

intel

# Microcomputer Programmable Logic Handbook





*Intel the Microcomputer Company:*

*When Intel invented the microprocessor in 1971, it created the era of microcomputers. Whether used as microcontrollers in automobiles or microwave ovens, or as personal computers or supercomputers, Intel's microcomputers have always offered leading-edge technology. In the second half of the 1980s, Intel architectures have held at least a 75% market share of microprocessors at 16 bits and above. Intel continues to strive for the highest standards in memory, microcomputer components, modules, and systems to give its customers the best possible competitive advantages.*

## **PROGRAMMABLE LOGIC HANDBOOK**

**1989**



Intel Corporation makes no warranty for the use of its products and assumes no responsibility for any errors which may appear in this document nor does it make a commitment to update the information contained herein.

Intel retains the right to make changes to these specifications at any time, without notice.

Contact your local sales office to obtain the latest specifications before placing your order.

The following are trademarks of Intel Corporation and may only be used to identify Intel Products:

Above, BITBUS, COMMputer, CREDIT, Data Pipeline, ETOX, FASTPATH, Genius, i, i<sup>2</sup>, ICE, ICEL, ICS, iDBP, iDIS, i<sup>2</sup>ICE, iLBX, i<sub>m</sub>, iMDDX, iMMX, Inboard, Insite, Intel, int<sub>e</sub>l, Intel376, Intel386, Intel486, int<sub>e</sub>IBOS, Intel Certified, Intelelevision, int<sub>e</sub>lligent Identifier, int<sub>e</sub>lligent Programming, Intellec, Intellink, iOSP, iPDS, iPSC, iRMK, iRMX, iSBC, iSBX, iSDM, iSXM, KEPRON, Library Manager, MAPNET, MCS, Megachassis, MICROMAINFRAME, MULTIBUS, MULTICHANNEL, MULTIMODULE, ONCE, OpenNET, OTP, PC BUBBLE, Plug-A-Bubble, PROMPT, Promware, QUEST, QueX, Quick-Erase, Quick-Pulse Programming, Ripplemode, RMX/80, RUPI, Seamless, SLD, SugarCube, UPI, and VLSICEL, and the combination of ICE, iCS, iRMX, iSBC, iSBX, iSXM, MCS, or UPI and a numerical suffix, 4-SITE, 376, 386, 486.

MDS is an ordering code only and is not used as a product name or trademark. MDS® is a registered trademark of Mohawk Data Sciences Corporation.

\*MULTIBUS is a patented Intel bus.

Additional copies of this manual or other Intel literature may be obtained from:

Intel Corporation  
Literature Sales  
P.O. Box 58130  
Santa Clara, CA 95052-8130



## **CUSTOMER SUPPORT**

### **EPLD HOTLINE**

The Intel EPLD Technical Hotline is manned by applications personnel from 8:00 a.m. to 5:00 p.m. (PST) every business day. The number (U.S. and Canada) is 1-800-323-EPLD (1-800-323-3753).

### **BBS**

Intel has a Bulletin Board System for registered iPLS II customers to electronically transfer information. A registered user with a modem can log onto the system. The current number is (916) 985-2308. If your communication software supports file transfers, you can receive utilities, software updates, and the latest information on EPLDs via the Bulletin Board.

### **CUSTOMER SUPPORT**

Customer Support is Intel's complete support service that provides Intel customers with hardware support, software support, customer training, and consulting services. For more information contact your local sales offices.

After a customer purchases any system hardware or software product, service and support become major factors in determining whether that product will continue to meet a customer's expectations. Such support requires an international support organization and a breadth of programs to meet a variety of customer needs. As you might expect, Intel's customer support is quite extensive. It includes factory repair services and worldwide field service offices providing hardware repair services, software support services, customer training classes, and consulting services.

### **HARDWARE SUPPORT SERVICES**

Intel is committed to providing an international service support package through a wide variety of service offerings available from Intel Hardware Support.

### **SOFTWARE SUPPORT SERVICES**

Intel's software support consists of two levels of contracts. Standard support includes TIPS (Technical Information Phone Service), updates and subscription service (product-specific troubleshooting guides and COMMENTS Magazine). Basic support includes updates and the subscription service. Contracts are sold in environments which represent product groupings (i.e., iRMX<sup>®</sup> environment).

### **CONSULTING SERVICES**

Intel provides field systems engineering services for any phase of your development or support effort. You can use our systems engineers in a variety of ways ranging from assistance in using a new product, developing an application, personalizing training, and customizing or tailoring an Intel product to providing technical and management consulting. Systems Engineers are well versed in technical areas such as microcommunications, real-time applications, embedded microcontrollers, and network services. You know your application needs; we know our products. Working together we can help you get a successful product to market in the least possible time.

### **CUSTOMER TRAINING**

Intel offers a wide range of instructional programs covering various aspects of system design and implementation. In just three to ten days a limited number of individuals learn more in a single workshop than in weeks of self-study. For optimum convenience, workshops are scheduled regularly at Training Centers worldwide or we can take our workshops to you for on-site instruction. Covering a wide variety of topics, Intel's major course categories include: architecture and assembly language, programming and operating systems, BITBUS<sup>™</sup> and LAN applications.



# Table of Contents

Alphanumeric Index .....	ix
<b>CHAPTER 1</b>	
<b>Overview</b>	
Overview .....	1-1
<b>CHAPTER 2</b>	
<b>EPLDs—Erasable Programmable Logic Devices</b>	
DATA SHEETS	
5C031, 300-Gate CHMOS H-Series Erasable Programmable Logic Device (H-EPLD) .....	2-1
5C032, 300-Gate CHMOS H-Series Erasable Programmable Logic Device (H-EPLD) .....	2-13
5C060, 600-Gate CHMOS H-Series Erasable Programmable Logic Device (H-EPLD) .....	2-26
5C090, 900-Gate CHMOS H-Series Erasable Programmable Logic Device (H-EPLD) .....	2-42
5C121, 1200-Gate CHMOS H-Series Erasable Programmable Logic Device .....	2-59
5C180, 1800-Gate CHMOS Erasable Programmable Logic Device .....	2-74
APPLICATION BRIEFS	
AB-8 Implementing Cascaded Logic in the 5C121 .....	2-106
AB-9 5C121 As a Three and One-Half Digit Display Driver .....	2-111
AB-10 Square Pegs in Round Holes—A Fitting Tutorial for the 5C121 .....	2-116
AB-11 16-Bit Binary Counter Implementation Using the 5C060 EPLD .....	2-128
AB-12 Designing a Mailbox Memory for Two 5C031s .....	2-138
AB-16 Atypical Latch/Register Construction in EPLDs .....	2-152
AB-22 5C032-25 vs. 16V8-25: A Device Comparison .....	2-159
APPLICATION NOTES	
AP-271 Applying the 5C121 Architecture .....	2-165
AP-272 The 5C060 Unification of a CHMOS System .....	2-177
AP-276 Implementing a CMOS Bus Arbiter/Controller in the 5C060 EPLD .....	2-188
AP-307 EPLDs, PLAs, and TTL—Comparing the “Hidden Costs” in Production .....	2-198
AP-321 Fitting the 5C180 .....	2-220
ENGINEERING REPORTS	
ER-22 5C180 vs. EP1800: A Comparison of Device Specifications .....	2-233
TECHNICAL PAPERS	
Techniques for Modular EPLD Designs .....	2-241
ARTICLE REPRINTS	
AR-450 Crosspoint Switch: A PLD Approach .....	2-251
AR-451 A Programmable Logic Mailbox for 80C31 Microcontrollers .....	2-255
AR-454 Regain Lost I/O Ports with Erasable PLDs .....	2-258
<b>CHAPTER 3</b>	
<b>Advanced Architecture EPLDs</b>	
DATA SHEETS	
5AC312, 1-Micron CHMOS Erasable Programmable Logic Device .....	3-1
5AC324, 1-Micron CHMOS EPLD .....	3-19
85C508, Fast 1-Micron CHMOS EPLD .....	3-38
5CBIC, Programmable BUS Interface Controller .....	3-45
APPLICATION NOTES	
AP-317 Implementing a PS/2 POS Using the 5AC312 EPLD .....	3-62
AP-319 Designing with the 5AC312/5AC324 EPLDs .....	3-74
TECHNICAL PAPERS	
Programmable and/Allocatable Based EPLD Addresses the Needs of Complex Combinational and Sequential Designs .....	3-83
Advanced Architecture PLDs Solve Common State Machine Problems .....	3-91

# Table of Contents (Continued)

## CHAPTER 4

### Development Support Tools

#### DATA SHEETS

iPLDS II, The Intel Programmable Logic Development System Version II .....	4-1
iUP-PC, Intel Universal Programmer for the Personal Computer .....	4-11
iUP-200A/iUP-201A Universal PROM Programmers .....	4-18

#### PRODUCT BRIEFS

SCHEMA II-PLD .....	4-25
iPLSII Macro Librarian .....	4-26
PLDUTIL .....	4-27

#### UTILITIES

PAL2ADF Utility .....	4-29
JED2HEX Conversion Utility .....	4-32

#### APPLICATION BRIEFS

AB-18 TTL Macro Library Listing for EPLD Designs .....	4-33
AB-21 EPLD Custom Macro Library Listing for EPLD Designs .....	4-37

#### APPLICATION NOTES

AP-311 Using Macros in EPLD Designs .....	4-41
AP-312 Creating Macros for EPLD Designs .....	4-52

#### TECHNICAL PAPERS

Tools for Optimizing PLD Designs .....	4-62
--	------

## CHAPTER 5

### Appendix

EPLD Third Party Programming Support .....	5-1
PLA to EPLD Replacement .....	5-2
Ordering Information .....	5-3
Device Feature Comparison .....	5-4
EPLD Customer Support .....	5-5
Compatible Computers for iPLDS II .....	5-6

# Alphanumeric Index

5AC312, 1-Micron CHMOS Erasable Programmable Logic Device .....	3-1
5AC324, 1-Micron CHMOS EPLD .....	3-19
5C031, 300-Gate CHMOS H-Series Erasable Programmable Logic Device (H-EPLD) .....	2-1
5C032, 300-Gate CHMOS H-Series Erasable Programmable Logic Device (H-EPLD) .....	2-13
5C060, 600-Gate CHMOS H-Series Erasable Programmable Logic Device (H-EPLD) .....	2-26
5C090, 900-Gate CHMOS H-Series Erasable Programmable Logic Device (H-EPLD) .....	2-42
5C121, 1200-Gate CHMOS H-Series Erasable Programmable Logic Device .....	2-59
5C180, 1800-Gate CHMOS Erasable Programmable Logic Device .....	2-74
5CBIC, Programmable BUS Interface Controller .....	3-45
85C508, Fast 1-Micron CHMOS EPLD .....	3-38
iPLDS II, The Intel Programmable Logic Development System Version II .....	4-1
iUP-200A/iUP-201A Universal PROM Programmers .....	4-18
iUP-PC, Intel Universal Programmer for the Personal Computer .....	4-11



Any of the following products may appear in this publication. If so, it must be noted that such products have counterparts manufactured by Intel Puerto Rico, Inc., Intel Puerto Rico II, Inc., and/or Intel Singapore, Ltd. The product codes/part numbers of these counterpart products are listed below next to the corresponding Intel Corporation product codes/part numbers.

Intel Corporation Product Codes/ Part Numbers	Intel Puerto Rico, Inc. Intel Puerto Rico II, Inc. Product Codes/ Part Numbers	Intel Singapore, Ltd. Product Codes/ Part Numbers	Intel Corporation Product Codes/ Part Numbers	Intel Puerto Rico, Inc. Intel Puerto Rico II, Inc. Product Codes/ Part Numbers	Intel Singapore, Ltd. Product Codes/ Part Numbers
376SKIT	p376SKIT		KM2	pKM2	
903	p903		KM4	pKM4	
904	p904		KM8	pKM8	
913	p913		KNLAN	pKNLAN	
914	p914		KT60	pKT60	
923	p923		KW140	pKW140	
924	p924		KW40	pKW40	
952	p952		KW80	pKW80	
953	p953		M1	pM1	
954	p954		M2	pM2	
ADAICE	pADAICE		M4	pM4	
B386M1	pB386M1		M8	pM8	
B386M2	pB386M2		MDS610	pMDS610	
B386M4	pB386M4		MDX3015	pMDX3015	
B386M8	pB386M8		MDX3015	pMDX3015	
C044KIT	pC044KIT		MDX3016	pMDX3016	
C252KIT	pC252KIT		MDX3016	pMDX3016	
C28	pC28		MDX457	pMDX457	
C32	pC32		MDX457	pMDX457	
C452KIT	pC452KIT		MDX458	pMDX458	
D86ASM	pD86ASM		MDX458	pMDX458	
D86C86	pD86C86		MSA96	pMSA96	
D86EDI	pD86EDI		NLAN	pNLAN	
DCM9111	pDCM9111		PCLINK		sPCLINK
DOSNET	pDOSNET		PCX344A	pPCX344A	
F1	pF1		R286ASM	pR286ASM	
GUPILOGICIID	pGUPILOGICIID		R286EDI	pR286EDI	
H4	pH4		R286PLM	pR286PLM	
I044	pI044		R286SSC	pR286SSC	
I252KIT	pI252KIT		R86FOR	pR86FOR	
I452KIT	pI452KIT		RCB4410		sRCB4410
I86ASM	pI86ASM		RCX920	pRCX920	
ICE386	pICE386		RMX286	pRMX286	
III010	pIII010		RMXNET	pRMXNET	
III086	pIII086		S301	pS301	
III086	pIII086		S386	pS386	
III111	pIII111		SBC010	pSBC010	
III186	pIII186		SBC012	pSBC012	sSBC012
III186	pIII186		SBC020	pSBC020	
III198	pIII198		SBC028	pSBC028	
III212	pIII212		SBC040	pSBC040	
III286	pIII286		SBC056	pSBC056	
III286	pIII286		SBC108	pSBC108	
III515	pIII515		SBC116	pSBC116	
III520	pIII520		SBC18603	pSBC18603	sSBC18603
III520	pIII520		SBC186410	pSBC186410	
III531	pIII531		SBC18651	pSBC18651	sSBC18651
III532	pIII532		SBC186530	pSBC186530	
III533	pIII533		SBC18678	pSBC18678	
III621	pIII621		SBC18848	pSBC18848	sSBC18848
III707	pIII707		SBC18856	pSBC18856	sSBC18856
III707	pIII707		SBC208	pSBC208	sSBC208
III815	pIII815		SBC214	pSBC214	
INA961	pINA961		SBC215	pSBC215	
IPAT86	pIPAT86		SBC220	pSBC220	sSBC220
KAS	pKAS		SBC221	pSBC221	
KC	pKC		SBC28610	pSBC28610	sSBC28610
KH	pKH		SBC28612	pSBC28612	
KM1	pKM1		SBC28614	pSBC28614	

