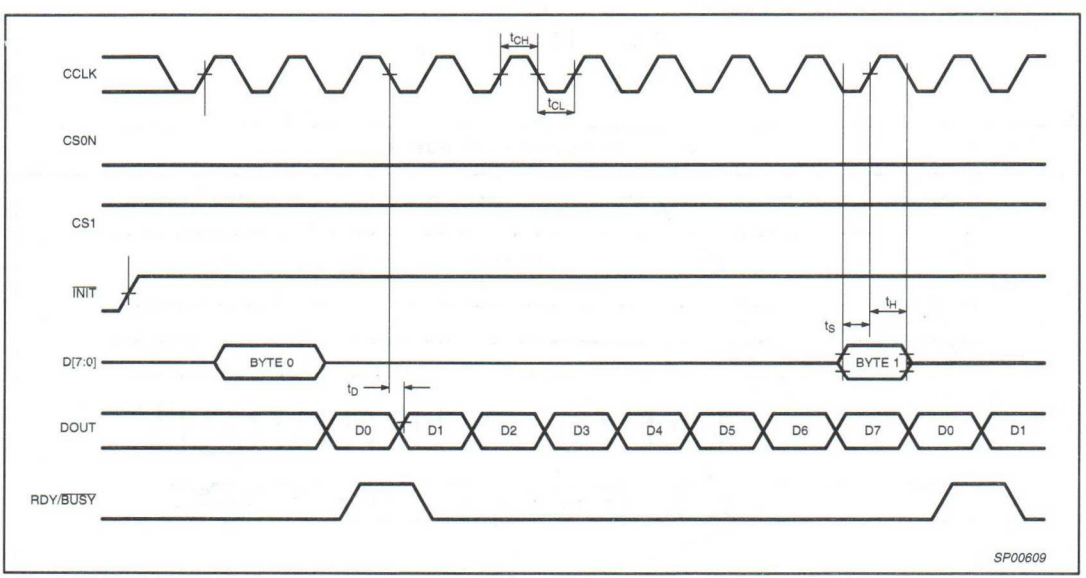


Figure 23. Synchronous Peripheral Configuration Mode Timing Diagram

Slave Serial Mode

The slave serial mode is primarily used when multiple PZ3320s are configured in a daisy-chain. The serial slave serial mode is also used on the PZ3320 evaluation board, which interfaces to the download cable. A device in the slave serial mode can be used as the lead device in a daisy-chain. Figure 24 shows the interface for the slave serial configuration mode.

The configuration data is provided into the PZ3320's din input synchronous with the configuration clock cclk input. After the PZ3320 has loaded its configuration data, it re-transmits incoming configuration data on dout. When used in daisy-chained operation, cclk is routed into all slave serial mode devices in parallel.

Multiple slave PZ3320s can be loaded with identical configurations simultaneously. This is done by loading the configuration data into the din inputs in parallel.

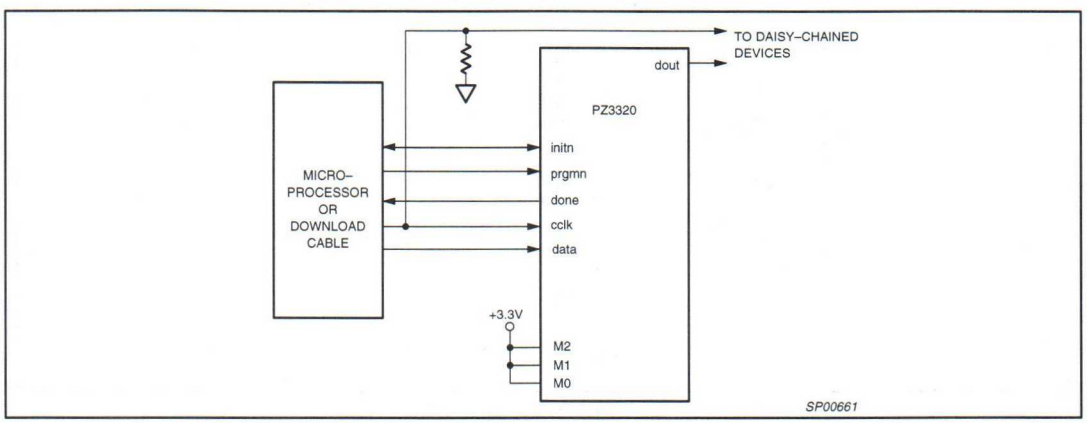


Figure 24. Slave Serial Configuration Schematic

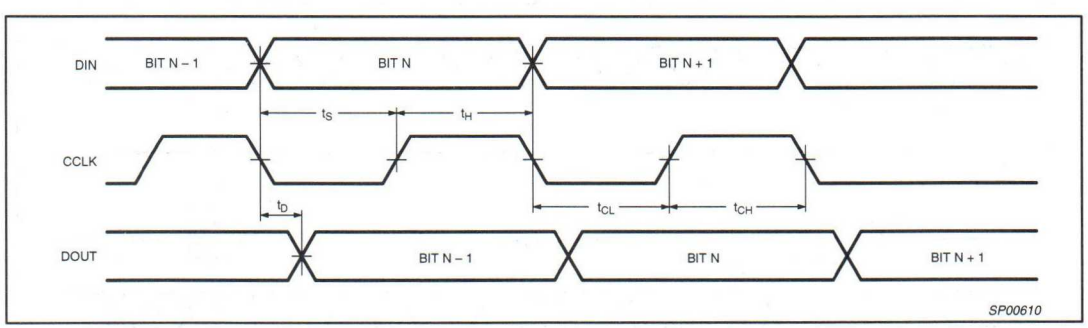


Figure 25. Slave Serial Configuration Mode Timing Diagram

Slave Parallel Mode

The slave parallel mode is essentially the same as the synchronous peripheral mode, except that cs1 and cs0n do not need to be driven, and there is no rdy_bsyn output. As in the synchronous peripheral mode, byte-wide data is input into D[7:0] on the rising edge of the cclk input. The first data byte is clocked in on the second cclk after initn goes high. Subsequent data bytes are clocks in on every eighth rising edge of cclk. The process repeats until all of the data is loaded into the PZ3320. The serial data begins shifting out on dout 0.5 cycles after the parallel data was loaded. It requires additional cclks after the last byte is loaded to complete the shifting. Figure 26 shows the interface for slave parallel mode.

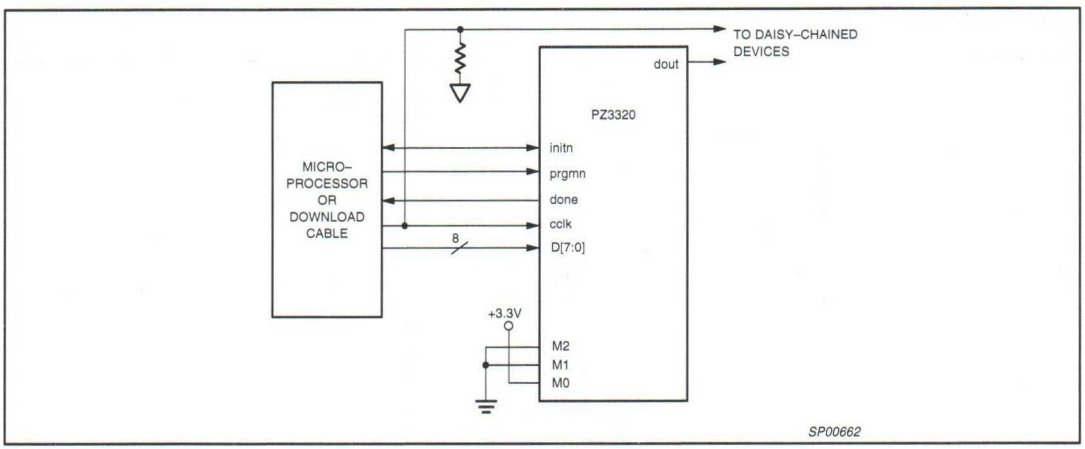


Figure 26. Slave Parallel Configuration Schematic

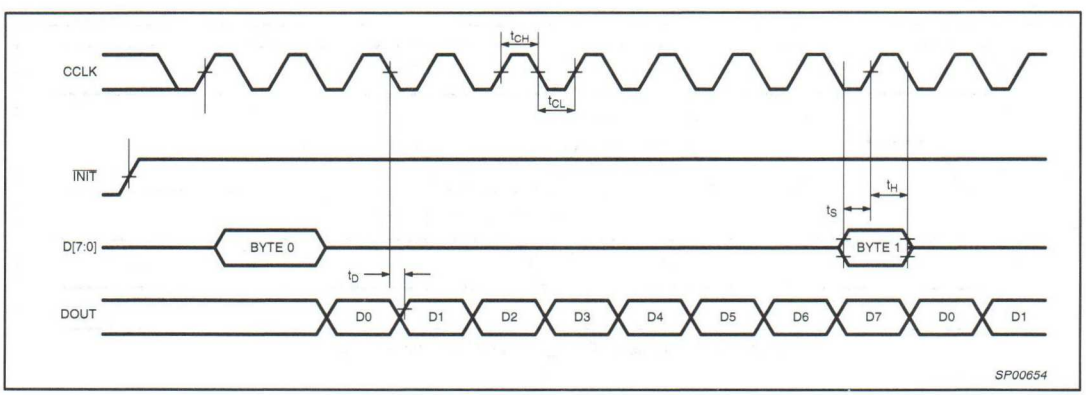


Figure 27. Slave Parallel Configuration Mode Timing Diagram

DAISY CHAIN OPERATION

Multiple PZ3320s can be configured by using a daisy-chain of PZ3320s. Daisy-chaining uses a lead PZ3320 and one or more PZ3320s configured in slave serial mode. The lead PZ3320 can be configured in any mode, but master parallel is typically used. Figure 28 shows the connections for loading multiple PZ3320s in a daisy-chain configuration.

Daisy-chained PZ3320s are connected in series. An upstream PZ3320 which has received the preamble outputs a high on dout until it has received the appropriate number of data frames. This ensures that downstream PZ3320s do not receive frame start bits. After loading and re-transmitting the preamble to a daisy-chain of slave devices, the lead device loads its configuration data frames. The loading of configuration data continues after the lead device has received its configuration data if the lead device's internal frame bit counter has not reached the length count. When the configuration RAM is full and the number of bits received is less than the length count field, the PZ3320 shifts data out on dout.

The generation of cclk for the daisy-chained devices which are in slave serial mode differs depending on the configuration mode of the lead device. A master parallel mode device uses its internal timing generator to produce an internal cclk at eight times its memory address rate (rclk). If the lead device is configured in either synchronous peripheral, slave serial mode, or slave parallel mode, cclk is routed to the lead device and to all of the daisy-chained devices. The configuration data is read into din of slave devices on the positive edge of cclk, and shifted out dout on the negative edge of cclk.

The development software can create a composite configuration file for configuring daisy-chained PZ3320s. The configuration data consists of multiple concatenated data packets. As seen in Figure 28, the initn pins for all of the PZ3320s are connected together. This is required to guarantee that power-up and initialization function correctly. In general, the done pins for all of the PZ3320s are also connected together as shown to guarantee that all of the PZ3320s enter the start-up state simultaneously. This may not be required, depending upon the start-up sequence desired.

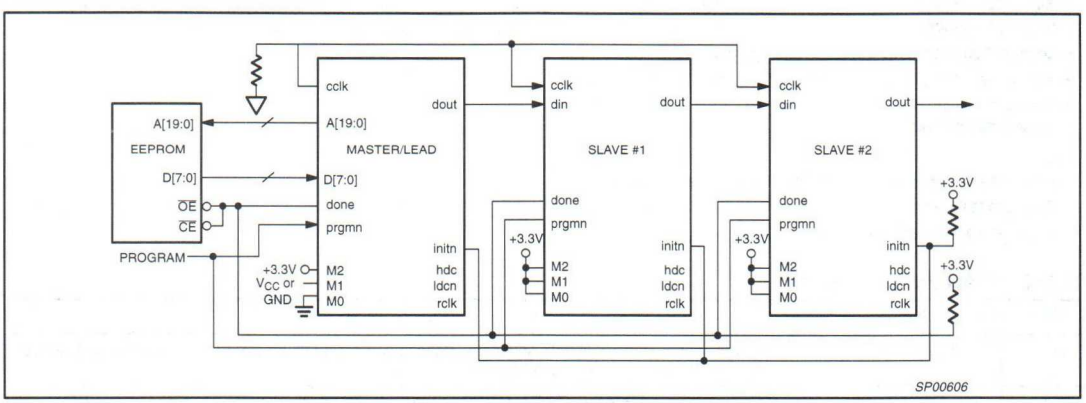


Figure 28. Daisy-chain Schematic

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 PZ3320'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 PZ3320, the TAP Port includes five pins (refer to Table 5) described in the JTAG specification: tck, tms, tdi, tdo, and trstn. These pins should be connected to an external pull-up resistor to keep the JTAG signals from floating when they are not being used.

Table 6 defines the dedicated pins used by the mandatory JTAG signals for the PZ3320.

The JTAG specifications define 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 PZ3320 is defined in Table 7. By supporting this set of low-level commands, the PZ3320 allows execution of all high-level boundary-scan commands.

Table 5. 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.
trstn	Test Reset	Forces TAP controller to test logic reset state. This signal is active low.

Table 6. PZ3320 JTAG Pinout by Package Type

DEVICE	(PIN NUMBER / MACROCELL #)				
	tck	tms	tdi	tdo	trstn

Table 7. PZ3320 Low-Level JTAG Boundary-Scan Commands

INSTRUCTION (Instruction Code) *Register Used*	DESCRIPTION
SAMPLE/PRELOAD (00010) *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 (00000) *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 (11111) *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 (00001) *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 (00101) *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.

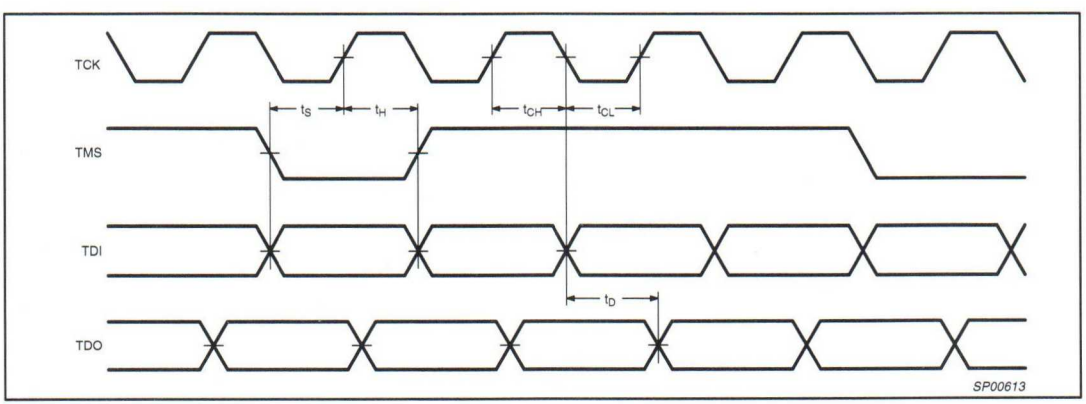


Figure 29. Boundary Scan Timing Diagram

Table 8. Boundary scan timing characteristics

SYMBOL	PARAMETER	MIN	MAX	UNIT
t_S	tdi/tms to tck setup time	20	–	ns
t_H	tdi/tms from tck hold time	0	–	ns
t_{CH}	tck high time	50	–	ns
t_{CL}	tck low time	50	–	ns
f_{TCK}	tck frequency	–	10	MHz
t_D	tck to tdo delay	–	20	ns

ABSOLUTE MAXIMUM RATINGS[4]

SYMBOL	PARAMETER	MIN	MAX	UNIT
V_{DD}	Supply voltage	–0.5	4.6	V
V_{IN}	Input voltage	–1.2		V
V_{OUT}	Output voltage	–0.5	V_{DD}+0.5	V
I_{IN}	Input current	–30	30	mA
T_J	Junction temperature range	–40	150	°C
T_{STG}	Storage temperature range	–65	150	°C

NOTE:
4. Stresses above these 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 temperature range: V_{DD} = 3.0V to 3.6V; 0°C < T_{amb} < 70°C

SYMBOL	PARAMETER	TEST CONDITIONS	MIN	MAX	UNIT
V_{IH}	Input high voltage	V_{DD} = 3.6V	2.0	V_{DD}+0.3	V
V_{IL}	Input low voltage	V_{DD} = 3.0V	–0.3	0.8	V
V_{OH}	Output high voltage	V_{DD} = 3.0V; I_{OH} = –8mA	2.4	–	V
V_{OL}	Output low voltage	V_{DD} = 3.0V; I_{OH} = 8mA	–	0.4	V
I_I	Input leakage current	V_{DD} = 3.6V; 0 < V_{IN} < V_{DD}	–10	10	µA
I_{DDSB}	Standby current	T_{amb} = 25°C; no output loads, inputs at V_{DD} or V_{SS}.	–	100	µA
C_{IN}	Input capacitance	T_{amb} = 25°C; V_{DD} = 3.3V; f = 1MHz	–	10	pF
C_{IO}	I/O capacitance	T_{amb} = 25°C; V_{DD} = 3.3V; f = 1MHz	–	10	pF
C_{CLK}	Clock pin capacitance	T_{amb} = 25°C; V_{DD} = 3.3V; f = 1MHz	–	12	pF
R_{DONE}	done pull-up resistor	V_{DD} = 3.0 V; V_{IN} = 0 V	10	30	kΩ
R_{PD}	Unused I/O pull-down resistor	V_{DD} = 3.6V; V_{IN} = V_{DD}	100	400	kΩ

AC ELECTRICAL CHARACTERISTICS FOR COMMERCIAL GRADE DEVICES

Commercial temperature range: V_{DD} = 3.0V to 3.6V; 0°C < T_{amb} < 70°C

SYMBOL	PARAMETER	MIN	MAX	UNIT
Timing requirements				
t_{CL}	Clock LOW time	2.5		ns
t_{CH}	Clock HIGH time	2.5		ns
t_{SU_PAL}	PAL setup time (Global clock)	4.0		ns
t_{SU_PLA}	PLA setup time (Global clock)	5.5		ns
t_{SU_XOR}	XOR setup time (Global clock)	6.5		ns
t_H	Hold time (Global clock)		0	ns
Output characteristics				
t_{PD_PAL}	Input to output delay through PAL		7.5	ns
t_{PD_PLA}	Input to output delay through PLA		9.0	ns
t_{PD_XOR}	Input to output delay through XOR		10.0	ns
t_{PDF_PAL}	Input (or feedback node) to internal feedback node delay time through PAL		4.0	ns
t_{PDF_PLA}	Input (or feedback node) to internal feedback node delay time through PLA		5.5	ns
t_{PDF_XOR}	Input (or feedback node) to internal feedback node delay time through XOR		6.5	ns
t_{CF}	Global clock to feedback delay		2.5	ns
t_{CO}	Global clock to out delay		6.0	ns
t_{CS}	Clock skew (variance for switching outputs with common global clock)		1.0	ns
f_{MAX1}	Maximum flip-flop toggle rate $\left(\frac{1}{t_{CL} + t_{CH}}\right)$	200		MHz
f_{MAX2}	Maximum internal frequency $\left(\frac{1}{t_{SU_PAL} + t_{CF}}\right)$	154		MHz
f_{MAX3}	Maximum external frequency $\left(\frac{1}{t_{SU_PAL} + t_{CO}}\right)$	100		MHz
t_{BUFF}	Output buffer delay (fast)		3.5	ns
t_{SSR}	Slow slew rate incremental delay		8.0	ns
t_{EA}	Output enable delay		8.0	ns
t_{ER}	Output disable delay[1]		8.0	ns
t_{GTSH}	Global 3-State enable		40.0	ns
t_{GTSR}	Global 3-State disable		40.0	ns
t_{RR}	Input to register reset		10.5	ns
t_{RP}	Input to register preset		10.5	ns
t_{GRR}	Global reset to register reset		40	ns
t_{GZIA}	Global ZIA delay		4.0	ns

NOTE:
1. Output C_L = 5.0pF.

DC ELECTRICAL CHARACTERISTICS FOR INDUSTRIAL GRADE DEVICES

Industrial temperature range: V_{DD} = 3.0V to 3.6V; −40°C < T_{amb} < 85°C

SYMBOL	PARAMETER	TEST CONDITIONS	MIN	MAX	UNIT
V_{IH}	Input high voltage	V_{DD} = 3.6V	2.0	V_{DD}+0.3	V
V_{IL}	Input low voltage	V_{DD} = 3.0V	−0.3	0.8	V
V_{OH}	Output high voltage	V_{DD} = 3.0V; I_{OH} = −8mA	2.4	–	V
V_{OL}	Output low voltage	V_{DD} = 3.0V; I_{OH} = 8mA	–	0.4	V
I_I	Input leakage current	V_{DD} = 3.6V; 0 < V_{IN} < V_{DD}	−10	10	µA
I_{DDSB}	Standby current	T_{amb} = 25°C; no output loads, inputs at V_{DD} or V_{SS}.	–	100	µA
C_{IN}	Input capacitance	T_{amb} = 25°C; V_{DD} = 3.3V; f = 1MHz	–	10	pF
C_{IO}	I/O capacitance	T_{amb} = 25°C; V_{DD} = 3.3V; f = 1MHz	–	10	pF
C_{CLK}	Clock pin capacitance	T_{amb} = 25°C; V_{DD} = 3.3V; f = 1MHz	–	12	pF
R_{DONE}	done pull-up resistor	V_{DD} = 3.0 V; V_{IN} = 0 V	10	30	kΩ
R_{PD}	Unused I/O pull-down resistor	V_{DD} = 3.6V; V_{IN} = V_{DD}	100	400	kΩ

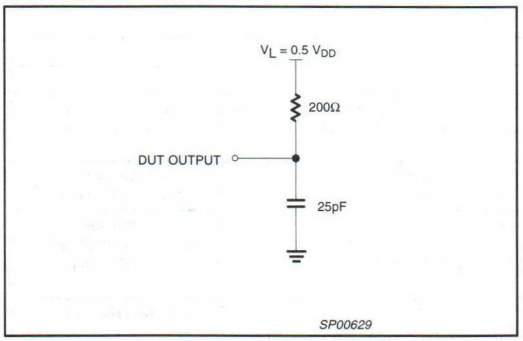
AC ELECTRICAL CHARACTERISTICS FOR INDUSTRIAL GRADE DEVICES

Industrial temperature range: V_{DD} = 3.0V to 3.6V; −40°C < T_{amb} < 85°C

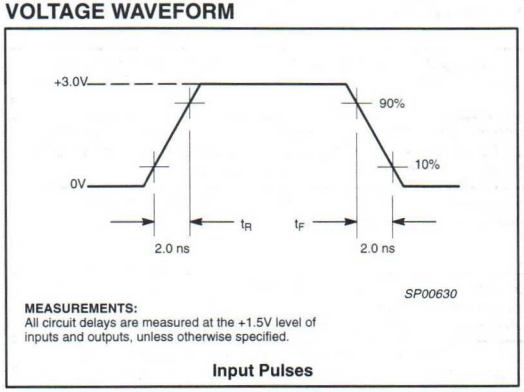
SYMBOL	PARAMETER	MIN	MAX	UNIT
Timing requirements				
t_{CL}	Clock LOW time	2.5		ns
t_{CH}	Clock HIGH time	2.5		ns
t_{SU_PAL}	PAL setup time (Global clock)	4.5		ns
t_{SU_PLA}	PLA setup time (Global clock)	6.0		ns
t_{SU_XOR}	XOR setup time (Global clock)	7.0		ns
t_H	Hold time (Global clock)		0	ns
Output characteristics				
t_{PD_PAL}	Input to output delay through PAL		8.0	ns
t_{PD_PLA}	Input to output delay through PLA		9.5	ns
t_{PD_XOR}	Input to output delay through XOR		10.5	ns
t_{PDF_PAL}	Input (or feedback node) to internal feedback node delay time through PAL		4.0	ns
t_{PDF_PLA}	Input (or feedback node) to internal feedback node delay time through PLA		5.5	ns
t_{PDF_XOR}	Input (or feedback node) to internal feedback node delay time through XOR		6.5	ns
t_{CF}	Global clock to feedback delay		2.5	ns
t_{CO}	Global clock to out delay		6.5	ns
t_{CS}	Clock skew (variance for switching outputs with common global clock)		1.0	ns
f_{MAX1}	Maximum flip-flop toggle rate $\left(\frac{1}{t_{CL} + t_{CH}}\right)$	200		MHz
f_{MAX2}	Maximum internal frequency $\left(\frac{1}{t_{SU_PAL} + t_{CF}}\right)$	143		MHz
f_{MAX3}	Maximum external frequency $\left(\frac{1}{t_{SU_PAL} + t_{CO}}\right)$	91		MHz
t_{BUFF}	Output buffer delay (fast)		4.0	ns
t_{SSR}	Slow slew rate incremental delay		8.0	ns
t_{EA}	Output enable delay		8.5	ns
t_{ER}	Output disable delay[1]		8.5	ns
t_{GTSH}	Global 3-State enable		40.0	ns
t_{GTSR}	Global 3-State disable		40.0	ns
t_{RR}	Input to register reset		11.0	ns
t_{RP}	Input to register preset		11.0	ns
t_{GRR}	Global reset to register reset		40	ns
t_{GZIA}	Global ZIA delay		4.5	ns

NOTE:
1. Output C_L = 5.0pF.

THEVENIN EQUIVALENT



VOLTAGE WAVEFORM



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

Input Pulses

Table 9. Pin Description

SYMBOL	TYPE	DESCRIPTION
V_{DD}	–	Positive power supply.
GND	–	Ground supply.
resetn	I	During configuration, resetn forces the start of initialization (see Figure 8). After configuration, resetn is a direct input which can be used to asynchronously reset all the flip-flops. If the global reset is not being used, this pin should be pulled high.
cclk	I/O	In the master modes, cclk is an output which strobes configuration data in. In the slave or synchronous peripheral mode, cclk is an input synchronous with the data on din or D[7:0]. After configuration, this pin should be pulled low.
done	I/O	done is a bi-directional signal with a weak pull-up resistor attached. As an output, done pulling high indicates configuration is complete. As an input, a low level on done will delay device initialization and the enabling of user I/O. If only one device is used, this pin can be left floating. If multiple devices are daisy chained, an external pull-up should be used (see Figure 28).
pgmn	I	pgmn is an active-low input that forces the restart of configuration and initialization (see Figure 8) and resets the boundary-scan circuitry. After configuration, the pin should be pulled high.
rdy_busyn	O	During configuration in peripheral mode, rdy_busyn indicates another byte can be written to the PZ3320. After configuration, the pin is a user-programmable I/O, and no external termination is required. See the section on terminations for more information.
rclk	O	During the master parallel configuration mode, rclk is an output signal to an external memory. rclk is not normally used. After configuration, this pin is a user-programmable I/O pin, and no external termination is required. See the section on terminations for more information.
din	I	During slave serial or master serial configuration modes, din accepts serial configuration data synchronous with cclk. During parallel configuration modes, din is the D[0] input. After configuration, the pin is a user-programmable I/O, and no external termination is required. See the section on terminations for more information.
M2 M0 M1	I	M2/M1/M0 are used to select the configuration mode as defined in Table 3. After configuration, the pins are user-programmable I/O, and no external termination is required. See the section on terminations for more information.
M3	I	M3 is used to select the frequency of the internal oscillator during configuration. When M3 is low, the oscillator is nominally 10MHz. When M3 is high, the oscillator is nominally 1.25MHz. After configuration, the pin is a user-programmable I/O, and no external termination is required. See the section on terminations for more information.
tdi tdo tck tms trstn	I O I I I	Test Data In, Test Data Out, Test Clock, Test Mode Select, Test Reset are dedicated pins for boundary-scan through the JTAG port. If JTAG is not being used, tdi, tck, tms, and trstn should be terminated with a weak pull-up resistor. tdo can be left unterminated. See section on terminations for more information.
hdc	O	High During Configuration (hdc) is output high when the PZ3320 is in the configuration state. hdc is used as a control output indicating that configuration is in progress. After configuration, the pin is a user-programmable I/O, and no external termination is required. See the section on terminations for more information.
ldcn	O	Low During Configuration (ldcn) is output low when the PZ3320 is in the configuration state. ldcn is used as a control output indicating that configuration is in progress. After configuration, the pin is a user-programmable I/O, and no external termination is required. See the section on terminations for more information.
initn	I/O	initn is an active-low bi-directional pin that holds the PZ3320 in a wait state before the start of configuration. During configuration, an internal pull-up is enabled. If only one device is used, this pin can be left floating. If multiple devices are daisy chained, an external pull-up should be used (see Figure 28). After configuration, the pin is a user-programmable I/O, and no external termination is required. See the section on terminations for more information.
gts	I	Global 3-State is an active-high dedicated input used to 3-state the I/Os. If this feature is not used, the pin should be pulled low.
cs0n cs1	I	cs0n/cs1 are used in the peripheral configuration mode. The PZ3320 is selected when cs0n is low and cs1 is high. After configuration, these pins are user-programmable I/O, and no external termination is required. See the section on terminations for more information.
A[19:0]	O	In the master parallel configuration mode, A[19:0] address the configuration EEPROM. After configuration, the pin is a user-programmable I/O, and no external termination is required. See the section on terminations for more information.
D[7:0]	I	During master parallel, peripheral, and slave parallel configuration modes, D[7:0] receive configuration data. After configuration, the pin is a user-programmable I/O, and no external termination is required. See the section on terminations for more information.
dout	O	During configuration, dout is the serial data out that is used to drive the din of daisy-chained slave devices. Data on dout changes on the falling edge of cclk. After configuration, the pin is a user-programmable I/O, and no external termination is required. See the section on terminations for more information.

960 macrocell SRAM CPLD

PZ3960C/PZ3960N

FEATURES

- Industry's largest CPLD—960 macrocells
- Industry's first SRAM-based CPLD
- Multiple power-up configuration modes
  - Master serial
  - Slave serial
  - Master parallel-up
  - Master parallel-down
  - Slave parallel
  - Synchronous peripheral
  - Other modes available, contact Philips at 1–888–CoolPLD
- Configuration times of under 1.0 seconds
- IEEE 1149.1 compliant JTAG testing capability
  - 5 pin JTAG interface
  - IEEE 1149.1 TAP controller
- 3.3 volt device
- Innovative XPLA2 Architecture combines extreme flexibility and high speeds
- 8 synchronous clock networks with programmable polarity at every macrocell
- Up to 96 asynchronous clocks support complex clocking needs
- Innovative XOR structure at every macrocell provides excellent logic reduction capability
- Logic expandable to 36 product terms on a single macrocell
- PCI compliant
- Advanced 0.35µ SRAM process
- Design entry and verification using industry standard and Philips CAE tools
- Innovative Control Term structure provides either sum terms of product terms in each logic block for:
  - 3-State buffer control
  - Asynchronous macrocell register reset/preset
- Global 3-State pin facilitates 'bed of nails' testing without sacrificing logic resources
- Programmable slew rate control
- Small form factor 492 pin PBGA package provides 384 I/O pins
- Available in commercial and industrial temperature ranges

Table 1. PZ3960C/PZ3960N Features

	PZ3960C/PZ3960N
Usable gates	30,000
Maximum inputs	384
Maximum I/Os	384
Number of macrocells	960
Propagation delay (ns)	7.5
Package	492-pin PBGA

DESCRIPTION

The PZ3960 device is a member of the CoolRunner™ family of high-density SRAM-based CPLDs (Complex Programmable Logic Device) from Philips Semiconductors. This device combines high speed and deterministic pin-to-pin timing with high density. The PZ3960 uses the patented Fast Zero Power (FZP) design technique that combines high speed and low power for the first time ever in a CPLD. FZP allows the PZ3960 to have true pin-to-pin timing delays of 7.5ns, and standby currents of 100 microamps 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 since the bipolar era) with a cascaded chain of pure CMOS gates, both standby and dynamic power are dramatically reduced when compared to other CPLDs. The FZP design technique is also what allows Philips to offer a true CPLD architecture in a high density device. Competitors offer CPLDs that are approximately half the density of the PZ3960, and yet consume over two times the power.

The Philips PZ3960C/PZ3960N devices use the new patent-pending XPLA2™ (eXtended Programmable Logic Array) architecture. This architecture combines the best features of both PAL- and PLA-type logic structures to deliver high speed and flexible logic allocation that results in superior ability to make design changes with fixed pinouts. The XPLA2™ architecture is constructed from 80 macrocell Fast Modules that are connected together by an interconnect array. Within each Fast Module are four Logic Blocks of 20 macrocells each. Each Logic Block contains a PAL structure with four dedicated product terms for each macrocell. In addition, each Logic Block has 32 additional product terms in a PLA structure that can be shared through a fully programmable OR array to any of the 20 macrocells. This combination efficiently allocates logic throughout the Logic Block, which increases device density and allows for design changes without re-defining the pinout or changing the system timing. The PZ3960 offers pin-to-pin propagation delays of 7.5ns through the PAL array of a Fast Module; and if the PLA array is used, an additional 1.5ns is added to the delay, no matter how many PLA product terms are used. If the interconnect array between Fast Modules is used, there is a second fixed addition to the propagation delay of 4.0ns. This means that the worst case pin-to-pin propagation delay within a fast module is 7.5 + 1.5 = 9.0 ns, and the delay from any pin to any other pin across the entire chip is 7.5 + 4.0 = 11.5ns if only the PAL array is used, and 7.5 + 1.5 + 4.0 = 13.0ns if the PLA array is used. This deterministic timing allows you to establish system timing before the logic design is even started.

Each macrocell also has a two input XOR gate with the dedicated PAL product terms on one input and the PLA product terms on the other input. This patent-pending Versatile XOR structure allows for very efficient logic optimization compared to competing XOR structures that have only one product term as the second input to the XOR gate. The Versatile XOR allows an 8 bit XOR function to be implemented in only 20 product terms, compared to 65 product terms for the traditional XOR approach.

The PZ3960 is SRAM-based, which means that it is configured at power up by one of many different methods. The device may be reconfigured any number of times. See the configuration section of this data sheet for more information. The device supports the full JTAG specification (IEEE 1149.1) through an industry standard JTAG interface.

Software support for the PZ3960 is through industry standard CAE tools (Cadence, Mentor, Synopsys, Synario, Viewlogic, MINC, Exemplar Logic, and Orcad) as well as Philips' own XPLA Designer. Entry methods include both text (ABEL, PHDL, VHDL, Verilog) and/or schematic. Design verification uses industry standard simulators for functional and timing simulation, and development tools are supported on personal computer, SPARC, and HP Workstation platforms. Device fitting uses either MINC or Philips Semiconductors developed tools.

ORDERING INFORMATION

ORDER CODE	PACKAGE, PROPAGATION DELAY	DESCRIPTION	DRAWING NUMBER
PZ3960C7EB	492-pin PBGA, 7.5ns t_{PD}	Commercial temp. range, 3.3 volt power supply ± 10%	SOT514–1
PZ3960N8EB	492-pin PBGA, 8.0ns t_{PD}	Industrial temp. range, 3.3 volt power supply ± 10%	SOT514–1

XPLA2 ARCHITECTURE

Figure 1 shows a high level block diagram of the PZ3960 implementing the XPLA2 architecture. The XPLA2 architecture is a multi-level, modular hierarchy that consists of Fast Modules interconnected by a Global Zero Power Interconnect Array (GZIA). The GZIA is a virtual crosspoint switch that connects the Fast Modules together. Each Fast Module accepts 64 bits from the GZIA and outputs 64 bits to the GZIA. Each Fast Module is essentially an 80 macrocell CPLD with four logic blocks of 20 macrocells each inside. There are eight dedicated, low-skew, global clocks for the device; and each Fast Module has access to any two of these clocks (there are additional asynchronous clocks available in the Fast Modules, see Figure 3). There are also Global 3-state (gts) and Global Reset (rstn) pins that are common to all Fast Modules. When gts is pulled high, all output buffers in the device will be disabled, causing all I/O pins to be tri-stated. When rstn is pulled low, all flip-flops of the device will be reset.

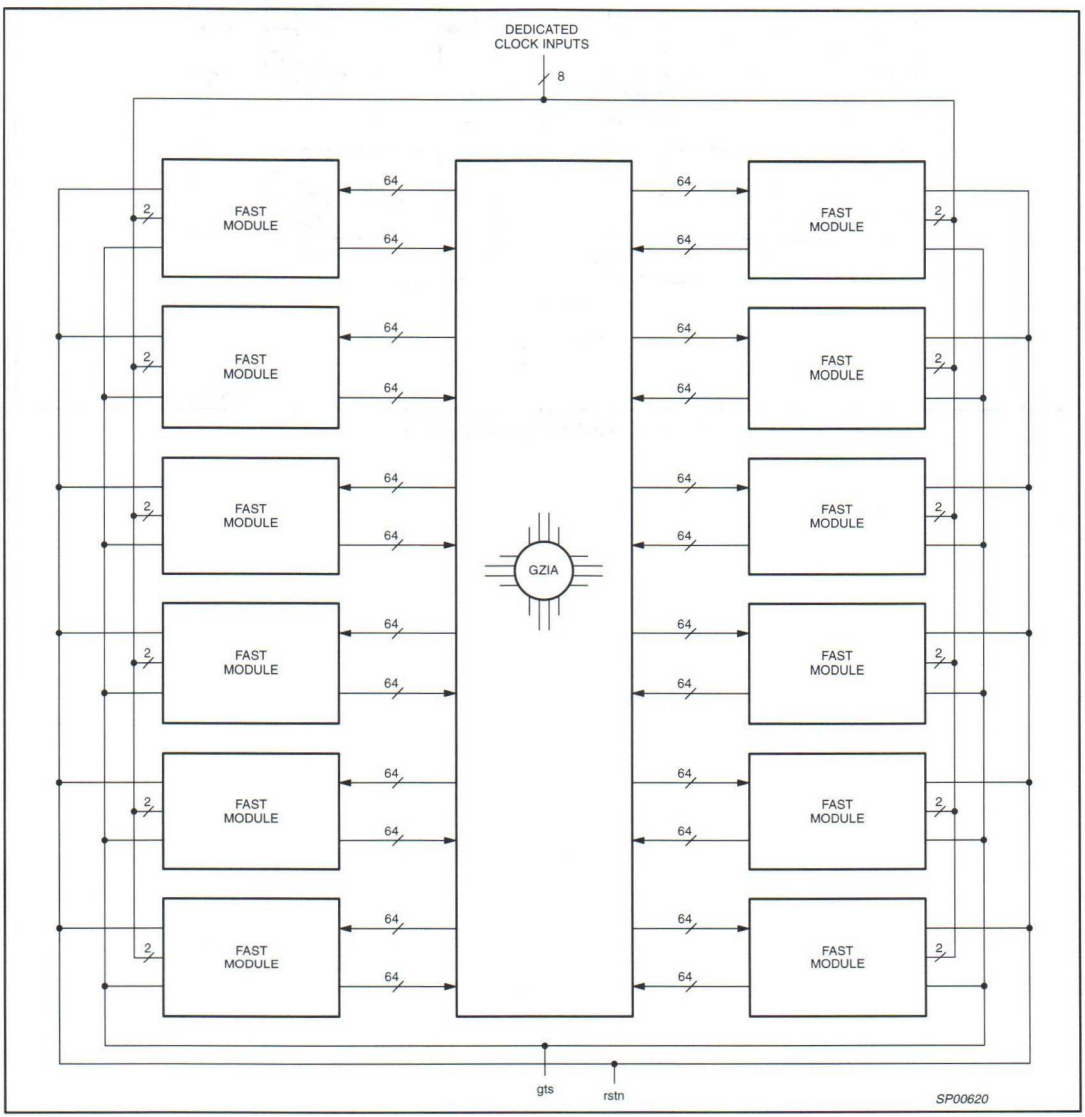


Figure 1. Philips XPLA2 CPLD Architecture

XPLA2 Fast Module

Each Fast Module consists of four Logic Blocks of 20 macrocells each. Eight of the 20 macrocells in each Logic Block are connected to I/O pins and the remaining 12 can be used as buried nodes. These four Logic Blocks are connected together by the Local Zero Power Interconnect Array (LZIA). The LZIA is a virtual crosspoint switch that connects the Logic Blocks to each other and to the GZIA. The feedback from all 80 macrocells, input from the I/O pins, and the 64 bit input bus from the GZIA are input into the LZIA. The LZIA outputs 36 signals into each Logic Block and 64 signals into the GZIA.

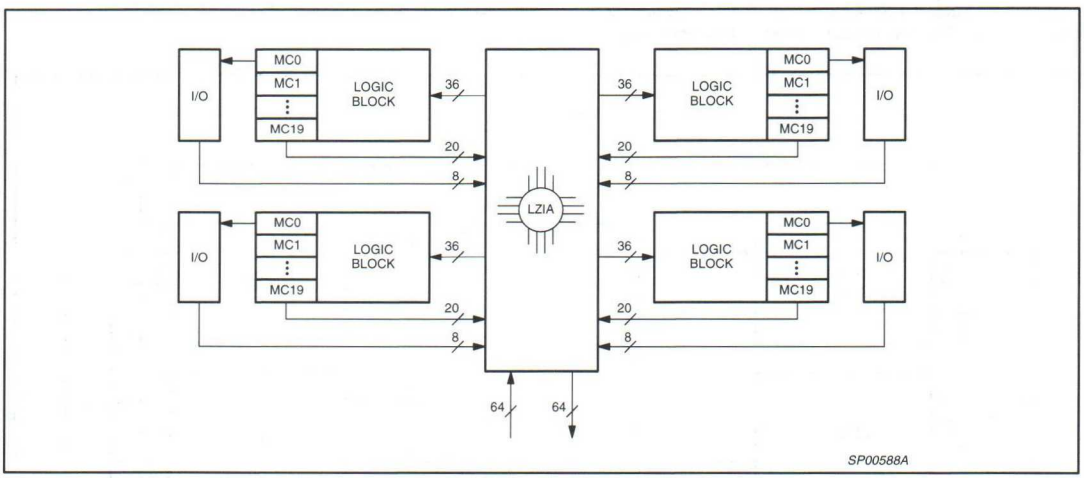


Figure 2. Philips XPLA2 Fast Module

XPLA2 Logic Block Architecture

Figure 3 illustrates the XPLA2 Logic Block architecture. Each Logic Block contains 8 control terms, a PAL array, a PLA array, and 20 macrocells. The 36 inputs from the LZIA are available to all control terms and to each product term in both the PAL and the PLA array. The 8 control terms can individually be configured as either SUM or PRODUCT terms, and are used to control the asynchronous preset and reset functions of the macrocell registers, the output enables of the 20 macrocells, and for asynchronous clocking. 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.

Each macrocell has 4 dedicated product terms from the PAL array. When additional logic is required, each macrocell takes the extra product terms from the PLA array. The PLA array consists of 32 extra product terms that are shared between the 20 macrocells of the Logic Block. The PAL product terms can be connected to the PLA product terms through either an OR gate or an XOR gate. One input to the XOR gate can be connected to all the PLA terms, which provides for extremely efficient logic synthesis. An eight bit XOR function can be implemented in only 20 product terms. Each macrocell can use the output from the OR gate or the XOR gate in either normal or inverted state.

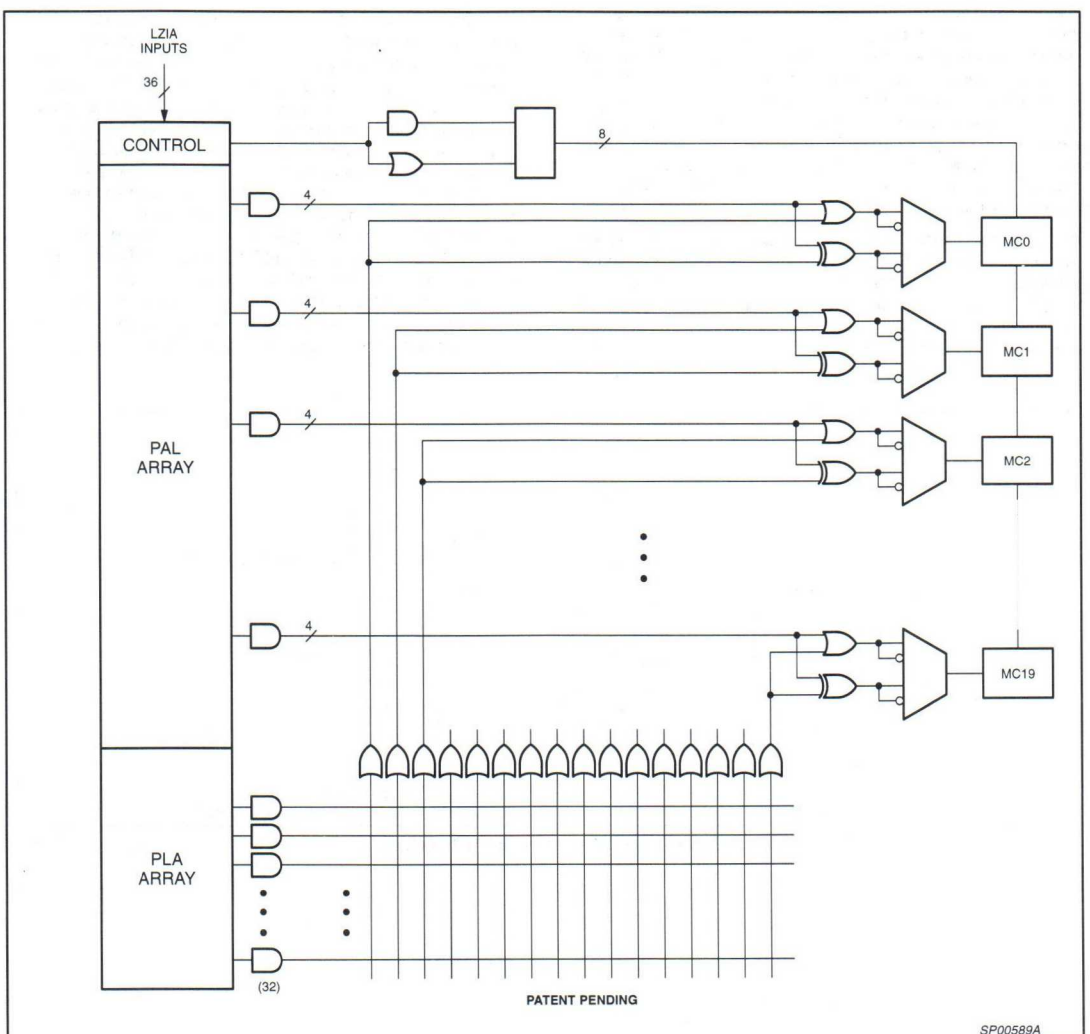


Figure 3. Philips XPLA2 Logic Block Architecture

XPLA2 Macrocell Architecture

Figure 4 shows the XPLA2 macrocell architecture used in the PZ3960. The macrocell can be configured as either a D- or T-type flip-flop or a combinatorial logic function. A D-type flip-flop is generally more useful for implementing state machines and data buffering while a T-type flip-flop is generally more useful in implementing counters. Each of these flip-flops can be clocked from any one of four sources. Two of the clock sources (CLK0 and CLK1) are from the eight dedicated, low-skew, global clock networks designed to preserve the integrity of the clock signal by reducing skew between rising and falling edges. These clocks are designated as a "synchronous" clocks and must be driven by an external source. Both CLK0 and CLK1 can clock the macrocell flip-flops on either the rising edge or the falling edge of the clock signal. The other clock sources are designated as "asynchronous" and are connected to two of the eight control terms (CT6 and CT7) provided in each logic block. These clocks can be individually configured as any PRODUCT term or SUM term equation created from the 36 signals available inside the logic block. Thus, in each Logic Block, there are up to four possible clocks; and in each Fast Module, there are up to 10 possible clocks. Throughout the entire device, there are up to 104 possible clocks—eight from the dedicated, low-skew, global clocks, and two for each of the 48 logic blocks.

The remaining six control terms of each logic block (CT0–CT5) are used to control the asynchronous preset/reset of the flip-flops and the enable/disable of the output buffers in each macrocell. Control terms CT0 and CT1 are used to control the asynchronous preset/reset of the macrocell's flip-flop. Note that the power-on reset leaves all macrocells in the "zero" state when power is properly applied, and that the preset/reset feature for each macrocell can also be disabled. Each macrocell can choose between an asynchronous reset or an asynchronous preset function, but both cannot be simultaneously used on the same register. The global rstn function can always be used, regardless of whether or not asynchronous reset or preset control terms are enabled. Control terms CT2, CT3, CT4 and CT5 are used to enable or disable the macrocell's output buffer. Having four dedicated output enable control terms ensures that the CoolRunner™ devices are PCI compliant. The output buffers can also be always enabled or always disabled. All CoolRunner™ devices also provide a Global 3-State (gts) pin, which, when pulled high, will 3-State all the outputs of the device. This pin is provided to support "In-Circuit Testing" or "Bed-of-Nails" testing used during manufacturing.

For the macrocells in the Logic Block that are associated with I/O pins, there are two feedback paths to the LZIA: one from the macrocell, and one from the I/O pin. The LZIA feedback path before the output buffer is the macrocell feedback path, while the LZIA feedback path after the output buffer is the I/O pin feedback path. When these macrocells are used as outputs, the output buffer is enabled, and either feedback path can be used to feedback the logic implemented in the macrocell. When the I/O pins are used as inputs, the output buffer of these macrocells will be 3-Stated and the input signal will be fed into the LZIA via the I/O feedback path. In this case the logic functions implemented in the buried macrocell can be fed back into the LZIA via the macrocell feedback path. For macrocells that are not associated with I/O pins, there is one feedback path to the LZIA. Logic functions implemented in these buried macrocells are fed back into the LZIA via this path. All unused inputs and I/O pins should be properly terminated. Please refer to the section on terminations.

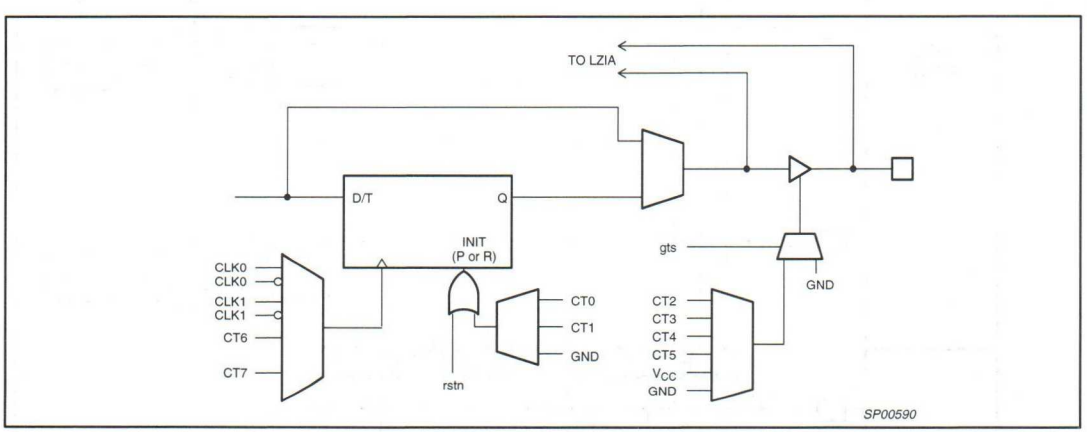


Figure 4. PZ3960 Macrocell Architecture

Simple Timing Model

Figure 5 shows the PZ3960 timing model. The PZ3960 timing model is very simple compared to the models of competing architectures. There are three main timing parameters: the pin-to-pin delay for combinatorial logic functions (t_{PD}), the input pin to register set up time (t_{SU}), and the register clock to valid output time (t_{CO}). As the model shows, timing is only dependent on whether or not you use the PLA array, and whether or not the logic function is created within a single Fast Module or uses the GZIA. The timing starts with a set time for t_{PD} and t_{SU} through the PAL array in a Fast Module, and there are fixed delays added for use of the PLA array or the GZIA. The t_{CO} timing specification never changes. For example, a combinatorial logic function of four or fewer product terms constructed from inputs within the same logic block would have a t_{PD} delay of 7.5ns. If the logic function were more than four product terms wide, the delay would be t_{PD} plus the fixed PLA delay, or 7.5 + 1.5 = 9.0ns. A function that used the PAL array and inputs from a different Fast Module would have a propagation delay of t_{PD} plus the fixed GZIA delay, or 7.5 + 4.0 = 11.5ns.

This simple timing model allows designers to determine whether or not the device will meet system timing specifications up front. In competing devices, the user is unable to determine if the design will meet system timing requirements 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, the fan-out of a signal, the varying number of X and Y routing channels used, etc. The simplicity of the PZ3960 timing model gives you pin-to-pin delay information before the design is set. Further, the timing in the PZ3960 device will not vary with place and route iterations caused by design changes. This allows the PZ3960 device to meet your timing requirements even when you make changes to the design.

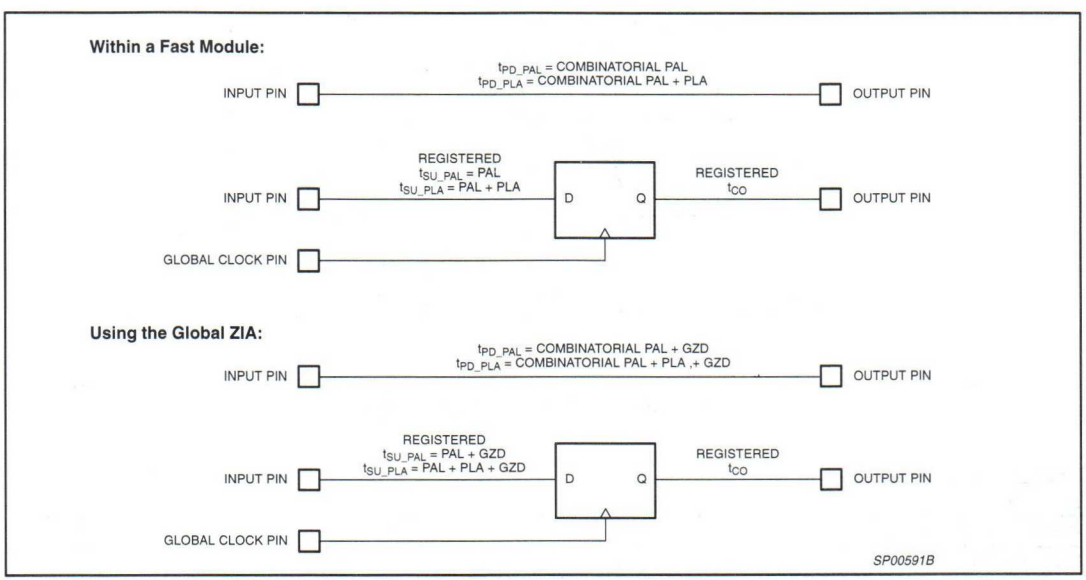


Figure 5. PZ3960 Timing Model

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 product terms 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. This also makes it possible to manufacture high density CPLDs like the PZ3960 that consume a fraction of the power of competing devices. Refer to Figure 6 and Table 2 showing the I_{DD} vs. Frequency of the PZ3960 TotalCMOS™ CPLD (data taken with 30 16-bit counters @ 3.3V, 25°C).

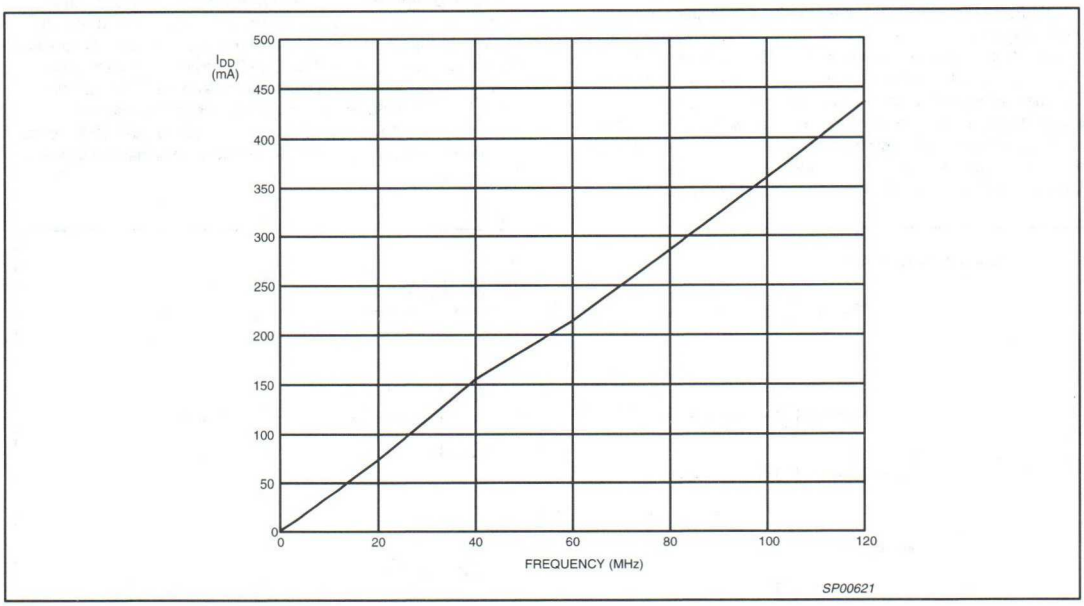


Figure 6. I_{DD} vs. Frequency @ V_{DD} = 3.3V, 25°C

Table 2. I_{DD} vs. Frequency

V_{DD} = 3.3V

FREQUENCY (MHz)	0	1	20	40	60	80	100	120
Typical I_{DD} (mA)	0.1	4.1	76.7	150.1	222.2	294.6	364	441.6

Terminations

The CoolRunner™ PZ3960C/PZ3960N CPLDs are TotalCMOS™ devices. As with other CMOS devices, it is important to consider how to properly terminate unused inputs and I/O pins when fabricating a PC board. Allowing unused inputs and I/O pins to float can cause the voltage to be in the linear region of the CMOS input structures, which can increase the power consumption of the device. It can also cause the voltage on a configuration pin to float to an unwanted voltage level, interrupting device operation.

The PZ3960C/PZ3960N CPLDs have programmable on-chip pull-down resistors on each I/O pin. These pull-downs are automatically activated by the fitter software for all unused I/O pins. Note that an I/O macrocell used as buried logic that does not have the I/O pin used for input is considered to be unused, and the pull-down resistors will be turned on. We recommend that any unused I/O pins on the PZ3960C/PZ3960N device be left unconnected.

There are no on-chip pull-down structures associated with dedicated pins used for device configuration or special device functions like global reset and global 3-state. Philips recommends that these pins be terminated consistent with the description given in Table 9. Philips recommends the use of weak pull-up and pull-down resistors for terminating these pins. These pins can be directly connected to V_{CC} or GND, but using the external pull-up resistors maintains maximum design flexibility.

When using the JTAG Boundary Scan functions, it is recommended that 10k pull-up resistors be used on the tdi, tdo, tck, and trstn pins. The tdo signal pin can be left floating unless it is connected to the tdi of another device. Letting these signals float can cause the voltage on tms to come close to ground, which could cause the device to enter JTAG/ISP mode at unspecified times.

CONFIGURATION INTRODUCTION

The Philips CoolRunner™ series are available in technologies which use non-volatile (EEPROM-based) and volatile (SRAM based) configuration memory. The functionality of the XPLA2 family of the CoolRunner™ series is defined by on-chip SRAM. The devices are configured in a manner similar to that of most FPGAs. This section describes the configuration of the PZ3960, and applies to all similarly configured devices to be produced by Philips.

Either the Philips or Minc fitter, XPLA Designer and PL-Designer, respectively, is used to generate a JEDEC file. The JEDEC file contains the configuration data, which is loaded into the PZ3960 configuration memory to control the PZ3960 functionality. This is done at power-up and/or with configure command. This section provides some of the trade-offs in selecting a configuration mode, and provides debug hints for configuration problems.

There are several different methods of configuring the PZ3960. The mode used is selected using the mode select pins. There are three basic configuration methods: master, slave, and peripheral. The configuration data can be transmitted to the PZ3960 serially or in parallel bytes. As a master, the PZ3960 generates the clock and control signals to strobe configuration data into the PZ3960. As a slave device, a clock is generated externally, and provided into the PZ3960's cclk pin. In the peripheral mode, the PZ3960 interfaces as a microprocessor peripheral. Table 3 lists the configuration modes.

Design Flow Overview

Figure 7 is a diagram of the steps used in configuring the PZ3960. The development system is used to generate configuration data in the JEDEC file. Using the <design>.jed file, there are two general methods of configuring the PZ3960. The utility **download** can load the configuration data from a PC or workstation hard disk into the PZ3960. This is one of the methods used on the PZ3960 evaluation board. Alternately, the PZ3960 can be loaded from non-volatile ICs such as serial or parallel EEPROMs.

Table 3. Configuration Modes

M2	M1	M0	cclk	CONFIGURATION MODE	DATA FORMAT
0	0	0	Output	Master serial	Serial
0	0	1	Input	Slave parallel	Parallel
0	1	0	Reserved		
0	1	1	Input	Synchronous peripheral	Parallel
1	0	0	Output	Master parallel – up	Parallel
1	0	1	Reserved		
1	1	0	Output	Master parallel – down	Parallel
1	1	1	Input	Slave serial	Serial

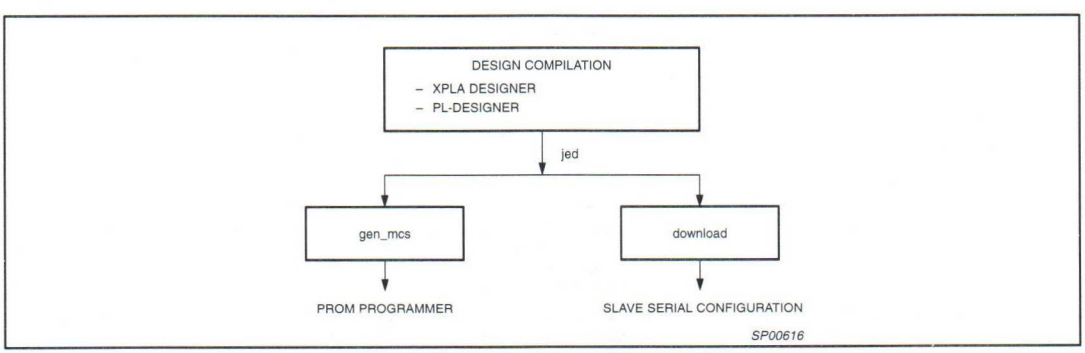


Figure 7. Design flow

PZ3960 STATES OF OPERATION

Prior to becoming operational, the PZ3960 goes through a sequence of states, including initialization, configuration, and start-up. This section discusses these three states. In the master configuration modes, the PZ3960 is the source of configuration clock (cclk). In this mode, the Initialization state is extended to ensure that, in daisy-chain operation, all daisy-chained slave devices are ready.

When configuration is initiated, a counter in the PZ3960 is set to 0 and begins to count configuration clock cycles applied to the PZ3960. As each configuration data frame is supplied to the PZ3960, it is internally assembled into data words. Each data word is loaded into the internal configuration memory. The configuration loading process is complete when the internal length count equals the loaded length count in the length count field, and the required end of configuration frame is written.

All configuration I/Os used as inputs operate with TTL-level input thresholds during configuration. All I/Os that are not used during the configuration process are 3-Stated with internal pull-downs. During configuration, registers are reset. The combinatorial logic begins to function as the PZ3960 is configured. Figure 8 shows the flow between the initialization, configuration, and start-up states. Figure 9 gives the general timing information for configuring the device.

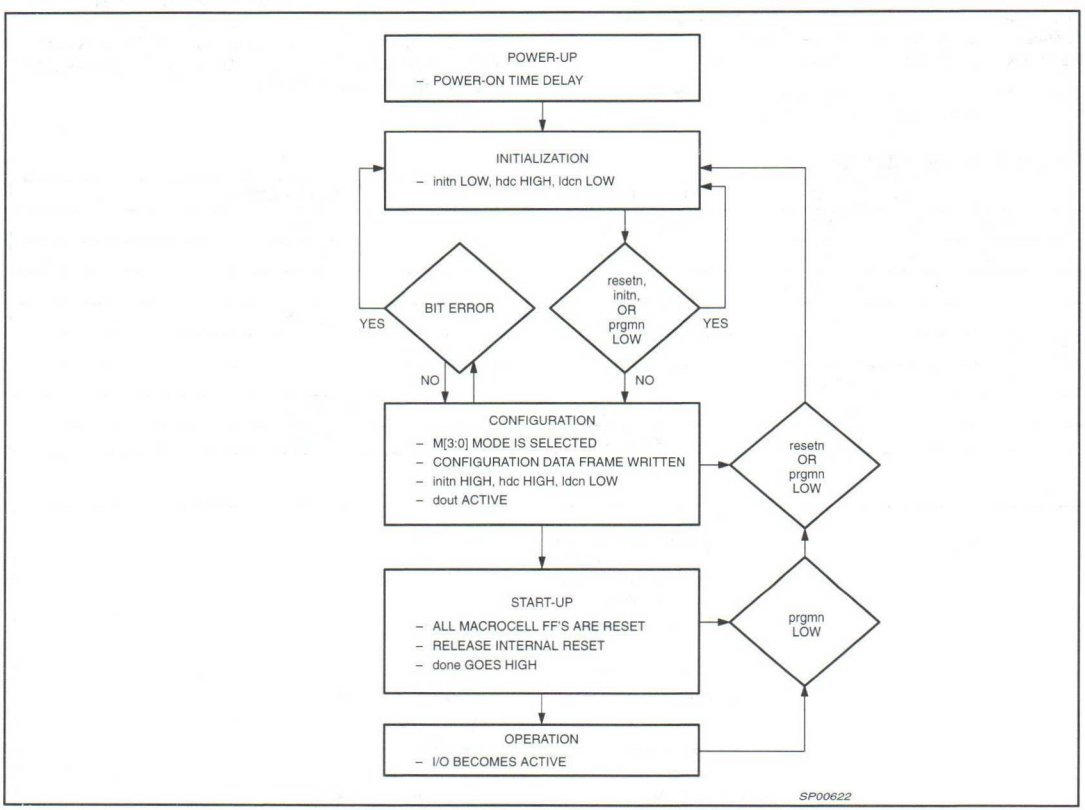


Figure 8. Flow chart of initialization, configuration, and operating states

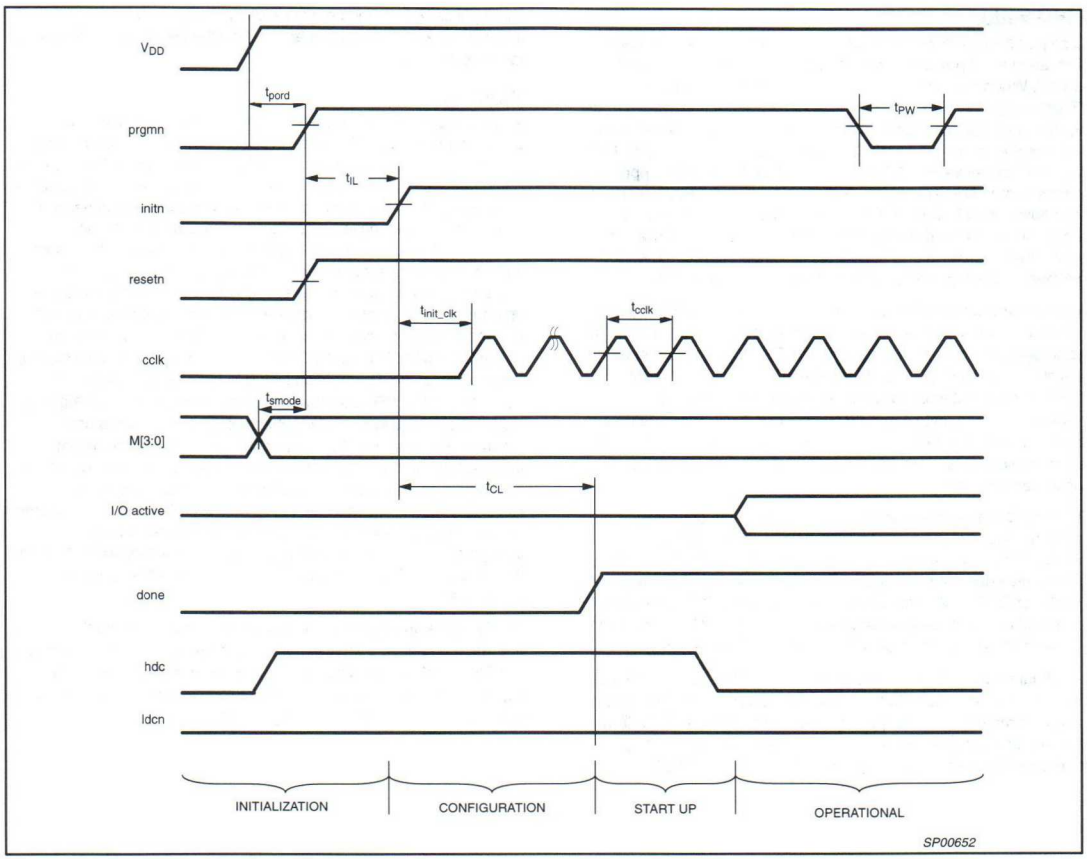


Figure 9. General configuration mode timing diagram

Initialization

Upon power-up, the device goes through an initialization process. First, an internal power-on-reset circuit is triggered when power is applied. When V_{DD} reaches the voltage at which portions of the PZ3960 begin to operate (1.5V), the configuration pins are set to be inputs or outputs based on the configuration mode, as determined by the mode select inputs M[2:0]. A time-out delay is initiated when V_{DD} reaches between 1.0V and 2.0V to allow the power supply voltage to stabilize. The initn and done outputs are low. At power-up, if the power supply does not rise from 1.0V to V_{DD} in less than 25ms, the user should delay configuration by inputting a low into prgmn, initn, or resetn until V_{DD} is greater than the recommended minimum operating voltage (2.75V for commercial devices).

When initialization is complete, the active-low initialization signal initn is released and must be pulled high by an external resistor. To synchronize the configuration of multiple PZ3960s, one or more initn pins should be wire-ANDed. If initn is held low by one or more PZ3960s or an external device (the PZ3960 remains in the initialization state), initn can be used to signal that the PZ3960s are not yet initialized. After initn goes high for two internal clock cycles, the mode select lines are sampled and the PZ3960 enters the configuration state.

The High During Configuration (hdc), Low During Configuration (ldcn), and done signals are active outputs in the PZ3960's initialization and configuration states. hdc, ldcn, and done can be used to provide control of external logic signals such as reset, bus enable, or EEPROM enable during configuration. For master parallel configuration modes, these signals provide EEPROM enable control and allow the data pins to be shared with user logic signals.

If configuration has begun, an assertion of resetn or prgmn initiates an abort, returning the PZ3960 to the initialization state. The resetn and/or prgmn pins must be pulled back high before the PZ3960 will enter the configuration state. During the start-up and operating states, only the assertion of prgmn causes a re-configuration.

During initialization and configuration, all I/O's are 3-stated and the internal weak pull-downs are active. See the section on terminations for more information.

Start-up

After configuration, the PZ3960 enters the start-up phase. This phase is the transition between the configuration and operational states. This transition occurs within three cclk cycles of the done pin going high (it is acceptable to have additional cclk cycles beyond the three required). The system design task in the start-up phase is to ensure that multi-function pins (see pin function on page 34) transition from configuration signals to user definable I/Os without inadvertently activating devices in the system or causing bus contention. The done signal goes high at the beginning of the start up phase, which allows configuration sources to be disconnected so that there is no bus contention when the I/Os become active. In addition to controlling the PZ3960 during start-up, additional start-up techniques to avoid contention include using isolation devices between the PZ3960 and other circuits in the system, re-assigning I/O locations, and keeping I/Os 3-stated until contentions are resolved. For example, Figure 10 shows how to use the global tri-state (gts) signal to avoid signal contention when the mode select pins (M3...M0) are used as I/O after configuration is finished. Holding gts high until after the mode pins are disconnected from the driving source allows pins M3 through M0 to transition from configuration pins to user definable I/O without signal contention. In this case, the I/O become active a t_{gtsr} delay after the gts pin is pulled low.

The flip-flops are reset one cycle after done goes high so that operation begins in a known state. The done outputs from multiple PZ3960s can be wire ANDed and used as an active-high ready signal, to disable PROMs with active-low enable(s), or to reset to other parts of the system (see Figure 28).

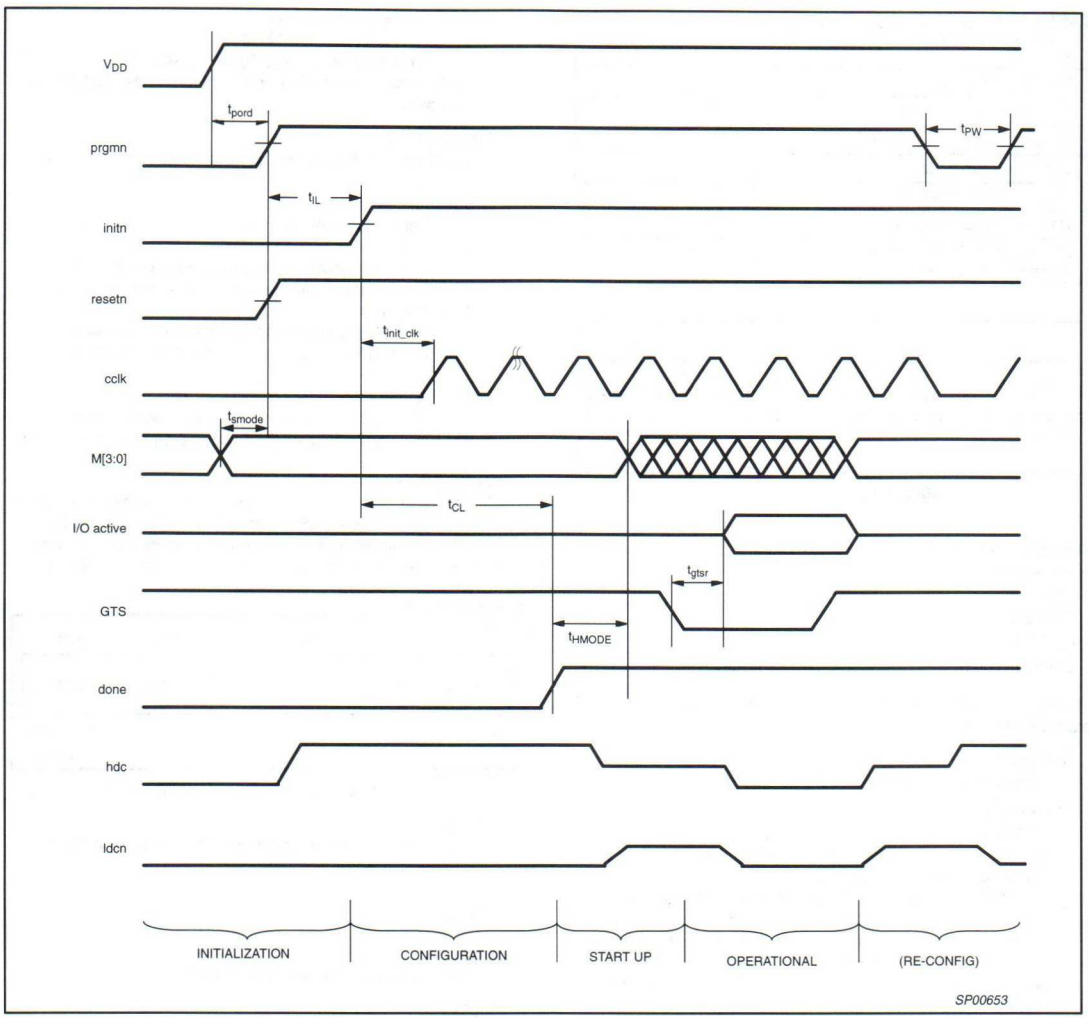


Figure 10. Using gts signal with power up to avoid signal contention with mode select pins

CONFIGURATION DATA FORMAT OVERVIEW

The PZ3960 functionality is determined by the state of internal configuration RAM. This section discusses the configuration data format, and the function of each field in configuration data packets.

Configuration Data Packets

Configuration of the PZ3960 is done using configuration packets. The configuration packet is shown in Figure 11. The data packet consists of a header and a data frame. There are five type of data frames. The header is shifted into the device first, followed by one data frame. Configuration of a single PZ3960 requires 1010 data packets, one for each address. All preceding data must contain only 1s. Once a device is configured, it re-transmits data of any polarity. Before and during configuration, all data re-transmitted out the daisy-chain port (dout) are 1s.

The ordering of the data packets may be random, but they cannot be mixed with other devices' data packets. Alignment bits are not required between data packets. If used, alignment bits must be included in the length count, and they must be at least 2 bits long.

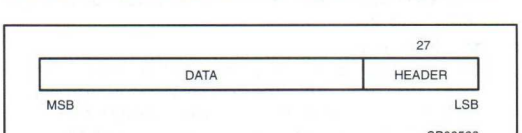


Figure 11. Data Packet

Table 4. Configuration Frame Size

DEVICE	PZ3960
Number of frames	1010
Data bits/standard frame	560
Data bits/compressed frame	14
Data bits/user_code frame	560
Data bits/isc_code frame	560
Data bits/security frame	559
Maximum configuration data— # bits/frame × # frames	565600

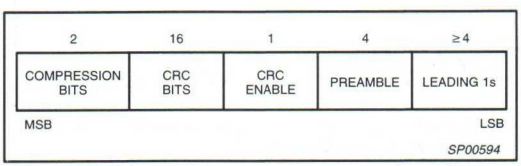


Figure 12. 27-bit Header

The header is fixed and consists of five fields:
- Leading 1s,
- Preamble,
- CRC Enable,
- CRC Bits,
- Compression Bits.

The leading 1s enter the device first. The following is a description of each field in the header.

Leading 1s:
This is a four or greater bit field consisting of 1s.

Preamble:
This is a four bit field which indicates the start of a frame when the least significant bit of the preamble is a 0. There are two valid preambles:
0010 – indicates that the data packet configures the device receiving the 0010 preamble)
0100 – indicates end of configuration of the device receiving the 0100 preamble
All other values of the preamble field force configuration of the entire system to restart.

The segments CRC Enable, CRC Bits, and Compression Bits are valid only if the Preamble field is 0010.

Cyclic Redundancy Check (CRC) Enable:
In this single bit field, a 0 disables CRC checking of the data stream. If the CRC is disabled the 16 bit CRC field must be the default described below. A 1 enables CRC error checking of the data stream.

CRC Error Checking:
The CRC field is a 16 bit field. The default value is 1010_1010_1010_1010. The calculated value is from data, address, stop bit, and first alignment bit (starting with crc_reg[15:0] = [0]). Using verilog operators, the crc is calculated as:

```
crc_reg[14:2] <= cr_reg[14:2] << 1;
cr_reg[2] <= cr_reg[15]^din^cr_reg[1];
cr_reg[1] <= cr_reg[0];
cr_reg[0] < cr_reg[15]^din;
cr_reg[15] <= cr_reg[15]^din^cr_reg[14];
```

If a CRC error is detected, configuration is halted and must be restarted.

Compression Bits:
This 2-bit field defines the use of compression of the data packets.
00 – Standard mode:
The data packet contains both address and data
01 – Reset mode:
The data packet contains only the address field. This pattern causes the configuration register to be reset.
10 – Hold mode:
The data packet contains only the address field. This pattern causes the configuration register to hold its value.
11 – Set mode:
The data packet contains only the address field. This pattern causes the configuration register to be set.

Data Frames

The five types of data frames are standard, compressed, user_code, isc_code, and security. All fields must be completely filled, with 1s used to fill unused bits. The security frame must be the last frame sent to a device. The definition of each frame is described below:

Standard frame

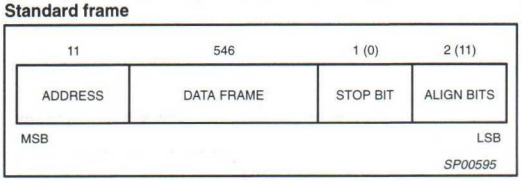


Figure 13. Standard Frame

Address:
This is an 11 bit filed for providing 1011 (1008 SRAM plus 3 user) addresses.

Data:
546 bit field.

Stop bit:
This is a one bit field which must be 0.

Align bit:
This is a two bit field which must be 11.

Compressed frame

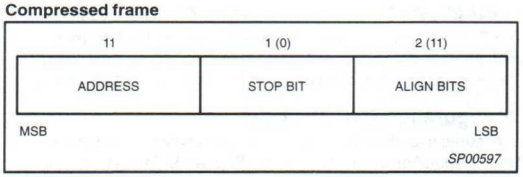


Figure 14. Compressed Frame

The compressed frame contains no data.

User code frame
The user code is located at address 1008D.

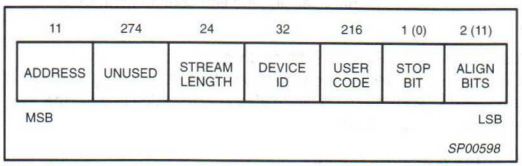


Figure 15. User code Frame

Stream length:
This is a 24 bit field containing the length of the data stream transmitted to configure all of the devices in the daisy chain. This field is only used by a PZ3960 if it is in the master mode.

Device ID:
This is a 32-bit field containing PZ3960 device ID:
492 SBGA: 0000_001_001_101000_1_000_00000010101_1

User code:
This is a 216 bit field reserved for user information.

ISC code frame
The isc_code address is 1009.

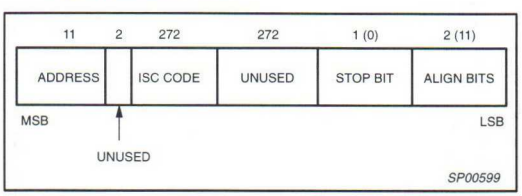


Figure 16. ISC Frame

The ISC frame allows the user to write an ISC code to the device.

Security frame

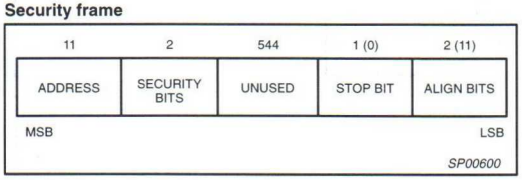


Figure 17. Security Frame

Security bits:
This is a two bit field specifying the level of security.
00 – Unlimited readback allowed.
01 – Readback operation allowed once.
10 – Readback operation allowed once.
11 – Readback operation is disabled.

Re-configuration

To reconfigure the PZ3960 when the device is operating in the system, a low pulse is input into prgmn. The configuration data in the PZ3960 is cleared, and the I/Os not used for configuration are 3-Stated. The PZ3960 then samples the mode select inputs and begins re-configuration. When configuration is compete, done is released, allowing it to be pulled high.

Bit Stream Error Checking

There are three different types of bit stream error checking in the PZ3960:
- ID frame,
- Frame alignment, and
- CRC checking.

An optional ID data frame can be sent to a specified address in the PZ3960. This ID Frame contains a unique code which is compared with the value in the PZ3960 ID register. Any differences are flagged as an ID error.

CRC checking is done on each frame if enabled by setting the CRCen bit in the header. If there is an error, a CRC error is flagged. When an error occurs, the PZ3960 is forced into the initialization state, forcing initn low. The PZ3960 remains in this state until either the resetn or prgmn pins is asserted.

PZ3960 CONFIGURATION MODES

The method for configuring the PZ3960 is selected by the M0, M1, and M2 inputs. The M3 input is used to select the frequency of the internal oscillator, which is the source for cclk in master configuration modes. The nominal frequencies of the internal oscillator are 1.25MHz and 10MHz. The 1.25MHz frequency is selected when the M3 input is unconnected or driven to a high state.

Master Serial Mode

In the master serial mode, the PZ3960 loads the configuration data from an external serial ROM. The configuration data is either loaded automatically at start-up or on a command to reconfigure. Serial EEPROMs from Altera, Atmel, Lucent, Microchip, and Xilinx can be used to configure the PZ3960 in the master serial mode. This provides a simple four-pin interface in an eight-pin package. Serial EEPROMs are available in 32K, 64K, 128K, 256K, and 1M bit densities.

Configuration in the master serial mode can be done at power-up and/or upon a configure command. The system or the PZ3960 must activate the serial EEPROM's $\overline{\text{RESET}}$/OE and $\overline{\text{CE}}$ inputs. At power-up, the PZ3960 and serial EEPROM each contain internal power-on reset circuitry which allows the PZ3960 to be configured without the system providing an external signal. The power-on reset circuitry causes the serial EEPROMs' internal address pointer to be reset. After power-up, the PZ3960 automatically enters its initialization phase.

The serial EEPROM/PZ3960 interface used depends on such factors as the availability of a system reset pulse, availability of an intelligent host to generate a configure command, whether a single serial EEPROM is used or multiple serial ROMs are cascaded, whether the serial EEPROM contains a single or multiple configuration programs, etc.

Data is read into the PZ3960 sequentially from the serial ROM. The DATA output from the serial EEPROM is connected directly into the din input of the PZ3960. The cclk output from the PZ3960 is connected to the CLOCK input of the serial EEPROM. During the configuration process, cclk clocks one data bit into the PZ3960 on each rising edge.

Since the data and clock are direct connects, the PZ3960/serial EEPROM interface task is to use the system or PZ3960 to enable the $\overline{\text{RESET}}$/OE and $\overline{\text{CE}}$ of the serial EEPROM(s). There are several methods for enabling the serial ROM's $\overline{\text{RESET}}$/OE and $\overline{\text{CE}}$ inputs. The serial EEPROM's $\overline{\text{RESET}}$/OE is programmable to function with

resetn active-high and $\overline{OE}$ active-low, or resetn active-low and OE active-high.

In Figure 18, three serial EEPROMs are cascaded to configure a PZ3960. The host generates a 500ns low pulse into the PZ3960's prgmn input and into the serial EEPROMs' $\overline{RESET}$/OE input, which has been programmed to function with resetn active-low and OE active-high. The PZ3960 done is routed to the $\overline{CE}$ pin. The low on done enables the serial EEPROMs. At the completion of configuration, the high on the PZ3960's done disables the EEPROM(s).

When configuration data requirements exceed the capacity of a single serial EEPROM, multiple serial EEPROMs can be cascaded to support the configuration of a single (or multiple) PZ3960(s). After the last bit from the first serial ROM is read, the serial ROM outputs $\overline{CEO}$ low and 3-States the DATA output. The next serial ROM recognizes the low on $\overline{CE}$ input and outputs configuration data on the DATA output. After configuration is complete, the PZ3960's done output into $\overline{CE}$ disables the serial EEPROMs.

In applications in which a serial EEPROM stores multiple configuration programs, the subsequent configuration program(s) are stored in EEPROM locations that follow the last address for the previous configuration program. The user must ensure that a high output on the PZ3960 done signal does not reset the serial EEPROM address pointer, causing the first configuration to be reloaded.

Contention on the PZ3960's din pin must be avoided. During configuration, din receives configuration data. After configuration, it is a user I/O at start-up. An alternative is to use ldcn to drive the serial EEPROMs' $\overline{CE}$ pin.

Master Parallel Mode

The master parallel configuration mode is generally used to interface to industry-standard byte-wide memory such as 256K and larger EEPROMs. Figure 20 provides the interface for master parallel mode. The PZ3960 outputs a 20-bit address on A[19:0] to memory and reads one byte of configuration data on the rising edge of rclk. The parallel bytes are internally serialized starting with the least significant bit, D0. There are two parallel master modes: master up, and master down. In master up, the starting memory address is 00000 Hex and the PZ3960 increments the address for each byte loaded. In master down, the starting memory address is FFFFFH and the PZ3960 decrements the address.

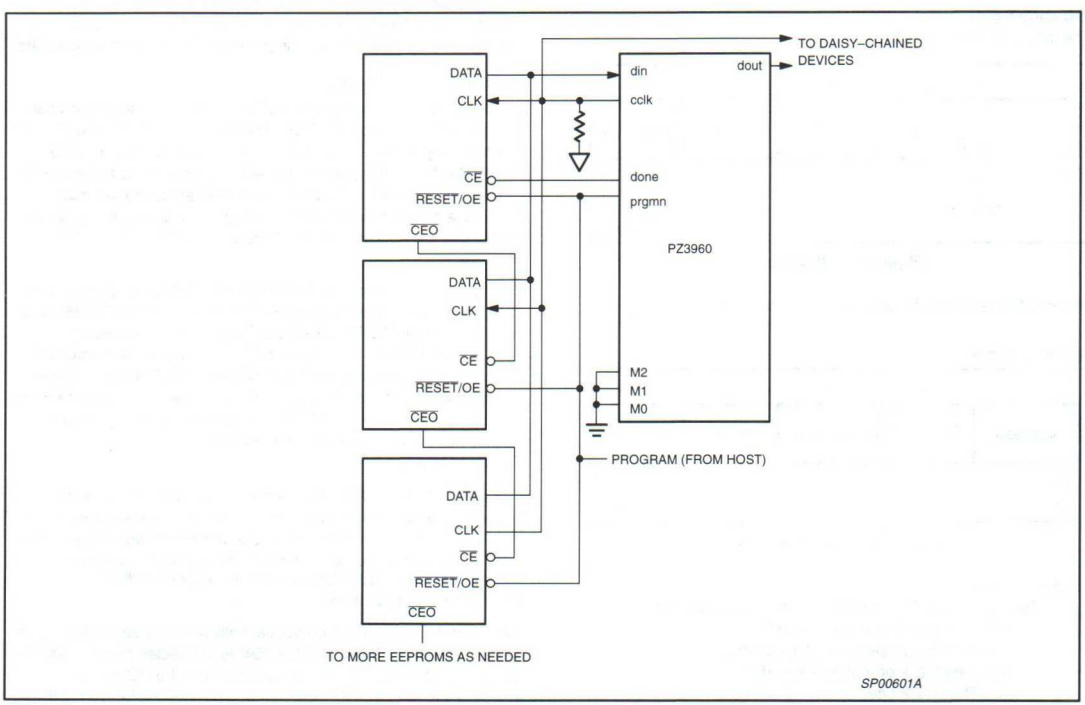


Figure 18. Master Serial Configuration

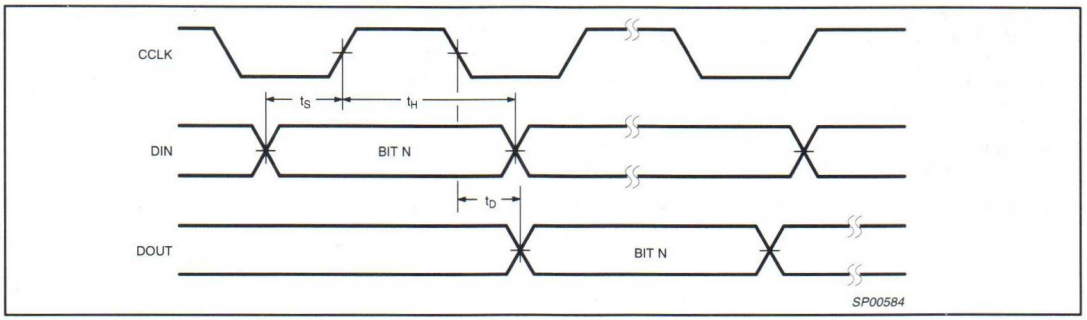


Figure 19. Master Serial Configuration Mode Timing Diagram

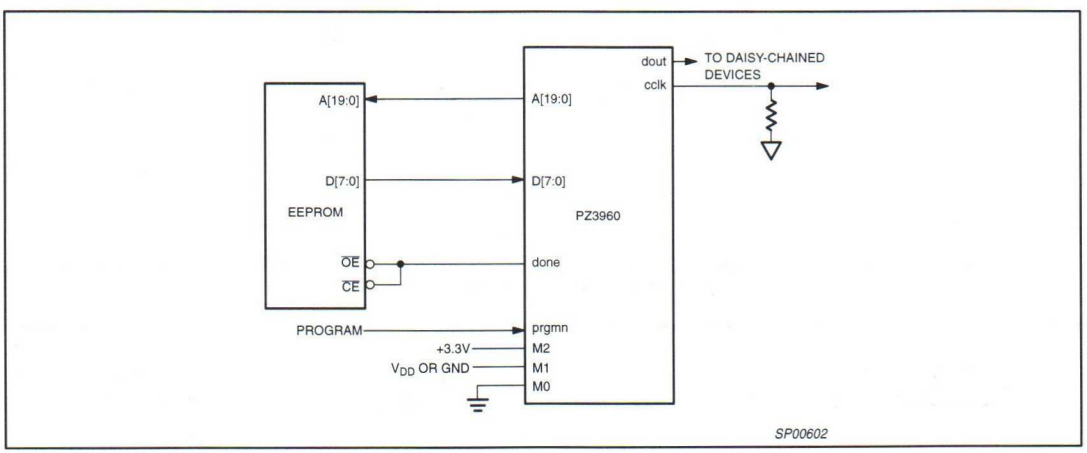


Figure 20. Master Parallel Configuration

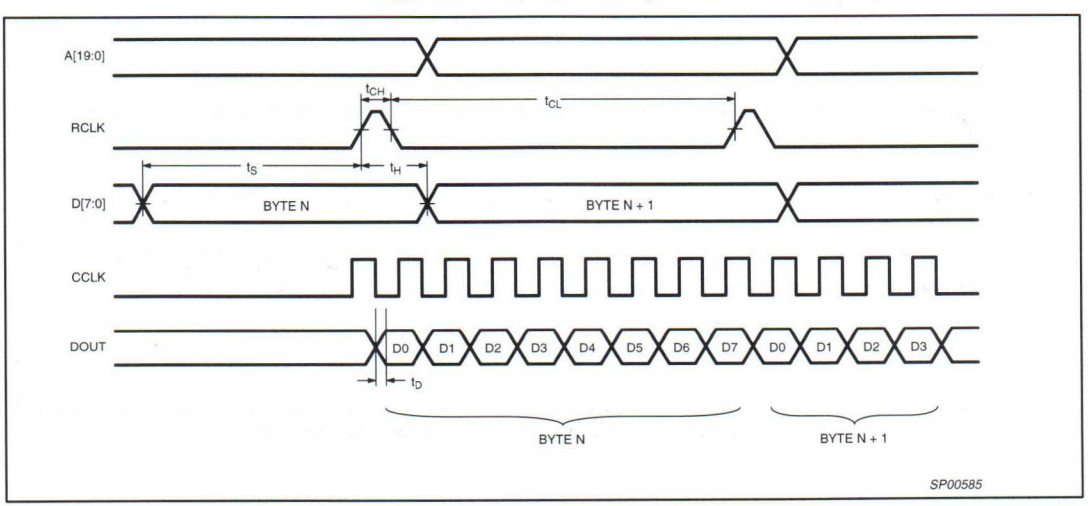


Figure 21. Master Parallel Configuration Mode Timing Diagram

Synchronous Peripheral Mode

In the synchronous peripheral mode, byte-wide data is input into D[7:0] on the rising edge of the cclk input. The first data byte is clocked in on the second cclk after initn goes high. Subsequent data bytes are clocked in on every eighth rising edge of cclk. The rdy_busyn signal is an output which acts as an acknowledge. rdy_busyn goes high one cclk after a byte of data is clocked in on D[7:0] and returns low one cclk cycle later. The process repeats until all of the data is loaded into the PZ3960. The serial data begins shifting out on dout 0.5 cycles after the parallel data was loaded. It requires additional cclks after the last byte is loaded to complete the shifting. Figure 22 shows the interface for synchronous peripheral mode.

As with master modes, the peripheral modes can be used as the lead PZ3960 for daisy-chained PZ3960s.

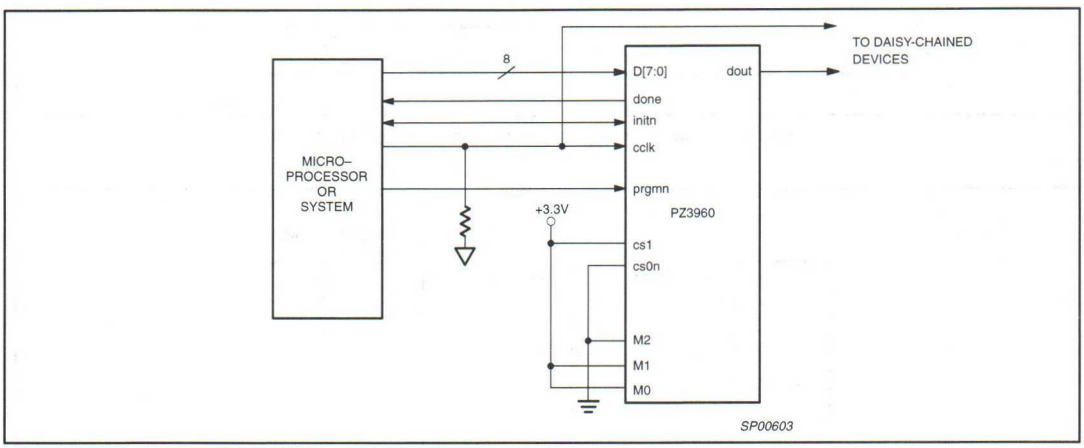


Figure 22. Synchronous Peripheral Configuration

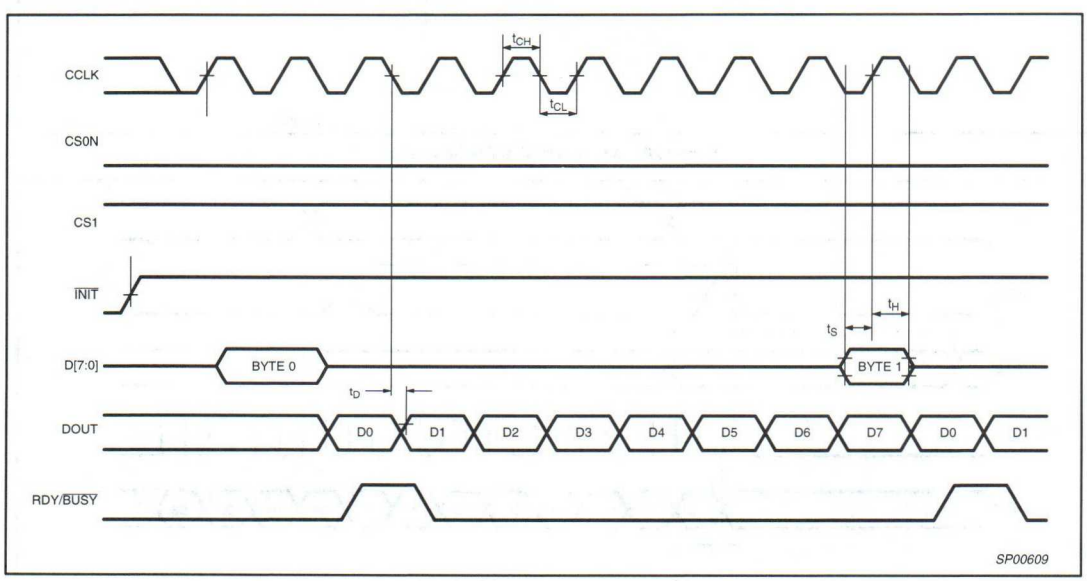


Figure 23. Synchronous Peripheral Configuration Mode Timing Diagram

Slave Serial Mode

The slave serial mode is primarily used when multiple PZ3960s are configured in a daisy-chain. The serial slave serial mode is also used on the PZ3960 evaluation board, which interfaces to the download cable. A device in the slave serial mode can be used as the lead device in a daisy-chain. Figure 24 shows the interface for the slave serial configuration mode.

The configuration data is provided into the PZ3960's din input synchronous with the configuration clock cclk input. After the PZ3960 has loaded its configuration data, it re-transmits incoming configuration data on dout. When used in daisy-chained operation, cclk is routed into all slave serial mode devices in parallel.

Multiple slave PZ3960s can be loaded with identical configurations simultaneously. This is done by loading the configuration data into the din inputs in parallel.