Intel Corporation Product Codes/ Part Numbers	Intel Puerto Rico, Inc. Intel Puerto Rico II, Inc. Product Codes/ Part Numbers	Intel Singapore, Ltd. Product Codes/ Part Numbers	Intel Corporation Product Codes/ Part Numbers	Intel Puerto Rico, Inc. Intel Puerto Rico II, Inc. Product Codes/ Part Numbers	Intel Singapore, Ltd. Product Codes/ Part Numbers
SBC28616	pSBC28616		SBCM310	pSBCM310	
SBC300	pSBC300		SBCM312	pSBCM312	
SBC301	pSBC301		SBCM320	pSBCM320	
SBC302	pSBC302		SBCM340	pSBCM340	
SBC304	pSBC304		SBE96	pSBE96	
SBC307	pSBC307		SBX217	pSBX217	
SBC314	pSBC314		SBX218	pSBX218	
SBC322	pSBC322		SBX270	pSBX270	
SBC324	pSBC324		SBX311	pSBX311	
SBC337	pSBC337		SBX328	pSBX328	
SBC341	pSBC341		SBX331	pSBX331	
SBC386	pSBC386	sSBC386	SBX344	pSBX344	
SBC386116	pSBC386116		SBX350	pSBX350	
SBC386120	pSBC386120		SBX351	pSBX351	
SBC38621	pSBC38621		SBX354	pSBX354	
SBC38622	pSBC38622		SBX488	pSBX488	
SBC38624	pSBC38624		SBX586		sSBX586
SBC38628	pSBC38628		SCHEMIIPLD	pSCHEMIIPLD	
SBC38631	pSBC38631		SCOM	pSCOM	
SBC38632	pSBC38632		SDK51	pSDK51	
SBC38634	pSBC38634		SDK85	pSDK85	
SBC38638	pSBC38638		SDK86	pSDK86	
SBC428	pSBC428	sSBC428	SXM217	pSXM217	
SBC464	pSBC464		SXM28612	pSXM28612	
SBC517	pSBC517		SXM386	pSXM386	
SBC519	pSBC519	sSBC519	SXM544	pSXM544	
SBC534	pSBC534	sSBC534	SXM552	pSXM552	
SBC548	pSBC548		SXM951	pSXM951	
SBC550	TSBC550		SXM955	pSXM955	
SBC550	pSBC550		SYP120	pSYP120	
SBC550	pSBC550		SYP301	pSYP301	
SBC552	pSBC552		SYP302	pSYP302	
SBC556	pSBC556	sSBC556	SYP31090	pSYP31090	
SBC569	pSBC569		SYP311	pSYP311	
SBC589	pSBC589		SYP3847	pSYP3847	
SBC604	pSBC604		SYR286	pSYR286	
SBC608	pSBC608		SYR86	pSYR86	
SBC614	pSBC614		SYS120	pSYS120	
SBC618	pSBC618		SYS310	pSYS310	
SBC655	pSBC655		SYS311	pSYS311	
SBC6611	pSBC6611		T60	pT60	
SBC8010	pSBC8010		TA096	pTA096	
SBC80204	pSBC80204		TA252	pTA252	
SBC8024	pSBC8024	sSBC8024	TA452	pTA452	
SBC8030	pSBC8030		W140	pW140	
SBC8605	pSBC8605	sSBC8605	W280	pW280	
SBC8612	pSBC8612		W40	pW40	
SBC8614	pSBC8614		W80	pW80	
SBC8630	pSBC8630	sSBC8630	XNX286DOC	pXNX286DOC	
SBC8635	pSBC8635	sSBC8635	XNX286DOCB	pXNX286DOCB	
SBC86C38	pSBC86C38	sSBC86C38	XNXIBASE	pXNXIBASE	
SBC8825	pSBC8825	sSBC8825	XNXIDB	pXNXIDB	
SBC8840	pSBC8840		XNXIDESK	pXNXIDESK	
SBC8845	pSBC8845	sSBC8845	XNXIPLAN	pXNXIPLAN	
SBC905	pSBC905		XNXIWORD	pXNXIWORD	
SBCLNK001	pSBCLNK001				







## INTRODUCTION

In today's increasingly competitive marketplace, system designers need to squeeze out every little edge they can get from their designs. This has led to a trend towards better performance, smaller system sizes, lower power requirements and greater system reliability with a strong emphasis on preventing easy duplication of the system design. This trend provided the impetus to the system designers to move away from standard SSI and MSI logic components (54/74 & 4000 series Bipolar and CMOS families) towards a growing class of IC devices variously called 'ASIC' (application specific IC), 'USIC' (user specific IC) or, as referred to in this document, user defined logic.

User defined logic circuits allow system designers, for the first time, to tailor the actual silicon building blocks used in their systems to their individual system needs and requirements. Such customization provides the needed performance, reliability and compactness as well as design security. Cost per gate of logic implemented is also greatly reduced when user defined logic solutions are chosen over standard components.

User defined logic has therefore emerged as the fastest growing segment of the semiconductor industry and has presented its users, the system designers, with a wide range of implementation alternatives namely, programmable logic, gate arrays, standard cell and full custom design. The tradeoffs between these alternatives involves time-to-market, one-time engineering charges, expected unit volume, ease of use of design tools and familiarity with the design methodology.

This document discusses the reasons for the trend to user defined logic devices, briefly describes some of the user defined logic implementation alternatives and covers details on programmable logic devices, the only alternative that is completely user implementable. Tools used to design with programmable logic are also discussed here.

Details on Intel's programmable logic product line, including device terminology and nomenclature, architectural features and development tool features are also described in this document.

## WHY USER DEFINED LOGIC?

System designers prefer user customized ICs for the following reasons:

**a. SMALLER SYSTEM SIZES:** Customized components allow for reducing chip count and saving board space, resulting in smaller system physical dimensions.

**b. LOWER SYSTEM COSTS:** When custom LSI or VLSI components are used instead of standard SSI and MSI logic elements, there is a considerable saving in component cost per system, assembly and manufacturing cost, printed circuit board area and board costs and inventory costs.

**c. HIGHER PERFORMANCE:** Reduced number of ICs contributes to faster system speeds as well as lower power consumption.

**d. HIGHER RELIABILITY:** Since probability of failure is directly related to the number of ICs in the system, a system composed of customized LSI & VLSI chips is statistically much more reliable than the identical system made up of SSI/MSI devices.

**e. DESIGN SECURITY:** Systems designed with standard components can be replicated relatively easily whereas systems that contain user customized ICs cannot be copied because "reverse engineering" of the customized components is extremely difficult. Thus, use of customized ICs allows for the protection of proprietary designs.

**f. INCREASED FLEXIBILITY:** Customized components allow for the tailoring of systems to the end user's specific needs relatively easily. This also allows for upgradability and obsolescence protection.

## USER DEFINED IC— IMPLEMENTATION ALTERNATIVES

Currently, the choices available to the system designer for customization of ICs (see Figure 1) are as follows:

- (1) user programmable ICs—programmable logic devices
- (2) mask programmable ICs—gate arrays
- (3) standard cell based ICs
- (4) full custom ICs

Alternatives (1) & (2) are usually called 'Semicustom' because in these methods only a few (less than three) of the mask layers involved in the manufacture of the IC, are customized to the users' specifications. The later two alternatives (3) & (4), involve customization of all mask layers required to manufacture the ICs to the users' specifications and are therefore called 'Custom'.

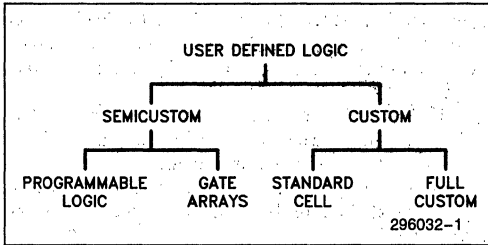


Figure 1. User Defined Logic Implementation Choices

**PROGRAMMABLE LOGIC**

Most user Programmable Logic Devices (PLD) are internally structured as variations of the PLA (programmable logic array) architecture, that is composed of an array of 'AND' gates connected to an array of 'OR' gates (see Figure 2). Programmable logic devices make use of the fact that any logic equation can be converted to an equivalent 'Sum-of-Products' form and can thus be implemented in the 'AND' and 'OR' architecture. This basic PLA structure has been augmented in most PLDs with input and output blocks containing registers, latches and feedback options, that let the user implement sequential logic functions in addition to combinational logic.

The number and locations of the programmable connections between the 'AND' and 'OR' matrices as well as the input and output blocks are predetermined by the architecture of the PLD. The user, depending on

his logic requirements, determines which of these connections he would like to remain open and which he would like to close, through the programming of the PLD. Programmability of these connections is achieved using various memory technologies such as fuses, EPROM cells, EEPROM cells or Static RAM cells (see Figure 3).

User programmability allows for instant customization, very similar to user programmable memories such as PROMs or EPROMs. The user can purchase a PLD off-the-shelf, use a development system running on a personal computer and, in a matter of a few hours, have customized silicon in his hands. Figure 4 compares user-defined logic alternatives.

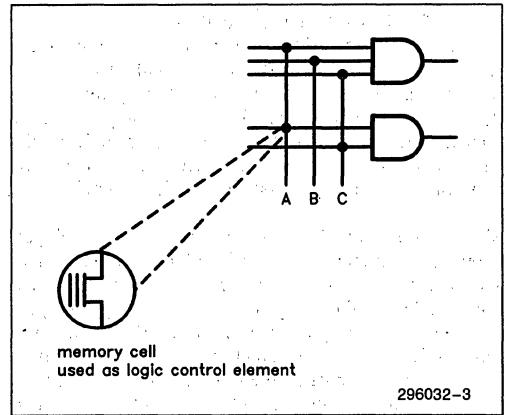


Figure 3. Programmable Connections

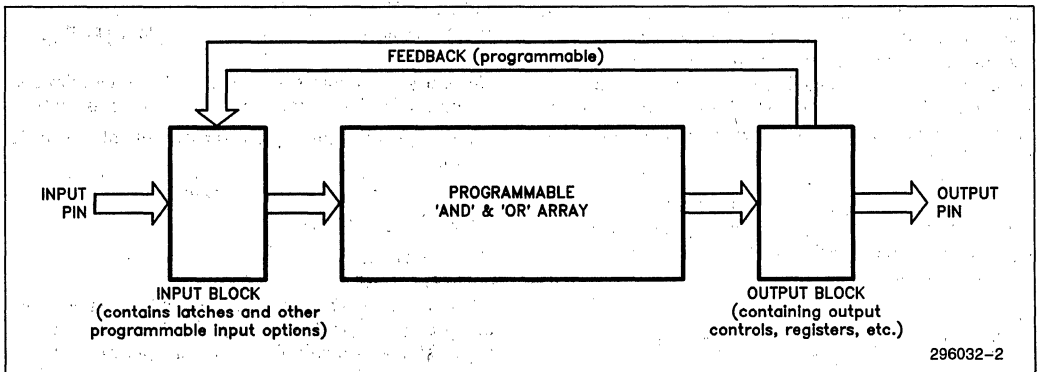


Figure 2. General Architecture of a PLD

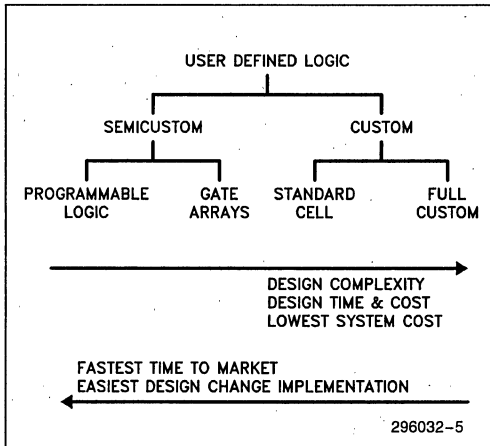


Figure 4. User Defined Logic Alternatives Compared

## LIMITATIONS OF BIPOLAR FUSE TECHNOLOGY FOR PROGRAMMABLE LOGIC DEVICES

Until 1985, all PLDs were built using Bipolar fuse technology. The bipolar fuse based devices, although offering the users the benefits of quick time to market and low development costs, had several inherent limitations.

**a. HIGH POWER CONSUMPTION:** Bipolar processes by nature are power hungry and as a consequence also make for very hot systems, often requiring cooling aids such as heat sinks and fans. They also cannot operate at lower voltages (2-3V) and have a lower level of noise immunity than MOS devices.

**b. LOWER INTEGRATION:** A fuse takes up a large amount of silicon area; this fact in conjunction with the large power requirements makes for smaller levels of integration.

**c. ONE-TIME PROGRAMMABILITY:** Bipolar fuses can only be blown once and cannot be reprogrammed. This does not allow for easy prototyping and could result in significant losses when preprogrammed parts are inventoried and design changes occur.

**d. TESTABILITY:** Since fuses can only be blown once, bipolar PLDs can only be destructively tested. Thus, testing is usually done by sampling or through addi-

tional testing elements incorporated in the chips, which can be blown to examine electrical characteristics. However, such testing methods never allow for 100% testability of all parts shipped. Thus, most users of bipolar programmable logic devices resort to extensive post-programming testing, specific to their applications.

## ERASABLE PROGRAMMABLE LOGIC DEVICES

Erasable programmable logic devices (EPLD) result from the matching of CMOS EPROM technology with the architectures of programmable logic devices. EPLDs use EPROM cells as logic control elements and therefore, when housed in windowed ceramic packages, can be erased with UV light and reprogrammed. Figure 5 shows the architecture of Intel EPLDs.

Other than the obvious benefit of reprogrammability, EPLDs offer several very significant benefits over bipolar PLDs. These are:

**1. LOW POWER CONSUMPTION:** Due to the CMOS technology, these products consume an order of magnitude less power than the equivalent bipolar devices. This allows for the design of complete CMOS systems, that can operate at lower voltages (less than 5V). Also, this makes for cooler systems that do not require cooling systems like fans.

**2. GREATER LOGIC DENSITY:** EPROM cells are an order of magnitude smaller than the smallest fuses. This means that the same function can be accommodated in significantly smaller die area, or that greater amounts of logic can now be incorporated on a single chip. Thus higher integration programmable logic devices result with the use of EPROM elements.

**3. TESTABILITY:** Since the EPROM cells are erasable, the entire EPROM array of the EPLD can be 100% factory tested. Thus, before the part is shipped to the customers, it can be completely tested by the programming and erasure of all the EPROM logic control bits. This testing is therefore independent of any application, in contrast to the bipolar PLDs that need application specific testing.

**4. ARCHITECTURAL ENHANCEMENTS:** The inherent testability of the EPROM elements allows for



significant architectural improvements over bipolar PLDs. New features, such as buried registers, programmable registers, programmable clock control, etc., can now be incorporated because of this testability. These new features allow for greatly increased utilization of the EPLDs and use of these devices in newer applications.

**5. DESIGN SECURITY:** EPLDs are provided with a 'security bit,' which when programmed does not allow anyone to read the programmed pattern. The logic programmed in an EPLD cannot be seen even if the die is examined (unlike bipolar PLDs—a blown fuse is clearly visible) as the stored charges are captured on a buried layer of polysilicon.

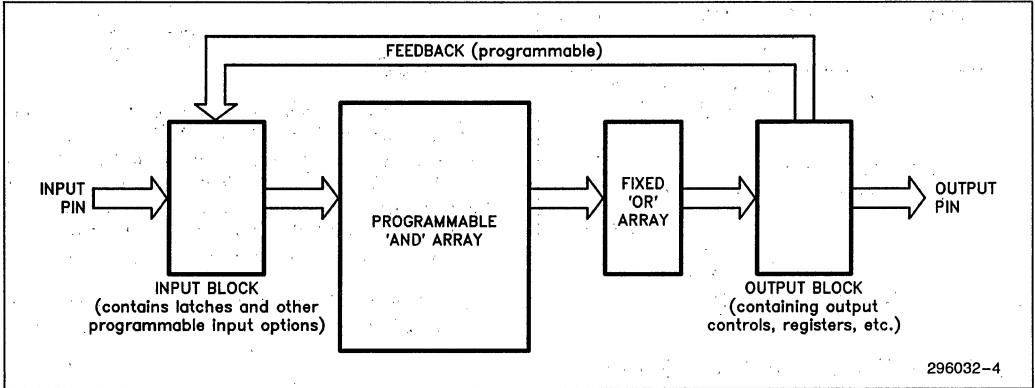


Figure 5. Architecture of Intel EPLDs

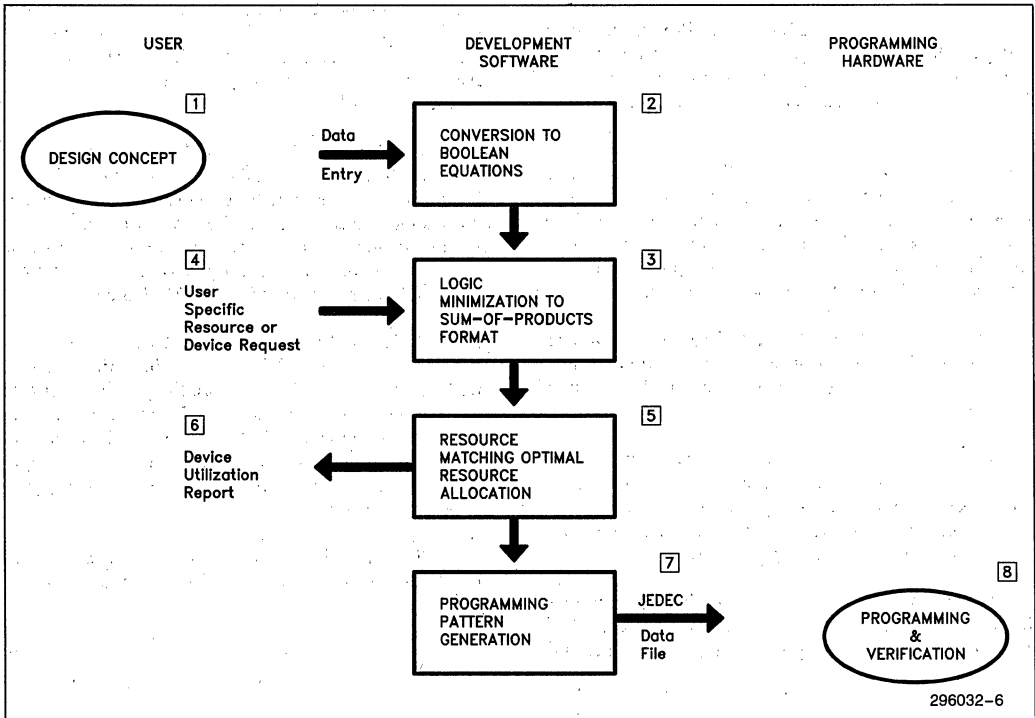


Figure 6. The PLD Design Process

The steps in a generalized design process of programmable logic is shown in Figure 6 and described in the following paragraphs.

**STEP 1:** The user decides on the logic he wants implemented in the PLD and enters the design into the PC or workstation. This **Design Entry** may be done by the following methods: (i)**SCHEMATIC CAPTURE**—A 'Mouse' or some other graphics input device is used to input schematics of the logic, (ii)**NET LIST ENTRY**—If the user has a hand drawn schematic he can enter the design into the computer by describing the symbols and interconnections in words using a standardized format called a net list (without using a graphics input device), (iii)**STATE EQUATION/DIAGRAM ENTRY**—Entry of a sequential design involving states and transitions between states. In the state diagram method circles represent states and the arrows interconnecting them represent the transitions. Equations or a state table can also be used to define a state machine, and (iv)**BOOLEAN EQUATIONS**—this is the most common design entry method. The logic is described in boolean algebraic equations.

**STEP 2:** The software converts all design entry data into boolean equations.

**STEP 3:** The boolean equations entered are converted to the sum of products format after logic reduction (minimization of the logic through heuristic algorithms).

**STEP 4:** The user has the ability to choose the PLD he would like the design implemented on. He can enter device choice and/or he can also enter in specific choices on the device as regards pinout he would like etc . . .

**STEP 5:** The software optimizes the logic equations to fit into the device using the minimum amount of resources (resources are input pins, output pins, registers and product terms and macrocells). This step is where the user requirements as regards required pins are taken into account. The user requests are viewed as constraints during the optimization process.

**STEP 6:** The software, at the end of the resource optimization/allocation, produces a report detailing the resources used up in fitting the design on the PLD. This report allows the user to incrementally stuff in logic by going back to Step 1 from this stage. Also, if the design overflowed the PLD, i.e., did not fit in the user chosen device, the software lists out the resources needed to complete the fit. The requirements such as four more inputs, one register more and one more output (are needed to complete the design) gives the user data in choosing a bigger PLD or in partitioning the initial design to fit into two devices.

**STEP 7:** The next step is to generate the appropriate programming pattern for the PLD. This is a standard

"JEDEC" format interface and allows the output of the design software to be compatible with any piece of PROM programming hardware.

**STEP 8:** PROM programmer is used to program the pattern stored in the JEDEC file onto the PLD. Also, at this stage fuse programmed PLDs (bipolar) are functionally tested using test vectors included in the JEDEC file information.

## CHMOS TECHNOLOGY IN EPLDs

EPLDs are manufactured with Intel's proprietary CHMOS (Complementary High Performance MOS) technology. The backbone of the process is the integration of both a P and an N channel MOS transistor on the same substrate. In addition, EPLD's programmable architecture makes use of Intel's proven EPROM cell for programmable array interconnections as well as macrocell configuration bits. These cells are programmed electrically and erased with ultraviolet light. For details on Intel's CHMOS technology and EPROM cells technology, refer to the *Components Quality/Reliability Handbook*, Order Number 210997.

## CHMOS DESIGN GUIDELINES

Designing with Intel EPLDs is relatively straightforward if the following guidelines are observed:

- Minimize the occurrence of ESD (electro-static discharge) when storing or handling EPLDs.
- Observe good design rules in printed circuit board layout.
- Provide adequate decoupling capacitance at both the device and the board level.
- Connect all unused inputs to  $V_{CC}$  or  $GND$  (CHMOS inputs should not be left floating).

## Electrostatic Discharge

The two most common sources of electrostatic discharge are the human body and a charged environment.

A charged human body that touches a device lead discharges electricity into the device. Electrostatic discharge from people handling devices has long been recognized by manufacturers and users of all MOS products. Human body static electricity can be controlled by using ground straps and anti-static spray on carpeted floors. CHMOS devices should also be stored and carried in conductive tubes or anti-static foam to minimize exposure to ESD from people.

Discharge also occurs when an integrated circuit is charged to one potential and then contacts a conductor at another potential. This type of ESD can be reduced

by grounding all work surfaces, grounding all handling equipment, removing static generators such as paper from the work area, and erasing EPLDs in metal tubes, metal trays, or conductive foam.

## PCB Layout

The best PCB performance is obtained when close attention is paid to  $V_{CC}$ , GND, and signal traces.  $V_{CC}$  and GND should be gridded to minimize inductive reactance and to approximate a trace layer. Clocks should be laid out to minimize crosstalk. Ensure adequate power supply and ground pins on the board connector.

## Decoupling

Decouple each EPLD with a ceramic capacitor in the range of 0.01 to 0.2  $\mu\text{F}$ , depending on board frequency and current consumption. For most applications, a 0.1  $\mu\text{F}$  capacitor will suffice. The following equation produces the exact value:

$$C = \frac{\Delta I_{CC}}{\Delta V / \Delta T}$$

where  $C$  = capacitor value  
 $\Delta I_{CC}$  = maximum switched current  
 $\Delta V$  = switching level  
 $\Delta T$  = switching time

For boards that contain mixed logic (EPLDs and TTL), observe both EPLD and TTL decoupling practices.

## Unused Inputs

To minimize noise receptivity and power consumption, all unused inputs to EPLDs should be connected to  $V_{CC}$  or GND. By default, iPLS II software assigns unused inputs to GND. These pins, shown on the pinout representation of the iPLS II report file, should be connected to ground on the PCB. Pins listed as RESERVED on the report file must be left floating. Pins marked N.C. have no internal device connections and can also be left floating.

## BOOLEAN MINIMIZATION TECHNIQUES FOR PLA ARCHITECTURES

Minimization plays an important role in logic design. Methods for minimization can be grouped into two classes. Class 1 includes manual methods for minimization, such as Boolean reduction or Karnaugh mapping. Class 2 is computer-assisted minimization.

Tabular methods like Karnaugh maps are efficient up to a certain point. Past that point, however, computer-assisted minimization plays a crucial part in efficient design. Even at the computer-assisted stage, the choice of minimizer software has an impact on time and the confidence level of the reduced equation (i.e., is it in the smallest possible form).

iPLS II software includes a minimizer that uses the ESPRESSO algorithms. ESPRESSO was developed by U.C. Berkeley during the summers of 1981 and 1982 in an effort to study the various strategies used by the MINI logic minimizer developed by IBM, [HON 74] and PRESTO developed by D. Brown [BRO 81]. ESPRESSO uses many of the core principles in MINI and PRESTO while improving on the speed and efficiency of their algorithms.

The primary advantage of the ESPRESSO minimizer becomes apparent when designing large finite state machines or complex, product-term intensive logic designs. In these cases, ESPRESSO arrives at the minimize solution sooner, and frequently reduces the logic to a smaller number of product terms. In certain cases where other CAD packages such as ABEL™ (PRESTO) or CUPL™ minimize equations to greater than 8 product terms, iPLS II further reduces these equations to allow the design to fit into devices supporting up to 8 product terms.

For more information on ESPRESSO, refer to *Logic Minimization Algorithms for VLSI Synthesis*, Brayton, Hachtel, McMullen, and Sangiovanni-Vincentelli, Kluwer Academic Publishers.

## References

- [BRO 81] D.W. Brown, "A State-Machine Synthesizer—SMS", Proc. 18th Design Automation Conference, pp. 301–304. Nashville, June 1981.
- [HON 74] S. J. Hong, R. G. Cain and D. L. Ostapko, "MINI: A heuristic approach to logic minimization." *IBM Journal of Research and Development*, Vol. 18, pp. 443–458, September 1974.

ABEL™ is a trademark of Data I/O Corporation

CUPL™ is a trademark of Personal CAD Systems, Inc.

## LOGIC REFRESHER COURSE

Minimization of EPLD logic equations is normally performed by sophisticated algorithms that eliminate the need for tedious manual reductions. The sections provided here contain logic reference tables for cases where manual reduction techniques may be desirable.

### Boolean Algebra

The Sum-of-Product architecture used in EPLDs makes Boolean algebra ideal for design analysis. The following tables summarize standard Boolean functions.