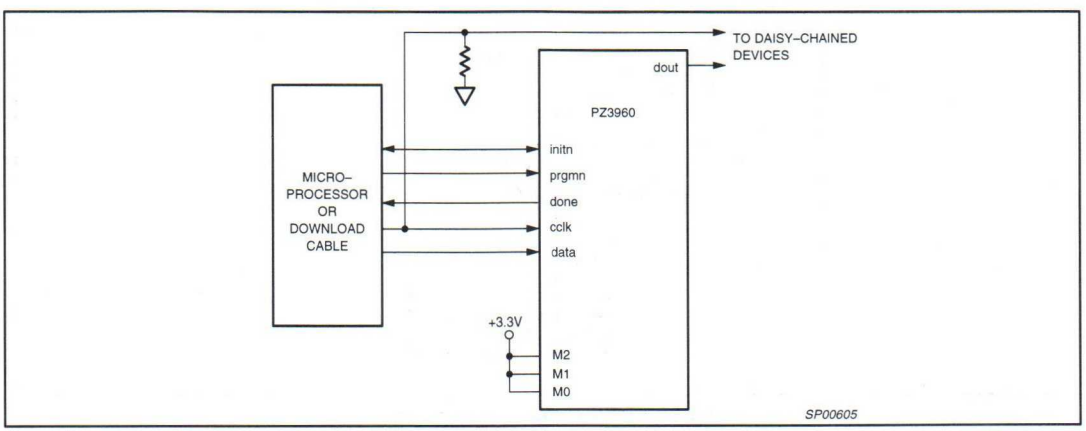


Figure 24. Slave Serial Configuration Schematic

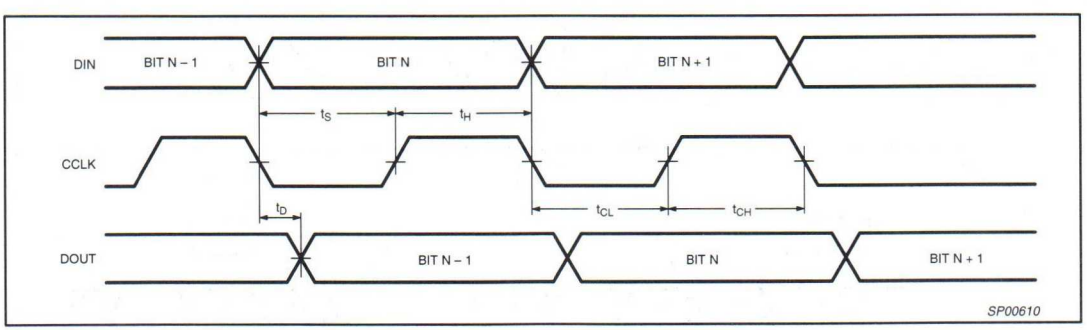


Figure 25. Slave Serial Configuration Mode Timing Diagram

Slave Parallel Mode

The slave parallel mode is essentially the same as the synchronous peripheral mode, except that cs1 and cs0n do not need to be driven, and there is no rdy_bsyn output. As in the synchronous peripheral mode, byte-wide data is input into D[7:0] on the rising edge of the cclk input. The first data byte is clocked in on the second cclk after initn goes high. Subsequent data bytes are clocks in on every eighth rising edge of cclk. The process repeats until all of the data is loaded into the PZ3960. The serial data begins shifting out on dout 0.5 cycles after the parallel data was loaded. It requires additional cclks after the last byte is loaded to complete the shifting. Figure 26 shows the interface for slave parallel mode.

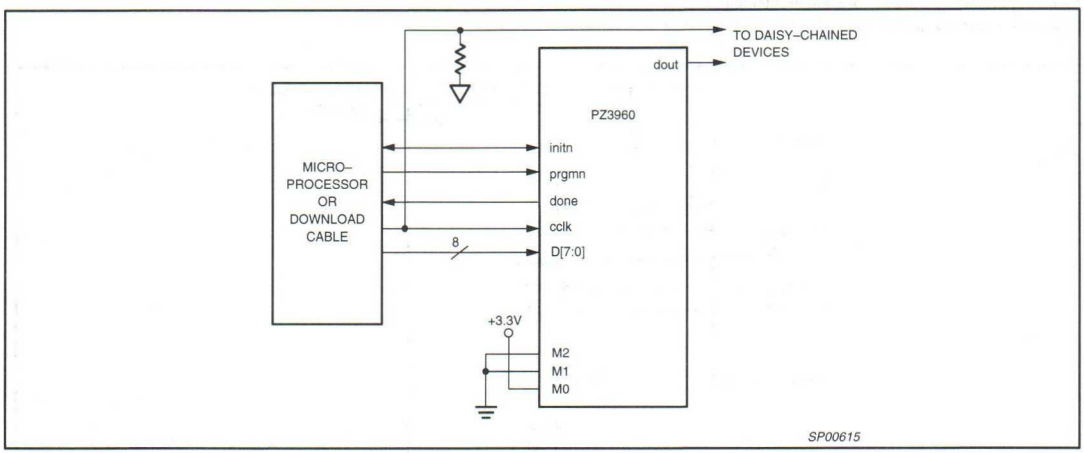


Figure 26. Slave Parallel Configuration Schematic

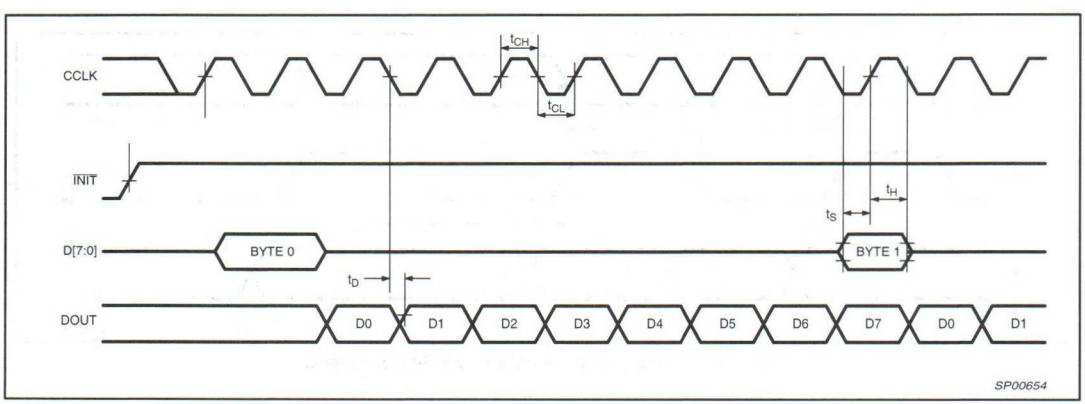


Figure 27. Slave Parallel Configuration Mode Timing Diagram

DAISY CHAIN OPERATION

Multiple PZ3960s can be configured by using a daisy-chain of PZ3960s. Daisy-chaining uses a lead PZ3960 and one or more PZ3960s configured in slave serial mode. The lead PZ3960 can be configured in any mode, but master parallel is typically used. Figure 28 shows the connections for loading multiple PZ3960s in a daisy-chain configuration.

Daisy-chained PZ3960s are connected in series. An upstream PZ3960 which has received the preamble outputs a high on dout until it has received the appropriate number of data frames. This ensures that downstream PZ3960s do not receive frame start bits. After loading and re-transmitting the preamble to a daisy-chain of slave devices, the lead device loads its configuration data frames. The loading of configuration data continues after the lead device has received its configuration data if the lead device's internal frame bit counter has not reached the length count. When the configuration RAM is full and the number of bits received is less than the length count field, the PZ3960 shifts data out on dout.

The generation of cclk for the daisy-chained devices which are in slave serial mode differs depending on the configuration mode of the lead device. A master parallel mode device uses its internal timing generator to produce an internal cclk at eight times its memory address rate (rclk). If the lead device is configured in either synchronous peripheral, slave serial mode, or slave parallel mode, cclk is routed to the lead device and to all of the daisy-chained devices. The configuration data is read into din of slave devices on the positive edge of cclk, and shifted out dout on the negative edge of cclk.

The development software can create a composite configuration file for configuring daisy-chained PZ3960s. The configuration data consists of multiple concatenated data packets. As seen in Figure 28, the initn pins for all of the PZ3960s are connected together. This is required to guarantee that power-up and initialization function correctly. In general, the done pins for all of the PZ3960s are also connected together as shown to guarantee that all of the PZ3960s enter the start-up state simultaneously. This may not be required, depending upon the start-up sequence desired.

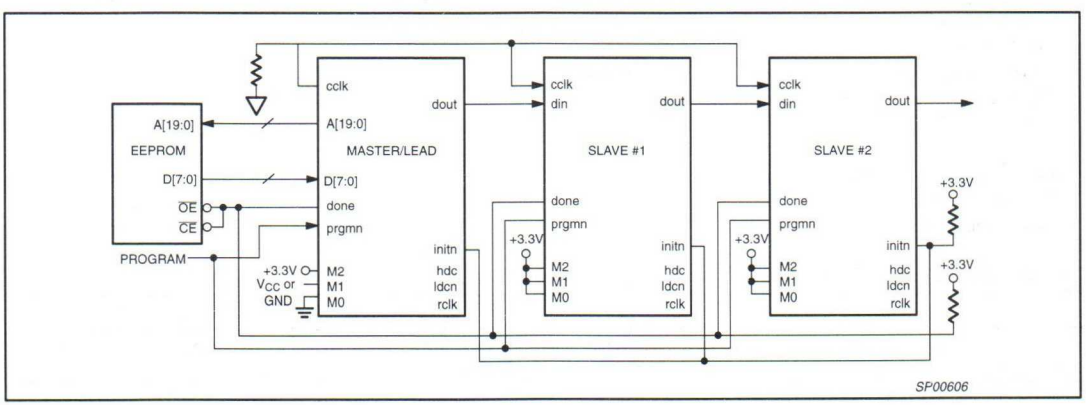


Figure 28. Daisy-chain Schematic

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 PZ3960'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 PZ3960, the TAP Port includes five pins (refer to Table 5) described in the JTAG specification: tck, tms, tdi, tdo, and trstn. These pins should be connected to an external pull-up resistor to keep the JTAG signals from floating when they are not being used.

Table 6 defines the dedicated pins used by the mandatory JTAG signals for the PZ3960.

The JTAG specifications define 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 PZ3960 is defined in Table 7. By supporting this set of low-level commands, the PZ3960 allows execution of all high-level boundary-scan commands.

Table 5. 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.
trstn	Test Reset	Forces TAP controller to test logic reset state. This signal is active low.

Table 6. PZ3960 JTAG Pinout by Package Type

DEVICE	(PIN NUMBER / MACROCELL #)				
	tck	tms	tdi	tdo	trstn
PZ3960 492 pin PBGA	P4	N4	P1	P3	N3

Table 7. PZ3960 Low-Level JTAG Boundary-Scan Commands

INSTRUCTION (Instruction Code) *Register Used*	**DESCRIPTION**
SAMPLE/PRELOAD (00010) *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 (00000) *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 (11111) *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 (00001) *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 (00101) *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.

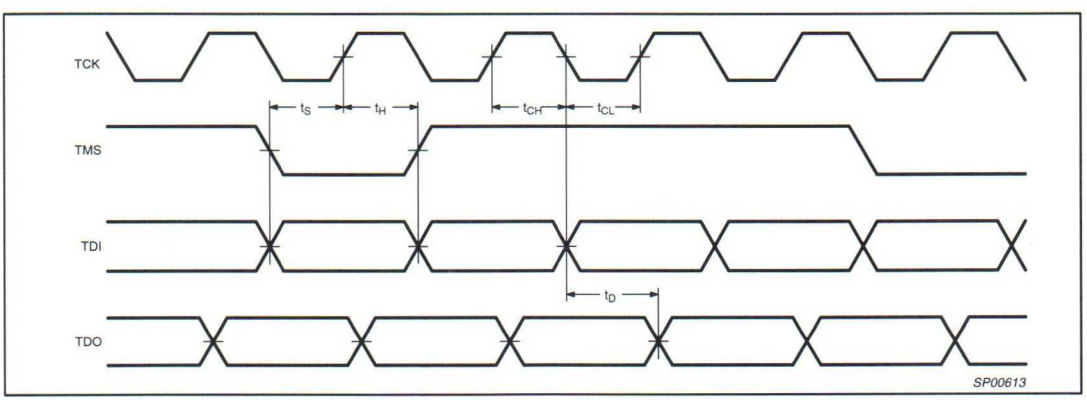


Figure 29. Boundary Scan Timing Diagram

Table 8. Boundary scan timing characteristics

SYMBOL	PARAMETER	MIN	MAX	UNIT
t_S	tdi/tms to tck setup time	20	–	ns
t_H	tdi/tms from tck hold time	0	–	ns
t_{CH}	tck high time	50	–	ns
t_{CL}	tck low time	50	–	ns
f_{TCK}	tck frequency	–	10	MHz
t_D	tck to tdo delay	–	20	ns

ABSOLUTE MAXIMUM RATINGS[1]

SYMBOL	PARAMETER	MIN	MAX	UNIT
V_{DD}	Supply voltage	–0.5	4.6	V
V_{IN}	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
T_J	Junction temperature range	–40	150	°C
T_{STG}	Storage temperature range	–65	150	°C

NOTE:
1. Stresses above these 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 temperature range: V_{DD} = 3.0V to 3.6V; 0°C < T_{amb} < 70°C

SYMBOL	PARAMETER	TEST CONDITIONS	MIN	MAX	UNIT
V_{IH}	Input high voltage	V_{DD} = 3.6V	2.0	V_{DD}+0.3	V
V_{IL}	Input low voltage	V_{DD} = 3.0V	–0.3	0.8	V
V_{OH}	Output high voltage	V_{DD} = 3.0V; I_{OH} = –8mA	2.4	–	V
V_{OL}	Output low voltage	V_{DD} = 3.0V; I_{OH} = 8mA	–	0.4	V
I_I	Input leakage current	V_{DD} = 3.6V; 0 < V_{IN} < V_{DD}	–10	10	µA
I_{DDSB}	Standby current	T_{amb} = 25°C; no output loads, inputs at V_{DD} or V_{SS}.	–	100	µA
C_{IN}	Input capacitance	T_{amb} = 25°C; V_{DD} = 3.3V; f = 1MHz	–	10	pF
C_{IO}	I/O capacitance	T_{amb} = 25°C; V_{DD} = 3.3V; f = 1MHz	–	10	pF
C_{CLK}	Clock pin capacitance	T_{amb} = 25°C; V_{DD} = 3.3V; f = 1MHz	–	12	pF
R_{DONE}	done pull-up resistor	V_{DD} = 3.0 V; V_{IN} = 0 V	10	30	kΩ
R_{PD}	Unused I/O pull-down resistor	V_{DD} = 3.6V; V_{IN} = V_{DD}	100	400	kΩ

AC ELECTRICAL CHARACTERISTICS FOR COMMERCIAL GRADE DEVICES

Commercial temperature range: V_{DD} = 3.0V to 3.6V; 0°C < T_{amb} < 70°C

SYMBOL	PARAMETER	MIN	MAX	UNIT
Timing requirements				
t_{CL}	Clock LOW time	2.5		ns
t_{CH}	Clock HIGH time	2.5		ns
t_{SU_PAL}	PAL setup time (Global clock)	4.0		ns
t_{SU_PLA}	PLA setup time (Global clock)	5.5		ns
t_{SU_XOR}	XOR setup time (Global clock)	6.5		ns
t_H	Hold time (Global clock)		0	ns
Output characteristics				
t_{PD_PAL}	Input to output delay through PAL		7.5	ns
t_{PD_PLA}	Input to output delay through PLA		9.0	ns
t_{PD_XOR}	Input to output delay through XOR		10.0	ns
t_{PDF_PAL}	Input (or feedback node) to internal feedback node delay time through PAL		4.0	ns
t_{PDF_PLA}	Input (or feedback node) to internal feedback node delay time through PLA		5.5	ns
t_{PDF_XOR}	Input (or feedback node) to internal feedback node delay time through XOR		6.5	ns
t_{CF}	Global clock to feedback delay		2.5	ns
t_{CO}	Global clock to out delay		6.0	ns
t_{CS}	Clock skew (variance for switching outputs with common global clock)		1.0	ns
f_{MAX1}	Maximum flip-flop toggle rate $\left(\frac{1}{t_{CL} + t_{CH}}\right)$	200		MHz
f_{MAX2}	Maximum internal frequency $\left(\frac{1}{t_{SU_PAL} + t_{CF}}\right)$	154		MHz
f_{MAX3}	Maximum external frequency $\left(\frac{1}{t_{SU_PAL} + t_{CO}}\right)$	100		MHz
t_{BUFF}	Output buffer delay (fast)		3.5	ns
t_{SSR}	Slow slew rate incremental delay		8.0	ns
t_{EA}	Output enable delay		8.0	ns
t_{ER}	Output disable delay[1]		8.0	ns
t_{GTSH}	Global 3-State enable		40.0	ns
t_{GTSR}	Global 3-State disable		40.0	ns
t_{RR}	Input to register reset		10.5	ns
t_{RP}	Input to register preset		10.5	ns
t_{GRR}	Global reset to register reset		40	ns
t_{GZIA}	Global ZIA delay		4.0	ns

NOTE:

1. Output C_L = 5.0pF.

DC ELECTRICAL CHARACTERISTICS FOR INDUSTRIAL GRADE DEVICES

Industrial temperature range: V_{DD} = 3.0V to 3.6V; –40°C < T_{amb} < 85°C

SYMBOL	PARAMETER	TEST CONDITIONS	MIN	MAX	UNIT
V_{IH}	Input high voltage	V_{DD} = 3.6V	2.0	V_{DD}+0.3	V
V_{IL}	Input low voltage	V_{DD} = 3.0V	–0.3	0.8	V
V_{OH}	Output high voltage	V_{DD} = 3.0V; I_{OH} = –8mA	2.4	–	V
V_{OL}	Output low voltage	V_{DD} = 3.0V; I_{OH} = 8mA	–	0.4	V
I_I	Input leakage current	V_{DD} = 3.6V; 0 < V_{IN} < V_{DD}	–10	10	µA
I_{DDSB}	Standby current	T_{amb} = 25°C; no output loads, inputs at V_{DD} or V_{SS}.	–	100	µA
C_{IN}	Input capacitance	T_{amb} = 25°C; V_{DD} = 3.3V; f = 1MHz	–	10	pF
C_{IO}	I/O capacitance	T_{amb} = 25°C; V_{DD} = 3.3V; f = 1MHz	–	10	pF
C_{CLK}	Clock pin capacitance	T_{amb} = 25°C; V_{DD} = 3.3V; f = 1MHz	–	12	pF
R_{DONE}	done pull-up resistor	V_{DD} = 3.0 V; V_{IN} = 0 V	10	30	kΩ
R_{PD}	Unused I/O pull-down resistor	V_{DD} = 3.6V; V_{IN} = V_{DD}	100	400	kΩ

AC ELECTRICAL CHARACTERISTICS FOR INDUSTRIAL GRADE DEVICES

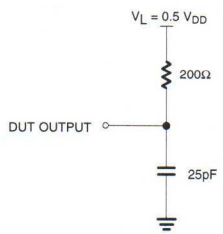
Industrial temperature range: V_{DD} = 3.0V to 3.6V; –40°C < T_{amb} < 85°C

SYMBOL	PARAMETER	MIN	MAX	UNIT
Timing requirements				
t_{CL}	Clock LOW time	2.5		ns
t_{CH}	Clock HIGH time	2.5		ns
t_{SU_PAL}	PAL setup time (Global clock)	4.5		ns
t_{SU_PLA}	PLA setup time (Global clock)	6.0		ns
t_{SU_XOR}	XOR setup time (Global clock)	7.0		ns
t_H	Hold time (Global clock)		0	ns
Output characteristics				
t_{PD_PAL}	Input to output delay through PAL		8.0	ns
t_{PD_PLA}	Input to output delay through PLA		9.5	ns
t_{PD_XOR}	Input to output delay through XOR		10.5	ns
t_{PDF_PAL}	Input (or feedback node) to internal feedback node delay time through PAL		4.0	ns
t_{PDF_PLA}	Input (or feedback node) to internal feedback node delay time through PLA		5.5	ns
t_{PDF_XOR}	Input (or feedback node) to internal feedback node delay time through XOR		6.5	ns
t_{CF}	Global clock to feedback delay		2.5	ns
t_{CO}	Global clock to out delay		6.5	ns
t_{CS}	Clock skew (variance for switching outputs with common global clock)		1.0	ns
f_{MAX1}	Maximum flip-flop toggle rate $\left(\frac{1}{t_{CL} + t_{CH}}\right)$	200		MHz
f_{MAX2}	Maximum internal frequency $\left(\frac{1}{t_{SU_PAL} + t_{CF}}\right)$	143		MHz
f_{MAX3}	Maximum external frequency $\left(\frac{1}{t_{SU_PAL} + t_{CO}}\right)$	91		MHz
t_{BUFF}	Output buffer delay (fast)		4.0	ns
t_{SSR}	Slow slew rate incremental delay		8.0	ns
t_{EA}	Output enable delay		8.5	ns
t_{ER}	Output disable delay[1]		8.5	ns
t_{GTSH}	Global 3-State enable		40.0	ns
t_{GTSR}	Global 3-State disable		40.0	ns
t_{RR}	Input to register reset		11.0	ns
t_{RP}	Input to register preset		11.0	ns
t_{GRR}	Global reset to register reset		40	ns
t_{GZIA}	Global ZIA delay		4.5	ns

NOTE:

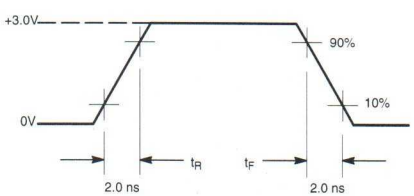
1. Output C_L = 5.0pF.

THEVENIN EQUIVALENT



SP00629

VOLTAGE WAVEFORM



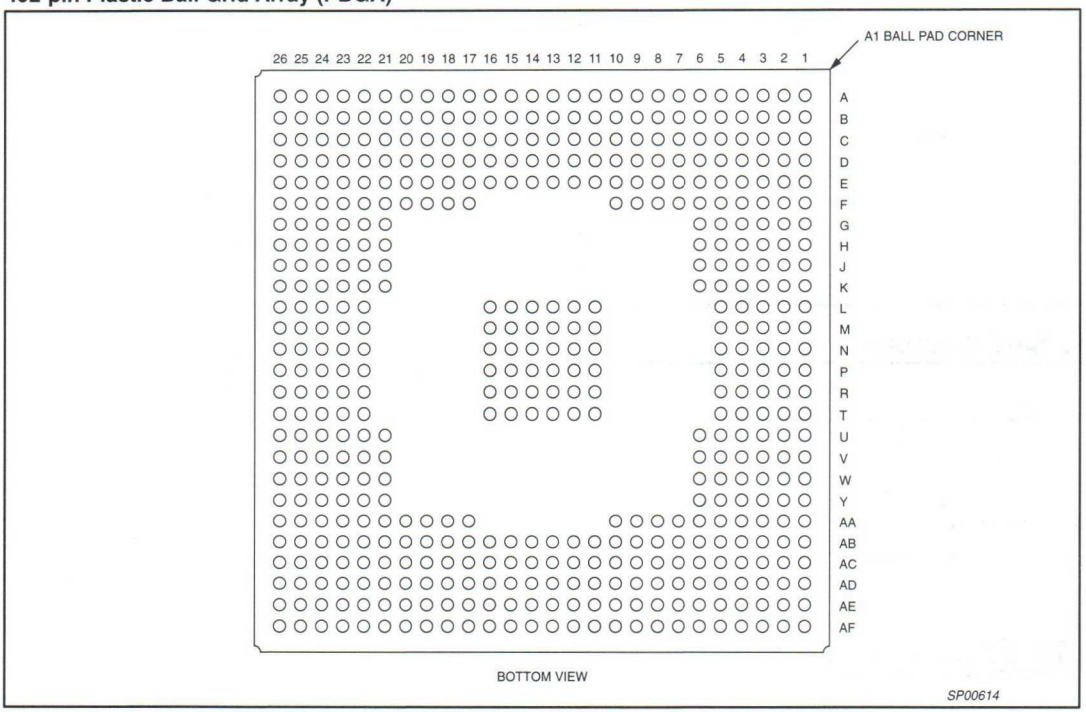
SP00630

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

Input Pulses

PINNING

492-pin Plastic Ball Grid Array (PBGA)



Pin Functions

Pkg Ball	Function
A1	F1_5
A2	F1_7
A3	F1_22
A4	F1_19
A5	F1_30
A6	F1_26
A7	F2_10
A8	F2_15
A9	F2_3
A10	F2_23
A11	F2_19
A12	F2_17
A13	F2_28
A14	F3_9
A15	F3_13
A16	F3_15
A17	F3_4
A18	F3_23
A19	F3_18
A20	F3_29
A21	F3_25
A22	F4_11
A23	F4_15
A24	F4_3
A25	F4_5
A26	F4_23
B1	F1_1
B2	F1_2
B3	F1_23
B4	F1_20
B5*	F1_31
B6	F1_27
B7	F2_9
B8	F2_13
B9	F2_2
B10	F2_7
B11	F2_22
B12	F2_18
B13	F2_29
B14	F3_10
B15	F3_14
B16	F3_3
B17	F3_5
B18	F3_21
B19	F3_17
B20	F3_28
B21	F3_24
B22	F4_12
B23	F4_0
B24	F4_4
B25	F4_6
B26	F4_21
C1	F1_14
C2	F1_0
C3	F1_4
C4	F1_21
C5	F1_16
C6	F1_28
C7*	F2_8
C8	F2_12
C9	F2_1
C10	F2_5
C11	F2_21
C12*	F2_31
C13	F2_27
C14	F2_24
C15	F3_12
C16	F3_2
C17	F3_6
C18	F3_20
C19	F3_16
C20	F3_27
C21*	F4_8
C22	F4_13
C23	F4_1
C24	F4_20
C25	F4_19
C26	F4_18
D1	F1_12
D2	F1_13
D3	F1_15
D4	F1_3
D5	F1_18
D6	F1_29
D7	F1_24
D8	F2_11
D9	F2_0
D10	F2_4
D11	F2_20
D12	F2_30
D13	F2_26
D14	F2_25
D15	F3_11
D16	F3_1
D17	F3_7
D18	F3_19
D19*	F3_31
D20	F3_26
D21	F4_9
D22	F4_14
D23	F4_7
D24	F4_17
D25	F4_16
D26*	F4_31
E1	F1_8
E2	F1_9
E3	F1_10
E4	F1_11
E5	GND
E6	F1_6
E7	F1_25
E8	F2_6
E9	V_{DD}
E10	F2_16
E11	V_{DD}
E12	V_{DD}
E13*	F3_8
E14	V_{DD}
E15	V_{DD}
E16	F3_0
E17	V_{DD}
E18	V_{DD}
E19	F3_22
E20	F4_10
E21	F4_22
E22	GND
E23	F4_30
E24	F4_29
E25	F4_28
E26	F4_27
F1	F0_28
F2	F0_27
F3	F0_25
F4	F0_24
F5	V_{DD}
F6	GND
F7	F1_17
F8	F2_14
F9	V_{DD}
F10	V_{DD}
F17	V_{DD}
F18	V_{DD}
F19	F3_30
F20	F4_2
F21	GND
F22	V_{DD}
F23	F4_26
F24	F4_25
F25	F4_24
F26	F5_8
G1	F0_17
G2	F0_16
G3	F0_30
G4	F0_29
G5	V_{DD}
G6	V_{DD}
G21	V_{DD}
G22	V_{DD}
G23	F5_9
G24	F5_11
G25	F5_12
G26	F5_13
H1	F0_22
H2	F0_21
H3	F0_19
H4	F0_18
H5	F0_5
H6	F0_15
H21	F5_31
H22	F5_21
H23	F5_14
H24	F5_0
H25	F5_1
H26	F5_2
J1	F0_4
J2	F0_6
J3	F0_7
J4	F0_23
J5	F0_20
J6	F0_26
J21	F5_10
J22	F5_4
J23	F5_3
J24	F5_5
J25	F5_6
J26	F5_7
K1	F0_0
K2	F0_1
K3	F0_2
K4	F0_3
K5	V_{DD}
K6	F0_31
K21	F5_15
K22	V_{DD}
K23	F5_23
K24	F5_22
K25	F5_20
K26	F5_19
L1	F0_11
L2	F0_14
L3*	F0_13
L4	F0_12
L5	F0_9
L11	GND
L12	GND
L13	GND
L14	GND
L15	GND
L16	GND
L22	V_{DD}
L23	F5_16
L24	F5_17
L25	F5_18
L26	F5_30
M1*	F0_8
M2	F0_10
M3	clk0
M4	clk1
M5	V_{DD}
M11	GND
M12	GND
M13	GND
M14	GND
M15	GND
M16	GND
M22	F5_25
M23	F5_26
M24	F5_27
M25	F5_29
M26	F5_28
N1	clk3
N2	clk2
N3	trstn
N4	tms
N5	gts
N11	GND
N12	GND
N13	GND
N14	GND
N15	GND
N16	GND
N22	V_{DD}
N23	resetn
N24	GND
N25	F5_24
N26	GND
P1	tdi
P2	F11_24
P3	tdo
P4	tck
P5	V_{DD}
P11	GND
P12	GND
P13	GND
P14	GND
P15	GND
P16	GND
P22	done
P23	pgmn
P24	cclk
P25	clk6
P26	clk7
R1	F11_28
R2	F11_29
R3*	F11_27
R4	F11_26
R5	F11_25
R11	GND
R12	GND
R13	GND
R14	GND
R15	GND
R16	GND
R22	V_{DD}
R23	clk5
R24	clk4
R25	F6_10
R26	F6_9
T1	F11_30
T2	F11_18
T3	F11_17
T4	F11_16
T5	V_{DD}
T11	GND
T12	GND
T13	GND
T14	GND
T15	GND
T16	GND
T22	F6_8
T23	F6_12
T24	F6_13
T25	F6_14
T26	F6_11
U1	F11_19
U2	F11_20
U3	F11_22
U4	F11_23
U5	V_{DD}
U6	F11_15
U21	F6_31
U22	V_{DD}
U23	F6_3
U24	F6_2
U25	F6_1
U26	F6_0
V1	F11_7
V2	F11_6
V3	F11_4
V4	F11_3
V5*	F11_5
V6	F11_10
V21	F6_26
V22	F6_21
V23	F6_23
V24	F6_7
V25	F6_6
V26	F6_4
W1	F11_2
W2	F11_1
W3	F11_0
W4	F11_14
W5	F11_21
W6*	F11_31
W21	F6_15
W22	F6_5
W23	F6_18
W24*	F6_19
W25	F6_20
W26	F6_22
Y1*	F11_13
Y2	F11_12
Y3	F11_11
Y4	F11_9
Y5	V_{DD}
Y6	V_{DD}
Y21	V_{DD}
Y22	V_{DD}
Y23	F6_29
Y24	F6_30
Y25	F6_16
Y26	F6_17
AA1*	F11_8
AA2	F10_24
AA3	F10_25
AA4	F10_26
AA5	V_{DD}
AA6	GND
AA7	F10_2
AA8	F9_30
AA9	V_{DD}
AA10	V_{DD}
AA17	V_{DD}
AA18	V_{DD}
AA19	F8_14
AA20	F7_17
AA21	GND
AA22	V_{DD}
AA23	F6_24
AA24	F6_25
AA25	F6_27
AA26	F6_28
AB1	F10_27
AB2	F10_28
AB3	F10_29
AB4	F10_30
AB5	GND
AB6	F10_22
AB7	F10_10
AB8	F9_22
AB9	V_{DD}
AB10	V_{DD}
AB11	F9_0
AB12	V_{DD}
AB13	V_{DD}
AB14	F8_24
AB15	V_{DD}
AB16	V_{DD}
AB17	F8_17
AB18	V_{DD}
AB19	F8_6
AB20	F7_26
AB21	F7_6
AB22	GND
AB23	F7_11
AB24	F7_10
AB25	F7_9
AB26	F7_8
AC1*	F10_31
AC2	F10_16
AC3	F10_17
AC4	F10_7
AC5	F10_14
AC6	F10_9
AC7	F9_26
AC8*	F9_31
AC9*	F9_19
AC10	F9_7
AC11	F9_1
AC12	F9_11
AC13	F8_25
AC14	F8_26
AC15	F8_30
AC16	F8_20
AC17	F8_4
AC18	F8_0
AC19	F8_11
AC20	F7_24
AC21	F7_29
AC22	F7_18
AC23	F7_3
AC24	F7_14
AC25	F7_13
AC26	F7_12
AD1	F10_18
AD2	F10_19
AD3	F10_20
AD4	F10_1
AD5*	F10_13
AD6*	F10_8
AD7*	F9_27
AD8	F9_16
AD9	F9_20
AD10	F9_6
AD11	F9_2
AD12	F9_12
AD13*	F9_8
AD14*	F8_27
AD15*	F8_31
AD16	F8_21
AD17*	F8_5
AD18	F8_1
AD19	F8_12
AD20*	F8_8
AD21	F7_28
AD22	F7_16
AD23	F7_21
AD24	F7_7
AD25	F7_0
AD26	F7_15
AE1	F10_21
AE2	F10_6
AE3	F10_4
AE4	F10_0
AE5	F10_12
AE6	F9_24
AE7	F9_28
AE8	F9_17
AE9	F9_21
AE10*	F9_5
AE11*	F9_3
AE12	F9_14
AE13	F9_10
AE14	F8_29
AE15	F8_18
AE16	F8_22
AE17	F8_7
AE18	F8_2
AE19*	F8_13
AE20	F8_9
AE21*	F7_27
AE22*	F7_31
AE23	F7_20
AE24*	F7_23
AE25	F7_2
AE26	F7_1
AF1	F10_23
AF2*	F10_5
AF3*	F10_3
AF4	F10_15
AF5	F10_11
AF6	F9_25
AF7	F9_29
AF8	F9_18
AF9*	F9_23
AF10	F9_4
AF11	F9_15
AF12*	F9_13
AF13	F9_9
AF14	F8_28
AF15	F8_16
AF16*	F8_19
AF17*	F8_23
AF18*	F8_3
AF19	F8_15
AF20	F8_10
AF21	F7_25
AF22	F7_30
AF23*	F7_19
AF24	F7_22
AF25	F7_5
AF26	F7_4

* Multi-function pin used during configuration. See Table 9 for information.

Table 9. Pin Description

SYMBOL	PIN NUMBER	TYPE	DESCRIPTION
V_{DD}	E9, E11, E12, E14, E15, E17, E18, F5, F9, F10, F17, F18, F22, G5, G6, G21, G22, K5, K22, L22, M5, N22, P5, R22, T5, U5, U22, Y5, Y6, Y21, Y22, AA5, AA9, AA10, AA17, AA18, AA22, AB9, AB10, AB12, AB13, AB15, AB16, AB18	–	Positive power supply.
GND	E5, E22, F6, F21, L11, L12, L13, L14, L15, L16, M11, M12, M13, M14, M15, M16, N11, N12, N13, N14, N15, N16, P11, P12, P13, P14, P15, P16, R11, R12, R13, R14, R15, R16, T11, T12, T13, T14, T15, T16, AA6, AA21, AB5, AB22	–	Ground supply.
resetn	N23	I	During configuration, resetn forces the start of initialization (see Figure 8). After configuration, resetn is a direct input which can be used to asynchronously reset all the flip-flops. If the global reset is not being used, this pin should be pulled high.
cclk	P24	I/O	In the master modes, cclk is an output which strobes configuration data in. In the slave or synchronous peripheral mode, cclk is an input synchronous with the data on din or D[7:0]. After configuration, this pin should be pulled low.
done	P22	I/O	done is a bi-directional signal with a weak pull-up resistor attached. As an output, done pulling high indicates configuration is complete. As an input, a low level on done will delay device initialization and the enabling of user I/O. If only one device is used, this pin can be left floating. If multiple devices are daisy chained, an external pull-up should be used (see Figure 28).
pgmn	P23	I	pgmn is an active-low input that forces the restart of configuration and initialization (see Figure 8) and resets the boundary-scan circuitry. After configuration, the pin should be pulled high.
rdy_busyn	E13	O	During configuration in peripheral mode, rdy_busyn indicates another byte can be written to the PZ3960. After configuration, the pin is a user-programmable I/O, and no external termination is required. See the section on terminations for more information.
rclk	C12	O	During the master parallel configuration mode, rclk is an output signal to an external memory. rclk is not normally used. After configuration, this pin is a user-programmable I/O pin, and no external termination is required. See the section on terminations for more information.
din	AC1	I	During slave serial or master serial configuration modes, din accepts serial configuration data synchronous with cclk. During parallel configuration modes, din is the D[0] input. After configuration, the pin is a user-programmable I/O, and no external termination is required. See the section on terminations for more information.
M2	AE22	I	M2/M1/M0 are used to select the configuration mode as defined in Table 3. After configuration, the pins are user-programmable I/O, and no external termination is required. See the section on terminations for more information.
M0	AE24		
M1	AF23		
M3	AD20	I	M3 is used to select the frequency of the internal oscillator during configuration. When M3 is low, the oscillator is nominally 10MHz. When M3 is high, the oscillator is nominally 1.25MHz. After configuration, the pin is a user-programmable I/O, and no external termination is required. See the section on terminations for more information.

SYMBOL	PIN NUMBER	TYPE	DESCRIPTION
tdi tdo tck tms trstn	P1 P3 P4 N4 N3	I O I I I	Test Data In, Test Data Out, Test Clock, Test Mode Select, Test Reset are dedicated pins for boundary-scan through the JTAG port. If JTAG is not being used, tdi, tck, tms, and trstn should be terminated with a weak pull-up resistor. tdo can be left unterminated. See section on terminations for more information.
hdc	C21	O	High During Configuration (hdc) is output high when the PZ3960 is in the configuration state. hdc is used as a control output indicating that configuration is in progress. After configuration, the pin is a user-programmable I/O, and no external termination is required. See the section on terminations for more information.
ldcn	D19	O	Low During Configuration (ldcn) is output low when the PZ3960 is in the configuration state. ldcn is used as a control output indicating that configuration is in progress. After configuration, the pin is a user-programmable I/O, and no external termination is required. See the section on terminations for more information.
initn	D26	I/O	initn is an active-low bi-directional pin that holds the PZ3960 in a wait state before the start of configuration. During configuration, an internal pull-up is enabled. If only one device is used, this pin can be left floating. If multiple devices are daisy chained, an external pull-up should be used (see Figure 28). After configuration, the pin is a user-programmable I/O, and no external termination is required. See the section on terminations for more information.
gts	N5	I	Global 3-State is an active-high dedicated input used to 3-state the I/Os. If this feature is not used, the pin should be pulled low.
cs0n cs1	B5 C7	I	cs0n/cs1 are used in the peripheral configuration mode. The PZ3960 is selected when cs0n is low and cs1 is high. After configuration, these pins are user-programmable I/O, and no external termination is required. See the section on terminations for more information.
A[19:0]	AF2, AF3, AD5, AD6, AD7, AC8, AC9, AF9, AE10, AE11, AF12, AD13, AD14, AD15, AF16, AF17, AD17, AF18, AE19, AE21	O	In the master parallel configuration mode, A[19:0] address the configuration EEPROM. After configuration, the pin is a user-programmable I/O, and no external termination is required. See the section on terminations for more information.
D[7:0]	L3, M1, R3, W6, V5, Y1, AA1, AC1	I	During master parallel, peripheral, and slave parallel configuration modes, D[7:0] receive configuration data. After configuration, the pin is a user-programmable I/O, and no external termination is required. See the section on terminations for more information.
dout	W24	O	During configuration, dout is the serial data out that is used to drive the din of daisy-chained slave devices. Data on dout changes on the falling edge of cclk. After configuration, the pin is a user-programmable I/O, and no external termination is required. See the section on terminations for more information.

Section 6
Related Products

CONTENTS

P3Z22V10	3V zero power, TotalCMOS™, universal PLD device	285
P5Z22V10	5V zero power, TotalCMOS™, universal PLD device	296
PZLCP	Low Cost Programmer for Philips CoolRunner devices and related adapters	307
PZVHDLEZ	Full VHDL synthesis and automatic optimization software for Philips Semiconductors CoolRunners	308

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.

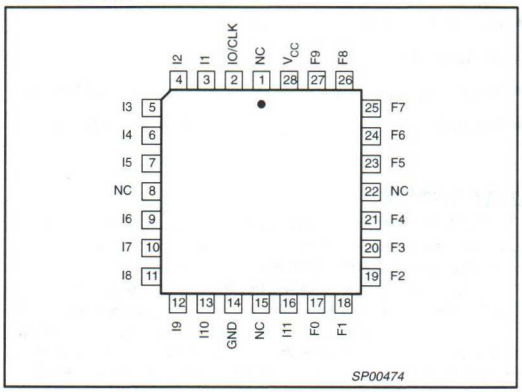
The P3Z22V10 uses the familiar AND/OR logic array structure, which allows direct implementation of sum-of-products equations. This device has a programmable AND array which drives a fixed OR array. The OR sum of products feeds an "Output Macro Cell" (OMC), which can be individually configured as a dedicated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

ORDERING INFORMATION

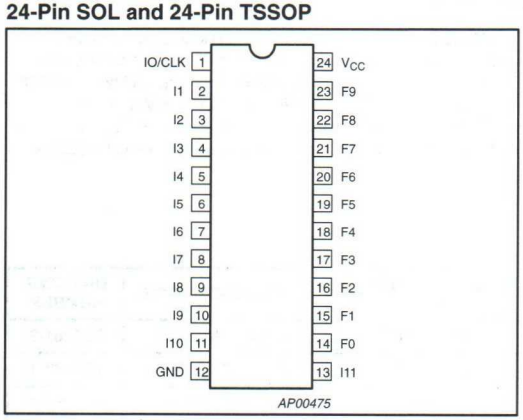
ORDER CODE	PACKAGE	PROPAGATION DELAY	TEMPERATURE RANGE	OPERATING RANGE	DRAWING NUMBER
P3Z22V10-DA	28-pin PLCC	10ns	0 to +70°C	V_{CC} = 3.3V ±10%	SOT261-3
P3Z22V10-DD	24-pin SOL	10ns	0 to +70°C	V_{CC} = 3.3V ±10%	SOT137-1
P3Z22V10-DDH	24-pin TSSOP	10ns	0 to +70°C	V_{CC} = 3.3V ±10%	SOT355-1
P3Z22V10-BA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



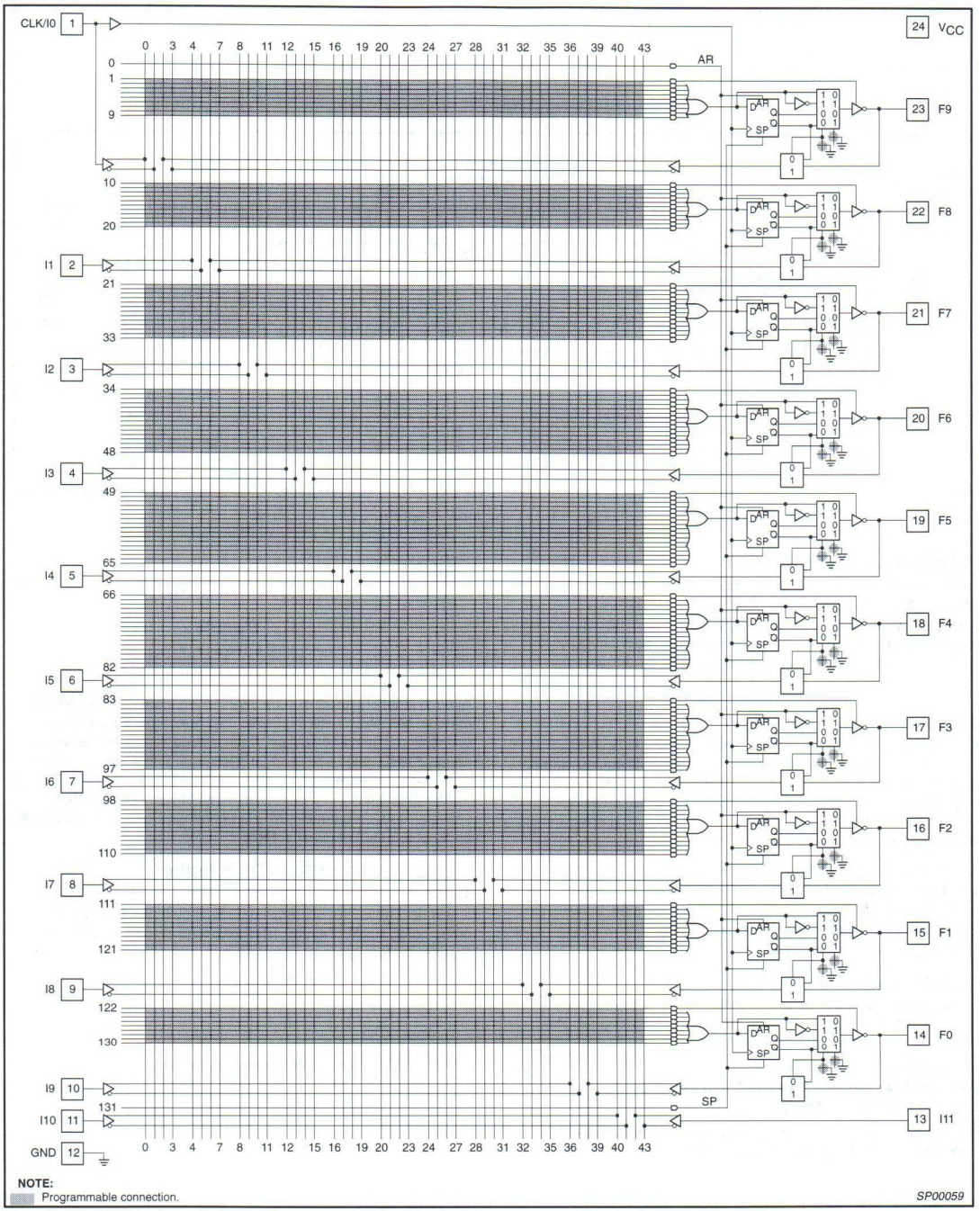
24-Pin SOL and 24-Pin TSSOP



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

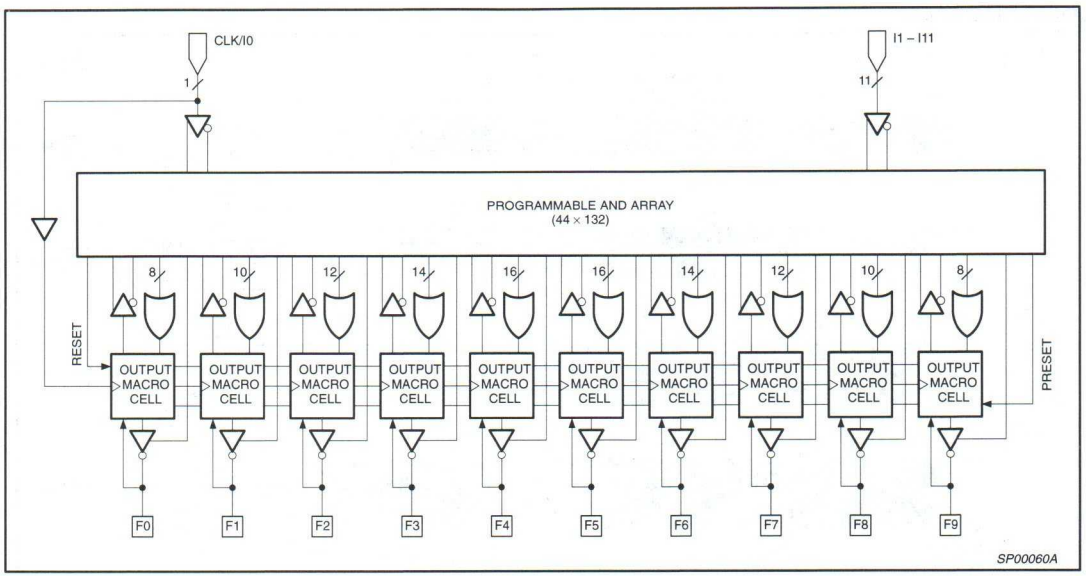


Figure 1. Functional Diagram

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:

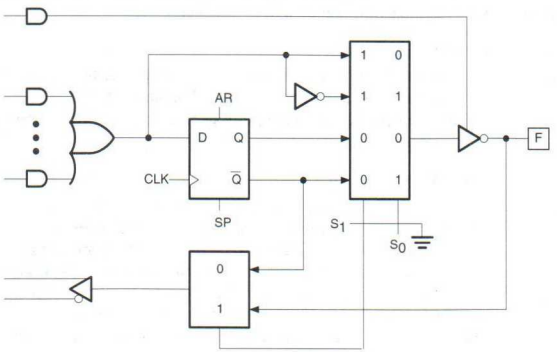
44 input lines:
- 24 input lines carry the True and Complement of the signals applied to the 12 input pins
- 20 additional lines carry the True and Complement values of feedback or input signals from the 10 I/Os

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.

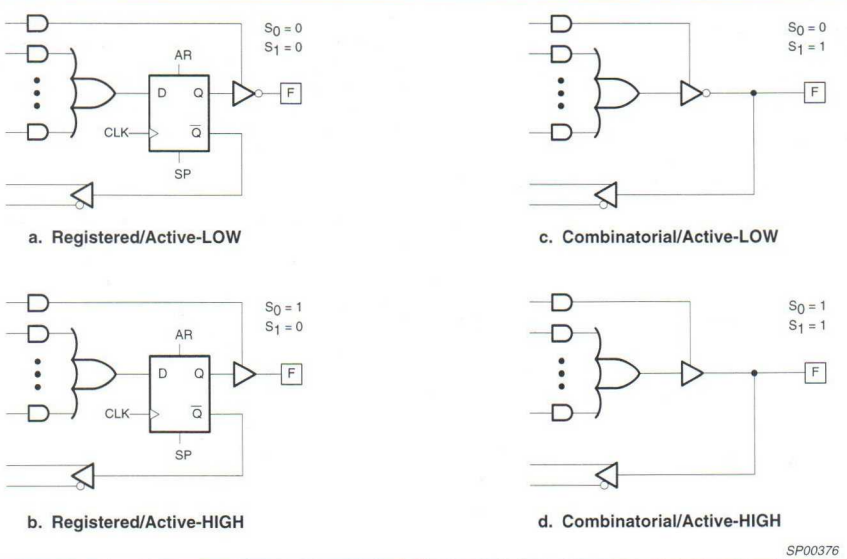


S_1	S_0	OUTPUT CONFIGURATION
0	0	Registered/Active-LOW/Macrocell feedback
0	1	Registered/Active-HIGH/Macrocell feedback
1	0	Combinatorial/Active-LOW/Pin feedback
1	1	Combinatorial/Active-HIGH/Pin feedback

0 = Unprogrammed fuse
1 = Programmed fuse

SP00484

Figure 2. Output Macro Cell Logic Diagram



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.

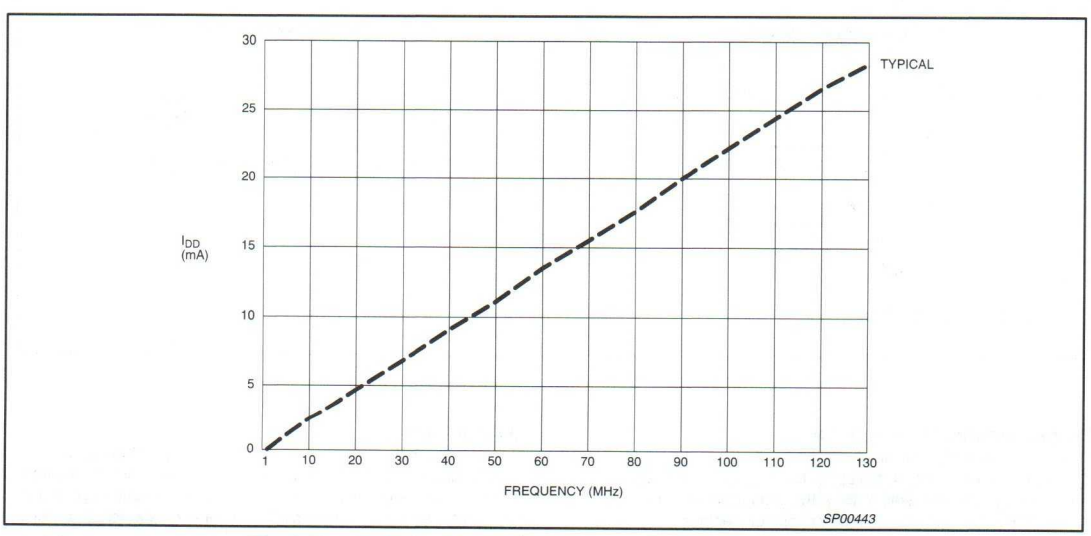


Figure 4. Typical I_{DD} vs. Frequency @ V_{DD} = 3.3V, 25°C (10-bit counter)

Table 1. Typical I_{DD} vs. Frequency

V_{DD} = 3.3V@25°C

FREQ (MHz)	1	10	20	30	40	50	60	70	80	90	100	110	120	130
Typical I_{DD} (mA)	0.2	1.5	3.0	4.5	6.0	7.4	8.9	10.4	11.8	13.2	14.5	15.8	17.0	18.2

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: $0°C \le T_{amb} \le +70°C$; $3.0 \le V_{DD} \le 3.6V$

SYMBOL	PARAMETER	TEST CONDITIONS	LIMITS			UNITS
			MIN.	TYP.	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	$V_{DD} = 3.0V$; $I_{OH} = -4mA$	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]	–10		10	
I_{OZ}	3-Stated output leakage current	$V_{IN} = 0$ to V_{DD}	–10		10	µA
		$V_{IN} = V_{DD}$ to 5.5V[2]	–10		10	
I_{DDQ}	Standby current	$V_{DD} = 3.6V$; $T_{amb} = 0°C$		25	45	µA
I_{DDD} [1]	Dynamic current	$V_{DD} = 3.6V$; $T_{amb} = 0°C$ @ 1MHz		.5	2	mA
		$V_{DD} = 3.6V$; $T_{amb} = 0°C$ @ 50MHz		10	15	mA
I_{SC}	Short circuit output current	1 pin/time for no longer than 1 second	–15		–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: $0°C \le T_{amb} \le +70°C$; $3.0 \le V_{DD} \le 3.6V$

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		9	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		80		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: $-40°C \le T_{amb} \le +85°C$; $3.0 \le V_{DD} \le 3.6V$

SYMBOL	PARAMETER	TEST CONDITIONS	LIMITS			UNITS
			MIN.	TYP.	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_{OH} = –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]	–10		10	µA
I_{OZ}	3-Stated output leakage current	V_{IN} = 0 to V_{DD}	–10		10	µA
		V_{IN} = V_{DD} to 5.5V[2]	–10		10	µA
I_{DDQ}	Standby current	V_{DD} = 3.6V; T_{amb} = –40°C		30	45	µA
I_{DDD} [1]	Dynamic current	V_{DD} = 3.6V; T_{amb} = –40°C @ 1MHz		.5	3	mA
		V_{DD} = 3.6V; T_{amb} = –40°C @ 50MHz		10	20	mA
I_{SC}	Short circuit output current	1 pin/time for no longer than 1 second	–15		–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

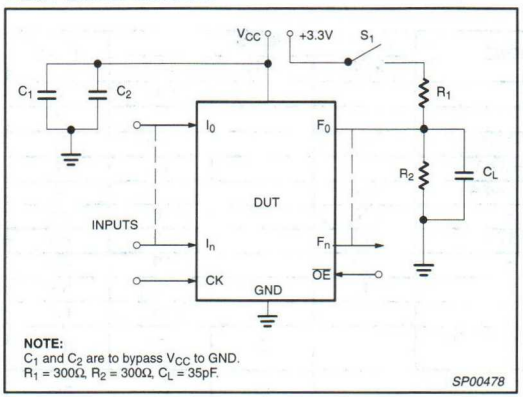
Industrial: $-40°C \le T_{amb} \le +85°C$; $3.0 \le V_{DD} \le 3.6V$

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.

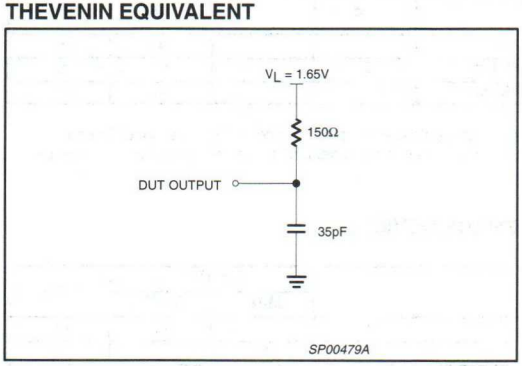
TEST LOAD CIRCUIT



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

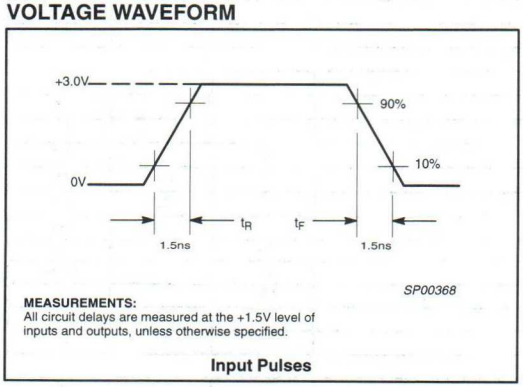
SP00478

THEVENIN EQUIVALENT



SP00479A

VOLTAGE WAVEFORM

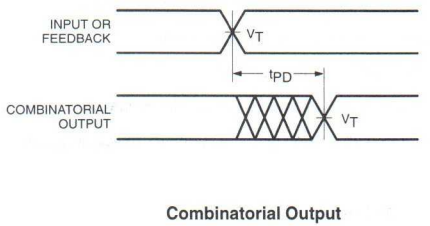


SP00368

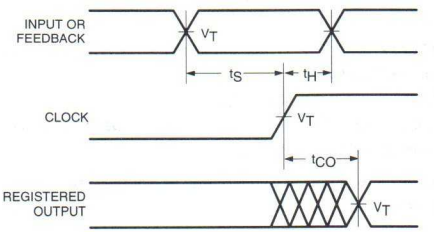
MEASUREMENTS:
All circuit delays are measured at the +1.5V level of inputs and outputs, unless otherwise specified.

Input Pulses

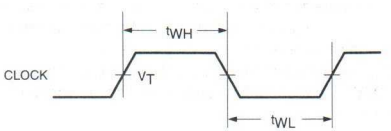
SWITCHING WAVEFORMS



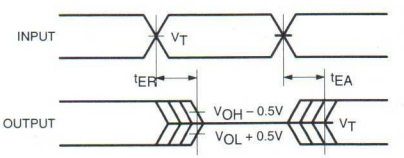
Combinatorial Output



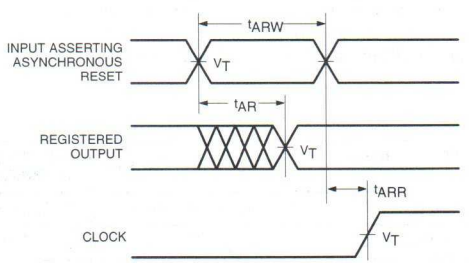
Registered Output



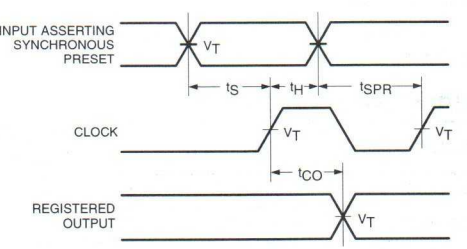
Clock Width



Input to Output Disable/Enable



Asynchronous Reset

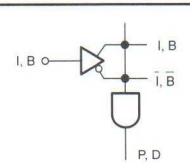


Synchronous Preset

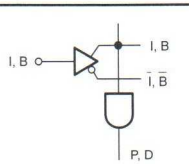
NOTES:
1. V_T = 1.5V.
2. Input pulse amplitude 0V to 3.0V.
3. Input rise and fall times 2.0ns max.

SP00065

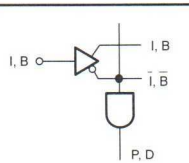
"AND" ARRAY – (I, B)



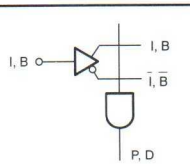
STATE	CODE
INACTIVE[1]	O



STATE	CODE
TRUE	H



STATE	CODE
COMPLEMENT	L



STATE	CODE
DON'T CARE	—

SP00008

NOTE:
1. This is the initial state.

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^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 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.

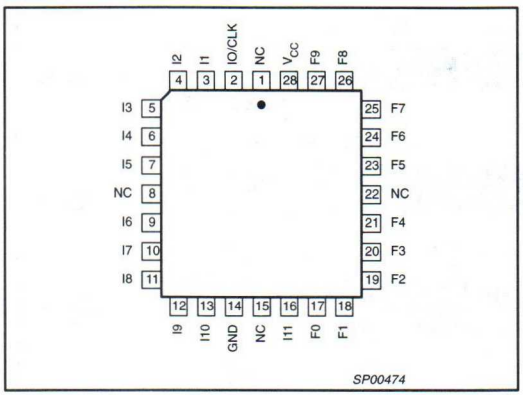
The P5Z22V10 uses the familiar AND/OR logic array structure, which allows direct implementation of sum-of-products equations. This device has a programmable AND array which drives a fixed OR array. The OR sum of products feeds an "Output Macro Cell" (OMC), which can be individually configured as a dedicated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

ORDERING INFORMATION

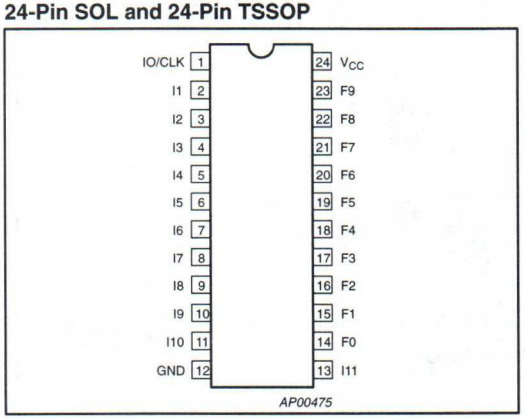
ORDER CODE	PACKAGE	PROPAGATION DELAY	TEMPERATURE RANGE	OPERATING RANGE	DRAWING NUMBER
P5Z22V10-7A	28-pin PLCC	7.5ns	0 to +70°C	V_{CC} = 5.0V ±5%	SOT261-3
P5Z22V10-7D	24-pin SOL	7.5ns	0 to +70°C	V_{CC} = 5.0V ±5%	SOT137-1
P5Z22V10-7DH	24-pin TSSOP	7.5ns	0 to +70°C	V_{CC} = 5.0V ±5%	SOT355-1
P5Z22V10–DA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



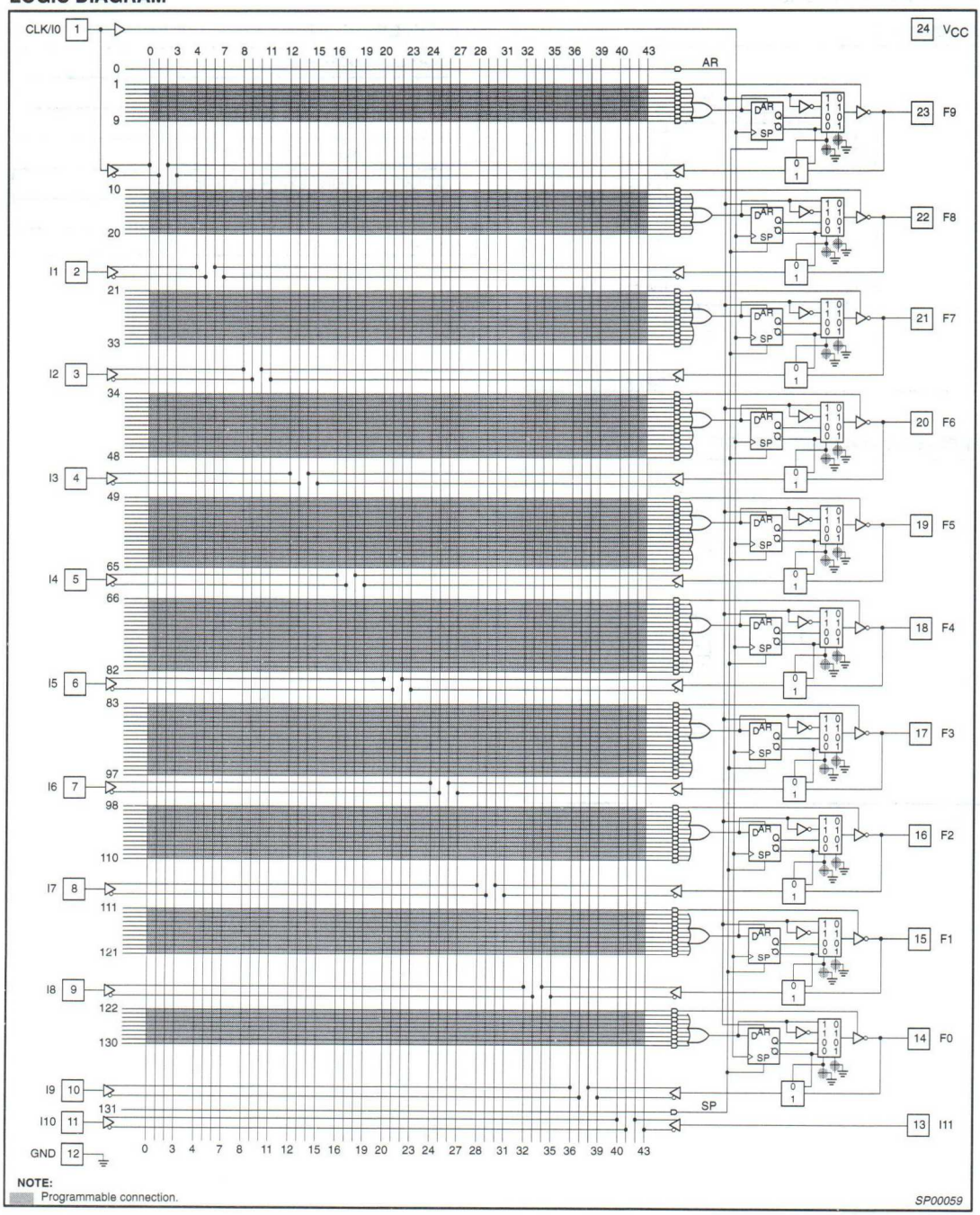
24-Pin SOL and 24-Pin TSSOP



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



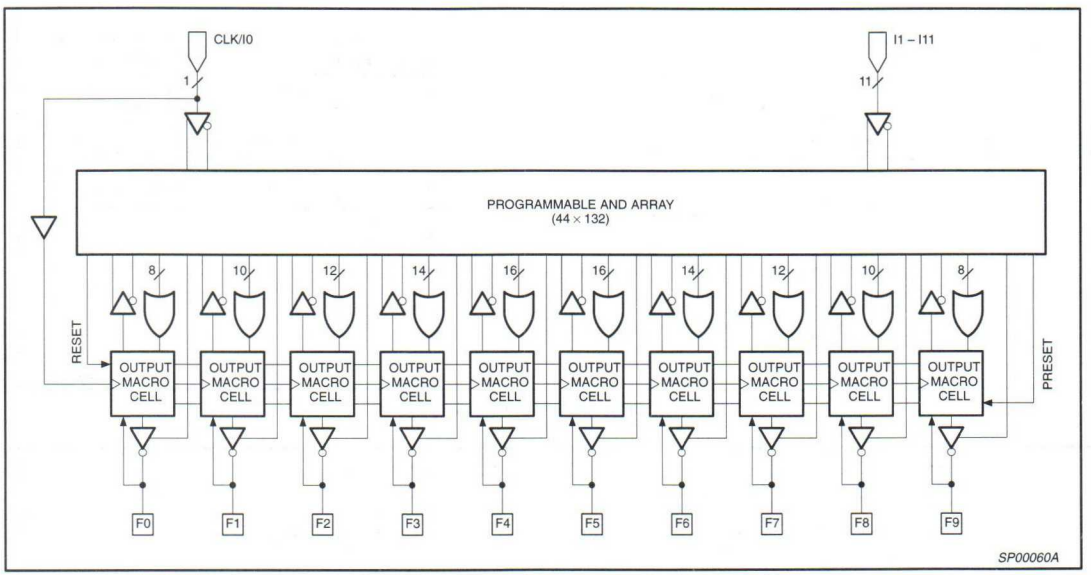


Figure 1. Functional Diagram

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:

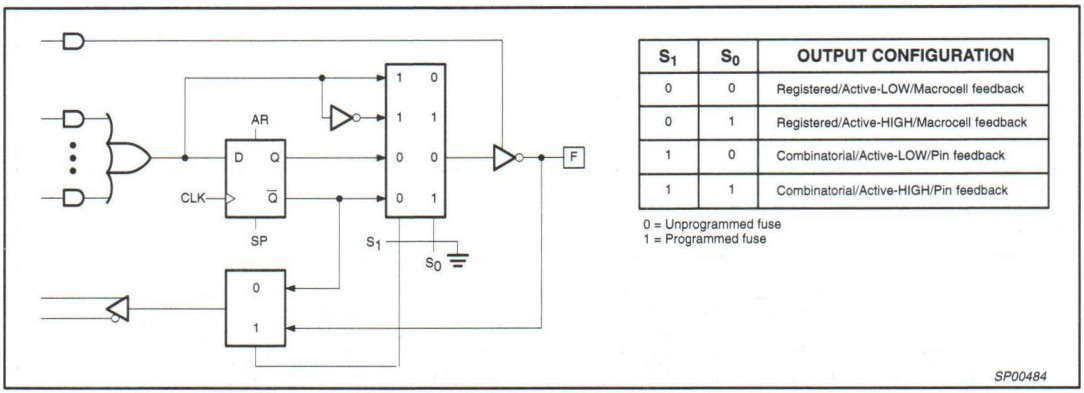
44 input lines:
- 24 input lines carry the True and Complement of the signals applied to the 12 input pins
- 20 additional lines carry the True and Complement values of feedback or input signals from the 10 I/Os

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

Figure 2. Output Macro Cell Logic Diagram

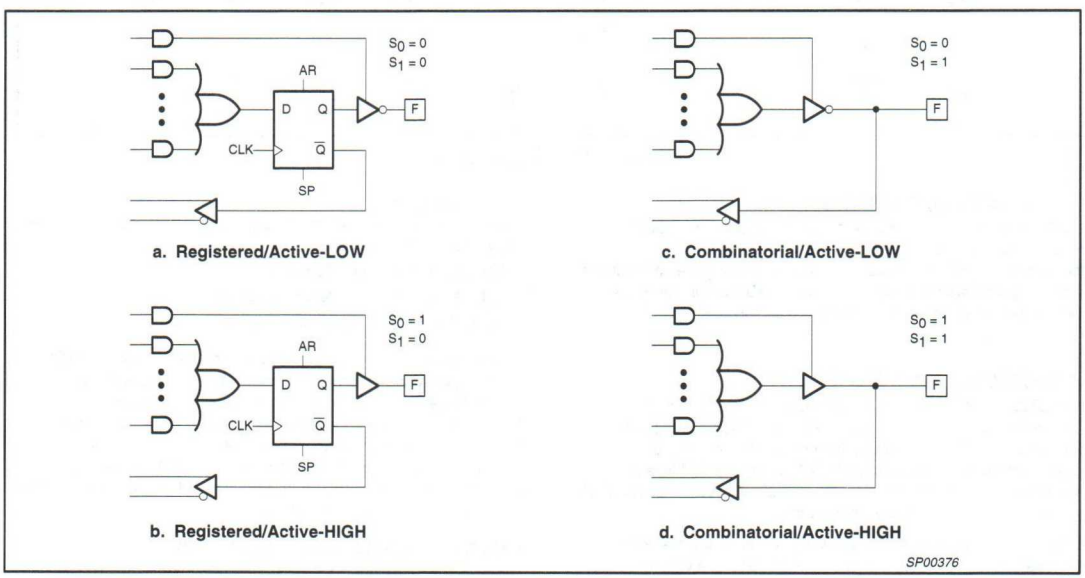


Figure 3. Output Macro Cell Configurations

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

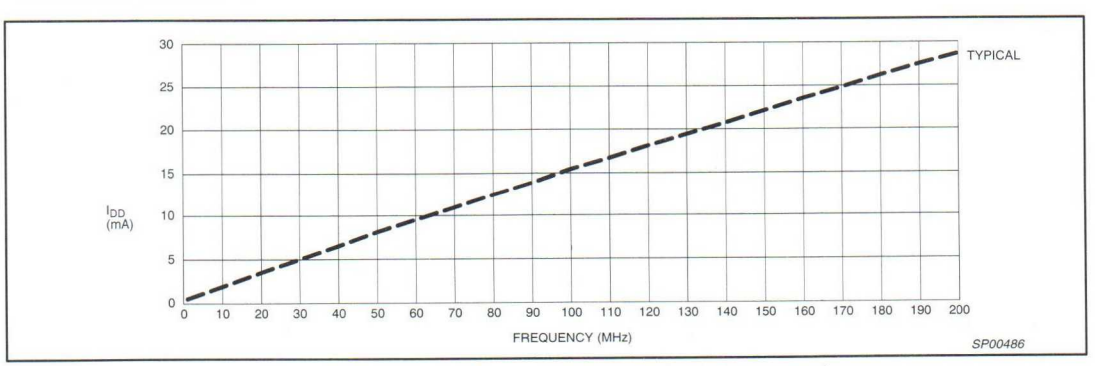


Figure 4. Typical I_{DD} vs. Frequency @ V_{DD} = 5V, 25°C (10-bit counter)

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: $0°C \le T_{amb} \le +70°C$; $4.75 \le V_{DD} \le 5.25V$

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	3	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: $0°C \le T_{amb} \le +70°C$; $4.75 \le V_{DD} \le 5.25V$

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

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

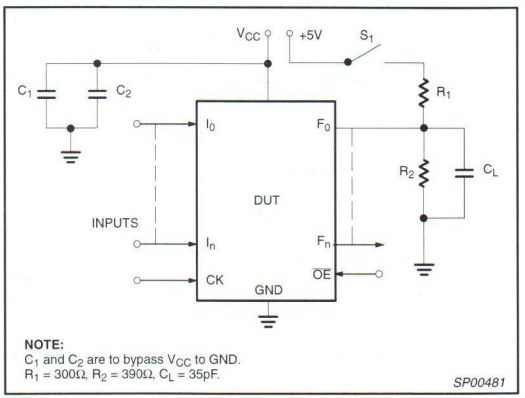
Industrial: $-40°C \le T_{amb} \le +85°C$; $4.5 \le V_{DD} \le 5.5V$

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.

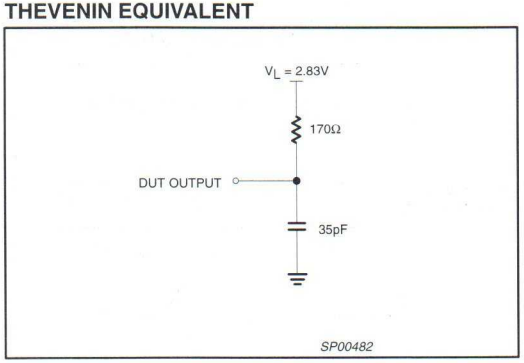
TEST LOAD CIRCUIT



NOTE:
C_1 and C_2 are to bypass V_{CC} to GND.
$R_1 = 300\Omega$, $R_2 = 390\Omega$, $C_L = 35pF$.

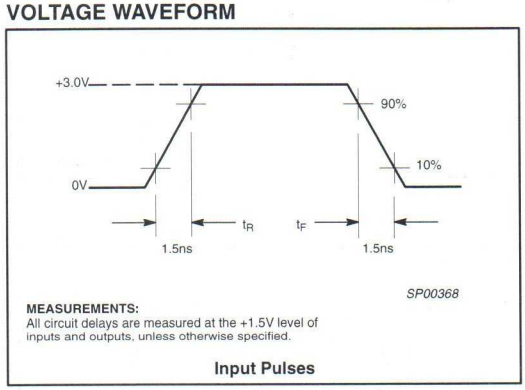
SP00481

THEVENIN EQUIVALENT



SP00482

VOLTAGE WAVEFORM

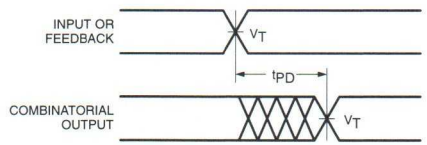


SP00368

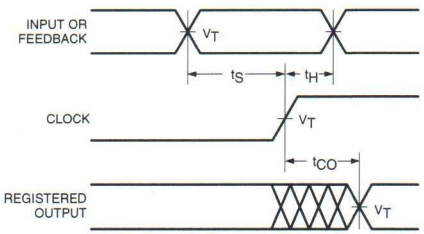
MEASUREMENTS:
All circuit delays are measured at the +1.5V level of inputs and outputs, unless otherwise specified.

Input Pulses

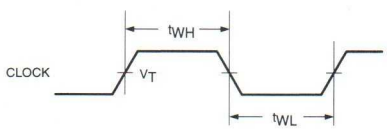
SWITCHING WAVEFORMS



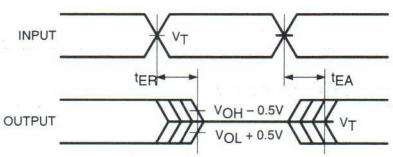
Combinatorial Output



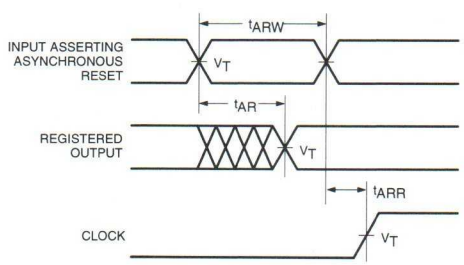
Registered Output



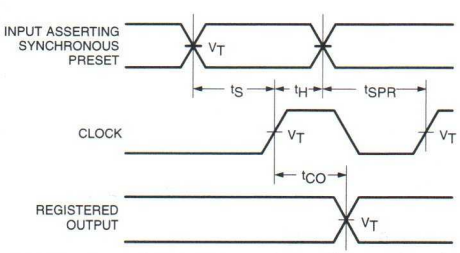
Clock Width



Input to Output Disable/Enable



Asynchronous Reset

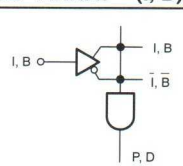


Synchronous Preset

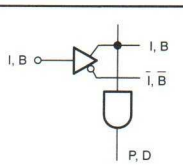
NOTES:
1. V_T = 1.5V.
2. Input pulse amplitude 0V to 3.0V.
3. Input rise and fall times 2.0ns max.

SP00065

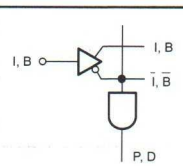
"AND" ARRAY – (I, B)



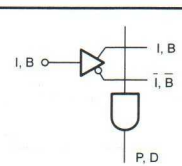
STATE	CODE
INACTIVE[1]	O



STATE	CODE
TRUE	H



STATE	CODE
COMPLEMENT	L



STATE	CODE
DON'T CARE	—

SP00008

NOTE:
1. This is the initial state.

Low Cost Programmer for Philips CoolRunner devices and related adapters

PZLCP

FEATURES

- Compact size, 5.5 x 4.25 inches (130 mm x 110 mm)
- Programs all CoolRunner™ devices—22V10 and CPLDs—both 3.3 and 5 volt
- Serial port connection to IBM-compatible personal computer
- CE marked for EMC (European requirements)
- Universal switched mode power supply (100 to 260 Vac) for use anywhere in the world
- On-screen and LED operational status indicators
- Simple and fast to use, complete with comprehensive context sensitive online help
- Base unit includes:
  - On board ZIF socket for 44 PLCC devices (PZX032C/N requires adapter)
  - Power supply
  - 5 ft, 6 in. (1.75 m) serial cable
  - 9 to 25 pin adapter
  - Interface software for Windows 3.1 or Windows 95
- Adapters are available to support all currently released devices and packages, such as:
  - 24 pin SOL
  - 24 pin TSSOP
  - 28 pin PLCC
  - 44 pin PLCC
  - 44 pin TQFP
  - 68 pin PLCC
  - 84 pin PLCC
  - 100 pin PQFP
  - 100 pin TQFP
  - 128 pin LQFP
  - 160 pin PQFP

DESCRIPTION

The PZLCP MSC CoolRunner programmer is a cost effective, simple to use, tool that supports Philips CoolRunner EECMOS devices. The base programmer has an on-board 44-pin PLCC socket, and is all that is needed for CoolRunner devices in the A44 package*. For other parts and packages, adapter boards are available to plug into the base programmer board. Free software updates are available on the manufacturer's website: http://www.he.nt/~pds.

ORDERING INFORMATION

ORDER CODE	DESCRIPTION	12NC NUMBER
PZLCP MSC	CoolRunner low cost programmer	935246050112
PZSOL24 MSC	24 SOL adapter for low cost programmer	935247560112
PZTSSOP24 MSC	24 TSSOP adapter for low cost programmer	935247590112
PZPLCC28 MSC	28 PLCC adapter for low cost programmer	935247510112
PZPLC44A MSC	44 PLCC adapter for low cost programmer (only for PZX032C/N)	935261495112
PZTQFP44 MSC	44 TQFP adapter for low cost programmer	935247570112
PZPLCC68 MSC	68 PLCC adapter for low cost programmer	935247520112
PZPLCC84 MSC	84 PLCC adapter for low cost programmer	935247530112
PZPQFP100 MSC	100 PQFP adapter for low cost programmer	935247540112
PZTQFP100 MSC	100 TQFP adapter for low cost programmer	935247580112
PZLQFP128 MSC	128 LQFP adapter for low cost programmer	935247500112
PZPQFP160 MSC	160 PQFP adapter for low cost programmer	935247550112

*PZ3032CS/NS and PZ5032CS/NS in the 44-pin PLCC package require adapter PZPLC44A MSC

Full VHDL synthesis and automatic optimization software for Philips Semiconductors CoolRunners

PZVHDLEZ

FEATURES

- Full VHDL synthesis and automatic optimization for Philips Semiconductors CoolRunner™ devices
- The most affordable, complete VHDL design environment for programmable logic
- Runs on Windows™ NT and Windows 95
- Includes an interactive VHDL tutorial—quickly teaches VHDL synthesis and simulation
- Simple, easy to use interface
- Full feature VHDL synthesis with hierarchical language support
- Compliant with IEEE 1076–1993 and 1987, and Synopsys VHDL policy

DESCRIPTION

VHDL Easy is a ground-breaking programmable logic design product from MINC Incorporated that offers state-of-the-art Very high speed integrated circuit Hardware Description Language (VHDL) synthesis, optimization, and device-specific mapping. Included in the VHDL tool set is a professional VHDL interactive training software to help with learning both synthesis and simulation, as well as Philips' XPLA Designer tool suite, including CoolRunner fitters, verification tools, and PLD libraries. This allows the designer to target any Philips CoolRunner device, allowing them to take advantage of Philips total CMOS, high speed, low power ICS in their design.

VHDL technology allows designers to use top–down design methodologies at higher levels of abstraction. VHDL will quicken and simplify design description, enabling a faster completion of complex circuitry. Maintenance and debugging are simplified with VHDL as well.

SYSTEM REQUIREMENTS

- Windows 95 or Windows NT
- Intel-compatible Pentium-class processor recommended
- 16MB RAM minimum
- CD ROM
- SVGA (800 x 600, 256 color) display or better

ORDERING INFORMATION

ORDER CODE	DESCRIPTION	DRAWING NUMBER
PZVHDLEZ MSC	VHDL Easy for Philips CoolRunner CPLD	NA

™ Windows is a trademark of Microsoft Corporation

Section 7
Application notes

CONTENTS

AN055	Metastability Characteristics for Philips CPLDs	311
AN057	Altera (AHDL) to Philips (PHDL) design conversion guidelines	313
AN058	Cadence/Synopsys design flows for targeting Philips CPLDs	319
AN059	Mentor Graphics Design Flow for targeting Philips Semiconductors CPLDs	331
AN060	Using Data I/O-Model Technology VHDL tools to target Philips Semiconductors CPLDs	351
AN062	Understanding the hidden costs of using high–power CPLDs	369
AN063	Probing internal nodes using XPLA software graphic simulator	370
AN064	Using sum of products control terms in Philips CoolRunner™ CPLDs	375
AN065	Understanding CoolRunner™ clocking options	378
AN066	XPLA Designer™ hierarchical PHDL design support	382
AN068	Terminating unused CoolRunner™ I/O pins	399
AN069	ISP design considerations for CoolRunner™ CPLDs	400
AN070	Verilog implementation of a Manchester Encoder/Decoder in Philips CPLDs	405
AN071	OrCAD Express Design Flow for Philips CPLDs	440
AN072	Implementing a UART in Philips CPLDs	481
AN073	Synplicity/Model Tech design flow for targeting Philips CPLDs	489
AN074	OrCAD Capture Schematic/PHDL Design Flow for Philips CPLDs	515
AN075	Clock modulation: How to synthesize additional clocks for counters and state machines	532
AN076	Using the Philips PZ3960 Evaluation Board	536
AN078	VHDL Easy Design Flow for Philips CPLDs	568
AN079	Viewlogic Intelliflow Design Flow for Philips CPLDs	598

Metastability Characteristics for Philips CPLDs

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 Philips CPLDs 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'.

Metastable Characteristics

Table 1 presents the metastability data for Philips 3 V CoolRunner CPLDs, manufactured on our 0.5 µm EECMOS process. Table 2 presents this data for Philips 5 V CoolRunner CPLDs.

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.

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:

$$MTBF = \frac{e^{(t'/\tau)}}{(T_0\ F_c\ F_1)}$$

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. 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. Metastability data for Philips 3 V CPLDs

	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

Table 2. Metastability data for Philips 5 V CPLDs

	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

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. This document also addresses pin capability issues between Altera's MAX7000 family and the Philips CoolRunner XPLA1 family.

Terminology

AHDL	Altera Hardware Description Language
CPLD	Complex Programmable Logic Device
CR32	CoolRunner 32 Macrocell CPLD
CR32CS	CoolRunner 32 Macrocell ISP and Enhanced Clocking CPLD
CR64	CoolRunner 64 Macrocell CPLD
CR64CS	CoolRunner 64 Macrocell ISP and Enhanced Clocking CPLD
CR128	CoolRunner 128 Macrocell CPLD
CR128CS	CoolRunner 128 Macrocell ISP and Enhanced Clocking CPLD
CR320	CoolRunner 320 Macrocell CPLD
CR960	CoolRunner 960 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
XPLA1	Philips CoolRunner CPLD family ranging from 32 to 128 Macrocells

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

The clocking approach taken is dependent on which CoolRunner family is being used. The XPLA1 enhanced clocking CPLDs (CR32CS, CR64CS, and CR128CS) have more clocking capabilities than the XPLA1 non-enhanced clocking CPLDs (CR32, CR64, and CR128). The XPLA1 non-enhanced clocking CPLDs have either 2 or 4 clock pins, depending on macrocell count (see Table 2). The XPLA1 enhanced clocking CPLDs have the same clocks pins but also contain 2 additional control term clocks per logic block. These control term clocks can be used to make any input a clock resource. Therefore, it is much easier to match the pinout of a corresponding Altera CPLD to a XPLA1 enhanced clocking device than to a XPLA1 non-enhanced clocking device. The main constraint for the XPLA1 enhanced clocking devices to be pin compatibility (in the context of clocking constraints) with the Altera MAX7000 family is the number of available clocks. As long as the maximum number of clocks is not exceeded, the XPLA1 enhanced clocking CPLDs should be pin compatible. Table 1 gives the number of clocks provided by the XPLA1 enhanced clocking CPLDs.

The approach is much different with the XPLA1 non-enhanced clocking CPLDs. 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.

Table 1. XPLA1 Enhanced Clocking Clock Resources

Device	# of Clocks Pins	# of Control Term Clocks	Total # of Clocks
CR32CS	2	4	6
CR64CS	4	8	12
CR128CS	4	16	20
CR3320	8	32	40
CR3960	8	96	104

Table 2. XPLA1 Non-enhanced Clocking Clock Pinouts

Package/Device	Clock Number	Clock Type	44 PLCC	44 TQFP	68 PLCC	84 PLCC	100 PQFP	100 TQFP	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	87	114	139
	CLK1	Sync/Async				44	42	40	53	62
	CLK2	Sync/Async				41	39	37	50	59
	CLK3	Sync/Async				4	94	92	119	144

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.

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 3 lists AHDL keywords in the first column and gives the PHDL equivalents in the second column.

Table 3. 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 4 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 4. 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 5 shows the AHDL and PHDL dot extension notations for connections to macrocell logic.

Table 5. 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	S 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

Altera (AHDL) to Philips (PHDL) design conversion guidelines

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 of 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, Applications and Architecture Development Manager. If you need more information, please contact me at 505–858–2790 or call the Philips CPLD Technical Support Line at 1–888–COOLPLD (1–888–266–5753) or 505–858–2996; or send email to coolpld@scs.philips.com.

Cadence/Synopsys Design Flows for targeting Philips CPLDs

AN058

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 the following lines in ad_decoder_tf.v.

$shm_open("ad_decoder.shm");

$shm_probe("AS");

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 is available from CPLD Applications.

A schematic can also be generated based on the modgen-generated verilog model.

Mentor Graphics Design Flow for targeting Philips CPLDs

AN059

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.:

Quickvhdl steps

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 &

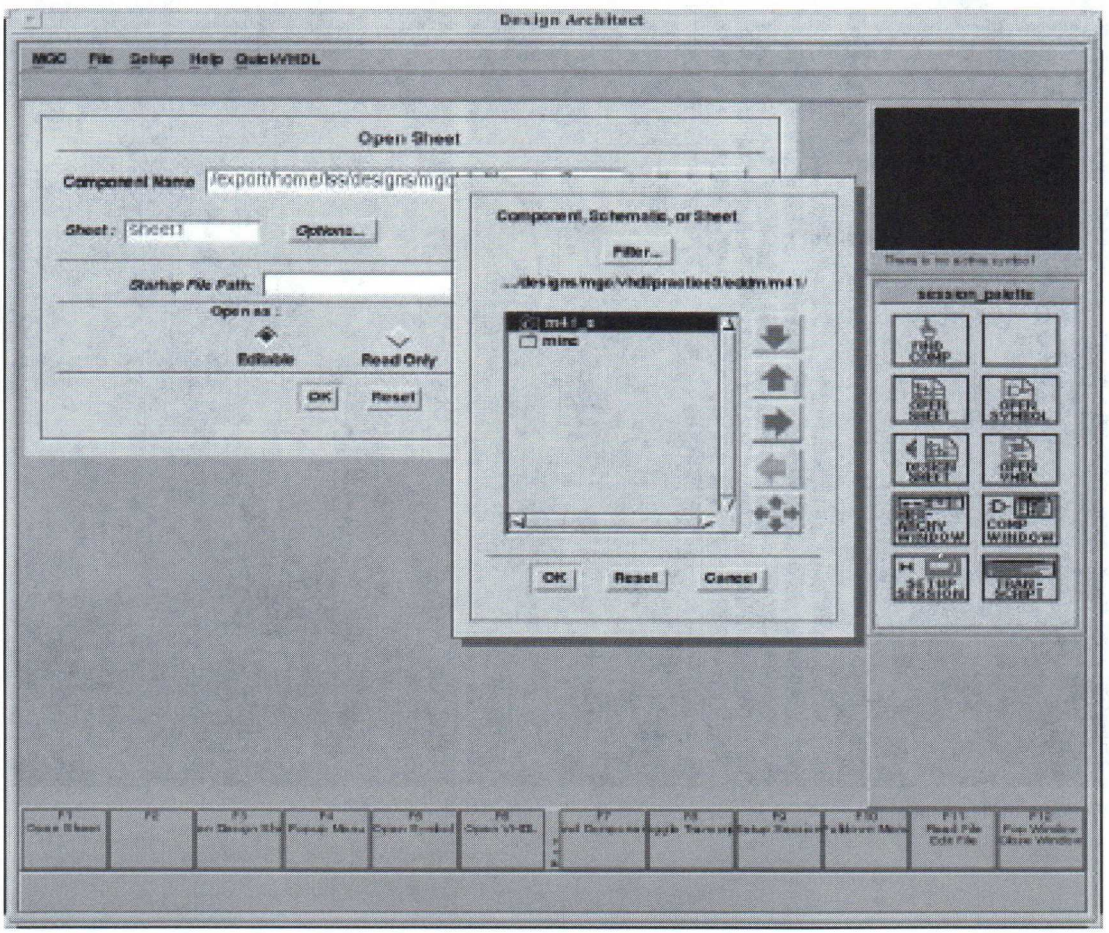


Figure 1. Opening a schematic in Design Architect..

Open the schemaitc by selecting Open Sheet in the palette window and selecting m41_s as shown.

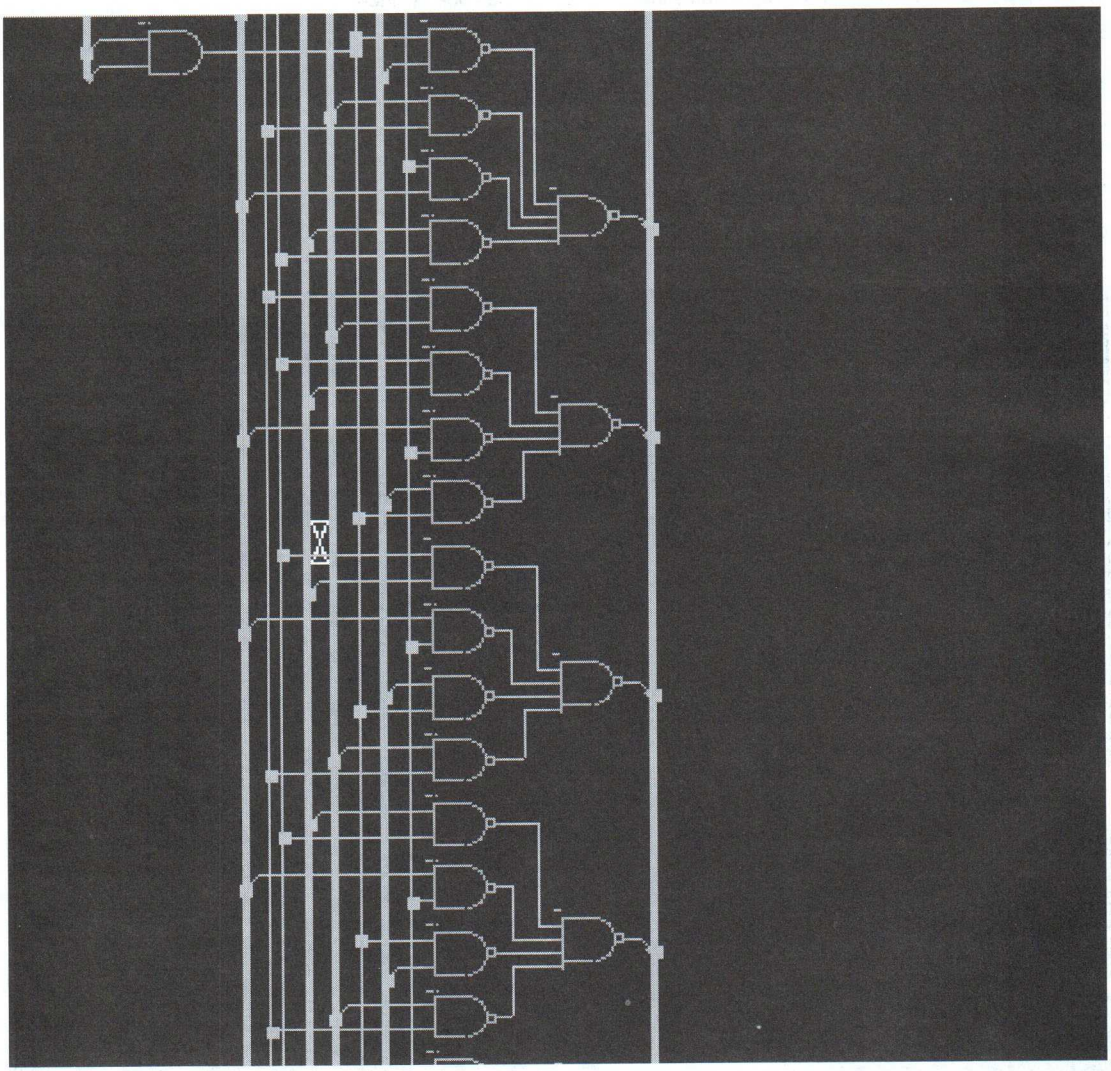


Figure 2. m41_s schematic

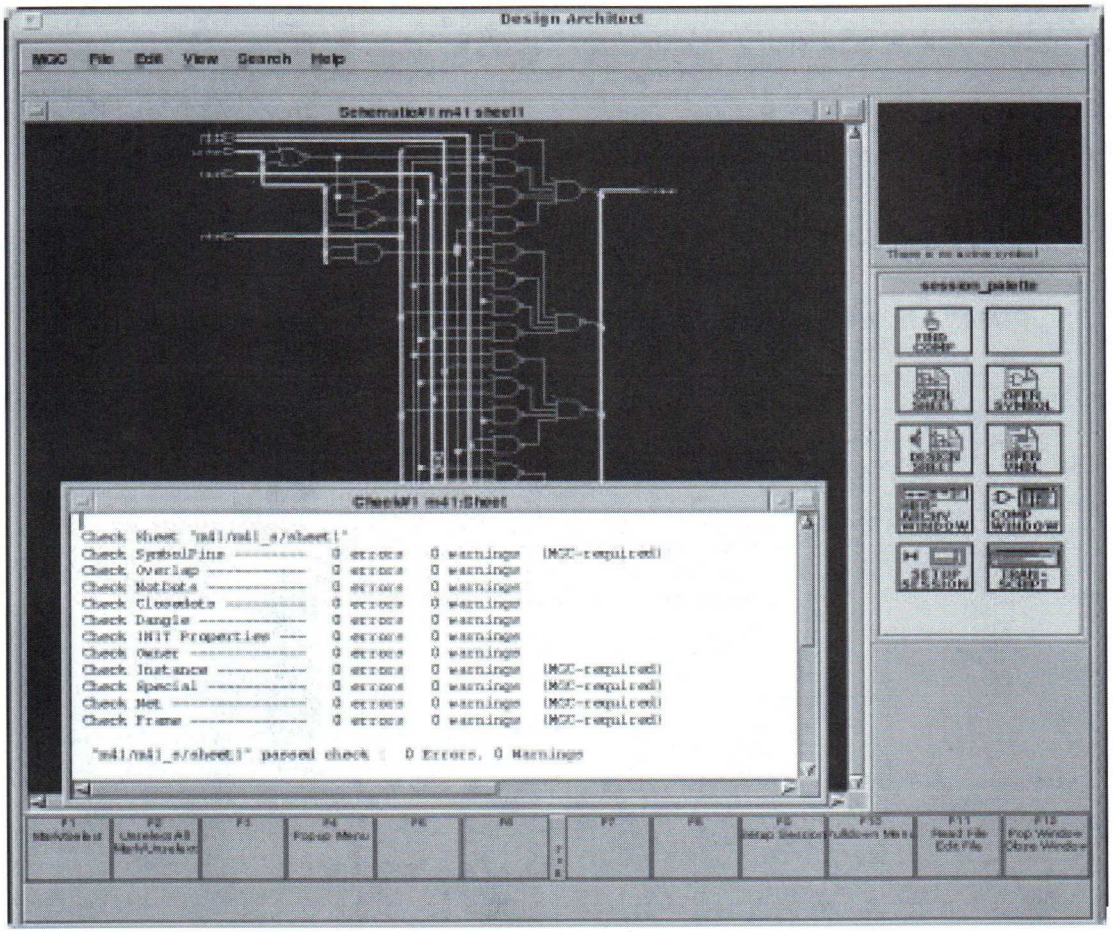


Figure 3. Electrical rules check results

With the schematic open, use the pull down menus from the Menu bar Check to ensure that there are no electrical rule violations.

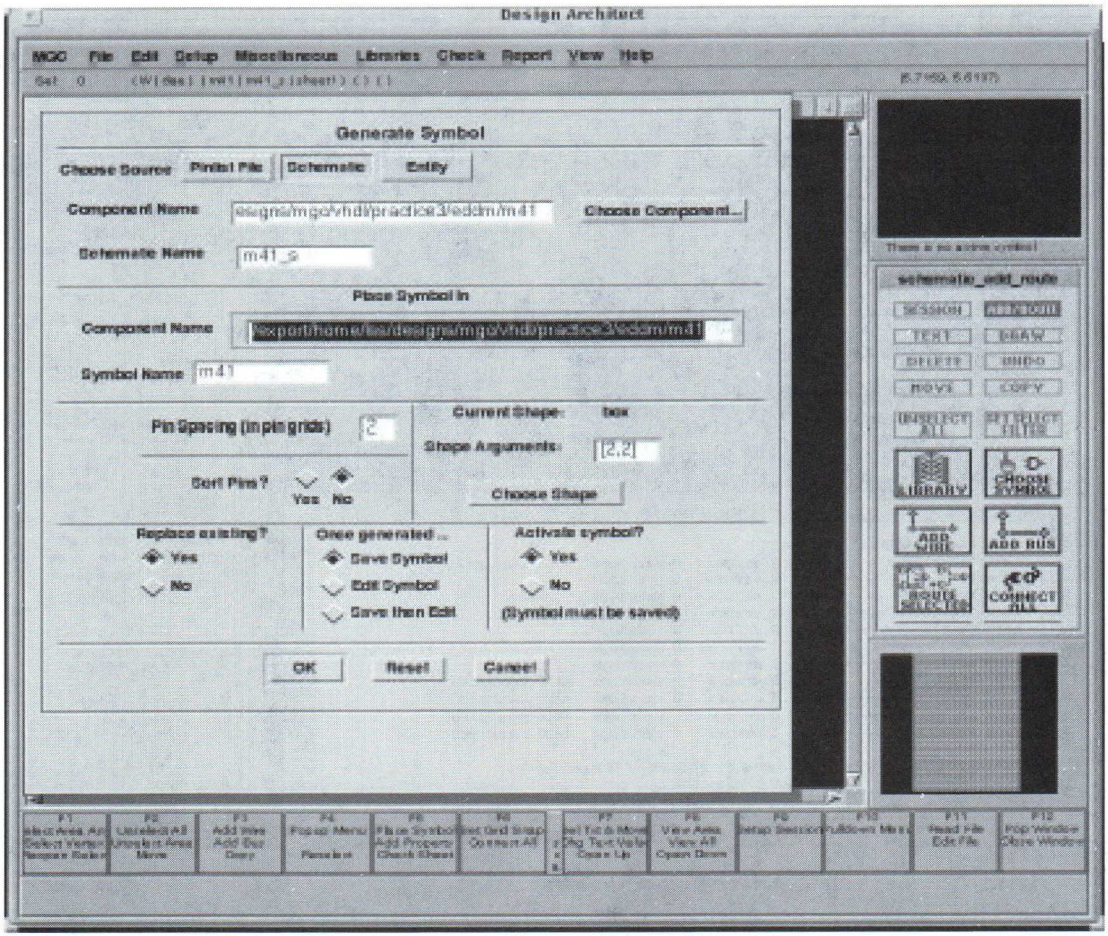


Figure 4. Generating a symbol

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.