#### Properties

$A * B$	$= B * A$	Commutative Property
$A + B$	$= B + A$	
$A * (B * C)$	$= (A * B) * C$	Associative Property
$A + (B + C)$	$= (A + B) + C$	
$A * (B + C)$	$= A * B + A * C$	Distributive Property
$A + B * C$	$= (A + B) * (A + C)$	

#### Postulates

$0 * 0 = 0$	$0 + 0 = 0$	$\overline{0} = 1$
$0 * 1 = 0$	$0 + 1 = 0$	$\overline{1} = 0$
$1 * 1 = 1$	$1 + 1 = 1$	

#### Theorems

$A * 0 = 0$	$A + 0 = A$	$\overline{\overline{A}} = A$
$A * 1 = A$	$A + 1 = 1$	
$A * A = A$	$A + A = A$	
$A * \overline{A} = 0$	$A + \overline{A} = 1$	

#### DeMorgan's Theorems

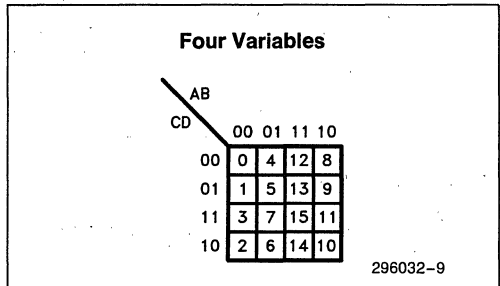
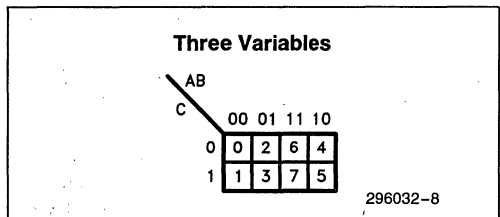
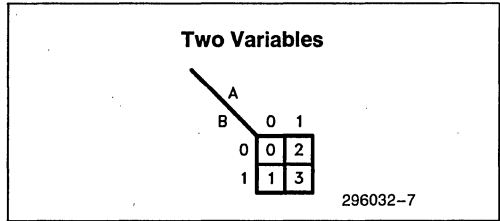
$\overline{(A + B + C + D)}$	$= \overline{A} * \overline{B} * \overline{C} * \overline{D}$
$\overline{(A * B * C * D)}$	$= \overline{A} + \overline{B} + \overline{C} + \overline{D}$

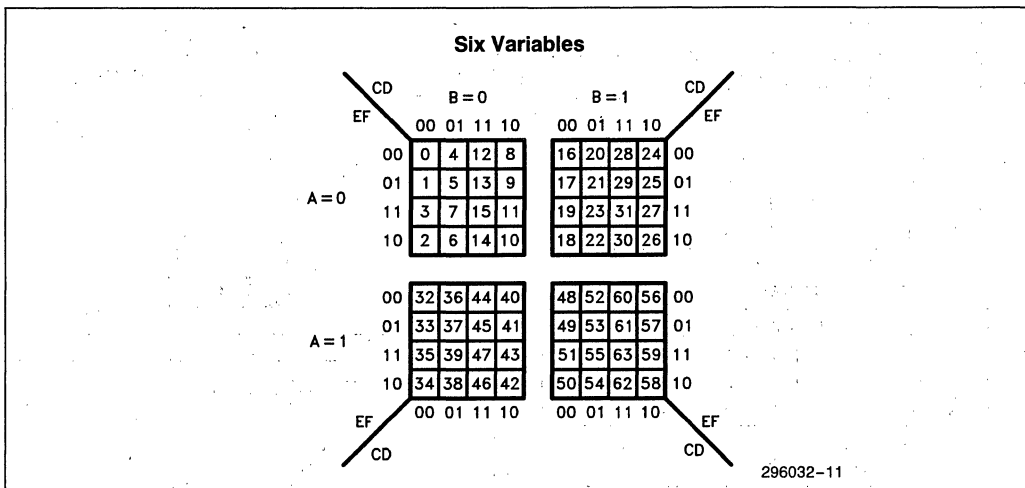
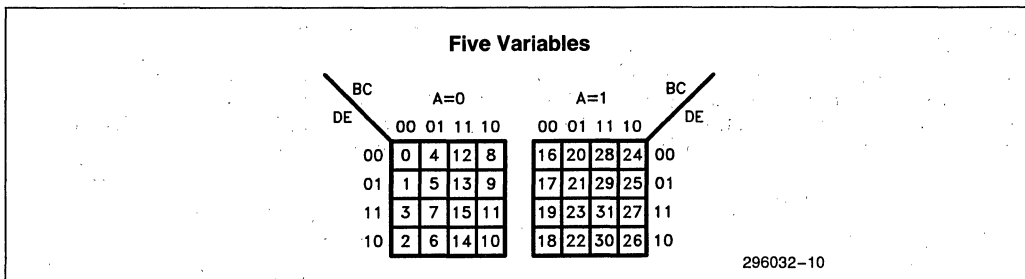
#### Logic Functions

$A * A$	$= A$ AND A
$A + A$	$= A$ OR A
$\overline{A}$	$= A$ NOT
$A \oplus B = A$ EXCLUSIVE OR B	$= A\overline{B} + \overline{A}B$

### Karnaugh Maps

Graphical representation of data is usually easier to analyze than strings of ones and zeros. The Karnaugh Map techniques take advantage of this capability and provide an important tool to the logic designer.





## Flip-Flop Tables

This subsection includes truth tables and excitation tables for the flip-flops supported by EPLDs.

**D Truth Table**

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

**D Excitation Table**

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

**T Truth Table**

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

**T Excitation Table**

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



**JK Truth Table**

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

**JK Excitation Table**

Q <sub>N</sub>	Q <sub>N+1</sub>	J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

**SR Truth Table**

S	R	Q <sub>N</sub>	Q <sub>N+1</sub>
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	Illegal	

**JK Excitation Table**

Q <sub>N</sub>	Q <sub>N+1</sub>	S	R
0	0	0	X
0	1	1	0
1	0	0	1
1	1	X	0

**NOTES:**

Q<sub>N</sub> = Present State  
 Q<sub>N+1</sub> = Next State  
 X = Don't Care

**AUTOMATIC STANDBY MODE (TURBO BIT)**

INTEL EPLDs contain a programmable bit, the Turbo Bit, that optimizes devices for speed or power savings. When TURBO = ON, EPLDs are optimized for speed. When TURBO = OFF, they are optimized for power savings by automatically entering standby mode

when input transitions are not detected over a short period of time. The following paragraphs describe how the Turbo Bit affects power and speed in EPLDs.

**Turbo Off (Low Power)**

Intel EPLDs contain circuitry that monitors all inputs for transitions. When a transition is detected while the device is in standby mode, the circuit generates an active pulse. The leading edge of this pulse wakes the device up and the device responds according to its programming, changing outputs as necessary. If no new transitions occur during the active pulse, the device enters standby mode again. Outputs are always held valid in standby mode. Input transitions that occur during the active mode interval retrigger the active pulse. The active pulse is different depending on the device (5C060, 5AC312, etc), but is typically 2-4 times the propagation delay for a particular device.

In applications with infrequent input transitions, standby mode can result in significant power savings (see the appropriate data sheet for standby power vs. active power). The slight speed loss associated with waking up a device is in the range of 0-10 ns, which is small enough to allow standby mode to be used with most applications (see the appropriate data sheet for effect of Turbo Bit on performance).

**Turbo On (Faster Speed)**

In cases where the slight speed loss associated with waking a device from standby mode cannot be traded off to save power, the Turbo bit can be enabled for maximum speed operation. With the Turbo Bit enabled, the device is always in active mode, thus avoiding the wakeup delay. Note that data sheet performance is specified with the Turbo Bit enabled.

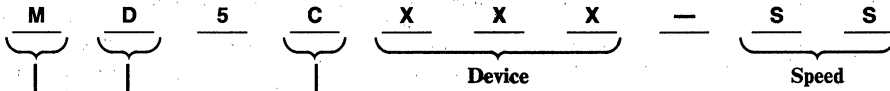
The Turbo Bit is enabled/disabled via a TURBO = ON or TURBO = OFF statement in an iPLS II ADF OPTIONS: statement. It can also be enabled/disabled by editing the JEDEC file using device programmable software. With TURBO = ON the device will be programmed for high speed; with TURBO = OFF the device will be programmed for automatic standby (power savings). The default state is OFF.

**PACKAGING**

Intel EPLDs are available in several packages to meet the wide requirements of customer applications. Current information on available packages is available from your local Intel field sales engineer. Detailed information on package dimensions, etc. for a particular package is provided in *Packaging Outlines and Dimensions*, Order Number 321369, which covers all Intel packages.

## ORDERING INFORMATION

Intel EPLDs are identified as follows:



**Technology**

- C — CHMOS
- AC — Advanced CHMOS

**Package Type**

- A — Hermetic, Pin Grid Array
  - D — Hermetic, Type D (Cerdip) Dip
  - N — Plastic, Leaded Chip Carrier
  - CJ — Ceramic, J Leaded Chip Carrier
  - P — Plastic Dip and Plastic Flatpack
  - R — Hermetic, Leadless Chip Carrier
  - X — Unpackaged Device
- A — Indicates automotive operating temperature range ( $-40^{\circ}\text{C}$  to  $+125^{\circ}\text{C}$ )
  - J — Indicates a JAN qualified device, but is for internal identification purposes only. All JAN devices must be ordered by M38510 part number. (Example: M38510/42001 BQB), and will be marked in accordance with MIL-M-38510 specifications.
  - L — Indicates extended operating temperature range ( $-40^{\circ}\text{C}$  to  $+85^{\circ}\text{C}$ ) express product with 160 + 8 hrs. dynamic burn-in.
  - \*M — Indicates military operating temperature range ( $-55^{\circ}\text{C}$  to  $+125^{\circ}\text{C}$ )
  - Q — Indicates commercial temperature range ( $0^{\circ}\text{C}$  to  $70^{\circ}\text{C}$ ) express product with 160 + 8 hrs. dynamic burn-in.
  - T — Indicates extended temperature range ( $-40^{\circ}\text{C}$  to  $+85^{\circ}\text{C}$ ) express product without burn-in.
  - No letter indicates commercial temperature range ( $0^{\circ}\text{C}$  to  $70^{\circ}\text{C}$ ) without burn-in.

**Examples:**

QD5C060-45 Commercial with burn-in, ceramic Dip, 060 (600 gate) device, 45 nanosecond.

\*On military temperature devices, B suffix indicates MIL-STD-883C level B processing.

---

# **EPLDs Erasable Programmable Logic Devices**

---

**2**

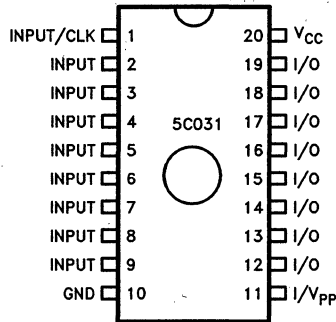




# 5C031

## 300 GATE CHMOS H-SERIES ERASABLE PROGRAMMABLE LOGIC DEVICE (H-EPLD)

- High Performance, Low Power Replacement for SSI & MSI Devices and Bipolar PLDs.
- Up to 18 Inputs (10 Dedicated & 8 I/O) and 8 Outputs.
- Eight Macrocells with Programmable I/O Architecture.
- 100% Generically Testable EPROM Logic Control Array.
- High Performance Upgrade for All Commonly Used 20-pin PLDs.
- CHMOS EPROM Technology Based UV Erasable.
- Programmable "Security Bit" Allows Total Protection of Proprietary Designs
- $I_{CC}$  (standby) 35 mA (max)  
 $I_{CC}$  (10 MHz) 40 mA (max)
- $t_{pD} = 40$  ns (max)
- 20-pin 0.3" Windowed Cerdip Package  
(See Packaging Spec., Order # 231369)
- 100% Compatible with EP310



Pin Configuration

290154-1



The Intel 5C031 H-EPLD (H-series Erasable Programmable Logic Device) is capable of implementing over 300 equivalent gates of user-customized logic functions through programming. This device can be used to replace bipolar programmable logic arrays and LS TTL and 74HC (CMOS) SSI and MSI logic devices. The 5C031 can also be used as a direct, low-power replacement for almost all common 20-pin fuse-based programmable logic devices. With its flexible programmable I/O architecture, this device has advanced functional capabilities beyond that of typical programmable logic.

The 5C031 H-EPLD uses CHMOS EPROM (floating gate) cells as logic control elements instead of fuses. The CHMOS EPROM technology reduces power consumption of H-EPLDs to less than 20% of a comparable bipolar device without sacrificing speed performance. In addition, the use of Intel's advanced CHMOS II-E EPROM process technology enables greater logic densities to be achieved with superior speed and low-power performance over other comparable devices. EPROM technology allows these devices to be 100% factory tested by programming and erasing all the EPROM logic control elements.

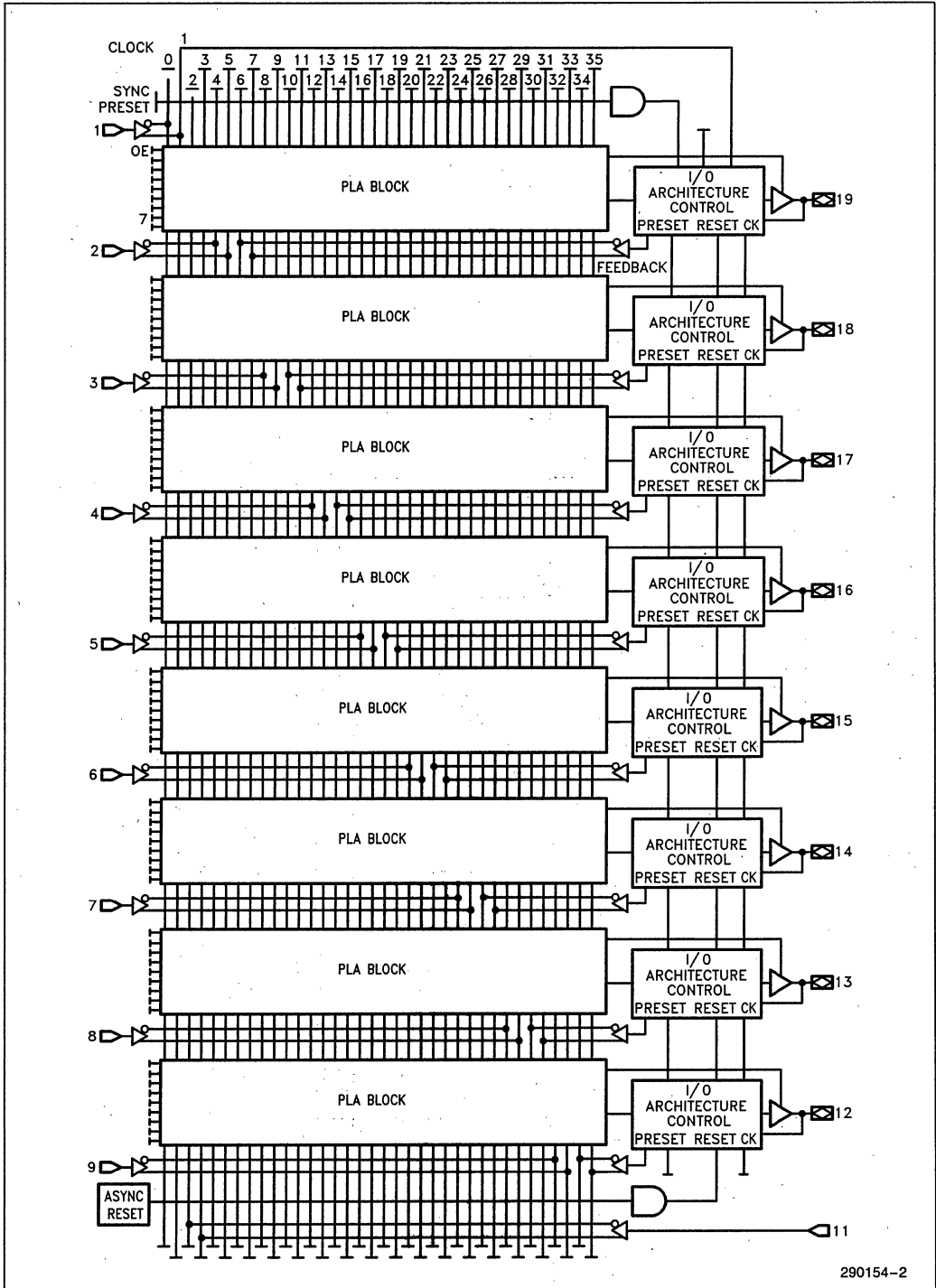
The 5C031 is housed in a windowed 0.3" 20-pin DIP and has the benefits of being an ideal prototyping tool with its highly flexible I/O architecture.

## ARCHITECTURE DESCRIPTION

The architecture of the 5C031 is based on the "Sum of Products" PLA (Programmable Logic Array) structure with a programmable AND array feeding into a fixed OR array. This device can accommodate both combinational and sequential logic functions. A proprietary programmable I/O architecture provides individual selection of either combinational or registered output and feedback signals, all with selectable polarity.

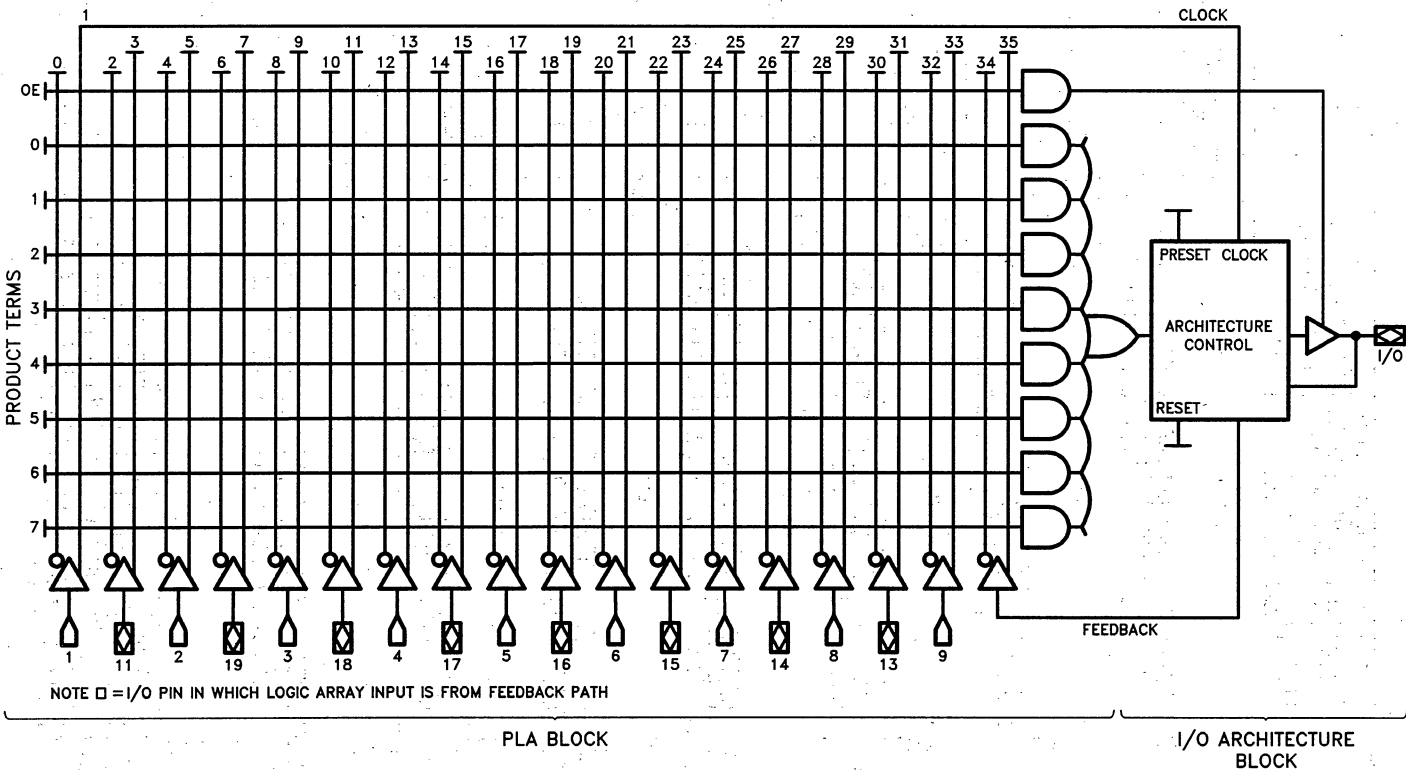
The 5C031 contains 10 dedicated inputs as well as 8 input/output pins. These I/O pins can be individually configured to be inputs, outputs or bi-directional I/O pins. Each of these I/O pins is connected to a macrocell. The 5C031 contains 8 identical macrocells organized as shown in Figure 1.

Each macrocell (see Figure 2) consists of a PLA (programmable logic array) block and an I/O architecture block, which contains a "D" type register. The PLA block consists of eight 36-input AND gates (TRUE & COMPLEMENT of 10 dedicated inputs plus the 8 feedback inputs from the eight macrocells), feeding into an OR gate. The output of this PLA block is fed into the I/O architecture block. The different I/O and feedback options that are achievable from the 5C031 I/O block are shown in Figure 3.



290154-2

Figure 1. 5C031 Architecture



290154-3

Figure 2. Logic Array Macrocell

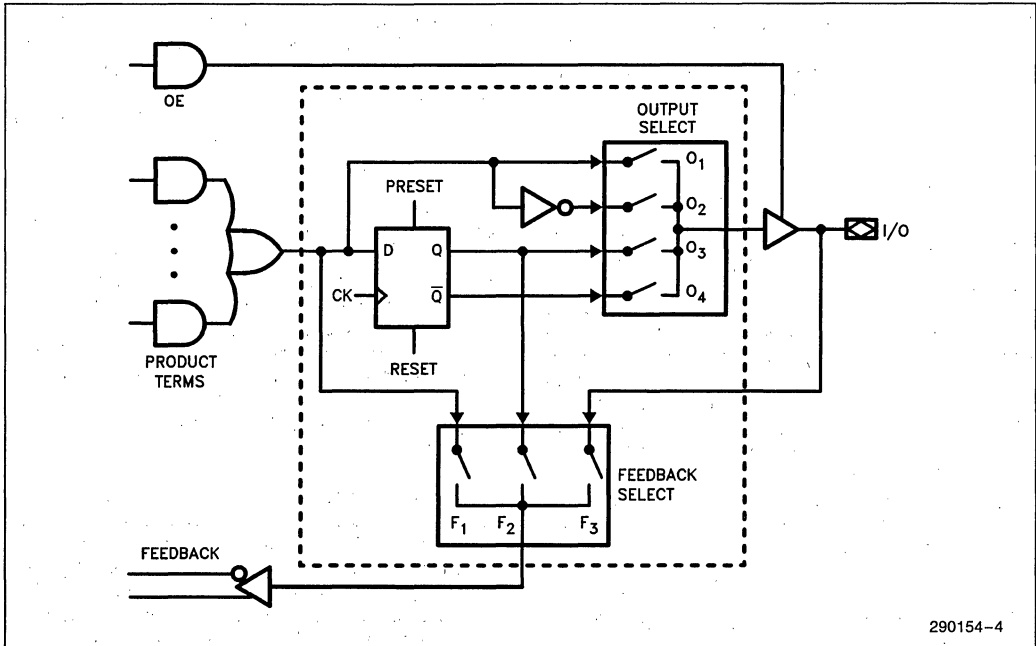


Figure 3. 5C031 I/O Architecture Control

290154-4

## 20 PIN CMOS COMPATIBILITY

The 5C031 is architected to be a logical superset of most 20 pin bipolar programmable array logic (PAL\*) devices. The I/O and logic sections of the 5C031 device can be configured to emulate any of the devices listed below. Designers can make use of this feature by reducing the power of PAL based systems (EPLDs are much lower power), replacing multiple PAL inventory items with a single EPLD. Designers can also create new 20 pin PLD configurations by utilizing the individual logic and output controls of each macrocell.

List of PAL devices logically compatible with the 5C031.

10H8	16L2
12H6	16L8
14H4	16R8
16H2	16R6
16H8	16R4
16C1	16P8A
10LB	16RP8A
12L6	16RP6A
14L4	16RP4A

\*PAL is a registered trademark of Monolithic Memories, Inc.

## Erased-State Configuration

Prior to programming or after erasing, the I/O structure is configured for combinatorial active low output with input (pin) feedback.

## ERASURE CHARACTERISTICS

Erasure characteristics of the 5C031 are such that erasure begins to occur upon exposure to light with wavelengths shorter than approximately 4000Å. It should be noted that sunlight and certain types of fluorescent lamps have wavelengths in the 3000–4000Å. Data shows that constant exposure to room level fluorescent lighting could erase the typical 5C031 in approximately three years, while it would take approximately one week to cause erasure when exposed to direct sunlight. If the 5C031 is to be exposed to these types of lighting conditions for extended periods of time, conductive opaque labels should be placed over the device window to prevent unintentional erasure.

The recommended erasure procedure for the 5C031 is exposure to shortwave ultraviolet light with a wavelength of 2537Å. The integrated dose (i.e., UV intensity  $\times$  exposure time) for erasure should be a minimum of fifteen (15) Wsec/cm<sup>2</sup>. The erasure time with this dosage is approximately 15 to 20 minutes using an ultraviolet lamp with a 12,000  $\mu$ W/cm<sup>2</sup> power rating. The 5C031 should be placed within one inch of the lamp tubes during erasure. The maximum integrated dose the 5C031 can be exposed to without damage is 7258 Wsec/cm<sup>2</sup> (1 week at 12,000  $\mu$ W/cm<sup>2</sup>). Exposure to high intensity UV light for longer periods may cause permanent damage to the device.

## PROGRAMMING CHARACTERISTICS

Initially, and after erasure, all the EPROM control bits of the 5C031 are connected (in the "1" state). Each of the connected control bits are selectively disconnected by programming the EPROM cells into their "0" state. Programming voltage and waveform specifications are available by request from Intel to support programming of the 5C031.

## intelligent Programming™ Algorithm

The 5C031 supports the intelligent Programming Algorithm which rapidly programs Intel H-ELPDs (and EPROMs) using an efficient and reliable method.

The intelligent Programming Algorithm is particularly suited to the production programming environment. This method greatly decreases the overall programming time while programming reliability is ensured as the incremental program margin of each bit is continually monitored to determine when the bit has been successfully programmed.

## FUNCTIONAL TESTING

Since the logical operation of the 5C031 is controlled by EPROM elements, the device is completely testable. Each programmable EPROM bit controlling the internal logic is tested using application-independent test program patterns. After testing, the devices are erased before shipment to customers. No post-programming tests of the EPROM array are required.

The testability and reliability of EPROM-based programmable logic devices is an important feature over similar devices based on fuse technology. Fuse-based programmable logic devices require a user to perform post-programming tests to insure proper programming. These tests must be done at the device level because of the cumulative error effect. For example, a board containing ten devices each possessing a 2% device fallout translates into an 18% fallout at the board level (it should be noted that programming fallout of fuse-based programmable logic devices is typically 2% or higher).

## DESIGN RECOMMENDATIONS

For proper operation, it is recommended that all input and output pins be constrained to the voltage range  $GND < (V_{IN} \text{ or } V_{OUT}) < V_{CC}$ . Unused inputs should be tied to an appropriate logic level (e.g. either  $V_{CC}$  or  $GND$ ) to minimize device power consumption. Reserved pins (as indicated in the iPLDS REPORT file) should be left floating (no connect) so that the pin can attain the appropriate logic level. A power supply decoupling capacitor of at least 0.2  $\mu$ F must be connected directly between  $V_{CC}$  and  $GND$  pins of the device.

As with all CMOS devices, ESD handling procedures should be used with the 5C031 to prevent damage to the device during programming, assembly, and test.

## DESIGN SECURITY

A single EPROM bit provides a programmable design security feature that controls the access to the data programmed into the device. If this bit is set, a proprietary design within the device cannot be copied. This EPROM security bit enables a higher degree of design security than fused-based devices since programmed data within EPROM cells is invisible even to microscopic evaluation. The EPROM security bit, along with all the other EPROM control bits, will be reset by erasing the device.

## LATCH-UP IMMUNITY

All of the input, I/O, and clock pins of the 5C031 have been designed to resist latch-up which is inherent in inferior CMOS structures. The 5C031 is designed with Intel's proprietary CHMOS II-E EPROM process. Thus, each of the 5C031 pins will not experience latch-up with currents up to 100 mA and voltages ranging from  $-1V$  to  $V_{CC} + 1V$ . Furthermore, the programming pin is designed to resist latch-up to the 13.5V maximum device limit.

## INTEL PROGRAMMABLE LOGIC DEVELOPMENT SYSTEM II (iPLDS II)

iPLDS II provides all the tools needed to design with Intel H-Series EPLDs or compatible devices. In addition to providing development assistance, iPLDS II insulates the user from having to know all the intricate details of EPLD architecture (the machine will optimize a design to benefit from architectural features). It contains comprehensive third generation software that supports four different design entry methods, minimizes logic, does automatic pin assignments and produces the best design fit for the selected EPLD. It is user friendly with guided menus, on-line Help messages and soft key inputs.

In addition, the iPLDS II contains programmer hardware in the form of an iUP-PC Universal Programmer-Personal Computer to enable the user to program EPLDs, read and verify programmed devices and also to graphically edit programming files. The software generates industry standard JEDEC object

code output files which can be downloaded to other programmers as well.