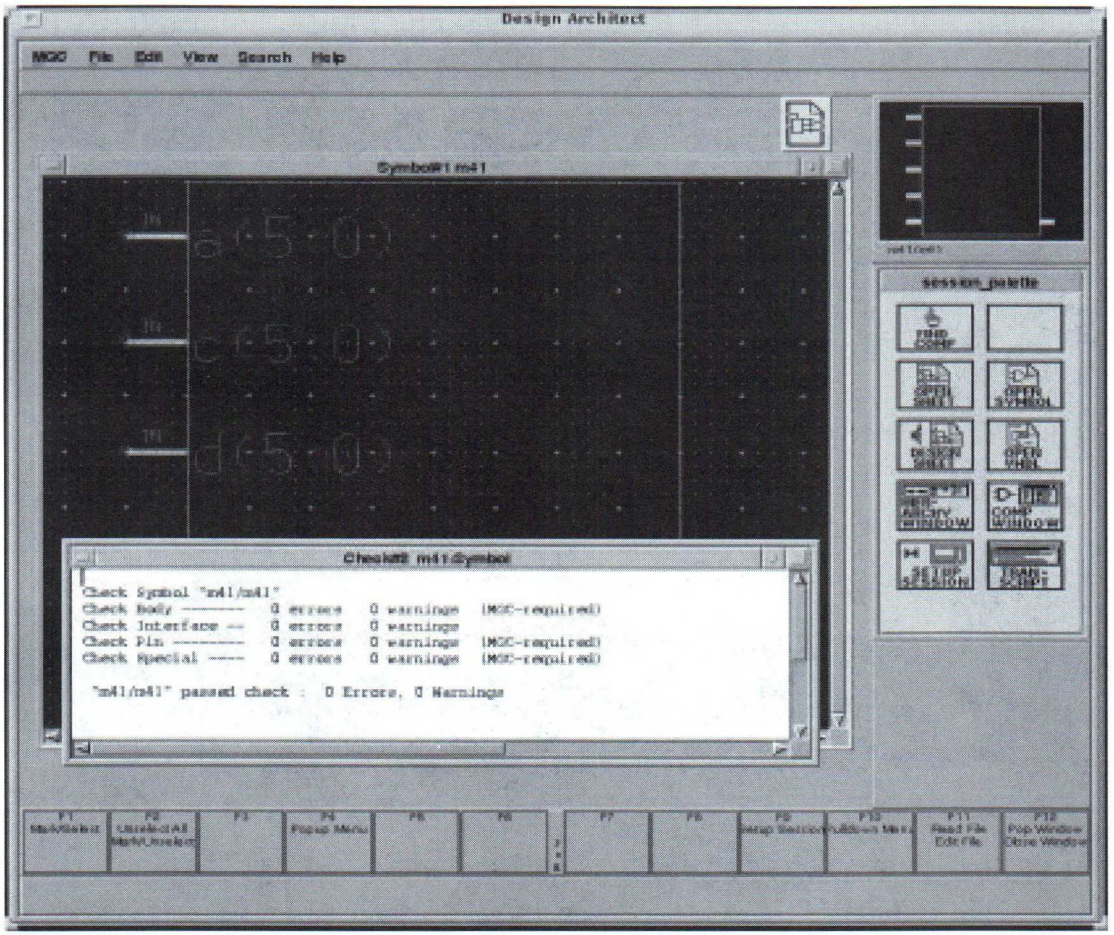


Figure 5. Check the symbol

Close the schematic (upper right corner). Use the palette to Open Symbol. From the Menu bar, check the symbol with default registration.

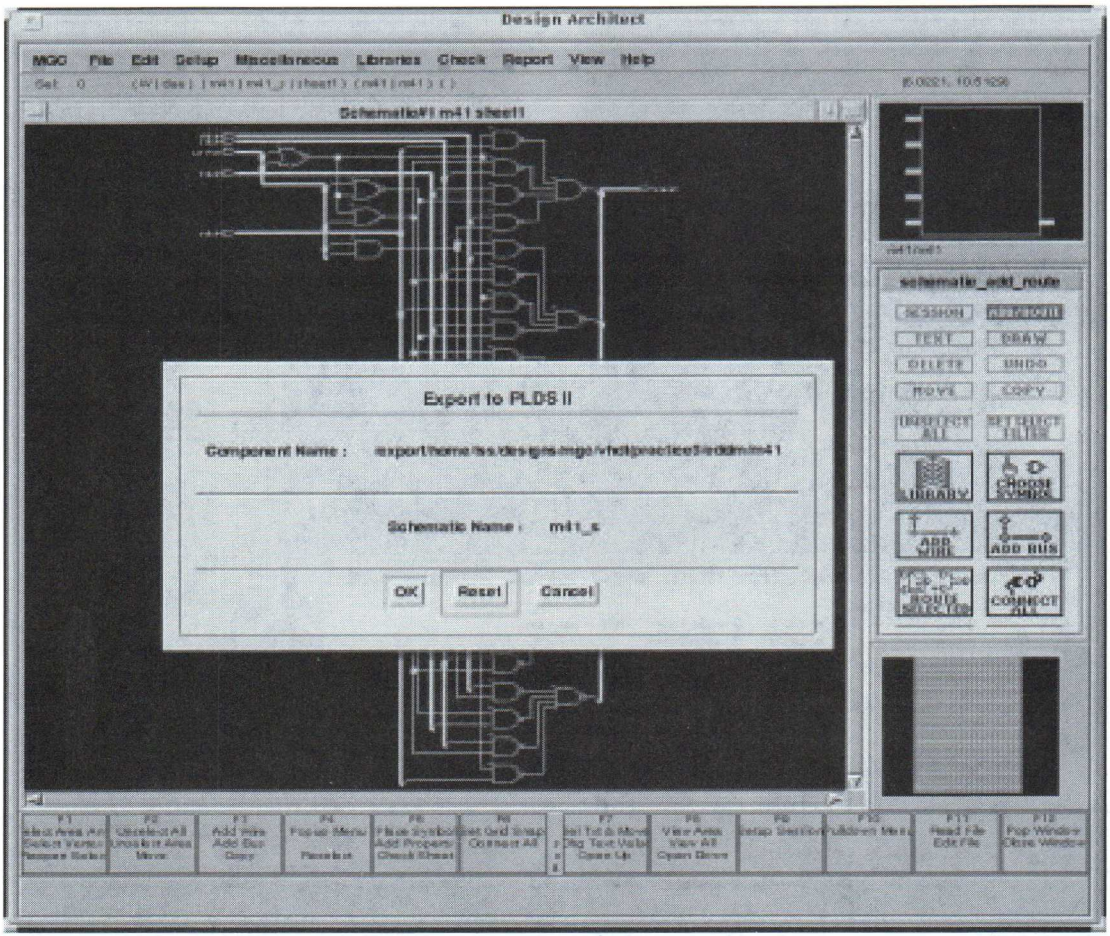


Figure 6. Export to PLDSII

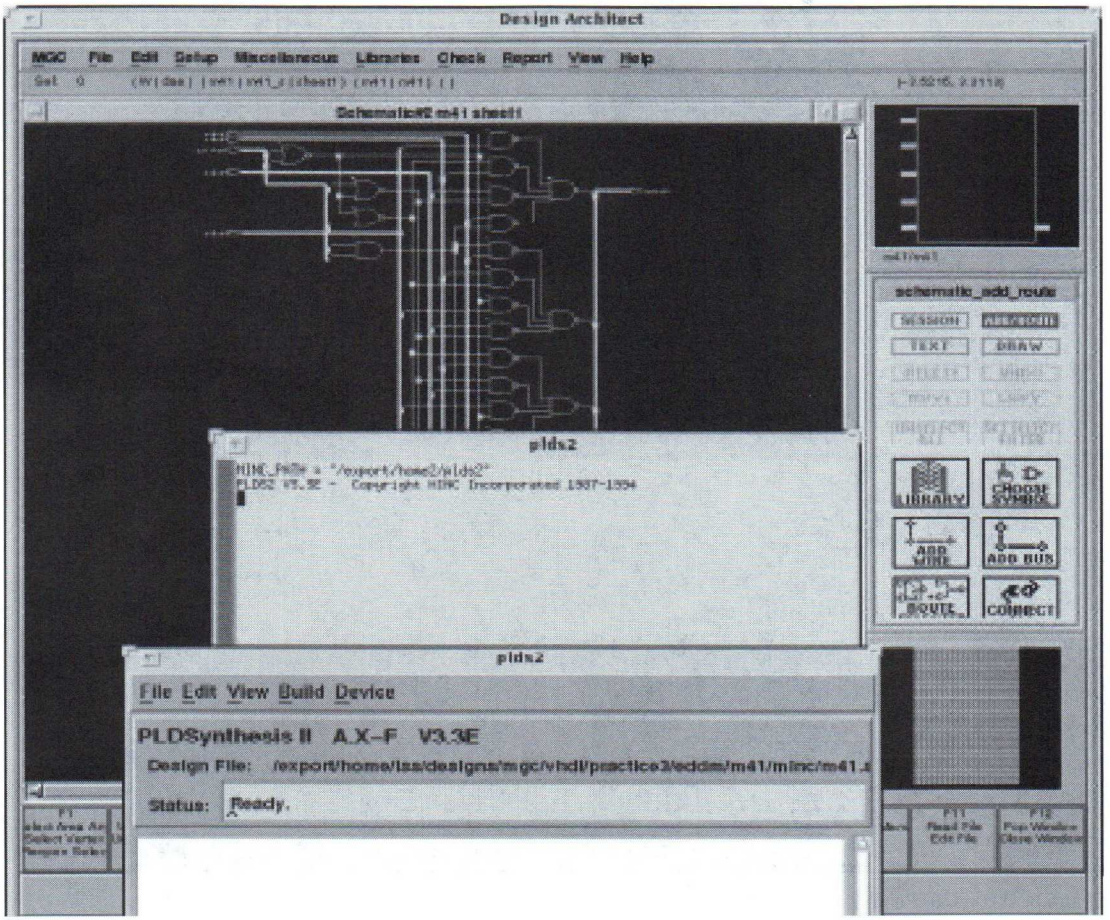


Figure 7. Invoking plds2

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.

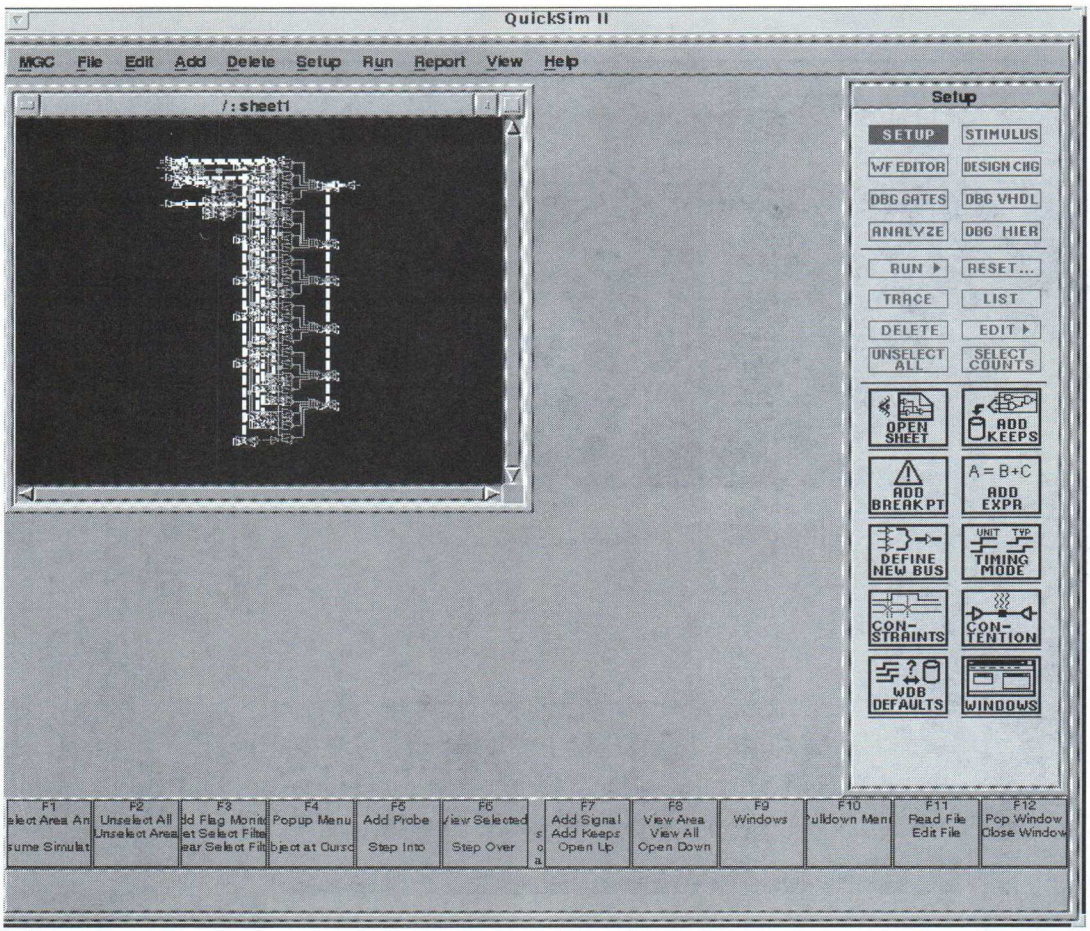


Figure 8. Using Quicksim II

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.

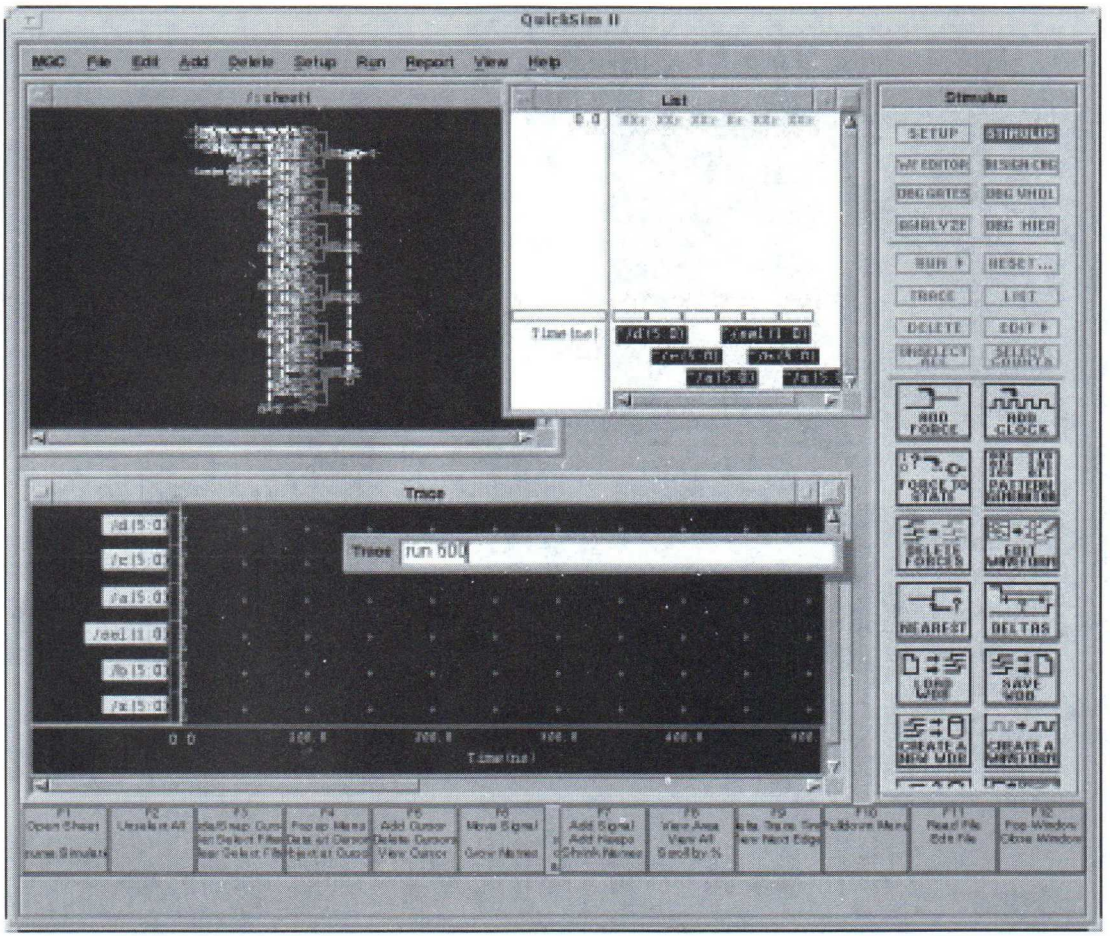


Figure 9. Schematic, Trace, and List windows displayed.

To run a simulation, from the palette rnenu, enter Stimulus and provide the input stimuli. Enter run 500 to run a simulation.

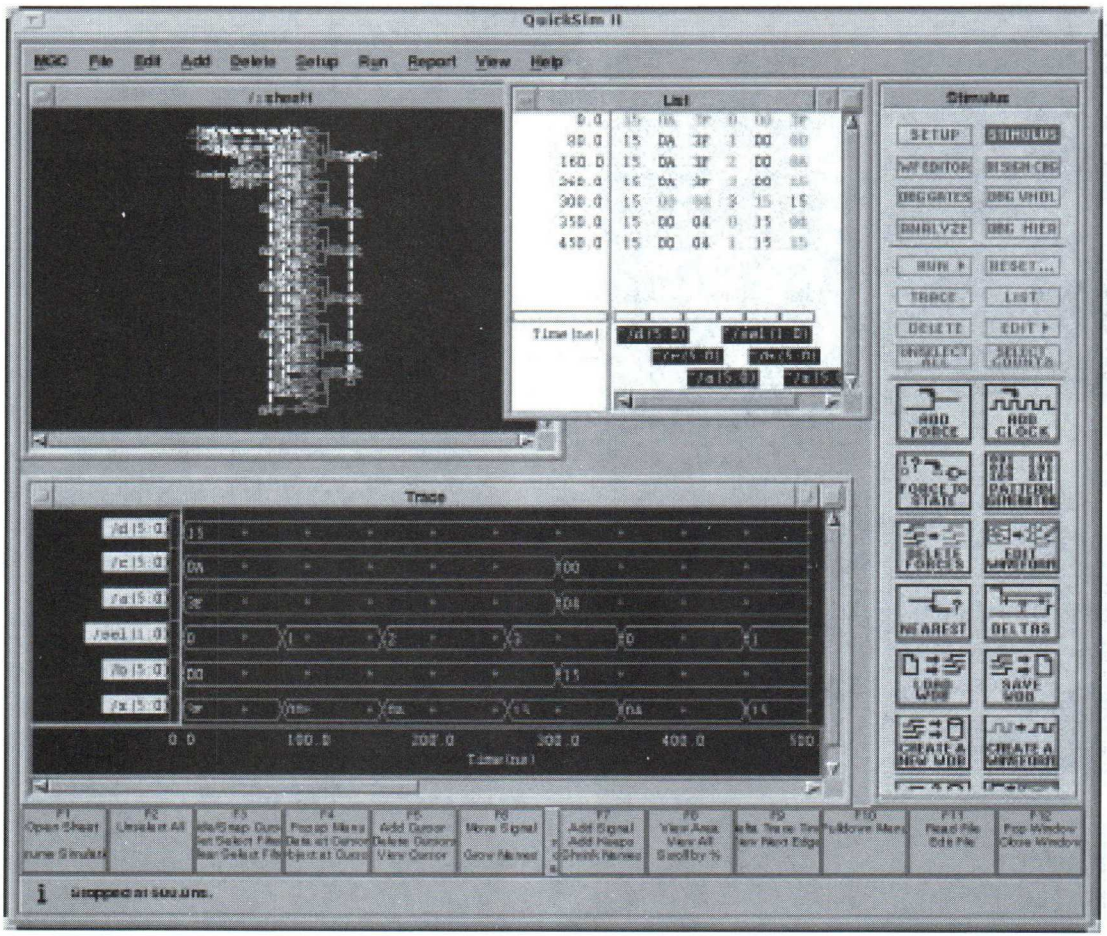


Figure 10. Simulation results

Using Data I/O-Model Technology VHDL tools to target Philips CPLDs

AN060

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.

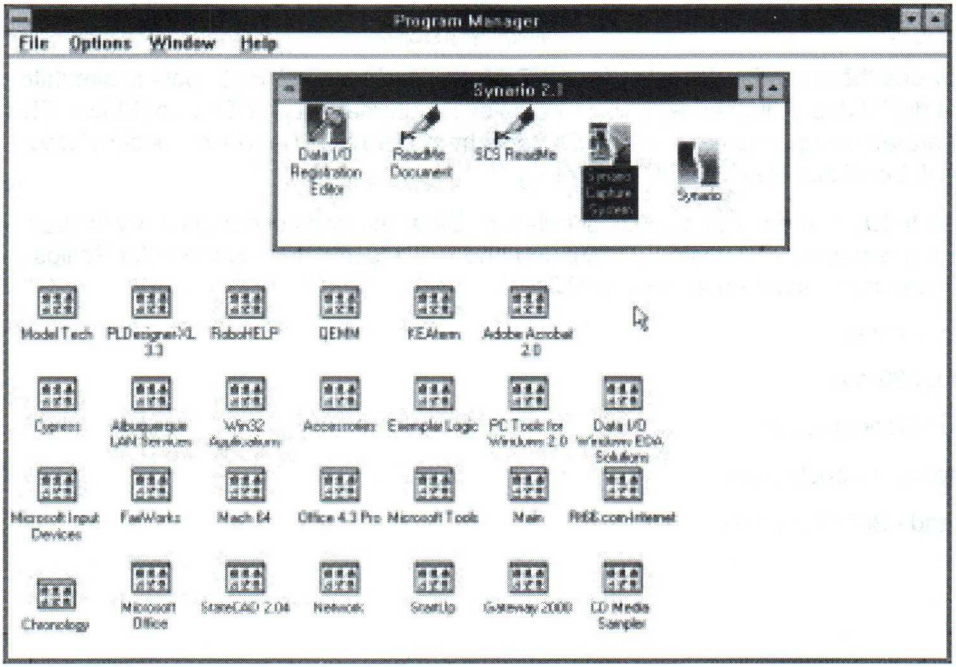


Figure 1: Program Manager and Synario 2.1 Group

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.

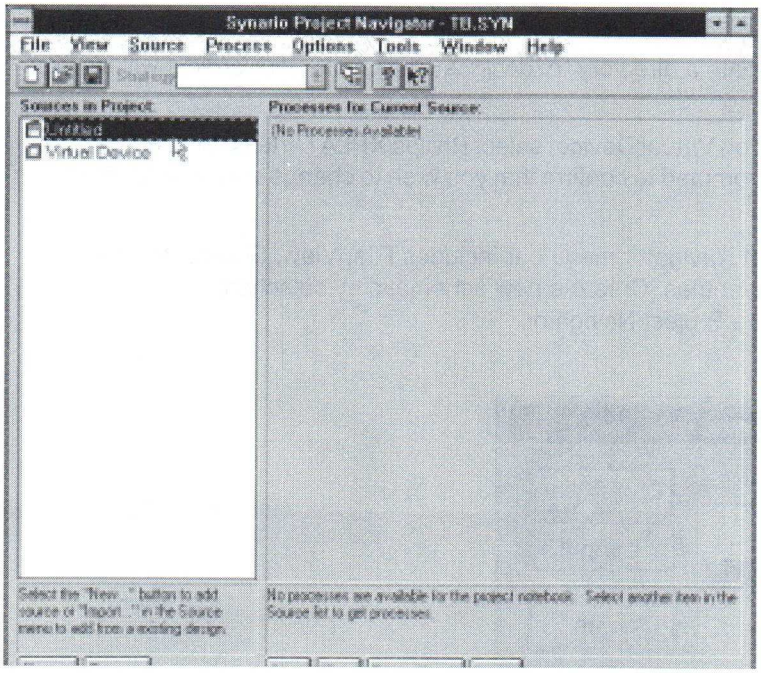


Figure 2: Project Navigator window

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.

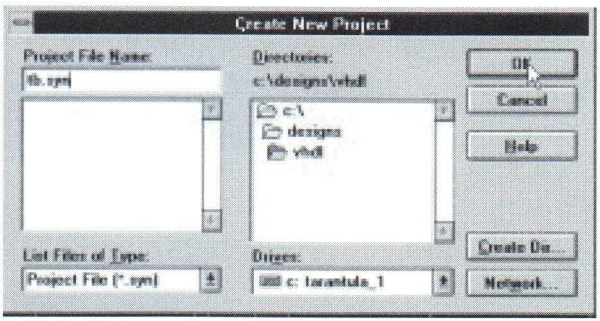


Figure 3 Project Navigator Dialog Box

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

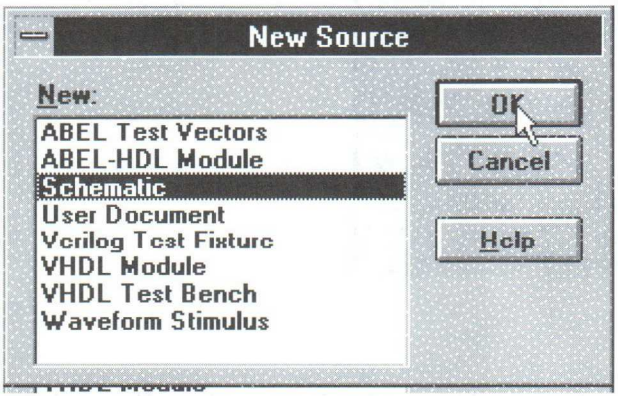


Figure 4: Creating a schematic

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.

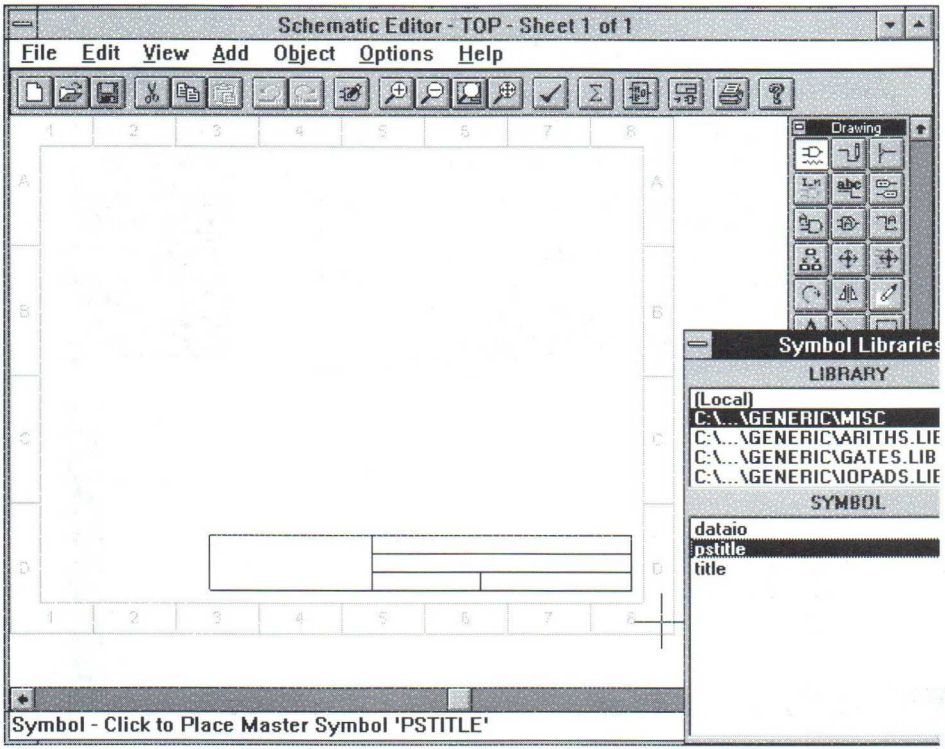


Figure 5: Adding a title box in the schematic editor.

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.

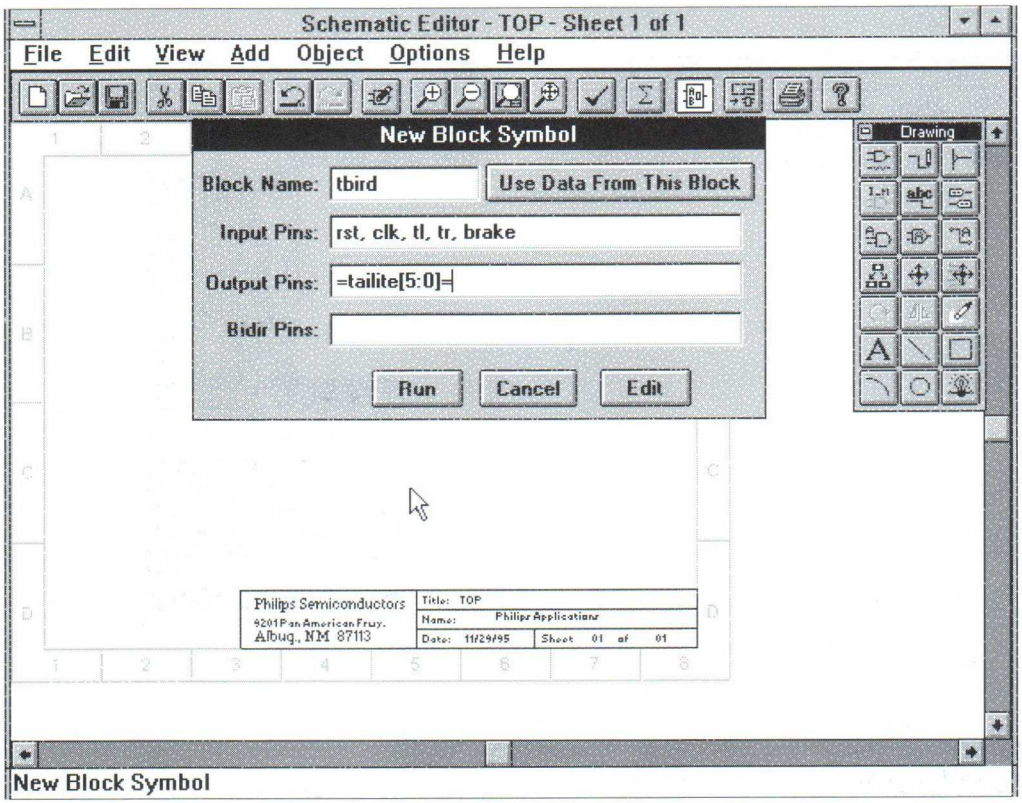


Figure 6: Schematic Editor - Creating a symbol

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.

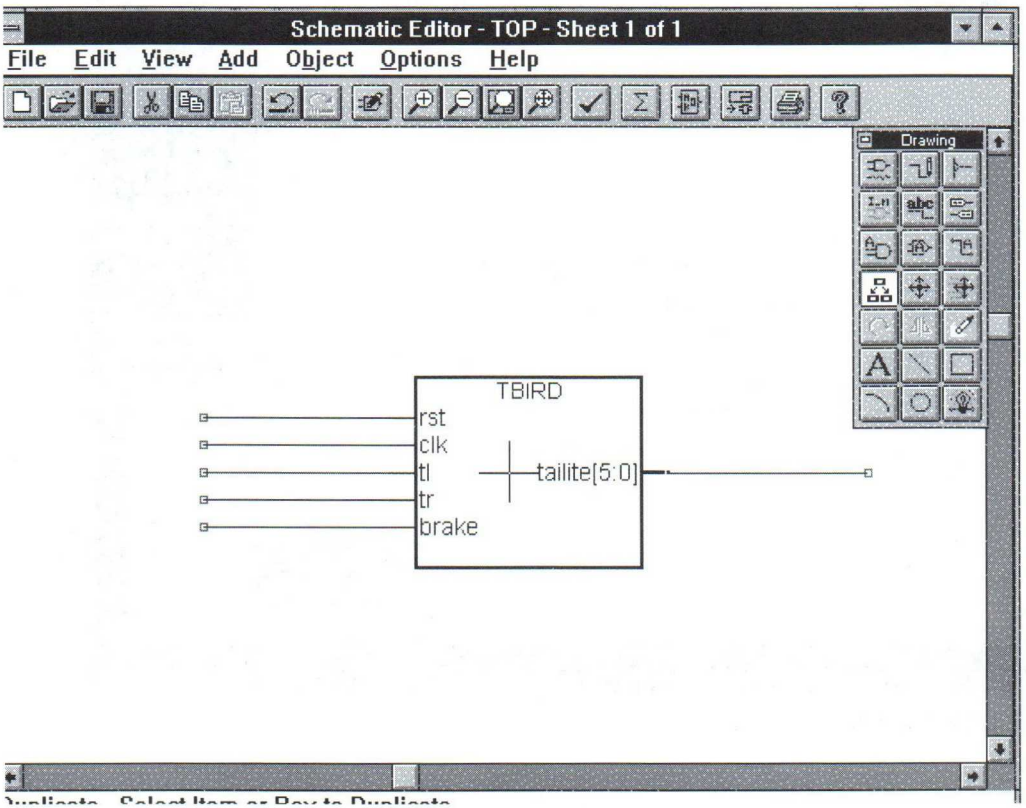


Figure 7: Adding wires on a schematic

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.

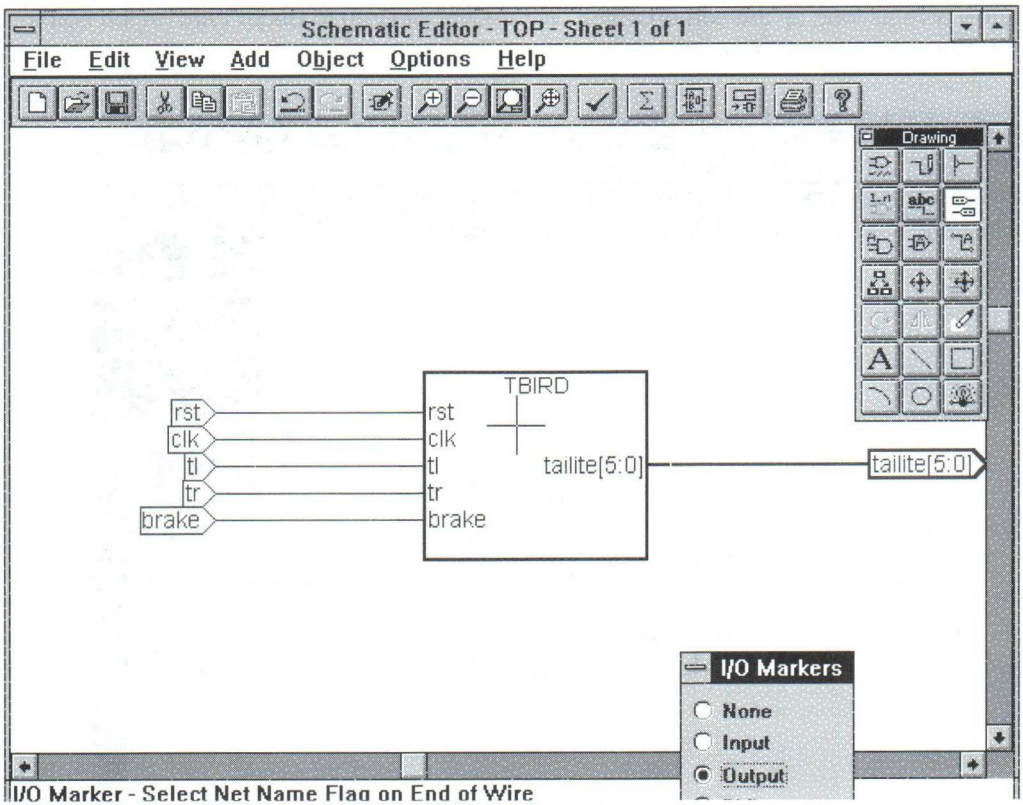


Figure 8: Labeling nets and adding I/O markers

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.

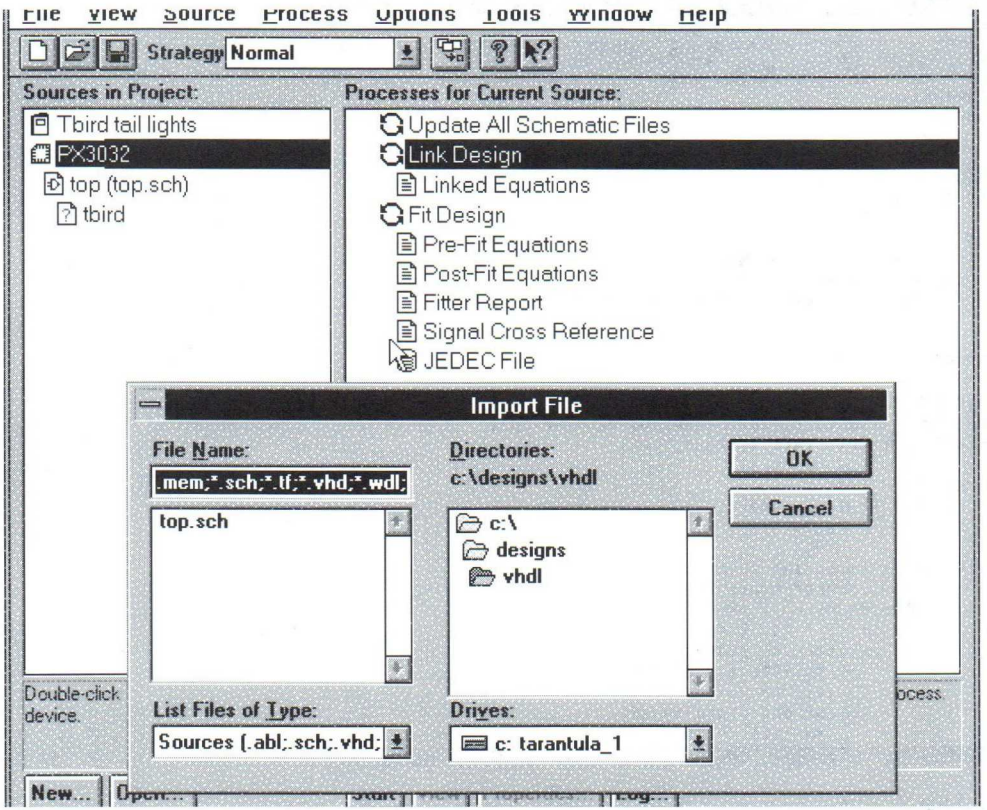


Figure 9: Project Navigator After Schematic and Symbol Creation.

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.

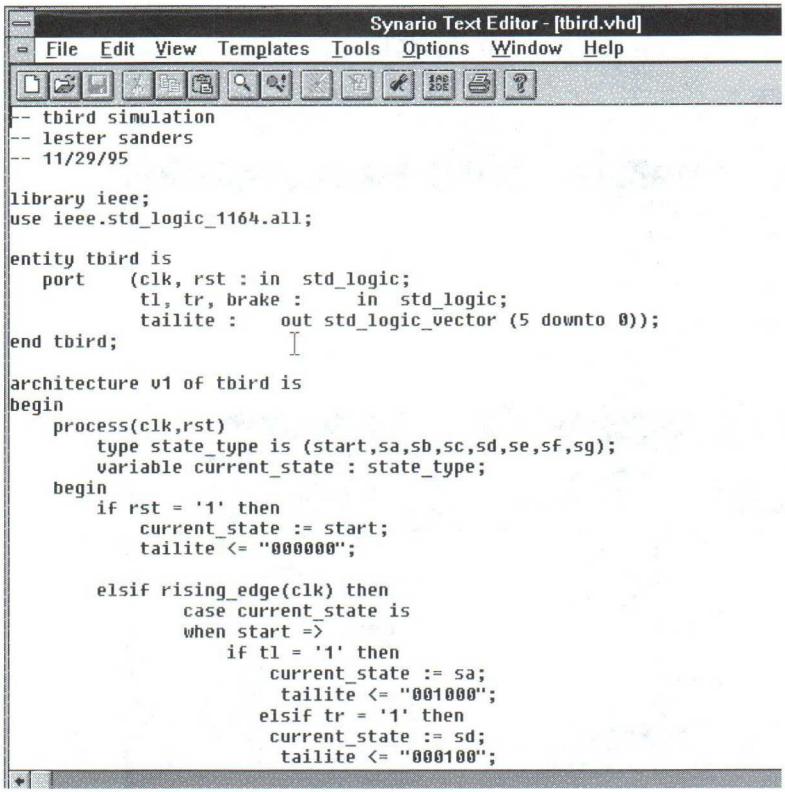


Figure 10: VHDL Source

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.

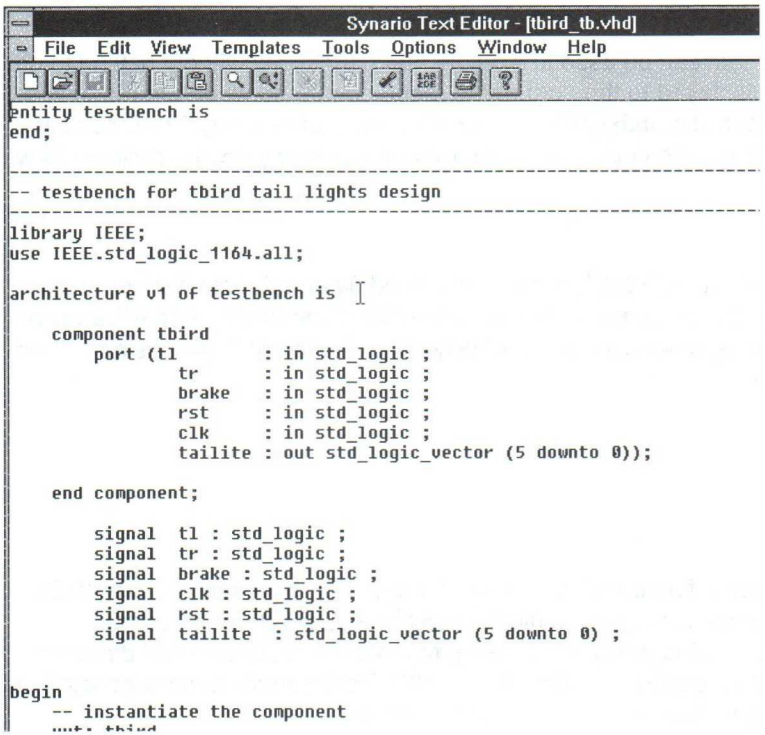


Figure 11 VHDL test bench tb_sim.vhd

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).

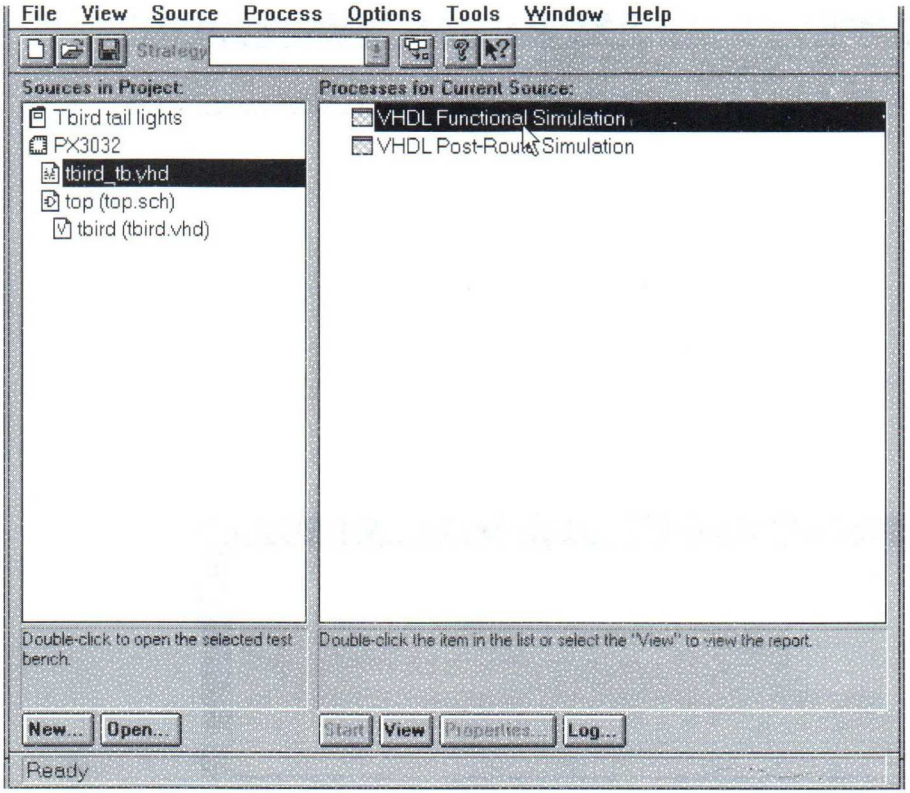


Figure 12: Simulation window

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 13, change directory to the design directory by entering cd \designs\vhdl\.

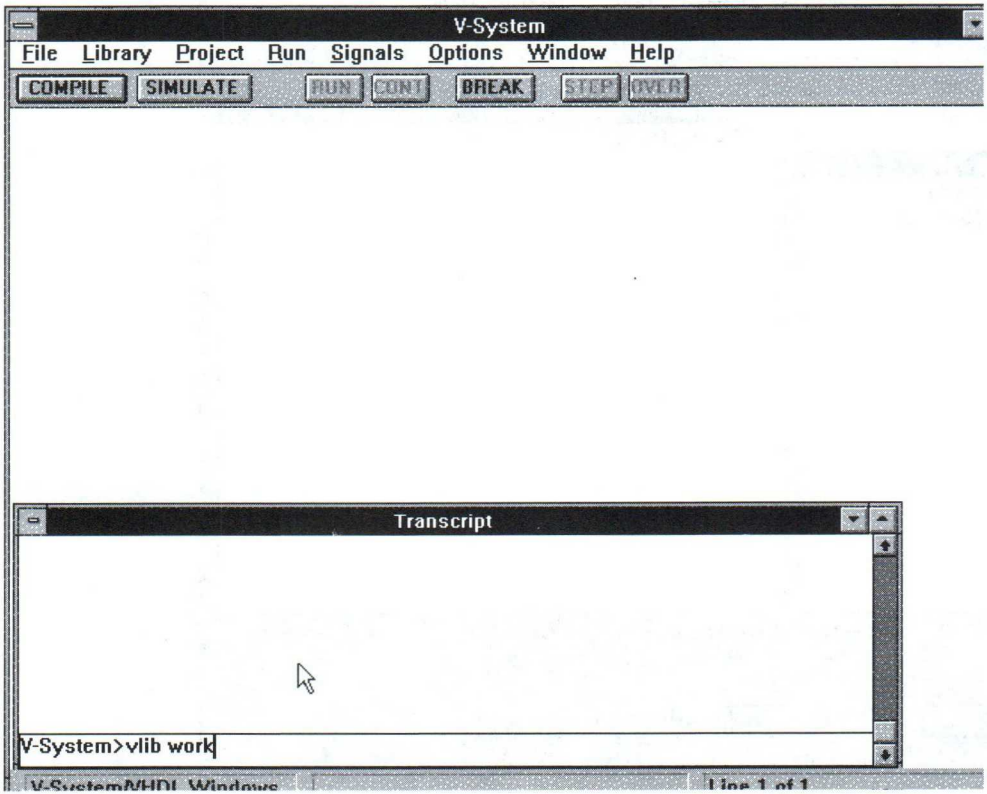


Figure 13: Entering commands in the transcript window

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 14.

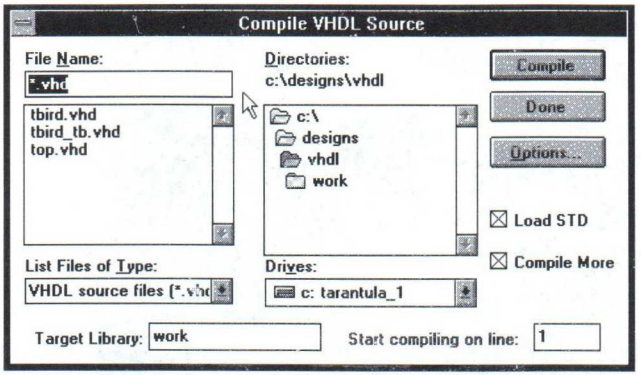


Figure 14: Compiling files

.

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 15, select Windows- Restore All followed by Windows-Tile Vertically from the menu bar.

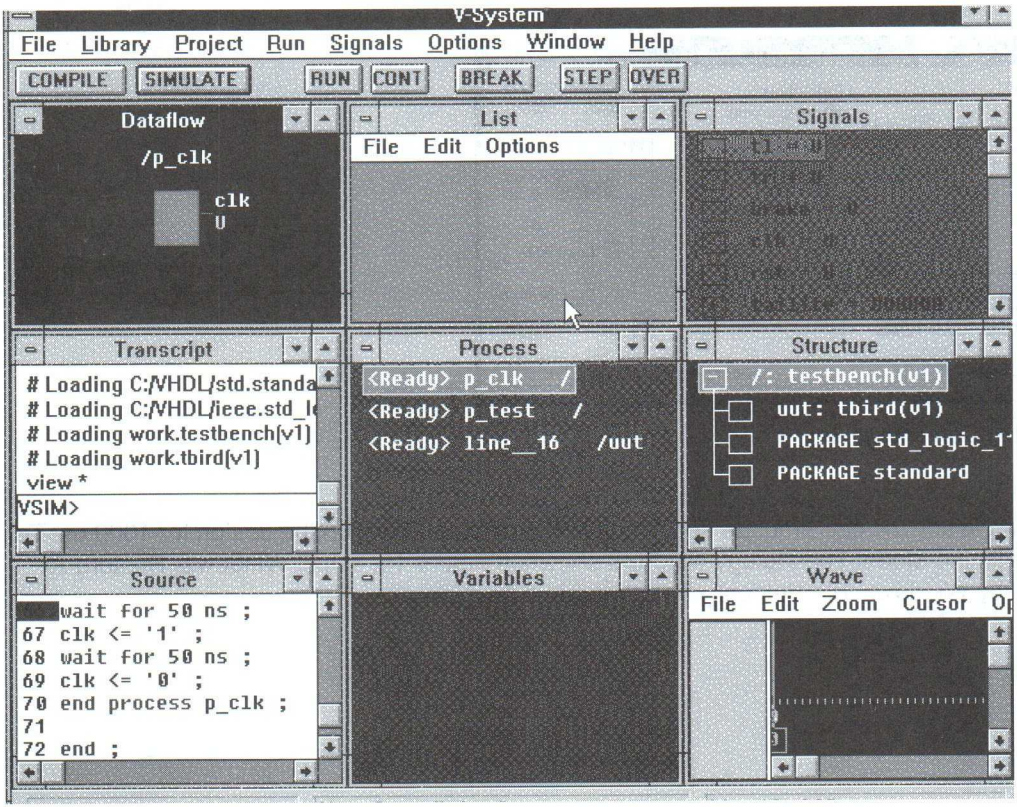


Figure 15: Simulation windows

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 16, 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.

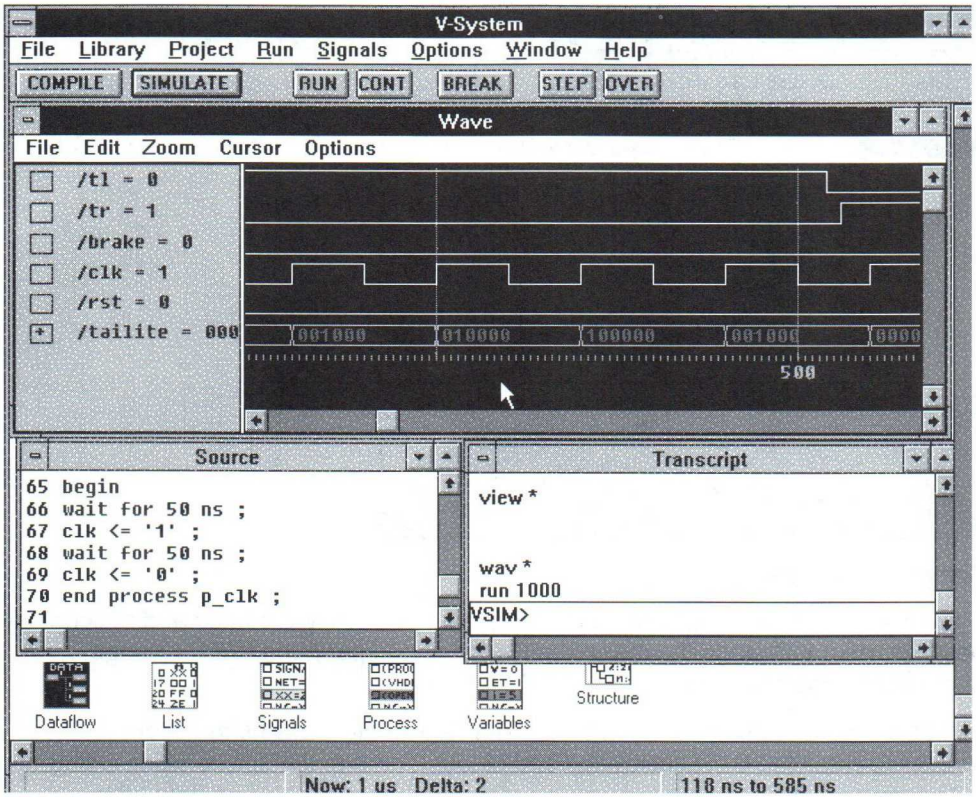


Figure 16: Waveform display

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 17). 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”.

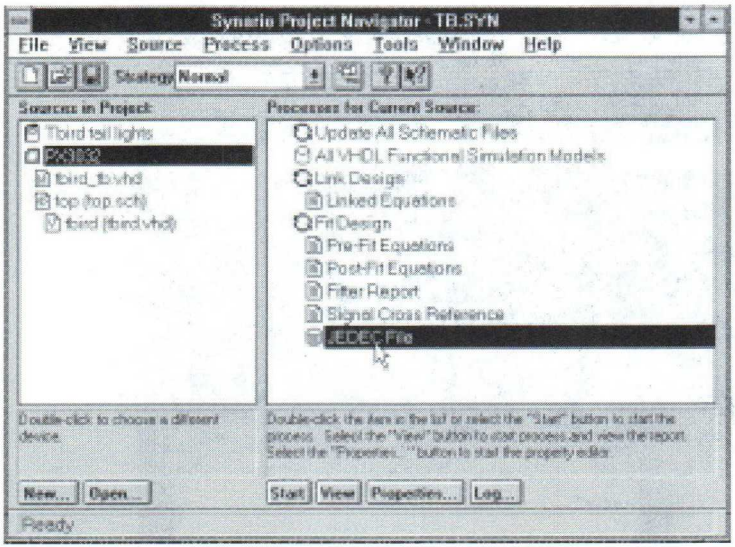


Figure 17:Compilation the design

As seen in Figure 17, 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 18) 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     : tb                                    |
ICE     : px3032                                |
TER     : xplafit   v8.10                       |
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 18: Partial display of fitter report

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 software 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 Software 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 Software 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 Software.

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

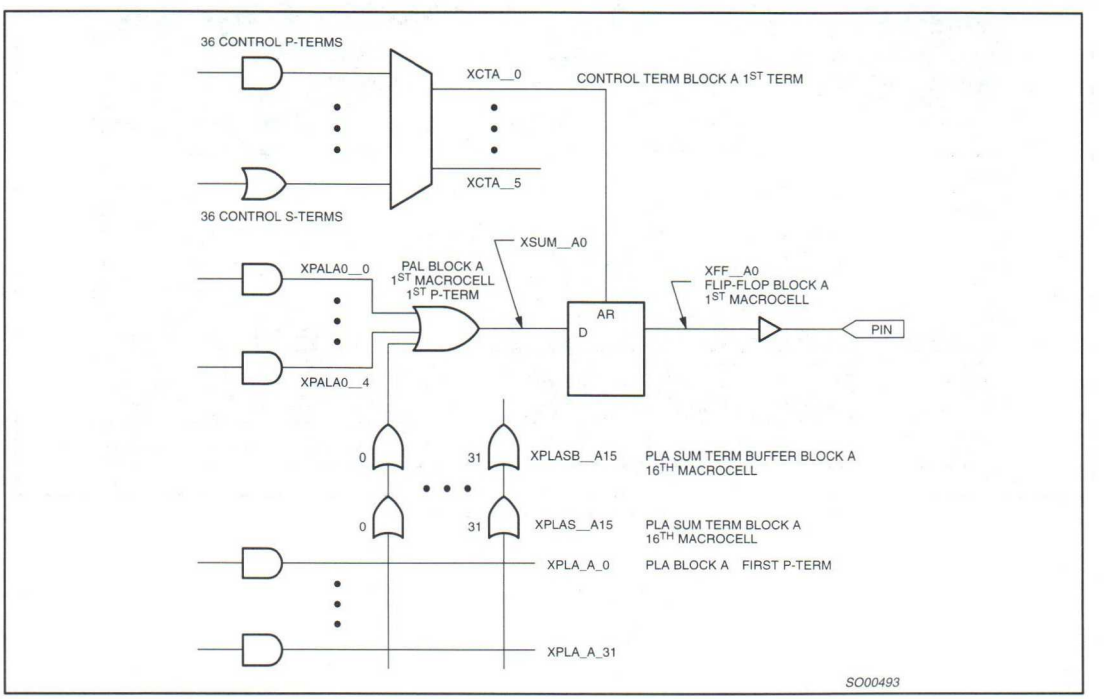


Figure 3. Internal Node Schematic

Probing internal nodes using XPLA software graphic simulator

AN063

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 Software timing simulator, let's run the simulator and add an internal node. First, start XPLA Software 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 Software 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.

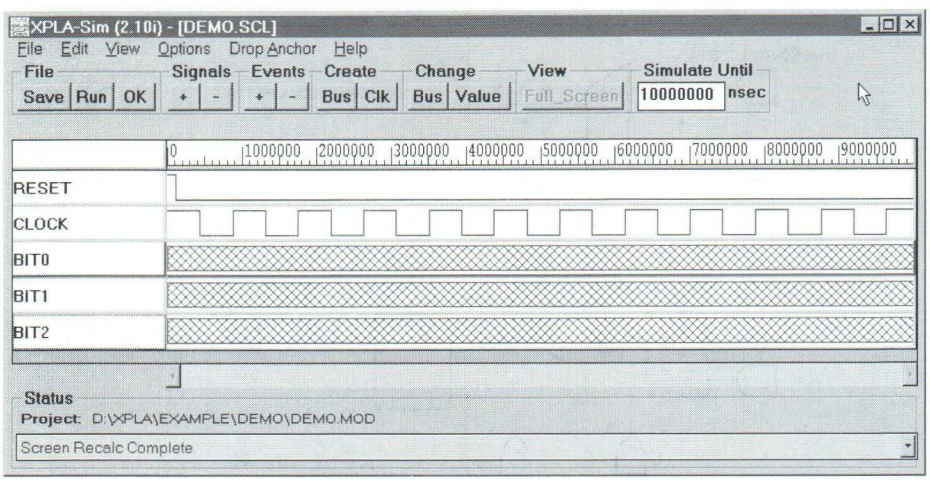


Figure 4. About to Click on File

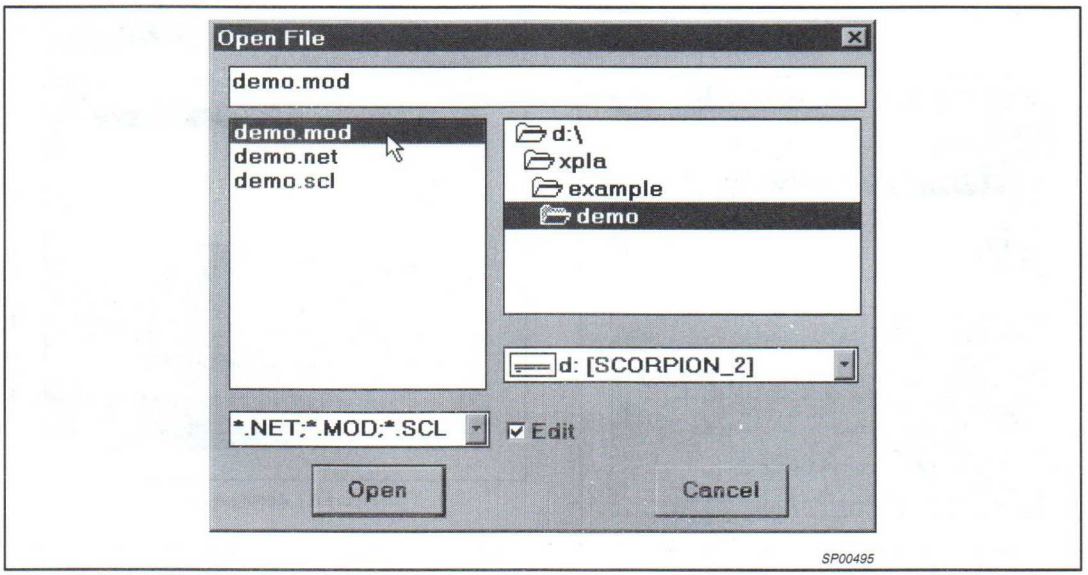


Figure 5. About to Edit .MOD File

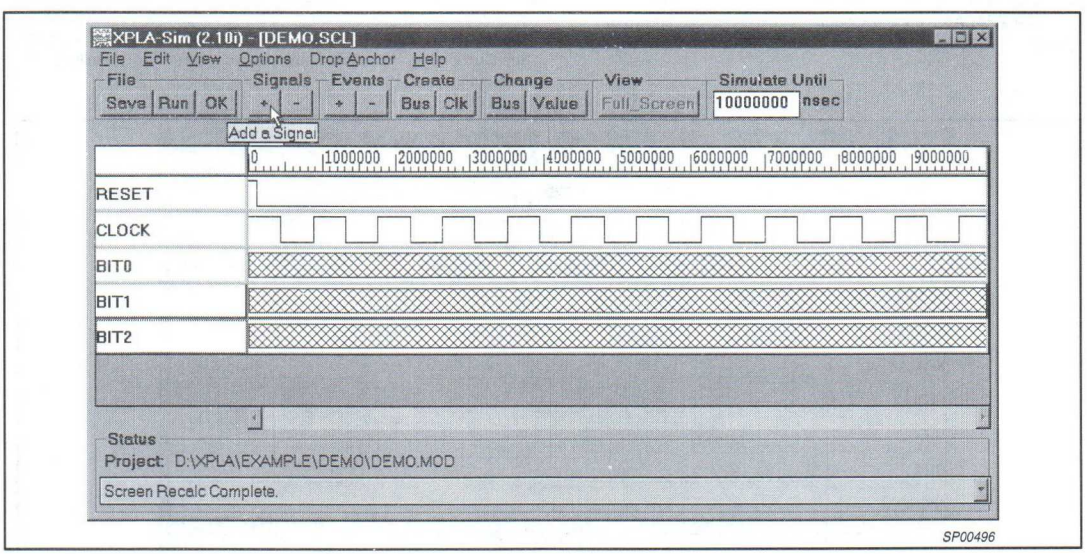


Figure 6. Adding Signals

Click the **Signals +** button. A dialog box will appear that looks like Figure 7.

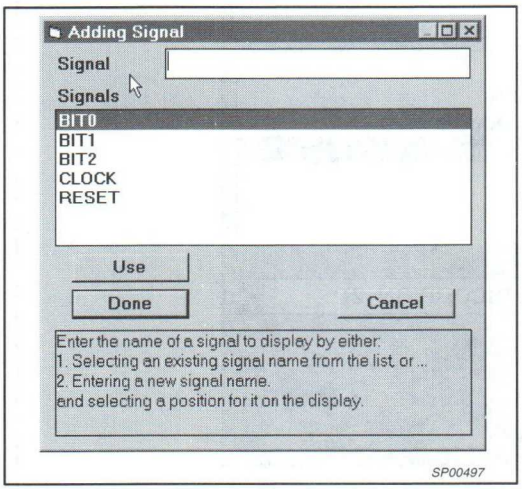


Figure 7. Adding Signal Dialog Box

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.

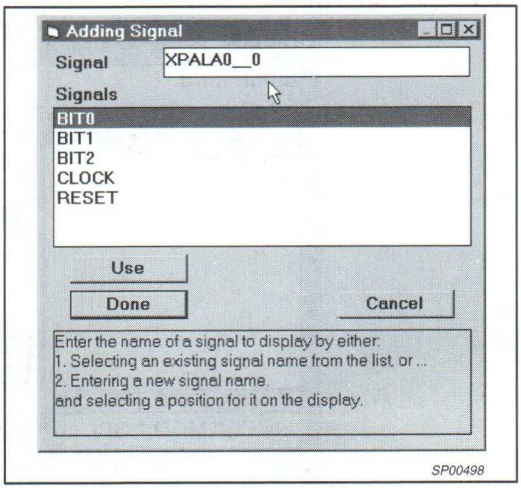


Figure 8. Adding A Signal

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.

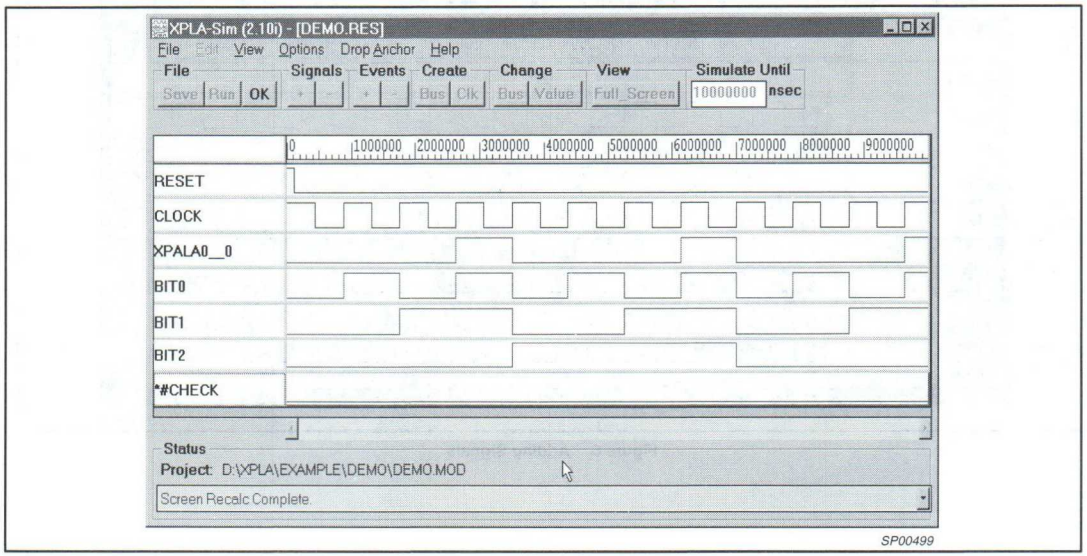


Figure 9. Simulation with Internal Node Signal Added

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, output enables, and clocks. 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.

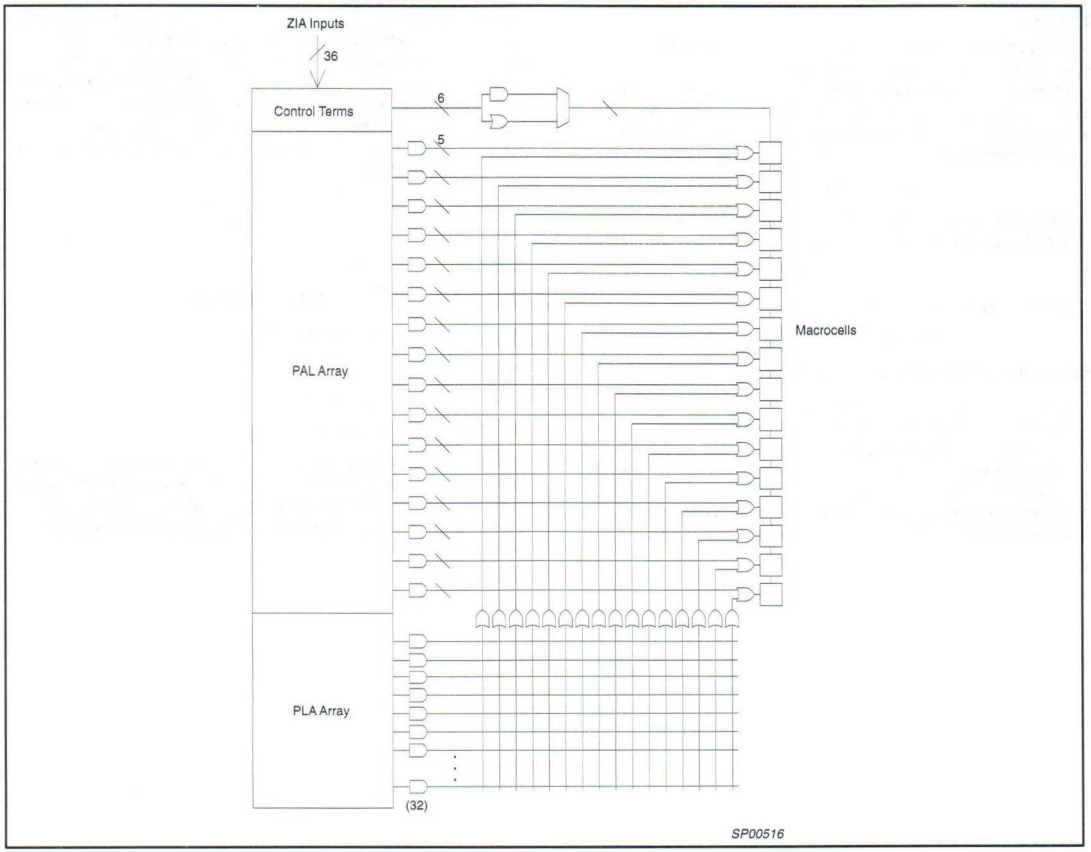


Figure 1 Philips XPLA Logic Block

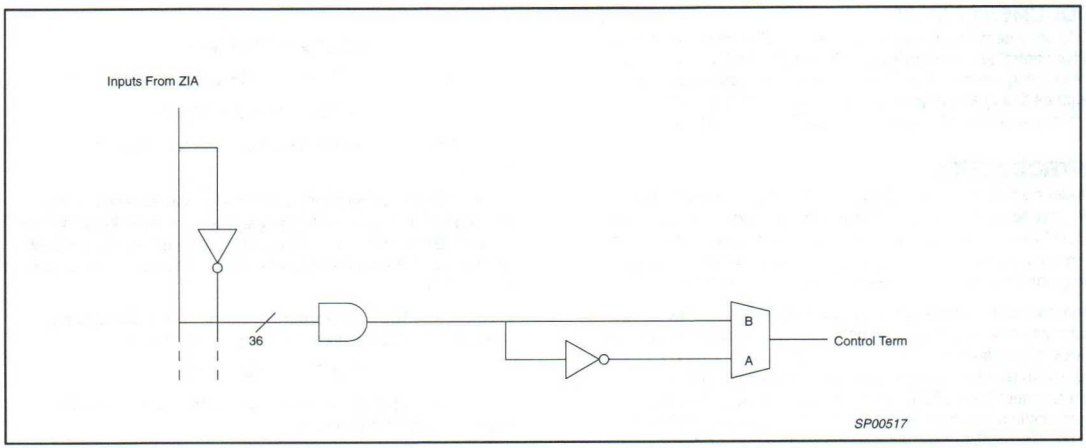


Figure 2. Product Term Implementation for Control Terms in Philips CPLDs.

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.

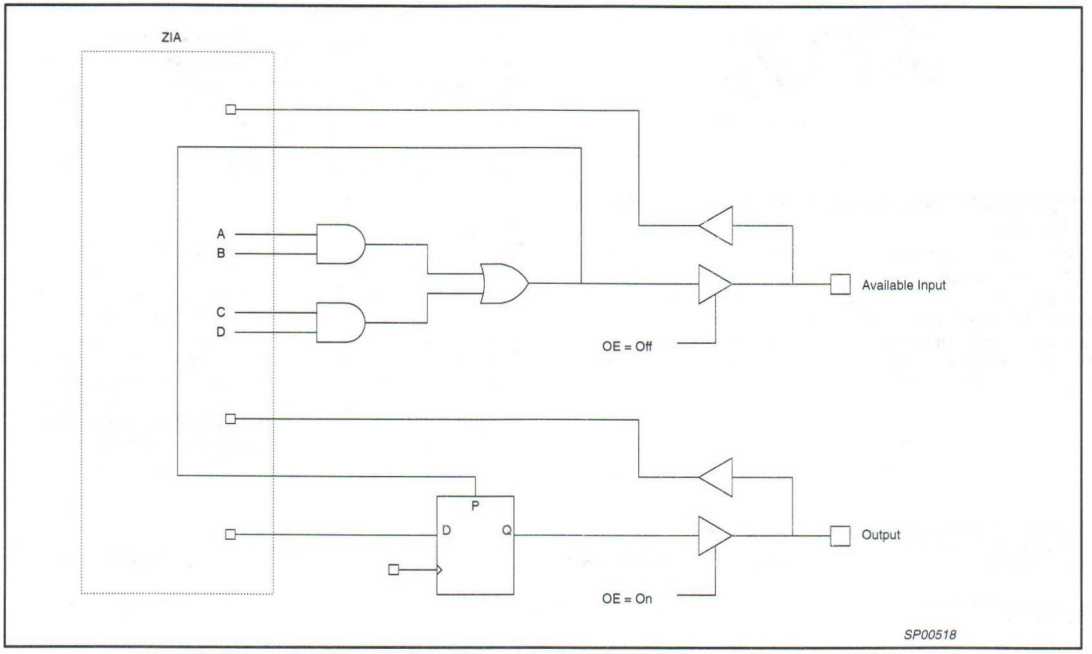


Figure 3 Schematic Representation of Sum of Products Control Term Implemented in Philips CPLD.

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

For more information, contact 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

UNDERSTANDING COOLRUNNER™ CLOCKING OPTIONS

The Coo!Runner™ 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

All CoolRunner devices provide multiple clock sources to each register of the device. These sources support both synchronous and asynchronous clocking. Each type of clock source has well defined capabilities, depending upon whether the clock is generated from a dedicated input, a multi-purpose I/O pin, or a logic block control term. The number of each type of clock varies with device density. Table 1 indicates the number of each type of clock found in various CoolRunner devices.

Dedicated Input Pin Clocks

The first type of clock source (clk0) is associated with a dedicated input pin. As Figure 1 shows, this input is attached directly to a low-skew, dedicated clock network. These sources can generate only synchronous (external) clocks to the associated clock network, and polarity of this clock is selectable at every macrocell. All of the CoolRunner™ devices have one or more of these clock sources. 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. With regard to timing, the Tsu and Tco specs in the datasheet refer to synchronous clocks.

Table 1. Clock resources by device type

Part Number	Dedicated Input Pin Clocks	I/O Pin Clocks	Control Term Clocks	Total Clock Resources
PZx032–/I	clk0	clk1	0	2
PZx032C/N	clk0	clk1	2 x 2 logic blocks = 4	6
PZx064–/I	clk0	clk1, clk2, clk3	0	4
PZ3064A/D	clk0	clk1, clk2, clk3	2 x 4 logic blocks = 8	12
PZ5064C/N	clk0	clk1, clk2, clk3	2 x 4 logic blocks = 8	12
PZx128–/I	clk0	clk1, clk2, clk3	0	4
PZ3128A/D	clk0	clk1, clk2, clk3	2 x 8 logic blocks = 16	20
PZ5128C/N	clk0	clk1, clk2, clk3	2 x 8 logic blocks = 16	20
PZ3960C/N	clk0–clk7	0	2 x 48 logic blocks = 96	104

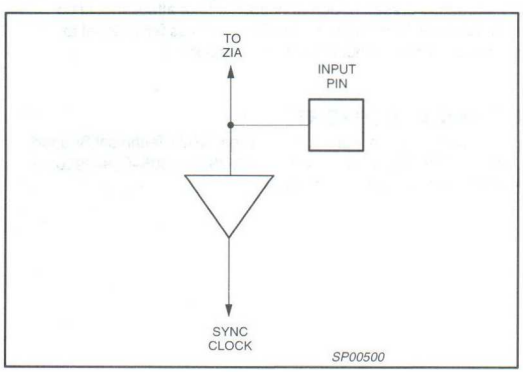


Figure 1.

I/O Pin Clocks

The second type of clock source comes from 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 in all respects. 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.

The I/O pin clocks can also be used to generate asynchronous 'equation-based' clocks. 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}. The timing for asynchronous clocks is different in that the Tco time is extended by the amount of time that it takes for the signal to propagate through the array and reach the clock network, and the Tsu time is reduced. This time is dependent on whether the PAL or PAL+PLA paths are used to generate the equation. For clock equations that use only the PAL path, Tco is extended by Tpdf_pal. Using the PAL+PLA path extends Tco by Tpdf_pla.

Control Term Clocks

The third type of clock source is provided by the control terms in each logic block. These types of clocks available only in the XPLA Enhanced and XPLA2 device families. Figure 4 shows the macrocell architecture for each of these families. In the XPLA Enhanced device family (Figure 4a), there are six control terms in each logic block, and in the XPLA2 device family (Figure 4b), there are eight control terms in each logic block. As shown for the XPLA Enhanced family, two of the six control terms are shared by the output enable multiplexer and the clock source multiplexer. These control terms can be used as either an output enable, a clock, or both. The XPLA2 family has two extra control terms that are dedicated to only the clock source multiplexer. These clocks can be individually configured as any PRODUCT term or SUM term equation created from the 36 signals available inside the logic block. If a SUM-OF-PRODUCTS equation is required, it must be implemented in a macrocell and then fed-back into a control term through the ZIA (see the application note Using Sum of Products Control Terms for more information).

Each control term clock is available to all the macrocells within a logic block, but it must be duplicated on another control term if the same clock is used in different logic blocks. These clocks are not attached to a low-skew clock network, and they must pass through the interconnect array and a single product term before reaching the flip-flop. Therefore, Tco time is extended by the amount of time that it takes for the signal to propagate through the appropriate array, and the Tsu time is reduced. Unlike the other two types of clock sources, control term clocks are not associated with specific pins and may be assigned to any I/O or dedicated input.

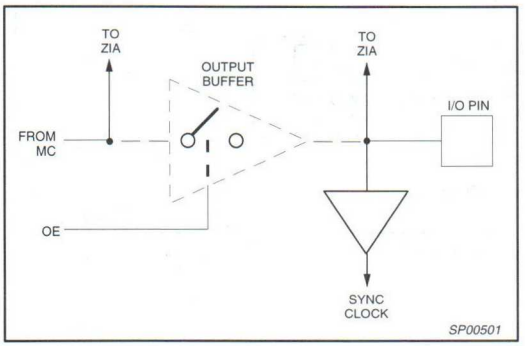


Figure 2.

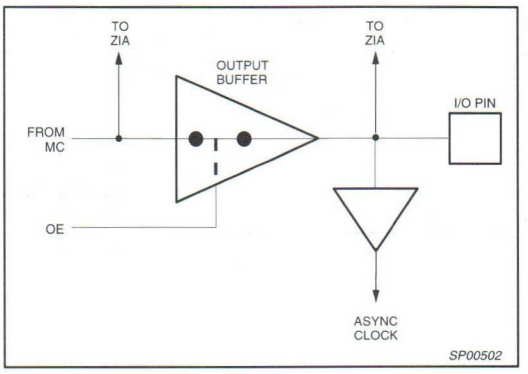


Figure 3.

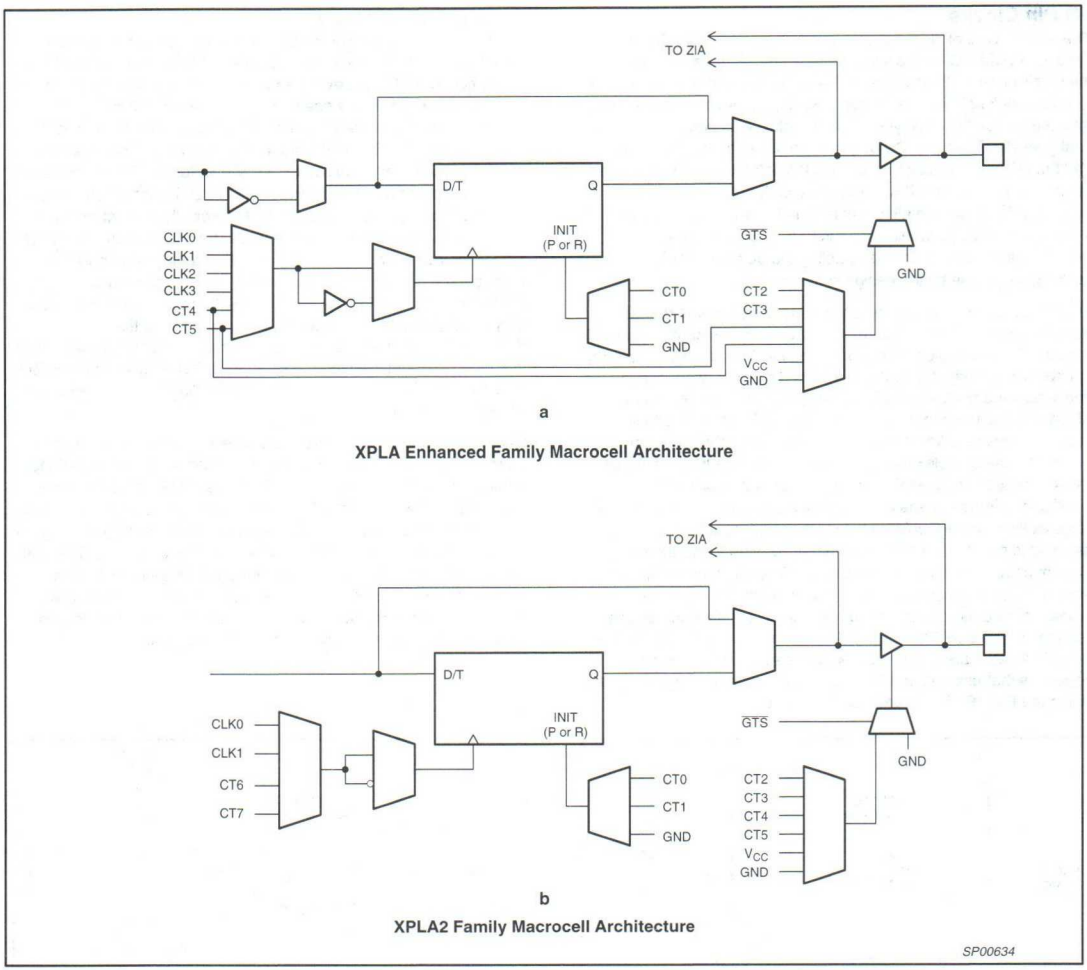


Figure 4. Macrocell Architectures

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;
```

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. The software will assign asynchronous clocks (product term or sum term) first to the control term clocks, until all available control term clocks are used. Then the design software will assign asynchronous clocks to a macrocell, enable the output buffer, and feed the clock equation to the register through the dedicated clock network as in Figure 3.

```
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;
```

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. It is important to note that the 'clock' width **must** be longer than Tpd!

```
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
```

```
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 istyp e'com,keep,retain';

equations

IL = (D & !CLK) # (IL & CLK) # (D & IL);
Q = (IL & CLK) # (Q & !CLK) # (IL & Q);

end
```

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 1 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.

Table 1. 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

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 toll free 1–888–COOLPLD or 1–505–858–2996.

APPENDIX A: HIERARCHY DESIGN – TWO-BIT ADDER

File: adder2.phd – Upper-Level Design File (Two-bit Adder)