The iPLDS II has interfaces to popular schematic capture packages to enable designs to be entered using schematics. An integrated schematic entry method is provided by SCHEMA II-PLD, a low-cost schematic capture package that supports EPLD primitives and user-defined macro symbols. SCHEMA II-PLD contains the EPLD Design Manager, which provides a single user interface to both SCHEMA II-PLD and iPLS II software. The other design formats supported are Boolean equation entry and State Machine design entry.

The iPLDS operates on the IBM† PC/XT, PC/AT, or other compatible machine with the following configuration:

1. At least one floppy disk drive and hard disk drive.
2. MS-DOS†† Operating System Version 3.0 or greater.
3. 512K Memory (640K recommended).
4. Intel iUP-PC Universal Programmer-Personal Computer and GUI Adaptor (supplied with iPLDS).
5. A color monitor is suggested.

Detailed information on the Intel Programmable Logic Development System II is contained in a separate Intel data sheet. (Order Number: 280168)

†IBM Personal Computer is a registered trademark of International Business Machines Corporation.

††MS-DOS is a registered trademark of Microsoft Corporation.

## ADF PRIMITIVES SUPPORTED

The following ADF primitives are supported by this device:

INP	RONF
CONF	ROCF
COCF	RORF
CORF	ROIF
COIF	NORF
NOCF	

## ORDERING INFORMATION

t <sub>PD</sub> (ns)	t <sub>CO</sub> (ns)	f <sub>MAX</sub> (MHz)	Order Code	Package	Operating Range
40	24	29.5	D5C031-40	CERDIP	Commercial
50	28	22.5	D5C031-50	CERDIP	Commercial

**ABSOLUTE MAXIMUM RATINGS\***

Symbol	Parameter	Min	Max	Units
$V_{CC}$	Supply Voltage(1)	-2.0	7.0	V
$V_{PP}$	Programming Supply Voltage(1)	-2.0	13.5	V
$V_I$	DC Input Voltage(1)(2)	-0.5	$V_{CC} + 0.5$	V
$t_{stg}$	Storage Temperature	-65	+150	°C
$t_{amb}$	Ambient Temperature(3)	-10	+85	°C

\*Notice: Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

**NOTES:**

1. Voltages with respect to ground.
2. Minimum DC input is -0.5V. During transitions, the inputs may undershoot to -2.0V or overshoot to 7.0V for periods less than 20 ns under no load conditions.
3. Under bias. Extended temperature versions are also available.

**RECOMMENDED OPERATING CONDITIONS**

Symbol	Parameter	Min	Max	Unit
$V_{CC}$	Supply Voltage	4.75	5.25	V
$V_{IN}$	Input Voltage	0	$V_{CC}$	V
$V_O$	Output Voltage	0	$V_{CC}$	V
$T_A$	Operating Temperature	0	+70	°C
$t_R$	Input Rise Time		500	ns
$t_F$	Input Fall Time		500	ns

**D.C. CHARACTERISTICS**  $T_A = 0^\circ$  to  $+70^\circ\text{C}$ ,  $V_{CC} = 5\text{V} \pm 5\%$ 

Symbol	Parameter/Test Conditions	Min	Typ	Max	Unit
$V_{IH}^{(4)}$	High Level Input Voltage	2.0		$V_{CC} + 0.3$	V
$V_{IL}^{(4)}$	Low Level Input Voltage	-0.3		0.8	V
$V_{OH}^{(5)}$	High Level Output Voltage $I_O = -4.0$ mA D.C., $V_{CC} = \text{min.}$	2.4			V
$V_{OL}$	Low Level Output Voltage $I_O = 4.0$ mA D.C., $V_{CC} = \text{min.}$			0.45	V
$I_I$	Input Leakage Current $V_{CC} = \text{max.}$ , $\text{GND} < V_{IN} < V_{CC}$			$\pm 10$	$\mu\text{A}$

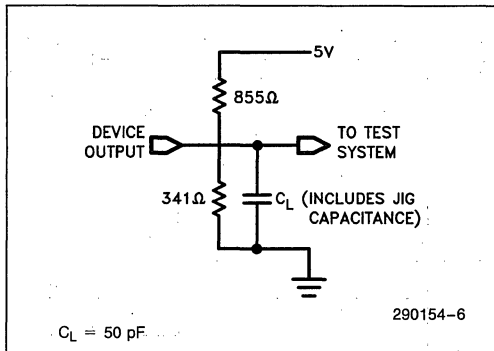
**D.C. CHARACTERISTICS**  $T_A = 0^\circ \text{ to } +70^\circ\text{C}$ ,  $V_{CC} = 5V \pm 5\%$  (Continued)

Symbol	Parameter/Test Conditions	Min	Typ	Max	Unit
$I_{OZ}$	Output Leakage Current $V_{CC} = \text{max.}, GND < V_{OUT} < V_{CC}$			$\pm 10$	$\mu\text{A}$
$I_{SC}^{(6)}$	Output Short Circuit Current $V_{CC} = \text{max.}, V_{OUT} = 0.5V$			10	mA
$I_{CC}$	Power Supply Current $V_{CC} = \text{max.}, V_{IN} = V_{CC}$ or GND No Load, Input Freq. = 1 MHz Active mode (Turbo = Off) Device prog. as 8-bit Ctr.		15	40	mA

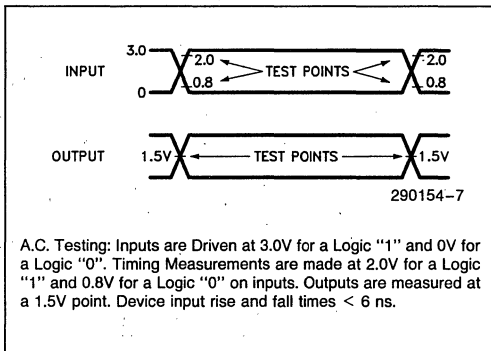
**NOTES:**

4. Absolute values with respect to device GND; all over and undershoots due to system or tester noise are included.
5.  $I_O$  at CMOS levels (3.84V) = -2 mA.
6. Not more than 1 output should be tested at a time. Duration of that test must not exceed 1 second.

**A.C. TESTING LOAD CIRCUIT**



**A.C. TESTING INPUT, OUTPUT WAVEFORM**



**CAPACITANCE**

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
$C_{IN}$	Input Capacitance	$V_{IN} = 0V, f = 1.0 \text{ MHz}$			20	pF
$C_{OUT}$	Output Capacitance	$V_{OUT} = 0V, f = 1.0 \text{ MHz}$			20	pF
$C_{CLK}$	Clock Pin Capacitance	$V_{IN} = 0V, f = 1.0 \text{ MHz}$			20	pF
$C_{VPP}$	$V_{PP}$ Pin	Pin 11			50	pF



**A.C. CHARACTERISTICS**  $T_A = 0^\circ\text{C to } +70^\circ\text{C}$ ,  $V_{CC} = 5V \pm 5\%$ , Turbo Bit Programmed<sup>(7)</sup>

Symbol	From	To	5C031-40 EP310-3			5C031-50 EP310			Unit
			Min	Typ	Max	Min	Typ	Max	
$t_{PD}$	I/O	Comb. Output			40			50	ns
$t_{PZX}^{(8)}$	I or I/O	Output Enable			40			50	ns
$t_{PXZ}^{(8)}$	I or I/O	Output Disable			40			50	ns
$t_{CLR}$	Asynch Reset	Q Reset			40			50	ns

**NOTES:**

 7. Typical Values are at  $T_A = 25^\circ\text{C}$ ,  $V_{CC} = 5V$ , Active Mode

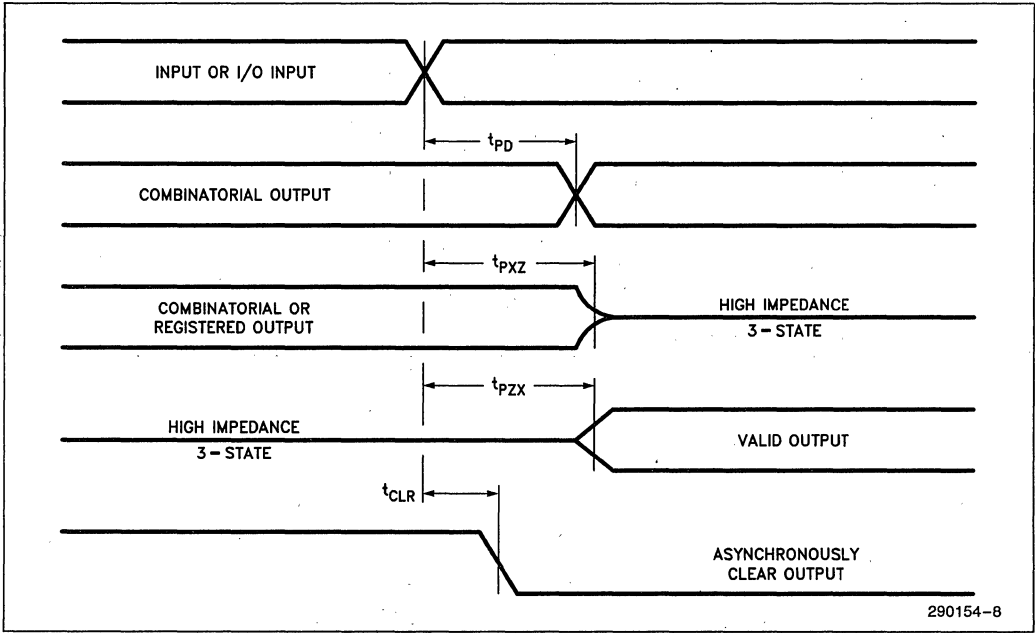
 8.  $t_{PZX}$  and  $t_{PXZ}$  are measured at  $\pm 0.5V$  from steady state voltage as driven by spec. output load.  $t_{PXZ}$  is measured with  $C_L = 5\text{ pF}$ .

**SYNCHRONOUS CLOCK MODE A.C. CHARACTERISTICS**
 $T_A = 0^\circ\text{C to } +70^\circ\text{C}$ ,  $V_{CC} = 5.0V \pm 5\%$ , Turbo Bit On<sup>(7)</sup>

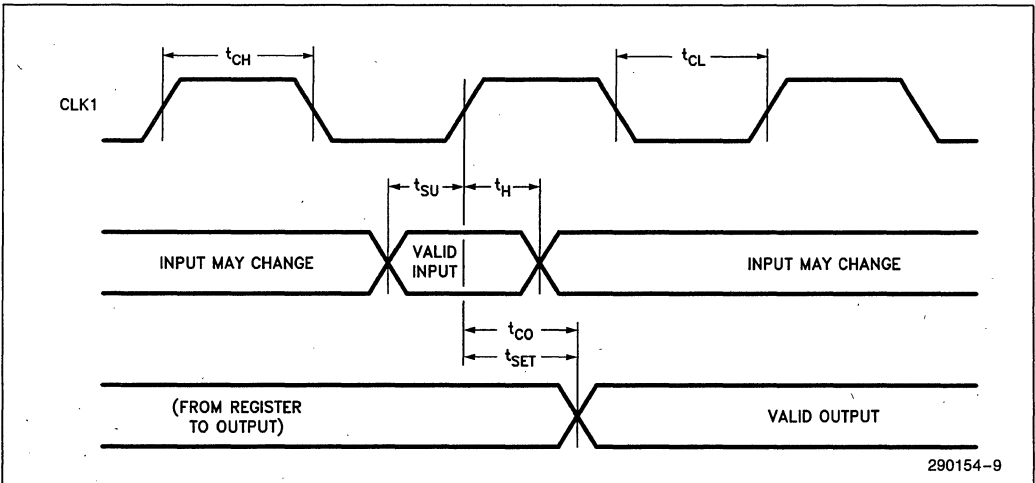
Symbol	Parameter	5C031-40 EP310-3			5C031-50 EP310			Unit
		Min	Typ	Max	Min	Typ	Max	
$f_{MAX}$	Max. Frequency (Pipelined) $1/(t_{CL} + t_{CH})$ — No Feedback			29.4			22.7	MHz
$f_{CNT}$	Max. Count Frequency $1/t_{CNT}$ — With Feedback			22.2			18.1	MHz
$t_{SU}$	I/O Setup Time to CLK	30			32			ns
$t_H$	I or I/O Hold after CLK High	0			0			ns
$t_{CO}$	CLK High to Output Valid			24			28	ns
$t_{CNT}$	Register Output Feedback to Register Input — Internal Path	45			55			ns
$t_{CH}$	CLK High Time	17			22			ns
$t_{CL}$	CLK Low Time	17			22			ns
$t_{SET}$	Synch. Set to Q Set			40			50	ns

SWITCHING WAVEFORMS

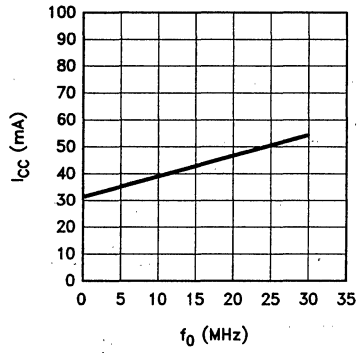
COMBINATORIAL MODE



SYNCHRONOUS CLOCK MODE



5C031 Current in Relation to Frequency



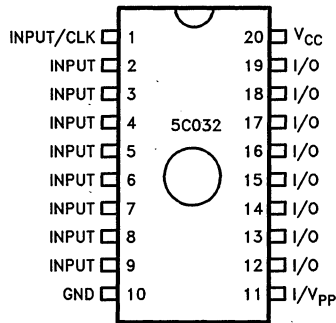
Conditions: T<sub>A</sub> = 0°C, V<sub>CC</sub> = 5.25V

290154-10



# 5C032 300 GATE CHMOS H-SERIES ERASABLE PROGRAMMABLE LOGIC DEVICE (H-EPLD)

- High Performance, Low Power Replacement for SSI & MSI Devices and Bipolar PLDs
- Up to 18 Inputs (10 Dedicated & 8 I/O) and 8 Outputs
- Eight Macrocells with Programmable I/O Architecture
- 100% Generically Testable EPROM Logic Control Array
- High Performance Upgrade for All Commonly Used 20-pin PLDs
- CHMOS EPROM Technology Based UV Erasable
- Programmable "Security Bit" Allows Total Protection of Proprietary Designs
- $I_{CC}$  (standby) 100  $\mu$ A (max)  
 $I_{CC}$  (10 MHz) 25 mA (max)
- $t_{PD} = 25$  ns (max)
- 20-pin 0.3" Ceramic and Plastic DIP Package  
(See Packaging Spec., Order #231369)
- 100% Compatible with EP320



Pin Configuration

290155-1

The Intel 5C032 H-EPLD (H-series Erasable Programmable Logic Device) is capable of implementing over 300 equivalent gates of user-customized logic functions through programming. This device can be used to replace bipolar programmable logic arrays and LS TTL and 74HC (CMOS) SSI and MSI logic devices. The 5C032 can also be used as a direct, low-power replacement for almost all common 20-pin fuse-based programmable logic devices. With its flexible programmable I/O architecture, this device has advanced functional capabilities beyond that of typical programmable logic.