```
Module          adder2;
Title           'Two-bit Adder'
Declarations
  a0, a1, b0, b1, cin, cout, s0, 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_0.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.cout;
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          = cin & !a & !b
                 # !cin & !a & b
                 # cin & a & b
                 # !cin & a & !b;
  cout         = cin & b
                 # cin & a
                 # a & b;
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_dec4npin 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      - fld_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_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 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
```

Terminating unused CoolRunner™ I/O pins AN068



The CoolRunner™ family of CPLDs are the first PLDs to employ a TotalCMOS™ design methodology. Because these devices are fabricated on CMOS process technology, it is important to consider the options available in terminating unused pins. Allowing unused inputs and I/O pins to float can cause the voltage on the pin to be in the linear region of the CMOS input structures, which can increase the power consumption of the device.

All unused dedicated inputs and JTAG/ISP function pins (when JTAG/ISP is used) on CoolRunner devices must be terminated. For unused I/O pins, some CoolRunner devices have on-chip, programmable, weak pull-down resistors that can be used for termination, but other devices require termination by the user. The table below indicates whether or not a particular CoolRunner device is equipped with the on-chip weak pull-downs.

Termination Options

For devices that do not have the on-chip weak pull-downs, Philips recommends using external 10 kΩ pull-up resistors on all inputs or I/O's that are not used. This provides the flexibility to use 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. It is also acceptable to terminate the pins inside by connecting them to V_{DD} or ground. This must be done in the design entry phase through either schematic or HDL.

When using the JTAG/ISP functions, 10 kΩ pull-up resistors should be used on each of the four mandatory signals. Letting these signals float can cause the voltage on TMS to come close to a logic low, which could cause the device to enter JTAG/ISP mode at unspecified times.

In CoolRunner CPLDs that have the on-chip weak pull-downs, the software fitter automatically activates them for all unused I/O pins. No current is consumed by the device when the internal pull-down is enabled. 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 L_{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.

Disabling the weak pull-downs

In certain cases, a design may require that unused I/O pins be left tri-stated, instead of being connected to the on-chip weak pull-down resistor. This is accomplished in software via 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';

In this case, the weak pull-down on pin 12, to which signal dingo is assigned, is disabled. The symbol name 'dingo' need not be declared separately.

Table 1.

Part Number	External termination recommended	On-chip weak pull-down resistors
XPLA Family		
PxZ22V10–/I	X	
PZx032–/I	X	
PZx064–/I		X
PZx128–/I		X
XPLA Enhanced Family		
PZx032C/N	X	
PZ3064A/D		X
PZ5064C/N		X
PZ3128A/D		X
PZ5128C/N		X
XPLA2 Family		
PZ3320C/N		X
PZ3960C/N		X

ISP design considerations for CoolRunner™ CPLDs AN069

Author: B. Wade Baker, Senior CPLD Specialist

ISP DESIGN CONSIDERATIONS FOR COOLRUNNER™ CPLDS

With the introduction of the Coolrunner PZ5128 and PZ3128 CPLDs, Philips Semiconductors entered the realm of ISP, (In System Programming). Now with 32, 64, and 960 macrocell devices, Philips covers the spectrum of CPLDs incorporating ISP. 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 10^{15} Ω 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 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 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 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.

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 Ω 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 V_{CC} for the ISP CPLD 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 $0.5V_{OUT}$. This is not a problem in this application, however, since the signal connections are at the end of the line.

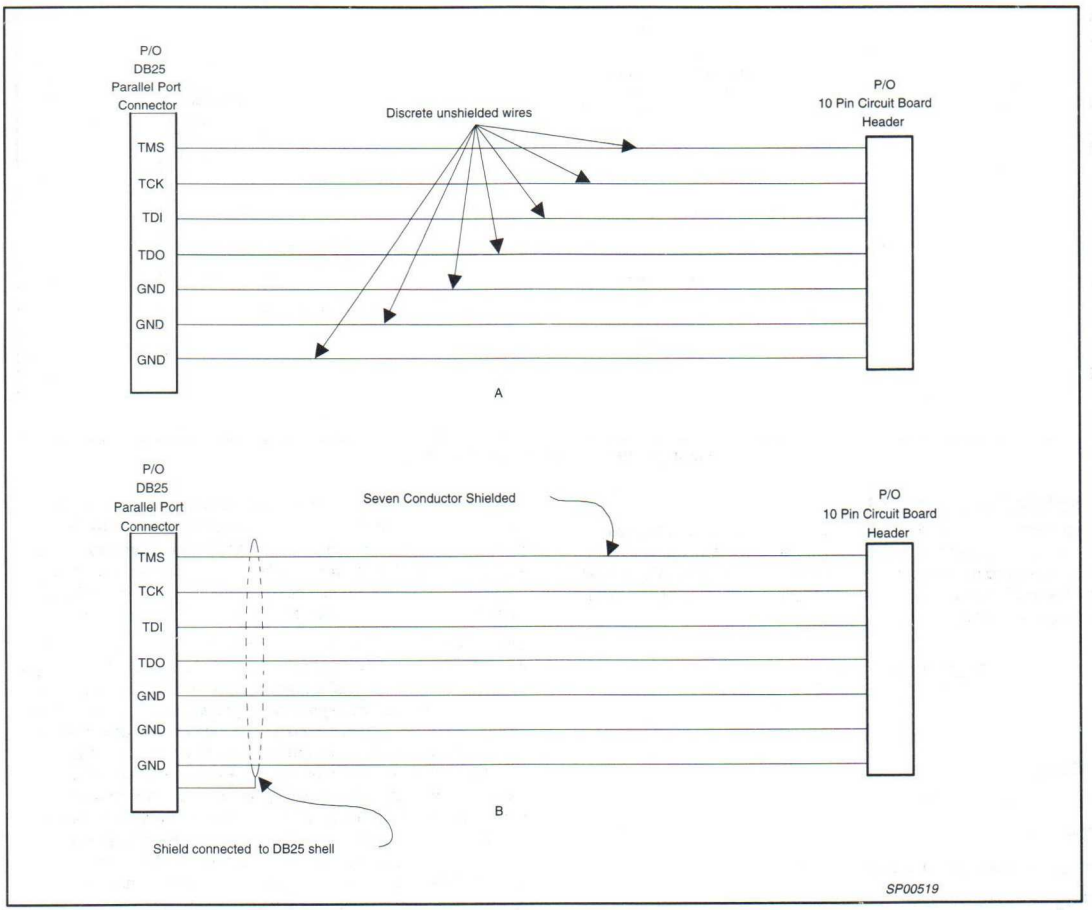


Figure 1. 'Bad' download cable design

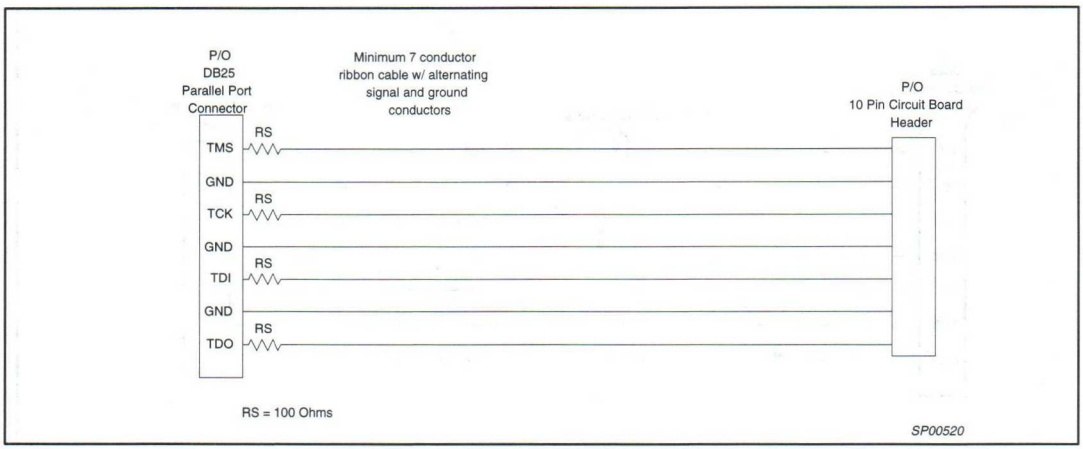


Figure 2. 'Good' download cable design

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:

$$L = \frac{t_r}{2t_{pd}} \times \frac{1}{\sqrt{1 + \frac{C_D Z_O}{t_r}}}$$

Where:

- L = trace length
- t_r = rise time
- t_{pd} = calculated propagation delay
- C_D = trace capacitance
- Z_O = 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

t_r in nanoseconds	C_D 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 V_{CC}. 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. As an example, 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, 10 µF 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

Again using the PZx128 as an example, we find that it 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 Ω, the current required will be 96 x (5V/50 Ω) = 9.6 amperes! This can safely be considered WORST CASE! With a 100 mV drop in V_{CC} and a 2 ns rise time, the capacitance required will be:

$$\frac{9.6\ A \times 2\ ns}{100\ mv} = 0.192\ \mu F$$

A 0.1 µF capacitor for every power/ground pair will therefore handle the maximum instantaneous current requirements quite easily.

FILTERING THE POWER SUPPLY

At high frequencies the 0.1 µF 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.

$$F = \frac{1}{\pi \times T_X}$$

Substitute:

- The period of the clock signal for T_X when calculating the fundamental frequency
- The pulse width of the clock signal for T_X when calculating the 1st harmonic
- The rise time of the clock signal for T_X when calculating the 2nd harmonic

For a system operating at 100MHz with a pulse width of 5 ns, the F_0 is 32MHz, F_1 is 64MHz, and F_2 is 159 MHz.

Using the formula for resonance,

$$f = \frac{1}{2\pi\sqrt{LC}}$$

and plugging in the lead inductance of 1.1nH for a 0.001 µF 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 0.1 µF 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 CoolRunner™ CPLD 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 devices 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.

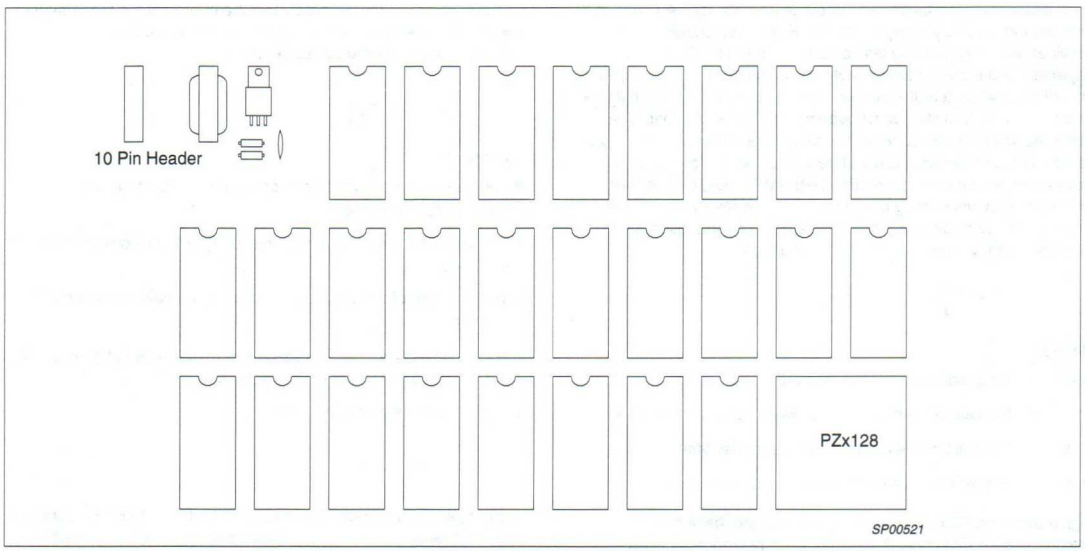


Figure 3. 'Bad' PCB layout

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.

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.

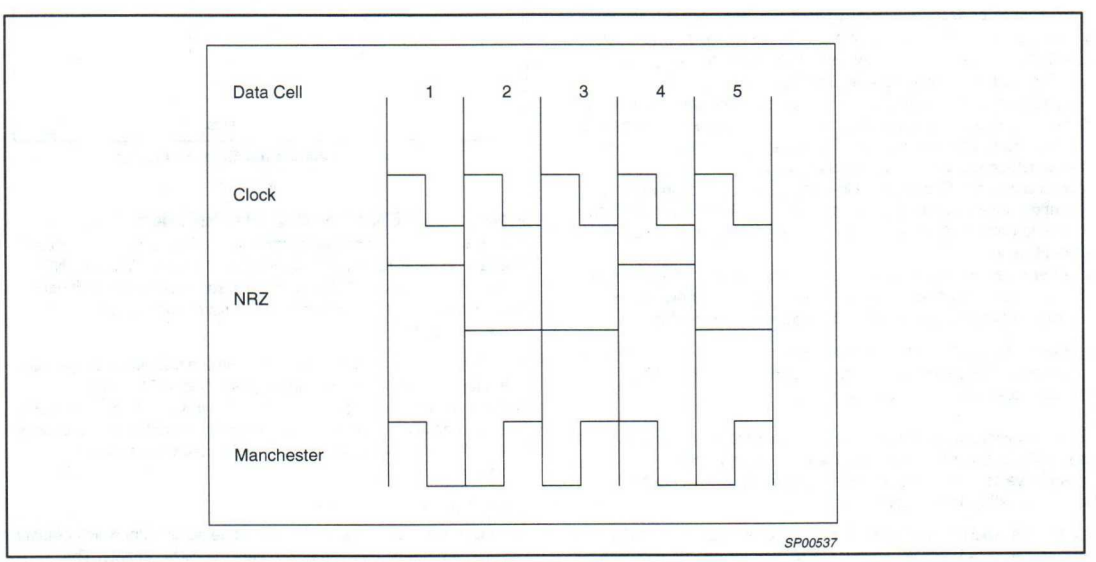


Figure 1. NRZ and Manchester Code Defined

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.

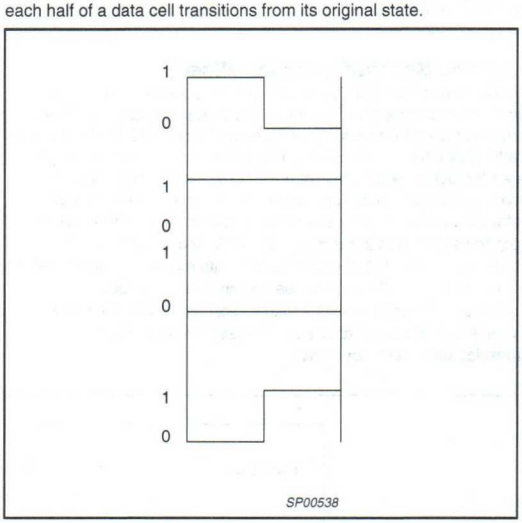


Figure 2. Manchester Error Detection

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 :

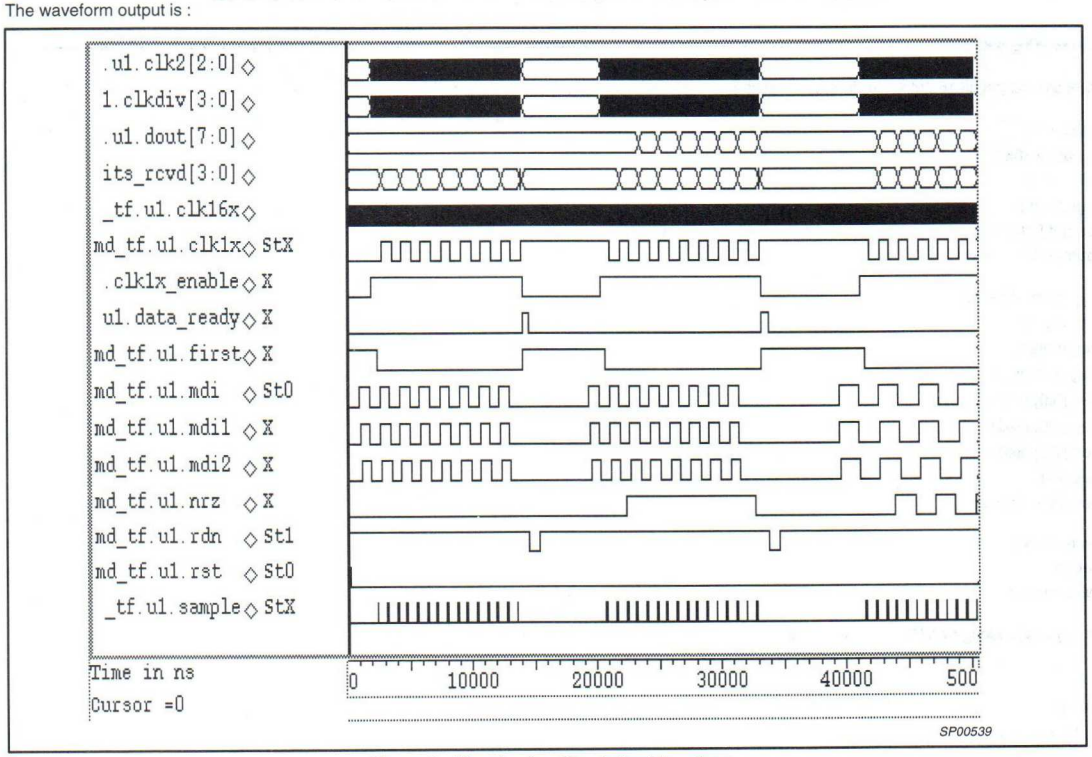


Figure 3. Manchester Simulation Waveform

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.

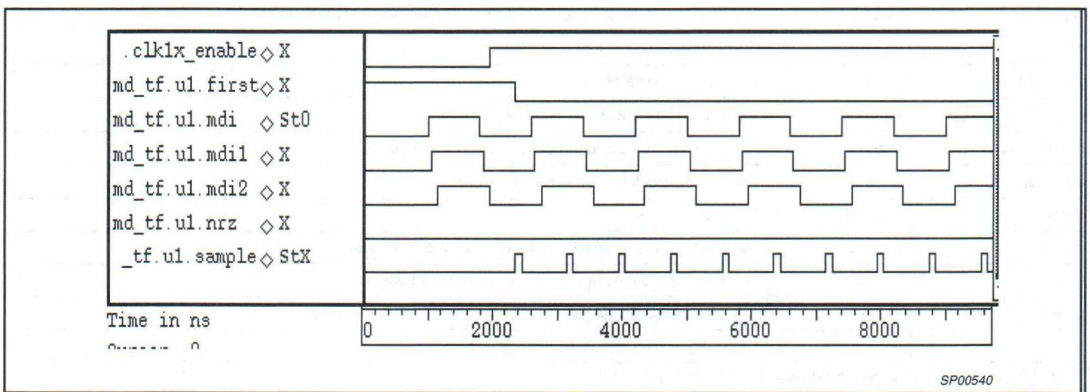


Figure 4. Waveform Showing Timing of Center Sampling in the Manchester Decoder

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.

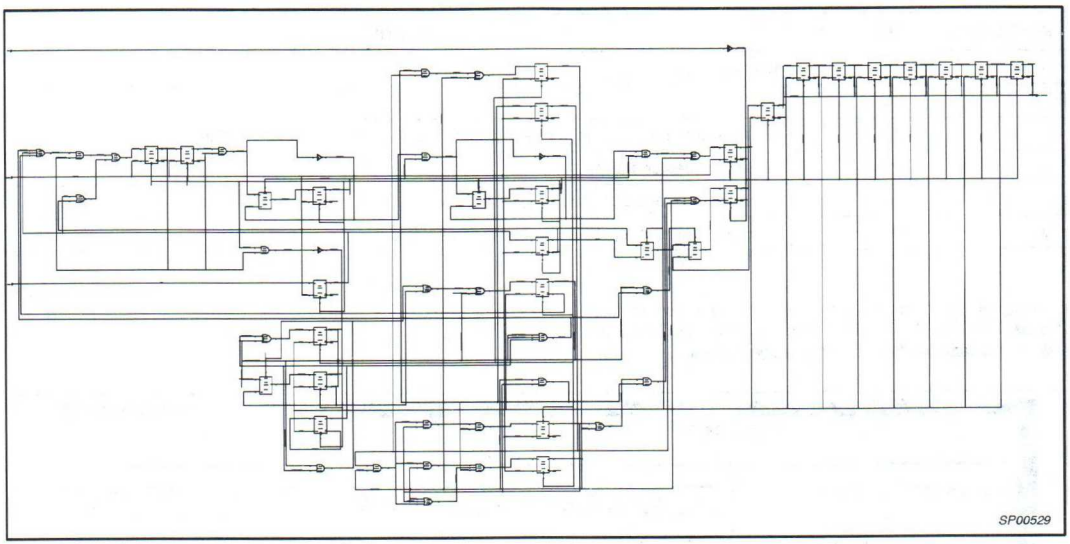


Figure 5. Manchester Decoder Schematic

Table 2. Manchester Encoder Pinout Functionality
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.

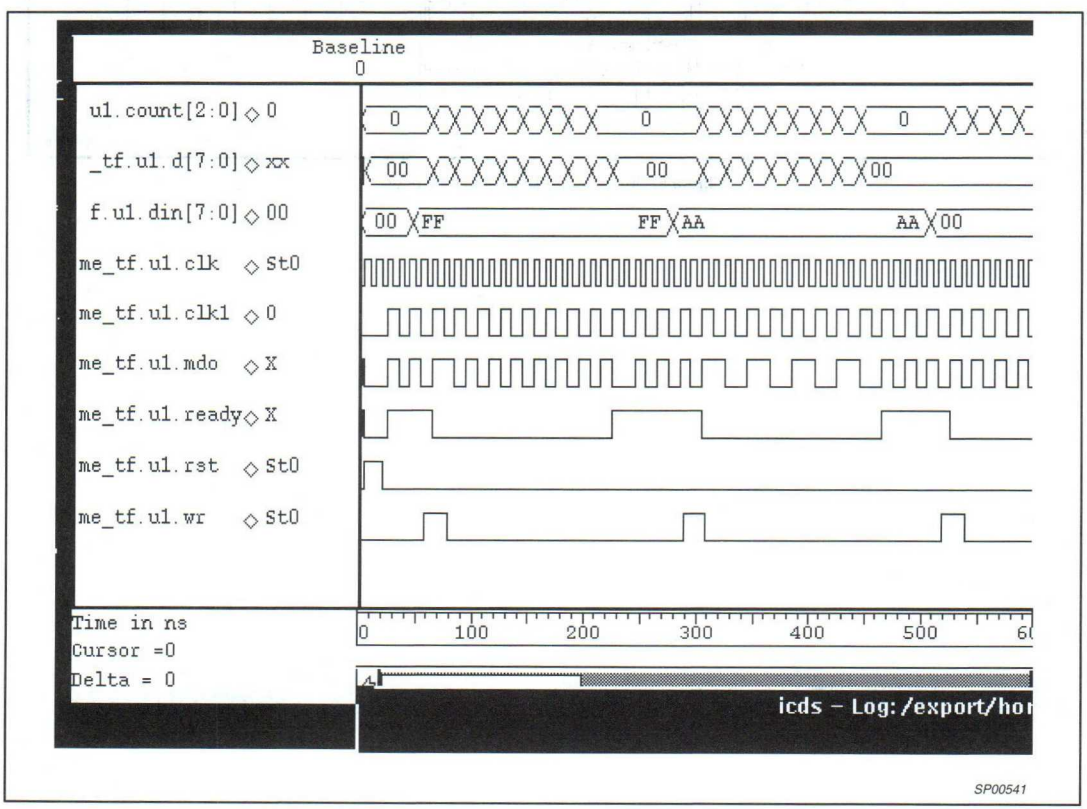


Figure 6. Waveform Results of Manchester Encoder Simulation

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.

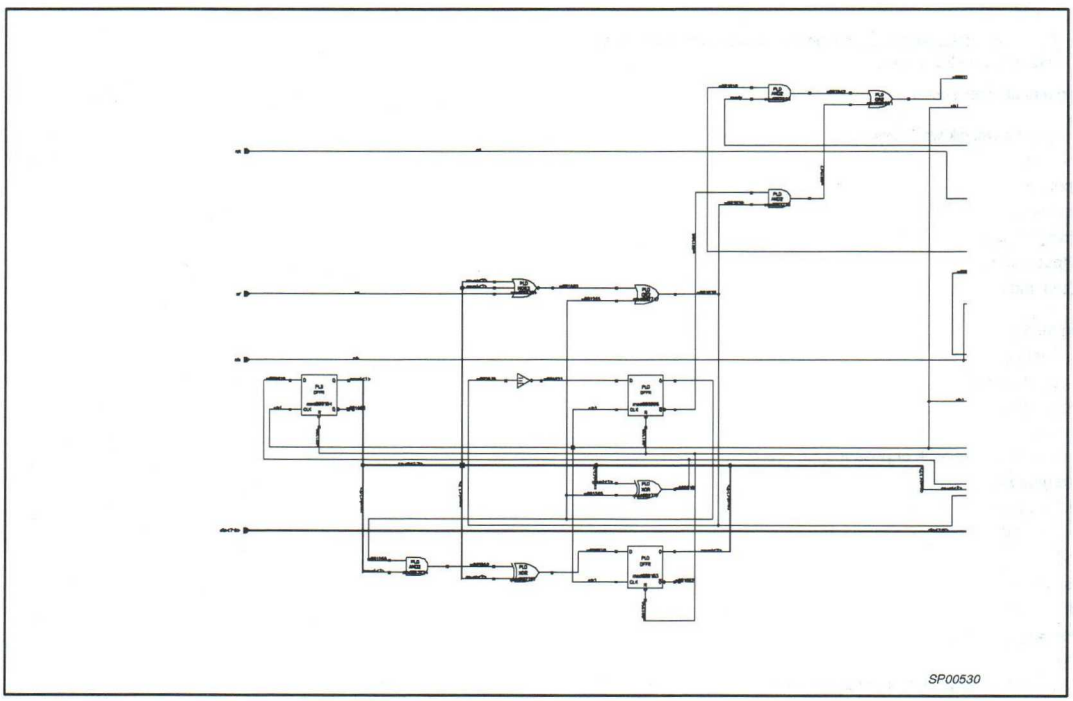


Figure 7. Manchester Encoder

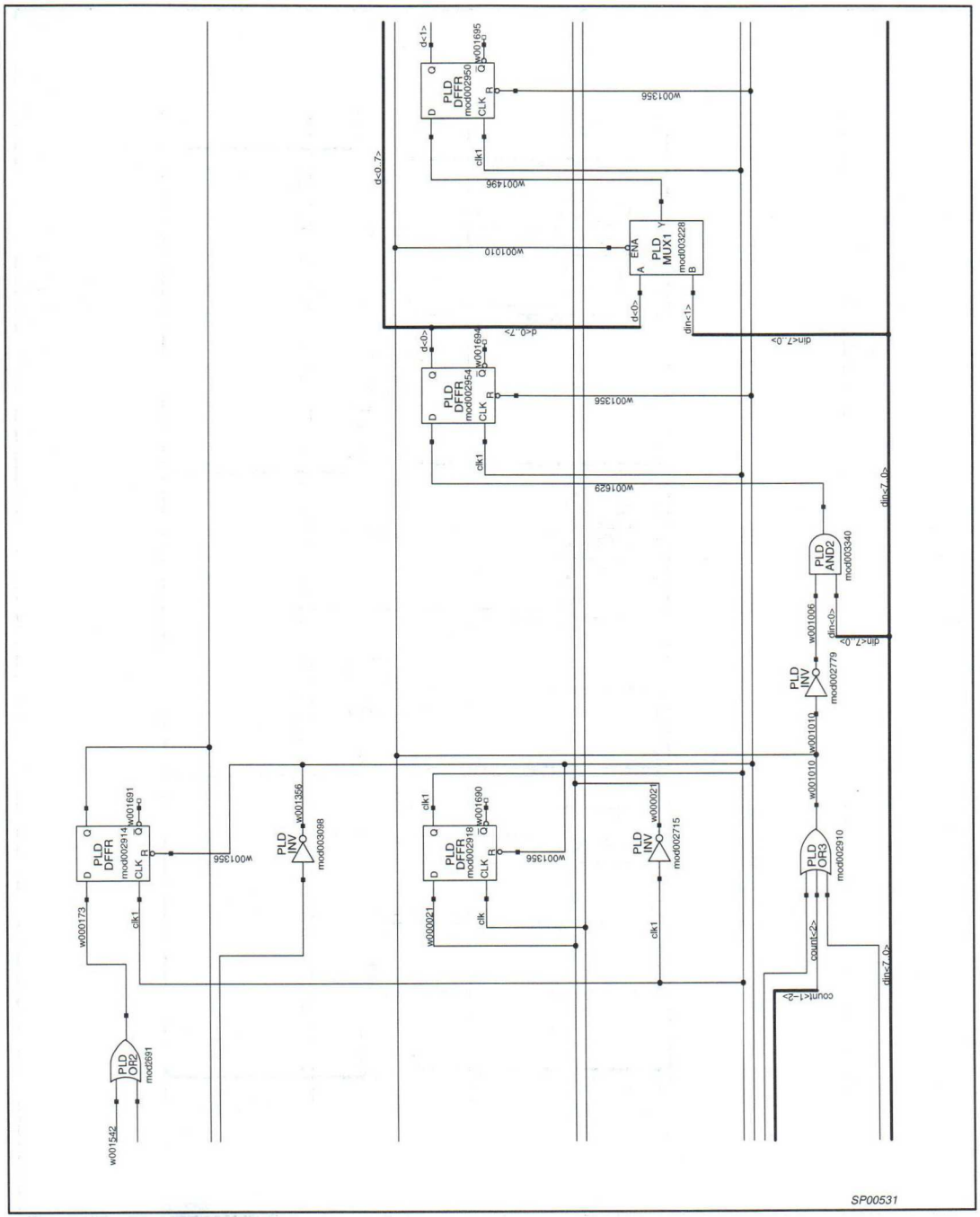


Figure 7A. Manchester Encoder Schematic (Cont.)

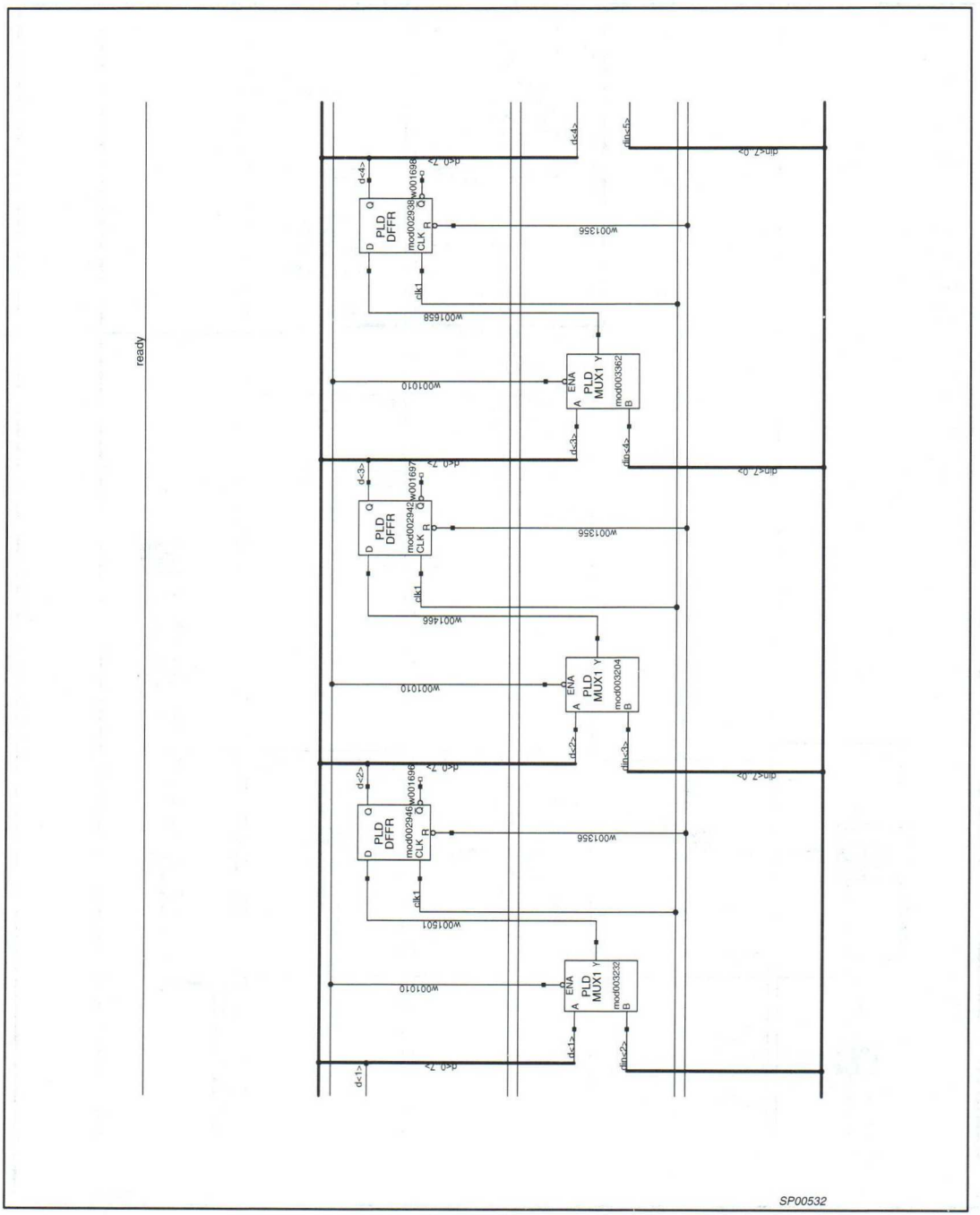


Figure 7B. Manchester Encoder Schematic (Cont.)

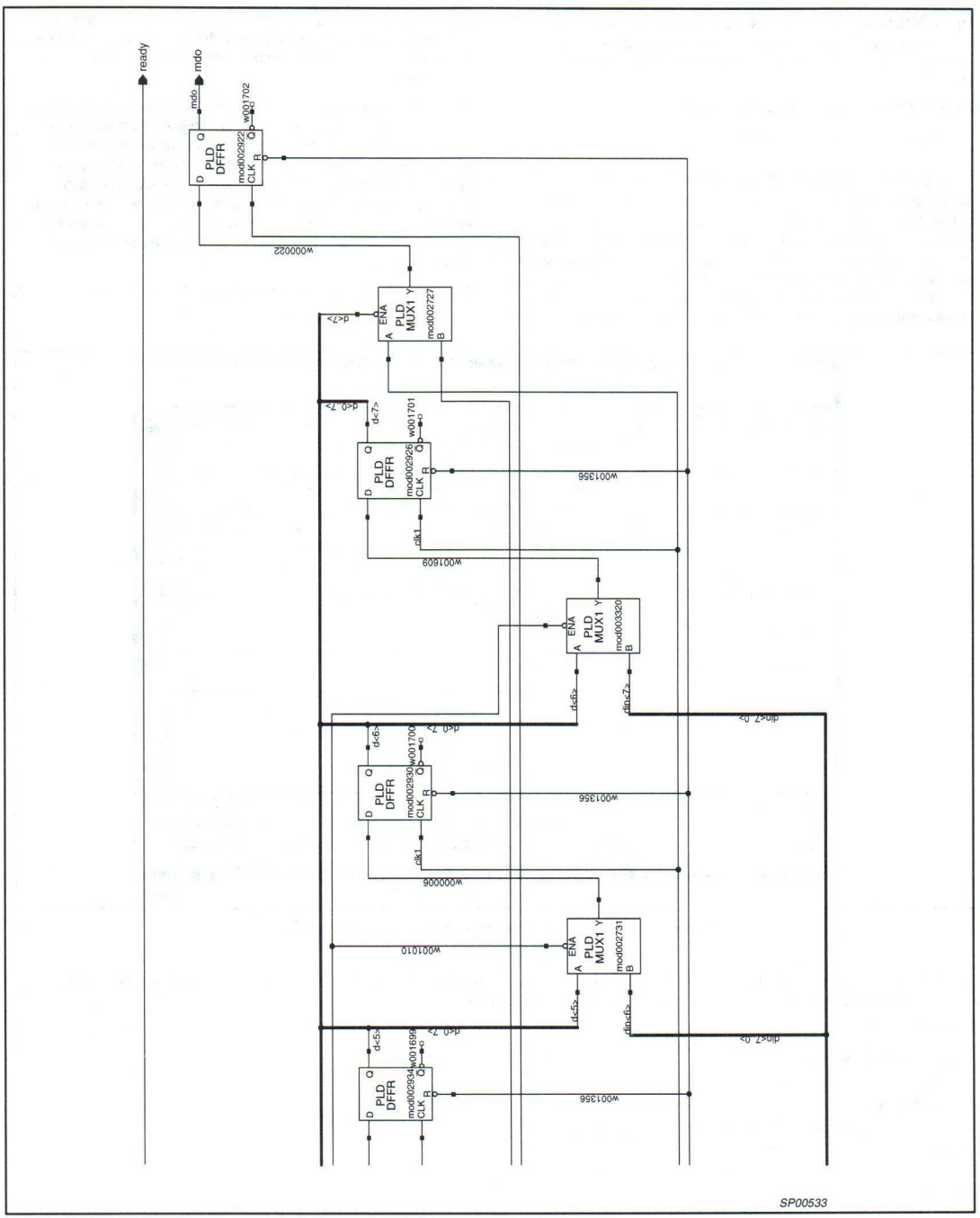


Figure 7C. Manchester Encoder Schematic (Cont.)

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. Tnis 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.

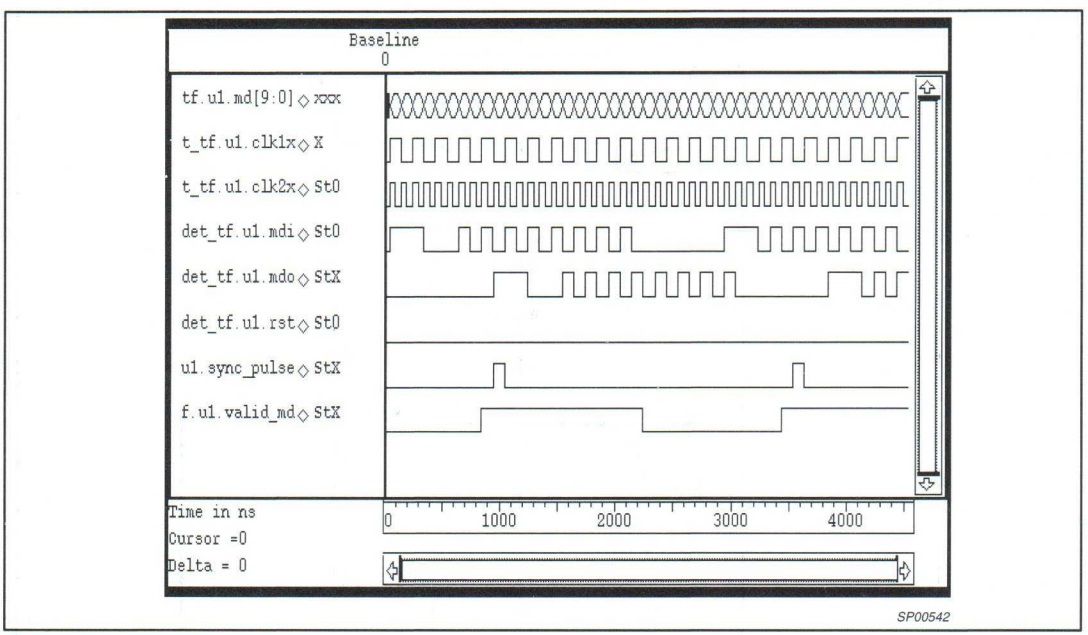


Figure 8. Sync and Manchester DetectorWaveform Results

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.

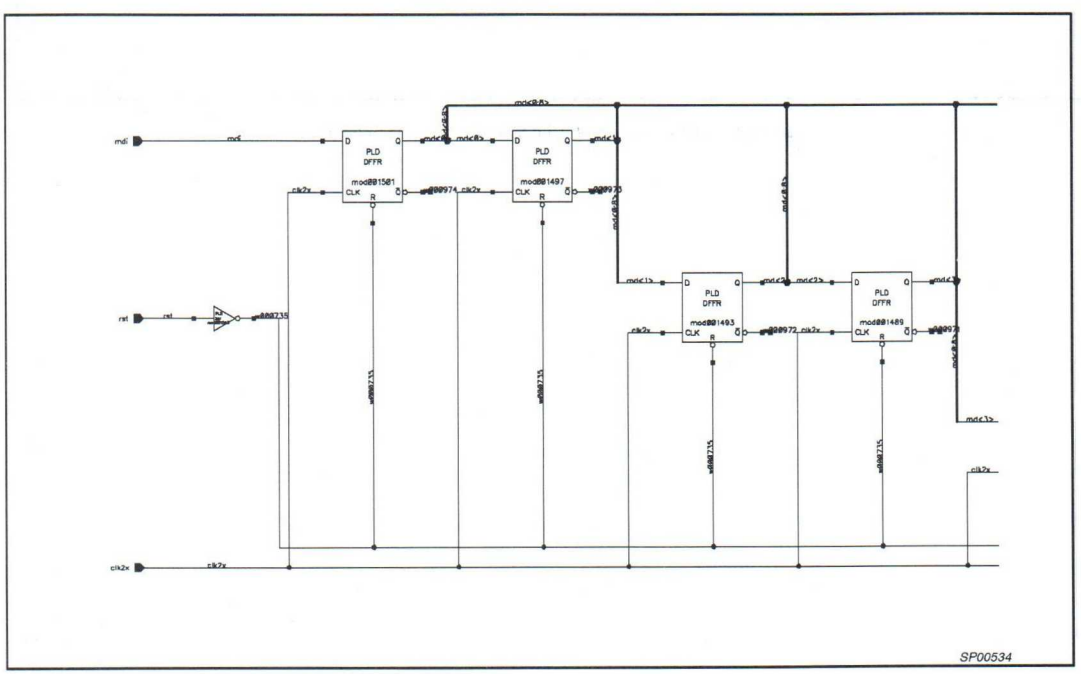


Figure 9. LHS of Sync and Valid Manchester Schematic

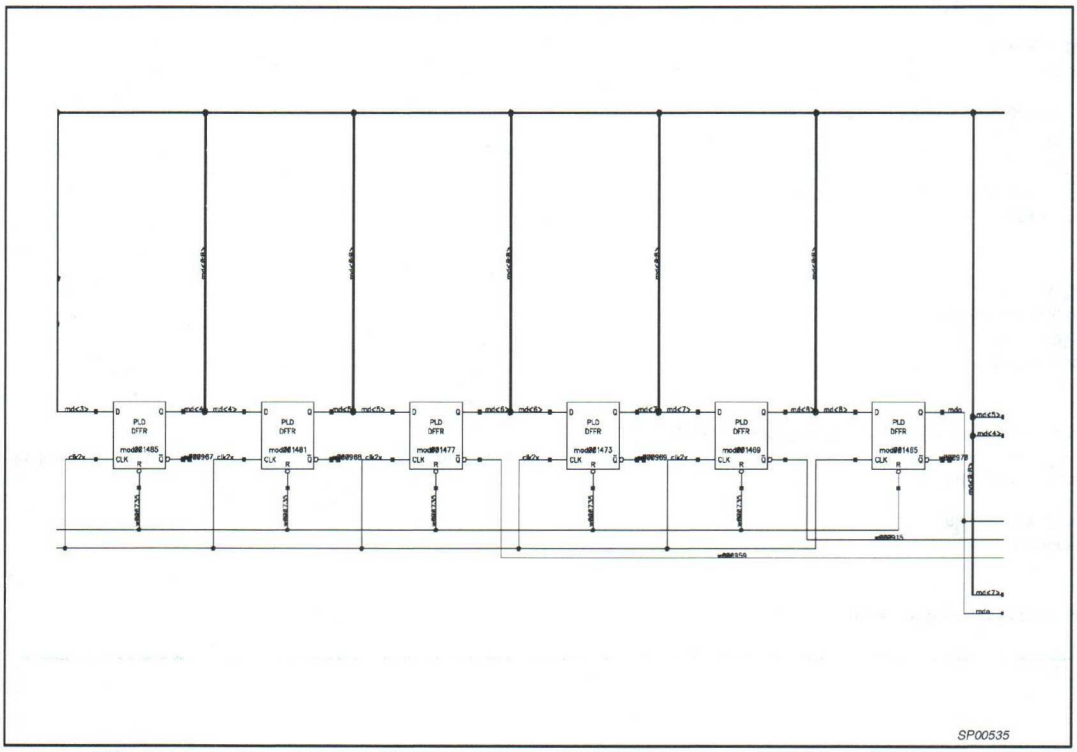


Figure 9A. LHS of Sync and Valid Manchester Schematic (Cont.)

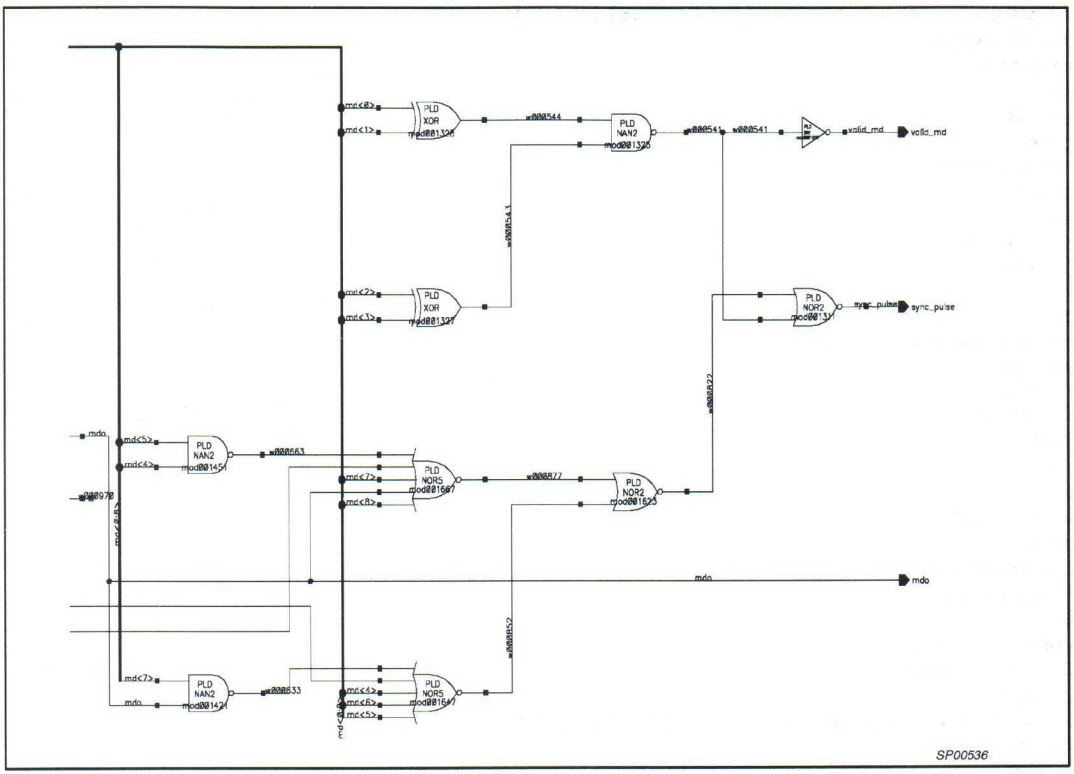


Figure 9B. LHS of Sync and Valid Manchester Schematic (Cont.)

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.

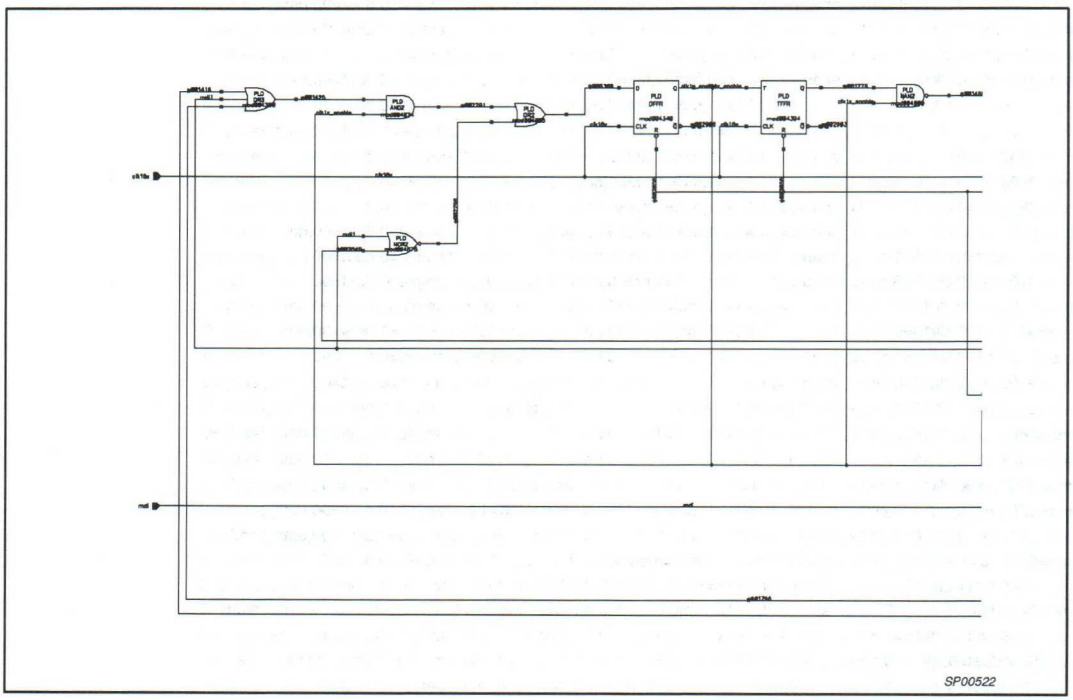


Figure 10. Manchester Decoder Schematic

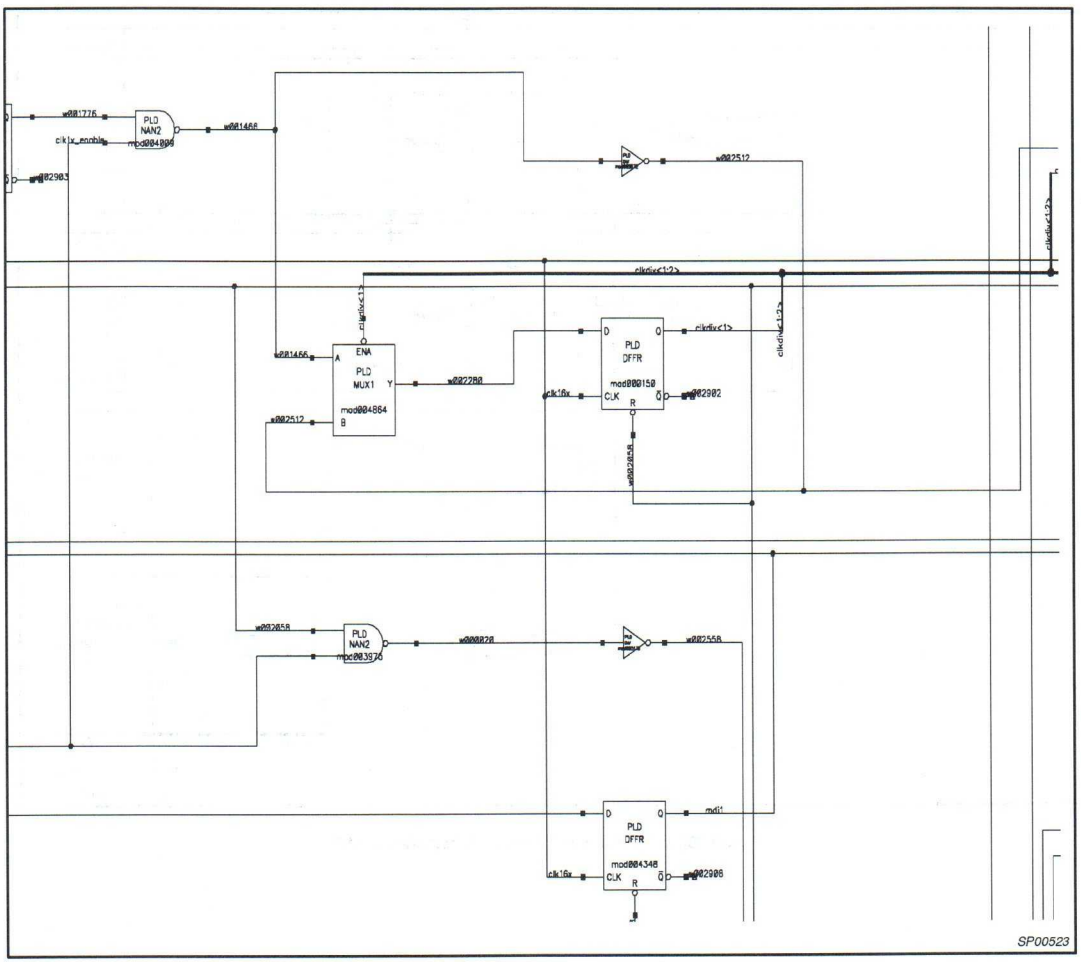


Figure 10A. Manchester Decoder Schematic (Cont.)

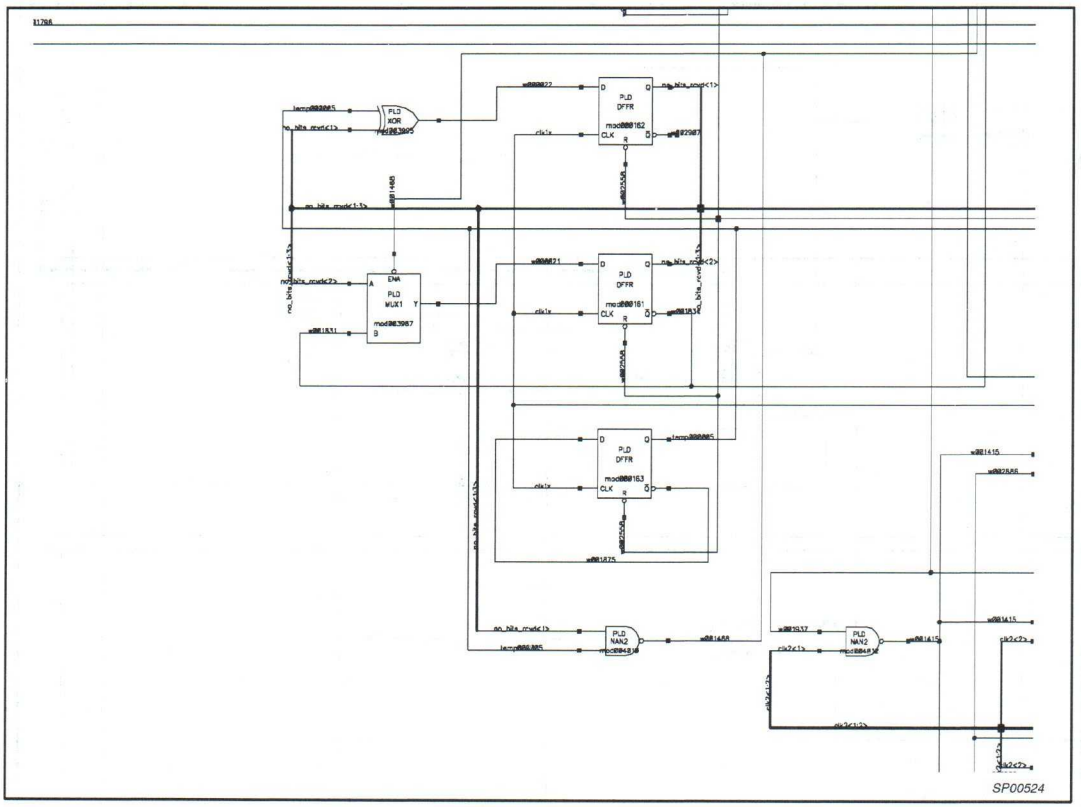


Figure 10B. Manchester Decoder Schematic (Cont.)

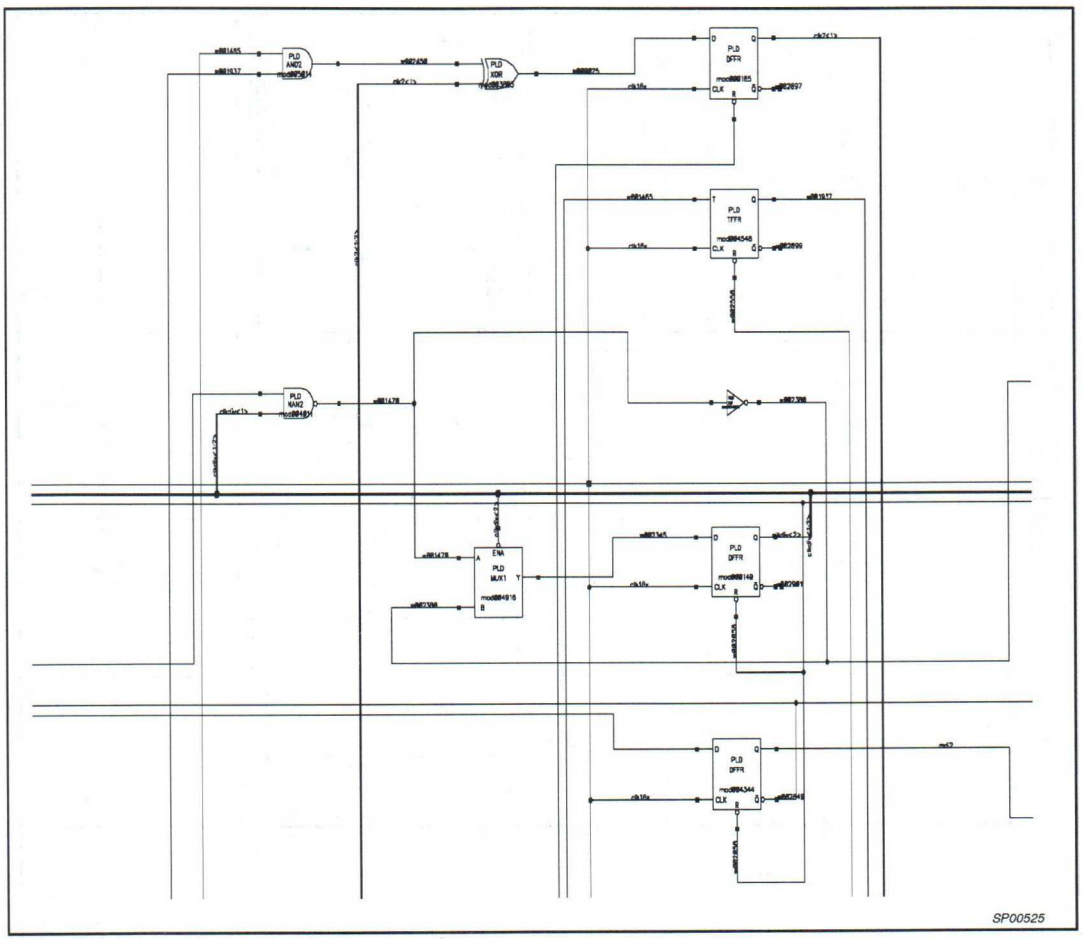


Figure 10C. Manchester Decoder Schematic (Cont.)

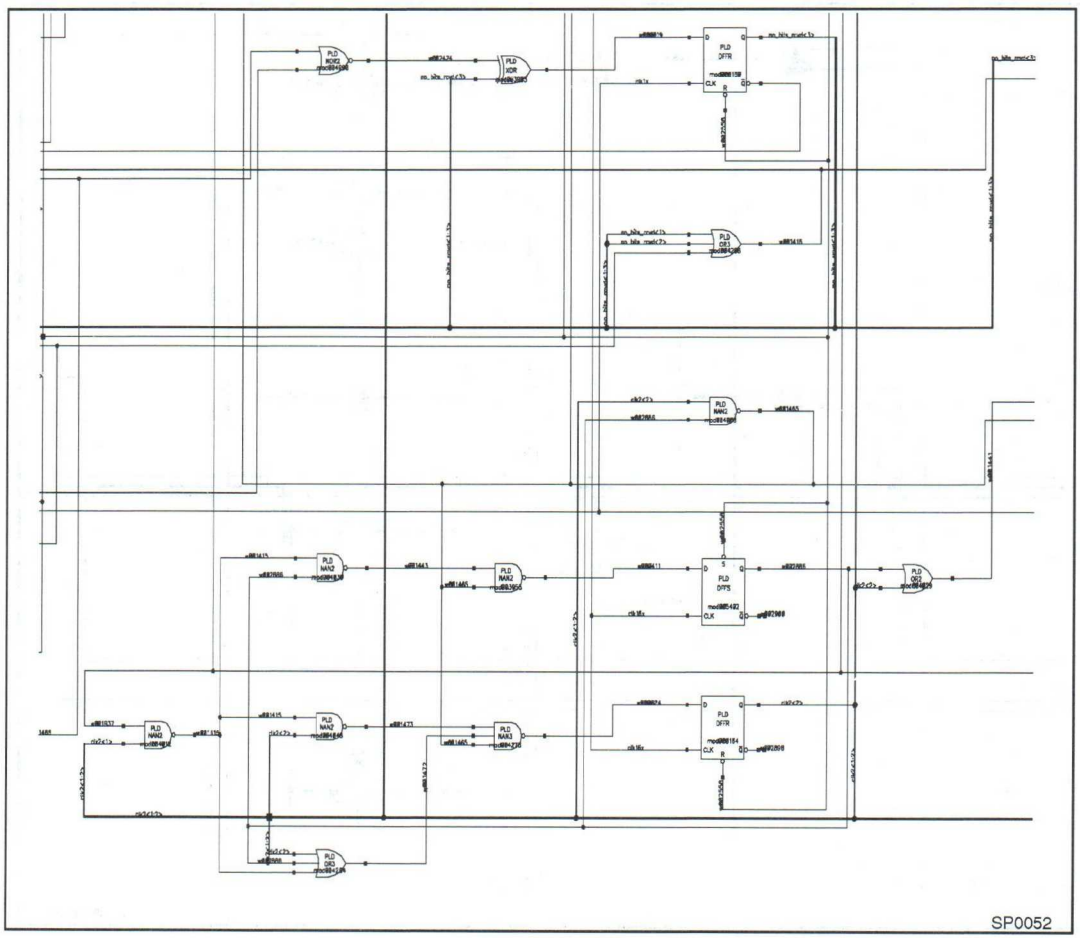


Figure 10D. Manchester Decoder Schematic (Cont.)

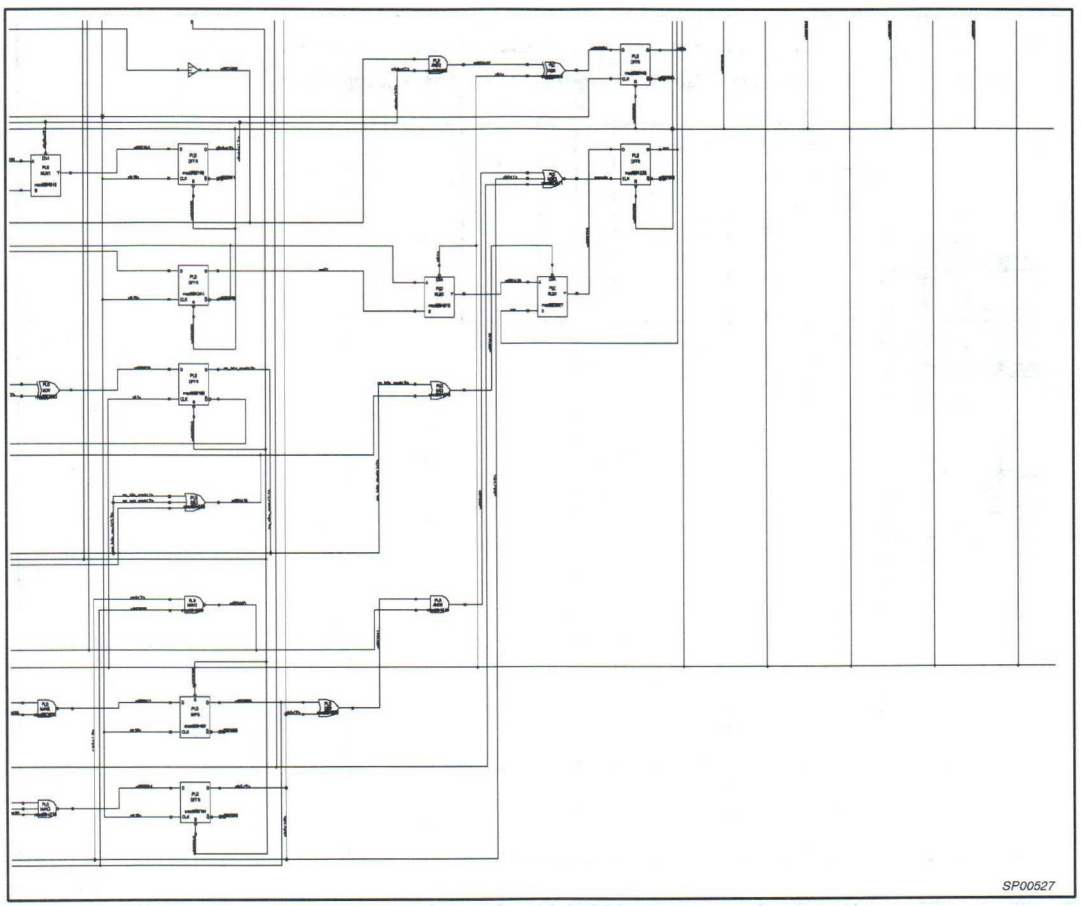


Figure 10E. Manchester Decoder Schematic (Cont.)

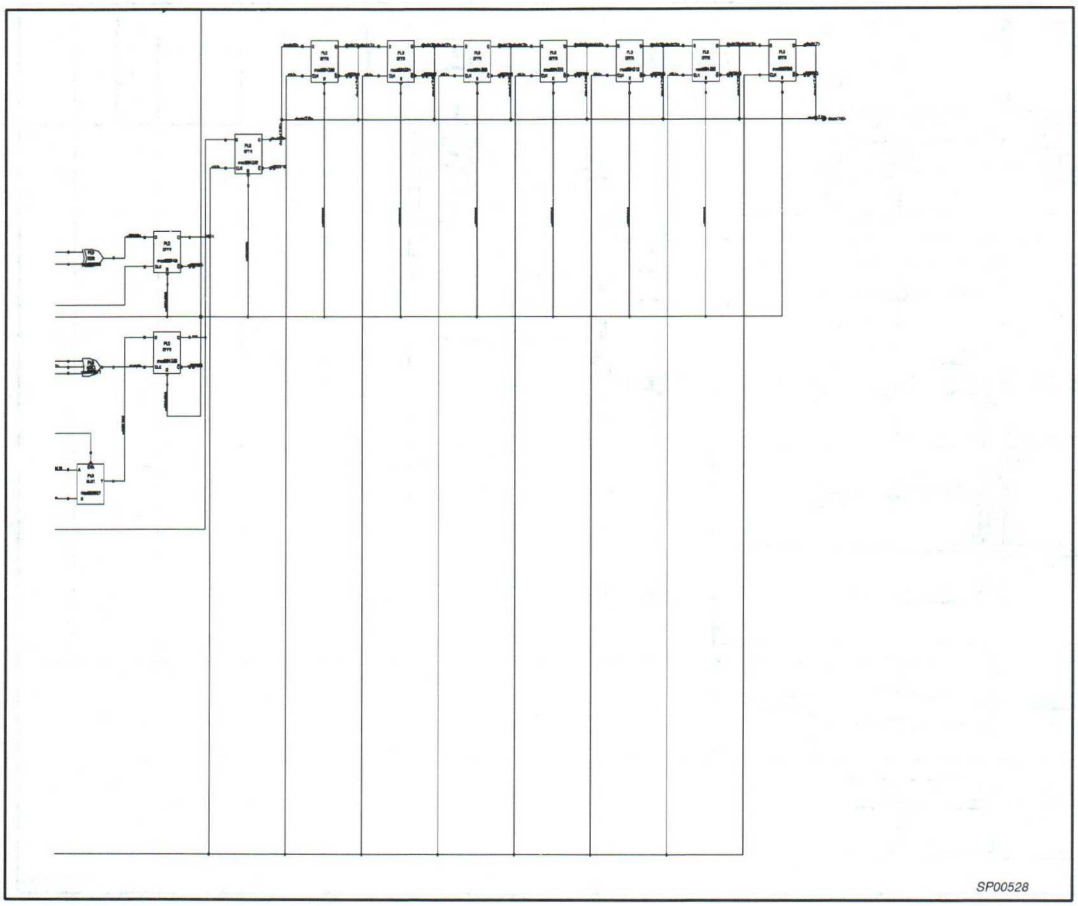


Figure 10F. Manchester Decoder Schematic (Cont.)

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 ;
#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 Design Flow for Philips CPLDs

AN071

INTRODUCTION

This note provides the steps for using OrCAD[1] Express and Philips Semiconductors' XPLA Designer 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 with the Altera 7000 Series CPLDs for devices up to 128 macrocells. Philips also produces CPLDs as large as 960 macrocells. An example design is generated using schematic,VHDL synthesis, and simulation tools from OrCAD Express, and compiled to a jedec file.

Two VHDL source files are imported and a mixed schematic/VHDL design entry is used. This illustrates one approach of many for targeting Philips CPLDs using OrCAD Express. Please see the OrCAD and Philips documentation to learn more about the capability of these tools. OrCAD Express is also target Philips CPLDs using Minc's PL-Designer.

Technical support for the design flow described in this application note is provided by:

Philips Technical Assistance

Telephone no. 888-coolpld

web site - http://www.coolpld.com

ftp site - www.coolpld.com

Fax on Demand - 800 282-2000

OrCAD Technical Assistance

techsupport@orcad.com - 503 671 9400

REFERENCES

OrCAD Express for Windows User's Guide

XPLA Designer User's Guide

Philips Complex Programmable Logic Devices Data Handbook IC27

INSTALLATION REQUIREMENTS

This design requires the following PC-based CAE tools:

OrCAD Express v 7.1

XPLA Designer v 2.55

The Philips CoolRunner series can be targeted using OrCad Capture in schematic only designs. Philips software supports Philips Hardware Description Language (PHDL), allowing Capture

(1) Philips acknowledges the trademarks of OrCAD, Inc mentioned in this document.

users to mix schematics and a HDL. This design targets the Philips PZ3064 CPLD using OrCAD Express' VHDL synthesis.

The symbols in the ps.olb library are:

AND2 - AND12	AND2B1	AND3B2	AND3B1	AND3B2
AND3B3	AND4B1	AND4B2	AND4B3	AND4B4
AND5B1	AND5B2	AND5B3	AND5B4	AND5B5
OR2 - OR12	OR2B1	OR2B2	OR3B1	OR3B2
OR3B3	OR4B1	OR4B2	OR4B3	OR4B4
OR5B1	OR5B2	OR5B3	OR5B4	OR5B5
ND2 - ND12	ND2B1	ND2B2	ND3B1	ND3B2
ND3B3	ND4B1	ND4B2	ND4B3	ND4B4
ND5B1	ND5B2	ND5B3	ND5B4	ND5B5
NR2 - NR12	NR2B1	NR2B2	NR3B1	NR3B2
NR3B3	NR4B1	NR4B2	NR4B3	NR4B4
NR5B1	NR5B2	NR5B3	NR5B4	NR5B5
XR2 -XR8	XNR2 - XNR8	DFF	DFFR	DFFS
DFFE	DFFRE	DFFSE	TFF	TFFR
TFFS	TFFE	TFFRE	TFFSE	JKFF
JKFFR	JKFFS	SRFF		

The symbols in the ps_ttl.olb library are:

PS7400	Quad 2-input NAND
PS7402	Quad 2-input NOR
PS7403	Quad 2-input NAND
PS7404	Hex Inverter
PS7408	Quad 2-input AND
PS7410	Triple 3-input NAND
PS7411	Triple 3-input AND
PS7414	Hex Inverting schmitt trigger
PS74138	3 to 8 line decoder/demultiplexer; inverting
PS74139	Dual 2 to 4 line decoder/demultiplexer
PS74151	8-input multiplexer
PS74153	Dual 4-input multiplexer

PS74154	4 to 16 line decoder/demultiplexer
PS74157	Quad 2-input data selector/multiplexer, non-inverting
PS74161	Presettable 4-bit binary counter, asynchronous reset
PS74162	Presettable synchronous BCD decade counter, synchrouous reset
PS74163	Presettable 4-bit binary counter, synchronous reset
PS74164	8-bit serial in, parallel out shift register
PS74166	8-bit parallel in, serial out shift register
PS74174	Hex D-type flip-flop with reset, positive edge trigger
PS74181	4-bit arithmetic logic unit
PS74190	Presettable synchrouous BCD decade up/down counter
PS74191	Presettable synchronous 4-bit binary up/down counter
PS7420	Dual 4-input NAND
PS7421	Dual 4-input AND
PS74244	Octal Line Driver, 3-state, output enable active low
PS74245	Octal Bus transceiver, 3-state
PS74251	8-input multiplexer, 3-state
PS74253	Dual 4-input multiplexer, 3-state
PS74257	Quad 3-input multiplexer, 3-state
PS74266	Quad 2-inupt EXCLUSIVE NOR
PS7427	Triple 3-input NOR
PS74280	9-bit odd/even parity generator/checker
PS74283	4-bit binary full adder with fast carry
PS74299	8-bit universal shift register, 3-state
PS7430	8-input NAND gate
PS7432	Quad 2-input OR
PS74373	Octal D type transparent latch, 3-state
PS744040	12-stage binary ripple counter
PS744511	BCD to 7 segment latch/decoder/driver
PS74573	Octal D type transparent latch, 3-state, bus oriented pinout
PS74574	Octal D type flip flop, positive edge rigger, 3-state, bus oriented pinout
PS74583	4-bit BCD full adder with fast carry
PS74594	8-bit shift register with output register
PS74595	8-bit serial-in/serial or parallel out shift register with output latches, 3-state
PS74597	8-bit shift register with input flip flops
PS74640	Octal Bus transceiver, 3-state, inverting

PS74688	8-bit magnitude comparator
PS744080	16-bit even/odd parity generator/checker
PS7474	Dual D type flip-flop with set and reset, positive edge trigger
PS7483	4-bit full adder
PS7485	4-bit magnitude comparator
PS7486	Quad 2-input EXCLUSIVE OR

The example uses a reference design of a manchester encoder-decoder (med) 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 encoder-decoder.

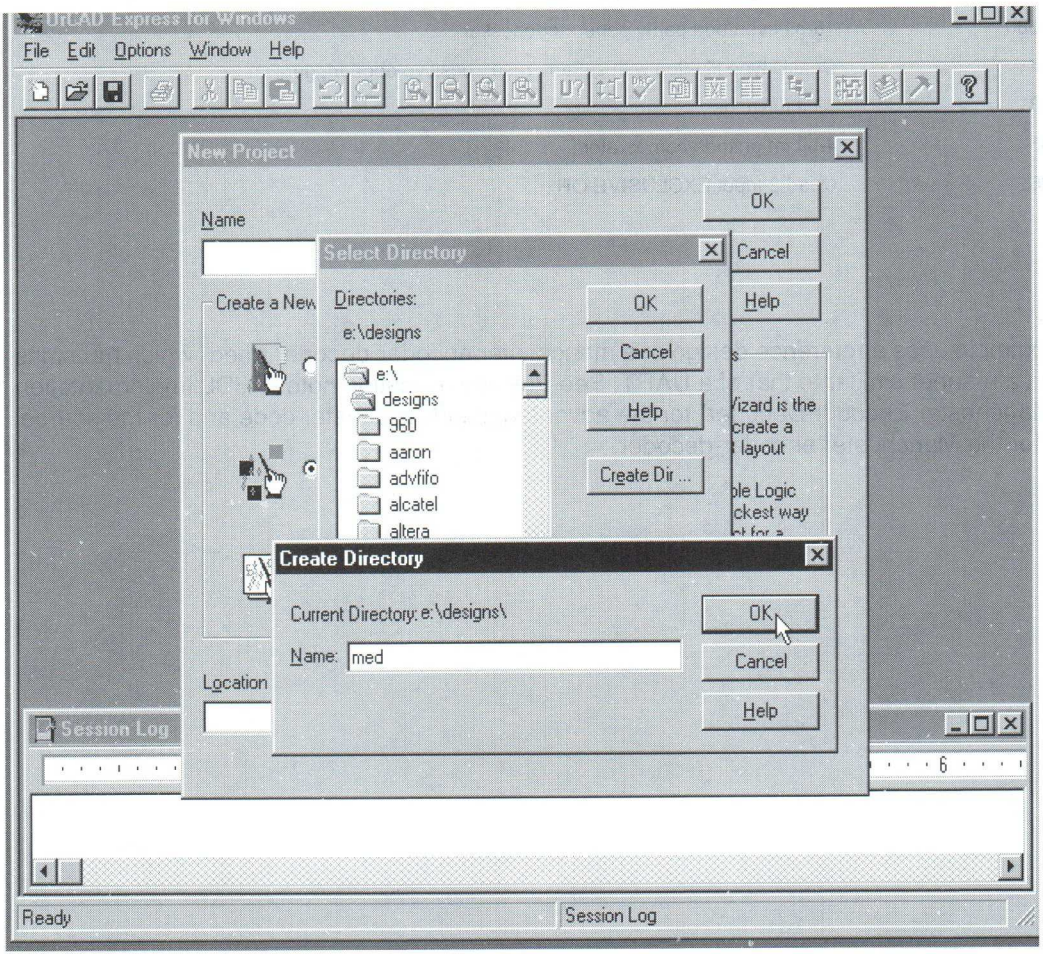


Figure 1. Starting a project

To begin, from the Windows Program Manager, double-click on the OrCAD Express icon to invoke OrCAD. Select File New - Project from the menu bar at the top. Select Programmable Logic Wizard, create the directory med, and name the project med. OrCAD Express supports top down and bottom up design. This example starts with a VHDL description of a manchester encoder (me.vhd) and creates a symbol with that name. See Chapters 4 and 7 of the OrCAD Express for Windows manual for project manangement using OrCAD.

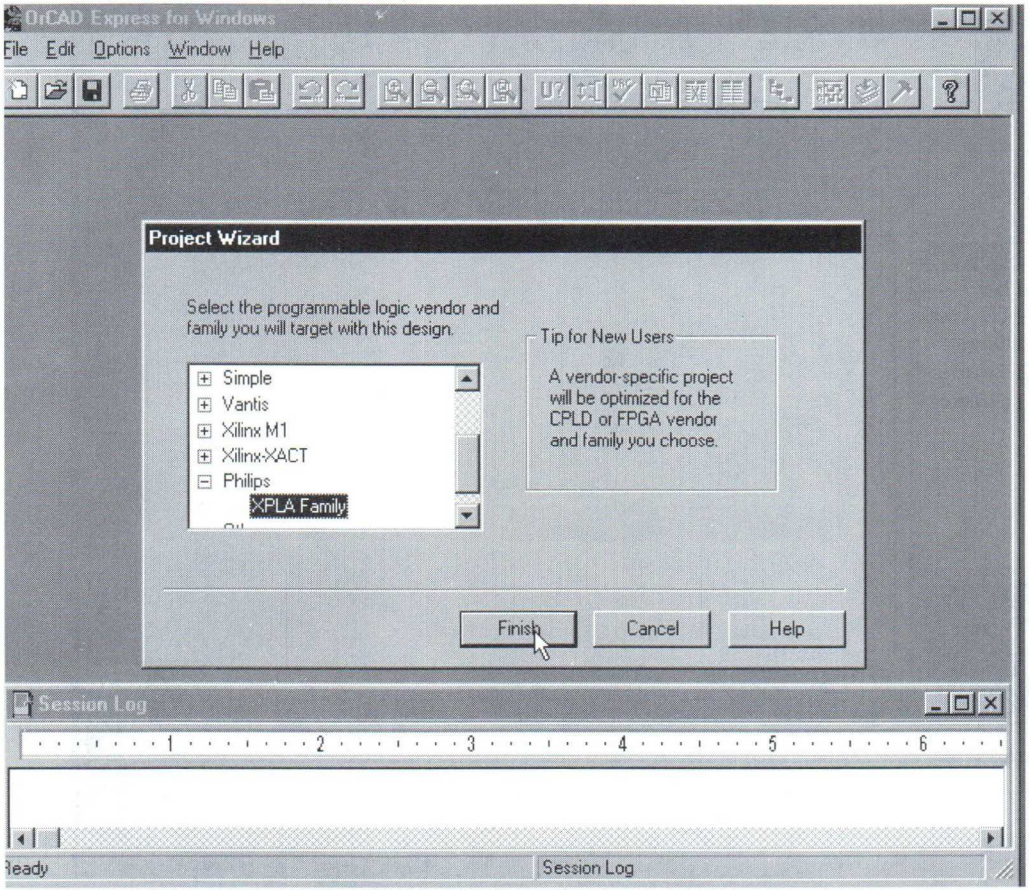


Figure 2. Targeting Philips CoolRunner CPLDs

Selecting Philips XPLA as a target allows either the XPLA 1 or XPLA 2 family to be used. The XPLA 1 family includes 5 volt (PZ5XXX) and 3 volt (PZ3XXX) devices in pinouts compatible with the Altera 7000 Series CPLDs. The XPLA architecture provides considerably more product terms than alternative CPLDs. Since a PLA structure is used in conjunction with a PAL structure, there is also considerably higher flexibility in the use of the PTs. The XPLA 2 family consists of the PZ3960, a 3 volt 960 macrocell device scheduled to sample in Q1 1998. The 320 macrocell PZ3320 is scheduled to sample in Q3 1998.

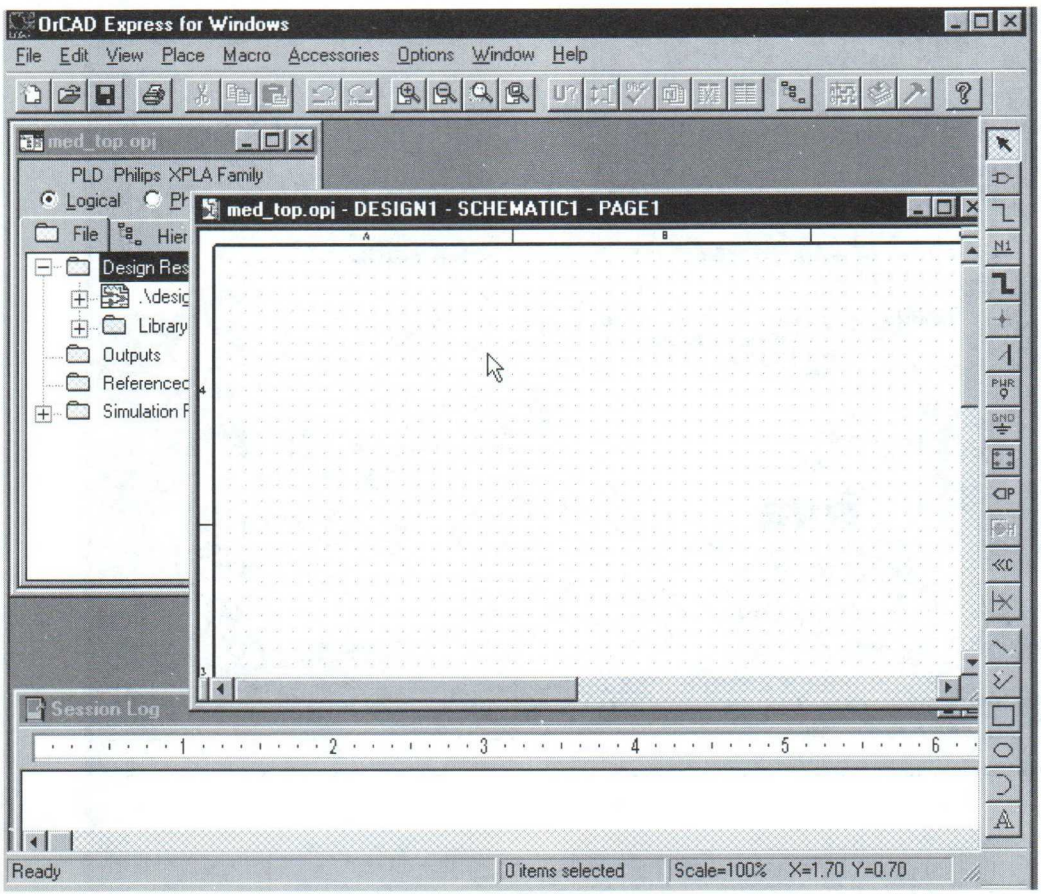


Figure 3. Creating a schematic

To create a schematic, select **File New - Design**. There are several methods of executing an operation in OrCAD Express, including the menu lbar at the top already used. In addition to the top pull down menu bar, the Express window contains fixed menus at the top and right hand side. Some of the operations which follow may be done faster using these fixed menus. In this note, many operations are done by clicking on the right hand side fixed menu.

Use the schematic window scroll bars to locate the center of the schematic. The schematic will contain symbols for a manchester encoder (me), manchester decoder (md), and a primitive cell from the Philips symbol library (ps.olb).

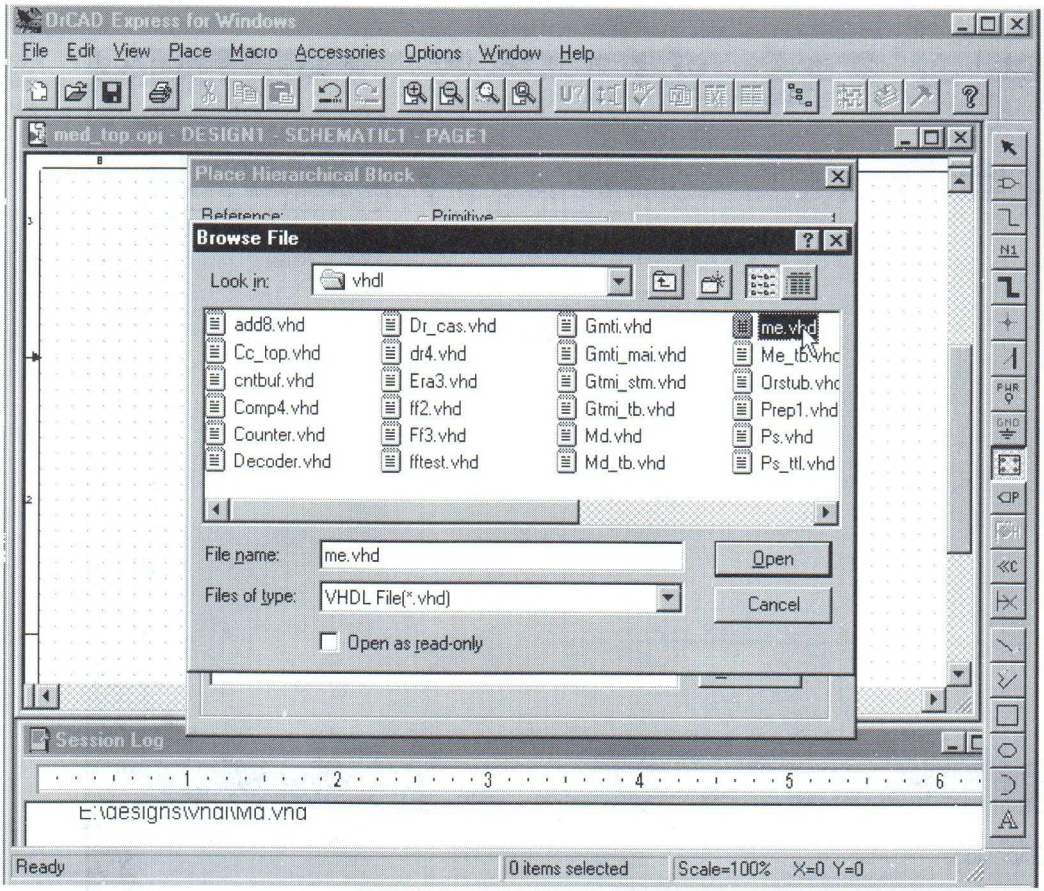


Figure 4. Placing a hierarchical block

OrCAD Express can create a symbol from VHDL source. The me and md symbols will be created and placed on the schematic. To create a symbol for me in which the functionality is defined in the me.vhd file, select Place Hierarchical Block by clicking on the icon (10th down) on the fixed menu.

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.

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 Open.

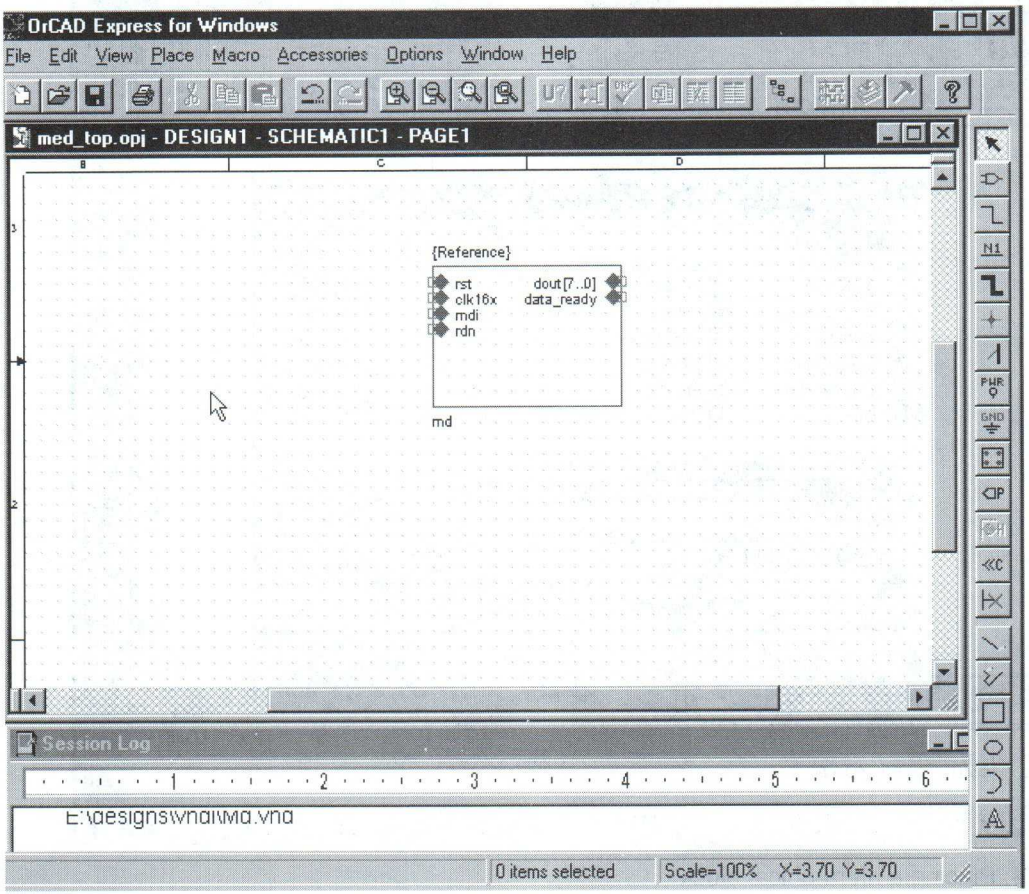


Figure 5. Placing the md hierarchical block

Once the symbol is generated, place it in the upper center of the schematic page by dragging the cursor from the top left corner to the bottom right corner to define the symbol size.

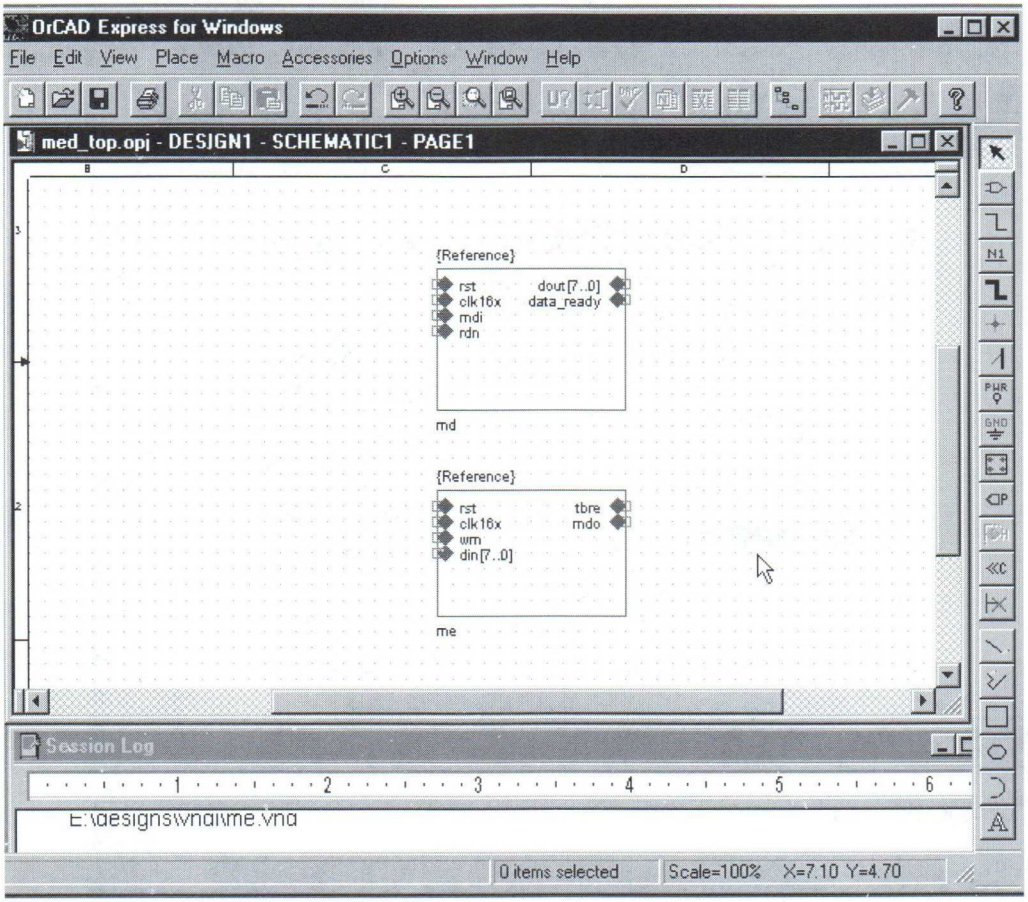


Figure 6. Schematic with me and md blocks placed

Repeat the Place Hierarchical Block steps for the manchester encoder, and add it to the schematic.

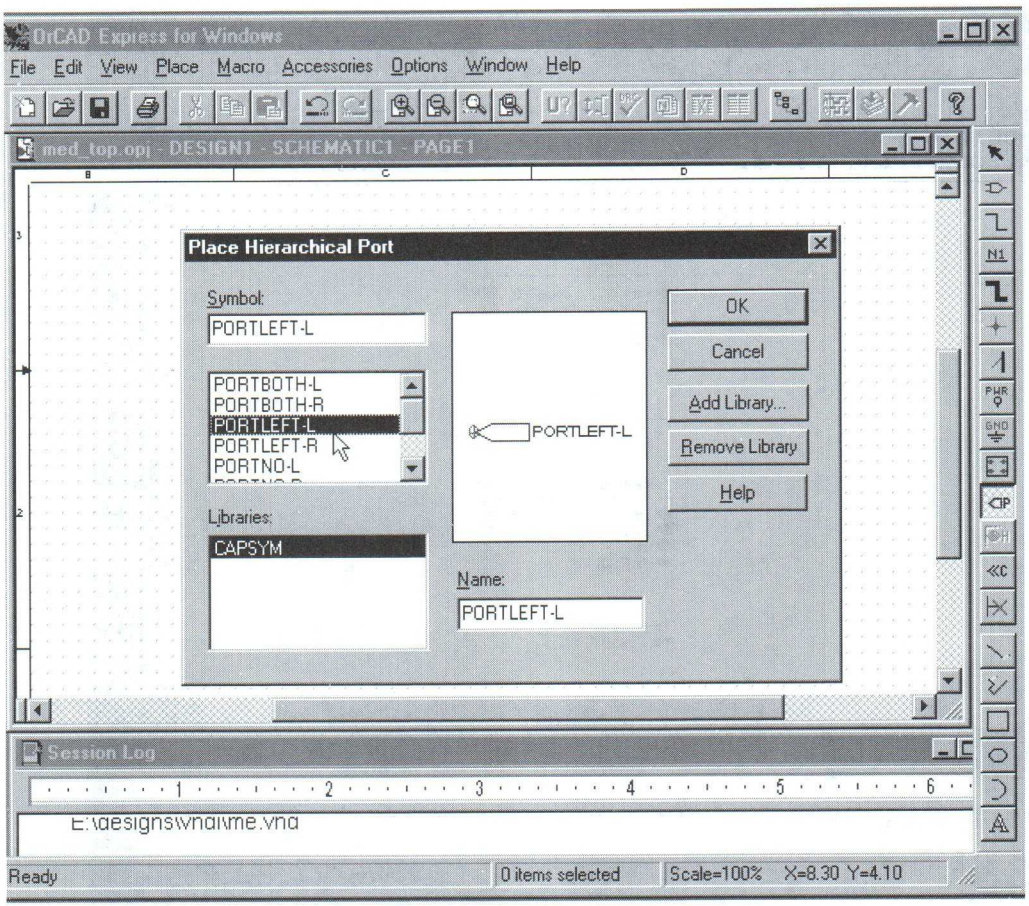


Figure 7. Placing primary inputs and outputs

From the fixed menu, select Place Hierarchical Ports (11th entry down) and add the ports provided in the dialog box. For outputs on the schematic right hand side, select the PORTLEFT-L entry. For inputs on the left hand side, select the PORTRIGHT-R entry.

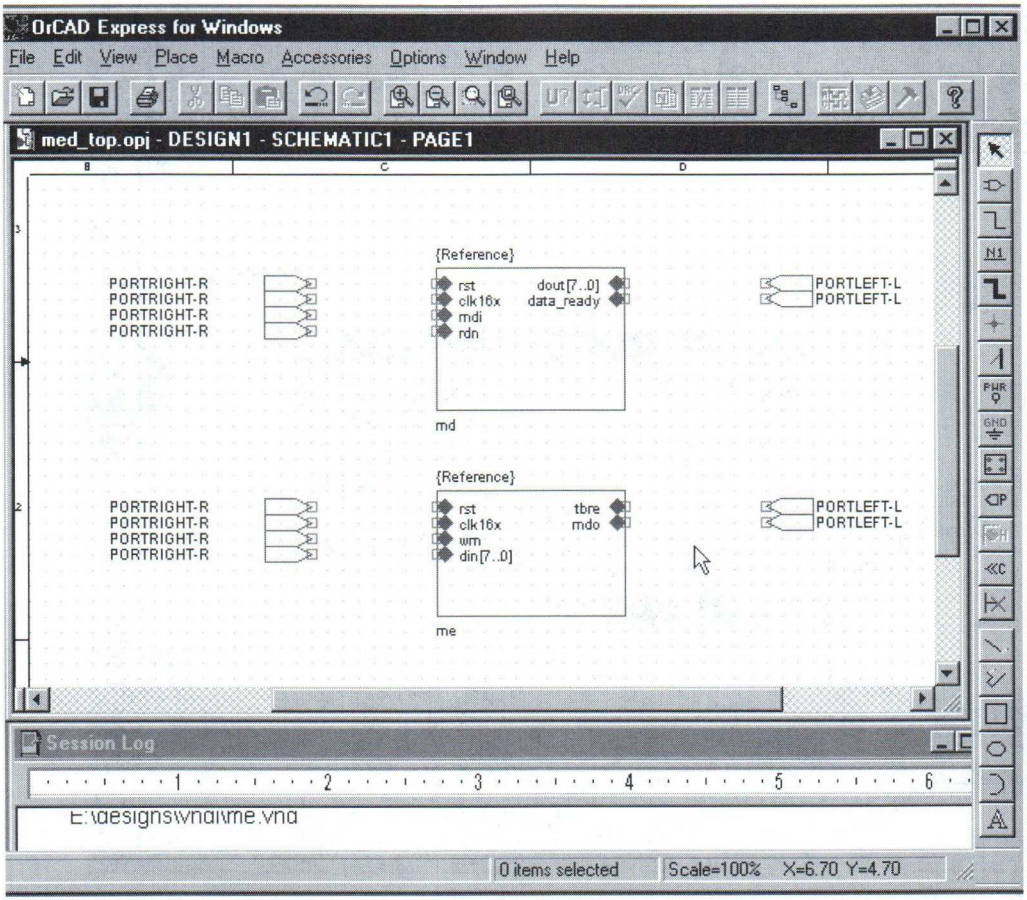


Figure 8. Schematic with hierarical ports added

Place the hierarchical ports adjacent to the symbol ports to which they are to be connected to on the schematic.

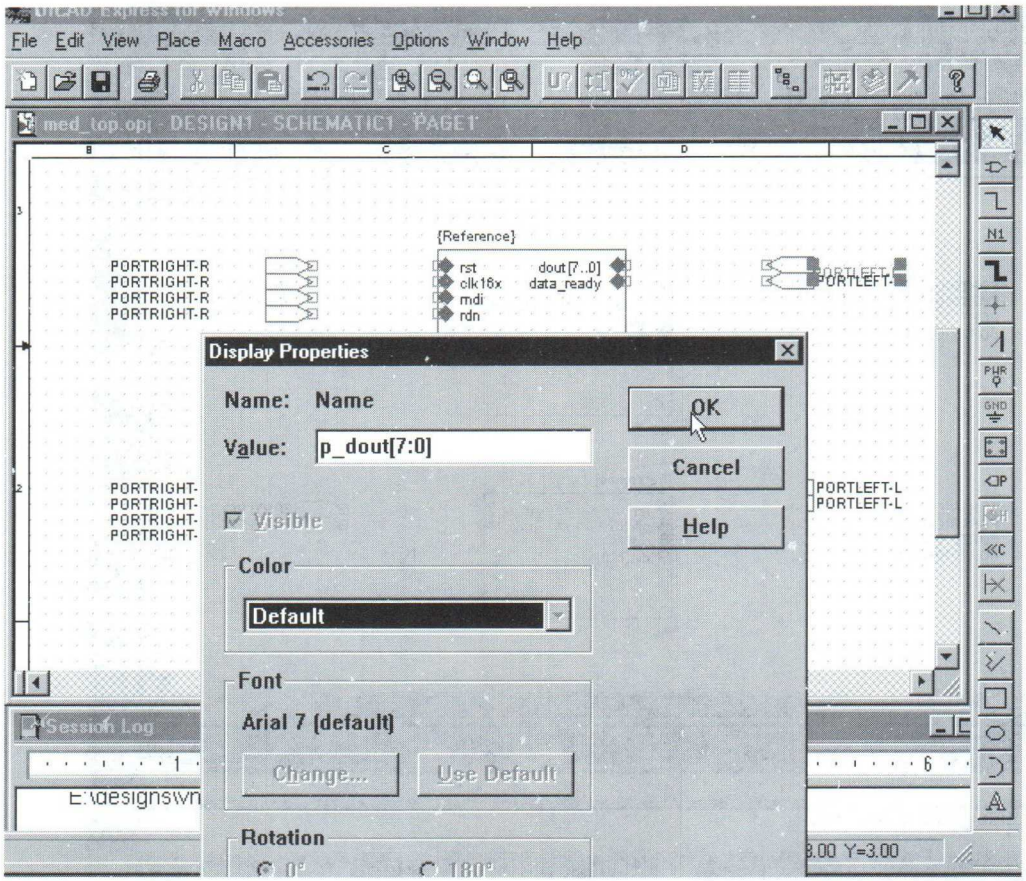


Figure 9. Labeling ports

Double click on the port name template and name the ports in the dialog box.

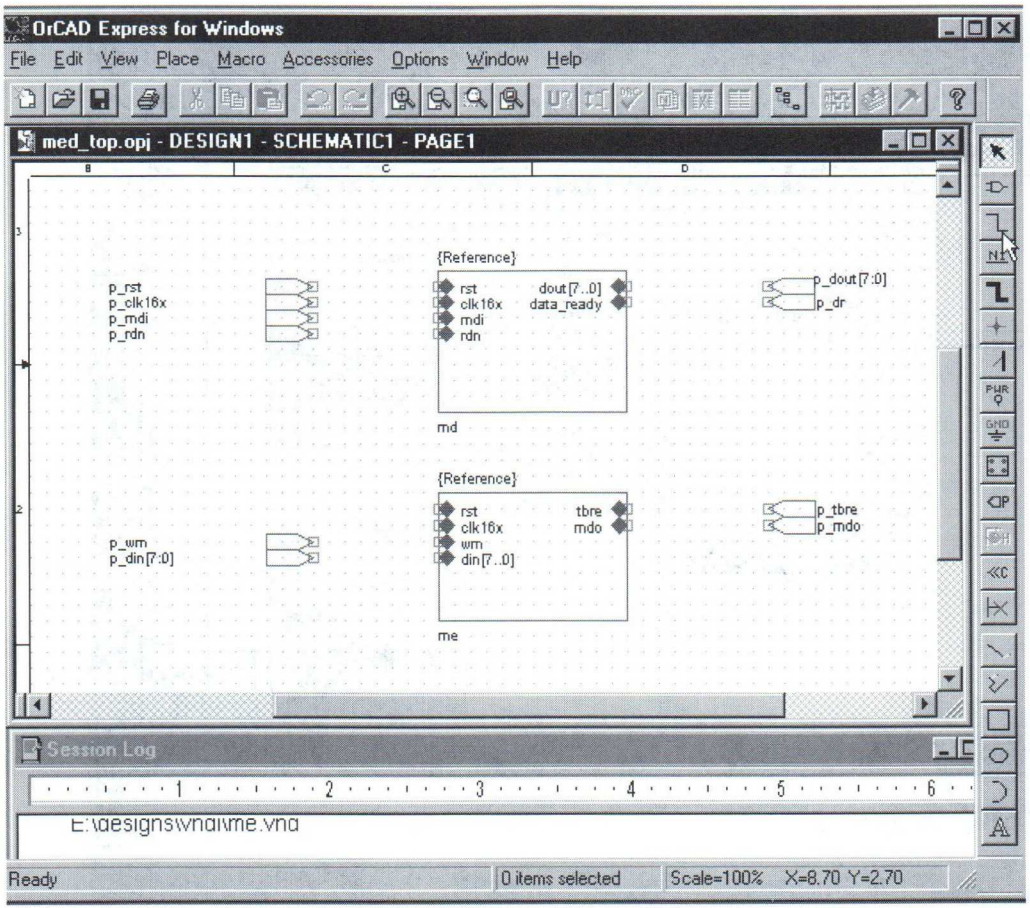


Figure 10. Schematic with labeled ports

The ports adjacent to rst and clk16x of me have been deleted since those adjacent to the md symbol can be routed to the ports on the me symbol.

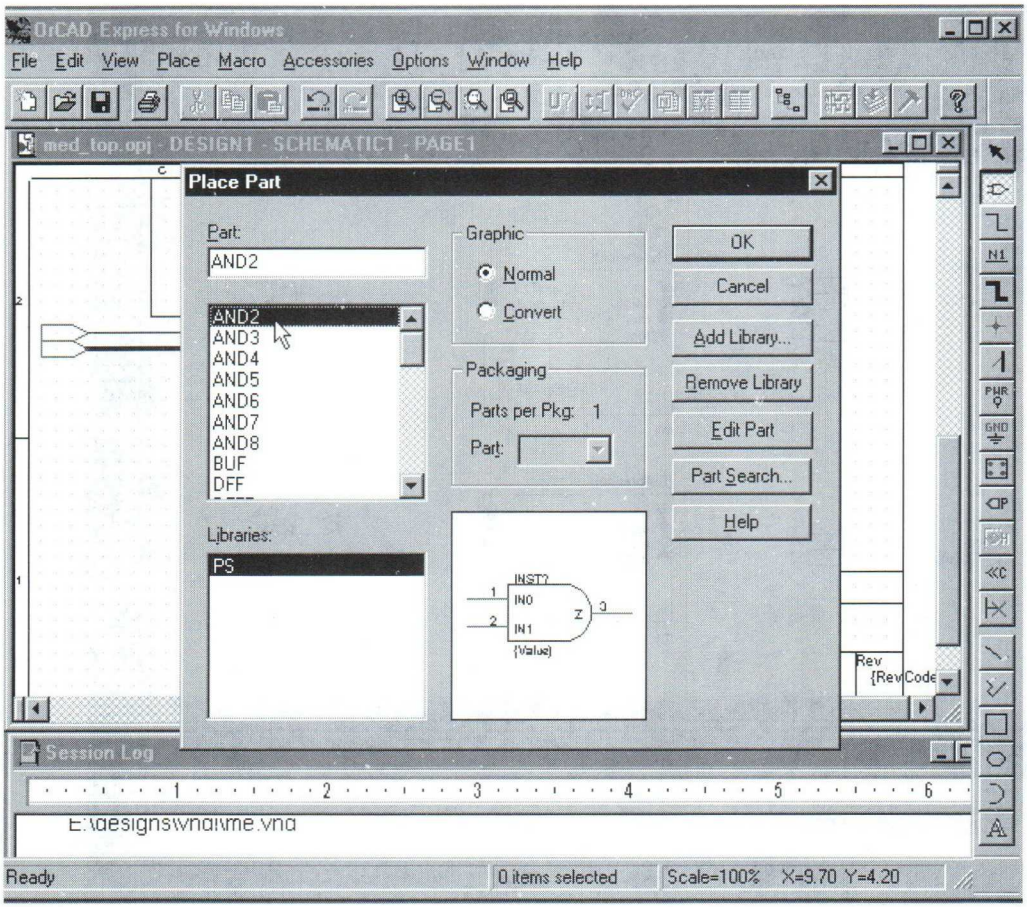


Figure 11. Adding a symbol from the ps library

The ps.olb library contains library primitives for schematic capture. This library contains basic gates. Hierarchical blocks can be built from these gates. A limited 74XX library is under devleopment. The Minc PL-Designer interface includes a library of 74XX cells.

Click on the fixed menu's part entry (2nd entry down) and select an AND2 from the ps library. Place the component on the schematic.

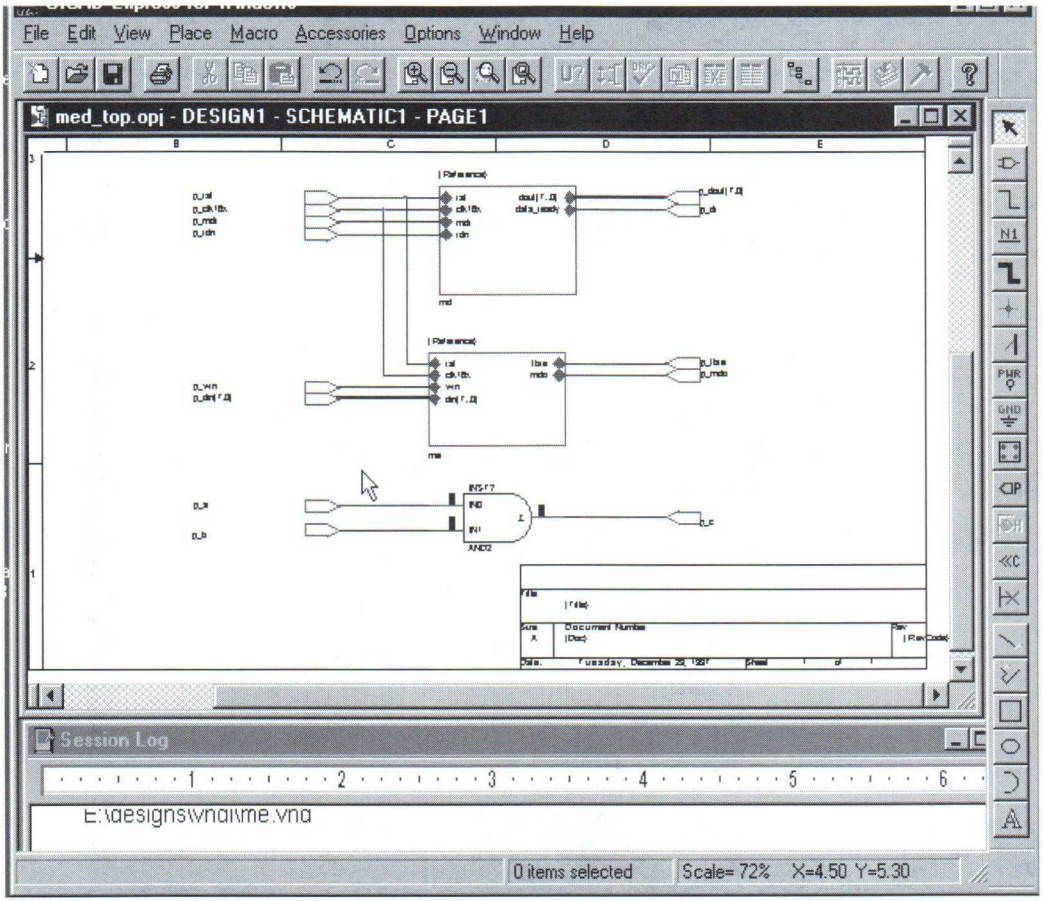


Figure 12. Routed schematic

From the fixed menu, select the wire icon (3rd entry down) and route the wiring as shown. For din[7:0] and dout[7:0], route the signals using Place Bus from the fixed menu (5th entry down).

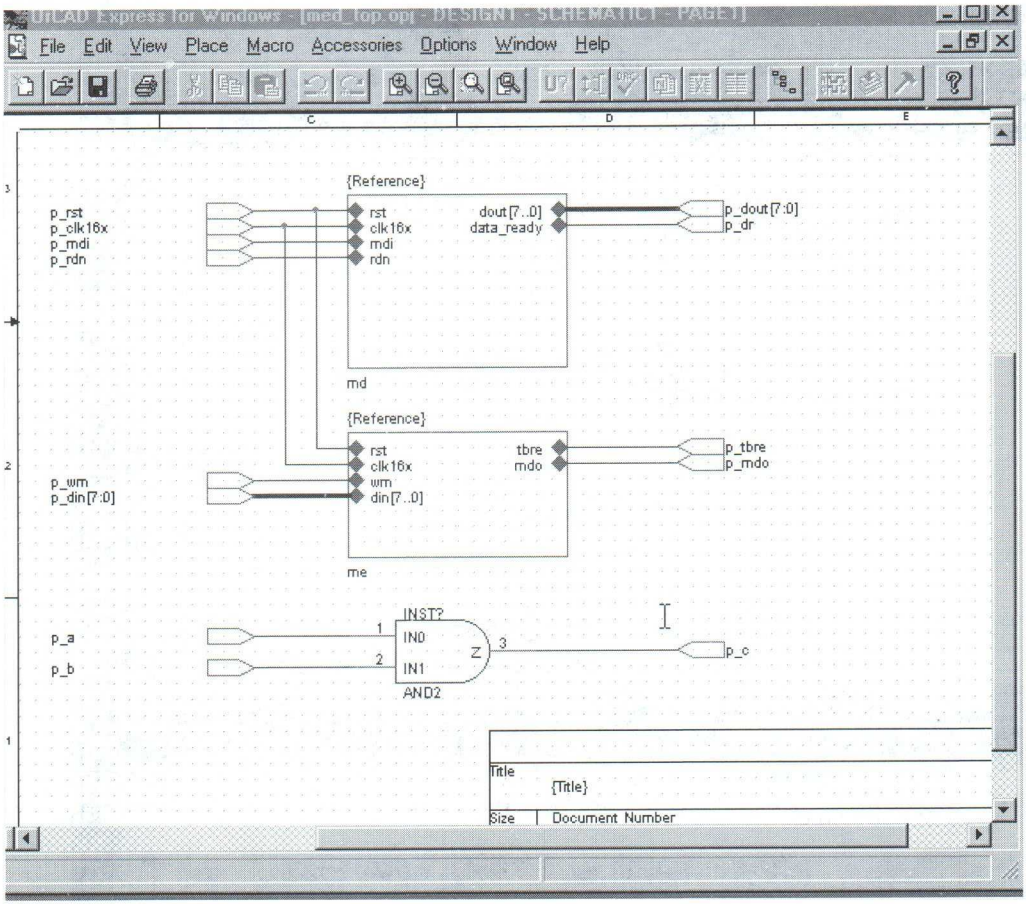


Figure 13. Completed schematic.

Select the me symbols, and **View - Descend Hierarchy** from the top menu lbar. Review the VHDL code, and select **Edit - Check VHDL Syntax.** Correct the VHDL syntax if necessary and rerun the step.

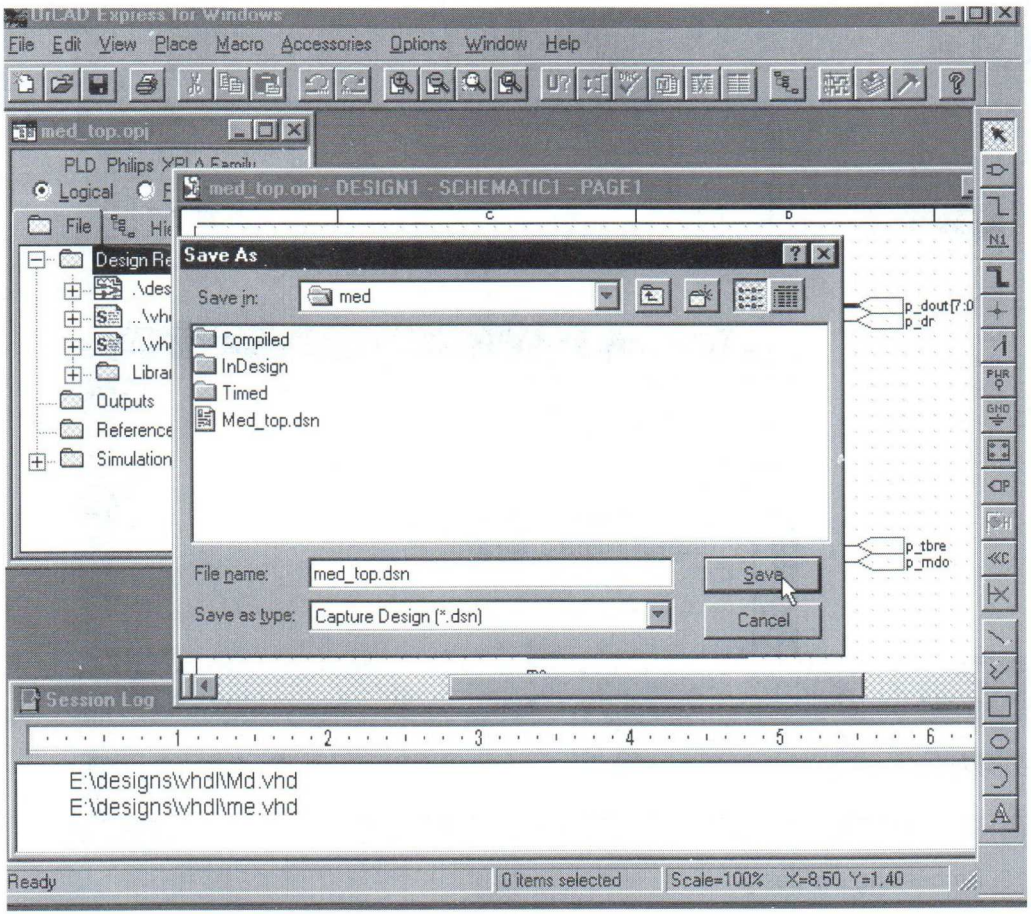


Figure 14. Save the design

From the top menu bar, select **File Save**. In the dialog box, migrate to the correct location and save the design as med_top.

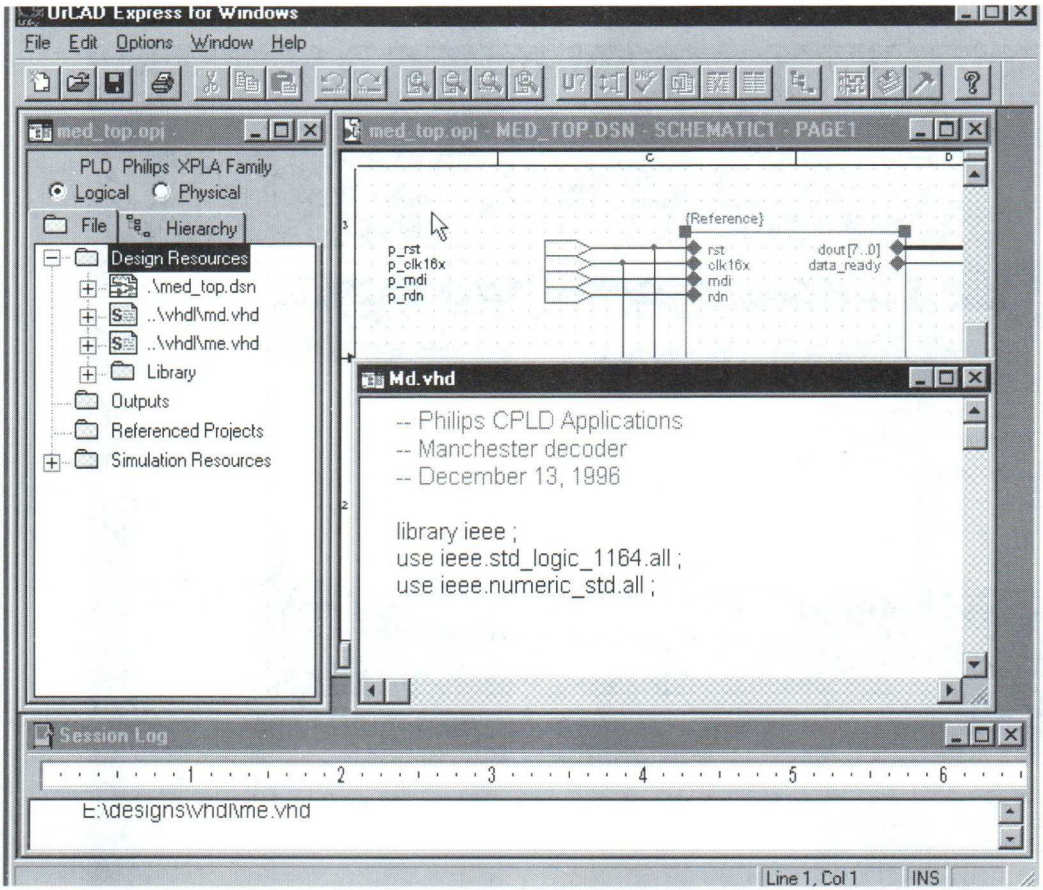


Figure 15. Processing the design

There are three steps in processing the design to an edif file. This is the design entry point for the fitter.

1. Update references
2. Design Rule Check
3. Build

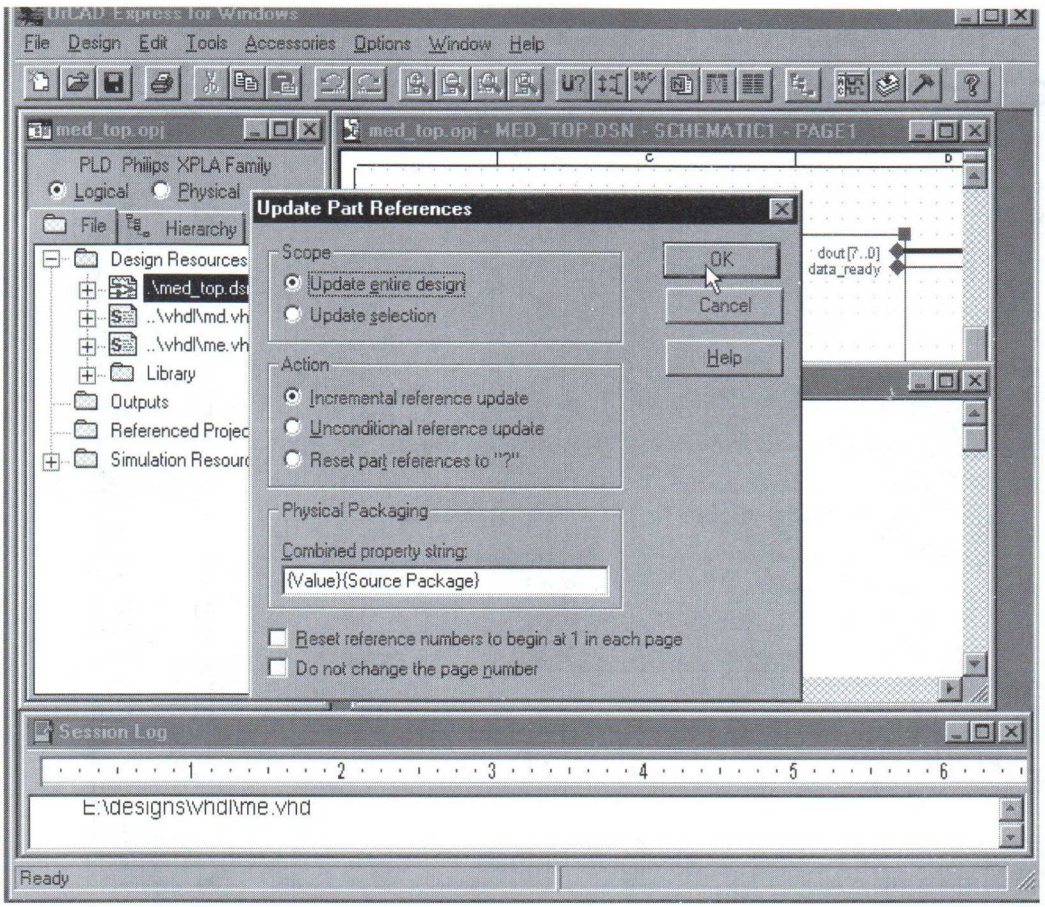


Figure 16. Update part references

The first step in processing the design is to select med_top.dsn as the design resource. From the menu bar, select **Tools-Update Part References**. Accept the defaults in the dialog box and click OK. This function provides unique instance numbers for each instance in the design.

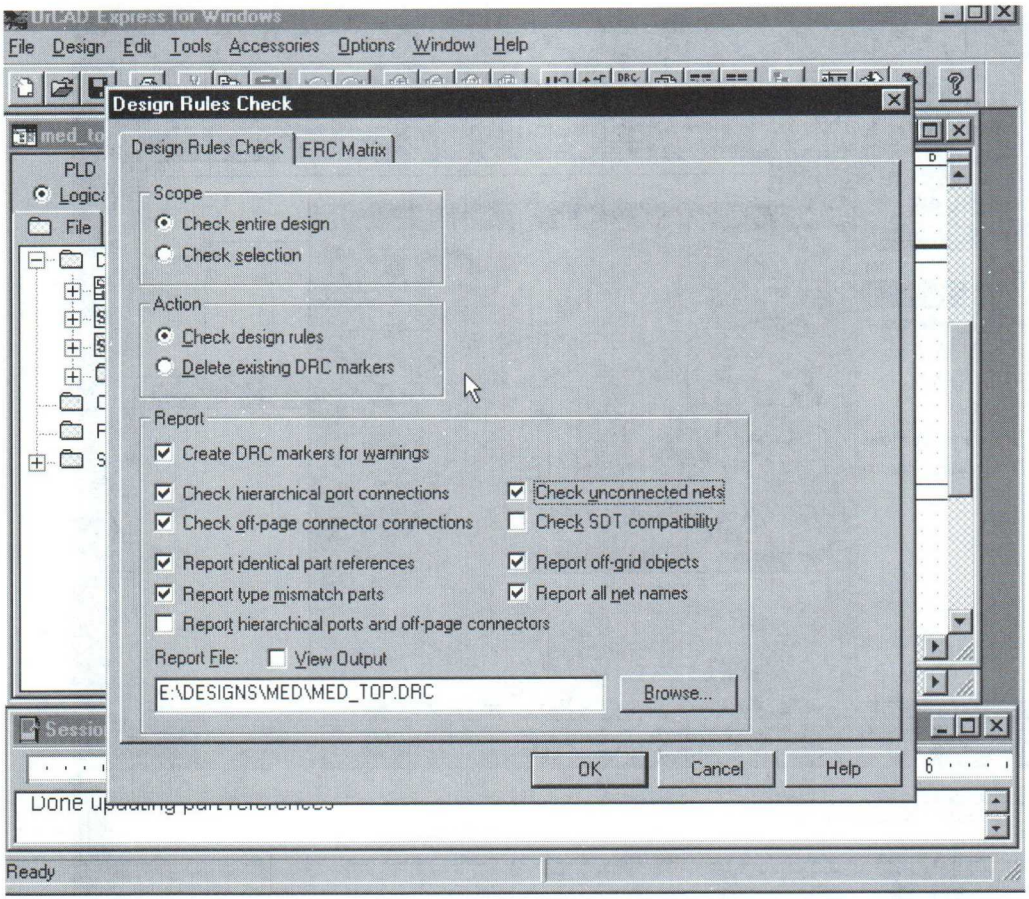


Figure 17. Design Rule Check

From the menu bar, select **Tools-Design Rule Check**. Enable the options shown, and click OK. the DRC generates the med_top.drc file in the output folder. DRC violations can be viewed in the session log. Correct any violations until DRC passes.

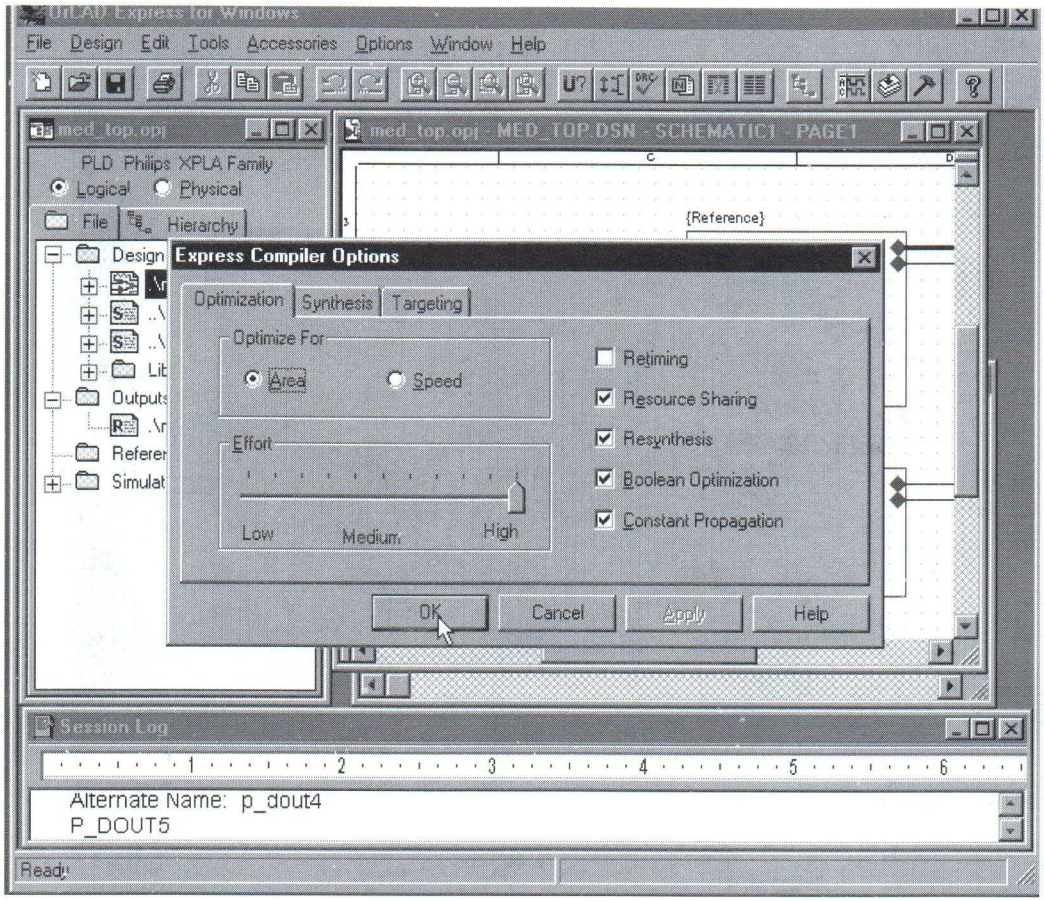


Figure 18. Compile operation

Designs can be compiled from within OrCAD Express or by invoking XPLA Designer. Both paths will be described.

From the menu bar, select **Tools - Build**. Click OK to accept the default compile options.

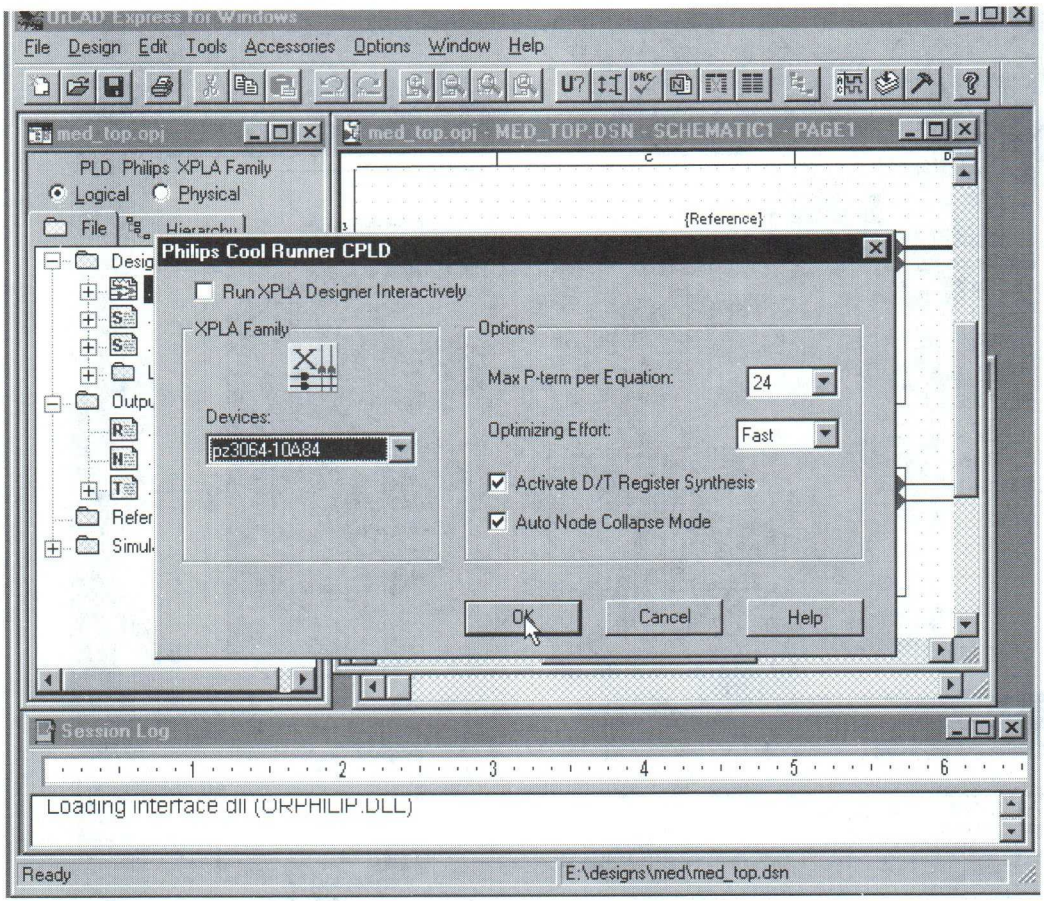


Figure 19. Compiling from within OrCAD Express

To run XPLA Designer from within the OrCAD Express environment, select the PZ3064-10A84 device, and compile options max p-term per equation as 24 and auto-node collapse as shown. The auto node collapse and max p-term per equation options generally have significant effect on the speed and area results. Clicking OK will cause design results to be placed in the output folder.

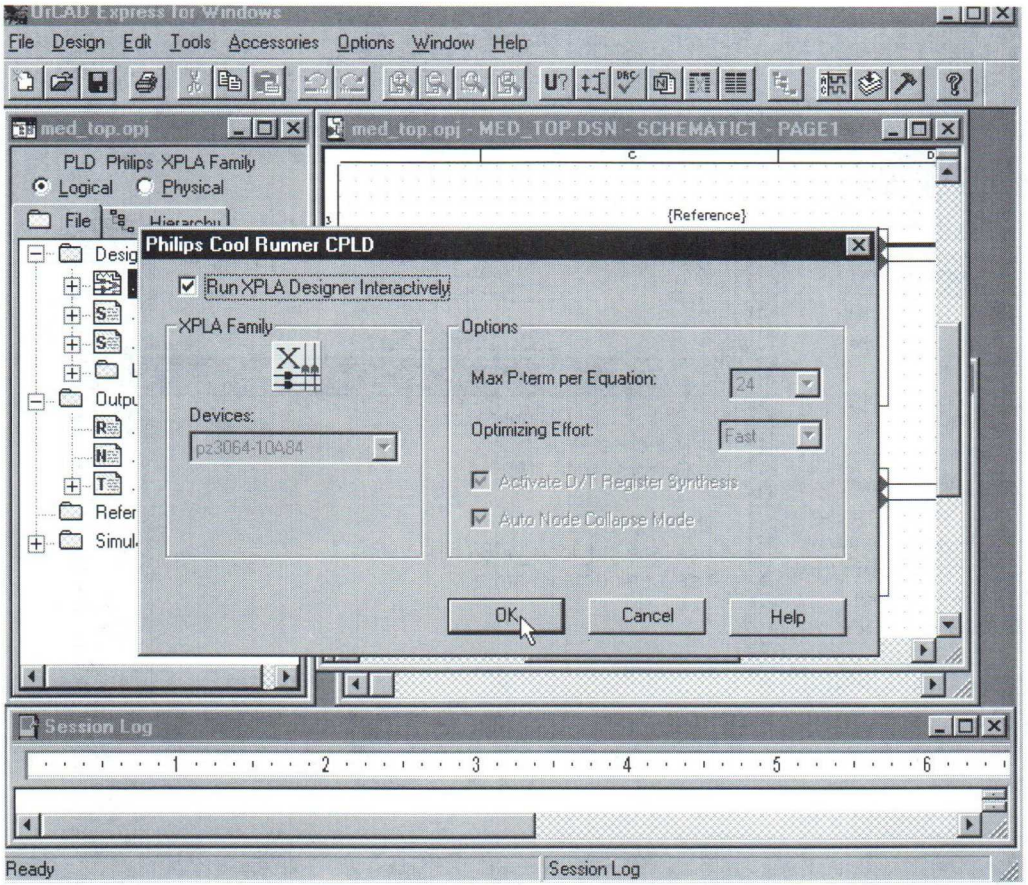


Figure 20. Running XPLA Designer interactively

To invoke the XPLA Designer graphical user interface, check Run XPLA Designer Interactively and OK in the pop up dialog box. XPLA Designer needs to be run interactively to generate pin assignments.

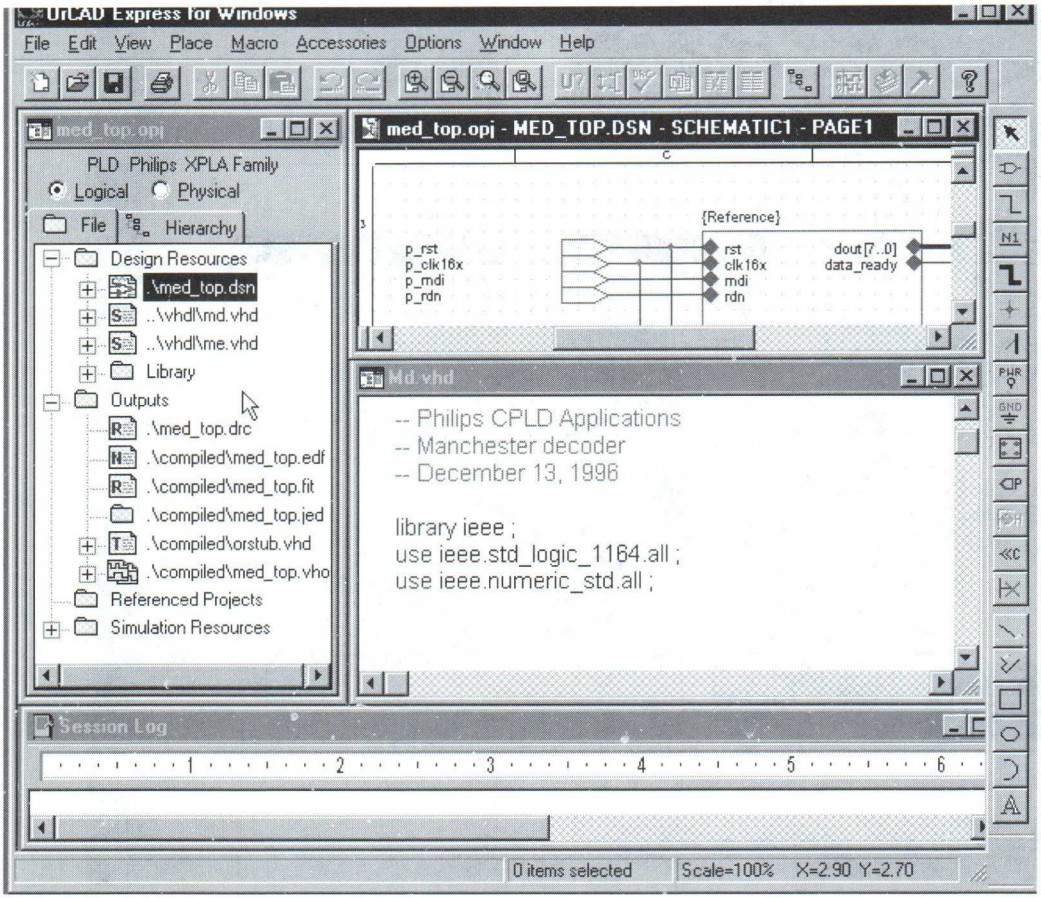


Figure 21. Med compilation results

When the compilation is successful, the output folder contains the results, including the jedec, fit, and vho simulation files. These files can be viewed from within OrCAD Express by double clicking on the file name.

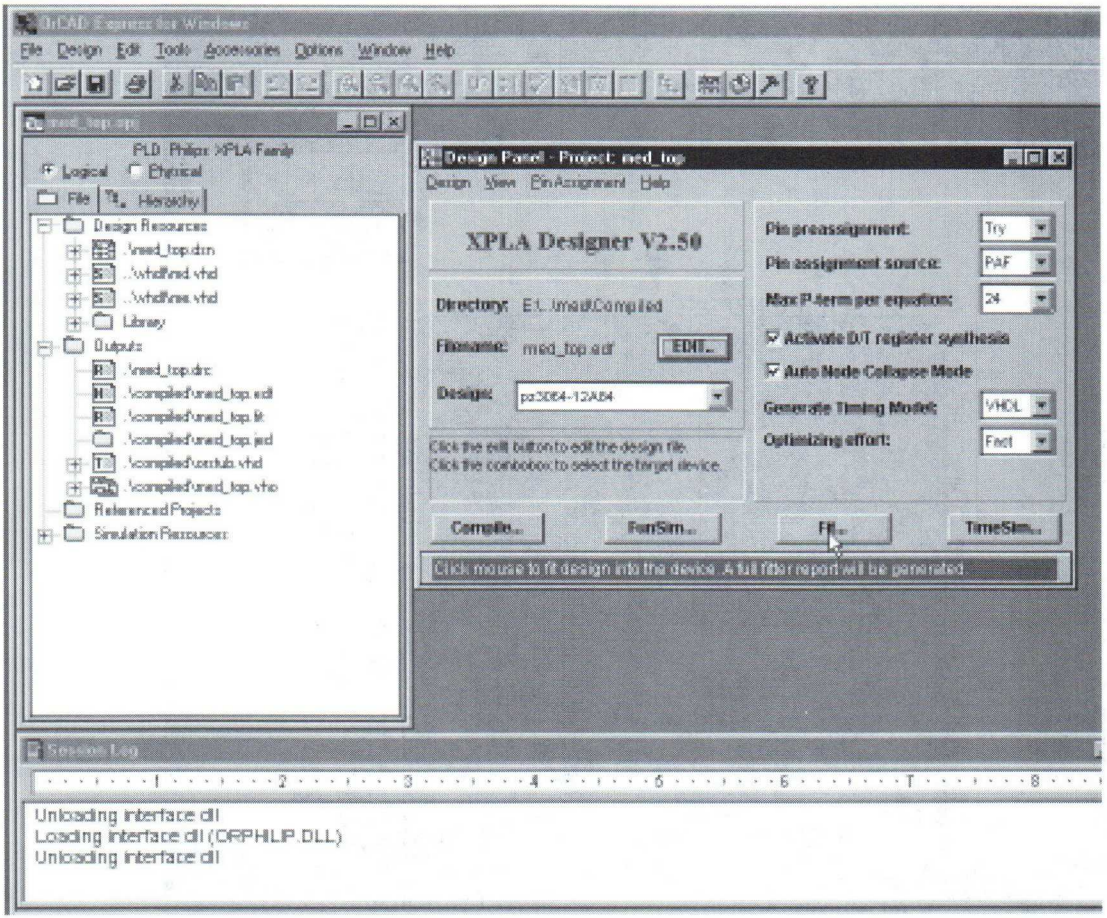


Figure 22. Running XPLA Designer interactively

This XPLA Designer graphical user interface is displayed. Enter VHDL in **Generate Timing Model** to create a delay annotated VHDL model.

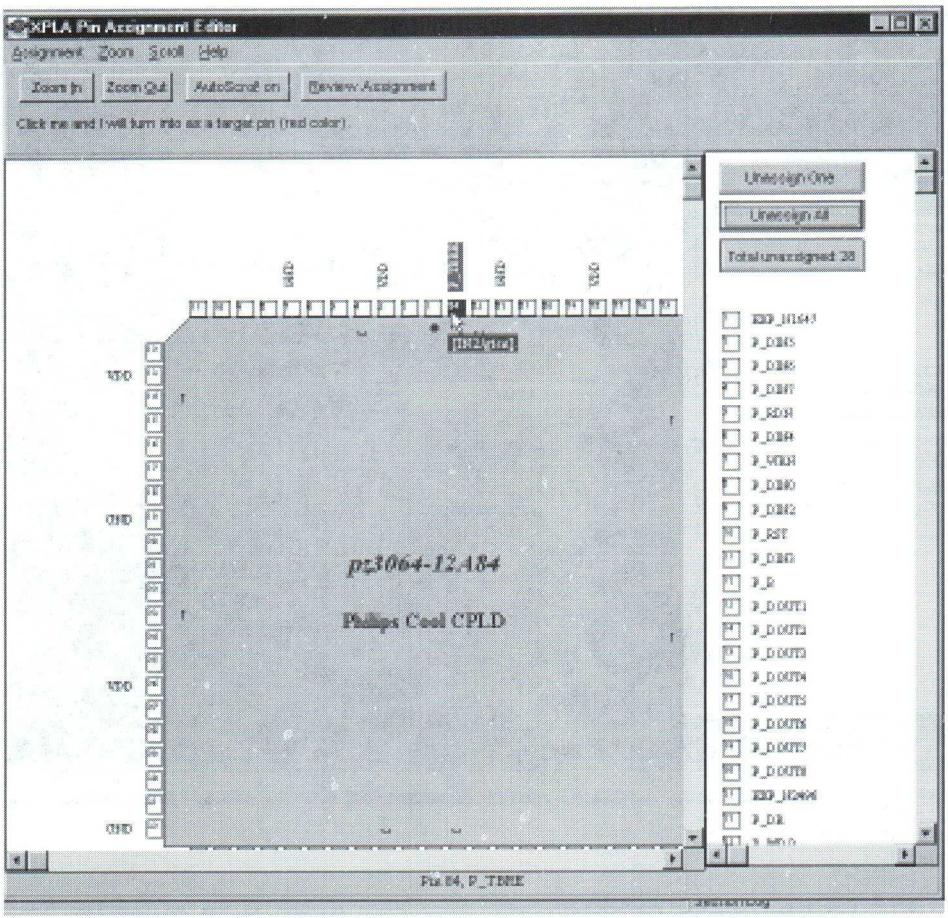


Figure 23. Pin assignment editor

To assign pins, select the **Pin Assignment** entry from the menu bar of XPLA Designer. Clicking **Unassign All** deletes assigned pins from package pins and moves them to the right window. To assign a pin, click on the pin name in the right window and click again on the desired location on the package. From the menu bar, select **Assignment Save**. When compiling in XPLA Designer, select PAF as the pin assignment source.

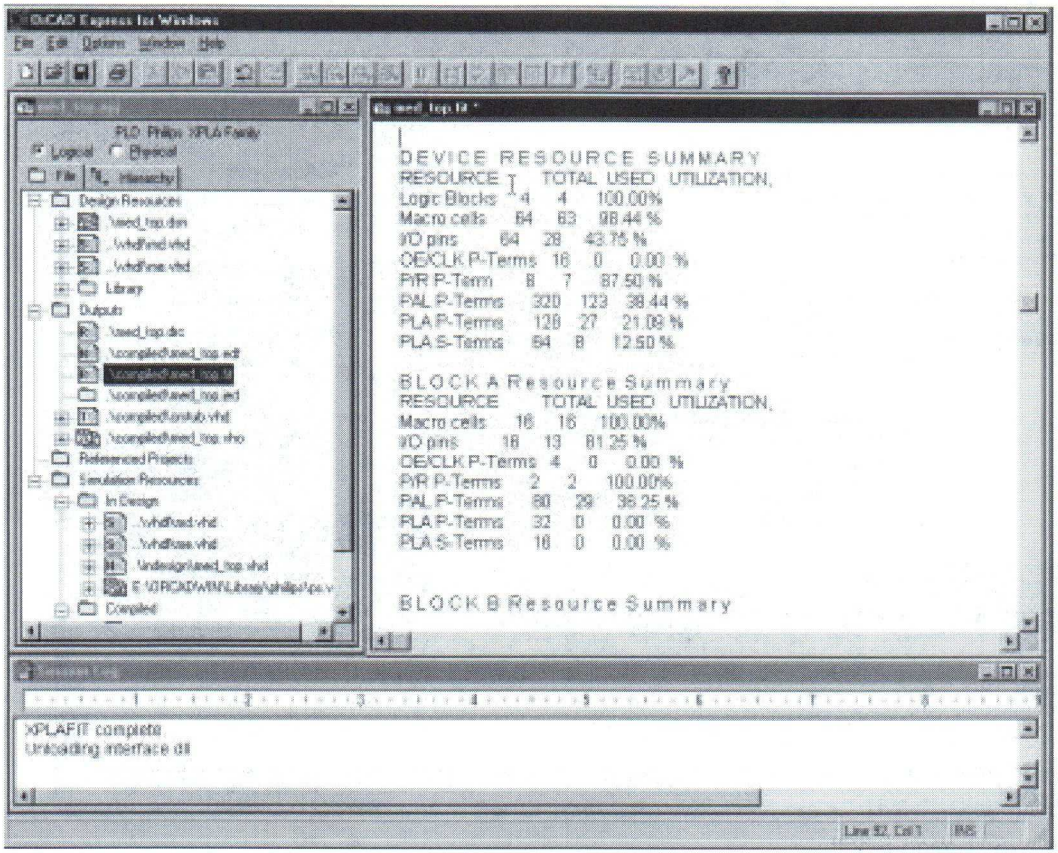


Figure 24. Fitter report

From the menu bar, **View - Fitter Report** wil cause the fitter report to be displayed. The first section contains the pin assignments. This is followed by a summary of the device utilization. Tha last section contains resource utilization on a logic block basis, A - D in the case of the PZ3064.

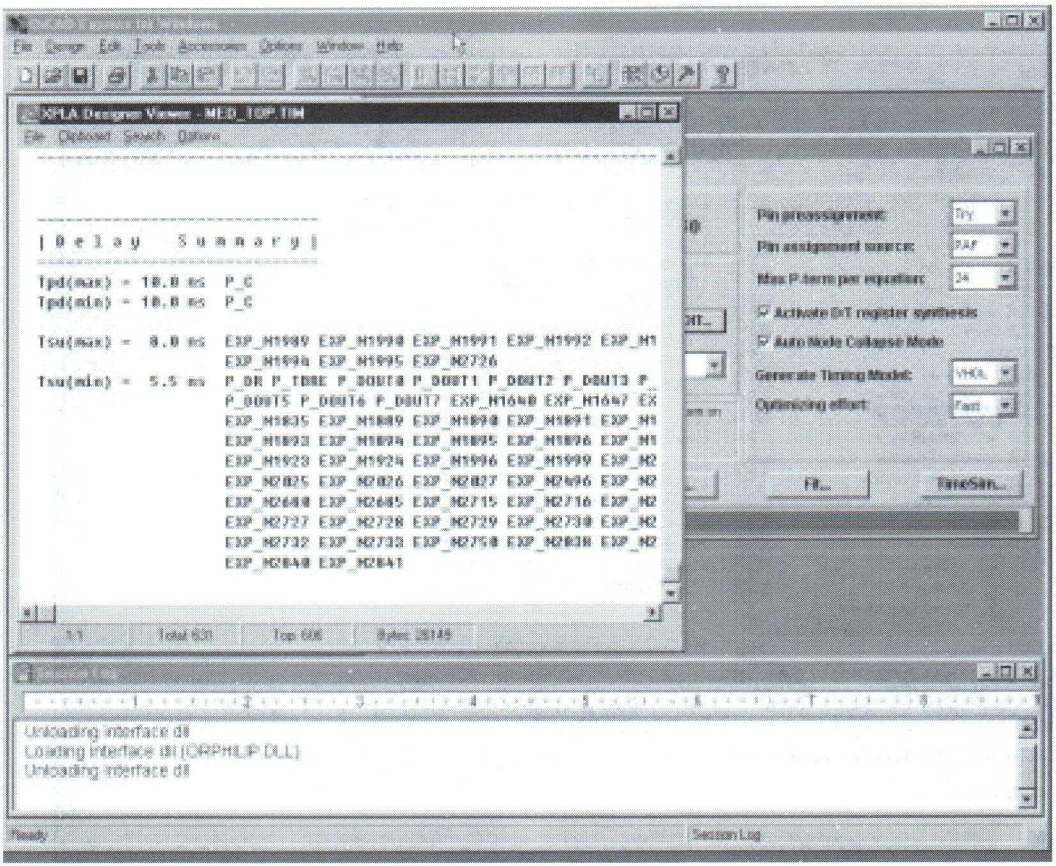


Figure 25. Timing results

Select **View - Timing Report** from the menu bar to display the timing results. The detailed fanout/fanin are given first. At the bottom of med_top.tim is a summary of the delays.

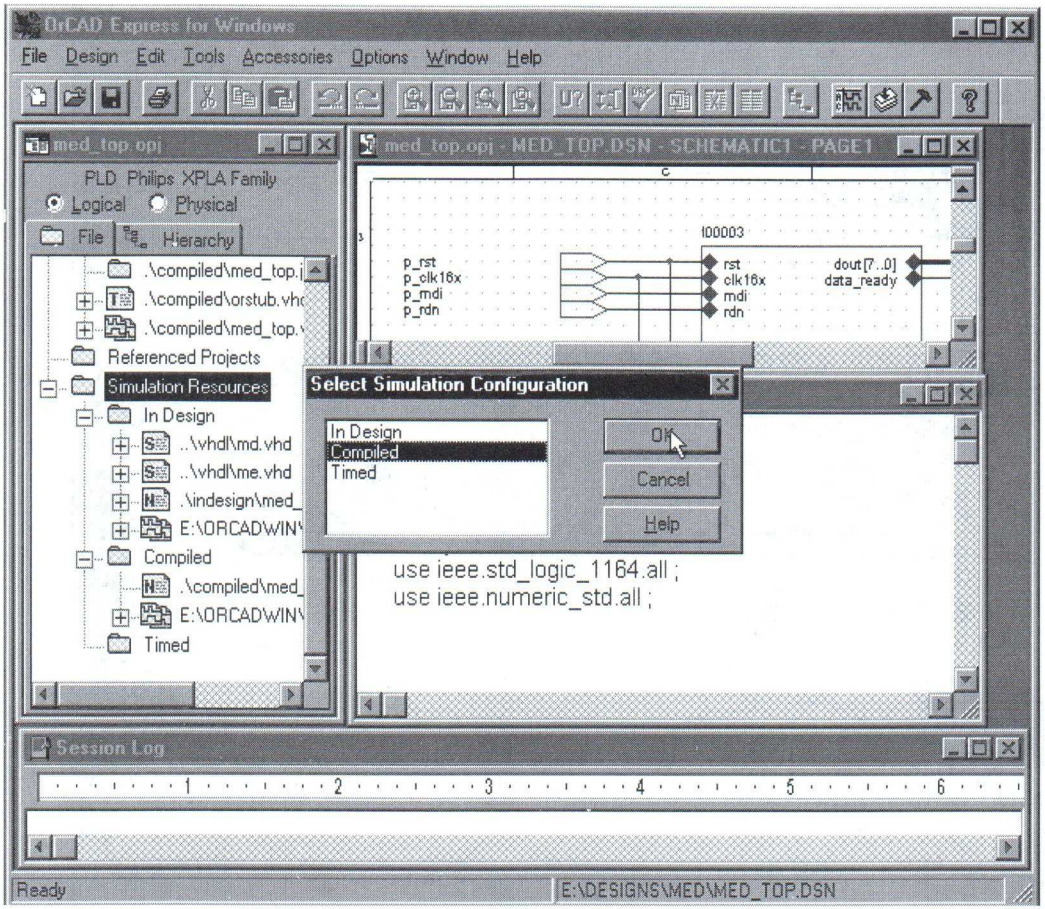


Figure 26. Starting a simulation

OrCAD Express Simulate allows users to simulate the design pre and post layout. Designs can be simulated using an interactive interface for generating stimuli or a VHDL testbench. Most schematic only based design use the interactive approach. Since this example uses vhdl as sourct, the VHDL testbench will be used.

Highlite Simulation Resources in the project window and select **Tools - Simulate** in the top menu bar. This provides a dialog box in which the type of simulation is selected.

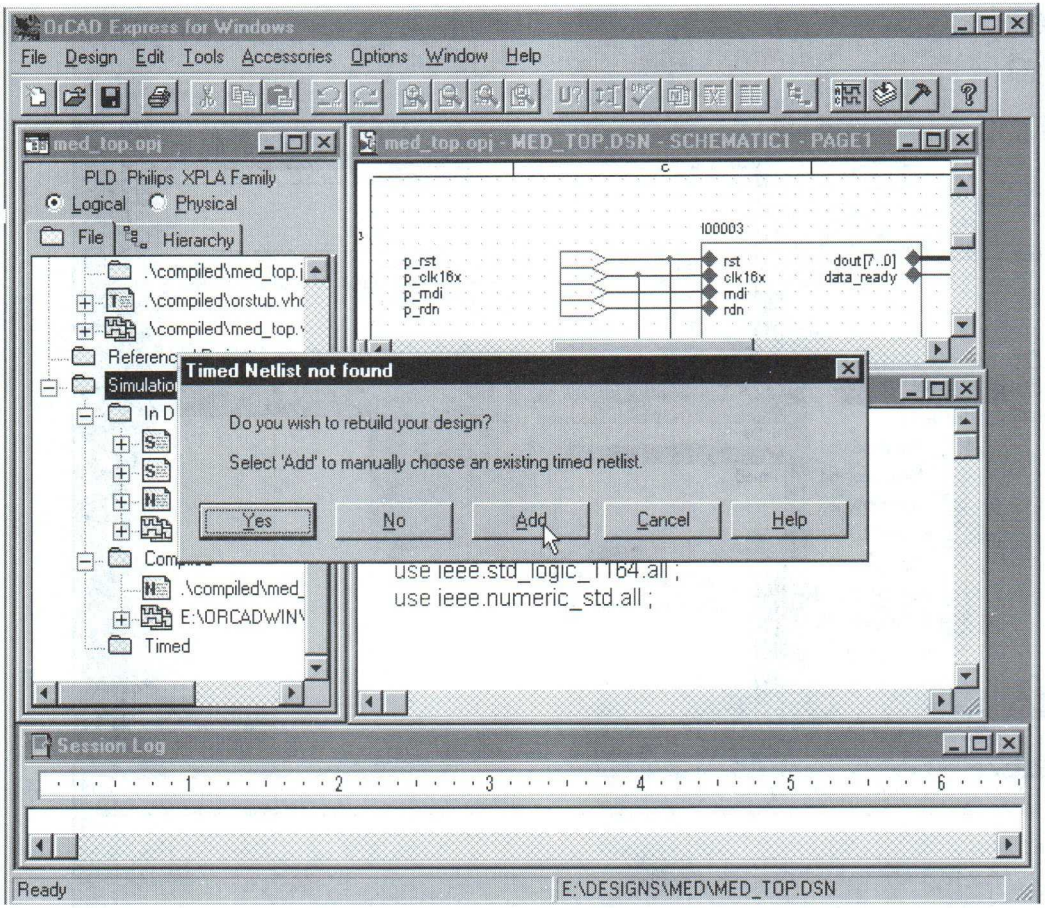


Figure 27. Timing simulation

If VHDL is selected as the option in **Generate Timing Model**, med_top.vhd is generated. This file is initially placed in the output folder. To use this simulation model, add it to the Timed folder as shown.

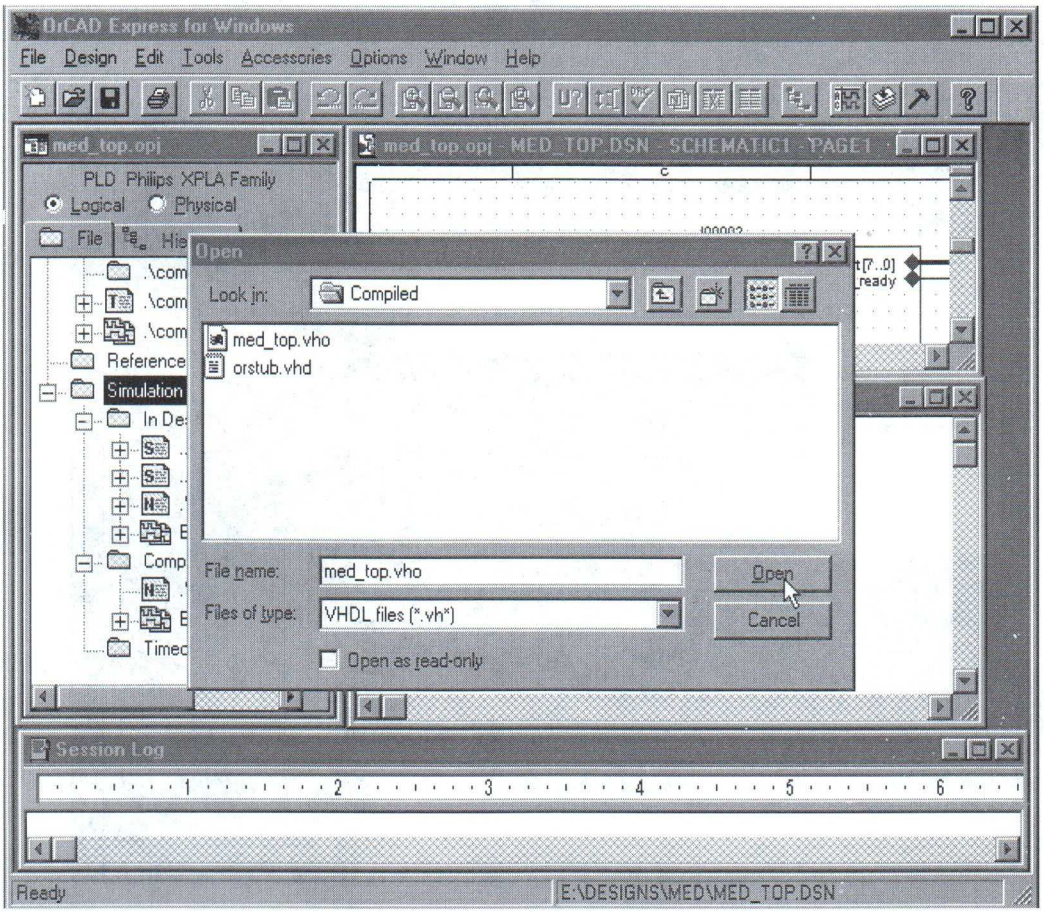


Figure 28. Adding med_top.vho to the timed folder

The dialog box allows the med_top.vho file to be moved to the Timed folder.

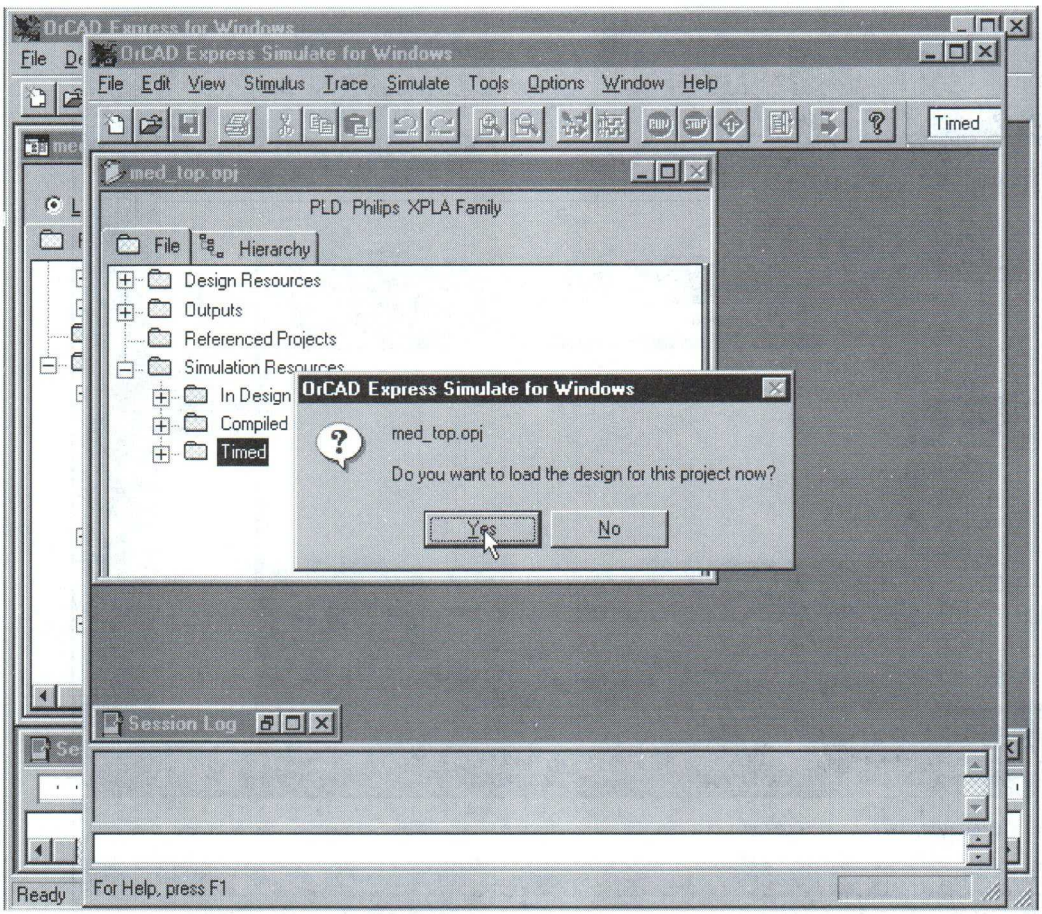


Figure 29. Initiating a simulation

The steps in simulation are

1. Start OrCAD Express Simulate.
2. Generate stimuli.
3. Set up displays of simulation results.
4. Running the simulation.

Select **Simulate - Reload Project** if the project is not loaded.

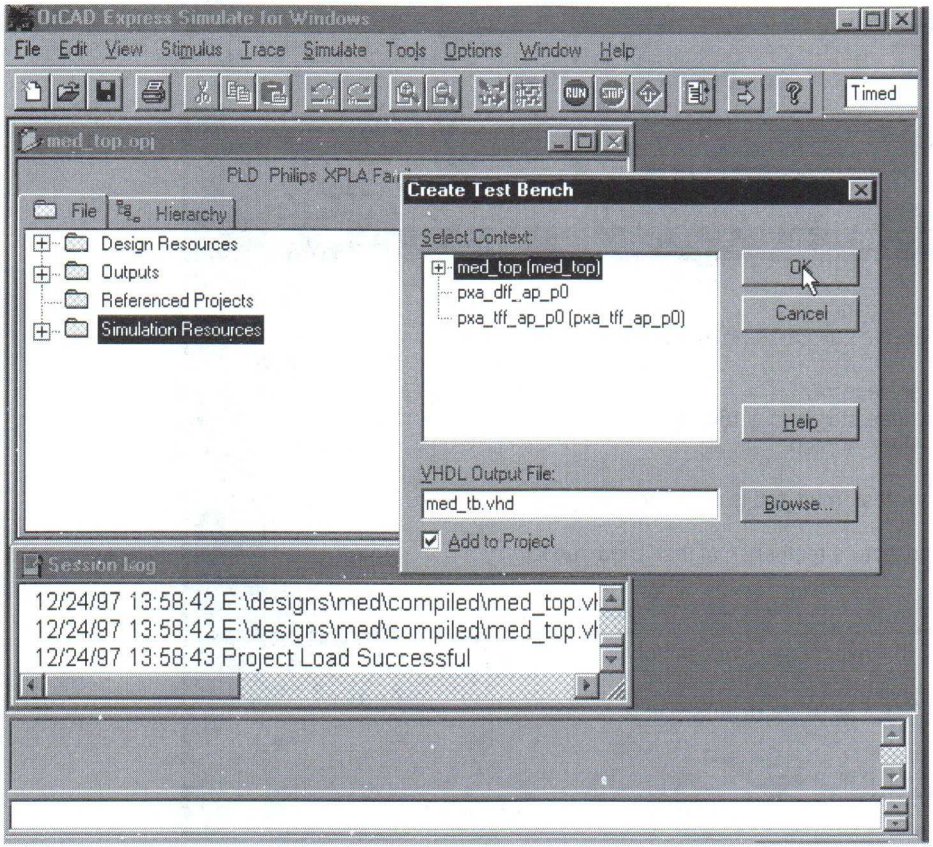


Figure 30. Generating stimuli

With Simulation Resources selected, select **Stimulus - Create Testbench** from the top toolbar. Highlite the med_top entry in the dialog box and click OK. Name the testbench med_tb.vhd. A vhdl testbench is created which can be compiled with the source med_top.vho.

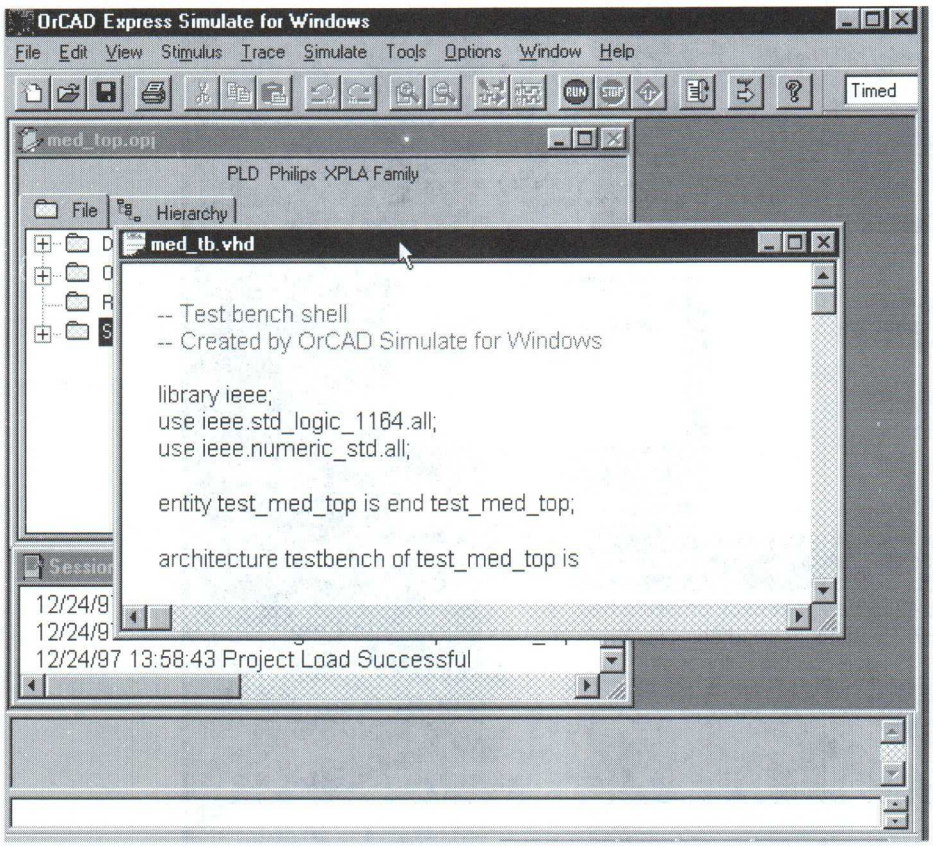


Figure 31. Editing med_tb.vhd

The med_tb.vhd file is displayed and ready for editing. Most of the fields required in a testbench are already filled out, including the signal names from med_top.vho.

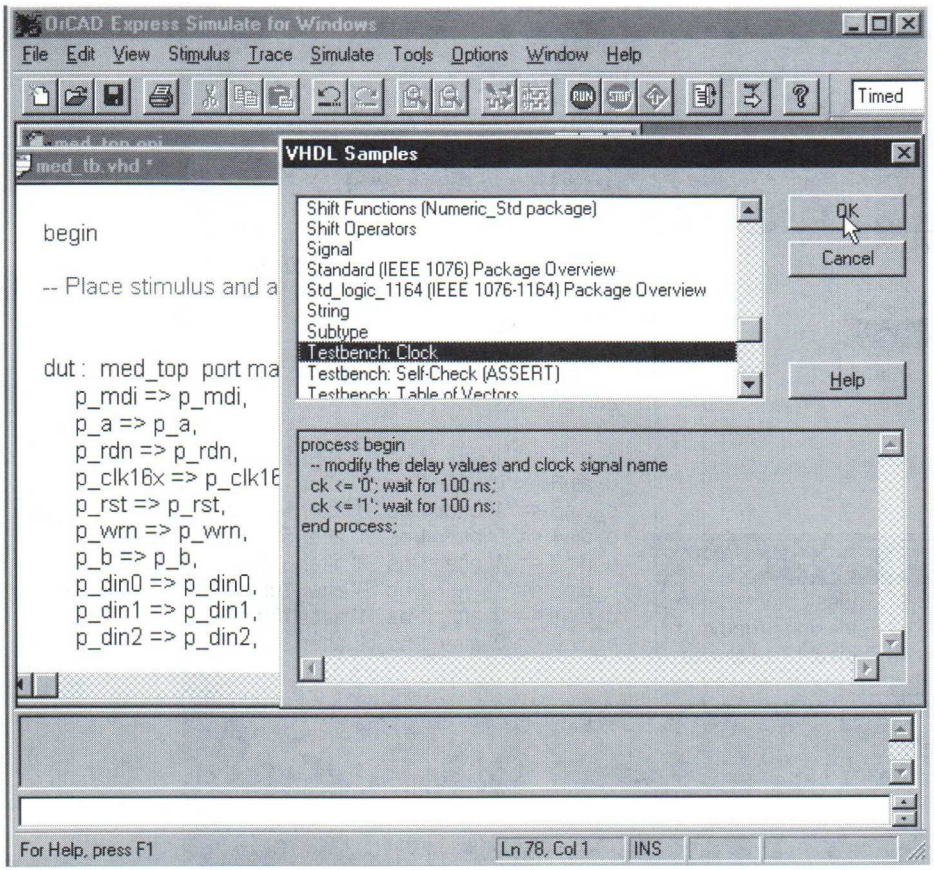


Figure 32. Providing the stimuli in the testbench.

Scroll the testbench to the section used for stimuli. The stimuli for med_top.vhd will be generated manually and using OrCAD templates. From the top toolbar, select **Edit -Samples** and locate the testbench clock in the dialog box.

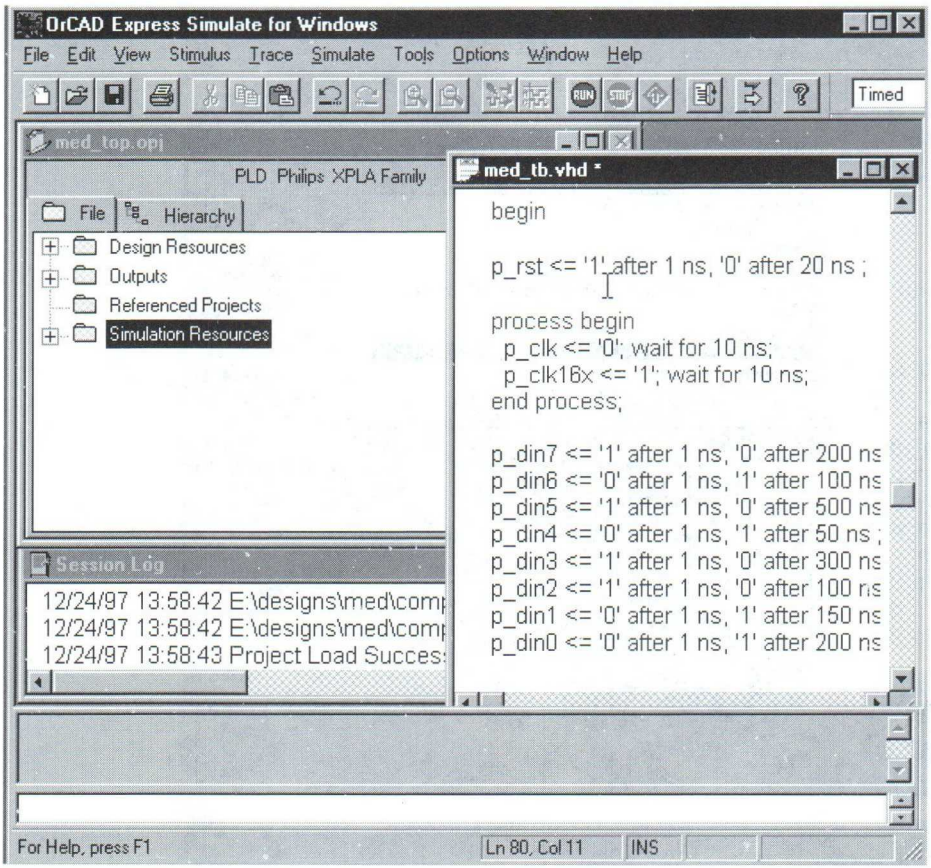


Figure 33. Near complete testbench

Edit the testbench clock to change the clock signal names to p_clk16x and clock period to 20 ns. Add the stimuli for the p_rst and p_din[7:0].

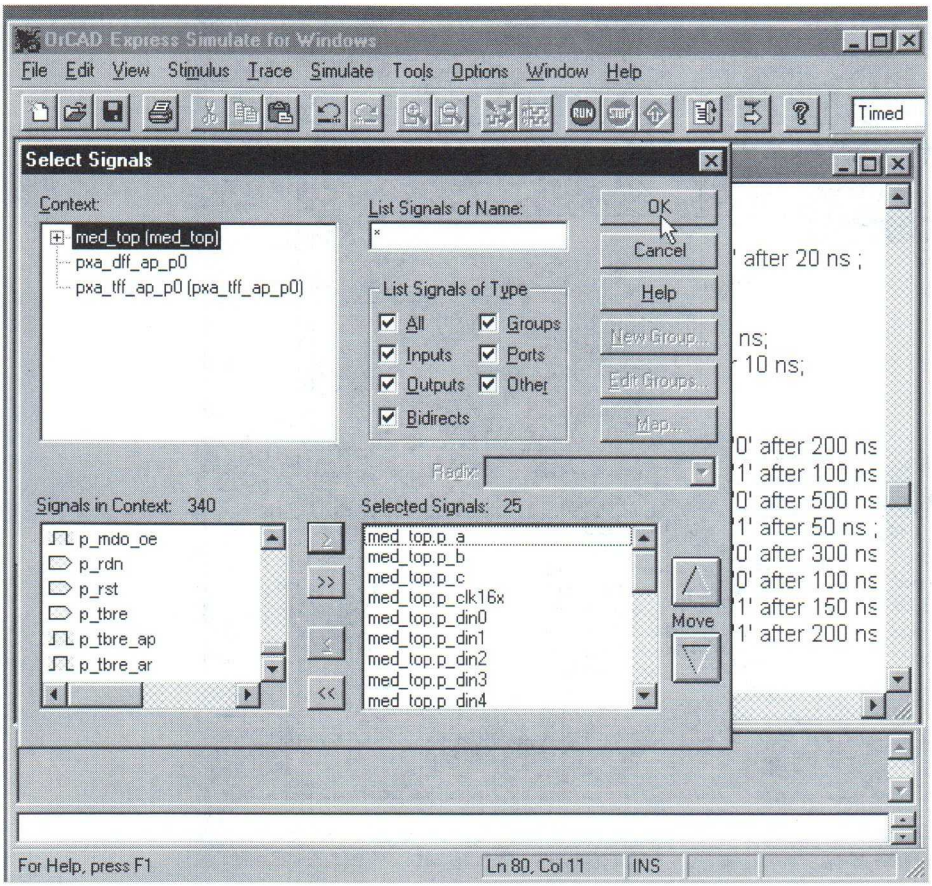


Figure 34. Adding waveform display signals

Trace allows the user to define which signals are to be displayed after a simulation run. Signals are displayed as waveforms and/or in list windows. Select **Trace - New Wave Window** and add the top level signals to be displayed in the waveform display. Use the > icon to move signals to from the left to the right hand window.

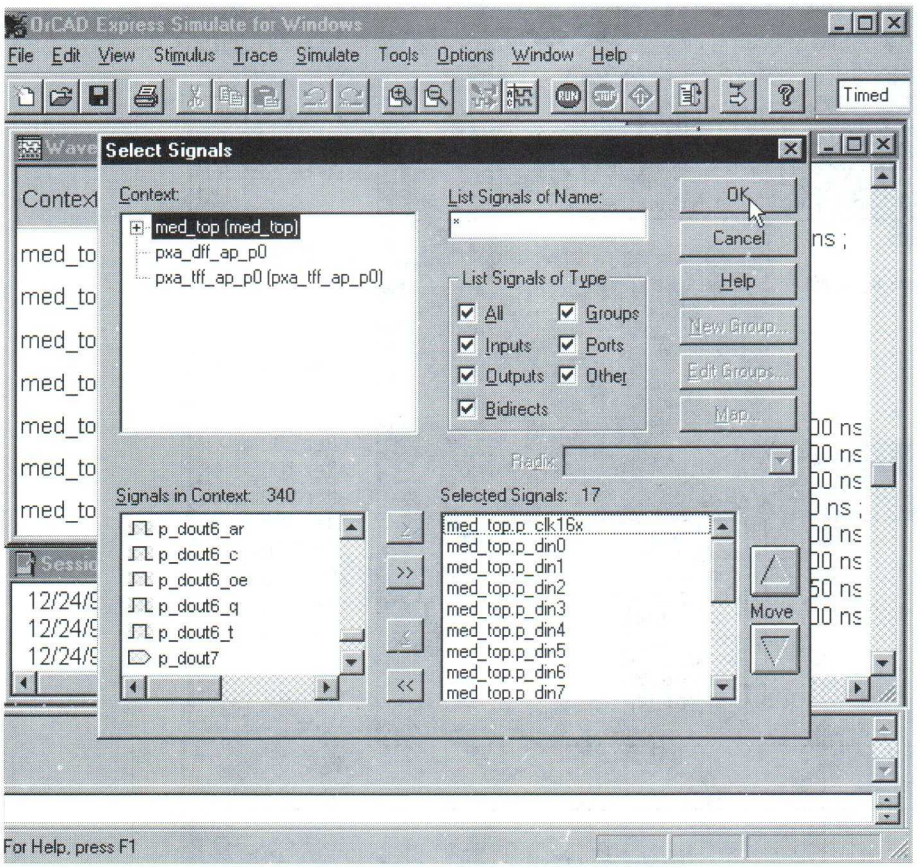


Figure 35. Selecting signals to be displayed in a List window

If desired, select **Trace - List New Window** and repeat the process on the previous page to define the signals displayed in a List window.

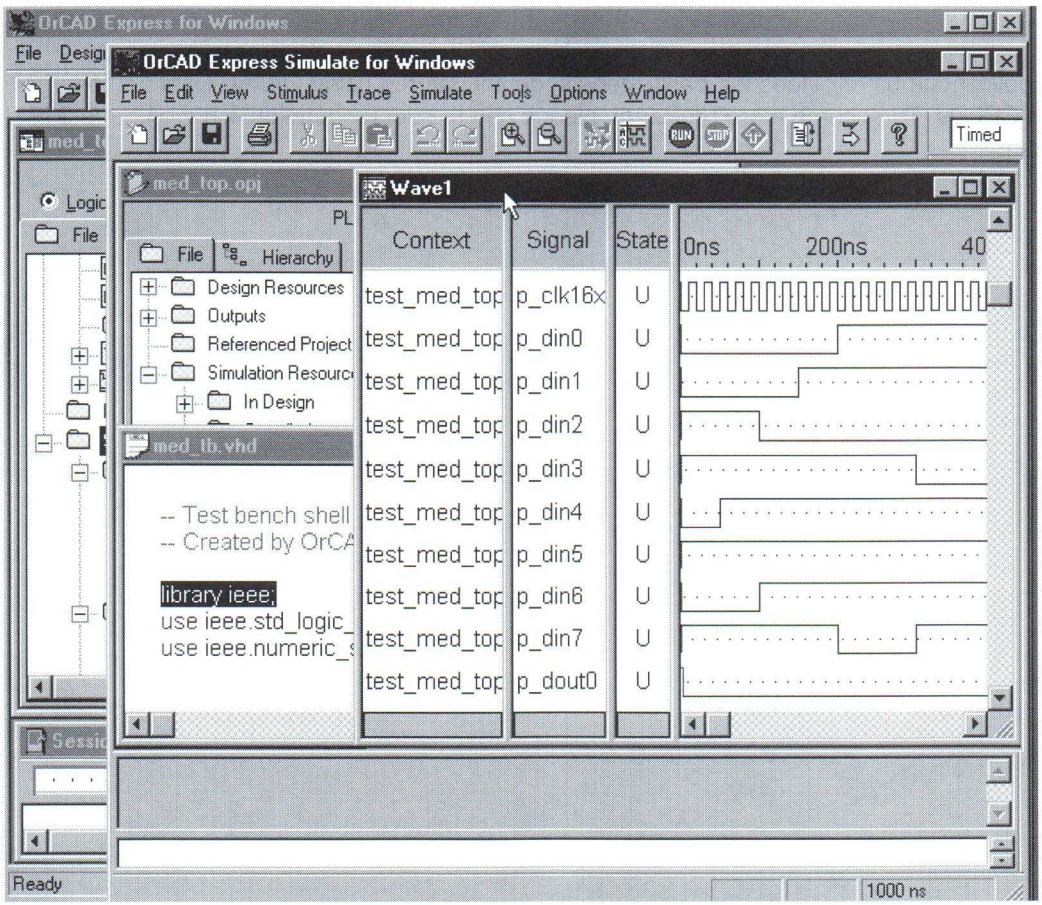


Figure 36. Running the simulation and viewing the results.

From the menu lbar, select **Simulate - Run To** and enter 1000 in the dialog box.

Conclusion

This application note provides the basic steps in getting started using OrCAD Express to target Philips CPLDs. To become more proficient, please see the documentation listed on page 3 of this note.

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.

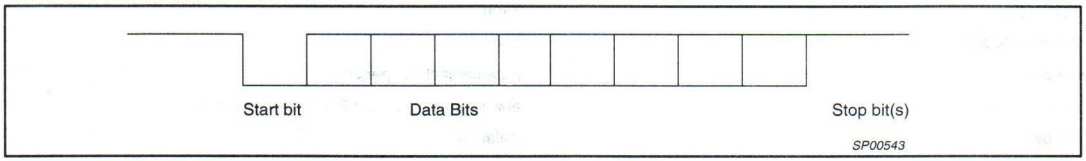


Figure 1. X UART Frame Format

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") ;

$fmonitor(receiver_chann,"T=%t,rst=%b,rxd=%b,rxd1=%b,rxd2=%b,
clk16x=%b,clk1x_enable=%b,clk1x=%b,no_bits_rcvd=%b,rsr=%h,rb
r=%h,data=%h",$t,rst,rxd,receiver.rxd1,receiver.rxd2,clk16x,receiver.
clk1x_enable,receiver.clk1x,receiver.no_bits_rcvd,receiver.rsr,receiv
er.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 synthesis 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.

Model Technology Functional Simulation

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.

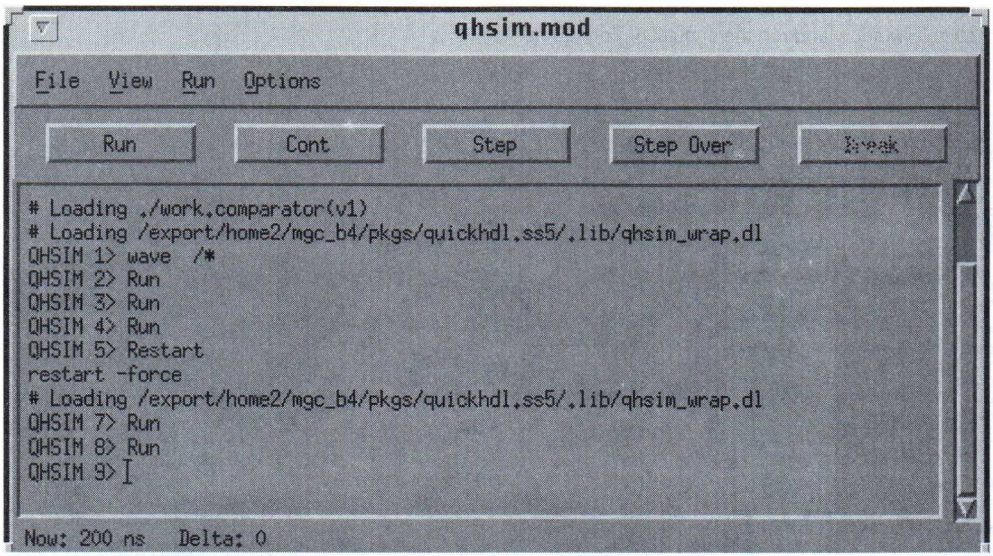
The vhdl testbench is

```
-- 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;
```



To do a funcional simulation,

qvlib work

qvmap work "/export/home/lss/designs/synplicity/vhdl/comparator/work"

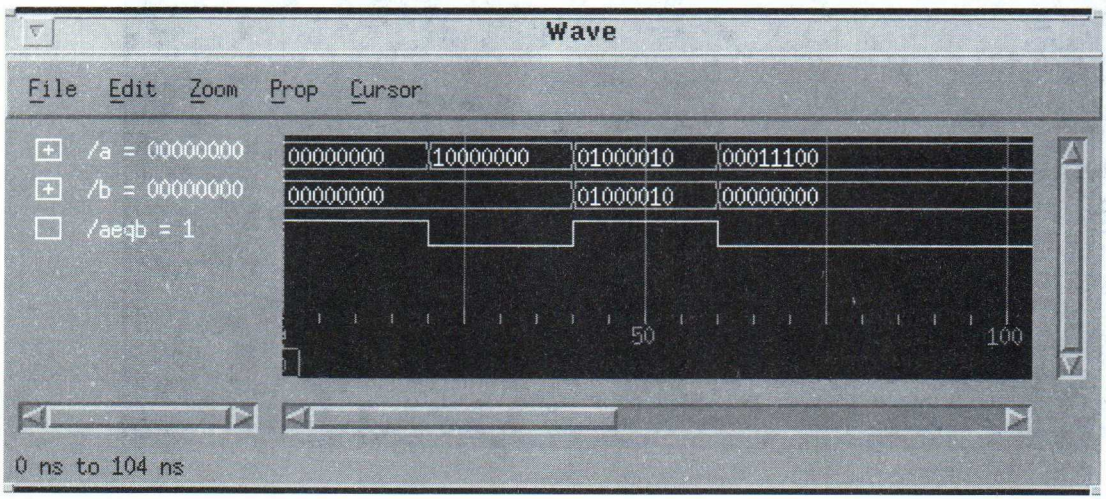
qvcom comparator.vhd

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.

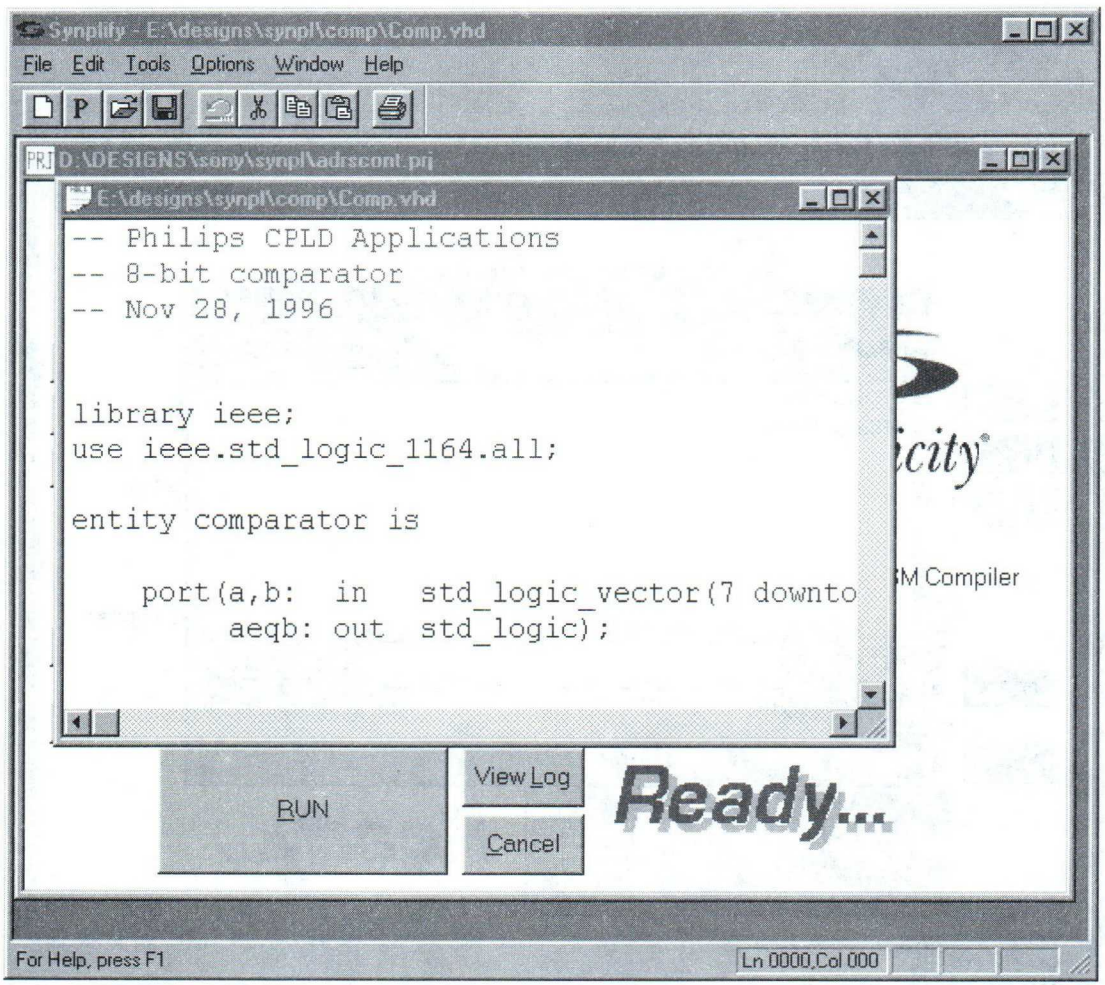


Synthesis

To invoke the Synplicity GUI, enter synplify and select the project.

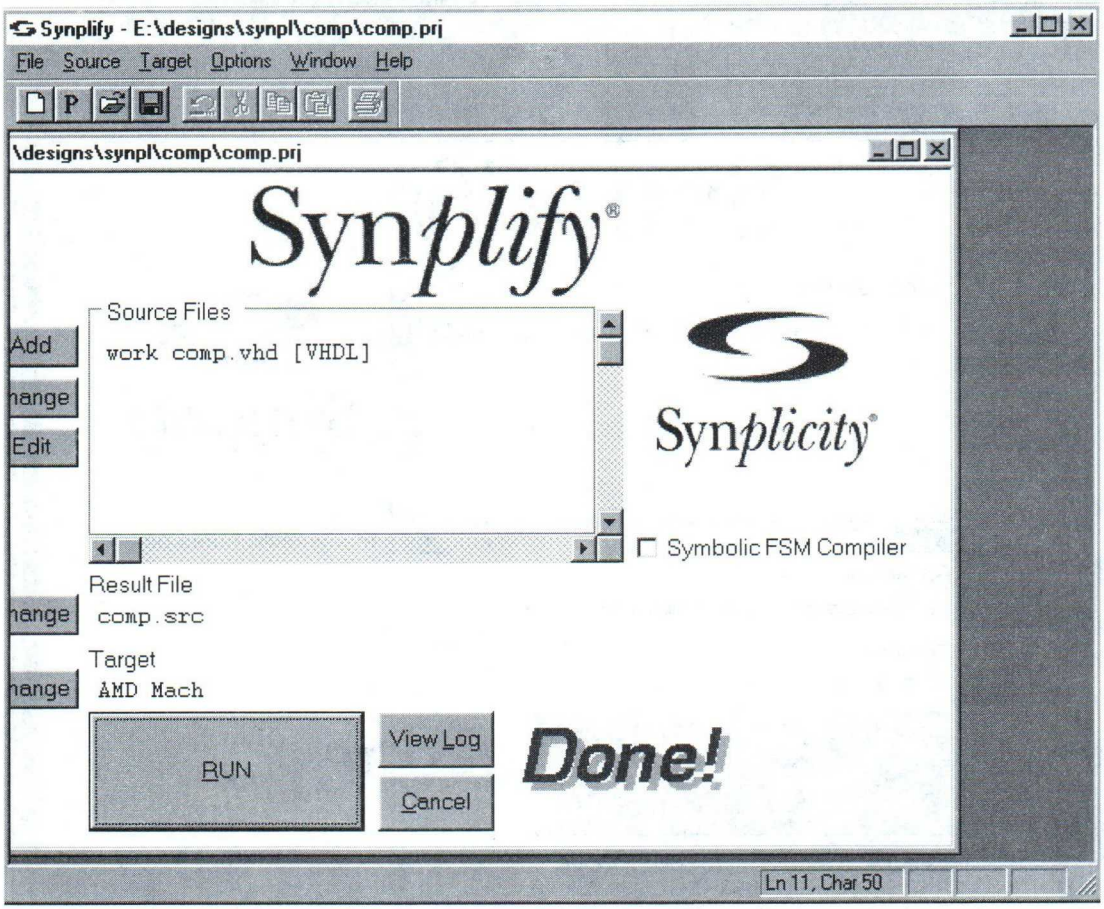


The VHDL source can be viewed from Synpicity.





The targe is selected - currently as AMD, soon as Philips Coolrunner:



Selecting run caues a src file to be created.

Using PLDesigner-XL

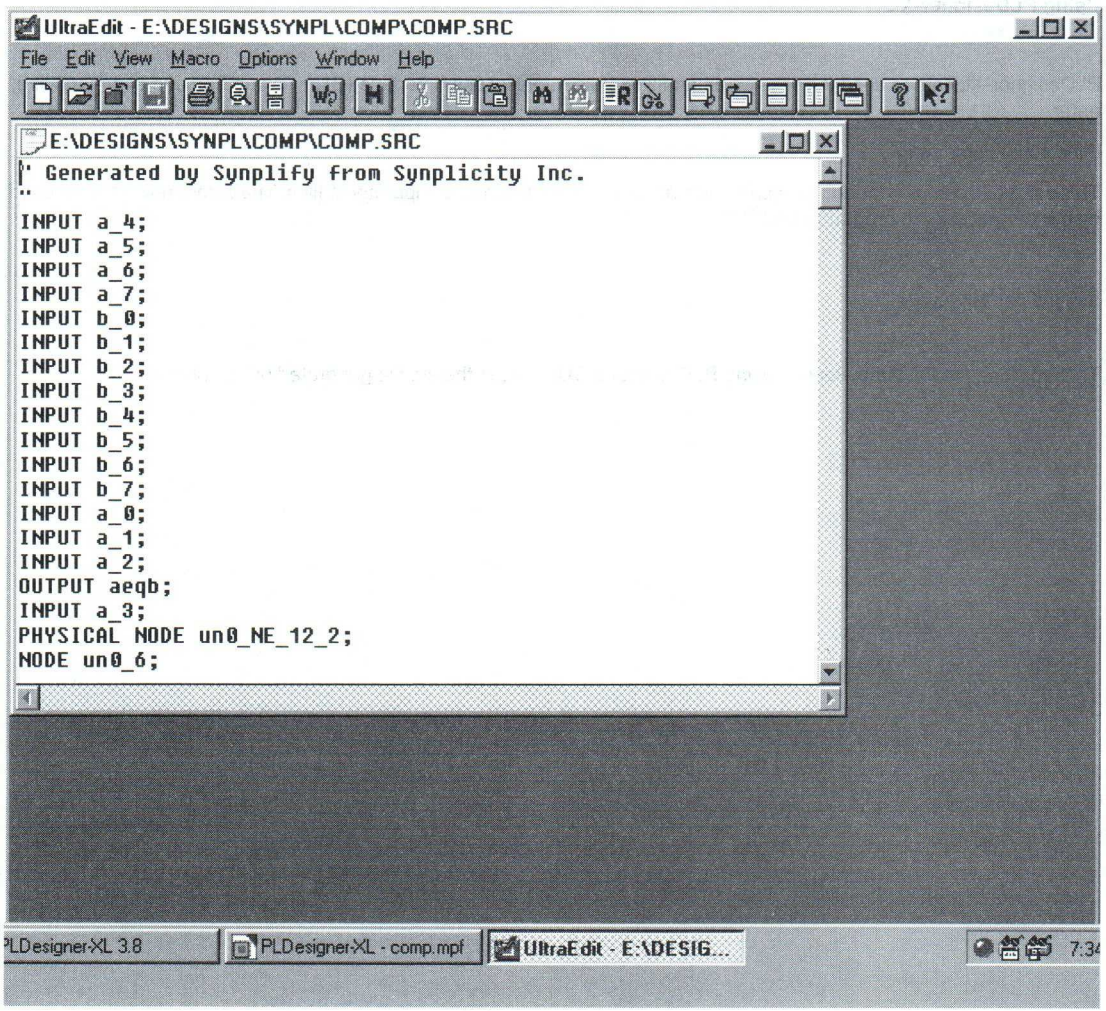
PLDesigner-XL can be used form the command line or its fraphical user interface (GUI).To use from the command line, enter

minc.script comparator

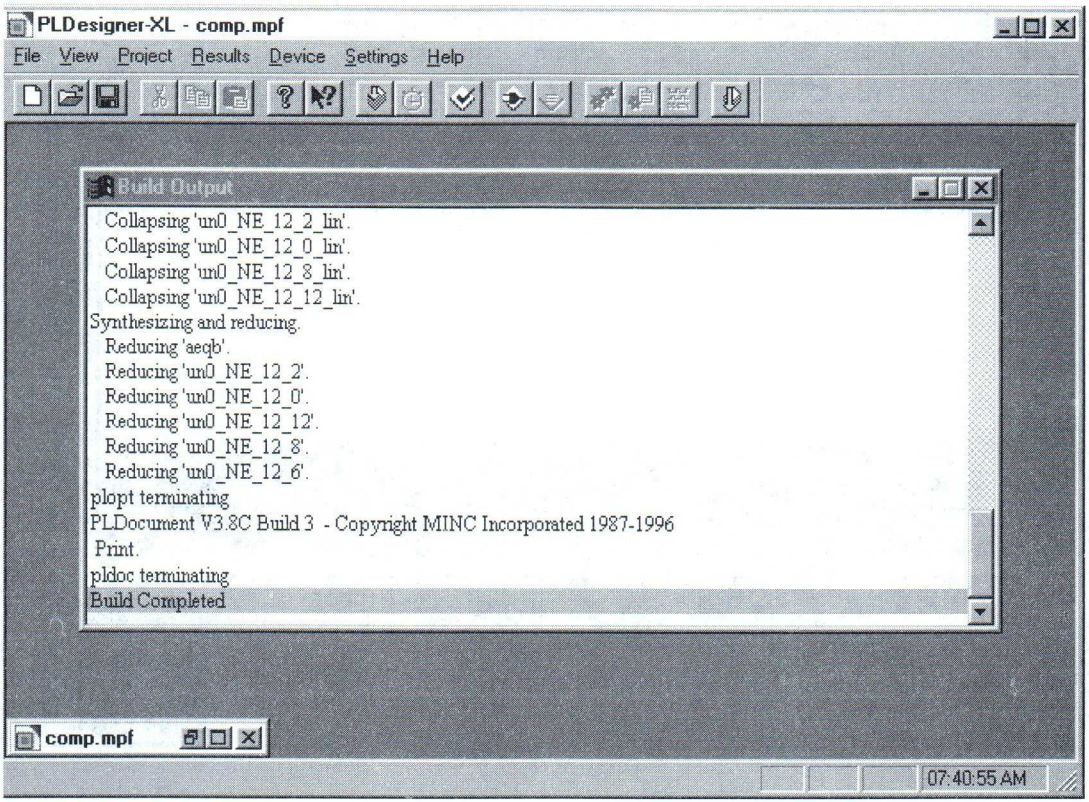
This produces comparator.doc, comparator01.rpt, comparator,npi, and comparator.j1 files. The comparator.j1 file is used by the programmer to program the CPLD.

Graphical User Interface

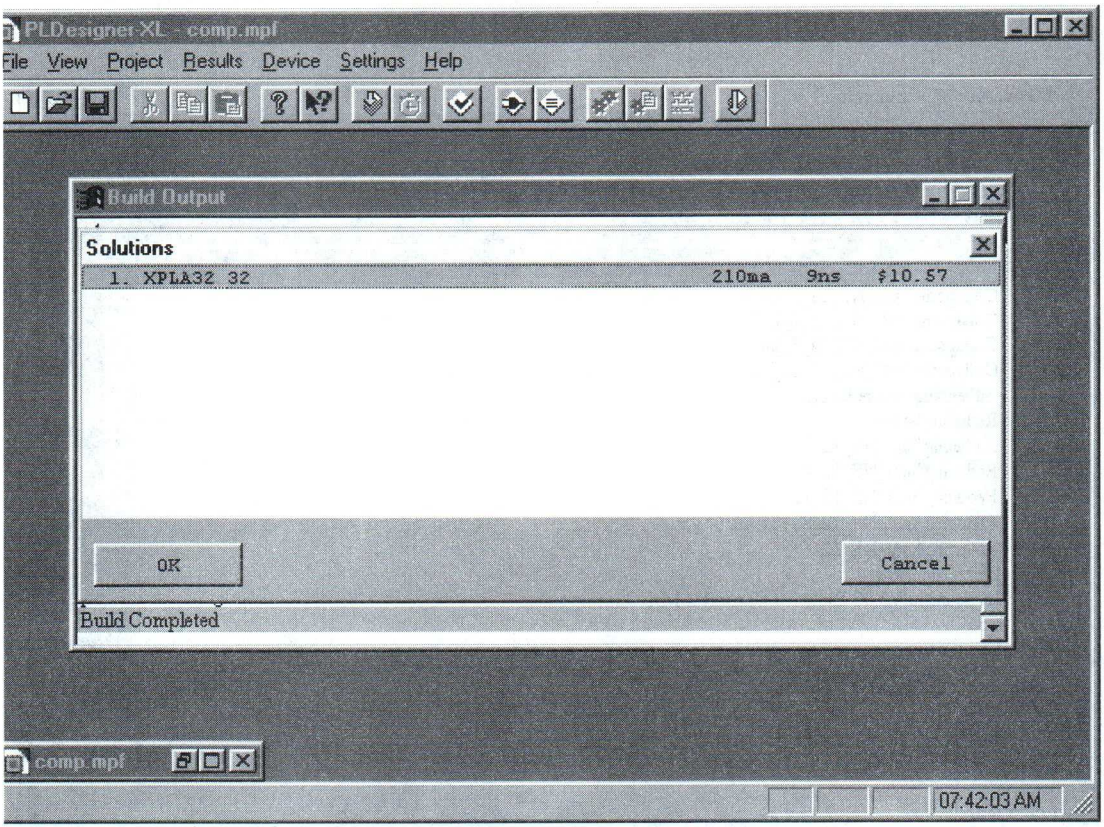
The following provides the steps in using PLDesigner's GUI. It uses the src file generated by Synplicity.



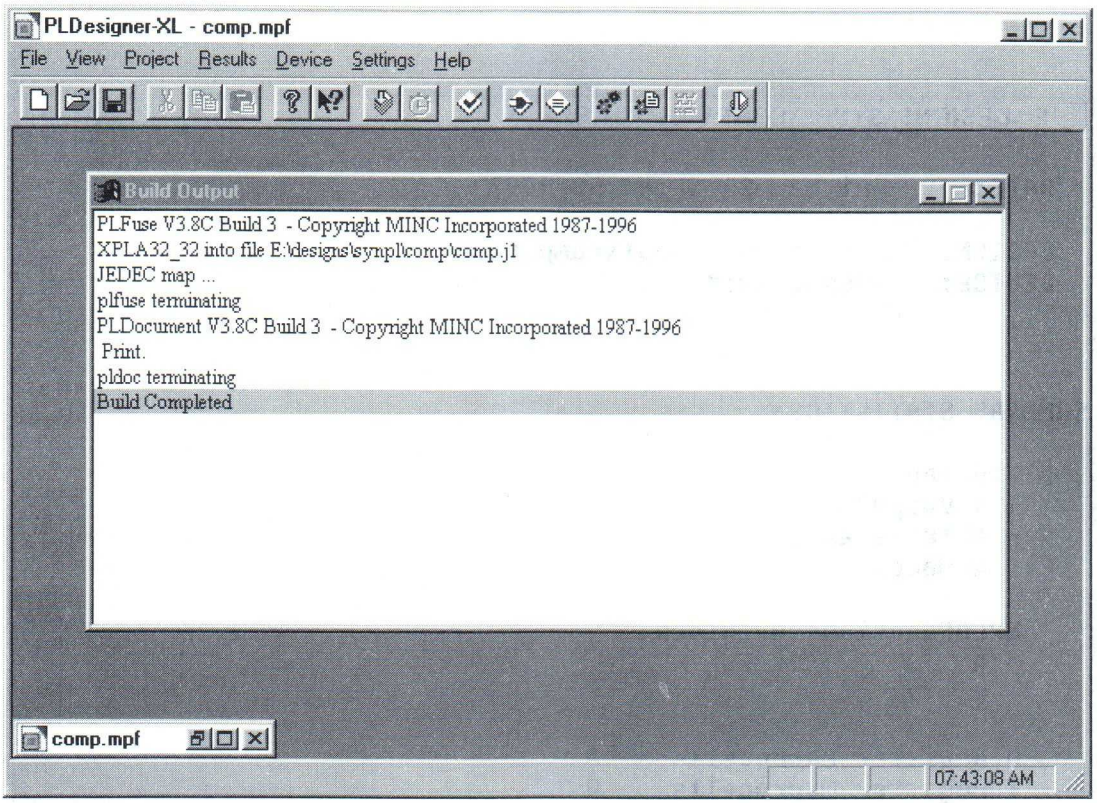
Selecting File Open comp allows the user to view/edit the source file.



The user can Build All or Compile/Partition. This shows the results of the Build All step.



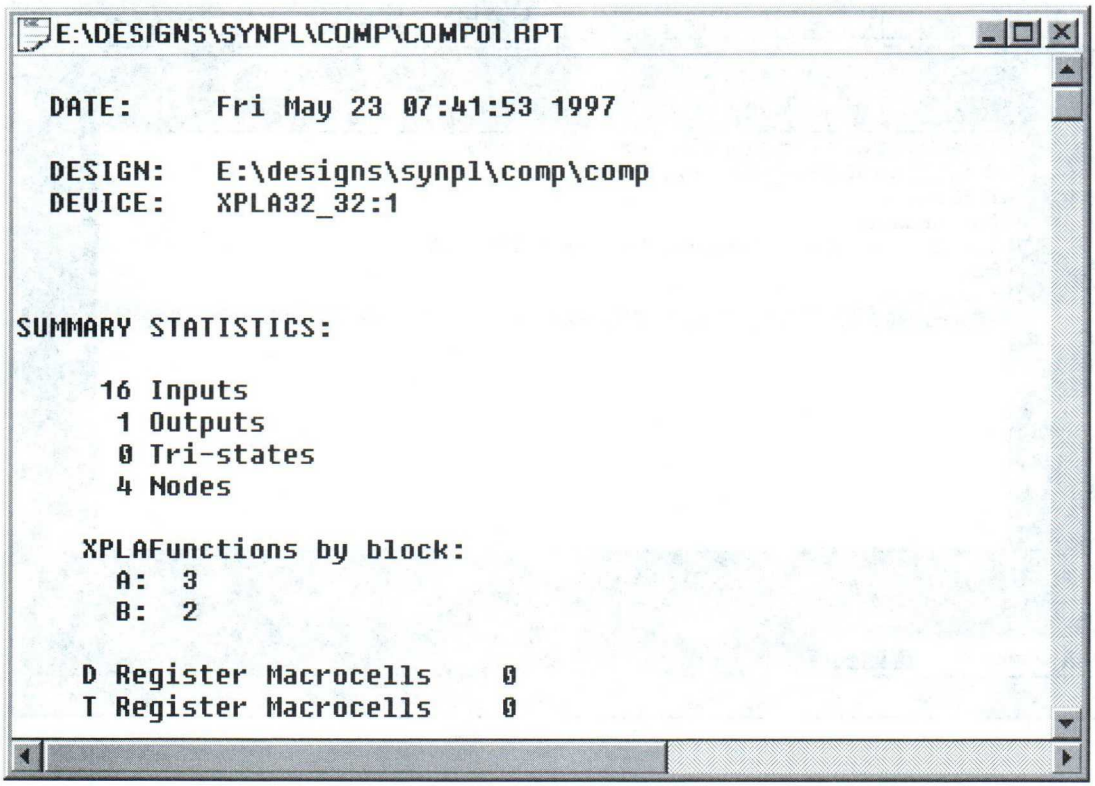
In this case the PZ3032 is a solution. The actual static power dissipation is under 70 microamps.

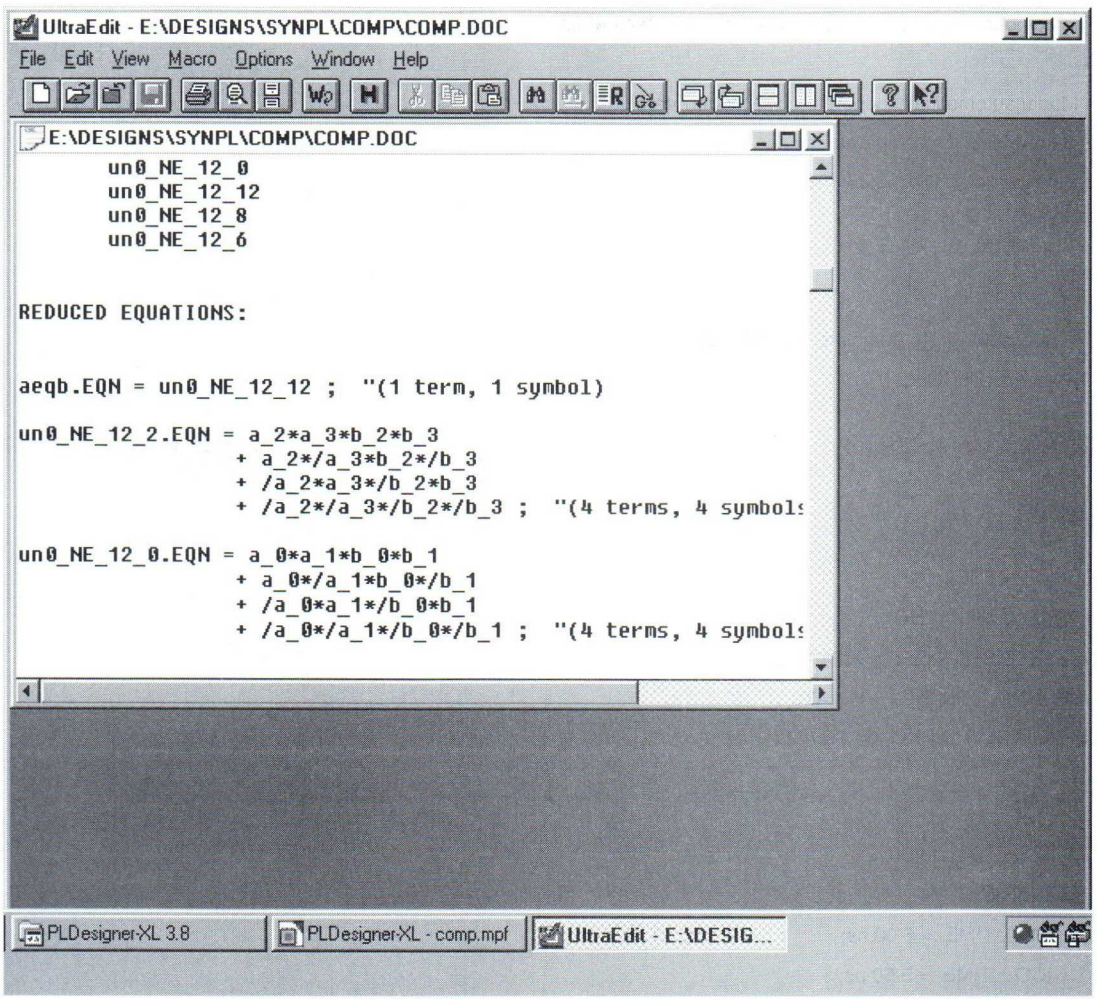


After Build and selecting a solution, generate the fusemap, timing model, and timing report.

```
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.





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.

OrCAD Capture Schematic/PHDL Design Flow for Philips CPLDs

AN074

INTRODUCTION

Philips Semiconductors provides XPLA Designer and libraries for use with OrCAD[1] Capture at no charge. This allows Capture users to target Philips CPLDs as large as 960 macrocells. This note discusses the use of Philips Hardware Description Language (PHDL) with OrCAD Capture.