The 5C032 H-EPLD uses CHMOS EPROM (floating gate) cells as logic control elements instead of fuses. The CHMOS EPROM technology reduces power consumption of H-EPLDs to less than 20% of a comparable bipolar device without sacrificing speed performance. In addition, the use of Intel's advanced CHMOS II-E EPROM process technology enables greater logic densities to be achieved with superior speed and low-power performance over other comparable devices. Intel's 5C032 has the benefit of "zero" stand-by power not available on other programmable logic devices. EPROM technology allows these devices to be 100% factory tested by programming and erasing all the EPROM logic control elements.

The 5C032 with its superior speed and power performance and its plastic package is an ideal production vehicle for high-volume manufacturing. Most commonly used 20-pin bipolar PLDs can be easily replaced with this device allowing for tremendous power consumption savings without sacrificing speed of operation.

## ARCHITECTURE DESCRIPTION

The architecture of the 5C032 is based on the "Sum of Products" PLA (Programmable Logic Array) structure with a programmable AND array feeding into a fixed OR array. This device can accommodate both combinational and sequential logic functions. A proprietary programmable I/O architecture provides individual selection of either combinational or registered output and feedback signals, all with selectable polarity.

The 5C032 contains 10 dedicated inputs as well as 8 input/output pins. These I/O pins can be individually configured to be inputs, outputs or bi-directional I/O pins. Each of these I/O pins is connected to a macrocell. The 5C032 contains 8 identical macrocells organized as shown in Figure 1.

Each macrocell (see Figure 2) consists of a PLA (programmable logic array) block and an I/O architecture block, which contains a "D" type register. The PLA block consists of eight 36-input AND gates (TRUE & COMPLEMENT of 10 dedicated inputs plus the 8 feedback inputs from the eight macrocells), feeding into an OR gate. The output of this PLA block is fed into the I/O architecture block. The different I/O and feedback options that are available in the 5C032 I/O block are shown in Figure 3.

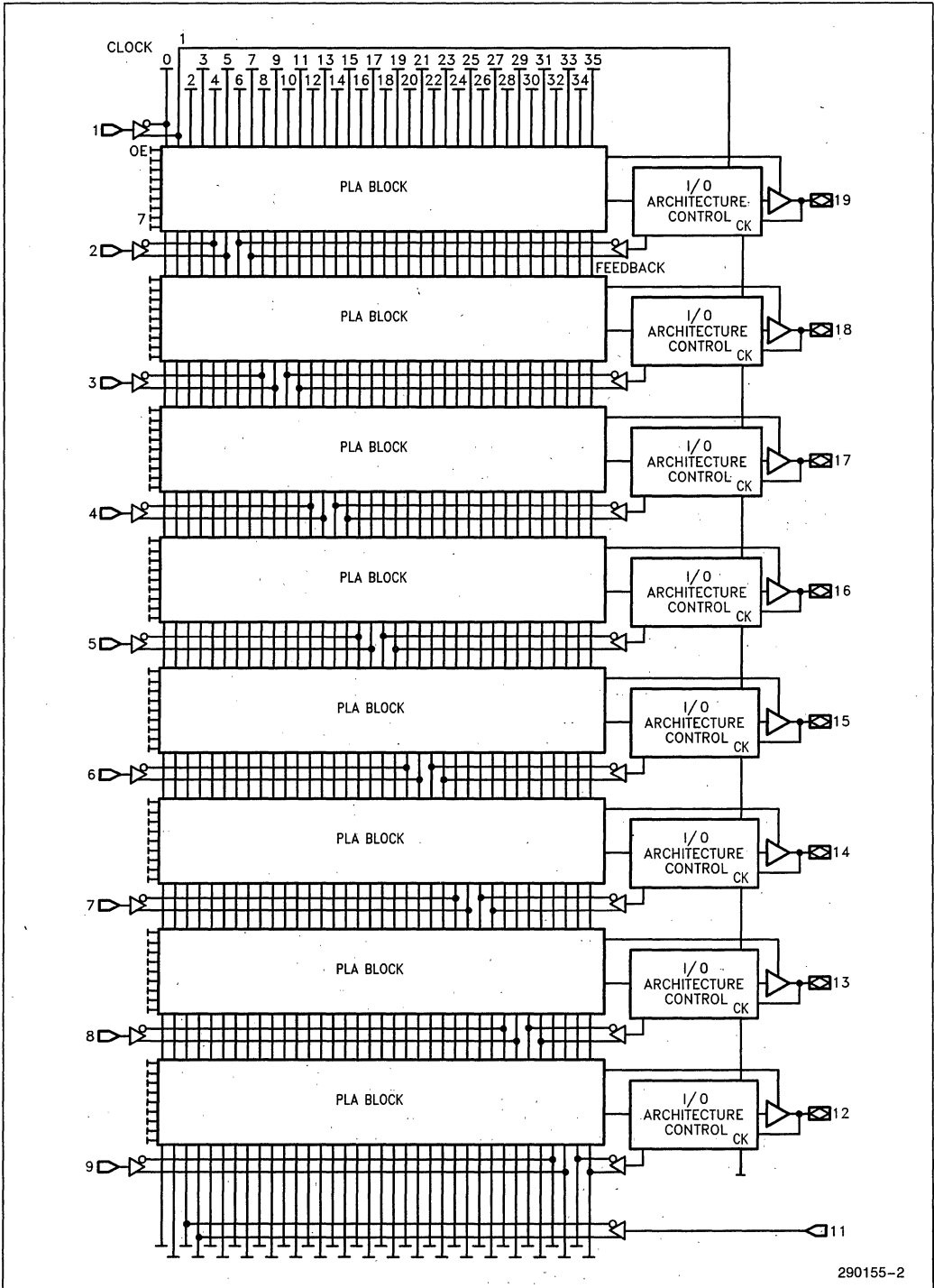


Figure 1. 5C032 Architecture

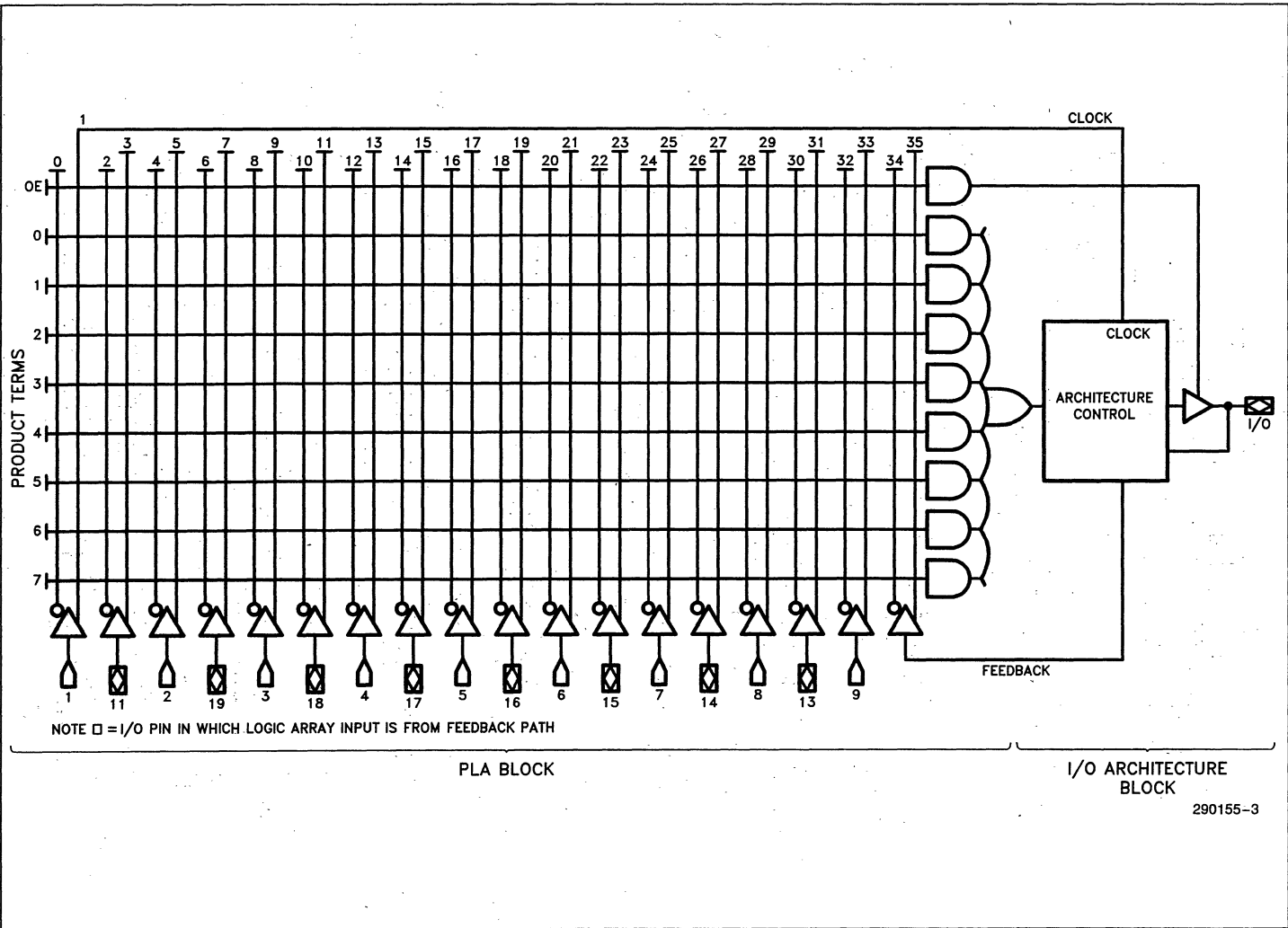


Figure 2. Logic Array Macrocell

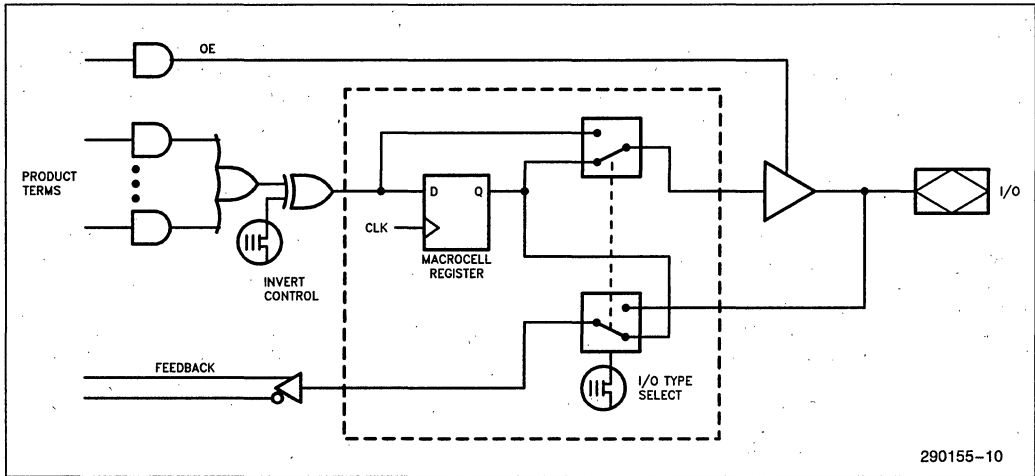


Figure 3. 5C032 I/O Architecture Control

## 20 PIN CMOS COMPATIBILITY

The 5C032 is architected to be a logical superset of most 20 pin bipolar programmable array logic (PAL\*) devices. The I/O and logic sections of the 5C032 device can be configured to emulate any of the devices listed below. Designers can make use of this feature by reducing the power of PAL based systems (EPLDs are much lower power), replacing multiple PAL inventory items with a single EPLD. Designers can also create new 20 pin PLD configurations by utilizing the individual logic and output controls of each macrocell.

List of PAL devices logically compatible with the 5C032.

16V8	16L2
10H8	16L8
12H6	16R8
14H4	16R6
16H2	16R4
16H8	16P8A
16C1	16RP8A
10LB	16RP6A
12L6	16RP4A
14L4	

\*PAL is a registered trademark of Monolithic Memories, Inc.



## Erased-State Configuration

Prior to programming or after erasing, the I/O structure is configured for combinatorial active low output with input (pin) feedback.

## ERASURE CHARACTERISTICS

Erasure characteristics of the 5C032 are such that erasure begins to occur upon exposure to light with wavelengths shorter than approximately 4000Å. It should be noted that sunlight and certain types of fluorescent lamps have wavelengths in the 3000–4000Å. Data shows that constant exposure to room level fluorescent lighting could erase the typical 5C032 in approximately three years, while it would take approximately one week to cause erasure when exposed to direct sunlight. If the 5C032 is to be exposed to these types of lighting conditions for extended periods of time, conductive opaque labels should be placed over the device window to prevent unintentional erasure.

The recommended erasure procedure for the 5C032 is exposure to shortwave ultraviolet light with a wavelength of 2537Å. The integrated dose (i.e., UV intensity  $\times$  exposure time) for erasure should be a minimum of fifteen (15) Wsec/cm<sup>2</sup>. The erasure time with this dosage is approximately 15 to 20 minutes using an ultraviolet lamp with a 12,000  $\mu$ W/cm<sup>2</sup> power rating. The 5C032 should be placed within one inch of the lamp tubes during erasure. The maximum integrated dose the 5C032 can be exposed to without damage is 7258 Wsec/cm<sup>2</sup> (1 week at 12,000  $\mu$ W/cm<sup>2</sup>). Exposure to high intensity UV light for longer periods may cause permanent damage to the device.