OrCAD offers OrCAD Capture and OrCAD Express design tools. OrCAD Express is OrCAD's high end tool offers the seamless integration of VHDL/schematic entry, and VHDL simulation, as well as the library and back end tools needed to target Philips CPLDs. OrCAD Capture is a schematic entry design tool. State machines and operations such as adders, comparators, counters, are easier to enter using a hardware description language than with a schematic. This note provides OrCAD Capture users with a method for obtaining access to a HDL and fitter at no cost.

Philips Semiconductor has developed a family of advanced 3-volt and 5-volt complex programmable logic devices (CPLDs). The XPLA 1 series, designated as the PZ5000 - (5-volt) and PZ3000 (3-volt) series devices, is footprint compatible with the Altera 7000 series devices. The XPLA 2 series consist of the PZ3960 and PZ3320[2]. The principle advantage of Philips CPLDs over all existing CPLDs is that they do not consume static power. The other advantages are 25% higher combinatorial logic capacity and an increased ability to fit designs with fixed pinouts. The XPLA 1 family is in-system programmable and programmable on Data I/O and BP Microsystems programmers. The XPLA 2 family uses SRAM to store its configuration memory, and is configured similarly to existing FPGAs.

The following documentation is related to this note.

OrCAD Capture for Windows User's Guide

Complex Programmable Logic Devices Data Book IC27 (XPLA 1 only)

PZ3960 Data Sheet

XPLA Designer User's Guide

PHDL Models of Commonly Used Digital Functions

Technical support is available from

coolpld@abq.sc.philips.com

(888) coolpld

http://www.coolpld.com

ftp://www.coolpld.com

DESIGN FLOW

XPLA Designer and the OrCAD libraries can be obtained from Philips website www.coolpld.com and are available on cdrom. The xpla\lib\orcad directory contains the following OrCAD libraries.

ps.olb - primitive library elements

ps_ttl.olb - 74xx library elements

(1) Philips recognizes the trademarks of OrCAD, Inc., including OrCAD Capture and OrCAD Express.
(2) The PZ3320 is scheduled to sample in Q3 1998.

The symbols in the ps.olb library are:

AND2 - AND12	AND2B1	AND3B2	AND3B1	AND3B2
AND3B3	AND4B1	AND4B2	AND4B3	AND4B4
AND5B1	AND5B2	AND5B3	AND5B4	AND5B5
OR2 - OR12	OR2B1	OR2B2	OR3B1	OR3B2
OR3B3	OR4B1	OR4B2	OR4B3	OR4B4
OR5B1	OR5B2	OR5B3	OR5B4	OR5B5
ND2 - ND12	ND2B1	ND2B2	ND3B1	ND3B2
ND3B3	ND4B1	ND4B2	ND4B3	ND4B4
ND5B1	ND5B2	ND5B3	ND5B4	ND5B5
NR2 - NR12	NR2B1	NR2B2	NR3B1	NR3B2
NR3B3	NR4B1	NR4B2	NR4B3	NR4B4
NR5B1	NR5B2	NR5B3	NR5B4	NR5B5
XR2 -XR8	XNR2 - XNR8	DFF	DFFR	DFFS
DFFE	DFFRE	DFFSE	TFF	TFFR
TFFS	TFFE	TFFRE	TFFSE	JKFF
JKFFR	JKFFS	SRFF		

The symbols in the ps_ttl.olb library are:

PS7400	Quad 2-input NAND
PS7402	Quad 2-input NOR
PS7403	Quad 2-input NAND
PS7404	Hex Inverter
PS7408	Quad 2-input AND
PS7410	Triple 3-input NAND
PS7411	Triple 3-input AND
PS7414	Hex Inverting schmitt trigger
PS74138	3 to 8 line decoder/demultiplexer; inverting
PS74139	Dual 2 to 4 line decoder/demultiplexer
PS74151	8-input multiplexer
PS74153	Dual 4-input multiplexer

PS74154	4 to 16 line decoder/demultiplexer
PS74157	Quad 2-input data selector/multiplexer, non-inverting
PS74161	Presettable 4-bit binary counter, asynchronous reset
PS74162	Presettable synchronous BCD decade counter, synchrouous reset
PS74163	Presettable 4-bit binary counter, synchronous reset
PS74164	8-bit serial in, parallel out shift register
PS74166	8-bit parallel in, serial out shift register
PS74174	Hex D-type flip-flop with reset, positive edge trigger
PS74181	4-bit arithmetic logic unit
PS74190	Presettable synchrouous BCD decade up/down counter
PS74191	Presettable synchronous 4-bit binary up/down counter
PS7420	Dual 4-input NAND
PS7421	Dual 4-input AND
PS74244	Octal Line Driver, 3-state, output enable active low
PS74245	Octal Bus transceiver, 3-state
PS74251	8-input multiplexer, 3-state
PS74253	Dual 4-input multiplexer, 3-state
PS74257	Quad 3-input multiplexer, 3-state
PS74266	Quad 2-inupt EXCLUSIVE NOR
PS7427	Triple 3-input NOR
PS74280	9-bit odd/even parity generator/checker
PS74283	4-bit binary full adder with fast carry
PS74299	8-bit universal shift register, 3-state
PS7430	8-input NAND gate
PS7432	Quad 2-input OR
PS74373	Octal D type transparent latch, 3-state
PS744040	12-stage binary ripple counter
PS744511	BCD to 7 segment latch/decoder/driver
PS74573	Octal D type transparent latch, 3-state, bus oriented pinout
PS74574	Octal D type flip flop, positive edge rigger, 3-state, bus oriented pinout
PS74583	4-bit BCD full adder with fast carry
PS74594	8-bit shift register with output register
PS74595	8-bit serial-in/serial or parallel out shift register with output latches, 3-state
PS74597	8-bit shift register with input flip flops
PS74640	Octal Bus transceiver, 3-state, inverting

PS74688	8-bit magnitude comparator
PS744080	16-bit even/odd parity generator/checker
PS7474	Dual D type flip-flop with set and reset, positive edge trigger
PS7483	4-bit full adder
PS7485	4-bit magnitude comparator
PS7486	Quad 2-input EXCLUSIVE OR

This note defines how to use Philips Hardware Description Language (PHDL) with OrCAD Capture. PHDL is a language whose syntax is virtually identical to Abel. Some reasons to mix PHDL and schematics in OrCAD Capture are:

1. State machines are generally easier to design and debug using a HDL.

2. Design entry is easier for functions such as the +,-,<,>, <=,>= operators, and speed/area trade-offs are easier using PHDL than with schematics.

3. Define functionality of library elements in ps_ls.olb library.

Some examples of PHDL are given in the \xpla\example directory. Additional examples are given in PHDL Models of Commonly Used Digital Functions, available from the web site. PHDL syntax is given in XPLA Designer User's Manual.

To use PHDL with schematic capture, do the following:

1. **Open Project** and schematic in OrCAD Capture.

2. On a schematic page, **Place Hierarchical Block** to define a symbol whose functionality is to be defined by PHDL. Add hierarchical pins to the block whose names match those defined as inputs, outputs, and inouts in the <module>.phd file.

3. Save <sc_design>.dsn.

4. **Tools - Update References**

5. **Tools - DRC**

6. **Tools - Create Netlist** - edif200 - Map names - write <sc_design>.edf

7. Create <module>.phd. From Dos, run **xplaopt** <module>.phd to generate <module>.blf

8. cd \xpla (xpla install directory). Add <module>.blf to xplalib.ord and delete the checksum (last line in xplalib.ord).

9. From Dos, enter **xplaopt** -lib xplalib.ord -chksum

10. cd <proj_dir>. Invoke XPLA Designer, **Open Design**, selecting Files of Type edif. Compile and Fit.

The following shows the steps for a mixed schematic/PHDL design entry of three instances of prep3, a state machine which would be very complex to enter using schematics. The p3.phd code is given below. Although provided below, test vectors are not supported in XPLA Designer.

module p3

"Prep3 - 3 instances

"Small state machine - 8 inputs, 8 registered outputs

```
"Inputs
    clk,rst,i7,i6,i5,i4,i3,i2,i1,i0   pin;
"Outputs
        o7,o6,o5,o4,o3,o2,o1,o0 pin istype 'buffer,reg_d';
B7,B6,B5,B4,B3,B2,B1,B0 node istype 'buffer,reg_d';
C7,C6,C5,C4,C3,C2,C1,C0 node istype 'buffer,reg_d';
        Q,QB,QC node istype 'buffer,reg_d';

"State Register assignment
OUT= [B7,B6,B5,B4,B3,B2,B1,B0];
inp    = [i7,i6,i5,i4,i3,i2,i1,i0];
inp1   = [B7.Q,B6.Q,B5.Q,B4.Q,B3.Q,B2.Q,B1.Q,B0.Q];
inp2   = [C7.Q,C6.Q,C5.Q,C4.Q,C3.Q,C2.Q,C1.Q,C0.Q];
sreg   = [o7,o6,o5,o4,o3,o2,o1,o0, Q];
sreg1  = [B7,B6,B5,B4,B3,B2,B1,B0,QB];
sreg2  = [C7,C6,C5,C4,C3,C2,C1,C0,QC];
        START  = [ 0, 0, 0, 0, 0, 0, 0, 0, 0]; "00
        SA     = [ 1, 0, 0, 0, 0, 0, 1, 0, 0]; "82
        SA1    = [ 0, 0, 0, 0, 0, 1, 0, 0, 0]; "04
        SB     = [ 0, 0, 1, 0, 0, 0, 0, 0, 0]; "20
        SC     = [ 0, 1, 0, 0, 0, 0, 0, 0, 1]; "40
        SD     = [ 0, 0, 0, 0, 1, 0, 0, 0, 0]; "08
        SE     = [ 0, 0, 0, 1, 0, 0, 0, 1, 0]; "11
        SF     = [ 0, 0, 1, 1, 0, 0, 0, 0, 0]; "30
        SG     = [ 0, 0, 0, 0, 0, 0, 1, 0, 0]; "02
        SG1    = [ 1, 0, 0, 0, 0, 0, 0, 0, 0]; "80
        START1 = [ 0, 1, 0, 0, 0, 0, 0, 0, 0]; "40
        START2 = [ 0, 0, 0, 0, 0, 0, 0, 1, 0]; "01
        H,L,C,X = 1,0,.C.,.X.;
Equations
        sreg.ar  = !rst;
        sreg.clk = clk;
        sreg1.ar  = !rst;
        sreg1.clk = clk ;
        sreg2.ar  = !rst;
```

```
    sreg2.clk  = clk ;
state_diagram sreg1"first instance
state START:
    if (inp == ^h3C) then SA
    else START;
state START1:
    if (inp == ^h3C) then SA
    else START;
state START2:
    if (inp == ^h3C) then SA
    else START;
state SA:
    if (inp == ^h1F) then SB
    else if (inp == ^h2A) then SC
    else SA1;
state SA1:
    if (inp == ^h1F) then SB
    else if (inp == ^h2A) then SC
    else SA1;
state SB:
    if (inp == ^hAA) then SE
    else SF;
state SC:
    goto SD;
state SD:
    goto SG1;
state SE:
    goto START1;
state SF:
    goto SG;
state SG:
    goto START2;
state SG1:
    goto START2;
state_diagram sreg2"second instance
```

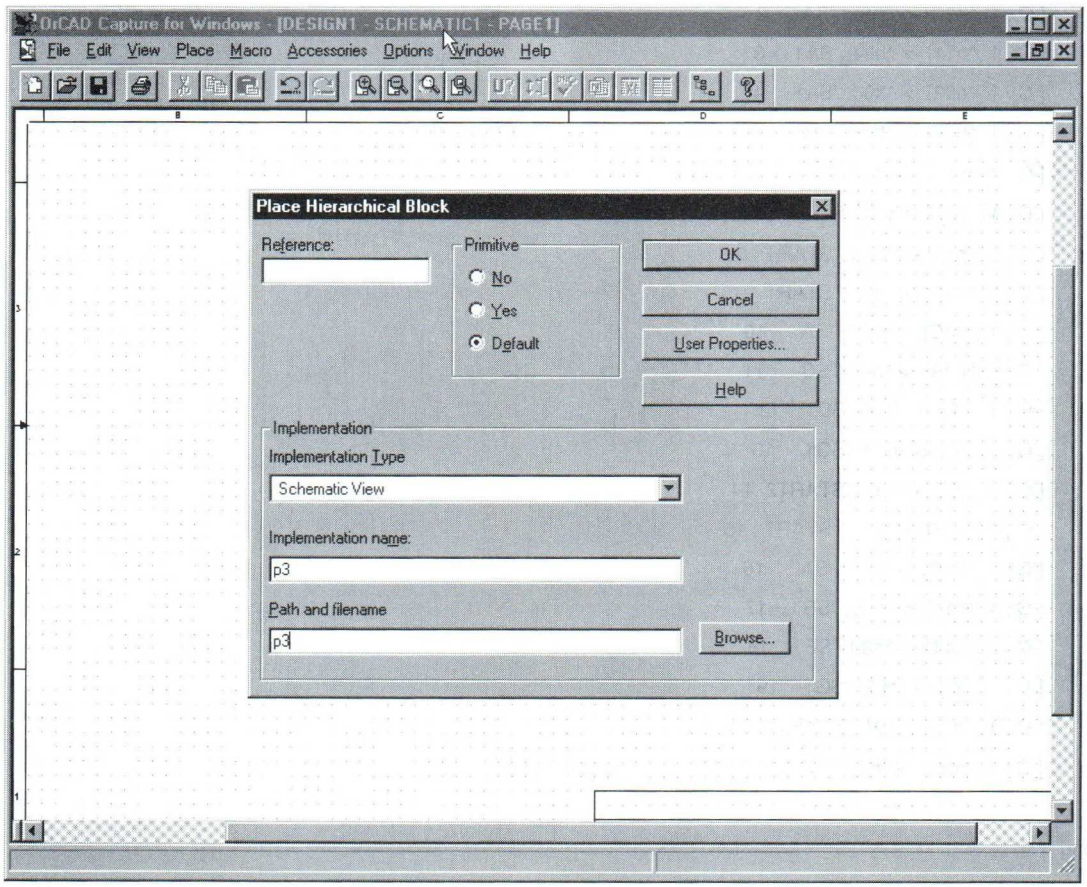
```
state START:
    if (inp1 == ^h3C) then SA
    else START;
state START1:
    if (inp1 == ^h3C) then SA
    else START;
state START2:
    if (inp1 == ^h3C) then SA
    else START;

state SA:
    if (inp1 == ^h1F) then SB
    else if (inp1 == ^h2A) then SC
    else SA1;
state SA1:
    if (inp1 == ^h1F) then SB
    else if (inp1 == ^h2A) then SC
    else SA1;
state SB:
    if (inp1 == ^hAA) then SE
    else SF;
state SC:
    goto SD;
state SD:
    goto SG1;
state SE:
    goto START1;
state SF:
    goto SG;
state SG:
    goto START2;
state SG1:
    goto START2;
state_diagram sreg"third instance
state START:
```

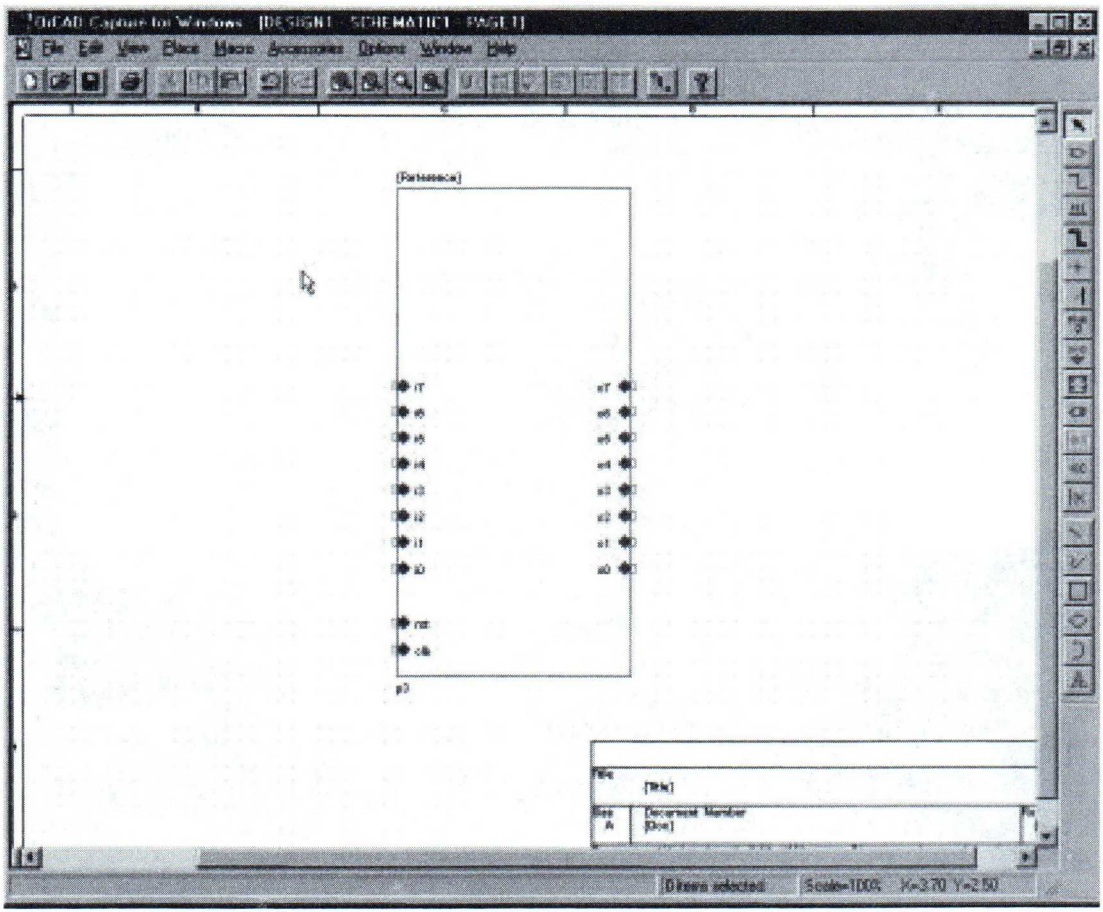
```
        if (inp2 == ^h3C) then SA
        else START;
state START1:
        if (inp2 == ^h3C) then SA
        else START;
state START2:
        if (inp2 == ^h3C) then SA
        else START;


state SA:
        if (inp2 == ^h1F) then SB
        else if (inp2 == ^h2A) then SC
        else SA1;
state SA1:
        if (inp2 == ^h1F) then SB
        else if (inp2 == ^h2A) then SC
        else SA1;
state SB:
        if (inp2 == ^hAA) then SE
        else SF;
state SC:
        goto SD;
state SD:
        goto SG1;
state SE:
        goto START1;
state SF:
        goto SG;
state SG:
        goto START2;
state SG1:
        goto START2;
/*
test_vectors
    ([,rst, inp] -> OUT)
```

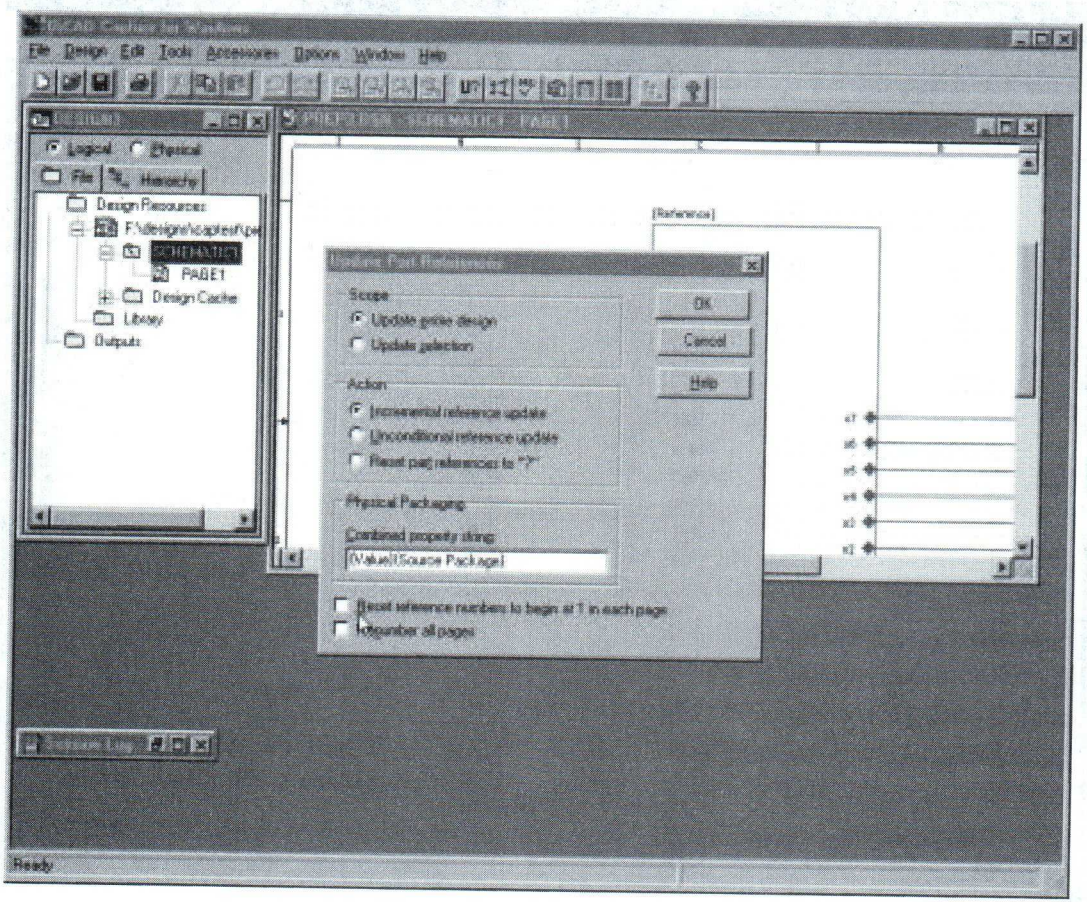
```
    [ 0 , 0 ,  X ] -> ^h00; "START   1
    [ C , 1 ,^h3C] -> ^h82; "SA      2
    [ C , 1 ,^h11] -> ^h04; "SA1     3
    [ C , 1 ,^h22] -> ^h04; "SA1     4
    [ C , 1 ,^h1F] -> ^h20; "SB      5
    [ C , 1 ,^hAA] -> ^h11; "SE      6
    [ C , 1 ,  X ] -> ^h40; "START1  7
    [ C , 1 ,^h11] -> ^h00; "START   8
    [ C , 1 ,^h22] -> ^h00; "START   9
    [ C , 1 ,^h3C] -> ^h82; "SA      10
    [ C , 1 ,^h2A] -> ^h40; "SC      11
    [ C , 1 ,  X ] -> ^h08; "SD      12
    [ C , 1 ,  X ] -> ^h80; "SG1     13
    [ C , 1 ,  X ] -> ^h01; "START2  14
    [ C , 1 ,^h55] -> ^h00; "START   15
    [ C , 1 ,^h3C] -> ^h82; "SA      16
    [ C , 1 ,^h1F] -> ^h20; "SB      17
    [ C , 1 ,^h11] -> ^h30; "SF      18
    [ C , 1 ,  X ] -> ^h02; "SG      19
    [ C , 1 ,  X ] -> ^h01; "START2  20
    [ C , 1 ,^h3C] -> ^h82; "SA      21
 */
end
```



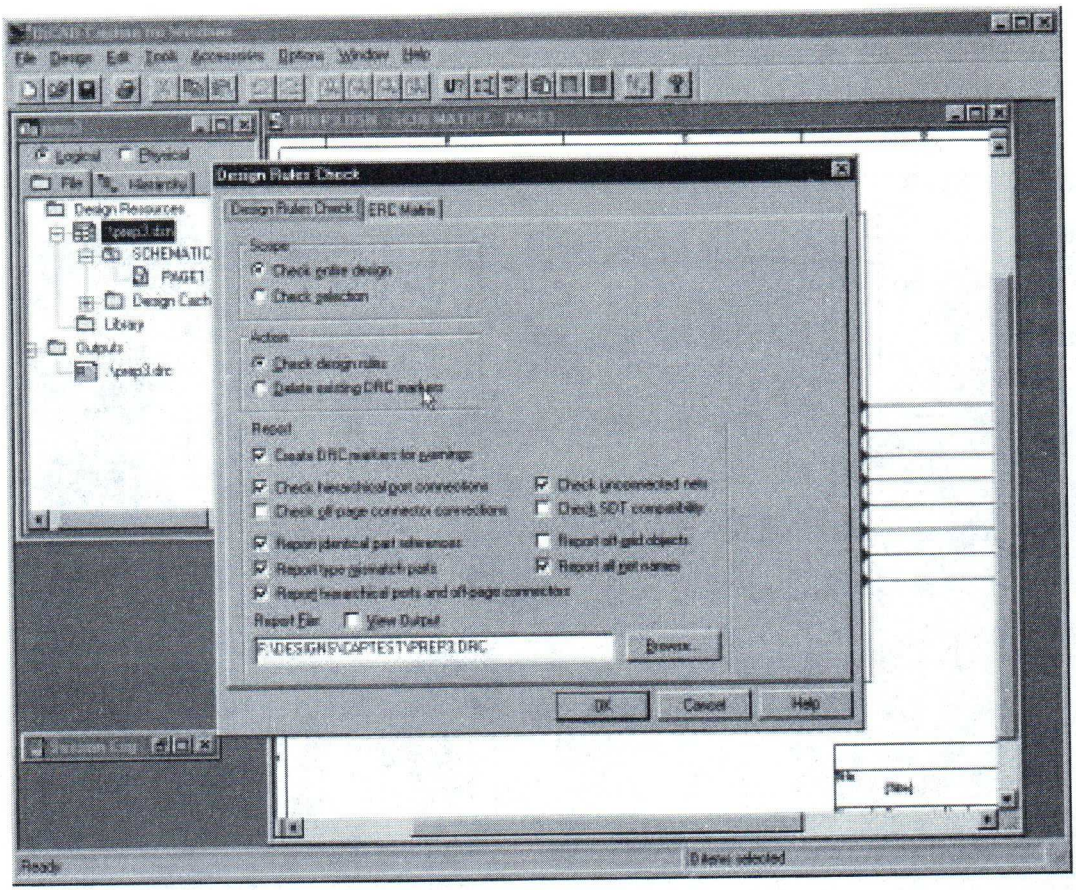
Create a project in Capture and open a design named prep3. Place a hierarchical block named p3.



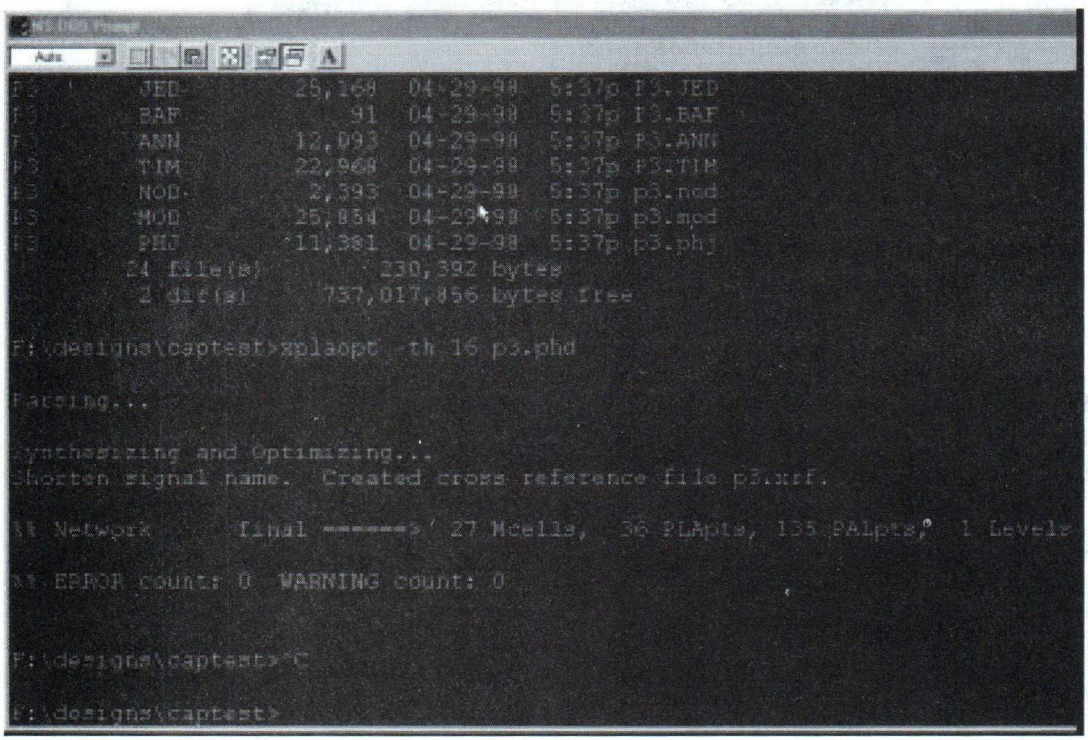
After placing symbol p3, place the hierarchical pins. The case of the pin names must match the case of the ports in the phdl file.



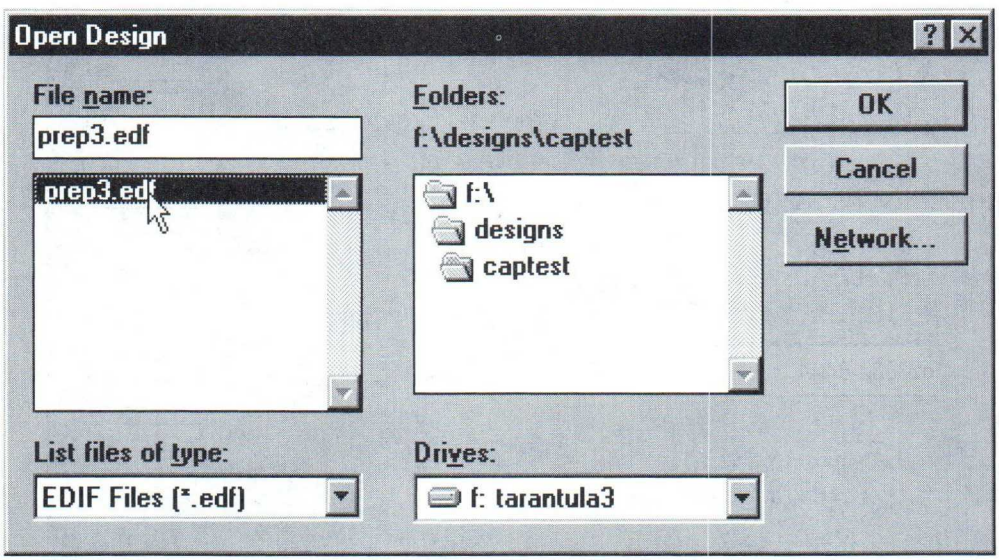
Use **Tools - Update Part References** to update the reference.



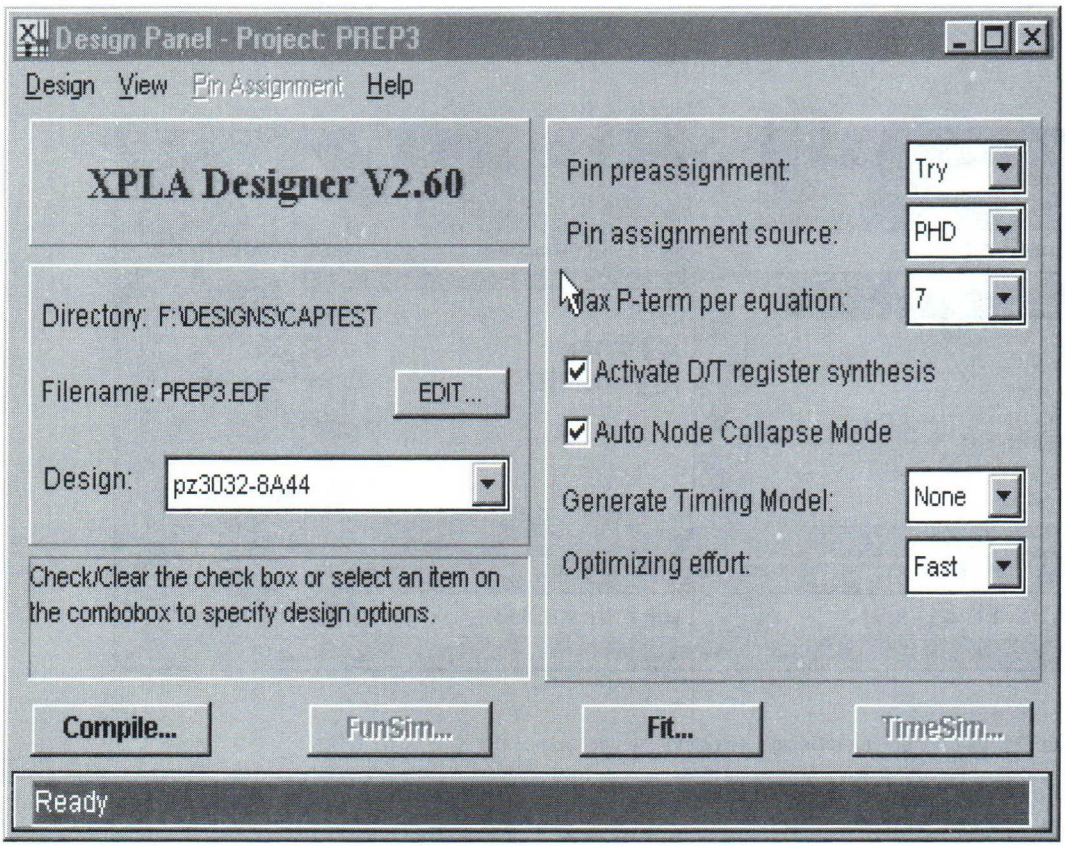
Run **Design Rule Check** with the options shown.



At the Dos prompt, run **xplaopt** -th 16 p3.phd. If the number of macrocells, PAL terms, or PLA terms are not optimal, vary the parameter th from 5 to 37. Edit xplalib.ord as described earlier: delete checksum, add p3.blf, and re-create checksum.



Invoke XPLA Designer, Open Design, selecting Files of Type EDIF.



Compile and fit the design.

The graphical pin assignment editor in XPLA Designer currently does not support the PZ3960 or P3Z22V10. A <design>.ctl file is used to define pinout. Then the design is compiled from the Dos prompt with

xplaopt -ctrl <design>.ctl

To specify pin assignments in these devices. An example <design>.ctl is given below.

```
[command]
-it phdl
-i usb.phd
-reg
-dev pz3960-12-qfp960
-xor exp
-pre keep
[pin_assignment]
<signal>:<pin_number>
```

where <signal>:<pin_number> is defined for all pins whose location needs to be defined.

XPLA Designer includes a functional and timing simulator. To get started using the timing simulator, compile and fit demo.phd from the \xpla\example directory (w/o schematic). Click on TimSim in the XPLA Designer. The demo example provides stimuli input. Click **Run** to run a simulation.

To provide test stimuli, do the following to edit the input waveforms.

Enter **Change Value**. From the pop up dialog box, set the inputs to 0 by clicking '0' and then clicking on the input waveform.

Click **OK**.

Click on **Events + -** Add transitions in stimuli by clicking on waveform (not signal name).

Click on **Create CLK** - after editing the dialog box, click on the waveform (not signal name)

When done editing stimuli, click on **OK** and then **Run**.

Clock modulation: How to synthesize additional clocks for counters and state machines

AN075

INTRODUCTION

The Philips CoolRunner™ devices allow for both synchronous and asynchronous (product term) clocking within its architecture. The 32 macrocell device allows for the use of two clocks, while the 64 and 128 macrocell devices provide the user with a total of four clocks. In some instances, designs require the use of more asynchronous clocks than are currently provided by the XPLA™ architecture. While the CoolRunner family will be enhancing its current flexibility by adding additional clocking resources shortly, this note deals with a technique to synthesize an additional asynchronous clock term as long as a higher frequency global clock is available. An excellent prerequisite applications note to read is 'Understanding CoolRunner Clocking Options'.

WHO CARES?

Some of our competitors have the ability to easily implement asynchronous clocks throughout their devices. This capability is afforded at the cost of other things, such as output enable control, flexible routing, and speed. Many designers, having implemented their designs with multiple clocks, wish to 'trade up' and enjoy the performance, power, and flexible routing capabilities of the CoolRunner family. There can be a porting issue if their designs exceed the number of clocks available in the XPLA architecture. This situation can be easily remedied if a higher speed (Nyquist) clock is available.

IN THE BEGINNING . . .

Many of the designs that require additional clocking have register elements that are used only as one bit state machines or as 'flags' to indicate events that have transpired. Typically a one or a zero is clocked into a register by this 'asynchronous' clock, and then reset or preset by a later event. These registered bits can easily be synthesized using 'soft' latches or registers (refer to 'Understanding CoolRunner Clocking Options'). However, if a higher speed clock is available in the design, then a clock enable technique can be used to gate the clock to a register. The higher speed clock may be used only if it is greater than two times the frequency of the 'asynchronous' clock, and only if the high speed clock is continuous or at least present when gating is required. Gating a clock in this fashion is called 'Clock Enabling', and is not provided for by hardware in the XPLA architecture. This technique was originally implemented manually using the schematic in Figure 1. The software tool XPLA Designer provides for a clock enable syntax by automatically synthesizing this logic. The syntax to implement a clock enable in a *.phd file is as follows:

```
equations
   data_out.clk = clk;
   data_out.d = data;
   data_out.ce = en;
end
```

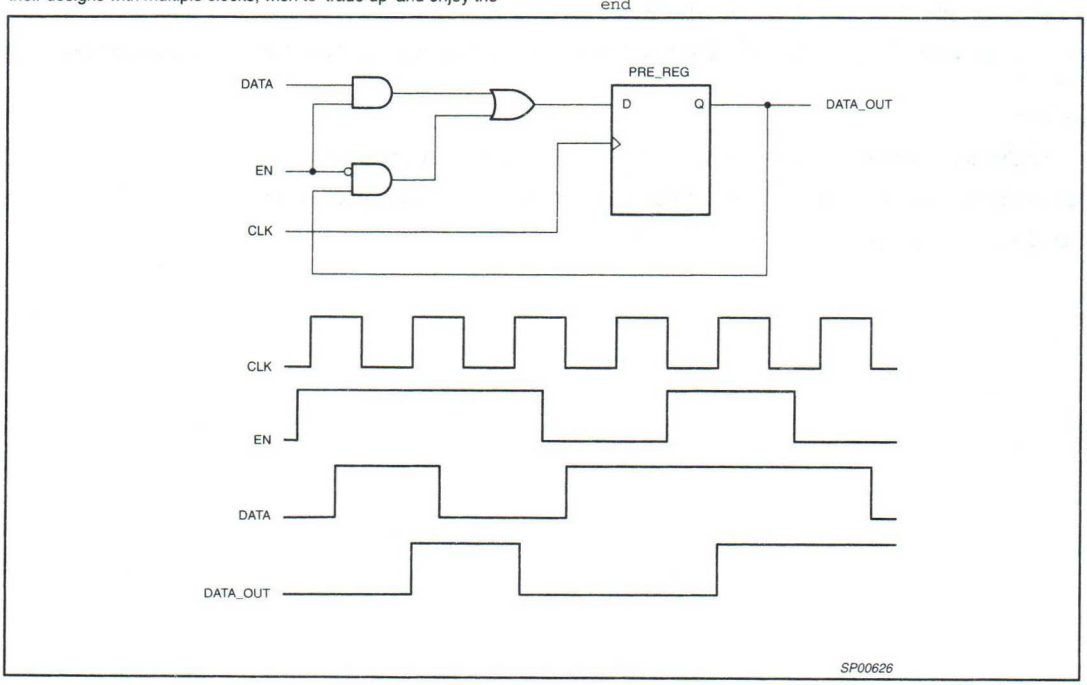


Figure 1.

Clock modulation: How to synthesize additional clocks for counters and state machines

AN075

This technique works well for setting or clearing a registered bit, and it is implemented without requiring an additional macrocell. However, it does not lend itself easily to counters or applications where the input data is not stable during enable. Clock enables favor the implementation of latches more so than registers. Additionally, metastability situations can occur, which will be discussed later. Edge triggered flip flops can be synthesized in 'soft' logic, however they require the use of two macrocells per bit of registered information.

SO WHAT NEXT?

If a designer wants to use additional clock terms beyond those provided by the XPLA architecture, and a faster clock is available in the design, then the clock can be modulated easily by adding a small amount of additional logic. Examine Figure 2.

In this instance, the designer is out of clock resources, but wants to implement a four bit counter to be clocked by the signal 'async_clk'. Note that the portion of the schematic contained in the dashed line is the clock modulation circuit, and it consists of only a 'D' type clock enabled flip flop, a 1n_2and gate, and an input register. The other portion of the circuit is a 'T' type counter, enabled with the 'enabler' signal. The input register (pre_reg) is used to delay the async_clk signal by one clock period and therefore lessening (but not removing entirely) the risk of metastability affecting the MTBF of the design.

At power up, all registers are initialized as reset. pre_reg and en_reg are cleared at this time, or cleared one clock period (plus T_{CO}) after async_clk is low. The output of en_reg is low, which readies the 'enabler' gate(1n_2and) for the high going edge of the registered async_clk. At this point (reset condition), 'enabler' is low, so none of the registers (other than the input register pre_reg) will clock.

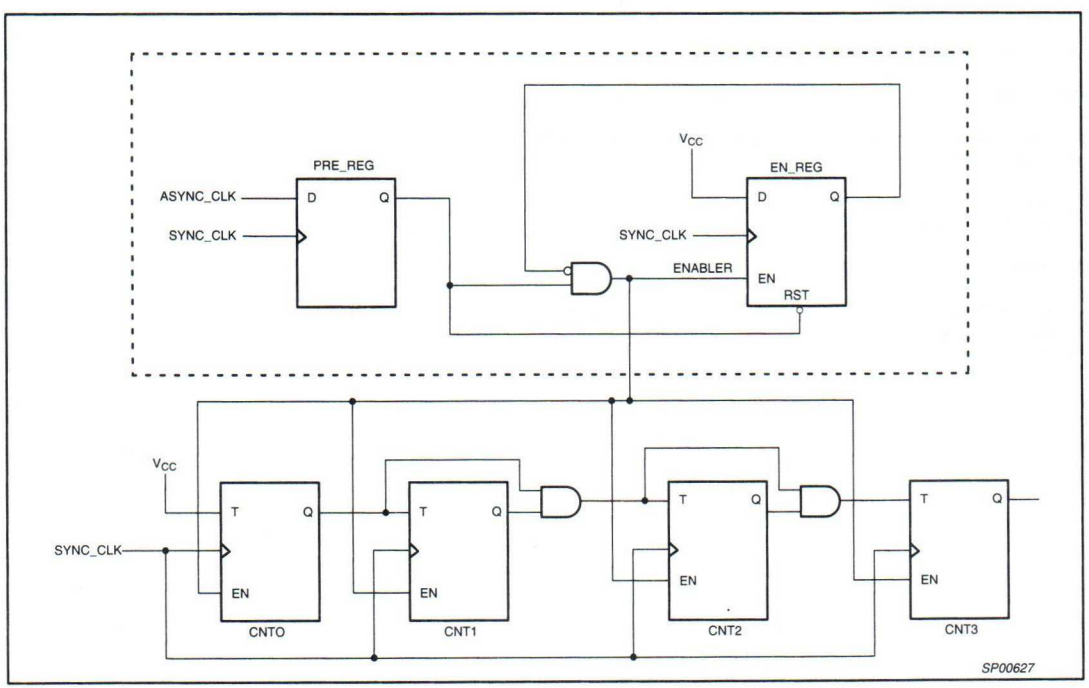


Figure 2.

Clock modulation: How to synthesize additional clocks for counters and state machines

AN075

Some time later, async_clk will transition from a low to a high. pre_reg.q is asserted, and 'enabler' asserts, allowing all of the enabled registers to clock at the next sync_clk pulse. When sync_clk does clock all of the registers, a logic level high is clocked through to the output of en_reg, which disables the clock enable control. When 'async_clk' returns low, it is clocked through pre_reg, and resets the en_reg circuitry. The entire modulation circuit only allows for one 'sync_clk' to be gated through to other registers per 'async_clk' low to high to low transition. This circuitry can be implemented in XPLA Designer using the following *.phd syntax:

```
Module enabler
Title 'Clock modulation circuitry and counter'

sync_clk     pin;                "synchronous clock must be 2x async clock freq.
async_clk    pin;
pre_reg      pin istype 'reg';   "registration register... defer metastability
en_reg       pin istype 'reg';   "modulation register

enabler      node istype 'com';  "node for enabling counters, state mach, etc.

cnt3..cnt0   pin istype 'reg';   "four bit test counter register

count = [cnt3..cnt0];            "name the counter something unique

equations

enabler = !en_reg.q & pre_reg.q; "define enabler signal

pre_reg.clk = sync_clk;          "clean up async timing
pre_reg.d = async_clk;

en_reg.clk = sync_clk;           "Clock available one time only!
en_reg.ce = enabler;
en_reg.d = 1;
en_reg.ar = !pre_reg.q;

count.clk = sync_clk;            "four bit counter stuff.
count.ce = enabler;
count = count + 1;

end
```

Notice how easily the 'enabler' signal can be distributed throughout the design. This technique can be used on groups of registers for counters, data buffers, and state machines. Additionally, this modulation circuitry can be implemented at a one time cost of two macrocells per asynchronous clock regardless of the amount of registers controlled.

SIMULATION OF ENABLER DESIGN

Figure 3 shows a screen capture from XPLA–Sim for the Enabler Design. Note that the counter bits (CNT3..CNT0) increment once for each rising edge of ASYNC_CLK, and that the edge of each CNTx transition is delayed by a maximum of two SYNC_CLK periods.

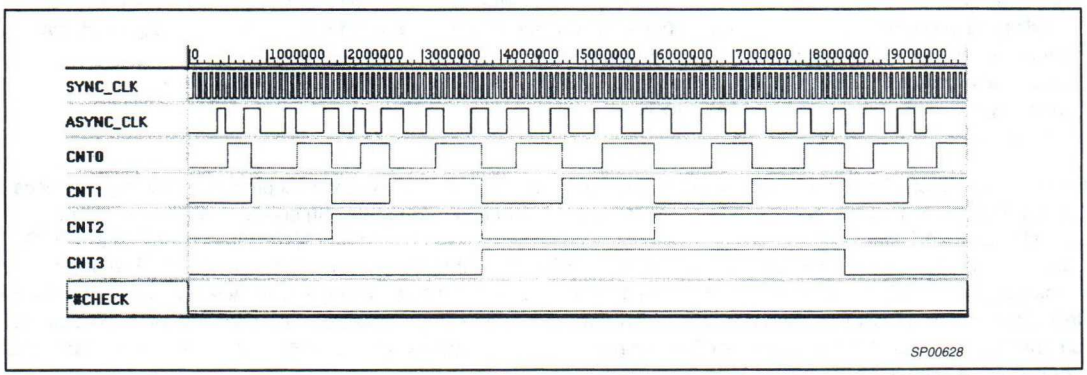


Figure 3.

Using the Philips PZ3960 Evaluation Board AN076

INTRODUCTION

This note discusses the use of the Philips PZ3960 evaluation board. The main functions of the evauation board are the following.

1. Using design entry tools such as schematic editors and programming languages as VHDL, Verilog, Abel, and PHDL, to test designs in silicon. The PZ3960 - PZ3128 interface on the evaluation board facilitates this, allowing functional silicon verification to be done with little or no lab equipment.
2. Configuration of the PZ3960 and programming the PZ3128.
3. Boundary scan operations such as 1149.1 tests.
4. PCI interface

Philips Semiconductor has developed two families of advanced 3-volt and 5-volt complex programmable logic devices (CPLDs). The XPLA 1 series, designated as the PZ5000 - (5-volt) and PZ3000 (3-volt) series devices, is footprint compatible with the Altera 7000 series devices. The XPLA 2 series consists of the 3-volt PZ3960 and the PZ3320[1]. The principle advantage of Philips CPLDs over all existing CPLDs is that they do not consume static power. The other advantages are 25% higher combinatorial logic capacity, high speed 3-volt operation, and an increased ability to fit logic with fixed pinouts. The XPLA 1 family is in-system programmable, and programmable on Data I/O and BP Microsystems programmers, as well as Philip's Low Cost Programmer. The XPLA 2 family uses SRAM to store the configuration, and is configured similarly to existing FPGAs.

The following documentation is related to the use of the Philips PZ3960 evaluation board.

Complex Programmable Logic Devices Data Handbook IC27
PZ3960 Data Sheet

Technical support is available from

coolpld@abq.sc.philips.com
(888) - coolpld
http://www.coolpld.com
ftp://www.coolpld.com

The following is required to use the PZ3960 evaluation board.

Design software -
Philip's XPLA Designer or Minc's[2] PL-Designer Design Entry software
Design entry software if PHDL or DSL isn't used
Download software for slave serial mode
Either PZ3128 evaluation board with JTAG programming software or Philips Low Cost Programmer with 84-pin PLCC socket adaptor
+9V, 500 mA power supply (provided) or 9V Nicad rechargeable battery
PZ3960 - 492-pin BGA
PZ3128 - 84 pin PLCC
Download cable (with a 9 to 25 pin adaptor)

(1) PZ3320 engineering samples are scheduled to be available Q4 1998
(2) Philips recognizes the trademarks of Minc, Intel, and Atmel.

Using the Philips PZ3960 Evaluation Board AN076

Evaluation board description

The PZ3960 evaluation board includes the following components:

PZ3960 and socket
Two 8-pin DIP switches - S1, S2
Nine volt power connector - J1
3 Serial EEPROMs/sockets (U6,U7,U8)
Parallel EEPROM/socket - Intel A28F400 4 MBit Boot Block Flash Memory - U3
Parallel Flash EEPROM/socket - Atmel AT29BV020 2 Megabit Battery Voltage CMOS Flash Memory - U5
10 MHz Clock oscillator - U11
On/Off switch
Power-On Indicator LED - DS1
DONE LED - DS2
2-digit LCD - U10
PCI interface

Download cable - The download cable provides the interface from the PC/workstation to the PZ3960's slave serial on and JTAG ports. The download cable contains a download module which is located near the evaluation board. The download module interfaces to the evaluation board through the download port (JP1) and the JTAG port (JP2) connectors. The download module contains a PS87C51and buffering circuitry. The module's download port connects to the evaluation board slave serial port through a 7-pin to 7-pin connector which is soldered directly to the board. The pinout of the 7 pin connector is listed in the table below.

JP1 PIN	FUNCTION	COLOR
1	VDD	Red
2	CCLK	Green
3	DIN	Yellow
4	DONE	Blue
5	$\overline{\text{PRGM}}$	Purple
6	GND	Black
7	GND	Black

Table 1. PZ3960 slave serial port (JP1)

The 7-pin JTAG port on the evaluation board interfaces to the 8 pin JTAG port on the download module using a flying connector as defined in Table 2.

JP2 PIN	FUNCTION
1	VDD
2	TMS
3	TCK
4	TDI
5	TDO_3
6	TRSTN
7	GND

Table 2. PZ3960 JTAG Port (JP2)

The JP3 and JP4 connectors, similar to the JP1 and JP2 connectors, are used for extending the configuration/JTAG Boundary Scan Ring (BSR) across evaluation boards, respectively. Note that pin 3 on JP3 is DOUT rather than DIN.

JP3 PIN	FUNCTION	COLOR
1	VDD	Red
2	CCLK	Green
3	DOUT	Yellow
4	DONE	Blue
5	$\overline{\text{PRGM}}$	Purple
6	GND	Black
7	GND	Black

Table 3. JP3 connector for Daisy chain operation

The JP4 connector pinout for extending the BSR is given below.

JP4 PIN	FUNCTION
1	VDD
2	TMS
3	TCK
4	TDI (TDO_3)
5	TDO_3
6	TRSTN
7	GND

Table 4. JP4 connector for extending BSR

Power Supply connection - (J1) The power connector is a 1 pin male jack. A 9-volt AC adaptor plugs directly into the jack. There is also a connection for a 9 volt rechargeable Nicad battery. A 3-volt regulator is provided.

On/Off switch (SW1) and LED (DS1) - This switches power to the components on the board, and a red LED indicates that power is applied to the board.

PROGRAM push-button switch (SW3) - The PROGRAM push-button switch is connected to the $\overline{\text{PRGM}}$ pin of the PZ3960. Pressing this switch pulls the $\overline{\text{PRGM}}$ pin to a logic low level.

Reset push-button switch (SW2) - The RESET push-button switch is connected to the $\overline{\text{RESET}}$ pin. When the RESET switch is pressed, $\overline{\text{RESET}}$ is pulled to a logic low level. $\overline{\text{RESET}}$ has two functions. During configuration, it forces the re-start of configuration. After configuration, it asynchronously resets the registers in the PZ3960.

DIP Switches (S1,S2) - The two 8-pin DIP switches are used for the four configuration mode select pins M[3:0] and 12 user-definable inputs to stimulate the PZ3960. When a switch is in the open position, a logic high is input to the I/O pin. When a switch is closed, a logic low is input.

SWITCH	PIN	SWITCH	PIN
S1 -1	AF19	S2-1	M0-AE24
S1-2	M26	S2-2	M1-AF23
S1-3	M25	S2-3	M2-AE22
S1-4	L26	S2-4	M3-AD20
S1-5	AF5	S2-5	P25
S1-6	CLK2	S2-6	AF8
S1-7	CLK3	S2-7	AF7
S1-8	GTSN	S2-8	P26

Table 5. 8-pin DIP switches S1, S2

Clock Oscillator - The 10 MHz clock oscillator is connected to the PZ3960 M3 pin and PZ328 pin 83.

The mode select pins determine the configuration method used. There are four methods to configure the part on the evaluation board. Mode Select pin M3 selects the frequency of the internal oscillator, which is the source of the configuration clock in master configuration modes. The internal oscillator is 1.25 MHz when M3 is high and 10 MHz when M3 is low. Pin M3 should be set high to avoid noise and timing problems.

Evaluation Board Operation

The PZ3960 evaluation board can be configured using the download cable, serial EEPROMs, or either the Intel or Atmel parallel EEPROM. The first step is to create a design and produce a <design>.jed file. Then, run

jed2mcs <design>.jed > <design>.mcs (for configurating non-volatile memory for MP, MS configuration)
or
jed2mcs <design>.jed > <design>.hex (for download operation in SS mode)

at the Dos (Unix) prompt. If the download cable is used, the <design>.hex file is used. If the onboard parallel EEPROM or serial EEPROMs are used, <design>.mcs must be loaded into the (E)EPROM. If EEPROM is not used in the configuration, it should be removed from the evaluation board. So if configuring using MS mode, remove the parallel EEPROMs.

Configuration using the Download cable in Slave Serial mode

1. Set the mode select pins M[3:0] to a high logic level.

2. Plug the supplied download cable into COM1 or COM2 of a PC, or /dev/ttya or /dev/ttyb for a workstation. Plug the download module 7-pin interface to the slave serial connector (JP1).

3. Plug the 9 volt adaptor into the power jack J1 on the evaluation board. Plug the other end of the 9 volt adaptor into an outlet.

4. Switch on the ON/OFF. The red LED on the evaluation board should light. The download module gets power from the evaluation board. When power is applied, both red and green LEDs should light. After a short time, the red LED on the download module should extinguish.

5. In the download software, select the appropriate download port, and then select download.

The download program takes less than one second to configure the device. After configuration, the DONE LED indicates that the DONE pin is high and that configuration is complete. Either "Download successful " or "Download failed" is displayed on the PC. If unlit, repeat the steps above.

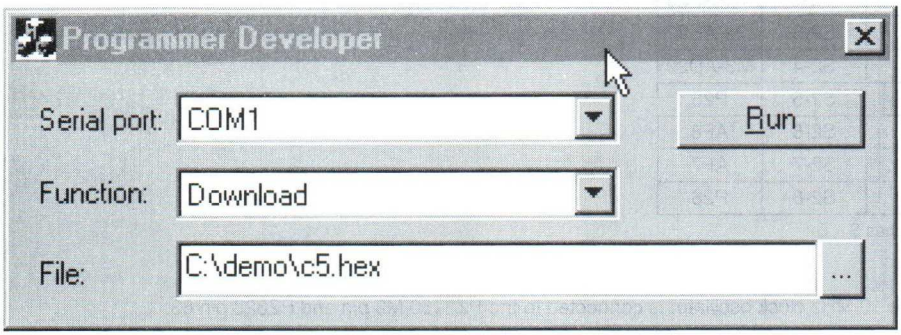


Figure 1. Download Graphical User Interface

Configuration using the serial EEPROM in Master Serial mode

After the mcs file is generated using **jed2mcs**, use a standard PROM programmer such as BP Microsystems or Data I/O to program the EEPROM(s). The serial (E)EPROMs tested by Philips Factory Applications include the re-programmable Atmel AT17256 and the one time programmable Xilinx XC17256. When programming the device, the polarity of the RESET/Output Enable must be set with $\overline{\text{RESET}}$ active low.

1. Set the mode select pins M[2:0] low and M[3] high.
2. Insert the programmed EEPROMs into the 8 pin sockets.
3. Plug the 9 volt adaptor into the evaluation board and plug the other end of the 9 volt adaptor into an outlet.
5. Switch on the ON/OFF so that the red LED is lit.

The serial EEPROM takes less than one second to configure the PZ3960 with M[3] high. After configuration, the DONE LED should light.

Configuration using the parallel EEPROM in Master Parallel mode

This procedure is used for configuring with either the Intel A28F400 or Atmel AT29BV020 flash EEPROM. Generate the <design>.jed file and run **jed2mcs** to generate the mcs file. Use a PROM programmer to program the non-volatile parallel memory.

1. Set the mode select pins M[1:0] low and M[3:2] high.

2. Insert the Intel A28F400 (Atmel AT29BV020) into the 44 pin QFP (32 pin PLCC) socket with the dimple toward the top of the board. The 44-pin QFP socket (U12) lid lifts and slides, and must be locked by sliding the lid vertically to ensure connectivity.

3. Insert the 9 volt adaptor into the jack on the evaluation board.

4. Switch on the ON/OFF switch. The red LED on the evaluation board should light. When configuration is successful, the green LED should light.

The EEPROM takes less than one second to configure the PZ3960. If not lit, repeat the steps above.

Exercises Using the Evaluation board for silicon verification of PZ3960/PZ3128 designs

Attach the download cable to the evaluation board and to the serial port of either a PC or workstation. This will be used to configure the PZ3960. Use either the PZ3128 evaluation board or the Philips Low Cost Programmer to program the PZ3128, and then insert the PZ3128-A84 into the 84-pin PLCC socket (U9). (1)

The following exercises are intended to provide familarity with the evaluation board. The source files for the exercises are included in this note and in electronic format in 960_code.zip on the Philips PPG ftp site www.coolpld.com.The examples here are given using VHDL, but some code exists in Verilog, Abel, and Philips Hardware Description Language (PHDL) as well. For all of these exercises, the pin assignment is provided in the <design>.ctl files on the ftp site.

Exercise 1. Make the PZ3960 display coolpld on the LCD.

1. Compile coolpld.vhd to a jedec file using either XPLA Designer or PL-Designer and the appropriate design entry tool. The coolpld.phd file is on the ftp site for users who are either more familiar with PHDL or without a VHDL compiler.
2. Run **jed2mcs** coolpld.jed > coolpld.mcs at the Dos (Unix) prompt to produce coolpld.mcs.
3. copy (cp) coolpld.mcs coolpld.hex
4. Connect the download cable as described earlier.
4. Invoke the Download software (available on the ftp site), load coolpld.hex, and download.
5. optional - The LCD pin assignment and decode is given in the figure and table below. Repeat these steps and write something else to the LCD.

The VHDL code for coolpld is given in the Appendix 3.

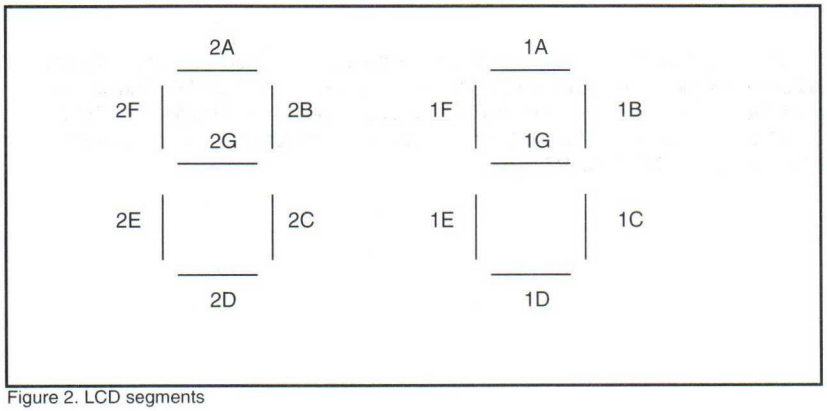


Figure 2. LCD segments

(1) Programming the PZ3128 through the Download cable's JTAG port has not been tested.

SEGMENT	PIN	SEGMENT	PIN
1A	AF15	2A	AE26
1B	AF14	2B	AE25
1C	AF11	2C	AF25
1D	AF10	2D	AC25
1E	AF13	2E	AC26
1F	AF24	2F	AD25
1G	Y26	2G	AD26
2DP	AB26	3DP	Y25

Table 6. LCD - PZ3960 interface

The 2DP and 3DP in Table 6 refer to decimal points to the left of the two digits. When driving the LCD, the PZ3960 should output a low on AF20 pin.

Exercise 2. This exercise uses both the PZ3960 and PZ3128 on the evaluation board to do to the function described below. This approach to testing functionality in silicon will be used in other exercises/tests.

1. PZ3128 provides stimuli to the PZ3960.

2. PZ3960 performs an operation and provide results back to the PZ3128.

3. PZ3128 checks results and provides a pass/fail (p_f) signal(s) back to the PZ3960.

4. PZ3960 displays 1 to the LCD if the test passes, 0 if it fails.

This is basically a software implementation of the VHDL testbench/Device Under Test (DUT) approach for software verification. In some cases, a customer suspects that a problem on a PCB is due to the Phillips CPLD. Running the design on the PZ3960 evaluation board allows the design problem to be isolated to the PCB or Philips tools/CPLD. In the following, the PZ3960 is the DUT and the PZ3128 is the testbench. If the silicon problem is in the PZ3128, the testbench can be done in the PZ3960 and the DUT in the PZ3128.

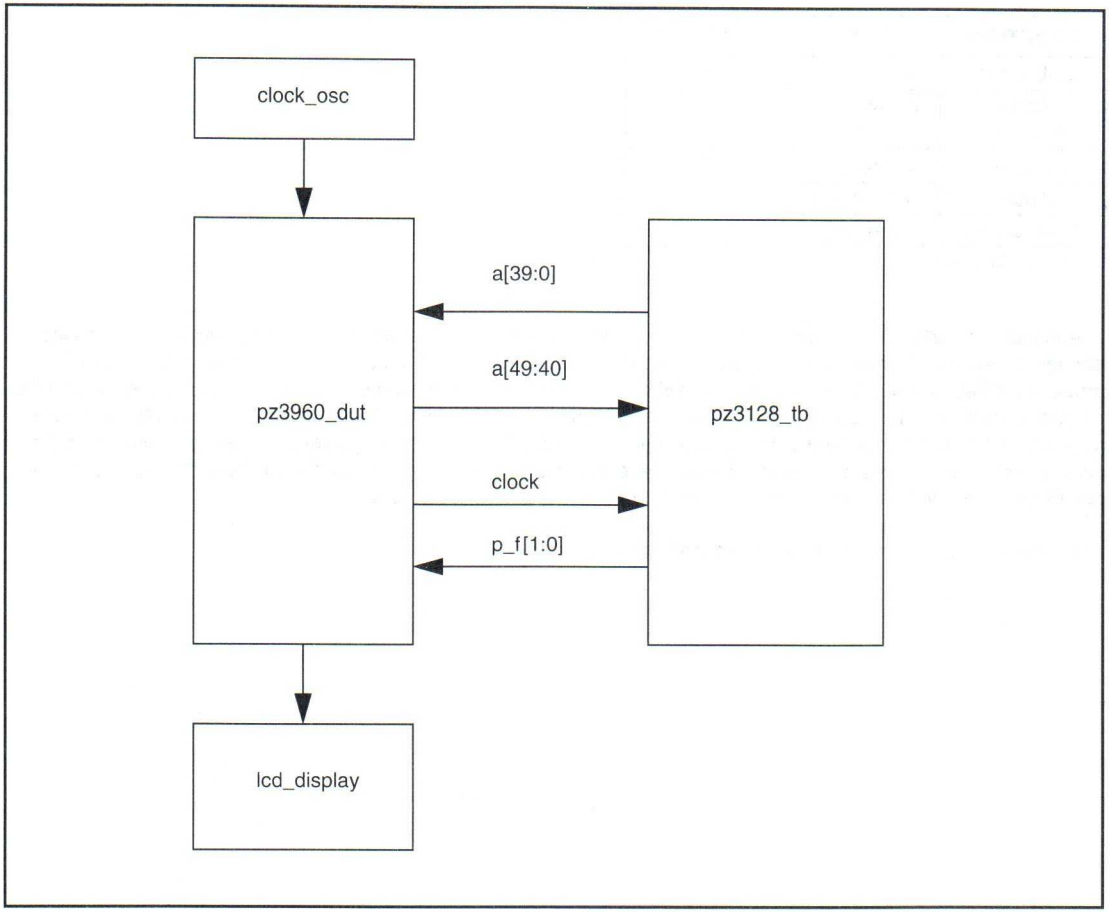


Figure 3. Testbench - DUT model for silicon verification

The 59 pin interface between the PZ3960 and PZ3128 is given in Appendix 1. The operation performed by the PZ3960 is a 4-1 multiplexing of 4 10-bit buses. The PZ3128 code for providing stimuli, checking results, and generating pass/fail status on p_f is given in Appendix 4 (pz3128_tb1.vhd). The PZ3960 code for the top level DUT is given in the Appendix 3 (pz3960_dut1.vhd). This includes components for the multiplexer, clock generator (count24.vhd, count30.vhd) , and LCD driver (lcd_drv.vhd) given in appendices listed below.

The 10 MHz oscillator output is routed to the PZ3960 and PZ3128. The LCD needs to be updated on the order of 1 Hz, and the pz3128 can provide stimuli/ check results at a rate much less than 10 MHz. Because of the PZ3960 capability, the clock divider is done in the PZ3960 using a counter.

Exercise 3. Extend Exercise 2 to do tests of multiple operations (adder, address decoder, comparator, data register), each with multiple test vectors. This is done by including the functions in the PZ3960 in Table 7.

FUNCTION	CODE	APPENDIX
LCD Driver	lcd_drv.vhd	6
Clock Generator	count30.vhd	7
Multiplexer	mux41.vhd	8
Adder	add10.vhd	9
Data register	dr10.vhd	10

Table 7. VHDL code

In Appendix 11, pz128_tb.vhd is the state machine which generates the test vectors. The state machine sequences through 4 test vectors each for each operation (adder, address decoder, comparator, data register). Appendis 12 provides pz3960_dut.vhd. This instantiates the vhdl components listed in the appendices and includes the source for the address decoder and comparator, as well as the other functions. The adder, address decoder, comparator, and data register each respond continuously to the stimuli provided on din[39:0]. The multiplexer accepts input from the adder, address decoder, comparator, and data register, and based on the mux_sel[1:0] inputs from the PZ3128, outputs the results on dout[9:0]. The functions done in the PZ3960 are given in the Figure 4.

Appendix 13 is a vhdl testbench to test the testbench.

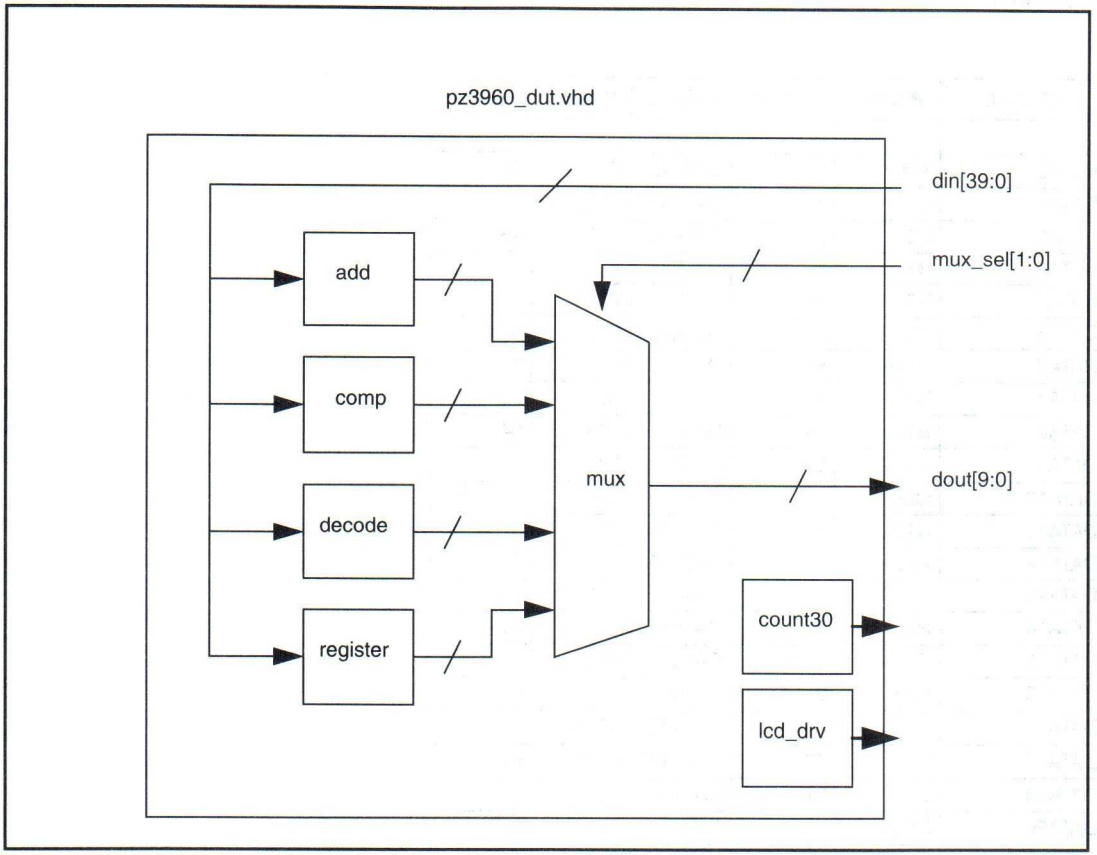


Figure 4. PZ3960 DUT performing multiple operations

Exercise 4. Extend Exercise 3 to add an operation such as a shift register or a subtraction. This requires adding states (in groups of 3) to PZ3128_tb to generate stimuli and test results for your operation. Then add the function to the PZ3960 code, and increase the size of the mux to transmit out the results. The mux_sel size should be increased.

Configuration

The following exercises are used in configuring the PZ3960. These require access to a non-volatile memory programmer such as the Data I-O Unisite or the BP Microsystems programmer.

Exercise 1. Load the coolpld design using Master Parallel Up mode by programming the Intel A28F400 with the Unisite. Set the mode select switches, insert the A28F400, and verify operation.

Exercise 2. Load the coolpld design using Master Parallel Up mode by programming the Atmel 29BV020 with the Unisite. Set the mode select switches, insert the At29BV020, and verify operation.

Exercise 3. Load the coolpld design using Master Serial mode by programming the Atmel AT 17256s. Set the mode select switches, insert the AT17256(s), and verify operation.

Appendix 1

SIGNAL	PZ3960	PZ3128	SIGNAL	PZ3960	PZ3128
DATA0	A16	12	DATA30	R26	44
DATA1	G26	1	DATA31	AF22	45
DATA2	E25	2	DATA32	U25	46
DATA3	B11	4	DATA33	B7	48
DATA4	V26	5	DATA34	V25	49
DATA5	W26	6	DATA35	A18	50
DATA6	AF26	8	DATA36	B17	51
DATA7	B19	9	DATA37	R25	52
DATA8	B14	10	DATA38	T26	54
DATA9	K25	11	DATA39	T25	55
DATA10	L25	15	DATA40	B10	56
DATA11	A12	16	DATA41	U26	57
DATA12	E26	17	DATA42	A10	58
DATA13	G25	18	DATA43	D25	60
DATA14	N25	20	DATA44	AB25	61
DATA15	B13	21	DATA45	B18	63
DATA16	H25	22	DATA46	A8	64
DATA17	A14	24	DATA47	AA26	65
DATA18	A11	25	DATA4;8	AA25	67
DATA19	B15	27	DATA49	J26	68
DATA20	B12	28	DATA50	A13	69
DATA21	F26	29	DATA51	K26	70
DATA22	B8	30	DATA52	A17	73
DATA23	A21	33	DATA53		
DATA24	B20	34	DATA54	B16	75
DATA25	AE23	35	DATA55	A15	76
DATA26	B9	36	DATA56	AF21	77
DATA27	W25	37	DATA57	H26	79
DATA28	A19	39	DATA58	B21	80
DATA29	J25	41			

Table 8. PZ3960 - PZ3128 Interface

Appendix 2

PIN	SIDE B	PKG BALL	SIDE A	PKG BALL
1	-12 V	-	TRSTN	N3
2	TCLK	P4	12 V	-
3	GND	GND	TMS	N4
4	TDO	P3	TDI	P1
5	5 V	NC	5 V	NC
6	5 V	NC	INTAN	C26
7	INTBN	C24	INTCN	C25
8	INTDN	B26	5 V	NC
9	PRSTN1	G4	RESERVED	-
10	RESERVED	-	3.3 V	VDD
11	PRSNT2N	F1	RESERVED	-
CONNECTOR KEY				
CONNECTOR KEY				
14	RESERVED	-	RESERVED	-
15	GND	GND	RST	N23
16	CLK	M3	3.3 V	VDD
17	GND	GND	GNTN	D24
18	REQN	A23	GND	GND
19	3.3 V	VDD	RESERVED	-
20	AD31	B5	AD30	A5
21	AD29	D6	3.3V	VDD
22	GND	GND	AD28	C6
23	AD27	B6	AD26	A6
24	AD25	E7	GND	GND
25	3.3v	VDD	AD24	D7
26	CBE3N	A24	IDSEL	D23
27	AD23	B3	3.3V	VDD
28	GND	GND	AD22	A3
29	AD21	C4	AD20	B4
30	AD19	A4	GND	GND
31	3.3 V	VDD	AD18	D5
32	AD17	F7	AD16	C5
33	CBE2N	F20	3.3V	VDD
34	GND	GND	FRAMEN	B25
35	IRDYN	A25	GND	GND
36	3.3 V	VDD	TRDYN	B24
37	DEVSELN	A22	GND	GND
38	GND	GND	STOPN	B22

PIN	SIDE B	PKG BALL	SIDE A	PKG BALL
39	LOCKN	F24	3.3V	VDD
40	PERRN	E20	SDONE	C22
41	3.3V	VDD	SBON	D22
42	SERRN	D21	GND	GND
43	3.3 V	VDD	PAR	F25
44	CBE1N	C23	AD15	D3
45	AD14	C1	3.3 V	VDD
46	GND	GND	AD13	D2
47	AD12	D1	AD11	E4
48	AD10	E3	GND	GND
49	M66EN	G3	AD9	E2
50	GND	GND	GND	GND
51	GND	GND	GND	GND
52	AD8	E1	CBE0N	B23
53	AD7	A2	3.3V	VDD
54	3.3 V	VDD	AD6	E6
55	AD5	A1	AD4	C3
56	AD3	D4	GND	GNC
57	GND	GND	AD2	B2
58	AD1	B1	AD0	C2
59	3.3 V	VDD	3.3 V	VDD
60	ACK64N	E21	REQ64N	A26
61	5 V	NC	5 V	NC
62	5 V	NC	5 V	NC

NC - No Connect

Table 9. PZ3960 PCI Interface

Appendix 3 coolpld.vhd

```
-- Philips CPLD Applications

library ieee ;
use ieee.std_logic_1164.all ;
use ieee.std_logic_arith.all ;
use ieee.std_logic_unsigned.all ;

entity coolpld is
port (rst,clk : in std_logic ;
bp_low : out std_logic ;
lcd_out : out std_logic_vector (16 downto 1) ;
count : inout std_logic_vector (21 downto 0)
) ;
end coolpld ;

architecture v1 of coolpld is
type state_type is (s0,s1,s2,s3,s4,s5,s6,s7,s8,s9,s10,s11,s12,s13,s14,s15) ;
signal present_state,next_state : state_type ;

begin

process (rst,clk)
begin
if rst = '1' then
count <= (others => '0') ;
elsif clk'event and clk = '1' then
count <= count + '1' ;
end if ;
end process ;

process (rst,count(21))
begin
if rst = '1' then
present_state <= s0 ;
elsif count(21)'event and count(21) = '1' then
present_state <= next_state ;
end if ;
end process ;

process (present_state)
begin
case present_state is
when s0 =>
lcd_out <= "0111001000000000" ;
next_state <= s1 ;

when s1 =>
lcd_out <= "0000000000000000" ;
next_state <= s2 ;

when s2 =>
```

```
lcd_out <= "0000000010111000" ;
next_state <= s3 ;

when s3 =>
lcd_out <= "0000000000000000" ;
next_state <= s4 ;

when s4 =>
lcd_out <= "0000000010111000" ;
next_state <= s5 ;

when s5 =>
lcd_out <= "0000000000000000" ;
next_state <= s6 ;

when s6 =>
lcd_out <= "0000000001100000" ;
next_state <= s7 ;

when s7 =>
lcd_out <= "0000000000000000" ;
next_state <= s8 ;

when s8 =>
lcd_out <= "1110011000000000" ;
next_state <= s9 ;

when s9 =>
lcd_out <= "0000000000000000" ;
next_state <= s10 ;

when s10 =>
lcd_out <= "0000000001100000" ;
next_state <= s11 ;

when s11 =>
lcd_out <= "0000000000000000" ;
next_state <= s12 ;

when s12 =>
lcd_out <= "1011110000000000" ;
next_state <= s13 ;

when s13 =>
lcd_out <= "0000000000000000" ;
next_state <= s14 ;

when s14 =>
lcd_out <= "0000000000000000" ;
next_state <= s15 ;

when s15 =>
lcd_out <= "0000000000000000" ;
```

```
next_state <= s0 ;

end case ;
end process ;

bp_low <= '0' ;

end v1 ;
```

Appendix 4

```
-- Philips CPLD Applications
-- pz3128_tb1 - Outputs to 10-bit busses a,b,c,d
-- Accepts response on 10-bit din bus
-- Feb 20, 1998

library ieee ;
use ieee.std_logic_1164.all ;

entity pz3128_tb1 is
port (din : in std_logic_vector (9 downto 0) ;
clk : in std_logic ;
rst : in std_logic ;
sw_in : in std_logic_vector (1 downto 0) ;
a : out std_logic_vector (39 downto 0)  ;
p_f : out std_logic_vector (1 downto 0)
) ;
end pz3128_tb1 ;

architecture v1 of pz3128_tb1 is
type state_type is (s0,s1,s2,s3,s4) ;
signal  present_state,next_state : state_type ;

begin

a(39 downto 30) <= "1111111111" ;
a(29 downto 20) <= "0000000000" ;
a(19 downto 10) <= "1010101010" ;
a(9 downto 0) <= "1111100000" ;

process (rst,clk)
begin
if rst = '1' then
present_state <= s0 ;
elsif clk'event and clk = '1' then
present_state <= next_state ;
end if ;
end process ;
```

```
process (present_state,sw_in,din)
begin
case present_state is
when s0 =>
p_f <= "00" ;
if sw_in = "00" then
next_state <= s1 ;
elsif sw_in = "01" then
next_state <= s2 ;
elsif sw_in = "10" then
next_state <= s3 ;
else
next_state <=  s4 ;
end if ;

when s1 =>
if din = "1111111111" then
p_f <= "11" ;
else
p_f <= "10" ;
end if ;
next_state <= s0 ;

when s2 =>
if din = "0000000000" then
p_f <= "11" ;
else
p_f <= "10" ;
end if ;
next_state <= s0 ;

when s3 =>
if din = "1010101010" then
p_f <= "11" ;
else
p_f <= "10" ;
end if ;
next_state <= s0 ;

when s4 =>
if din = "1111100000" then
p_f <= "11" ;
else
p_f <= "10" ;
end if ;
next_state <= s0 ;

when others =>
p_f <= "00" ;
next_state <= s0 ;


end case ;
```

```
end process ;

end v1 ;
```

Appendix 5

```
-- Philips CPLD Applications
-- Feb 6, 1998
-- pz3960_dut1 static test w switches controlling vectors

library ieee ;
use ieee.std_logic_1164.all ;
use ieee.std_logic_arith.all ;
use ieee.std_logic_unsigned.all ;

entity pz3960_dut1 is
port (rst,clk : in std_logic ;
sw_out : out std_logic_vector (1 downto 0)  ;
sw_in : in std_logic_vector (1 downto 0)  ;
p_f : in std_logic_vector (1 downto 0)  ;
rst_out : out std_logic ;
clk_out : out std_logic ;
bp_low : out std_logic ;
lcd_out : out std_logic_vector (15 downto 0)  ;
ain : in std_logic_vector (39 downto 0)  ;
aout : out std_logic_vector (9 downto 0)  ;
count : inout std_logic_vector (30 downto 0)
) ;
end pz3960_dut1;

architecture v1 of pz3960_dut1 is

component mux41
port (sel : in std_logic_vector (1 downto 0) ;
ain : in std_logic_vector (39 downto 0) ;
aout : out std_logic_vector (9 downto 0)
) ;
end component ;

component count30
port (rst,clk : in std_logic ;
count :  inout std_logic_vector (30 downto 0)
) ;
end component ;

component lcd_drv
port (rst,clk : in std_logic ;
p_f : in std_logic_vector (1 downto 0) ;
lcd_out : out std_logic_vector (15 downto 0)
) ;
end component ;

signal clk21 : std_logic ;
```

```
begin

sw_out <= sw_in ;
clk21 <= count(21) ;
bp_low <= '0' ;
rst_out <= rst ;
clk_out <= count(23) ;

u1 : count30 port map (rst,clk,count) ;
u2 : mux41 port map (sw_in(1 downto 0),ain,aout) ;
u3 : lcd_drv port map (rst,clk21,p_f,lcd_out) ;

end v1 ;
```

Appendix 6

```
-- Philips CPLD Applications
-- LCD driver
-- Feb 6, 1998

library ieee ;
use ieee.std_logic_1164.all ;

entity lcd_drv is
port (rst,clk : in std_logic ;
p_f : in std_logic_vector (1 downto 0)  ;
lcd_out : out std_logic_vector (15 downto 0)
) ;
end lcd_drv ;

architecture v1 of lcd_drv is
type state_type is (s0,s1,s2,s3) ;
signal present_state,next_state : state_type ;

begin

process (rst,clk)
begin
if rst = '1' then
present_state <= s0 ;
elsif clk'event and clk = '1' then
present_state <= next_state ;
end if ;
end process ;

process (present_state,p_f)
begin
case present_state is
when s0 =>
lcd_out <= "0000000000000000" ;
```

```
next_state <= s1 ;

when s1 =>
lcd_out <= "0000000000000000" ;
if p_f = "11" then
next_state <= s2  ;
elsif p_f = "10" then
next_state <= s3 ;
else
next_state <= s0 ;
end if ;

when s2 =>
lcd_out <= "0000000001100000" ; -- outputs 1
next_state <= s0 ;

when s3 =>
lcd_out <= "0000000001111110" ; -- outputs 0
next_state <= s0 ;

end case ;
end process ;

end v1 ;
```

Appendix 7

```
-- Philips CPLD Applications
-- Feb 6, 1998
-- counter dividing 10MHz clock to 1 Hz

library ieee ;
use ieee.std_logic_1164.all ;
use ieee.std_logic_arith.all ;
use ieee.std_logic_unsigned.all ;

entity count30 is
port (rst,clk : in std_logic ;
count : inout std_logic_vector (30 downto 0)
) ;
end count30 ;

architecture v1 of count30 is
begin

process (rst,clk)
begin
if rst = '1' then
count <= (others => '0') ;
elsif clk'event and clk = '1' then
count <= count +  '1' ;
end if ;
```

```
end process ;

end v1 ;
```

Appendix 8

```
-- Philips CPLD Applications
-- 10-bit 4-1 Mux
-- Feb 6, 1998

library ieee ;
use ieee.std_logic_1164.all ;

entity mux41 is
port (sel : std_logic_vector (1 downto 0) ;
ain : in std_logic_vector (9 downto 0) ;
bin : in std_logic_vector (9 downto 0) ;
cin : in std_logic_vector (9 downto 0) ;
din : in std_logic_vector (9 downto 0) ;
dout : out std_logic_vector (9 downto 0)
) ;
end mux41 ;

architecture v1 of mux41 is
begin

process (sel,ain,bin,cin,din)
begin
case sel is
when "00" =>
dout <= ain(9 downto 0) ;
when "01" =>
dout <= bin(9 downto 0) ;
when "10" =>
dout <= cin(9 downto 0) ;
when others =>
dout <= din(9 downto 0) ;
end case ;
end process ;
end v1 ;
```

Appendix 9

```
-- Philips CPLD Applications
-- March 6, 1998
-- 10-bit adder

library ieee ;
use ieee.std_logic_1164.all ;
use ieee.std_logic_arith.all ;

entity add10 is
port (
```

```
a,b : in unsigned (9 downto 0) ;
sum : out unsigned (9 downto 0)
) ;
end add10 ;

architecture v1 of add10 is
begin
sum <= a + b ;
end v1 ;
```

Appendix 10

```
-- Philips CPLD Applications
-- Mar 6,1998
-- 10-bit register

library ieee ;
use ieee.std_logic_1164.all ;

entity reg10 is
port (
rst,clk : in std_logic ;
din : in std_logic_vector (9 downto 0) ;
dout : out std_logic_vector (9 downto 0)
) ;
end reg10 ;

architecture v1 of reg10 is
begin

process (rst,clk)
begin
if rst = '1' then
dout <= (others => '0') ;
elsif clk'event and clk = '1' then
dout <= din ;
end if ;
end process ;

end v1 ;
```

Appendix 11

```
-- Philips CPLD Applications
-- pz128_tb   - Outputs to 10-bit busses a,b,c,d
-- Accepts response on 10-bit din bus
-- Feb 20, 1998
-- This tests a comparator, register, address decoder, and adder.
-- Three states are used for each vector. Results are tested in middle state.

library ieee ;
```

```
use ieee.std_logic_1164.all ;

entity pz128_tb is
port (din : in std_logic_vector (9 downto 0) ;
-- rst : in std_logic ;
clk : in std_logic ;
a : out std_logic_vector (39 downto 0)  ;
p_f : out std_logic_vector (1 downto 0) ;
sel : out std_logic_vector (1 downto 0)
) ;
end pz128_tb ;

architecture v1 of pz128_tb is
type state_type is
(s0,s1,s2,s3,s4,s5,s6,s7,s8,s9,s10,s11,s12,s13,s14,s15,s16,s17,s18,s19,s20,s21,s22,s23,s24,s25,s26,s27,s28,s29,s
30,s31) ;
signal  present_state,next_state : state_type ;

begin

process (clk)
begin
-- if rst = '1' then
-- present_state <= s0 ;
if clk'event and clk = '1' then
present_state <= next_state ;
end if ;
end process ;

process (present_state,din)
begin
case present_state is

when s0 =>
a(39 downto 0) <= (others => '0') ;
sel <= "00" ;
p_f <= "00" ;
next_state <=  s1 ;

-- Test cmp10

when s1 =>
a(39 downto 20) <= "11111000001111100000" ;
a(19 downto 0)  <= "11111000001111100000" ;
sel <= "00" ;
p_f <= "00" ;
next_state <= s2 ;

when s2 =>
a(39 downto 20) <= "11111000001111100000" ;
a(19 downto 0)  <= "11111000001111100000" ;
sel <= "00" ;
if din = "0000000001" then
```

```
p_f <= "11" ;
else
p_f <= "10" ;
end if ;
next_state <= s3 ;

when s3 =>
a(39 downto 20) <= "11111000001111100000" ;
a(19 downto 0)  <= "11111000001111100000" ;
sel <= "00" ;
p_f <= "00" ;
next_state <= s4 ;

-- Apply 2nd comparator vector (should produce eq = 0)

when s4 =>
a(39 downto 20) <= "11111001001111100001" ;
a(19 downto 0)  <= "11111000001111100000" ;
sel <= "00" ;
p_f <= "00" ;
next_state <= s5 ;

when s5 =>
a(39 downto 20) <= "11111001001111100000" ;
a(19 downto 0)  <= "11111000001111100000" ;
sel <= "00" ;
if din = "0000000001" then
p_f <= "11" ;
else
p_f <= "10" ;
end if ;
next_state <= s6 ;

when s6 =>
a(39 downto 20) <= "11111001001111100001" ;
a(19 downto 0)  <= "11111000001111100000" ;
sel <= "00" ;
p_f <= "00" ;
next_state <= s7 ;

-- Test reg10

when s7 =>
a(39 downto 10) <= (others => '0')  ;
a(9 downto 0)  <= "0000011111" ;
sel <= "01" ;
p_f <= "00" ;
next_state <= s8 ;

when s8 =>
a(39 downto 10) <= (others => '0')  ;
a(9 downto 0)  <= "0000011111" ;
sel <= "01" ;
```

```
if din = "0000011111" then
p_f <= "11" ;
else
p_f <= "10" ;
end if ;
next_state <= s9 ;

when s9 =>
a(39 downto 10) <= (others => '0') ;
a(9 downto 0) <= "0000011111" ;
sel <= "01" ;
p_f <= "00" ;
next_state <= s10 ;

-- Apply second register vector

when s10 =>
a(39 downto 10) <= (others => '0') ;
a(9 downto 0) <= "1111100000" ;
sel <= "01" ;
p_f <= "00" ;
next_state <= s11 ;

when s11 =>
a(39 downto 10) <= (others => '0') ;
a(9 downto 0) <= "1111100000" ;
sel <= "01" ;
if din = "1111100000" then
p_f <= "11" ;
else
p_f <= "10" ;
end if ;
next_state <= s12 ;

when s12 =>
a(39 downto 10) <= (others => '0') ;
a(9 downto 0) <= "1111100000" ;
sel <= "01" ;
p_f <= "10" ;
next_state <= s13 ;

-- Apply third register vector

when s13 =>
a(39 downto 10) <= (others => '0') ;
a(9 downto 0) <= "1111100000" ;
sel <= "01" ;
p_f <= "00" ;
next_state <= s14 ;

when s14 =>
a(39 downto 10) <= (others => '0') ;
a(9 downto 0) <= "1111100000" ;
```

```
sel <= "01" ;
if din = "1111100000" then
p_f <= "11" ;
else
p_f <= "10" ;
end if ;
next_state <= s15 ;

when s15 =>
a(39 downto 10) <= (others => '0') ;
a(9 downto 0) <= "1111100000" ;
sel <= "01" ;
p_f <= "00" ;
next_state <= s16 ;

-- Apply fourth register vector

when s16 =>
a(39 downto 10) <= (others => '0') ;
a(9 downto 0) <= "1111100000" ;
sel <= "01" ;
p_f <= "00" ;
next_state <= s17 ;

when s17 =>
a(39 downto 10) <= (others => '0') ;
a(9 downto 0) <= "1111100000" ;
sel <= "01" ;
if din = "1111100000" then
p_f <= "11" ;
else
p_f <= "10" ;
end if ;
next_state <= s18 ;

when s18 =>
a(39 downto 10) <= (others => '0') ;
a(9 downto 0) <= "1111100000" ;
sel <= "01" ;
p_f <= "00" ;
next_state <= s19 ;


-- Test memory address decoder

when s19 =>
a(39 downto 20) <= (others => '0') ;
a(19 downto 0) <= "00000000000000000000" ;
sel <= "10" ;
p_f <= "00" ;
next_state <= s20 ;

when s20 =>
```

```
a(39 downto 20) <= (others => '0') ;
a(19 downto 0)  <= "00000000000000000000" ;
sel <= "10" ;
if din = "0000000001" then
p_f <= "11" ;
else
p_f <= "10" ;
end if ;
next_state <= s21 ;

when s21 =>
a(39 downto 20) <= (others => '0') ;
a(19 downto 0)  <= "00000000000000000000" ;
sel <= "10" ;
p_f <= "00" ;
next_state <= s22 ;

-- apply 2nd address decoder vector

when s22 =>
a(39 downto 20) <= (others => '0') ;
a(19 downto 0)  <= "00000111111111111111" ;
sel <= "10" ;
p_f <= "00" ;
next_state <= s23 ;

when s23 =>
a(39 downto 20) <= (others => '0') ;
a(19 downto 0)  <= "11111111111111111111" ;
sel <= "10" ;
if din = "0000001111" then
p_f <= "11" ;
else
p_f <= "10" ;
end if ;
next_state <= s24 ;

when s24 =>
a(39 downto 20) <= (others => '0') ;
a(19 downto 0)  <= "11111111111111111111" ;
sel <= "10" ;
p_f <= "00" ;
next_state <= s25 ;

-- Test add10

when s25 =>
a(39 downto 20) <= (others => '0') ;
a(19 downto 10)  <= "0000000001" ;
a(9 downto 0)  <= "0000111000" ;
sel <= "11" ;
p_f <= "00" ;
next_state <= s26 ;
```

```
when s26 =>
a(39 downto 20) <= (others => '0') ;
a(19 downto 10)  <= "0000000001" ;
a(9 downto 0)  <= "0000111000" ;
sel <= "11" ;
if din = "0000111001" then
p_f <= "11" ;
else
p_f <= "10" ;
end if ;
next_state <= s27 ;

when s27 =>
a(39 downto 20) <= (others => '0') ;
a(19 downto 10)  <= "0000000001" ;
a(9 downto 0)  <= "0000111000" ;
sel <= "11" ;
p_f <= "00" ;
next_state <= s28 ;

-- apply 2nd adder vector

when s28 =>
a(39 downto 20) <= (others => '0') ;
a(19 downto 10)  <= "0000000011" ;
a(9 downto 0)  <= "0000111000" ;
sel <= "11" ;
p_f <= "00" ;
next_state <= s29 ;

when s29 =>
a(39 downto 20) <= (others => '0') ;
a(19 downto 10)  <= "0000000011" ;
a(9 downto 0)  <= "0000111000" ;
sel <= "11" ;
if din = "0000111011" then
p_f <= "11" ;
else
p_f <= "10" ;
end if ;
next_state <= s30 ;

when s30 =>
a(39 downto 20) <= (others => '0') ;
a(19 downto 10)  <= "0000000011" ;
a(9 downto 0)  <= "0000111000" ;
sel <= "11" ;
p_f <= "00" ;
next_state <= s31 ;

when s31 =>
a(39 downto 20) <= (others => '0') ;
```

```
a(19 downto 10)  <= "0000000000" ;
a(9 downto 0)  <= "0000000000" ;
sel <= "00" ;
p_f <= "00" ;
next_state <= s30 ;

when others =>
sel <= "00" ;
p_f <= "00" ;
next_state <= s0 ;

end case ;
end process ;

end v1 ;
```

Appendix 12

```
-- Philips CPLD Applications
-- March 6, 1998
-- pz960_dut

library ieee ;
use ieee.std_logic_1164.all ;
use ieee.std_logic_arith.all ;
use ieee.std_logic_unsigned.all ;

entity pz960_dut is
port (rst,clk : in std_logic ;
-- cnt_out : out std_logic_vector (28 downto 26)  ;
clk_out : out std_logic ;
p_f : in std_logic_vector (1 downto 0)  ;
sel : in std_logic_vector (1 downto 0)  ;
bp_low : out std_logic ;
lcd_out : out std_logic_vector (15 downto 0)  ;
din : in std_logic_vector (39 downto 0)  ;
dout : out std_logic_vector (9 downto 0)  ;
count : inout std_logic_vector (30 downto 0)
) ;
end pz960_dut ;

architecture v1 of pz960_dut is

component mux41
port (sel : in std_logic_vector (1 downto 0) ;
ain : in std_logic_vector (9 downto 0) ;
bin : in std_logic_vector (9 downto 0) ;
cin : in std_logic_vector (9 downto 0) ;
din : in std_logic_vector (9 downto 0) ;
dout : out std_logic_vector (9 downto 0)
) ;
end component ;
```

```
component count30
port (rst,clk : in std_logic ;
count :  inout std_logic_vector (30 downto 0)
) ;
end component ;

component lcd_drv
port (rst,clk : in std_logic ;
p_f : in std_logic_vector (1 downto 0) ;
lcd_out : out std_logic_vector (15 downto 0)
) ;
end component ;

component reg10
port (rst,clk : in std_logic ;
din : in std_logic_vector (9 downto 0) ;
dout : out std_logic_vector (9 downto 0)
) ;
end component ;

component add10
port (a,b : in unsigned (9 downto 0) ;
sum : out unsigned (9 downto 0)
) ;
end component ;

signal clk21 : std_logic ;

signal cmp_out : std_logic_vector (9 downto 0) ;
signal decod_out : std_logic_vector (9 downto 0) ;
signal reg10_out : std_logic_vector (9 downto 0) ;
signal sum_out : unsigned (9 downto 0) ;
signal mux_out : std_logic_vector (9 downto 0) ;
signal cs0 : std_logic ;
signal cs1n : std_logic ;
signal cs2n : std_logic ;
signal cs3n : std_logic ;
signal eq : std_logic ;

begin

-- Comparator

eq <= '1' when (din(39 downto 20) = din(19 downto 0)) else '0' ;

cmp_out <= eq & "000000000" ;

-- Memory Address Decoder

mem_ad_decod : process (din(19 downto 0))
begin
cs0 <= '0' ;
```

```
cs1n <= '1' ;
cs2n <= '1' ;
cs3n <= '1' ;
if din(19 downto 0) >= x"00000" and din(19 downto 0) < x"00255"  then cs0 <= '1' ;
elsif din(19 downto 0) >= x"00255" and din(19 downto 0) < x"01000" then cs1n <= '0' ;
elsif din(19 downto 0) >= x"01000" and din(19 downto 0) < x"10000" then cs2n <= '0' ;
elsif din(19 downto 0) >= x"10000" then cs3n <= '0' ;
end if ;
end process ;

decod_out(3) <= cs0  ;
decod_out(2) <= cs1n  ;
decod_out(1) <= cs2n  ;
decod_out(0) <= cs3n  ;

-- cnt_out <= count (28 downto 26)  ;
clk_out <= count(22) ;
clk21 <= count(21) ;
bp_low <= '0' ;
-- rst_out <= rst ;
dout <= mux_out ;

u1 : count30 port map (rst,clk,count) ;
u2 : mux41 port map (sel,cmp_out,reg10_out,decod_out,std_logic_vector(sum_out),mux_out) ;
u3 : lcd_drv port map (rst,clk21,p_f,lcd_out) ;
u4 : reg10 port map (rst,clk,din(9 downto 0),reg10_out) ;
u5 : add10 port map (unsigned(din(9 downto 0)),unsigned(din (19 downto 10)),sum_out) ;

end v1 ;
```

Appendix 13

```
-- CPLD Applications
-- March 9, 1998
-- pz128_tb

library ieee ;
use ieee.std_logic_1164.all ;
use ieee.std_logic_textio.all ;
use std.textio.all ;

entity testbench is
end;

library ieee ;
use ieee.std_logic_1164.all;

architecture v1 of testbench is

component pz128_tb

port (din: in std_logic_vector (9 downto 0) ;
rst : in std_logic ;
```

```
clk: in std_logic ;
a: out std_logic_vector (39 downto 0) ;
p_f: out std_logic_vector (1 downto 0)
);
end component;

signal  din : std_logic_vector (9 downto 0)  ;
signal  rst : std_logic ;
signal  clk : std_logic ;
signal  a : std_logic_vector (39 downto 0)  ;
signal  p_f : std_logic_vector (1 downto 0) ;

begin

dut: pz128_tb
port map (din => din,
 rst => rst,
 clk => clk,
 a => a ,
 p_f => p_f);


rst <= '1'after 1 ns, '0' after 10 ns ;

process
file INFILE : text is in "pz128_tb.stm" ;
variable LINE_BUFFER : line ;
variable VECTOR : std_logic_vector (9 downto 0) ;
begin
wait until clk = '0' and clk'event ;
while (not endfile(INFILE)) loop
readline (INFILE, LINE_BUFFER) ;
read (LINE_BUFFER,VECTOR) ;
din <= VECTOR (9 downto 0) ;
wait until clk = '0' and clk'event ;
end loop ;
wait ;
end process ;

process
begin

   clk <= '0' ;
   wait for 10 ns ;
   clk <= '1' ;
   wait for 10 ns ;

end process;

end  v1 ;
```

VHDL Easy Design Flow for Philips CPLDs

AN078

INTRODUCTION

This note provides the steps for using MINC[1] VHDL Easy and Philips Semiconductor's XPLA Designer tools to compile a digital design into Philips' Complex Programmable Logic Devices (CPLDs). Philips provides fast zero power CPLDs which offer propogation delays as low as 6nS and the lowest static and dynamic power.
This design is generated using VHDL synthesis via the VHDL Easy tool from Minc, Inc. and compiled to a jedec file using XPLA Designer.The VHDL source file is synthesized in MINC VHDL Easy. The resulting EDIF output is imported into XPLA Designer to implement the design into a Philips' CPLD.
Technical support for the design flow described in this application note is provided by:

Philips Technical Assistance

Telephone: 1-888-coolpld

E-Mail: coolpld@abq.sc.philips.com

Web Site: http://www.coolpld.com

Fax: 1-505-822-7804

Fax on Demand: 1-800-282-2000

REFERENCES

XPLA Designer User's Guide

Philips Complex Programmable Logic Devices Data Handbook

MINC VHDL Easy Manual

INSTALLATION REQUIREMENTS

This design requires that the following PC-based software has successfully been installed and all authorization files have been received:

MINC VHDL Easy V1.1

XPLA Designer v 2.55

The Philips CoolRunner series can be targeted using MINC VHDL Easy using VHDL source code. This design targets the Philips PZ3032 complex programmable logic device.This design is a manchester decoder. 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.

(1) Philips acknowledges the trademarks of the companies mentioned in this document.

PHILIPS XPLA ARCHITECTURE

When writing VHDL source code for a design targeted to a Philips CoolRunner device, certain architecture differences should be taken into account. Please refer to the Philips Complex Programmable Logic Devices Data Handbook for a full architectural description of these devices.

RESETS AND PRESETS

Resets and presets are active high. Philips XPLA architecture supports either an asynchronous preset, or an asynchronous reset, but not both for the D/T -type registers in the macrocells. Therefore, when describing the behavior of a register in VHDL, describe either the reset or the preset as asynchronous and the other as synchronous. The following code illustrates a D-type register with both asynchronous reset and preset:

```
-- D-type register with both asynchronous reset and preset
-- This behavior is not supported by Philips XPLA architecture
async_pre_rst: process(clk,pre,rst)
begin
              if (pre = '1') then
                     q <= '1';
              elsif (rst = '1') then
                     q <= '0';
              elsif (clk'event and clk= '1') then
                     q <= d;
              end if;
end process;
```

If a register is described in the VHDL source code with both an asynchronous preset and reset as illustrated in the code above, a warning is generated from the compiler in XPLA Designer and a fatal error is generated from the fitter in XPLA Designer.

The next two VHDL samples show how to describe a register with either an asynchronous reset with a synchronous preset, or an asynchronous preset with a synchronous reset. Both of these register descriptions are supported by the XPLA architecture and will not generate warnings or errors. These same examples can be modified for a T-type register if needed by the application.

```
-- D-type register with asynchronous reset and synchronous preset
async_rst: process(clk,rst)
begin
              if (rst = '1') then
                     q <= '0';
              elsif (clk'event and clk= '1') then
                     if (pre = '1') then
                            q <= '1';
                     else
                            q <= d;
                     end if;
              end if;
end process;
```

```
-- D-type register with asynchronous preset and synchronous reset
async_pre: process(clk,pre)
begin
            if (pre = '1') then
                  q <= '1';
            elsif (clk'event and clk= '1') then
                  if (rst = '1') then
                              q <= '0';
                  else
                              q <= d;
                  end if;
            end if;
end process;
```

OUTPUT ENABLES

Output enables are active high. There are no internal tri-statable busses within the XPLA architecture. Outputs can be tri-stated.

CLOCKS

The XPLA architecture provides dedicated global clock pins to low skew clock networks. The 32 macrocell devices only have two global clocks, one must be driven from an external source, the other can be driven from an external source or generated internally from a macrocell. The 64 and 128 macrocell devices have four global clocks. Again, one must be driven from an external source, the other 3 can be driven from an external source or generated internally from macrocells. Note that when global clocks are generated internally from macrocells, the pin dedicated to that global clock will output the clock waveform.

If the design requires more clocks than available with the current XPLA architecture, clock enables can be used in some cases to provide the same functionality without requiring an additional clock. The VHDL code below shows how clock enables are described:

```
clock_enable: process(clk)
begin
            if (clk'event and clk= '1') then
                  if (clk_en = '1') then
                              q <= d;
                  else
                              q <= q;
                  end if;
            end if;
end process;
```

DESIGN FLOW

The VHDL Easy design flow for Philips CPLDs starts with synthesizing the VHDL design using Minc's VHDL Easy.

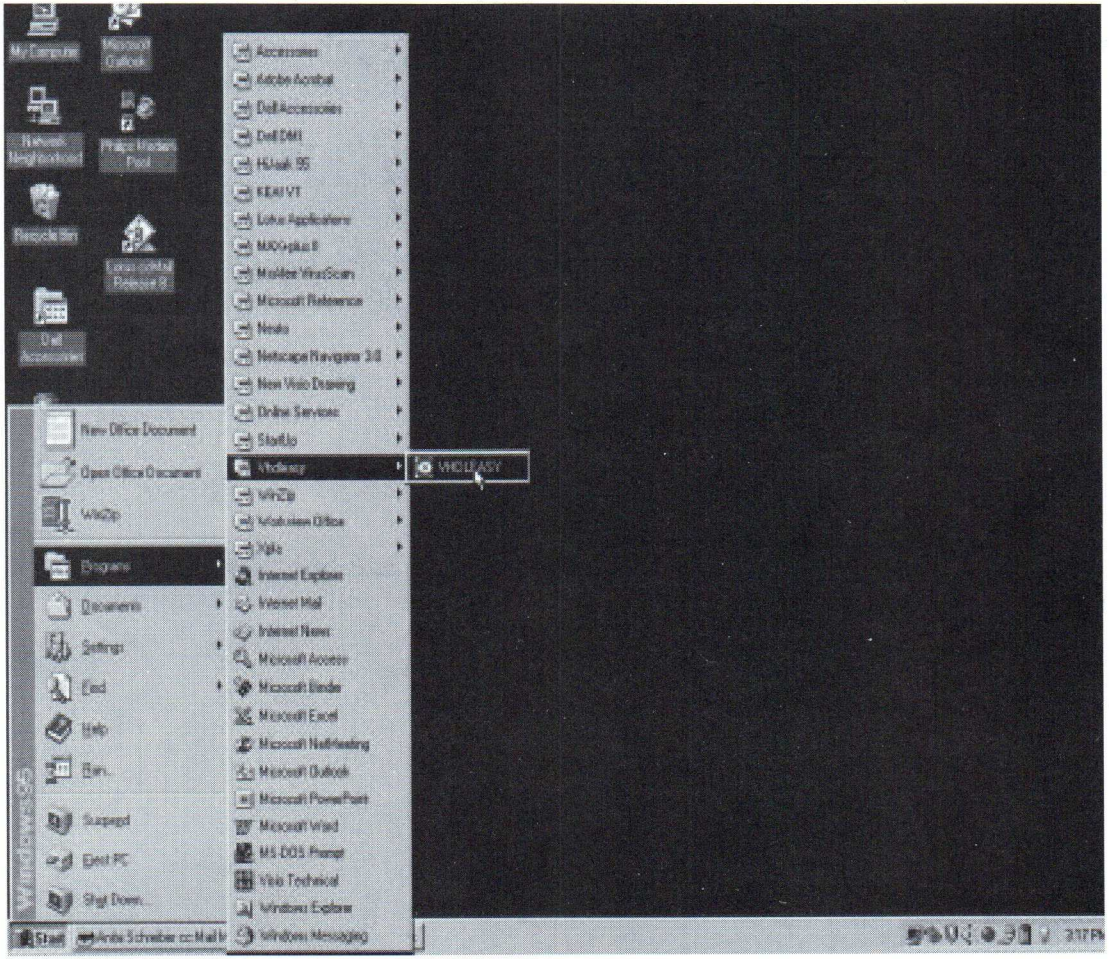


Figure 1:Starting VHDL Easy

To start VHDL Easy, double-click on the VHDL Easy Icon or select **Start | Programs | VHDL EASY.**

This will bring up the VHDL Easy front panel. Click on **Select File** to open the VHDL design.

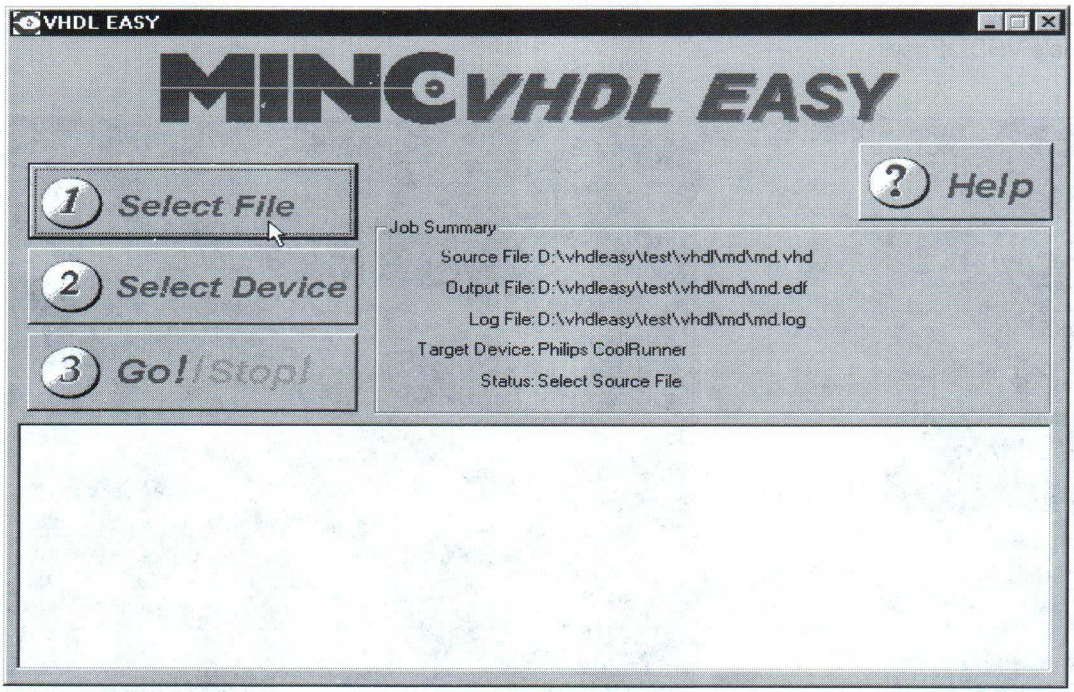


Figure 2:VHDL Easy Front Panel

This will bring up a file choice window that will allow the design to be opened. Once the folder is opened that contains the design file, double-click on the design file.

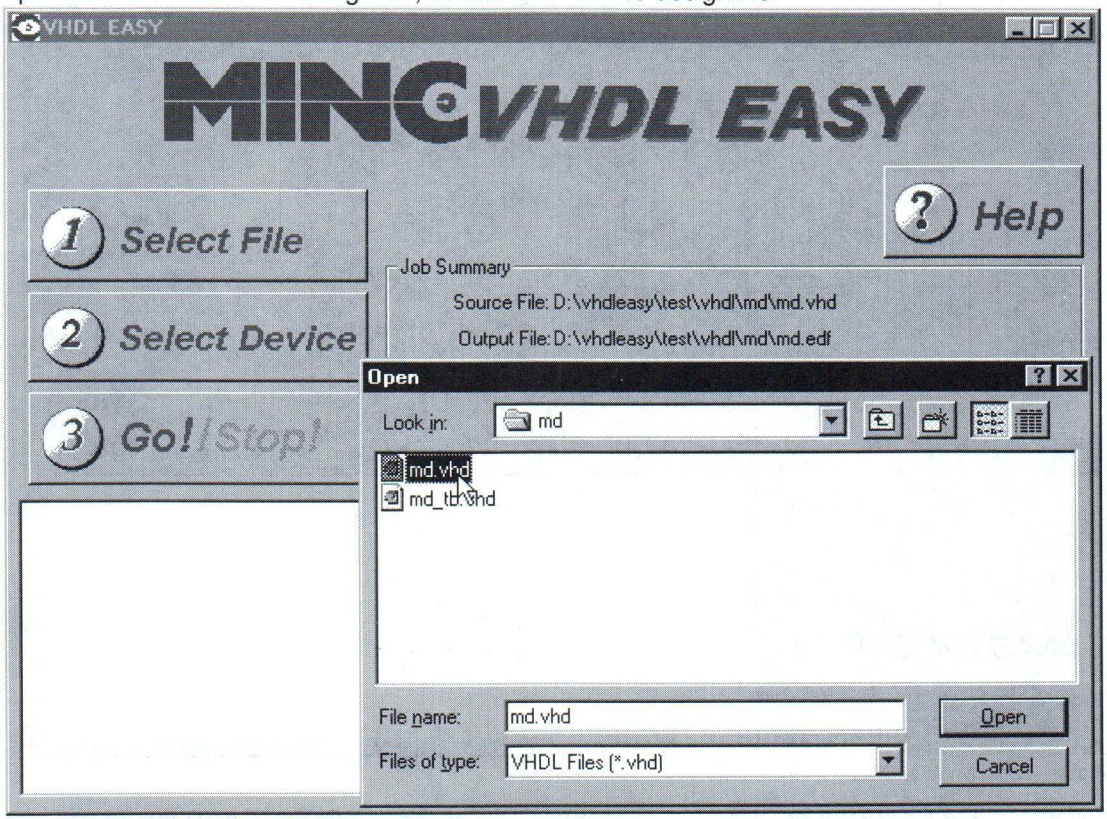


Figure 3:Opening the design file with VHDL Easy

The **Job Summary** area on the VHDL Easy front panel displays **Source File**, **Output File**, **Log File**, and **Status** information. **Source File** is the design file you selected. **Output File** is the device-specific netlist output file VHDL Easy will create after synthesis. **Log File** is a file containing synthesis build information and is created after synthesis. **Status** indicates the current activity of VHDL Easy.

Status displays **Select Device**. Therefore the next step is to select the Philips CoolRunner CPLD as the target device. Click on **Select Device.** A device family selection menu appears.

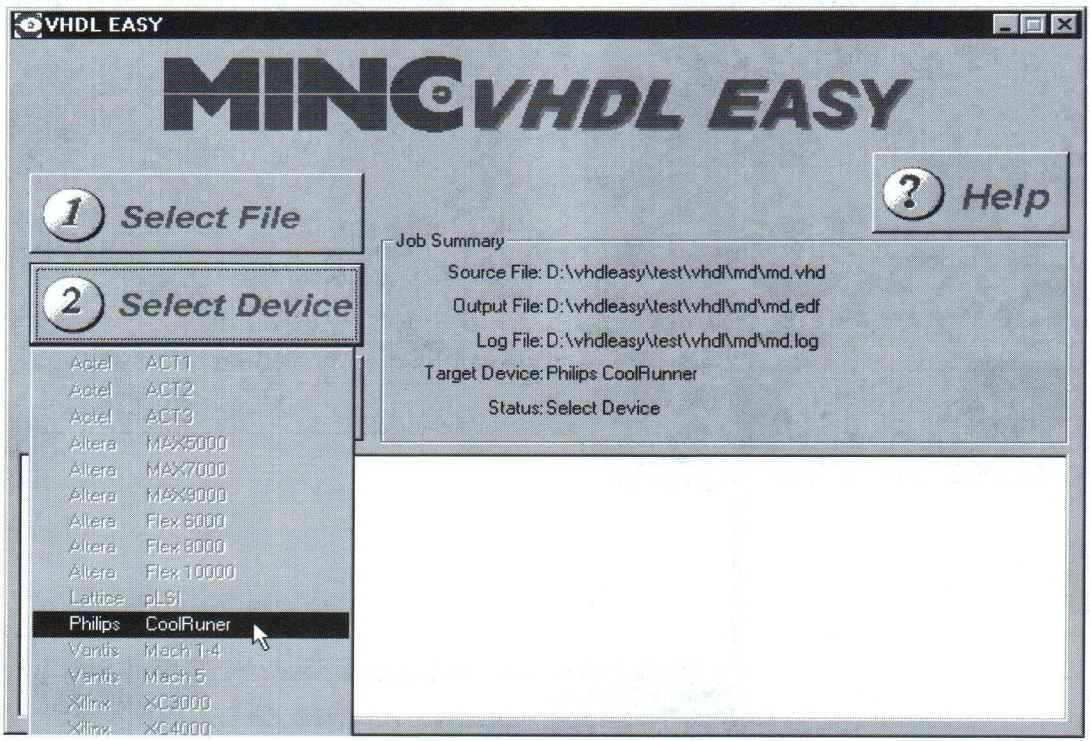


Figure 4:Selecting the CPLD Device Family

Select **Philips CoolRunner**. Note that the actual CPLD within the Philips CoolRunner family is not selected at this time. The actual device for the design will be selected within XPLA Designer. **Target Device** in the **Job Summary** area on the VHDL Easy front panel has been updated to show **Philips CoolRunner**.

The design is now ready to be synthesized. Click **Go!** to start the synthesis.



Figure 5:Starting the VHDL Easy synthesis process

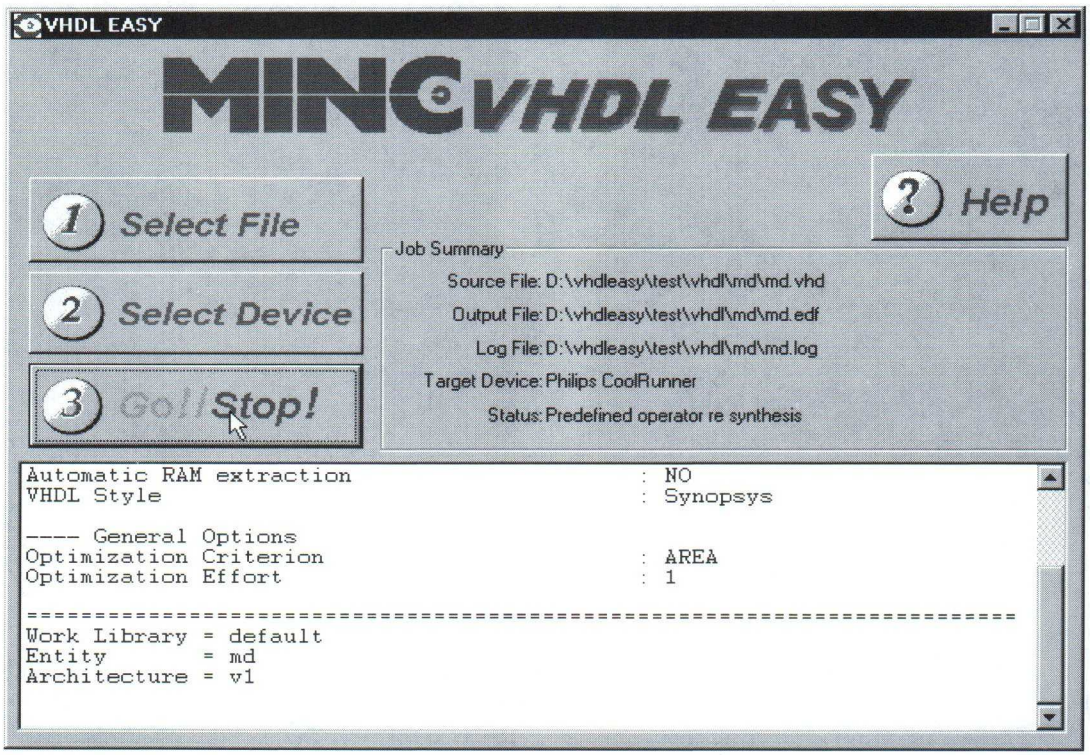


Figure 6:VHDL Easy design synthesis

As the design is synthesized, the status of the synthesis process is displayed in the lower window of the VHDL Easy front panel. Any errors or problems that occur during the design synthesis are displayed in this window and are also written to the log file. If the synthesis does not complete successfully, check the log file for details, correct the source design, and re-synthesize. Please refer to the VHDL Easy User Manual for details about the log file.

Notice that **Stop!** is now highlighted in red. Clicking **Stop!** halts the synthesis process.

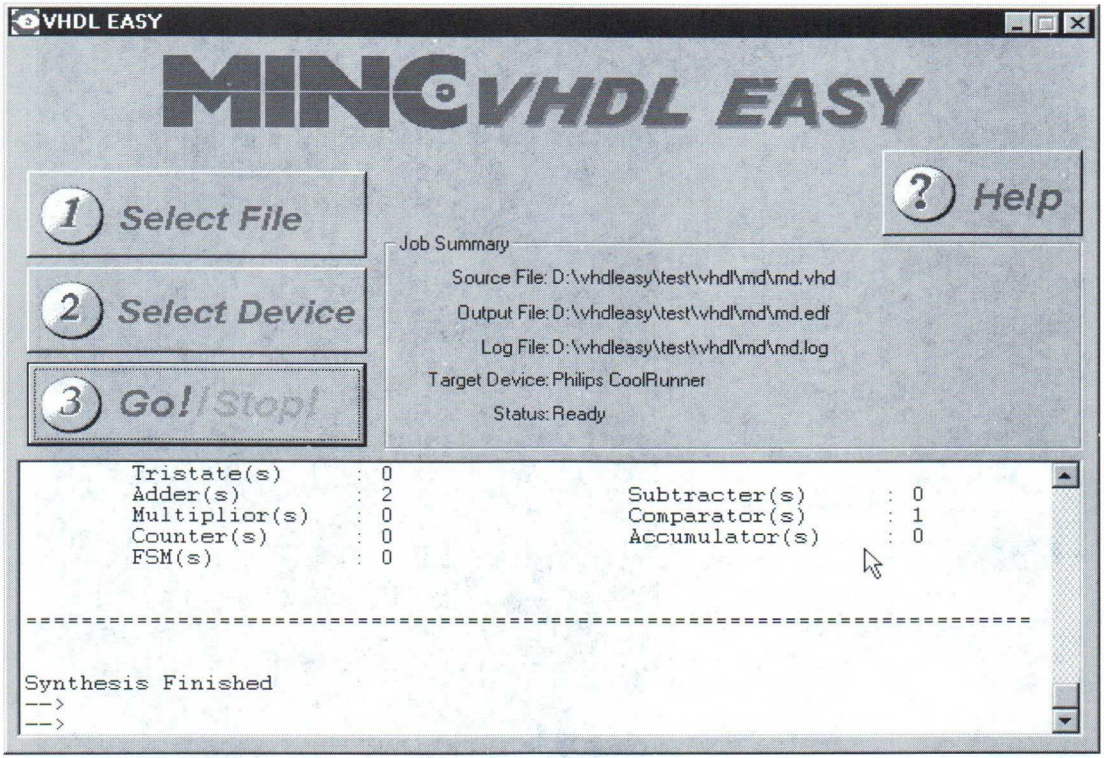


Figure 7:Synthesis Complete

When the design synthesis successfully completes, the lower window of the VHDL Easy front panel shows that the synthesis is finished. **Stop!** is no longer highlighted in red, but is greyed-out. The design has now been synthesized, and the EDIF file, md.edf, has been created. This will be the file that is input into XPLA Designer to create the JEDEC file for the device.

To start XPLA Designer, double-click on the XPLA Designer Icon or select **Start | Programs | XPLA | XPLA Designer.**

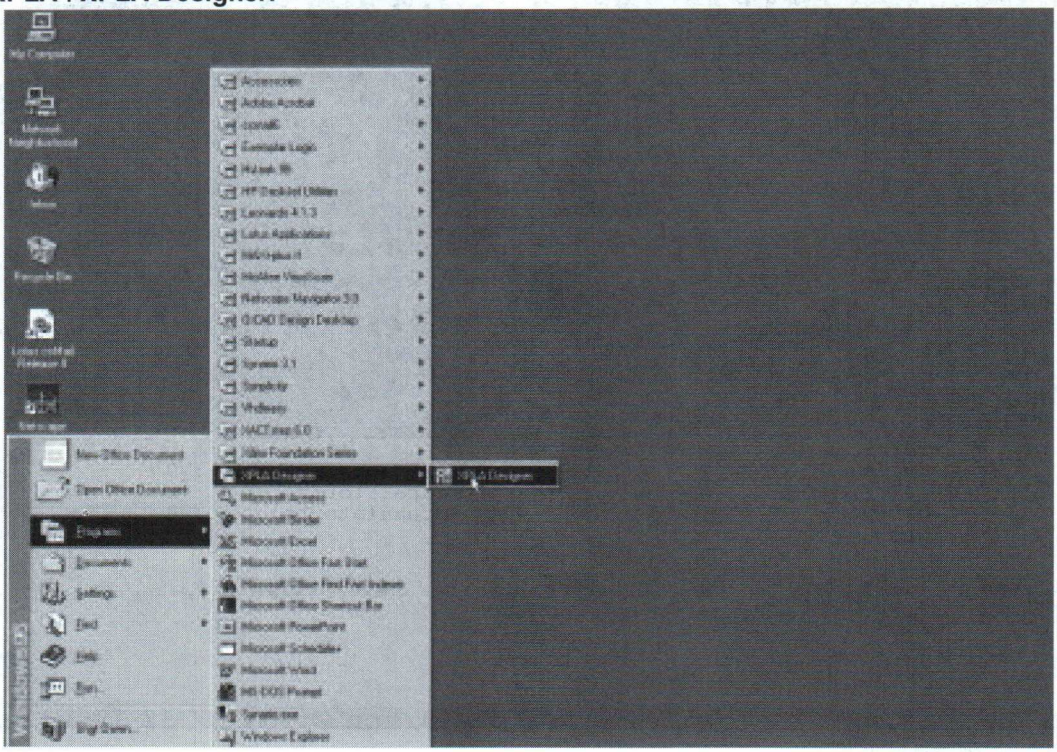


Figure 8:Starting XPLA Designer

This will bring up the XPLA Designer front panel.

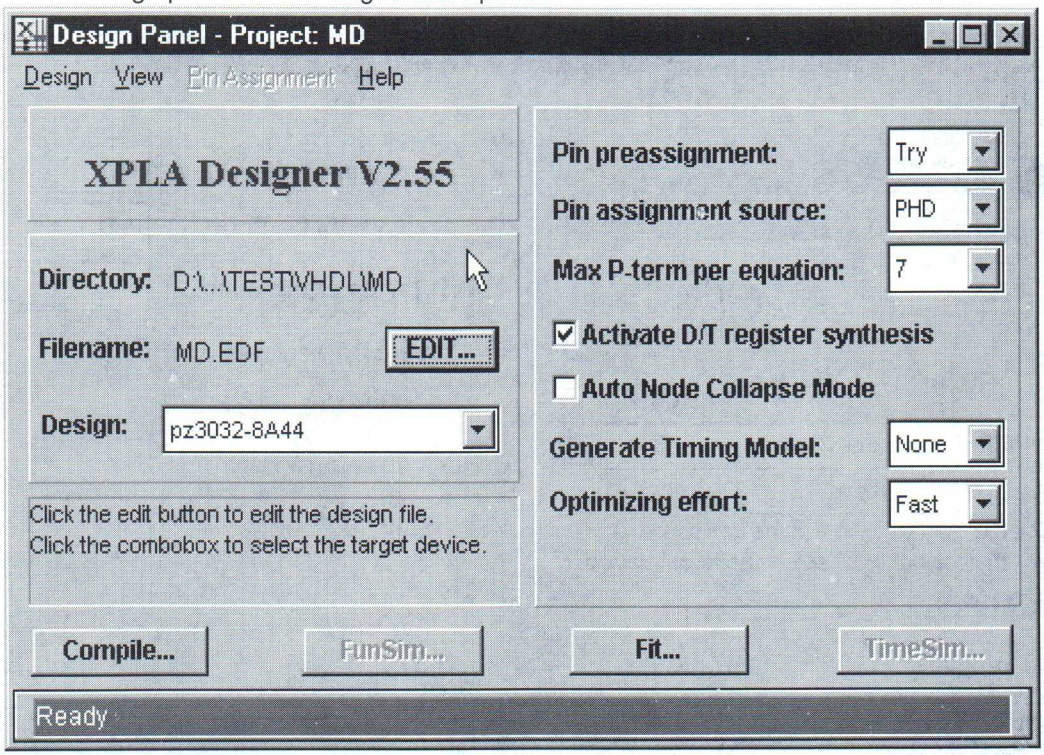


Figure 9:XPLA Designer Front Panel

The first step is to open the EDIF file created by VHDL Easy.

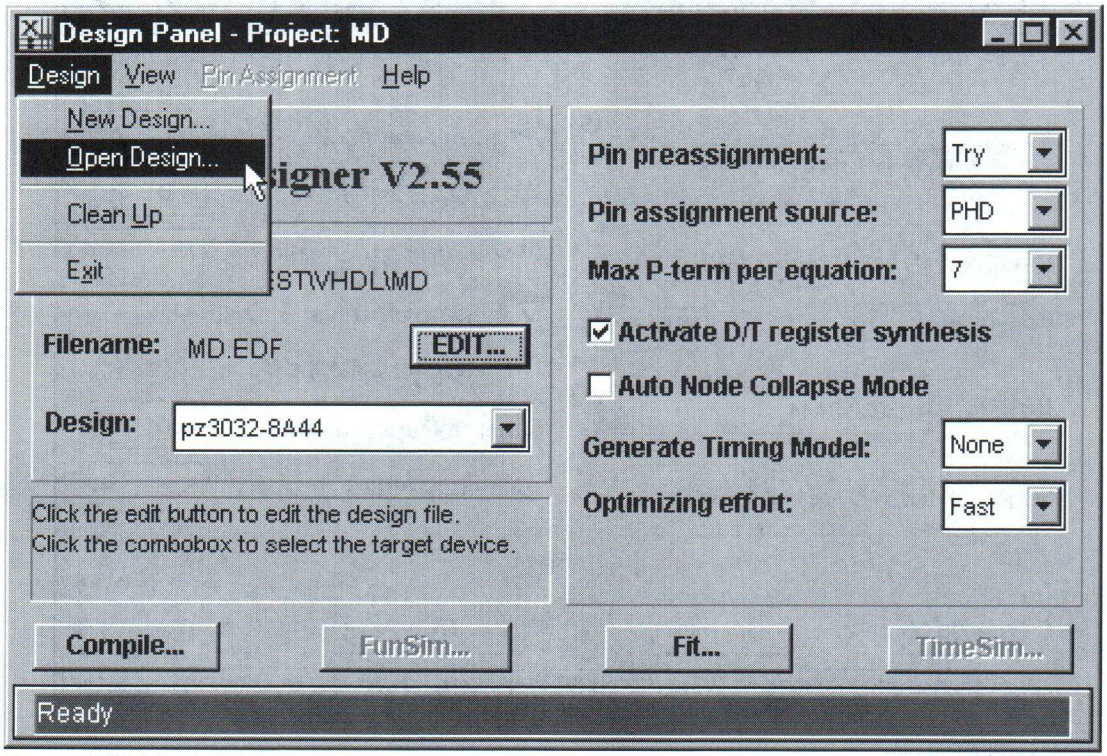


Figure 10:XPLA Design Menu

Click on **Design** I **Open Design**.

This will bring up a dialog box that allows you to select the format of the design and the design to be opened.

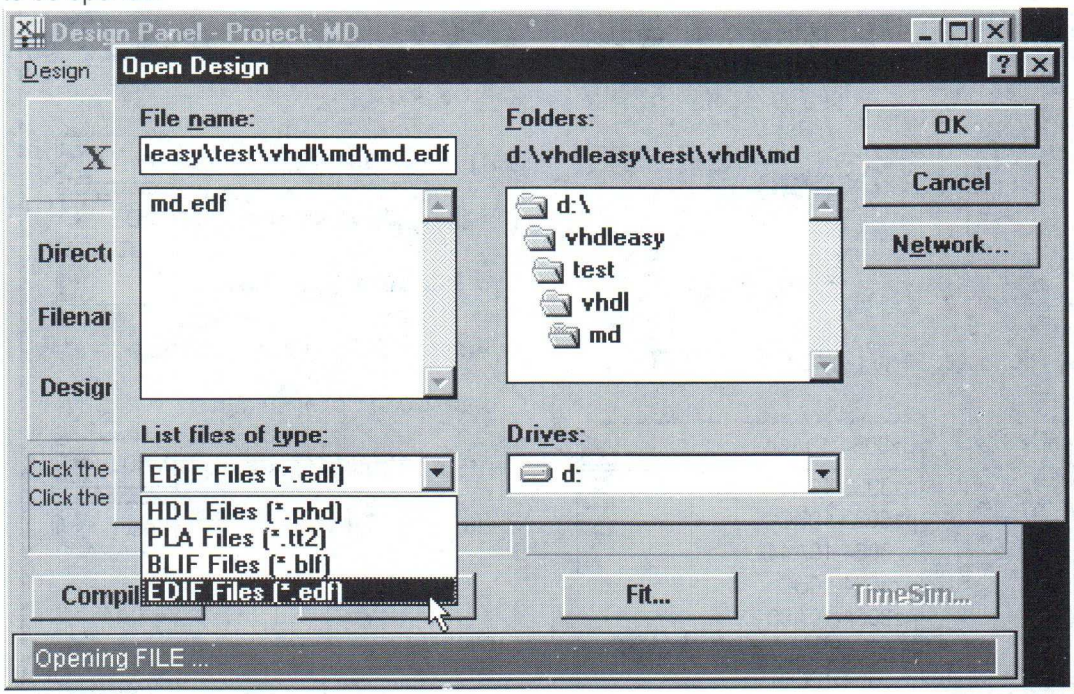


Figure 11:Selecting XPLA Designer Source File

Select the folder that contains the design file. Under **List files of type:** select **EDIF Files (*.edf)**. This will display all files of that type in the selected folder. Click on **md.edf** and then click **OK**. The design has now been selected as the source file for XPLA Designer.

To select the specific CPLD within the Philips CoolRunner family, select **Design:**.This will bring up a device selection menu.

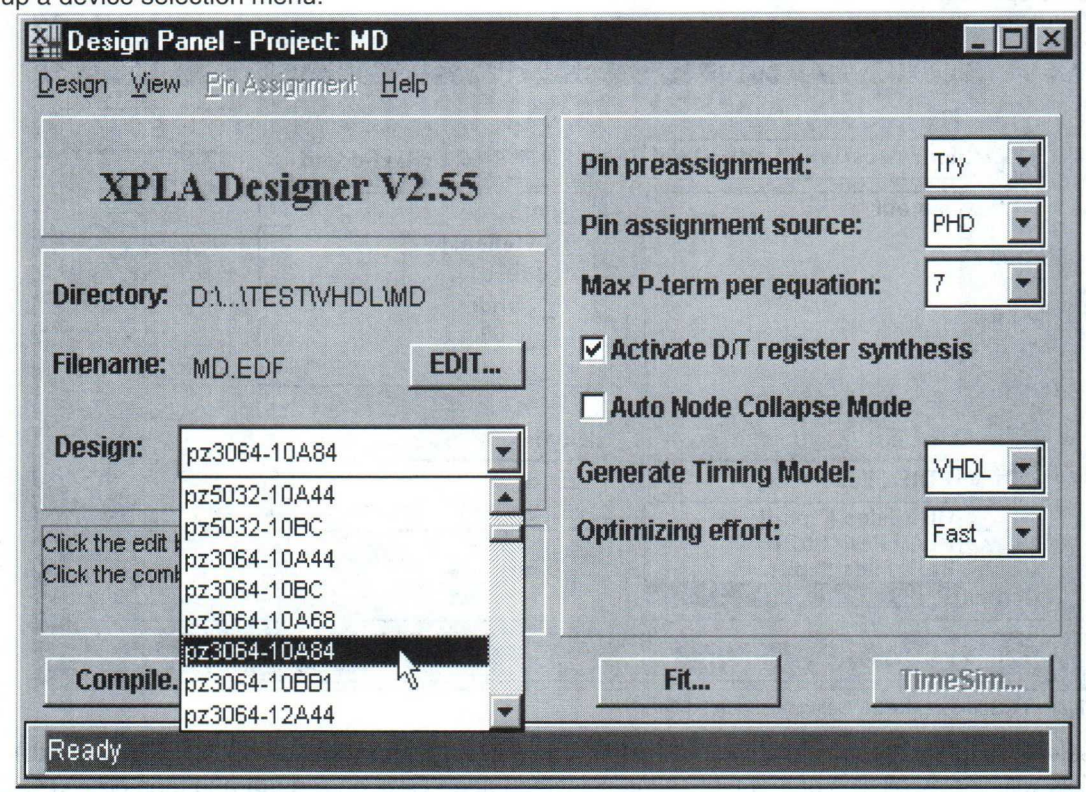


Figure 12:Selecting the device in XPLA Designer

Select the device in the voltage, speed grade and package desired. For this example, a 3V, 10ns TPD 64macrocell device in an 84 pin PLCC package has been chosen. Note that a device must be selected, there is not a selection to have XPLA designer chose the device based on what is required to fit the design. See Appendix A for a complete description of device part numbers to aid in this selection process.

Since the source of the design is a VHDL file, pins are not assigned. It is necessary to do an initial compilation and fitting of the design to create a Pin Assignment File (PAF). Once this file has been created, the Pin Assignment Editor can be run to change the pin-out if required.

Click on Fit... to do an initial fit of the device and create a PAF file.

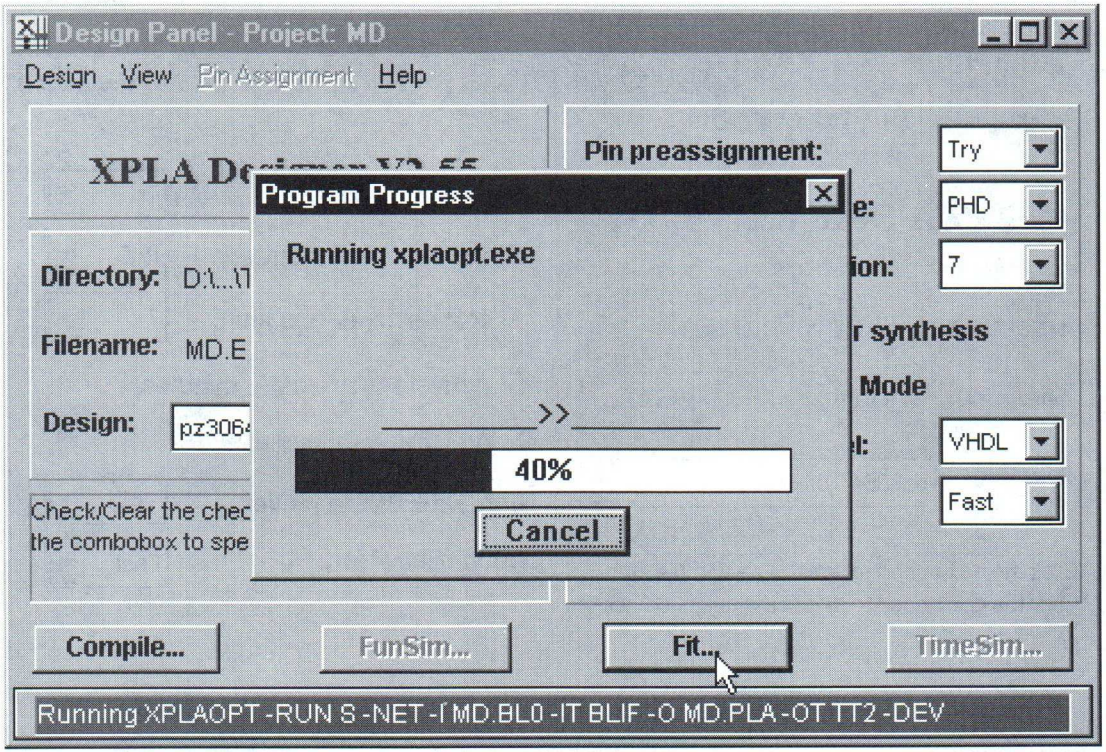


Figure 13:Doing an initial fit in XPLA Designer

This will run the compiler and the fitter with the default options. XPLA Designer will assign pins and write these assignments to a PAF. This step is necessary to generate the PAF and allow the Pin Assignment Editor to be executed.

Notice that after the initial fit has completed, Pin Assignment is no longer greyed-out on the top menu panel of XPLA Designer.

Start the Pin Assignment Editor by clicking on **Pin Assignment**. The Pin Assignment Editor can only be started if a PAF exists.

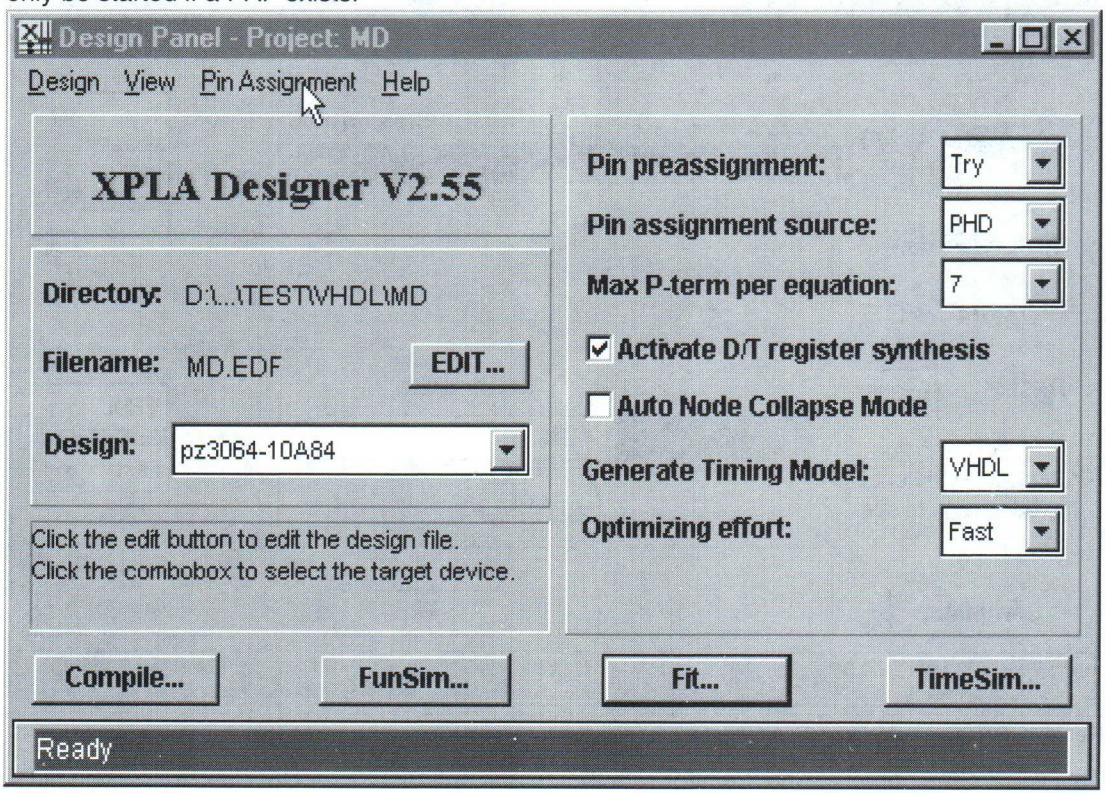


Figure 14:Starting the Pin Assignment Editor in XPLA Designer

This brings up the XPLA Pin Assignment Editor.

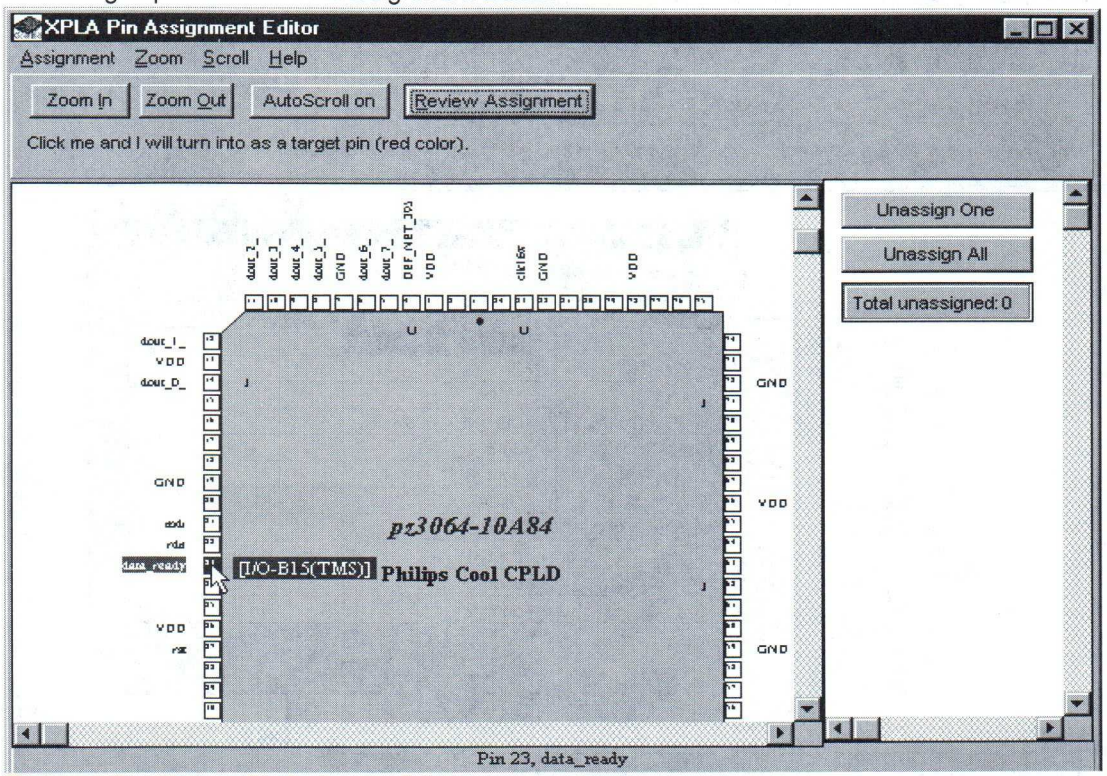


Figure 15:XPLA Designer Pin Assignment Editor

The Pin Assignment Editor allows the designer to assign pins, unassign pins, swap pins, and review current pin assignments.

If the pin-out of the device is already determined by the system requirements, the pins can now be assigned using the Pin Assignment Editor. Clicking **Unassign All** deletes assigned pins from the package pins and moves them to the right window. To assign a pin, click on the pin name in the right window and click again on the desired location on the package. From the menu bar, select **Assignment I Save**. This will write the current pin assignments to the PAF.

To review current pin assignments, click on **Review Assignment**.

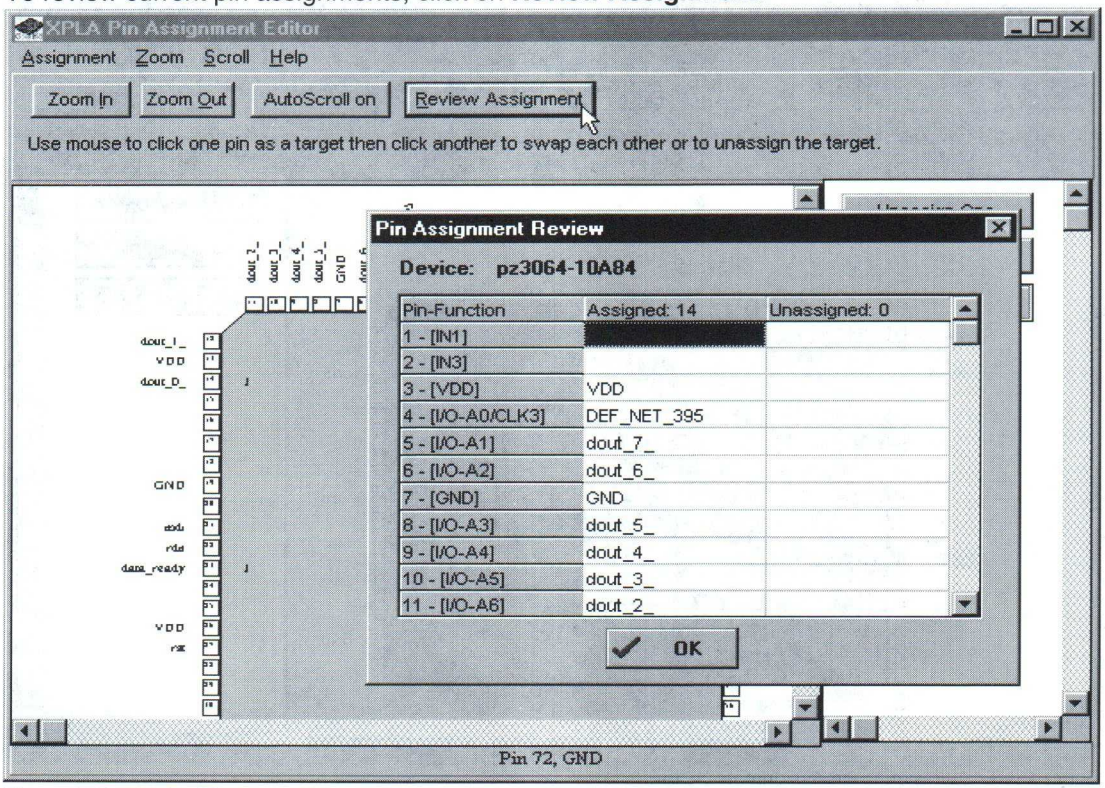


Figure 16:Reviewing Pin Assignments in XPLA Designer

This table can be used to review and change the pin-out of the device.

XPLA Designer allows the designer to specify whether a pin pre-assignment should be kept, ignored, or whether the software should try the previous assignment.

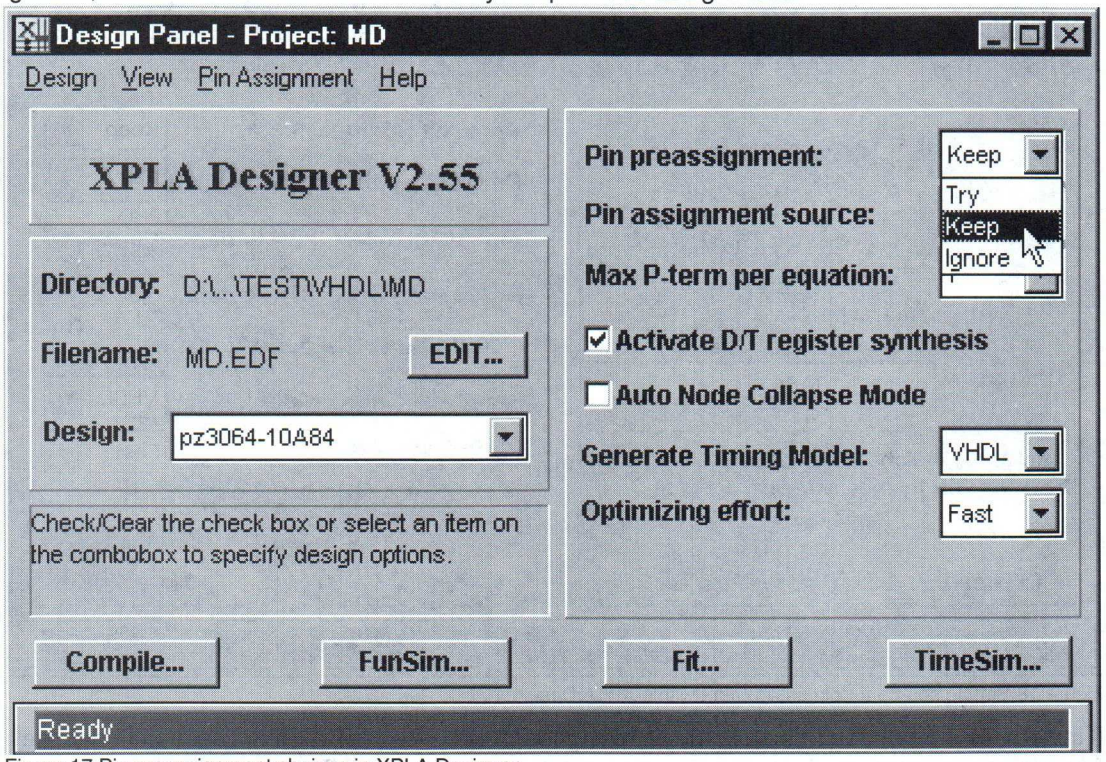


Figure 17:Pin preassignment choices in XPLA Designer

The **Try** selection will attempt to fit the design with the pin assignments specified in the **Pin Assignment Source** field. If the design can not be fit with these pin assignments, the fitter will remove the pre-assigned pins and attempt to fit the design with no pre-assigned pins. A warning message will tell the user if the pre-assigned pins have been removed. If the device then fits, the fitter assigns pins and writes these assignments in the Pin Assignment File (PAF).

The **Keep** selection will attempt to fit the design with the pin assignments specified in the **Pin Assignment Source** field. If the design can not be fit with these pin assignments, the fitter will notify the user that the device could not fit. It will not unlock the pins under this option.

The **Ignore** selection will attempt to fit the design and will ignore the pin assignments specified in the **Pin Assignment Source** field. If the design fits with no pre-assigned pins, the fitter will assign pins and write these assignments in the Pin Assignment File. (PAF).

Once the PAF file has been generated, it can be used as the source for pin assignment.

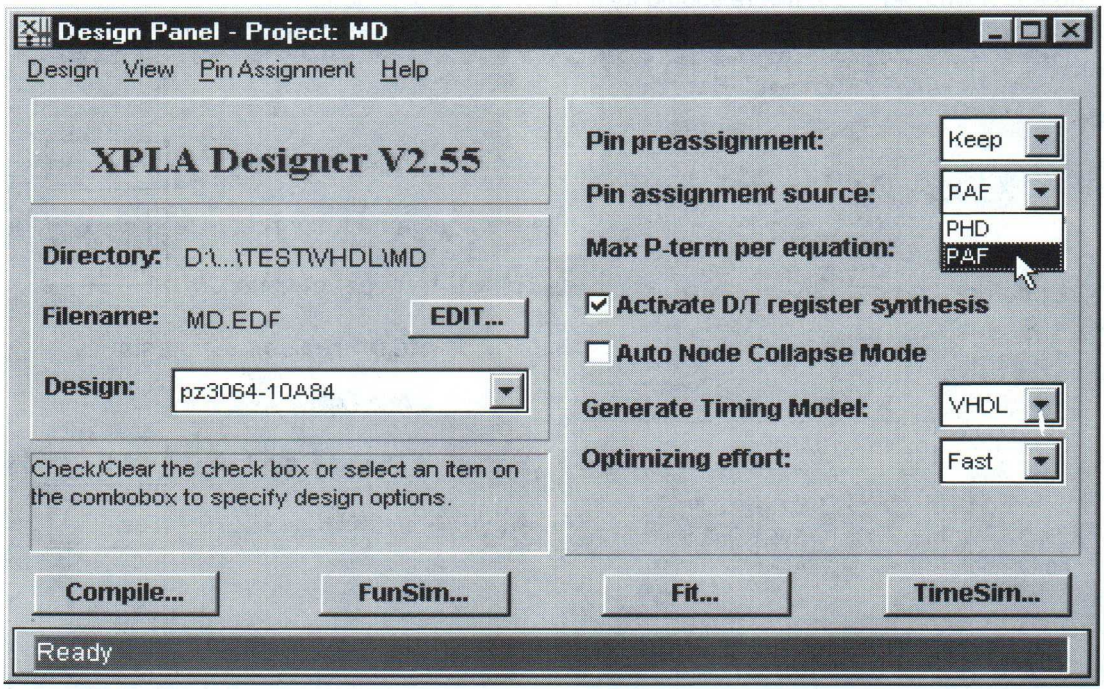


Figure 18:Setting source for pin assignment within XPLA Designer

If the device pin-out determined in the pin assignment editor or the PAF must be maintained, set **Pin preassignment:** to **Keep** and **Pin assignment source:** to **PAF**.

Please refer to the **XPLA Designer User's Manual** (available at www.coolpld.com) for more detailed information on setting these options.

Other options on the XPLA designer can be set to control the optimization and fitting processes.

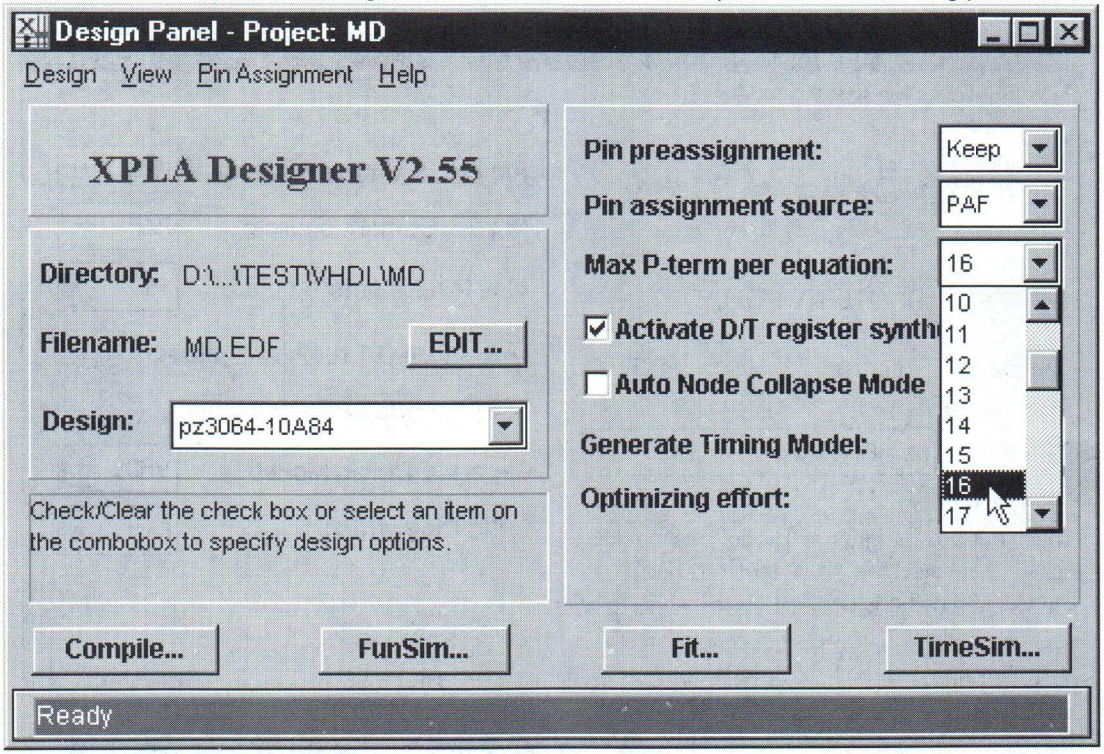


Figure 19:Setting Max P-term per equation in XPLA Designer

Max P-term per equations: can be set to any number between **5** and **37**. This allows the designer to control how wide logic functions are implemented and trade speed for density. Wide logic functions can be implemented in two different manners: using more PLA product terms or using more than one pass through the logic array. Each macrocell in the XPLA architecture has 5 dedicated PAL product terms and has access to 32 PLA product terms. This parameter can determine if PLA terms are used to implement wide logic functions, or if only PAL product terms are used with more than one pass through the logic array.

Implementing wide logic functions using 1 or up to 32 of the PLA product terms will add an additional delay (see Philips CPLD Data Handbook for PLA delay time). Implementing wide logic functions using multiple passes through the logic array add a Tpd delay for every additional logic array pass. If speed is the most important parameter, the design should specify a larger value for this field. If fitting the design is perceived to be an issue (density problem) and speed is not as important, then the user should specify a smaller value for this field.

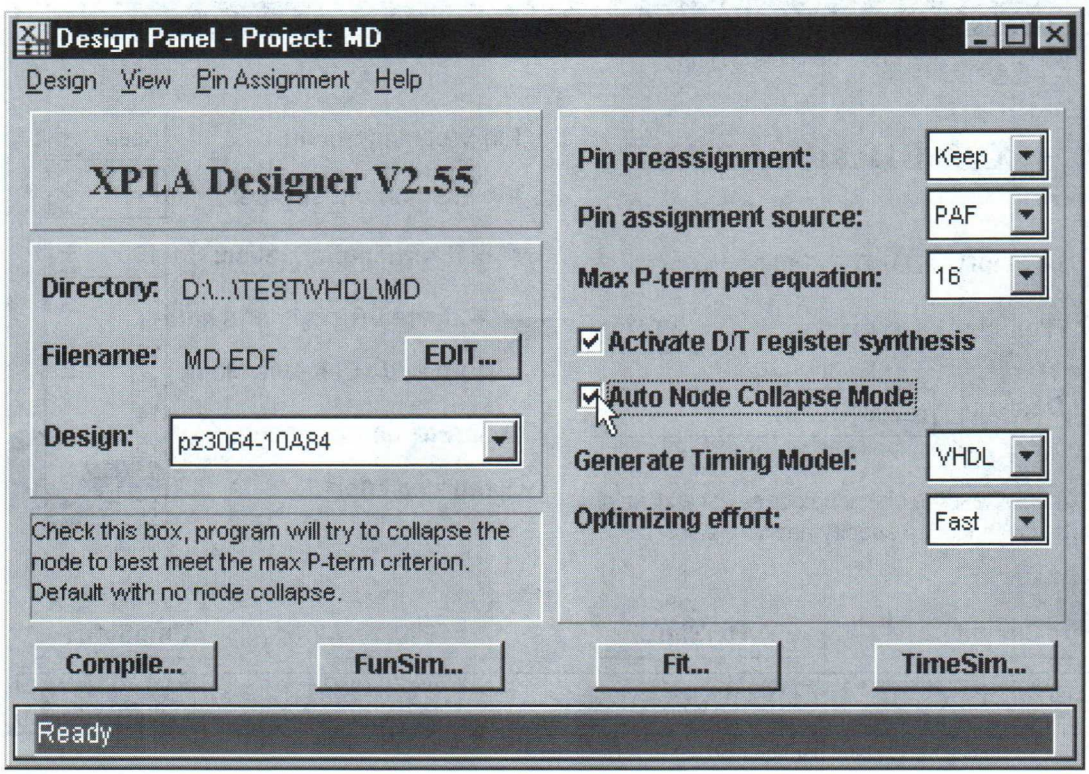


Figure 20:Activating D/T Register Synthesis and Auto Node Collapse Mode in XPLA Designer

Activate D/T register synthesis allows the software to optimize certain designs to T -type flip-flops. This selection instructs XPLA Designer to minimize the number of product terms using either D or T type flip-flops. When the selection is activated, the software will implement each equation using all D or all T type flip-flops and it will pick the flip-flop type that requires the minimum number of product terms. If the number of product terms are the same with either implementation, the software will default to a D type flip-flop.

If this selection is not activated, the XPLA Designer software will use the type of flip-flop specified in the EDIF source file.

Auto Node Collapse Mode is used to minimize the number of passes through the logic array by collapsing internal nodes into other logic. Collapsing the node frees a macrocell at the expense of using more PLA terms, thus maximizing design performance because one pass through the PAL and PLA arrays is faster than multiple passes through the PAL array.

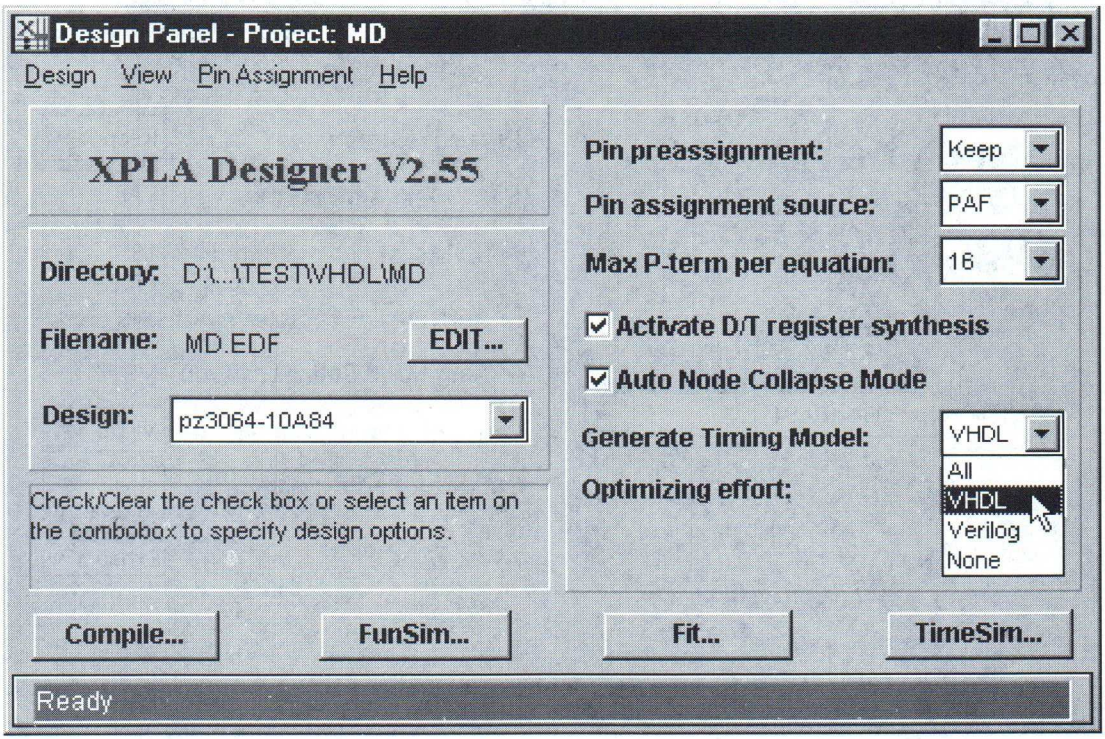


Figure 21:Selecting VHDL as the timing model in XPLA Designer

Generate Timing Model can be set to generate either a VHDL or Verilog model with device timing. This can be set to **All**, **VHDL**, **Verilog**, or **None**. If the device is to be simulated as part of a system or against a set of test vectors, set this option to **VHDL** or **Verilog** or **All**.

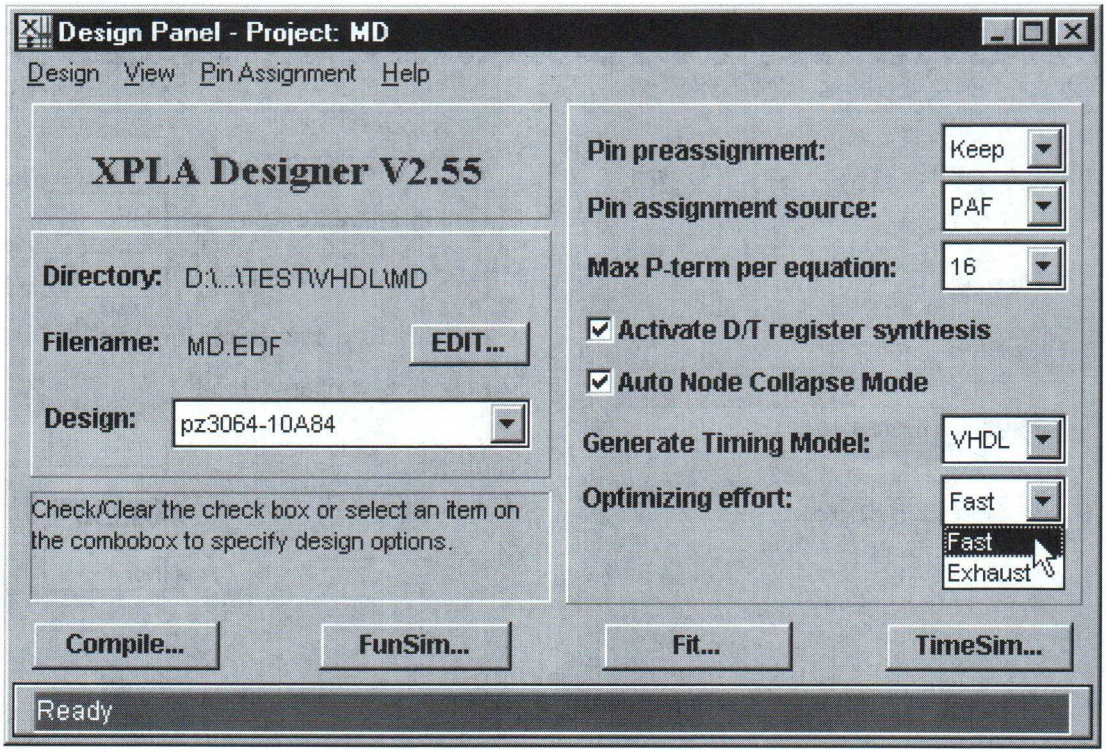


Figure 22:Setting the Optimizing effort in XPLA Designer

Optimizing effort: can be set to **Fast** or **Exhaust** to control program run-time. If the design will not fit with **Optimizing effort:** set to **Fast**, change the selection to **Exhaust** and re-compile.

Once all of the compilation and fitting parameters have been set, click on **Compile...**.

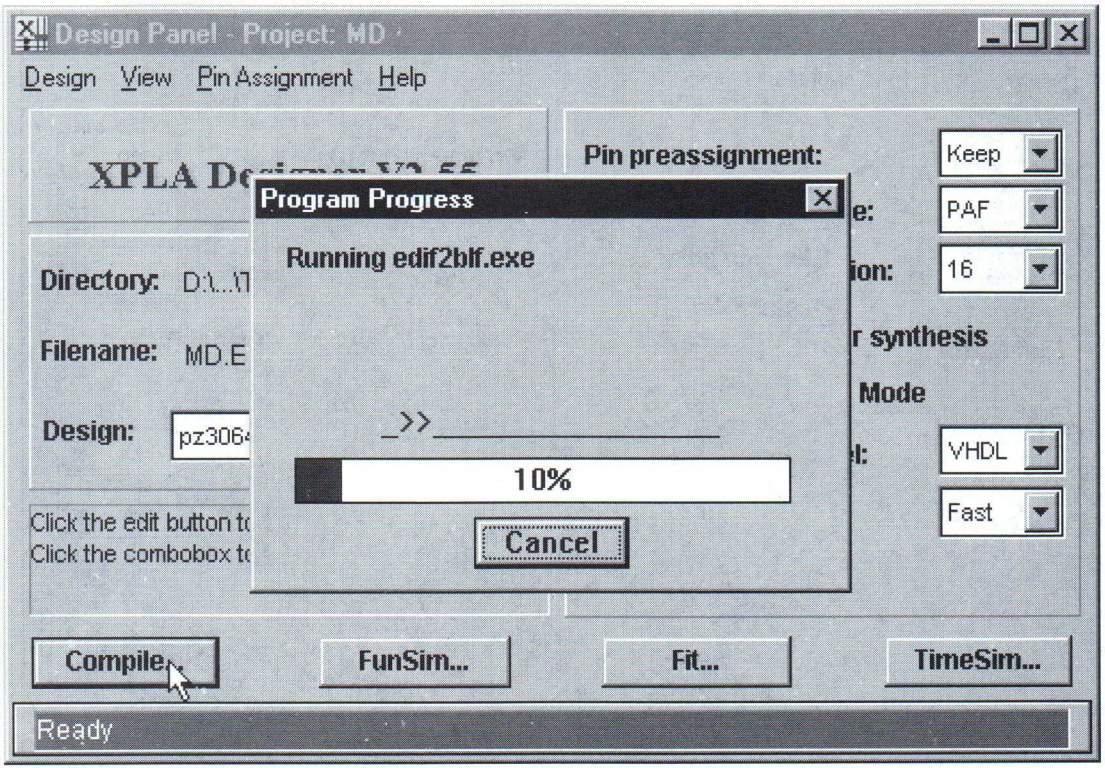


Figure 23:Compiling the design in XPLA Designer

The progress of the compilation program is displayed.

Once the compilation has successfully completed, the design can be functionally simulated within XPLA Designer by clicking on **FunSim...**. It is beyond the scope of this application note to cover the simulation features of XPLA Designer. Please refer to the **XPLA Designer User's Manual** for more information on simulation.

The design is now ready for fitting into the selected device. Click on **Fit...**

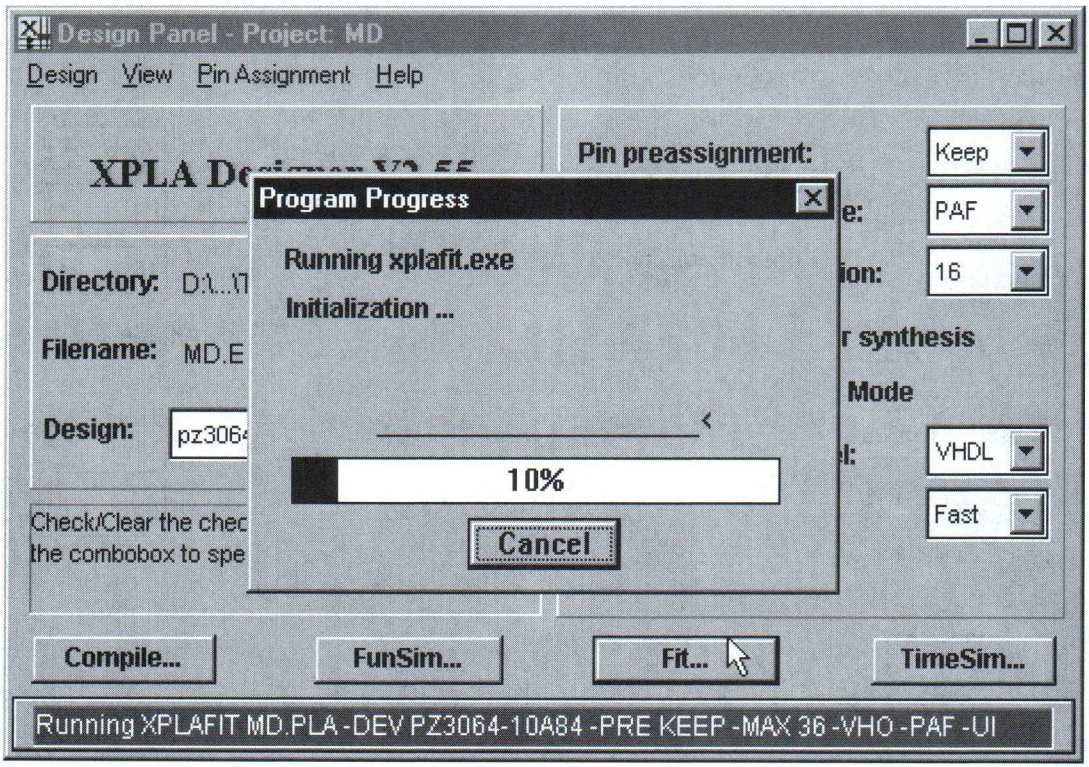


Figure 24:Fitting the design in XPLA Designer

The progress of the fitting program is displayed.

When the device has successfully fit into the selected the device, the following is displayed:

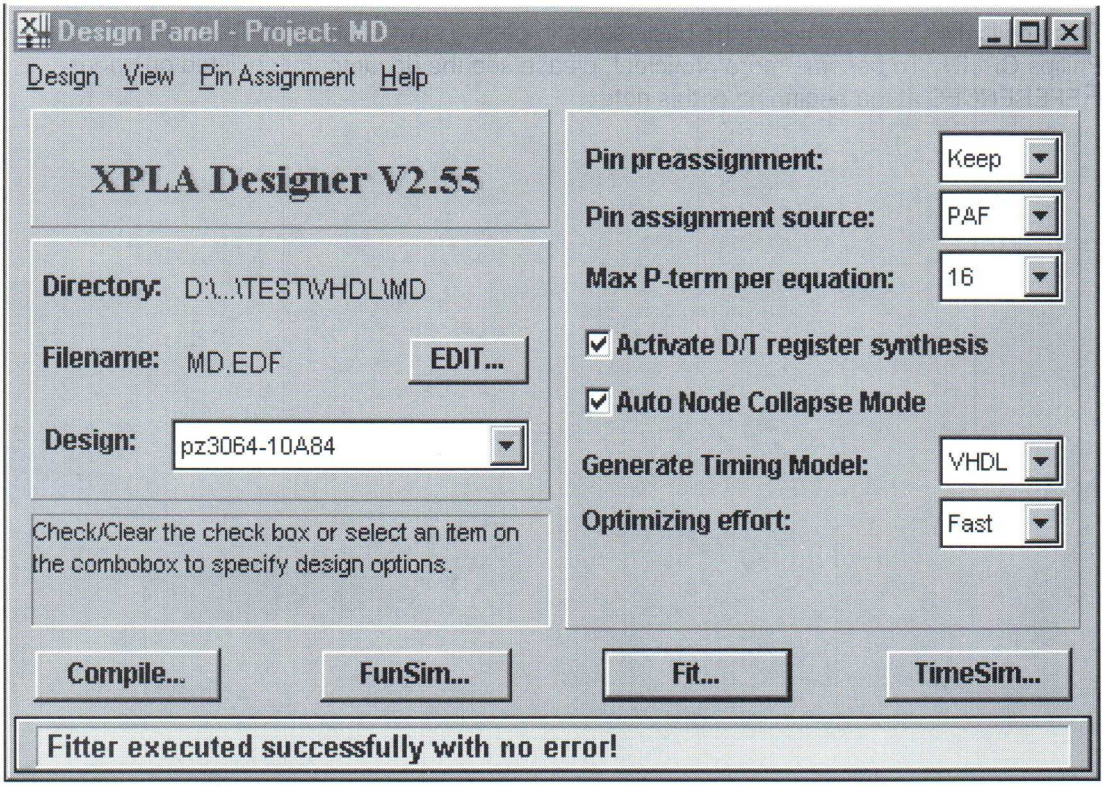


Figure 25:Successful completion of XPLA Designer

At this point, the VHDL or Verilog timing models can be viewed as well as the annotation and fitter reports. The JEDEC file has been created and resides in the project directory.

CONCLUSION

This application notes provides the basic steps in getting started using VHDL Easy to target Philips CPLDs. To become more proficient, please see the documentation listed on under REFERENCES at the beginning of this note.

APPENDIX A- PHILIPS CPLD PART NUMBERING SYSTEM

PZXYYYK(S)ZZYYY					
X = SUPPLY VOLTAGE	**YYY = MACROCELL COUNT**	**K = OPERATING TEMPERATURE**	**S = ISP**	**ZZ = SPEED GRADE (TPD)**	**YYY= PACKAGE DESIGNATOR**
3 = 3.3V 5=5V	32 64 128 320 960	- = Commercial I = Industrial C = Commercial, enhanced clocking N = Industrial, enhanced clocking A = Commercial, enhanced clocking, 0.35 micron process with 5 V tolerant I/Os D = Industrial, enhanced clocking, 0.35 micron process with 5 V tolerant I/Os		6 = 6ns 7=7.5ns 8=8ns 10=10ns 12=12ns 15=15ns	A44 = 44 pin PLCC A68 = 68 pin PLCC A84 = 84 pin PLCC BB1 = 100 pin PQFP BB2 = 160 pin PQFP BC = 44 pin TQFP BE = 128 pin LQFP BP = 100 pin TQFP EB = 492 pin PBGA

Viewlogic Intelliflow Design for Philips CPLDs

AN079

INTRODUCTION

This note provides the steps for using Viewlogic Intelliflow[1] to simulate and compile a digital design into Philips' Complex Programmable Logic Devices(CPLDs).
This design is generated using Viewlogic's Viewdraw to capture the schematic of a four -bit counter. Intelliflow is then used to optimize and fit the design into a Philips' CPLD. Designers are strongly encouraged to run the Viewlogic Intelliflow tutorial before attempting a design.

Technical support for the design flow described in this application note is provided by:

Philips Technical Assistance

Telephone: 1-888-coolpld

E-Mail: coolpld@abq.sc.philips.com

Web Site: http://www.coolpld.com

Fax: 1-505-822-7804

Fax on Demand: 1-800-282-2000

REFERENCES

XPLA Designer User's Guide

Philips Complex Programmable Logic Devices Data Handbook

Viewlogic Intelliflow References

INSTALLATION REQUIREMENTS

This design requires that the following PC-based software has successfully been installed and all authorization files have been received:

Viewlogic Workview Office V7.4

Viewlogic Intelliflow/ViewPLD with latest V7.4 patches

Philips' CoolRunner optimizer and fitter

The Philips CoolRunner series can be targeted using a Viewlogic schematic with the DIO symbol libraries, or by ABEL source code. This four-bit counter design targets the Philips PZ3032 CPLD using a Viewlogic schematic.

(1) Philips acknowldeges the trademarks of the companies mentioned in this document.

DESIGN FLOW

The Viewlogic Intelliflow design flow starts with setting up a Viewlogic project. It is assumed that the reader knows how to start Workview Office and is somewhat familiar with its basic functionality.

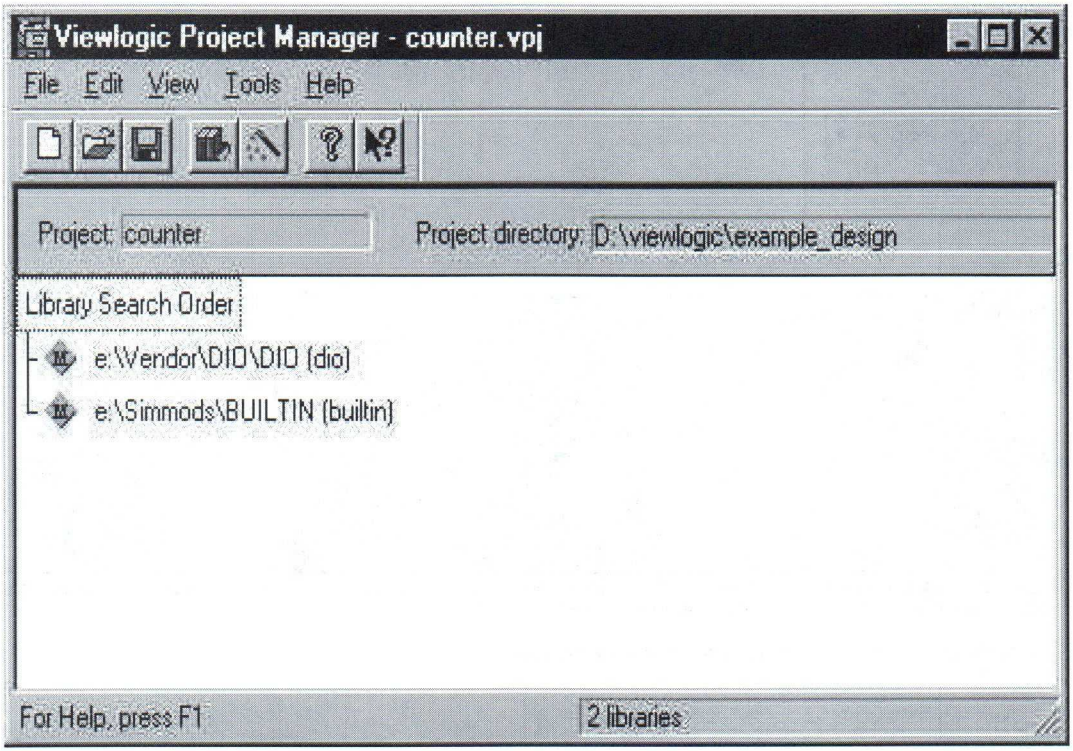


Figure 1:Creating a project in Viewlogic

Create a project for your design. The libraries necessary for this project are the DIO library and the Builtin library.

Now start Intelliflow.

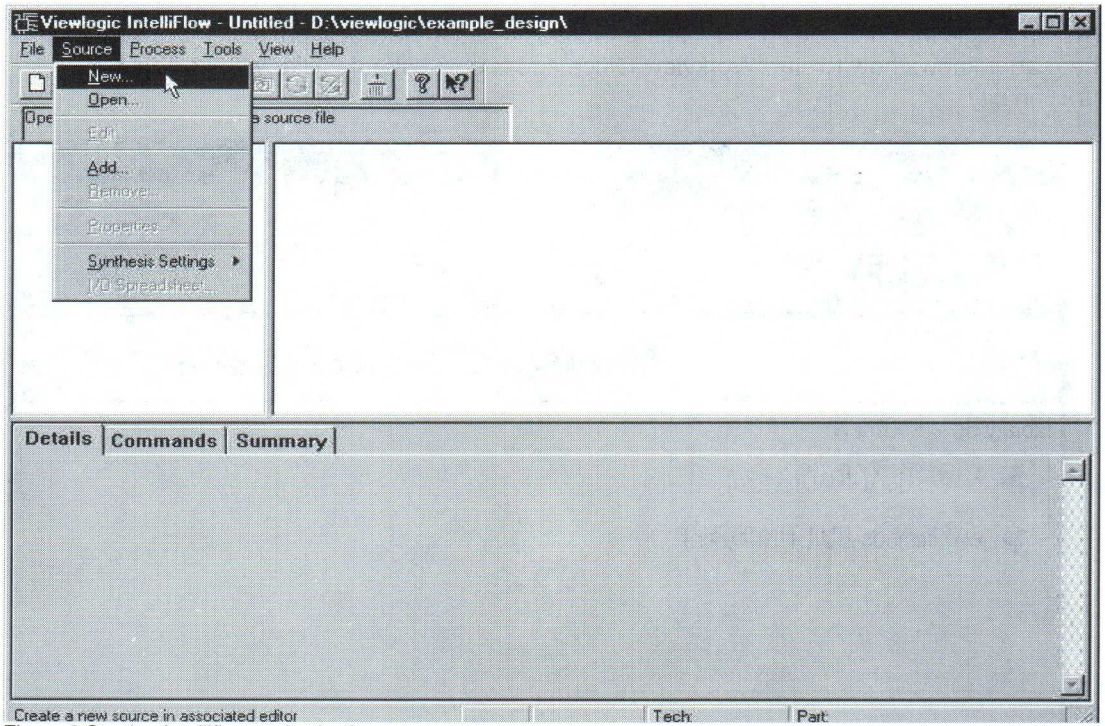


Figure 2:Starting Intelliflow and selecting a new source

Select **Source** I **New** to create a new source file.

This will bring up a choice window that allows you to select the source file type.

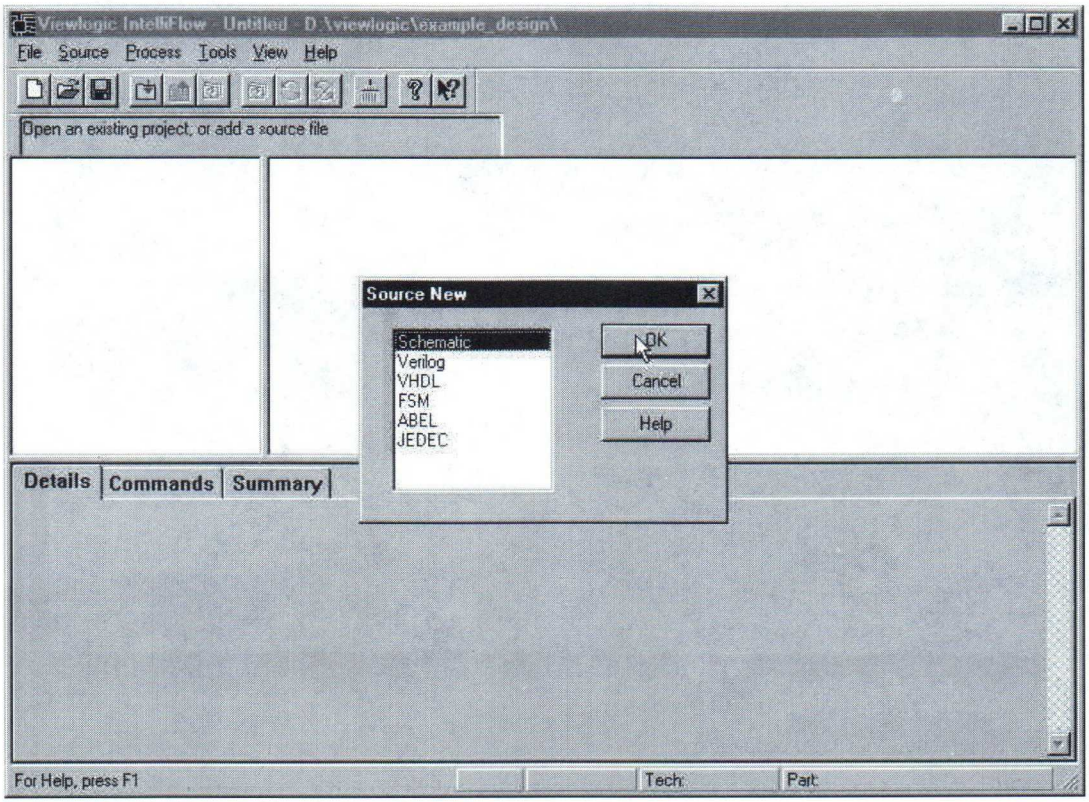


Figure 3:Selecting Schematic as the type of source file

Select **Schematic** and **OK**.

This will start Viewdraw.

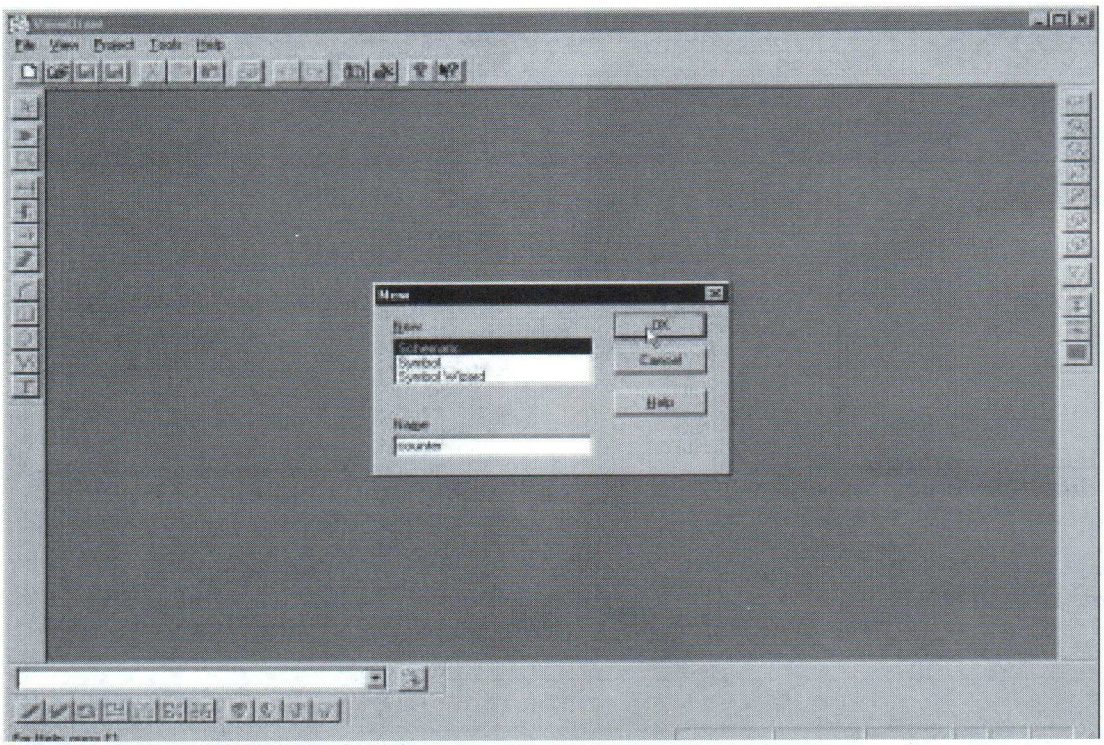


Figure 4:Starting a new Viewdraw schematic

Select **File | New**. A dialog box appears. Select **Schematic** and **OK**.

Symbols for the schematic will come from the DIO or Builtin libraries.

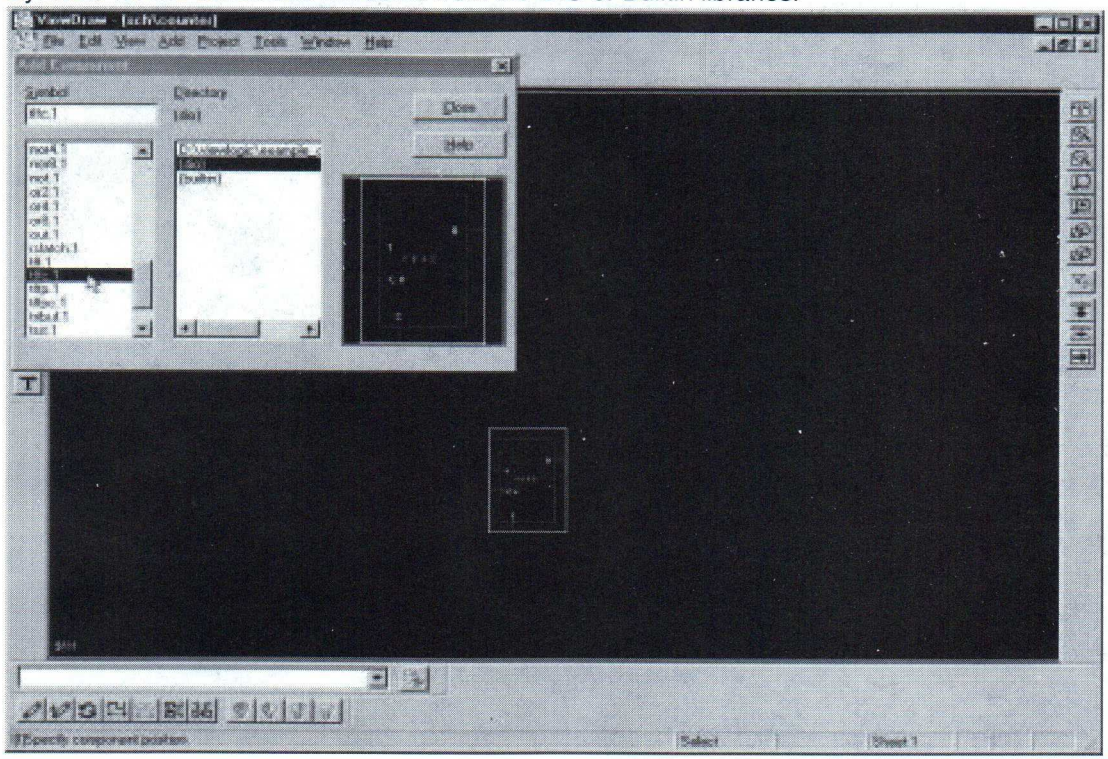


Figure 5:Selecting components for the schematic

Add TFFC flip-flops from the DIO library. PWR, GND, and BI symbols will be in the Builtin library.

Add symbols to complete the schematic.

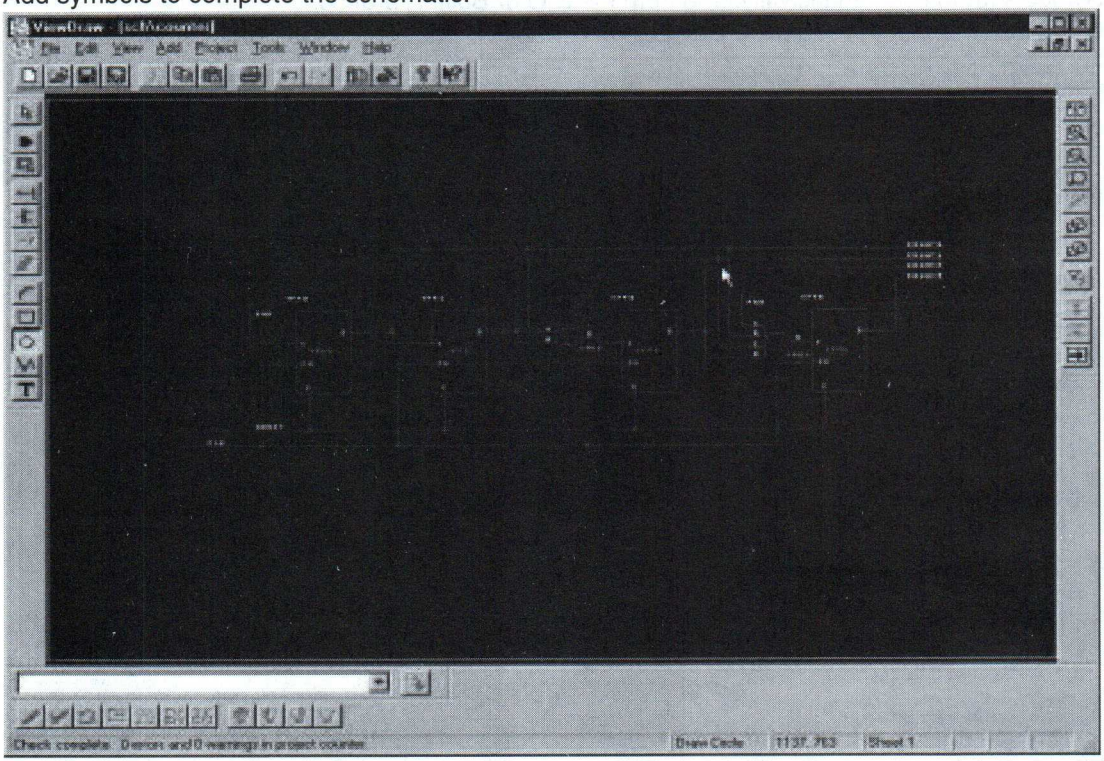


Figure 6:4-bit counter schematic

When the schematic is complete, save and check the schematic to write out the wir files.

Now the schematic needs to be added as the source file in Intelliflow.

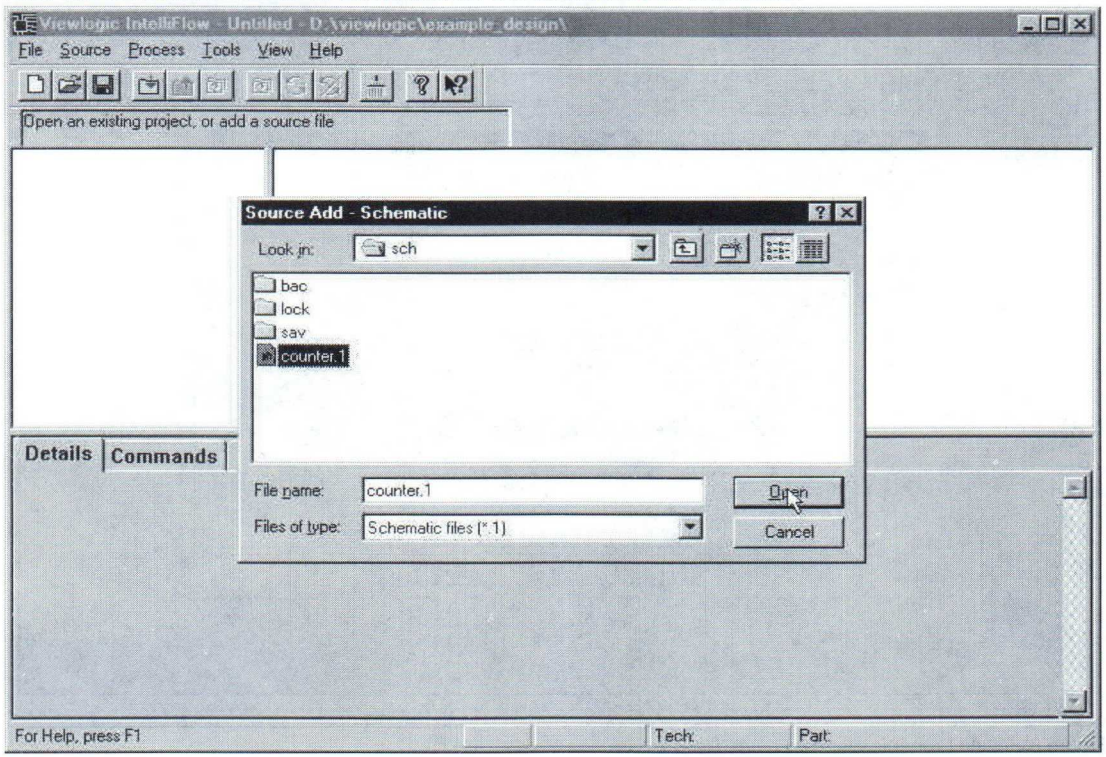


Figure 7:Adding schematic as the source in Intelliflow

Select **Source** I **Add**. Select **Schematic files** for **Files of type:**. Select the schematic just created and click **Open**.

The selection of the Philips' CoolRunner CPLD is done by setting the properties of the source file.

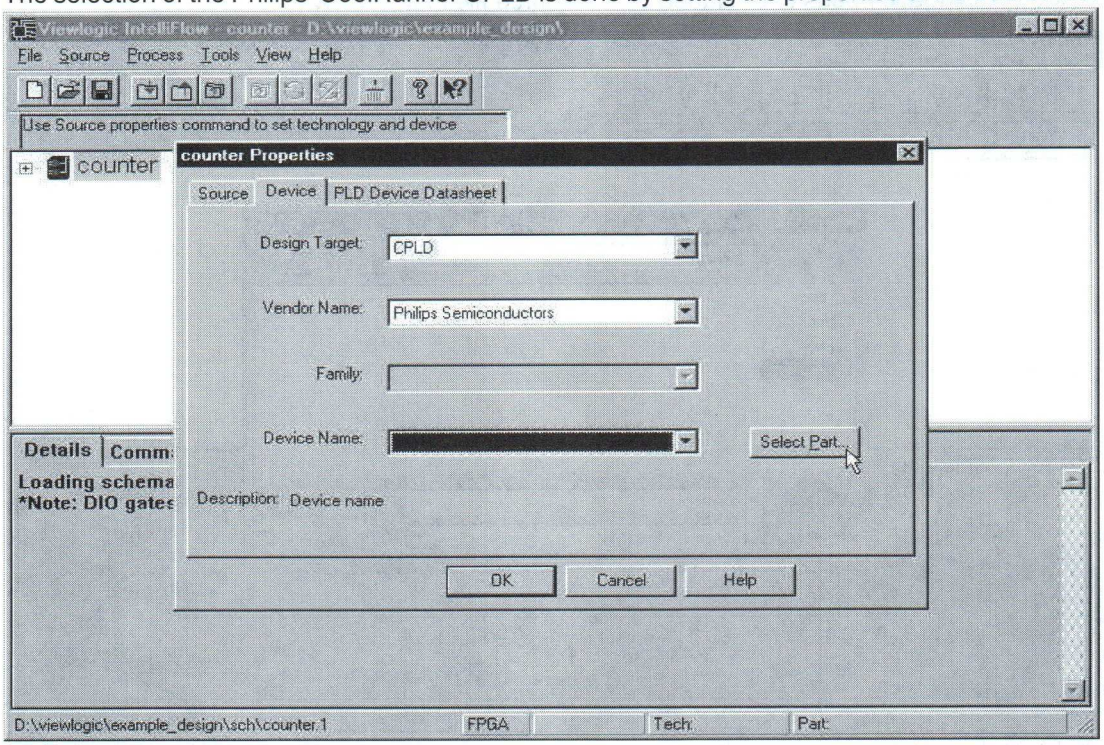


Figure 8:Setting properties for the source file

Right-click on **counter** and select **Properties...**.In the dialog box, select **CPLD** as the **Design Target** and **Philips Semiconductors** as the **Vendor Name**. Click **Select Part...** to select the specific CPLD desired.

This will open a device selection box.

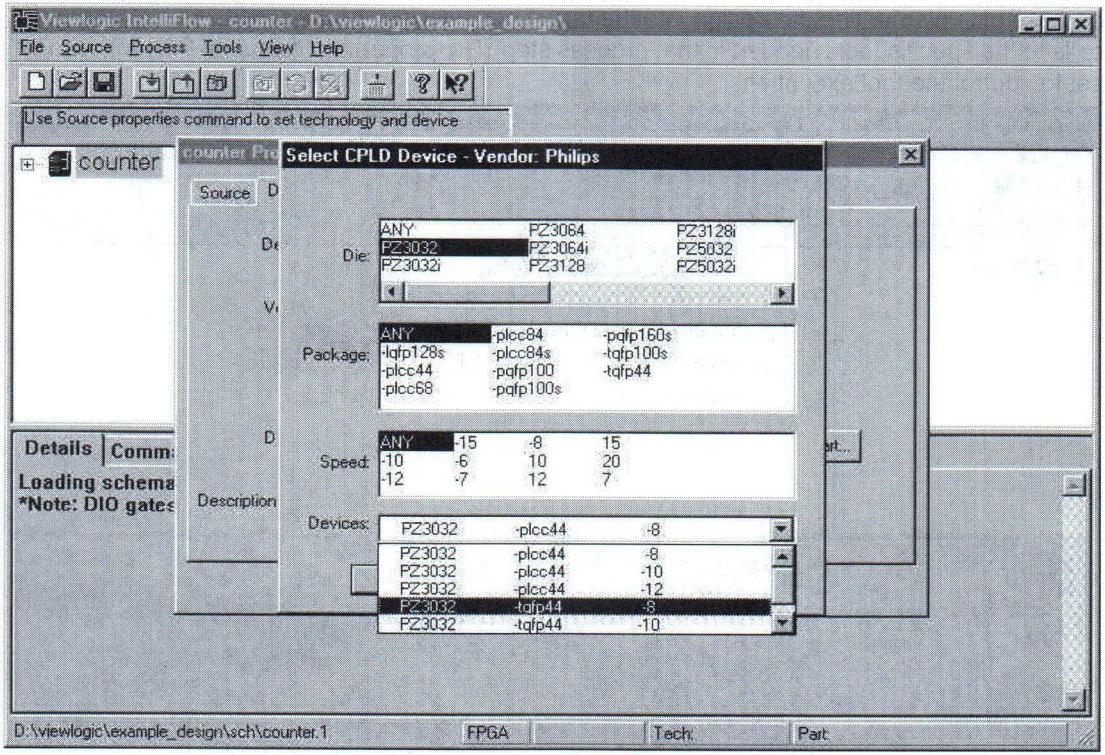


Figure 9:Choosing the Philips' CoolRunner CPLD

For **Die**, select the desired macrocell count. The part numbers containing "**i**" are industrial temperature parts. Select the desired package and speed grade. If **ANY** is chosen, the menu select list under **Devices:** shows the possible macrocell, package, and speed grade combinations. The target device can be chosen from this list.

Once the target device has been selected, Intelliflow updates the process window to show the process of implementing the design into the target device. Each of the process steps contains calls to the specific tools needed in that process step. The properties of each of the tools can be set to control the tool execution.

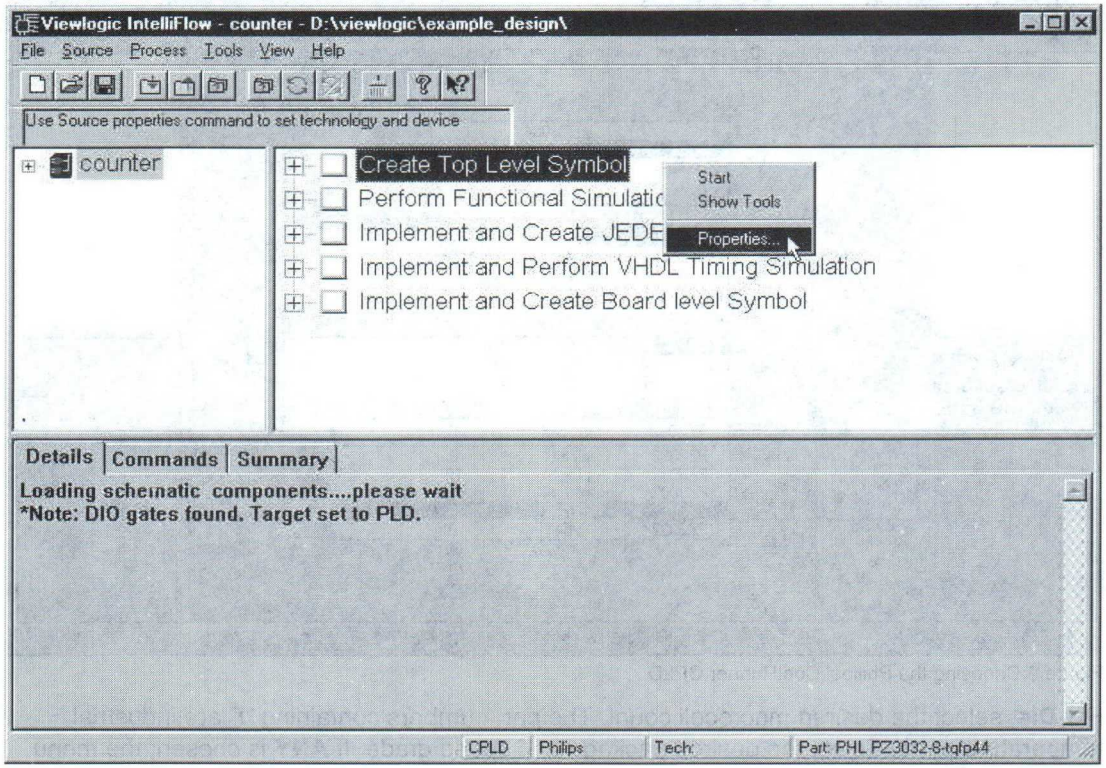


Figure 10:Process flow for Philips CPLDs

Right-click on **Create Top Level Symbol**. A small menu box appears with **Start**, **Show Tools**, and **Properties....** **Start** starts the process step, **Show Tools** shows all underlying tools making up this process step, and **Properties...** allows the designer to set parameters for the tools to control their execution.

Select **Properties...**

This will bring up a dialog box that allows you to select the spacing and length of the symbol pins.

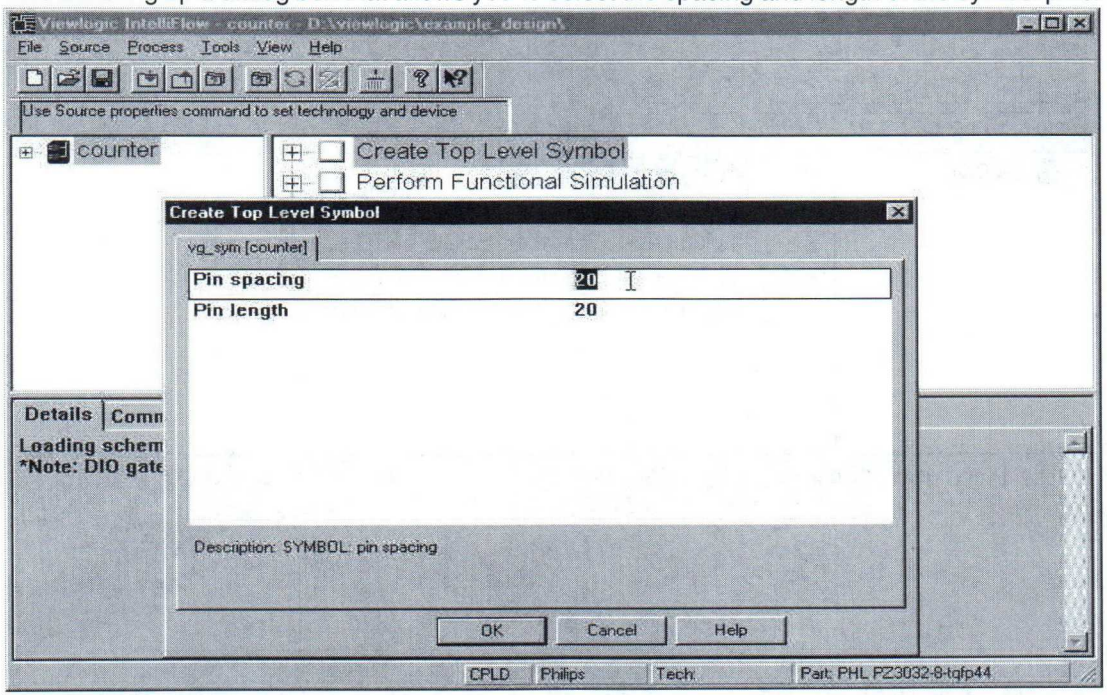


Figure 11:Setting properties for the Top Level Symbol

Change the **Pin spacing** to **20** and click **OK**.

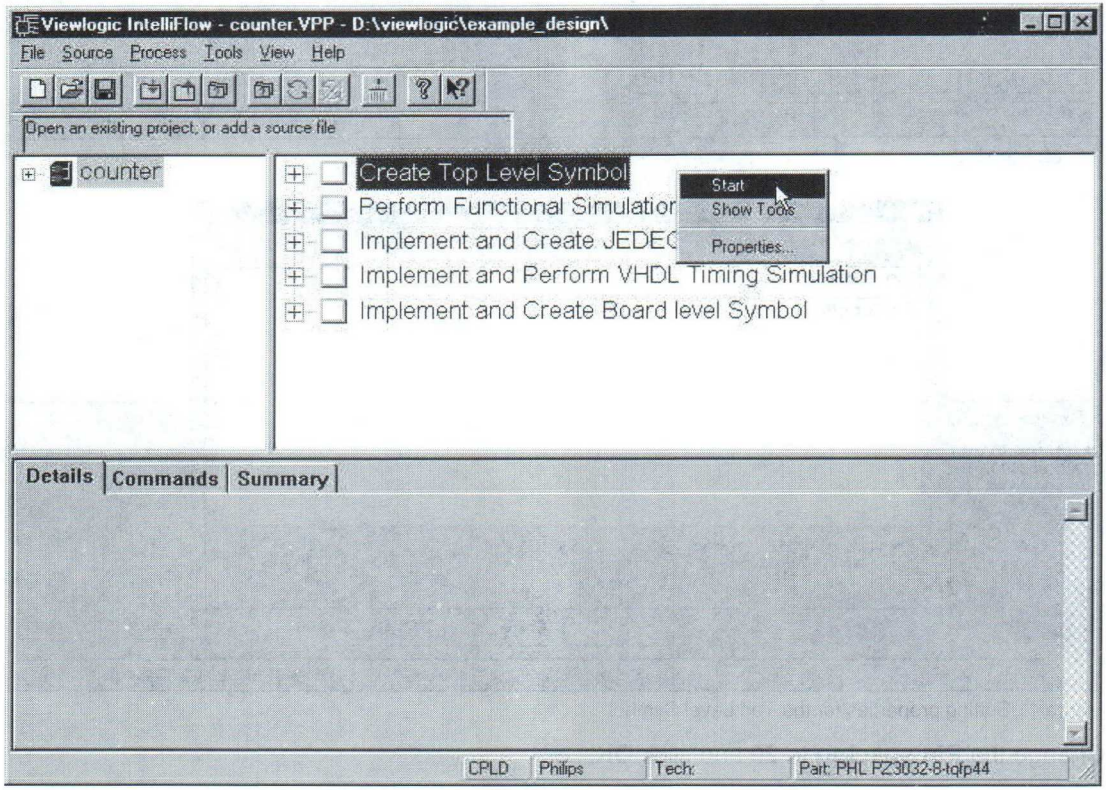


Figure 12:Starting the **Create Top Level Symbol** process

Right-click on **Create Top Level Symbol** again and select **Start**. This will start the process to create a top-level symbol of the counter schematic.

When the symbol has been successfully completed, the symbol is shown in Viewdraw.

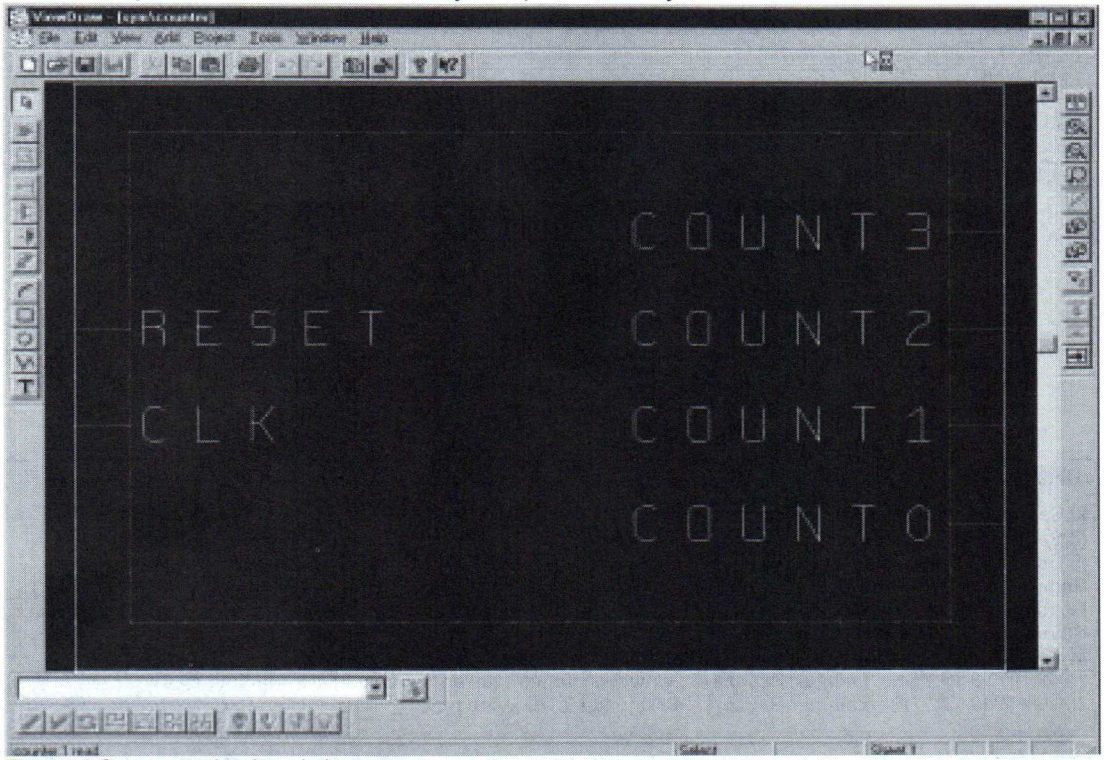


Figure 13:Counter top-level symbol

The next step in the process is to functionally simulate the design.

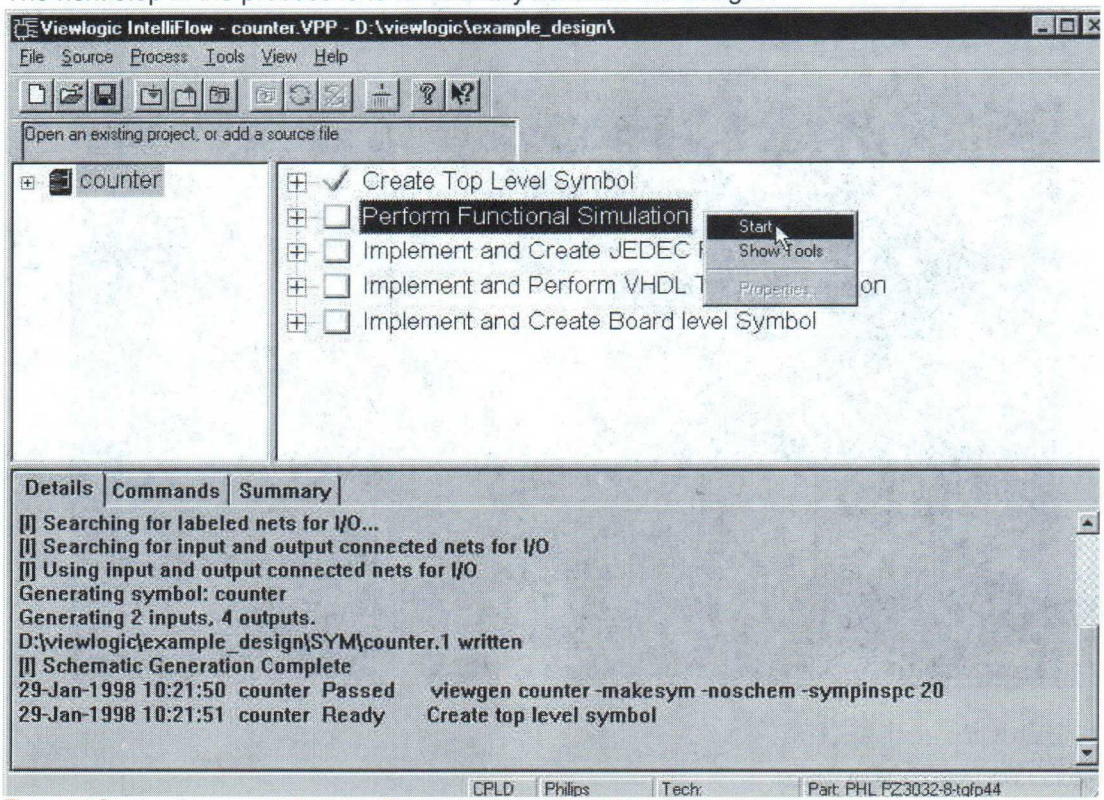


Figure 14:Starting the process, **Perform Functional Simulation**

Right-click on **Perform Functional Simulation** and select **Start**.

The process of creating a simulator netlist begins.

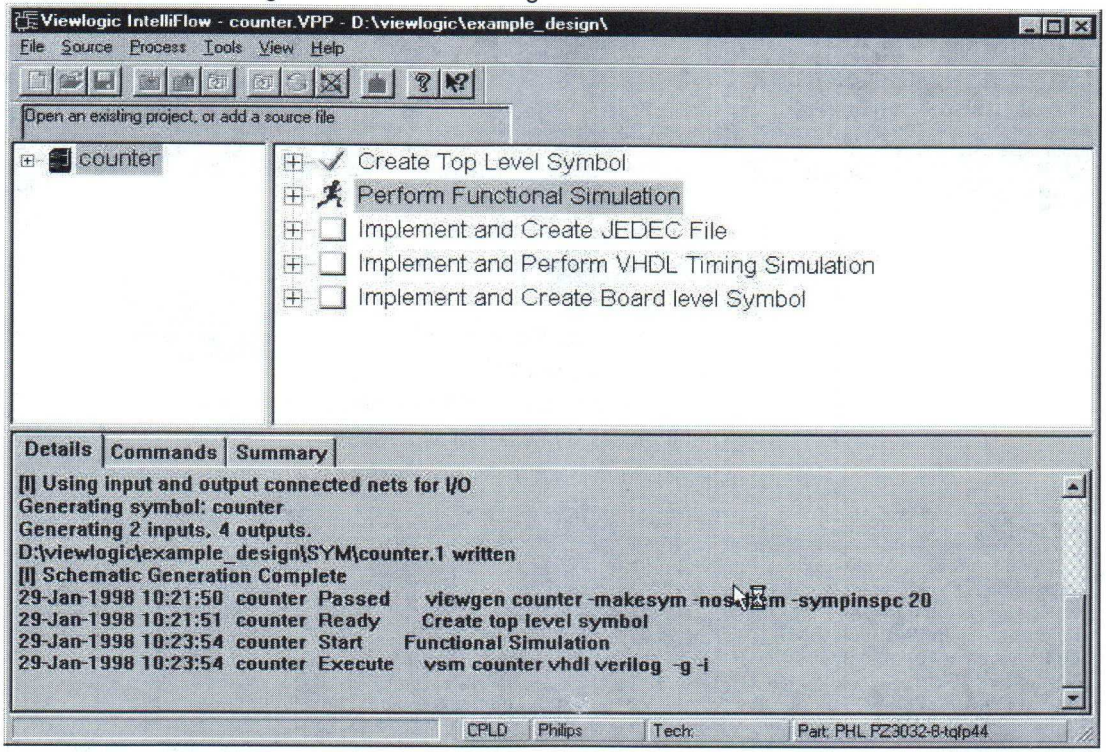


Figure 15:Creating simulation netlist

This application note will not detail the steps involved in functionally simulating the design. If the design does not function properly or as desired, edit the schematic. Save and check the corrected schematic and start the **Perform Functional Simulation** process again until the desired behavior is achieved. It is not necessary to repeat the **Create Top Level Symbol** step.

Now that the schematic represents the functionality desired, the design can be implemented in a Philips' CPLD.

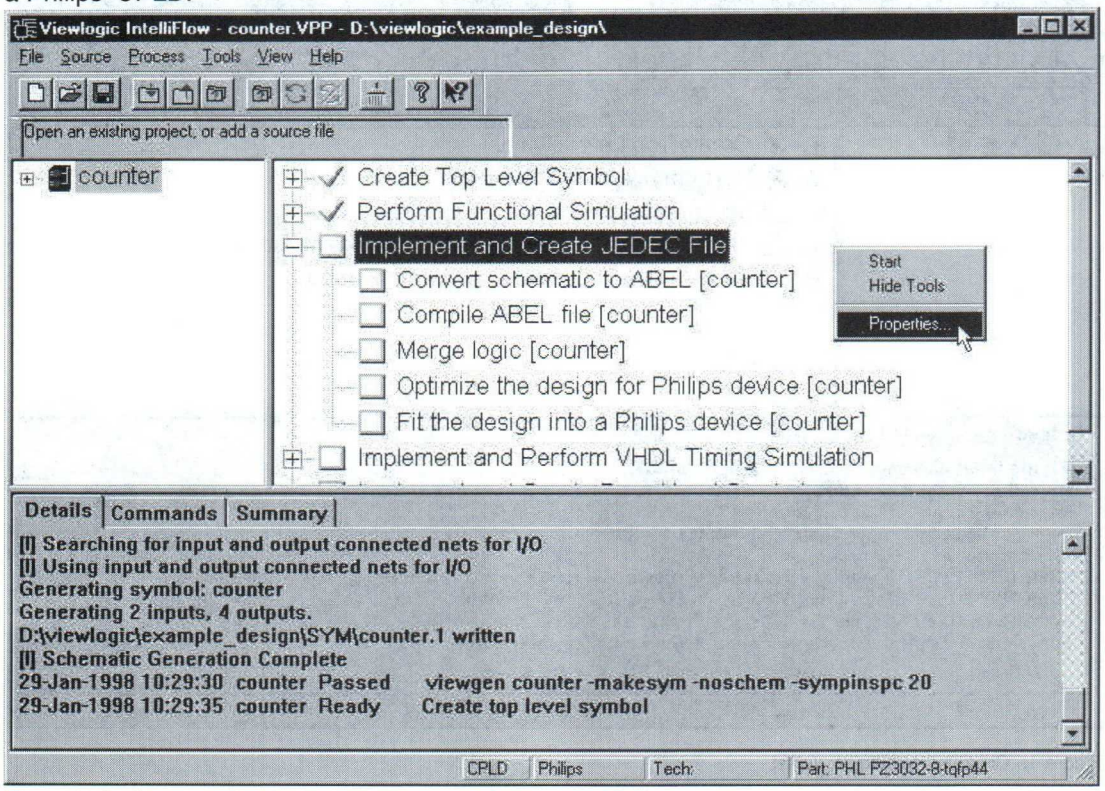


Figure 16:Setting properties for the **Implement and Create JEDEC File** process

By clicking on the "+" next to **Implement and Create JEDEC File**, the steps involved in this process are shown. The first step in this process is to create a ABEL file from the schematic. This ABEL file is then compiled and optimized. The optimized ABEL file is written as a TT2 file. This file is then input into the compiler and fitter for Philips' CoolRunner CPLDs.

Right-click on **Implement and Create JEDEC File** and select **Properties...**.

A window appears with tabs for each of the tools executed in this process step.

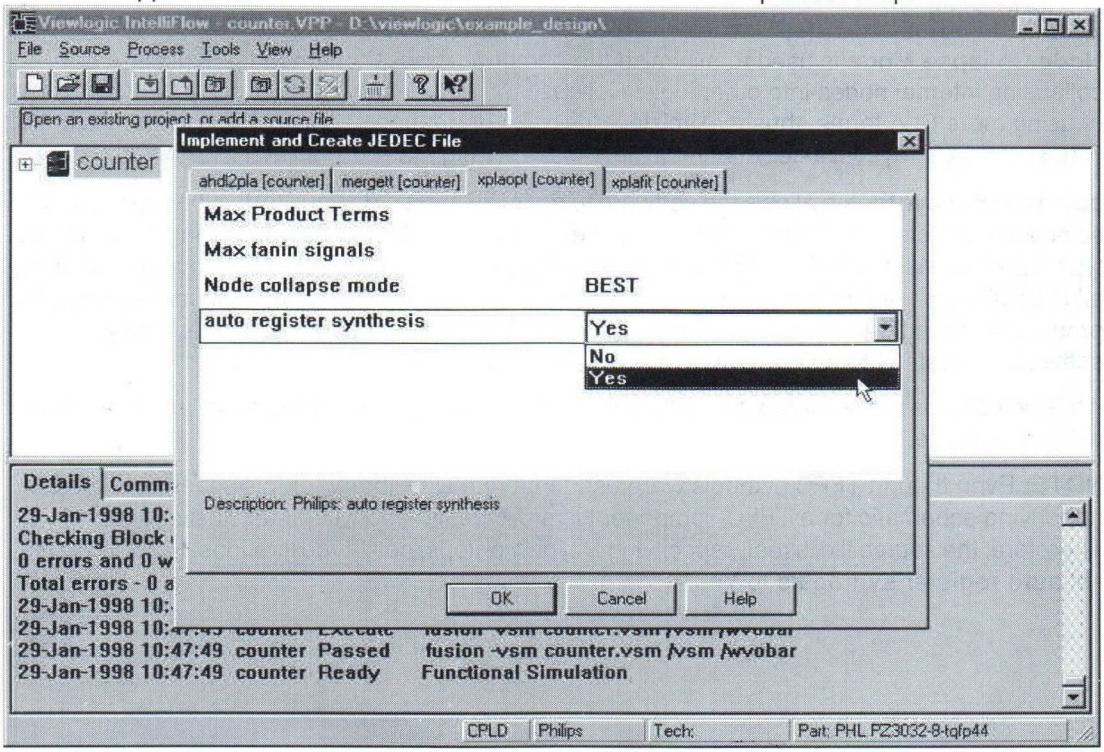


Figure 17:Setting parameters for XPLA Compiler

Select the **xplaopt [counter]** tab. This allows parameters to be set to control the compiler for Philips' CoolRunner CPLDs.

Max Product Terms: can be set to any number between **5** and **37**. This allows the designer to control how wide logic functions are implemented and trade speed for density. Wide logic functions can be implemented in two different manners: using more PLA product terms or using more than one pass through the logic array. Each macrocell in the XPLA architecture has 5 dedicated PAL product terms and has access to 32 PLA product terms. This parameter can determine if PLA terms are used to implement wide logic functions, or if only PAL product terms are used with more than one pass through the logic array.

Implementing wide logic functions using 1 or up to 32 of the PLA product terms will add an additional delay (see Philips CPLD Data Handbook for PLA delay time). Implementing wide logic functions using multiple passes through the logic array add a Tpd delay for every additional logic array pass. If speed is the most important parameter, the design should specify a larger value for this field. If fitting the design is perceived to be an issue (density problem) and speed is not as important, then the user should specify a smaller value for this field.

Max fanin signals controls the number of signals allowed as inputs to the logic block. This should be set to 36.

Node Collapse Mode is used to minimize the number of passes through the logic array by collapsing internal nodes into other logic. Collapsing the node frees a macrocell at the expense of using more PLA terms, thus maximizing design performance because one pass through the PAL and PLA arrays is faster than multiple passes through the PAL array.

auto register synthesis allows the software to optimize certain designs to T -type flip-flops. This selection instructs XPLA Designer to minimize the number of product terms using either D or T type flip-flops. When the selection is activated, the software will implement each equation using all D or all T type flip-flops and it will pick the flip-flop type that requires the minimum number of product terms. If the number of product terms are the same with either implementation, the software will default to a D type flip-flop.

If this selection is not activated, the XPLA Designer software will use the type of flip-flop specified in the source file.

NOTE: Even though TFFC symbols were used from the DIO library for this schematic, the underlying schematic for a TFFC component is a D type flip-flop with an exclusive-or gate. Therefore, the source file is really specifying D type flip-flops, not T type flip-flops. For this design, set **auto register synthesis** to Yes.

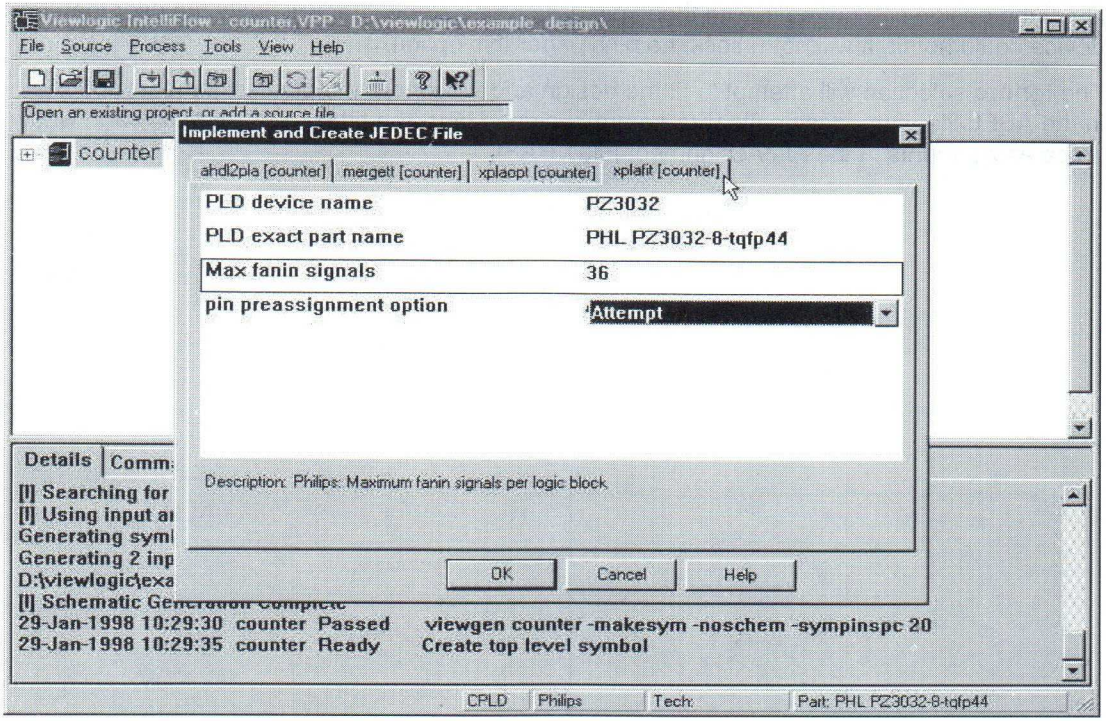


Figure 18:Setting parameters for XPLA Fitter

Click on the **xplafit [counter]** tab. This allows parameters to be set to control the fitter for Philips' CoolRunner CPLDs.

The device selected earlier should appear as **PLD device name**. The **PLD exact part name** should show the Philips' CoolRunner CPLD part number. **Max fanin signals** should again be set to **36**.

The choices for **pin preassignment option** are **Attempt**, **Keep**, and **Ignore**. Note that at this time, the only way to pre-assign pins to the design is to edit the ABEL source file generated from the schematic and assign the desired pin numbers to the I/O signals. Note that if the schematic changes, the ABEL file will be re-generated and the designer will have to re-enter this information into this file.

The **Attempt** selection will attempt to fit the design with the pin assignments specified in the ABEL file. If the design can not be fit with these pin assignments, the fitter will remove the pre-assigned pins and attempt to fit the design with no pre-assigned pins. A warning message will tell the user if the pre-assigned pins have been removed. If the device then fits, the fitter assigns pins and writes these assignments in the Pin Assignment File (PAF).

The **Keep** selection will attempt to fit the design with the pin assignments specified in the ABEL file. If the design can not be fit with these pin assignments, the fitter will notify the user that the device could not fit. It will not unlock the pins under this option.

The **Ignore** selection will attempt to fit the design and will ignore the pin assignments specified in the ABEL file. If the design fits with no pre-assigned pins, the fitter will assign pins and write these assignments in the Pin Assignment File. (PAF).

Once the parameters for the compiler and fitter have been specified, the process can be started.

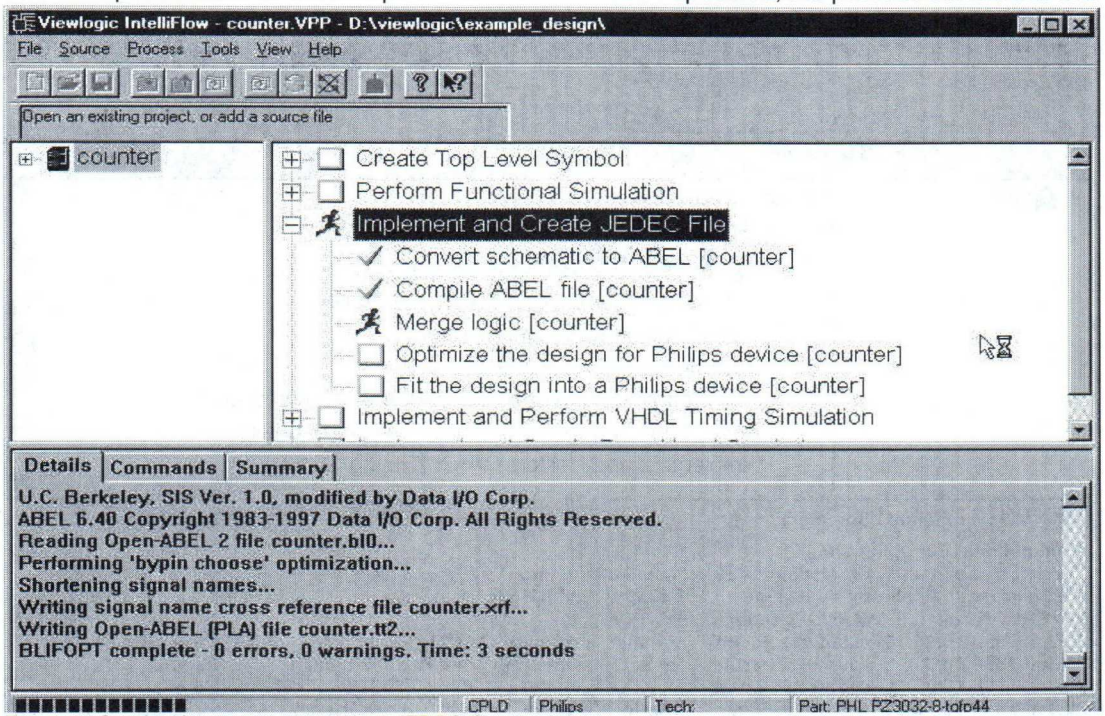


Figure 19:Starting the process to create a JEDEC file

Right-click on **Implement and Create a JEDEC File** and select **Start**. The window shows the status of the process. Clicking on the **Details** tab in the lower window shows the output messages of the individual tools as they run. Clicking on the **Commands** tab in the lower window shows the command line sent to each of the tools. Clicking on the **Summary** tab in the lower window shows when processes have started and stopped.

When that process has successfully completed, all tools are checked. The JEDEC file is ready to be programmed into the device.

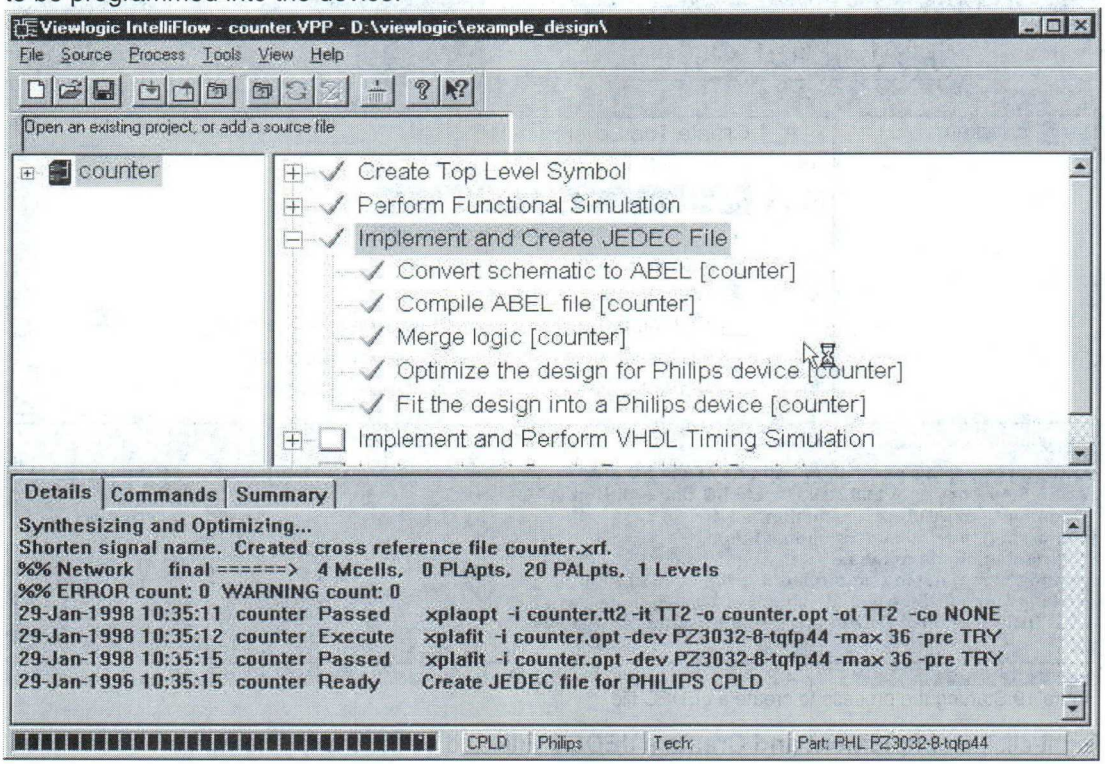


Figure 20:Successful completion of the creation of a Philips' CPLD JEDEC file

To verify the timing of the device, the designer can run **Implement and Perform VHDL Timing Simulation** to simulate a timing model of the device.

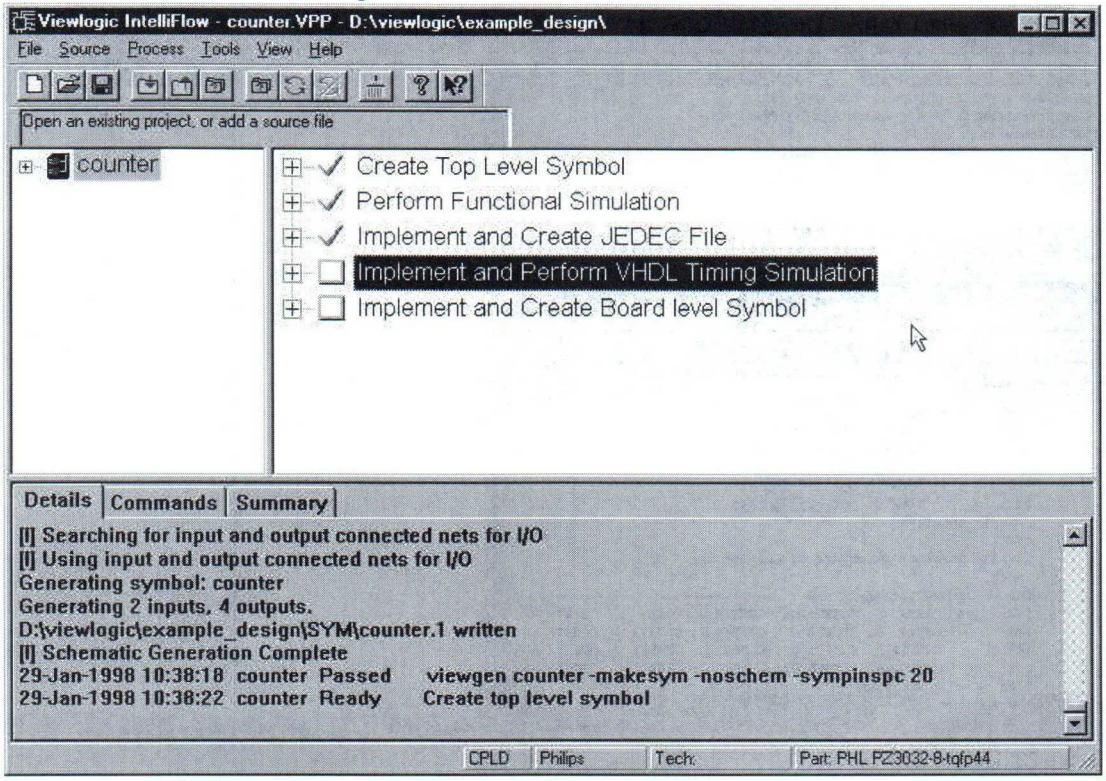


Figure 21:Starting the **Implement and Perform VHDL Timing Simulation Process**

Right-click on **Implement and Perform VHDL Timing Simulation** and select **Start**.

This starts Speedwave. Refer to Viewlogic documentation for detailed instructions on running Speedwave.

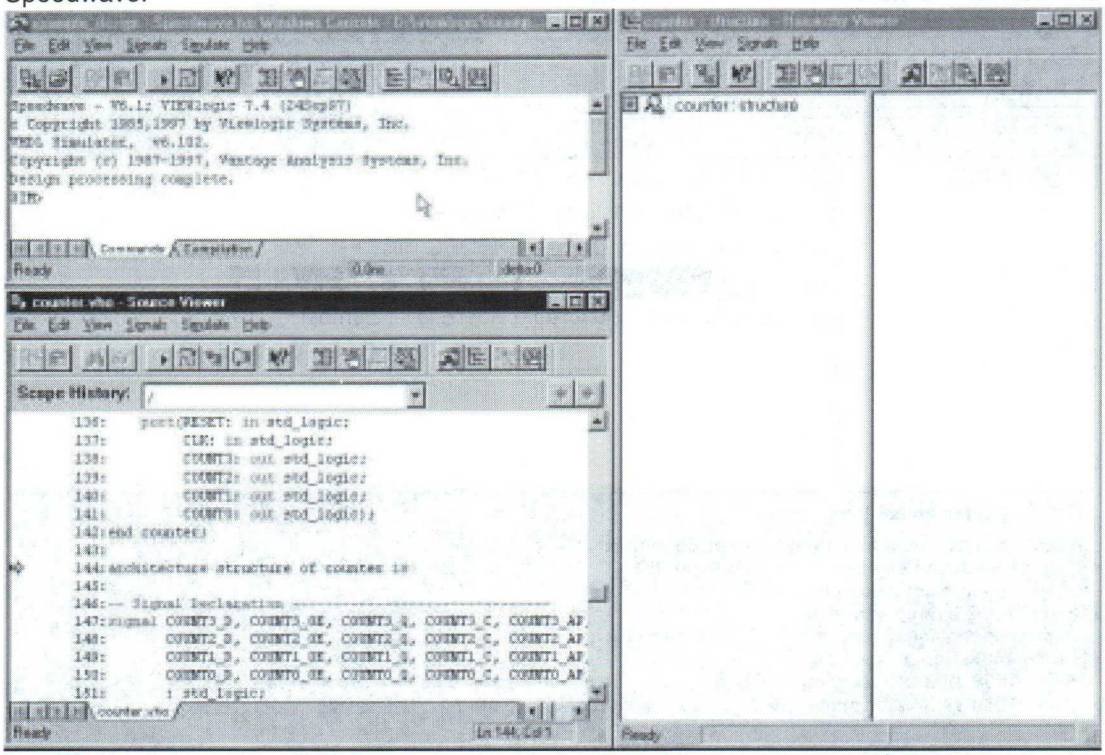


Figure 22:Speedwave

If a command file has been written to describe the stimulus, enter the name of the command file at the **SIM>** prompt.

A sample command file is shown below:

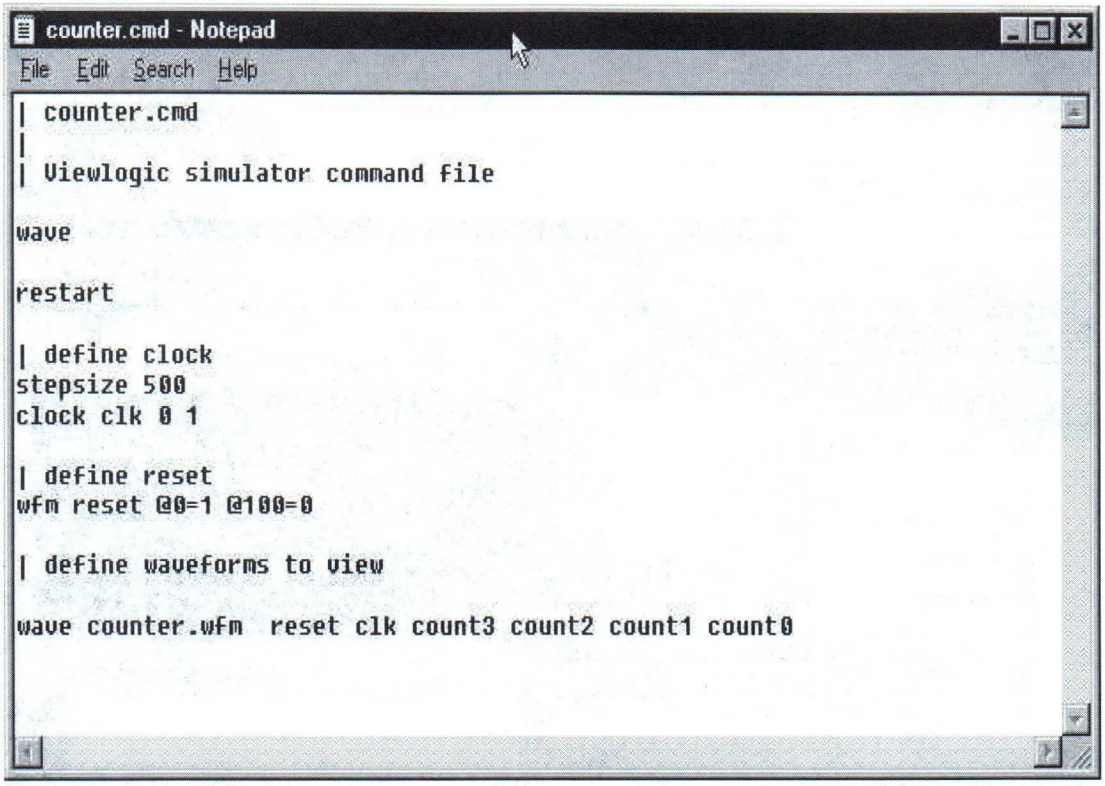


Figure 23:Command file for the counter design

This file defines the clock as 50ns high and 50ns low. The reset signal is active-high. It will start the simulation high and then go low after 10ns. The wave command will open a waveform window to allow review of the simulation results.

Start the simulation. The Vwaves window will display the resulting waveforms.

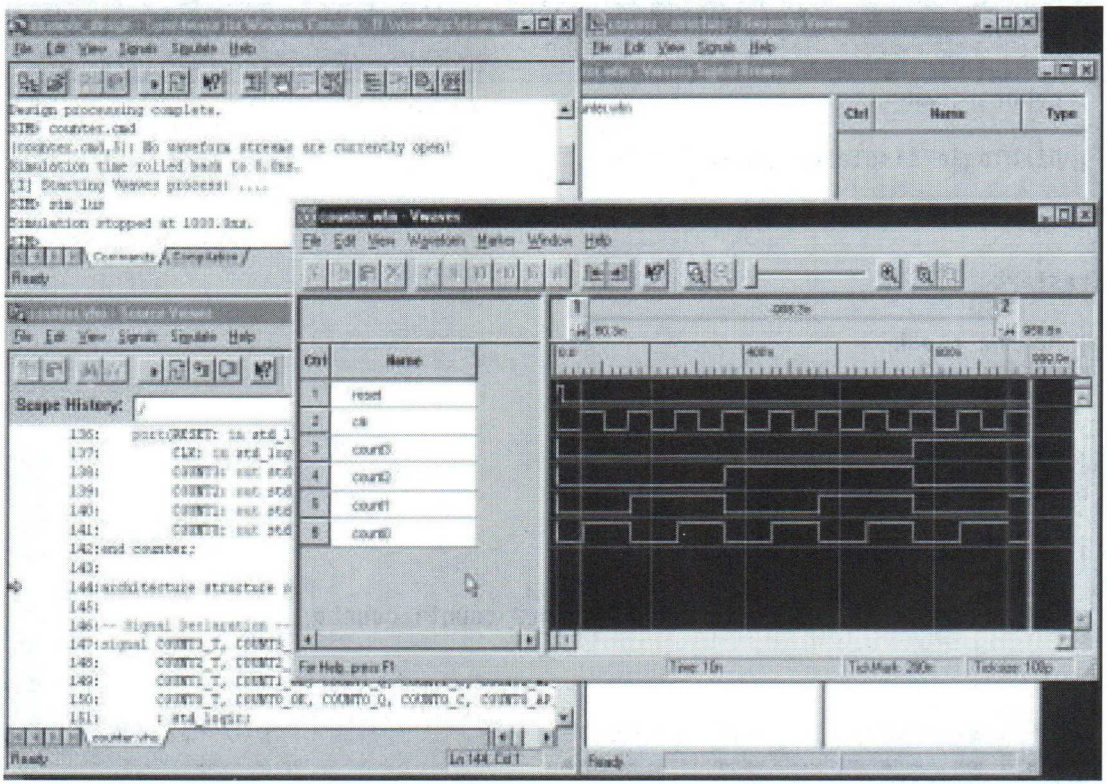


Figure 24:VHDL Timing Model Simulation Results

Congratulations! The process of targeting a design to Philips' CoolRunner CPLDs and verifying the design are complete.

CONCLUSION

This application notes provides the basic steps in getting started using Viewlogic Intelliflow to target Philips CPLDs. To become more proficient, please see the documentation listed on under REFERENCES at the beginning of this note.

APPENDIX A - PHILIPS CPLD PART NUMBERING SYSTEM

PZXYYYK(S)ZZYYY					
X = SUPPLY VOLTAGE	**YYY = MACROCELL COUNT**	**K = OPERATING TEMPERATURE**	**S = ISP**	**ZZ = SPEED GRADE (TPD)**	**YYY= PACKAGE DESIGNATOR**
3 = 3.3V 5 = 5V	32 64 128 320 960	- = Commercial I = Industrial C = Commercial, enhanced clocking N = Industrial, enhanced clocking A = Commercial, enhanced clocking, 0.35 micron process with 5 V tolerant I/Os D = Industrial, enhanced clocking, 0.35 micron process with 5 V tolerant I/Os		6 = 6ns 7=7.5ns 8=8ns 10=10ns 12=12ns 15=15ns	A44 = 44 pin PLCC A68 = 68 pin PLCC A84 = 84 pin PLCC BB1 = 100 pin PQFP BB2 = 160 pin PQFP BC = 44 pin TQFP BE = 128 pin LQFP BP = 100 pin TQFP EB = 492 pin PBGA

Section 8
Package information

CONTENTS

Soldering 628

SO24: plastic small outline package; 24 leads; body width 7.5 mm ..... SOT137-1 .. 630

TSSOP24: plastic thin shrink small outline package; 24 leads; body width 4.4 mm ..... SOT355-1 .. 631

PLCC28: plastic leaded chip carrer; 28 leads; pedestal ..... SOT261-3 .. 632

PLCC44: plastic leaded chip carrier; 44 leads ..... SOT187-2 .. 633

TQFP44: plastic thin quad flat package; 44 leads; body 10 x 10 x 1.0 mm ..... SOT376-1 .. 634

PLCC68: plastic leaded chip carrier; 68 leads; pedestal ..... SOT188-3 .. 635

PLCC84: plastic leaded chip carrier; 84 leads; pedestal ..... SOT189-3 .. 636

QFP100: plastic quad flat package; 100 leads (lead length 1.6 mm); body 14 x 20 x 2.8 mm ..... SOT382-1 .. 637

QFP160: plastic quad flat package; 160 leads (lead length 1.6 mm); body 28 x 28 x 3.4 mm; high stand-off height ..... SOT322-2 .. 638

TQFP100: plastic thin quad flat package; 100 leads; body 14 x 14 x 1.0 mm ..... SOT386-1 .. 639

LQFP128: plastic low profile quad flat package; 128 leads; body 14 x 20 x 1.4 mm ..... SOT425-1 .. 640

BGA492: plastic ball grid array package; 492 balls; body 35 x 35 x 1.75 mm ..... SOT514-1 .. 641

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.

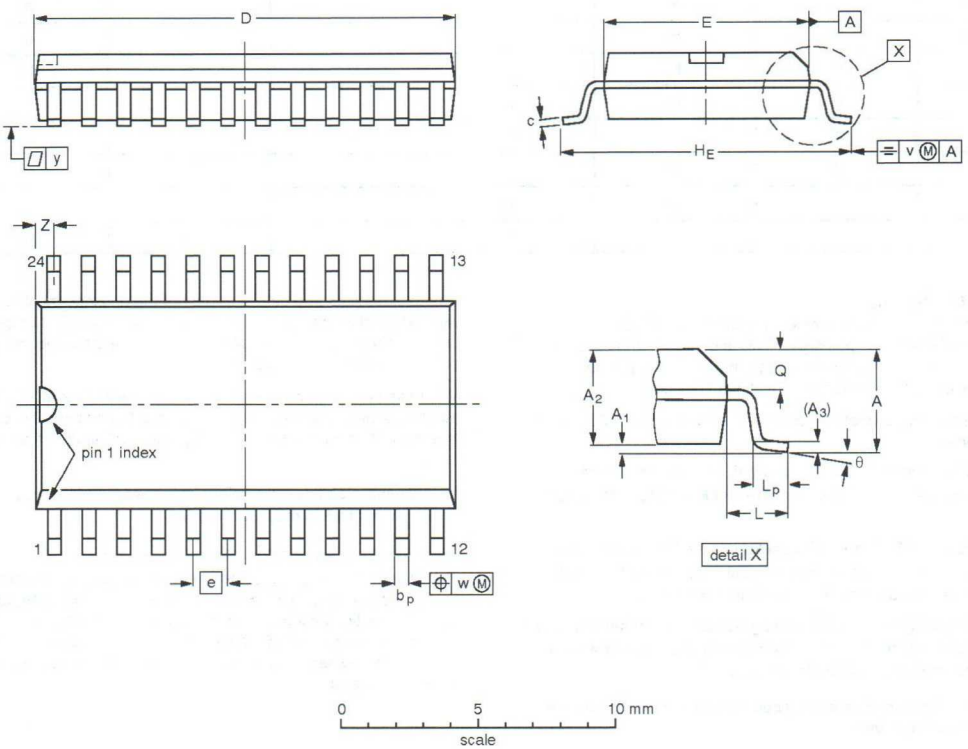
A mildly-activated flux will eliminate the need for removal of corrosive residues in most applications.

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.

Package outlines

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	0.419 0.394	0.055	0.043 0.016	0.043 0.039	0.01	0.01	0.004	0.035 0.016	

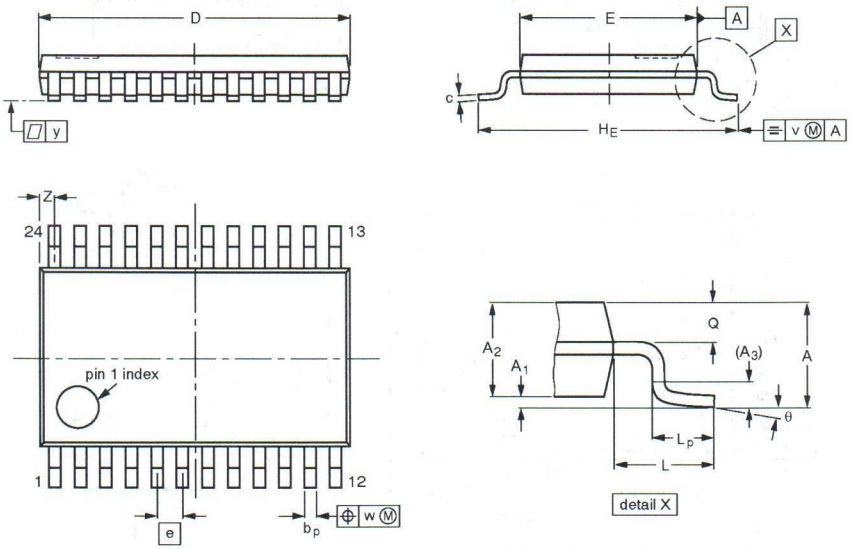
Note

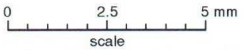
1. Plastic or metal protrusions of 0.15 mm maximum per side are not included.

OUTLINE VERSION	REFERENCES				EUROPEAN PROJECTION	ISSUE DATE
	IEC	JEDEC	EIAJ			
SOT137-1	075E05	MS-013AD				~~95-01-24~~ 97-05-22

Package outlines

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°

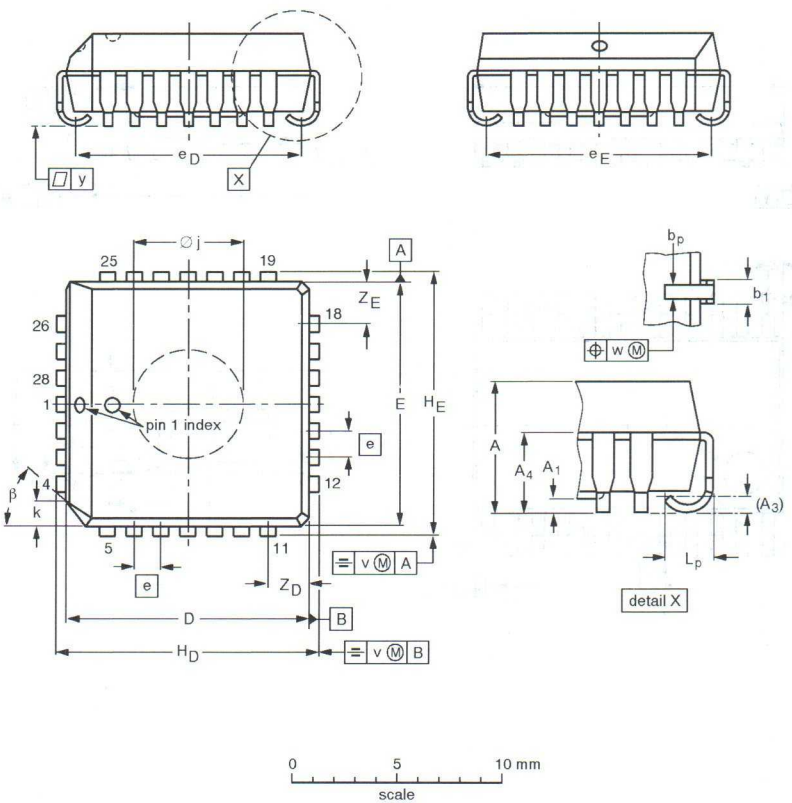
Notes

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	$\varnothing_j$	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	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	

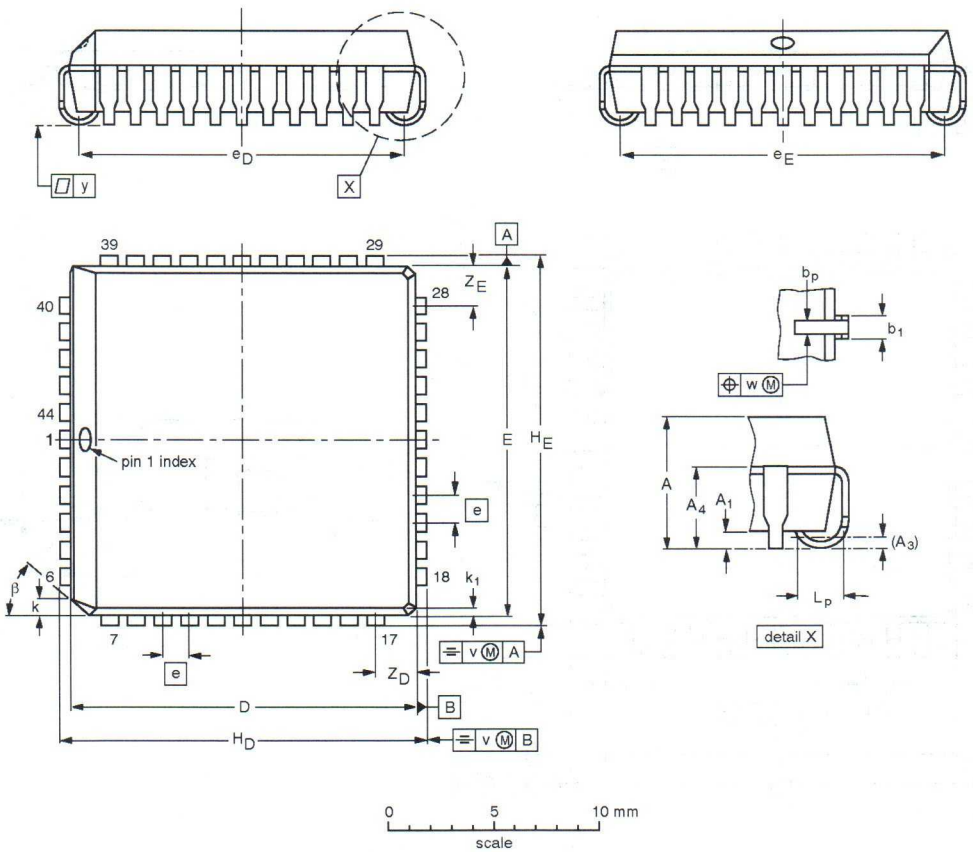
Note

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				~~95-02-25~~ 97-12-16

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°
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	

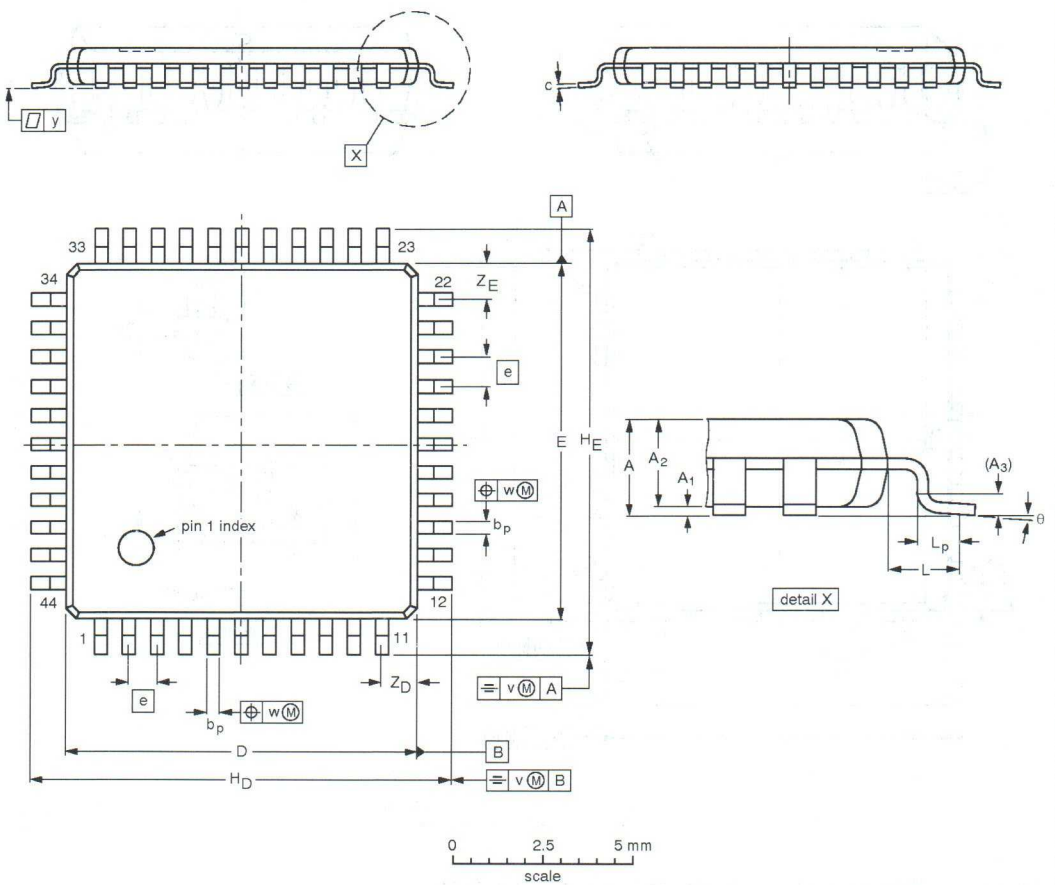
Note

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				~~95-02-25~~ 97-12-16

Package outlines

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	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	0.2	0.2	0.1	1.2 0.8	1.2 0.8	7° 0°

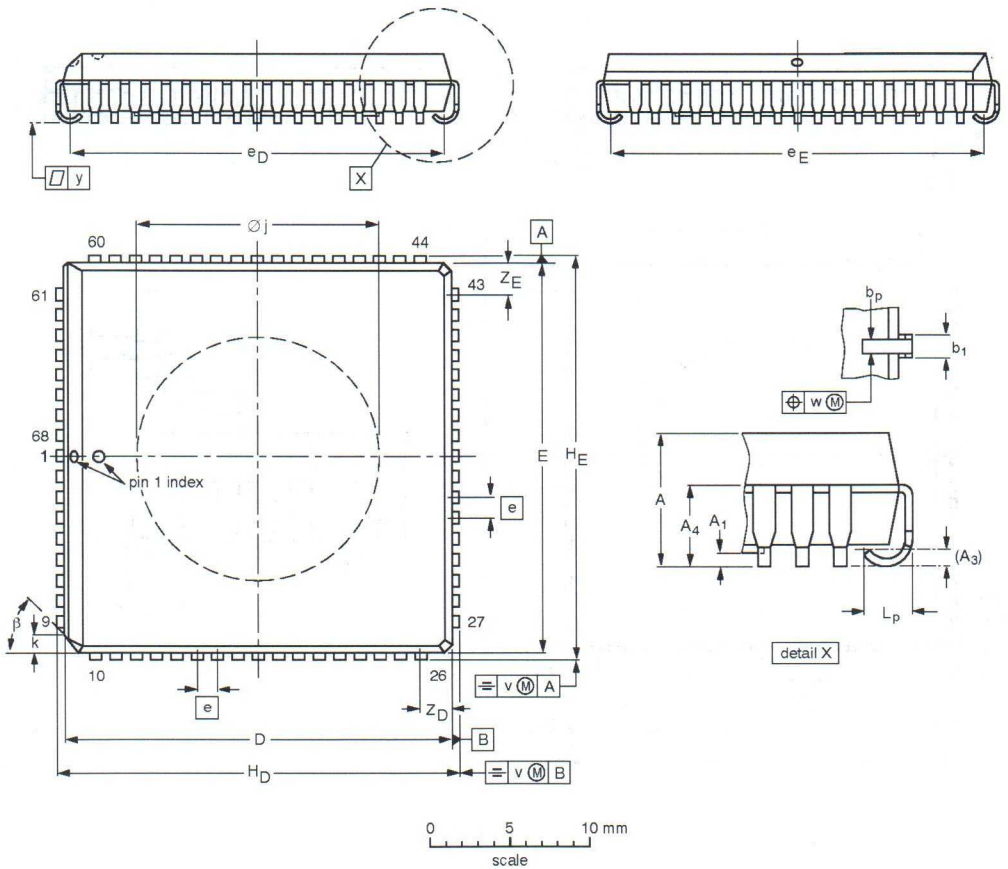
Note

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						~~96-04-02~~ 97-08-04

Package outlines

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	

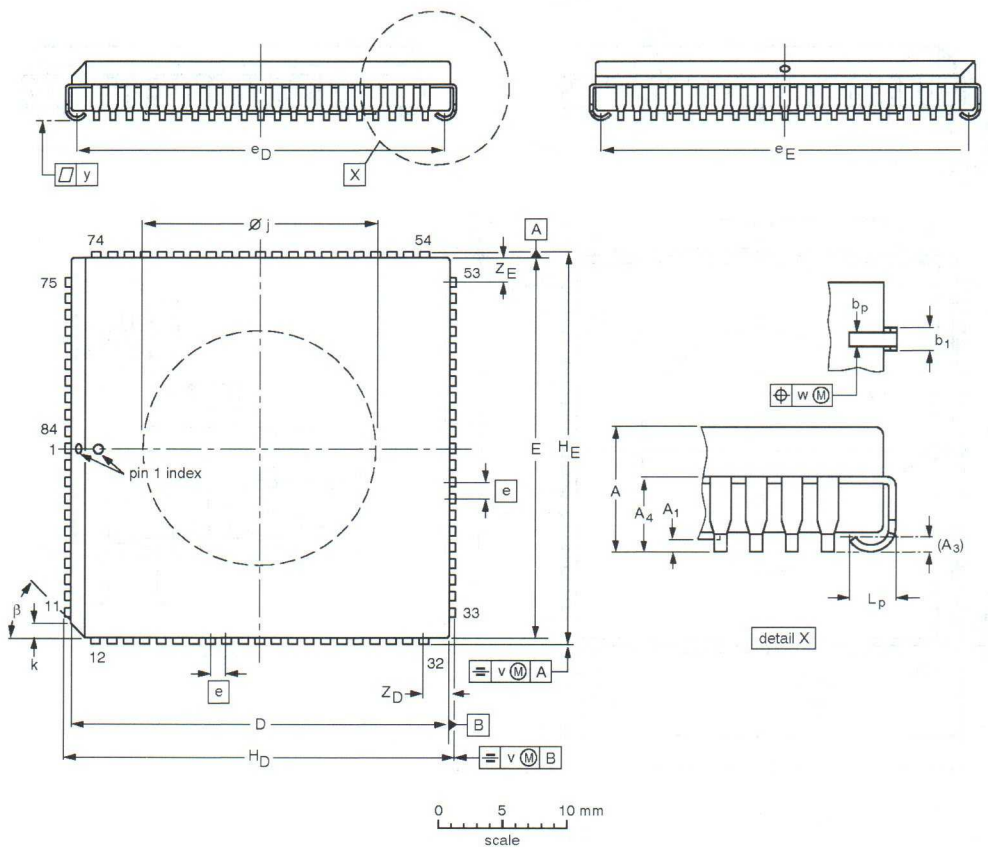
Note

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				~~95-02-25~~ 97-12-16

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	

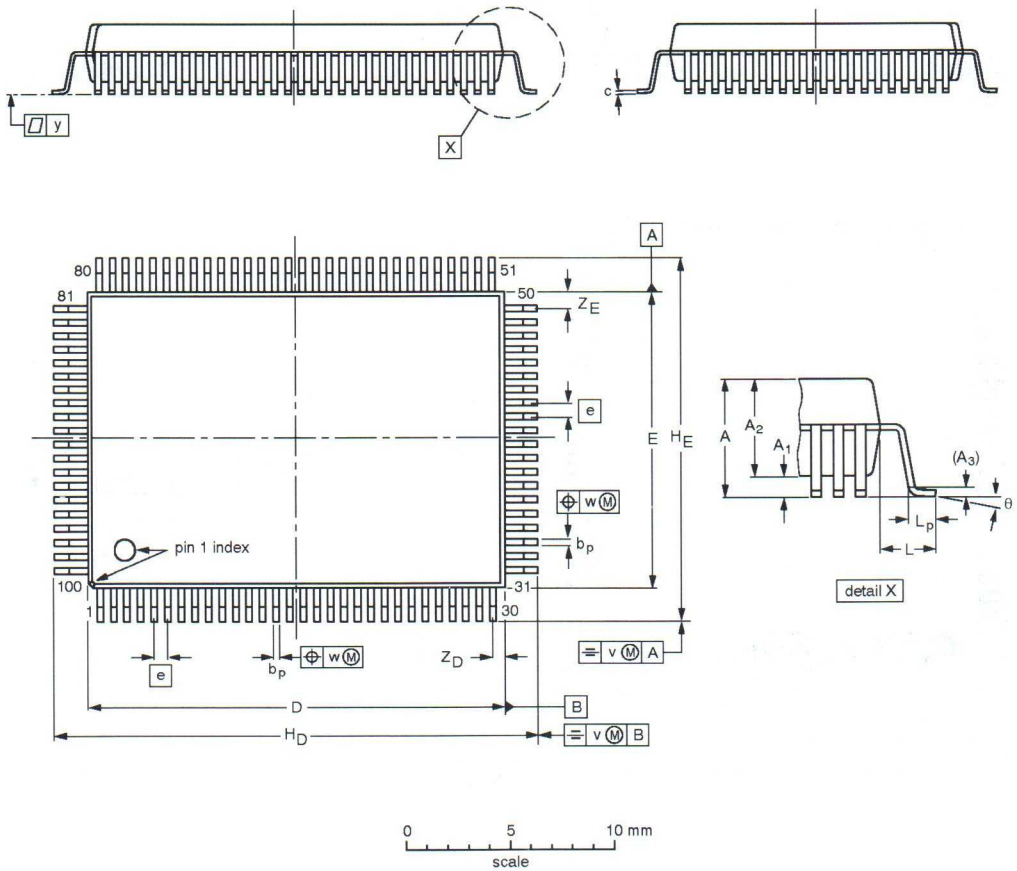
Note

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

Package outlines

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	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	0.20	0.12	0.10	0.68 0.45	0.68 0.45	7° 0°

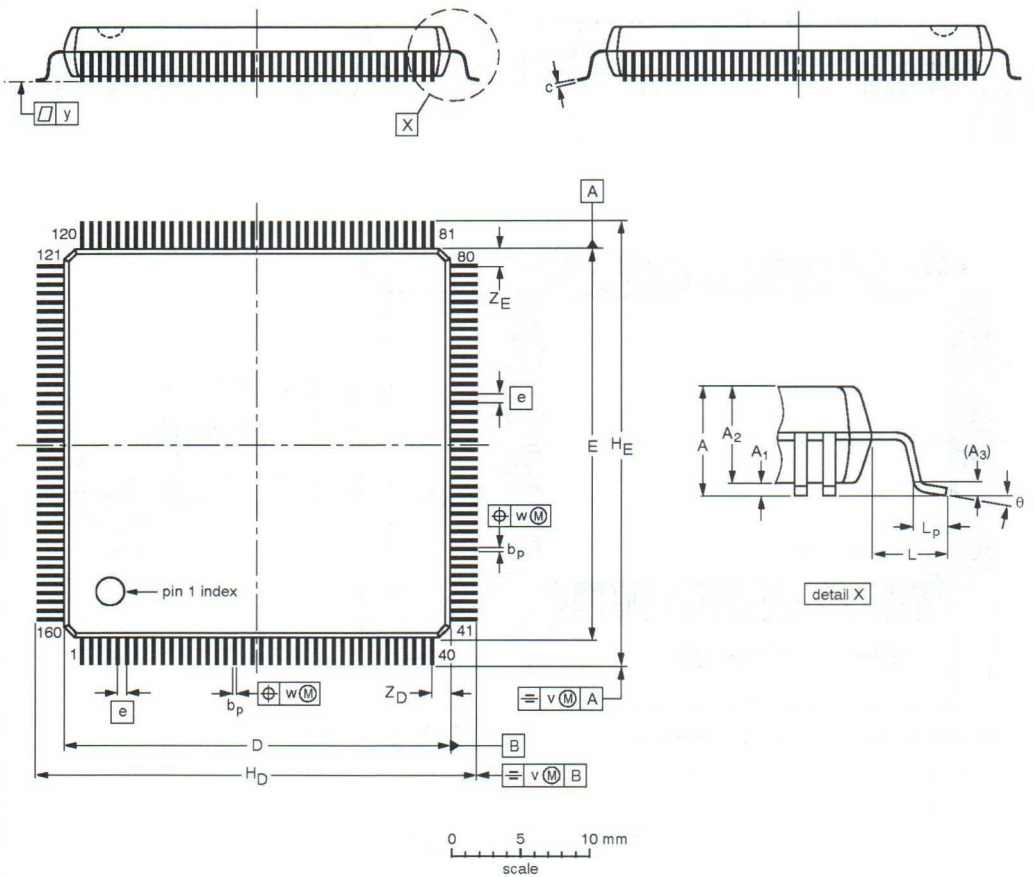
Note

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				~~95-02-04~~ 97-08-04

Package outlines

QFP160: plastic quad flat package;
160 leads (lead length 1.6 mm); body 28 x 28 x 3.4 mm; high stand-off height **SOT322-2**



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	v	w	y	$Z_D^{(1)}$	$Z_E^{(1)}$	θ
mm	4.07	0.50 0.25	3.70 3.15	0.25	0.38 0.22	0.23 0.13	28.1 27.9	28.1 27.9	0.65	31.45 30.95	31.45 30.95	1.6	1.03 0.73	0.3	0.15	0.1	1.5 1.1	1.5 1.1	7^o 0^o

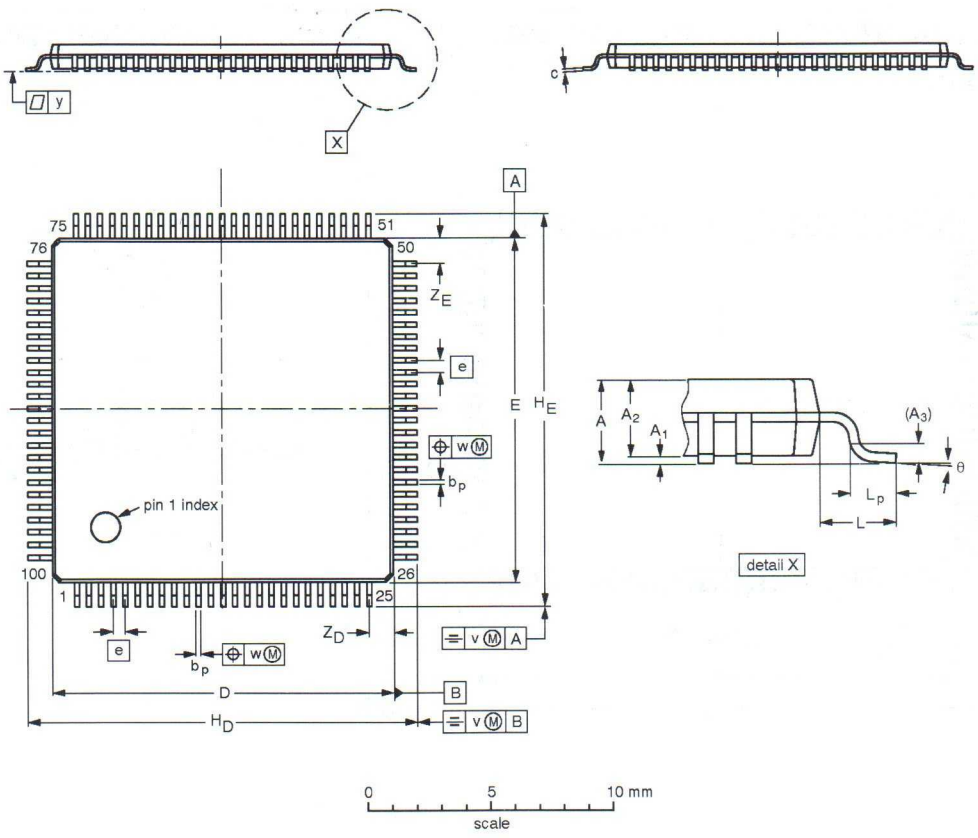
Note

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			
SOT322-2		MO112DD1				~~96-03-14~~ 97-08-04

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	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	0.2	0.08	0.1	1.15 0.85	1.15 0.85	7^o 0^o

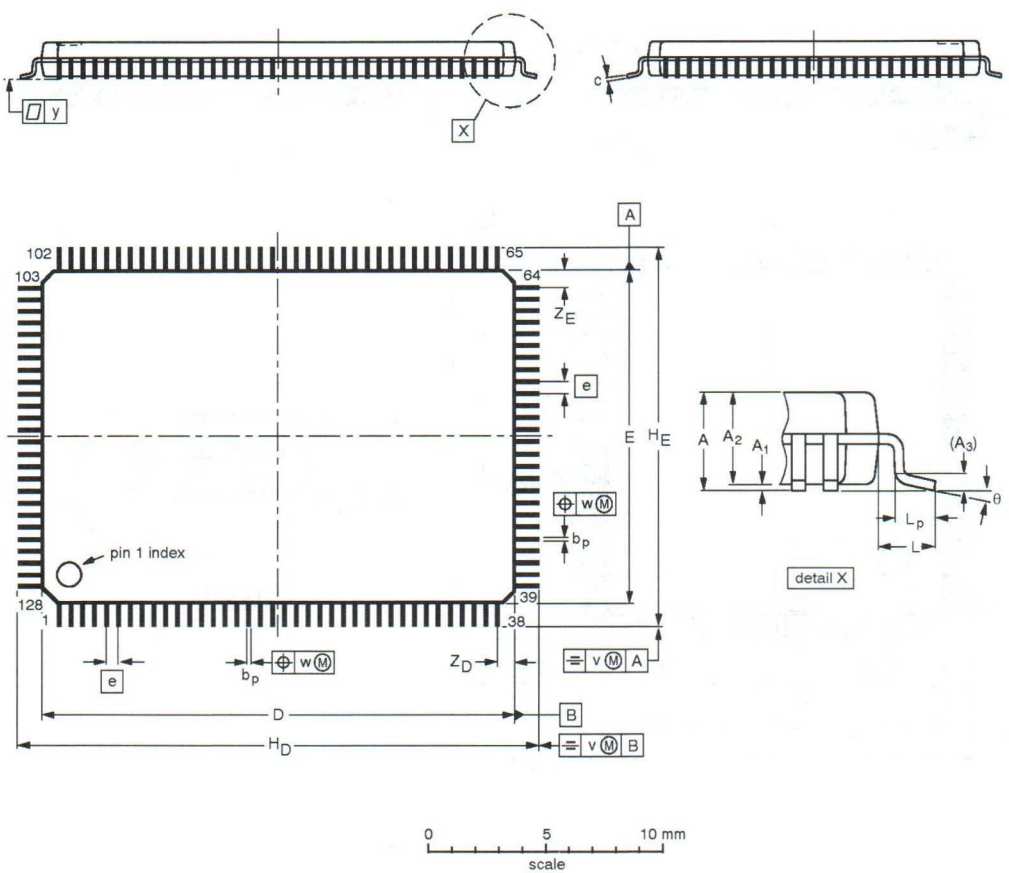
Note

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						~~96-04-02~~ 97-08-04

Package outlines

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	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	0.2	0.12	0.1	0.81 0.59	0.81 0.59	7° 0°

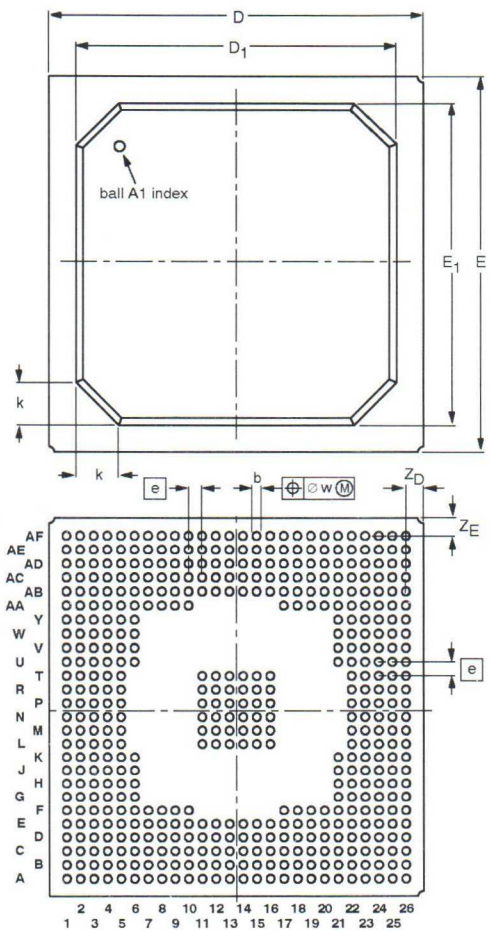
Note

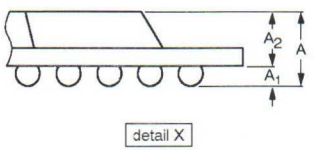
1. Plastic or metal protrusions of 0.25 mm maximum per side are not included.

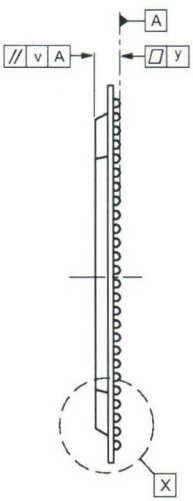
OUTLINE VERSION	REFERENCES				EUROPEAN PROJECTION	ISSUE DATE
	IEC	JEDEC	EIAJ			
SOT425-1						~~96-04-02~~ 97-08-04

Package outlines

BGA492: plastic ball grid array package; 492 balls; body 35 x 35 x 1.75 mm — SOT514-1







0 10 20 mm

scale

DIMENSIONS (mm are the original dimensions)

UNIT	A max.	A_1	A_2	b	D	D_1	E	E_1	e	k	v	w	y	Z_D	Z_E
mm	2.54	0.7 0.5	1.84 1.62	0.9 0.6	35.1 34.9	30.7 30.0	35.1 34.9	30.7 30.0	1.27	4.1 3.9	0.35	0.3	0.15	1.93 1.28	1.93 1.28

OUTLINE VERSION	REFERENCES				EUROPEAN PROJECTION	ISSUE DATE
	IEC	JEDEC	EIAJ			
SOT514-1						98-04-08

Appendix A
Data handbook system

Data handbook system 644

DATA HANDBOOK SYSTEM

Philips Semiconductors data handbooks contain ali 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, Audio and CD/DVD 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^2C Peripherals
IC13	Programmable Logic Devices (PLD)
IC14	8048-based 8-bit Microcontrollers
IC15	FAST TTL Logic Series
IC16	CMOS ICs for Clocks, Watches and Real Time Clocks
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	Integrated Circuit Packages
IC27	Complex Programmable Logic Devices

Discrete Semiconductors

Book	*Title*
SC01	Small-signal and Medium-power Diodes
SC02	Power Diodes
SC03	Power 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
SC13	Power MOS Transistors
SC14	RF Wideband Transistors
SC16	Wideband Hybrid Amplifier Modules for CATV
SC17	Semiconductor Sensors
SC18	Discrete Semiconductor Packages
SC19	RF & Microwave Power Transistors, RF Power Modules and Circulators/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 and Multimedia Tubes
DC05	Wire Wound Components

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
PA06a	Surface Mounted Ceramic Multilayer Capacitors
PA06b	Leaded 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.

Australia: North Ryde, Tel. +61 2 9805 4455, Fax. +61 2 9805 4466
Austria: Wien, Tel. +43 1 60 101 12 41, Fax. +43 1 60 101 12 11
Belarus: Minsk, Tel. +375 172 200 924/733, Fax. +375 172 200 773
Benelux: Eindhoven, Tel. +31 40 2783 749, Fax. +31 40 2788 399
Brazil: São Paulo, Tel. +55 11 821 2333, Fax. +55 11 829 1849
Canada: Scarborough, Tel. 1 416 292 5161, Fax. 1 416 754 6248
China: Shanghai, Tel. +86 21 6354 1088, Fax. +86 21 6354 1060
Denmark: Copenhagen, Tel. +45 32 883 333, Fax. +45 31 571 949
Finland: Espoo, Tel. 358 9 615 800, Fax. 358 9 615 80510
France: Suresnes, Tel. +33 1 4099 6161, Fax. +33 1 4099 6493
Germany: Hamburg, Tel. +49 40 2489-0, Fax. +49 40 2489 1400
Greece: Tavros, Tel. +30 1 4894 339/+30 1 4894 239, Fax. +30 1 4814 240
Hong Kong: Kowloon, Tel. +852 2784 3000, Fax. +852 2784 3003
India: Mumbai, Tel. +91 22 4930 311, Fax. +91 22 4930 966/4950 304
Indonesia: Jakarta, Tel. +62 21 794 0040, Fax. +62 21 794 0080
Ireland: Dublin, Tel. +353 1 7640 203, Fax. +353 1 7640 210
Israel: Tel Aviv, Tel. +972 3 6450 444, Fax. +972 3 6491 007
Italy: Milano, Tel. +39 2 6752 2531, Fax. +39 2 6752 2557
Japan: Tokyo, Tel. +81 3 3740 5135, Fax. +81 3 3740 5035
Korea (Republic of)**:** Seoul, Tel. +82 2 709 1472, Fax. +82 2 709 1480
Malaysia: Pulau Pinang, Tel. +60 3 750 5213, Fax. +60 3 757 4880
Mexico: El Paso, Tel. +52 915 772 4020, Fax. +52 915 772 4332
New Zealand: Aukland, Tel. +64 9 815 4000, Fax. +64 9 849 7811
Norway: Oslo, Tel. +47 22 74 8000, Fax +47 22 74 8341
Pakistan: Karachi, Tel. +92 21 587 4641-49,
Fax. +92 21 577 035/+92 21 587 4546
Philippines: Manila, Tel. +63 2 816 6345, Fax. +63 2 817 3474
Poland: Warszawa, Tel. +48 22 612 2594, Fax. +48 22 612 2327
Portugal: Linda-A-Velha, Tel. +351 1 416 3160/416 3333,
Fax. +351 1 416 3174/416 3366
Russia: Moscow, Tel. +7 95 755 6918, Fax. +7 95 755 6919
Singapore: Singapore, Tel. +65 350 2000, Fax. +65 355 1758
South Africa: Johannesburg, Tel. +27 11 470 5911, Fax. +27 11 470 5494
Spain: Barcelona, Tel. +34 3 301 63 12, Fax. +34 3 301 42 43
Sweden: Stockholm, Tel. +46 8 5985 2000, Fax. +46 8 5985 2745
Switzerland: Zürich, Tel. +41 1 488 22 11, Fax. +41 1 481 7730
Taiwan: Taipei, Tel. +886 2 2134 2900, Fax. +886 2 2134 2929
Thailand: Bangkok, Tel. +66 2 745 4090, Fax. +66 2 398 0793
Turkey: Istanbul, Tel. +90 212 279 2770, Fax. +90 212 282 6707
United Kingdom: Dorking Tel. +44 1306 512 000, Fax. +44 1306 512 345
United States:

- Ann Arbor, MI, Tel. +1 734 996 9400, Fax. +1 734 761 2776
- Saugerties, NY, Tel. +1 914 246 2811, Fax. +1 914 246 0487
- San Jose, CA, Tel. +1 408 570 5600, Fax. +1 408 570 5700

Yugoslavia (Federal Republic of)**:** Belgrade, Tel. +381 11 625 344/373,
Fax. +381 11 635 777
Internet:

- Passive Components: www.passives.comp.philips.com

For all other countries apply to:

Philips Components, Marketing Communications, Building BF-1,
P.O. Box 218, 5600 MD EINDHOVEN, The Netherlands
Fax. +31–40–2724547.

North American Sales Offices, Representatives and Distributors

PHILIPS SEMICONDUCTORS
811 East Arques Avenue
P.O. Box 3409
Sunnyvale, CA 94088-3409

ALABAMA
Huntsville
Philips Semiconductors
Phone: (256) 464-9101
(256) 464-0111

Elcom, Inc.
Phone: (256) 830-4001

ARIZONA
Scottsdale
Thom Luke Sales, Inc.
Phone: (602) 451-5400

Tempe
Philips Semiconductors
Phone: (602) 820-2225

CALIFORNIA
Calabasas
Philips Semiconductors
Phone: (818) 880-6304

Centaur Corporation
Phone: (818) 878-5800

Granite Bay
B.A.E. Sales, Inc.
Phone: (916) 652-6777

Irvine
Philips Semiconductors
Phone: (714) 453-0770

Centaur Corporation
Phone: (714) 261-2123

San Diego
Philips Semiconductors
Phone: (619) 560-0242

Centaur Corporation
Phone: (619) 278-4950

San Jose
B.A.E. Sales, Inc.
Phone: (408) 452-8133

Sunnyvale
Philips Semiconductors
Phone: (408) 991-3737

COLORADO
Englewood
Philips Semiconductors
Phone: (303) 792-9011

Thom Luke Sales, Inc.
Phone: (303) 649-9717

CONNECTICUT
Wallingford
JEBCO, Inc.
Phone: (203) 265-1318

FLORIDA
(Norcross, Georgia)
Elcom, Inc.
Phone: (770) 447-8200

GEORGIA
Norcross
Elcom, Inc.
Phone: (770) 447-8200

IDAHO
(Englewood, Colorado)
Thom Luke Sales, Inc.
Phone: (303) 649-9717

ILLINOIS
Itasca
Philips Semiconductors
Phone: (630) 250-0050

INDIANA
Indianapolis
Mohrfield Marketing, Inc.
Phone: (317) 546-6969

Kokomo
Philips Semiconductors
Phone: (765) 459-5355

Leo
Mohrfield Marketing, Inc.
Phone: (219) 627-5355

KANSAS
(Bloomington, Minnesota)
High Technology Sales, Inc.
Phone: (612) 844-9933

KENTUCKY
(Indianapolis, Indiana)
Mohrfield Marketing, Inc.
Phone: (317) 546-6969

MARYLAND
(Rockville Centre, New York)
S-J Associates, Inc.
Phone: (516) 536-4242

MASSACHUSETTS
Chelmsford
JEBCO, Inc.
Phone: (978) 256-5800

Westford
Philips Semiconductors
Phone: (978) 692-6211

MICHIGAN
Farmington Hills
Philips Semiconductors
Phone: (248) 848-7600

Novi
Mohrfield Marketing, Inc.
Phone: (248) 380-8100

MINNESOTA
Bloomington
High Technology Sales, Inc.
Phone: (612) 844-9933

MISSOURI
(Bloomington, Minnesota)
High Technology Sales, Inc.
Phone: (612) 844-9933

NEBRASKA
(Bloomington, Minnesota)
High Technology Sales, Inc.
Phone: (612) 844-9933

NEW JERSEY
Toms River
Philips Semiconductors
Phone: (732) 505-1200
(732) 240-1479

NEW MEXICO
(Scottsdale, Arizona)
Thom Luke Sales, Inc.
Phone: (602) 451-5400

NEW YORK
Rockville Centre
S-J Associates, Inc.
Phone: (516) 536-4242

(Chelmsford, Massachusetts)
JEBCO, Inc.
Phone: (978) 256-5800

NORTH CAROLINA
Cary
Philips Semiconductors
Phone: (919) 462-1332
(919) 462-6361

Raleigh
Elcom, Inc.
Phone: (919) 743-5200

OHIO
(Indianapolis, Indiana)
Mohrfield Marketing, Inc.
Phone: (317) 546-6969

OKLAHOMA
(Richardson, Texas)
OM Associates, Inc.
Phone: (972) 690-96746

OREGON
Beaverton
Philips Semiconductors
Phone: (503) 627-0110

Cascade-Tech
Phone: (503) 645-9660

PENNSYLVANIA
(Indianapolis, Indiana)
Mohrfield Marketing, Inc.
Phone: (317) 546-6969

(Rockville Centre, New York)
S-J Associates, Inc.
Phone: (516) 536-4242

TENNESSEE
Dandridge
Philips Semiconductors
Phone: (423) 397-5557

TEXAS
Austin
OM Associates, Inc.
Phone: (512) 794-9971

Houston
Philips Semiconductors
Phone: (281) 999-1316

OM Associates, Inc.
Phone: (281) 376-6400

Richardson
Philips Semiconductors
Phone: (972) 644-1610

OM Associates, Inc.
Phone: (972) 690-6746

VIRGINIA
(Rockville Centre, New York)
S-J Associates, Inc.
Phone: (516) 536-4242

WISCONSIN
(Bloomington, Minnesota)
High Technology Sales, Inc.
Phone: (612) 844-9933

WASHINGTON
Kirkland
Cascade-Tech
Phone: (425) 822-7299

CANADA
PHILIPS SEMICONDUCTORS CANADA, LTD.

Calgary, Alberta
Tech-Trek, Ltd.
Phone: (403) 291-6866

Kanata, Ontario
Tech-Trek, Ltd.
Phone: (613) 599-8787

Mississauga, Ontario
Tech-Trek, Ltd.
Phone: (905) 238-0366

Richmond, B.C.
Tech-Trek, Ltd.
Phone: (604) 276-8735

Ville St. Laurent, Quebec
Tech-Trek, Ltd.
Phone: (514) 337-7540

MEXICO
Guadalajara
Mepco Centralab, Inc./Philips
Phone: 8-011-52-3-122-2325

Monterrey
Mepco Centralab, Inc./Philips
Phone: 8-011-52-8-399-0164

El Paso, TX
Philips Components
Phone: (915) 772-4020

PUERTO RICO
(Norcross, Georgia)
Elcom, Inc.
Phone: (770) 447-8200

DISTRIBUTORS

Contact one of our local distributors:
Allied Electronics
Arrow Electronics
Future Electronics
Hamilton Hallmark
Marshall Industries
Newark Electronics
Penstock
Richardson Electronics
Zeus Electronics

06/11/98

©Philips Electronics North America Corporation 1998 Printed in U.S.A.

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. +43 160 1010,
Fax. +43 160 101 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: 51 Rue Carnot, 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, Band Box Building, 2nd floor,
254-D, Dr. Annie Besant Road, Worli, MUMBAI 400 025,
Tel. +91 22 493 8541, Fax. +91 22 493 0966

Indonesia: PT Philips Development Corporation, Semiconductors Division,
Gedung Philips, Jl. Buncit Raya Kav.99-100, JAKARTA 12510,
Tel. +62 21 794 0040 ext. 2501, Fax. +62 21 794 0080

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 108-8507,
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

Pakistan: see Singapore

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 319762,
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: Al. Vicente Pinzon, 173, 6th floor, 04547-130 SÃO PAULO, SP,
Brazil, Tel. +55 11 821 2333, Fax. +55 11 821 2382

Spain: Balmes 22, 08007 BARCELONA,
Tel. +34 93 301 6312, Fax. +34 93 301 4107

Sweden: Kottbygatan 7, Akalla, S-16485 STOCKHOLM,
Tel. +46 8 5985 2000, Fax. +46 8 5985 2745

Switzerland: Allmendstrasse 140, CH-8027 ZÜRICH,
Tel. +41 1 488 2741 Fax. +41 1 488 3263

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, International Marketing & 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

© Philips Electronics N.V. 1998

SCH60

All rights are reserved. Reproduction in whole or in part is prohibited without the prior written consent of the copyright owner.

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 other industrial or intellectual property rights.

Printed in USA

215101/32.2M/CR02/pp648

Date of release: July 1998

Document order number: 9397 750 04161

Philips Semiconductors

PHILIPS

Let's make things better.