## PROGRAMMING CHARACTERISTICS

Initially, and after erasure, all the EPROM control bits of the 5C032 are connected (in the "1" state). Each of the connected control bits are selectively disconnected by programming the EPROM cells into their "0" state. Programming voltage and waveform specifications are available by request from Intel to support programming of the device.

## intelligent Programming™ Algorithm

The 5C032 supports the intelligent Programming Algorithm which rapidly programs Intel H-ELPDs (and EPROMs) using an efficient and reliable method. The intelligent Programming Algorithm is particularly suited to the production programming environment.

This method greatly decreases the overall programming time while programming reliability is ensured as the incremental program margin of each bit is continually monitored to determine when the bit has been successfully programmed.

## FUNCTIONAL TESTING

Since the logical operation of the 5C032 is controlled by EPROM elements, the device is completely testable. Each programmable EPROM bit controlling the internal logic is tested using application-independent test program patterns. After testing, the devices are erased before shipment to customers. No post-programming tests of the EPROM array are required.

The testability and reliability of EPROM-based programmable logic devices is an important feature over similar devices based on fuse technology. Fuse-based programmable logic devices require a user to perform post-programming tests to insure proper programming. These tests must be done at the device level because of the cumulative error effect. For example, a board containing ten devices each possessing a 2% device fallout translates into an 18% fallout at the board level (it should be noted that programming fallout of fuse-based programmable logic devices is typically 2% or higher).

## DESIGN RECOMMENDATIONS

For proper operation, it is recommended that all input and output pins be constrained to the voltage range  $GND < (V_{IN} \text{ or } V_{OUT}) < V_{CC}$ . Unused inputs should be tied to an appropriate logic level (e.g. either  $V_{CC}$  or  $GND$ ) to minimize device power consumption. Reserved pins (as indicated in the iPLDS REPORT file) should be left floating (no connect) so that the pin can attain the appropriate logic level. A power supply decoupling capacitor of at least 0.2  $\mu$ F must be connected directly between  $V_{CC}$  and  $GND$  pins of the device.

As with all CMOS devices, ESD handling procedures should be used with the 5C032 to prevent damage to the device during programming, assembly, and test.

## DESIGN SECURITY

A single EPROM bit provides a programmable design security feature that controls the access to the data programmed into the device. If this bit is set, a proprietary design within the device cannot be copied. This EPROM security bit enables a higher degree of design security than fused-based devices since programmed data within EPROM cells is invisi-

ble even to microscopic evaluation. The EPROM security bit, along with all the other EPROM control bits, will be reset by erasing the device.

### AUTOMATIC STAND-BY MODE

The 5C032 contains a programmable bit, the Turbo Bit, that optimizes operation for speed or for power savings. When the Turbo Bit is programmed (TURBO = ON), the device is optimized for maximum speed. When the Turbo bit is not programmed (TURBO = OFF), the device is optimized for power savings by entering standby mode during periods of inactivity.

Figure 4 shows the device entering standby mode approximately 100 ns after the last input transition. When the next input transition is detected, the device returns to active mode. Wakeup time adds an additional 15 ns to the propagation delay through the device as measured from the first input. No delay will occur if an output is dependent on more than one input and the last of the inputs changes after the device has returned to active mode.

After erasure, the Turbo Bit is unprogrammed (OFF); automatic standby mode is enabled. When the Turbo Bit is programmed (ON), the device never enters standby mode.

### LATCH-UP IMMUNITY

All of the input, I/O, and clock pins of the 5C032 have been designed to resist latch-up which is inherent in inferior CMOS structures. The 5C032 is designed with Intel's proprietary CHMOS II-E EPROM

process. Thus, each of the 5C032 pins will not experience latch-up with currents up to 100 mA and voltages ranging from -1V to  $V_{CC} + 1V$ . Furthermore, the programming pin is designed to resist latch-up to the 13.5V maximum device limit.

### INTEL PROGRAMMABLE LOGIC DEVELOPMENT SYSTEM II (iPLDS II)

iPLDS II provides all the tools needed to design with Intel H-Series EPLDs or compatible devices. In addition to providing development assistance, iPLDS II insulates the user from having to know all the intricate details of EPLD architecture (the machine will optimize a design to benefit from architectural features). It contains comprehensive third generation software that supports four different design entry methods, minimizes logic, does automatic pin assignments and produces the best design fit for the selected EPLD. It is user friendly with guided menus, on-line Help messages and soft key inputs.

In addition, the iPLDS II contains programmer hardware in the form of an iUP-PC Universal Programmer-Personal Computer to enable the user to program EPLDs, read and verify programmed devices and also to graphically edit programming files. The software generates industry standard JEDEC object code output files which can be downloaded to other programmers as well.

The iPLDS II has interfaces to popular schematic capture packages to enable designs to be entered using schematics. An integrated schematic entry method is provided by SCHEMA II-PLD, a low-

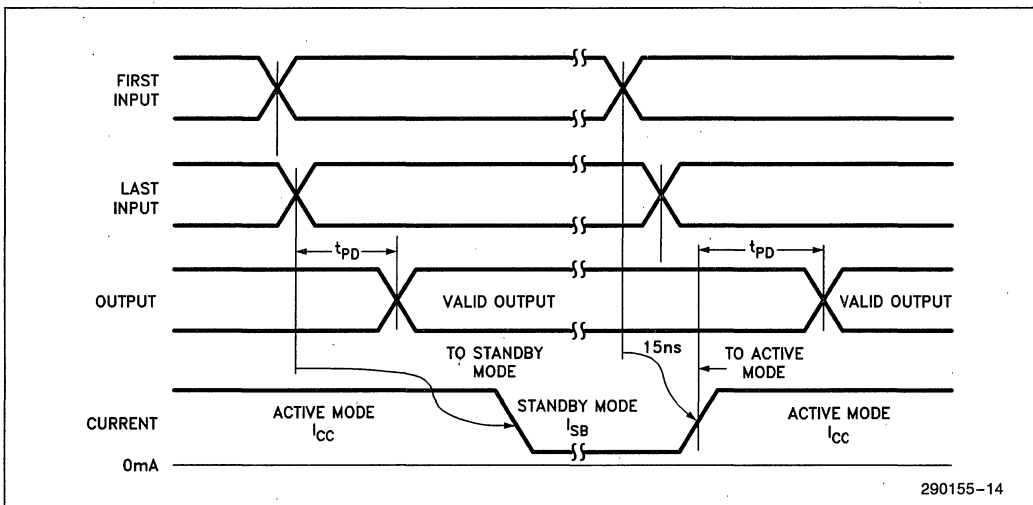


Figure 4. 5C032 Standby and Active Mode Transitions

cost schematic capture package that supports EPLD primitives and user-defined macro symbols. SCHEMA II-PLD contains the EPLD Design Manager, which provides a single user interface to both SCHEMA II-PLD and iPLS II software. The other design formats supported are Boolean equation entry and State Machine design entry.

The iPLDS operates on the IBM† PC/XT, PC/AT, or other compatible machine with the following configuration:

1. At least one floppy disk drive and hard disk drive.
2. MS-DOS†† Operating System Version 3.0 or greater.
3. 512K Memory (640K recommended).
4. Intel iUP-PC Universal Programmer-Personal Computer and GUPI Adaptor (supplied with iPLDS II).
5. A color monitor is suggested.

Detailed information on the Intel Programmable Logic Development System II is contained in a separate Intel data sheet. (Order Number: 280168)

† IBM Personal Computer is a registered trademark of International Business Machines Corporation.

†† MS-DOS is a registered trademark of Microsoft Corporation.

### ADF PRIMITIVES SUPPORTED

The following ADF primitives are supported by this device:

INP	RONF
CONF	RORF
COIF	NORF

### ORDERING INFORMATION

t <sub>PD</sub> (ns)	t <sub>CO</sub> (ns)	f <sub>MAX</sub> (MHz)	Order Code	Package	Operating Range
25	15	50	D5C032-25	CERDIP	Commercial
			P5C032-25	PDIP	
30	17	43.5	D5C032-30	CERDIP	Commercial
			P5C032-30	PDIP	
35	20	40	D5C032-35	CERDIP	Commercial
			P5C032-35	PDIP	

**ABSOLUTE MAXIMUM RATINGS\***

Symbol	Parameter	Min	Max	Unit
$V_{CC}$	Supply Voltage(1)	-2.0	7.0	V
$V_{PP}$	Programming Supply Voltage(1)	-2.0	13.5	V
$V_I$	DC Input Voltage(1)(2)	-0.5	$V_{CC} + 0.5$	V
$t_{stg}$	Storage Temperature	-65	+150	°C
$t_{amb}$	Ambient Temperature(4)	-10	+85	°C

\*Notice: Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

**NOTES:**

1. Voltages with respect to ground.
2. Minimum DC input is -0.5V. During transitions, the inputs may undershoot to -2.0V or overshoot to 7.0V for periods less than 20 ns under no load conditions.
3. Under bias, Extended temperature versions are also available.
4. Extended temperature versions also available.

**RECOMMENDED OPERATING CONDITIONS**

Symbol	Parameter	Min	Max	Unit
$V_{CC}$	Supply Voltage	4.75	5.25	V
$V_{IN}$	Input Voltage	0	$V_{CC}$	V
$V_O$	Output Voltage	0	$V_{CC}$	V
$T_A$	Operating Temperature	0	+70	°C
$t_R$	Input Rise Time		500	ns
$t_F$	Input Fall Time		500	ns

**D.C. CHARACTERISTICS**  $T_A = 0^\circ\text{C to }70^\circ\text{C}$ ,  $V_{CC} = 5V \pm 5\%$ 

Symbol	Parameter/Test Conditions	Min	Typ	Max	Unit
$V_{IH}^{(5)}$	High Level Input Voltage	2.0		$V_{CC} + 0.3$	V
$V_{IL}^{(5)}$	Low Level Input Voltage	-0.3		0.8	V
$V_{OH}^{(6)}$	High Level Output Voltage $I_O = -4.0 \text{ mA D.C.}, V_{CC} = \text{min.}$	2.4			V
$V_{OL}$	Low Level Output Voltage $I_O = 4.0 \text{ mA D.C.}, V_{CC} = \text{min.}$			0.45	V
$I_I$	Input Leakage Current $V_{CC} = \text{max.}, \text{GND} < V_{IN} < V_{CC}$			$\pm 10$	$\mu\text{A}$
$I_{OZ}$	Output Leakage Current $V_{CC} = \text{max.}, \text{GND} < V_{OUT} < V_{CC}$			$\pm 10$	$\mu\text{A}$

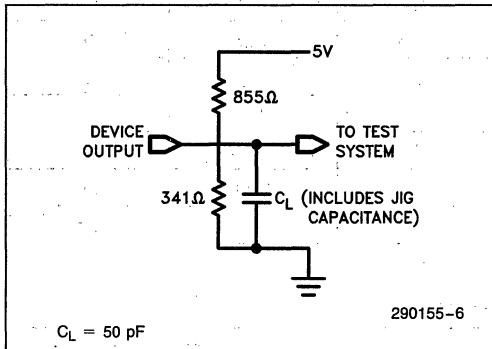
**D.C. CHARACTERISTICS**  $T_A = 0^\circ\text{C}$  to  $70^\circ\text{C}$ ,  $V_{CC} = 5V \pm 5\%$  (Continued)

Symbol	Parameter/Test Conditions	Min	Typ	Max	Unit
$I_{SC}^{(7)}$	Output Short Circuit Current $V_{CC} = \text{max.}, V_{OUT} = 0.5V$			10	mA
$I_{SB}^{(8)}$	Standby Current $V_{CC} = \text{max.}, V_{IN} = V_{CC}$ or GND, Standby Mode		10	100	$\mu\text{A}$
$I_{CC}^{(9)}$	Power Supply Current $V_{CC} = \text{max.}, V_{IN} = V_{CC}$ or GND, No Load, Input Freq. = 10 MHz Active Mode (Turbo = Off), Device Prog. as 8-bit Ctr.		15	25	mA

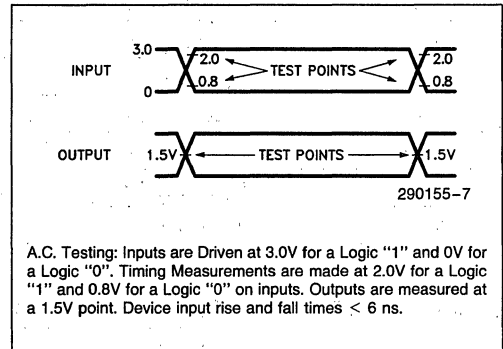
**NOTES:**

5. Absolute values with respect to device GND; all over- and undershoots due to system or tester noise are included.
6.  $I_O$  at CMOS levels (3.84V) = -2 mA.
7. Not more than 1 output should be tested at a time. Duration of that test must not exceed 1 second.
8. With Turbo Bit = Off, device automatically enters standby mode approximately 100 ns after last input transition.
9. Maximum Active Current at operational frequency is less than 40 mA.

**A.C. TESTING LOAD CIRCUIT**



**A.C. TESTING INPUT, OUTPUT WAVEFORM**



**CAPACITANCE**

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
$C_{IN}$	Input Capacitance	$V_{IN} = 0V, f = 1.0 \text{ MHz}$			10	pF
$C_{OUT}$	Output Capacitance	$V_{OUT} = 0V, f = 1.0 \text{ MHz}$			10	pF
$C_{CLK}$	Clock Pin Capacitance	$V_{IN} = 0V, f = 1.0 \text{ MHz}$			10	pF
$C_{VPP}$	$V_{PP}$ Pin	Pin 11			20	pF

**A.C. CHARACTERISTICS**  $T_A = 0^\circ\text{C to } +70^\circ\text{C}$ ,  $V_{CC} = 5V \pm 5\%$ , Turbo Bit On<sup>(10)</sup>

Symbol	From	To	5C032-25			5C032-30			5C032-35			Non- <sup>(8)</sup> Turbo Mode	Unit
			Min	Typ	Max	Min	Typ	Max	Min	Typ	Max		
$t_{PD}$	I or I/O	Comb. Output			25			30			35	+ 15	ns
$t_{PZX}^{(11)}$	I or I/O	Output Enable			25			30			35	+ 15	ns
$t_{PXZ}^{(11)}$	I or I/O	Output Disable			25			30			35	+ 15	ns

**NOTES:**

 10. Typ. values are at  $T_A = 25^\circ\text{C}$ ,  $V_{CC} = 5V$ , Active Mode.

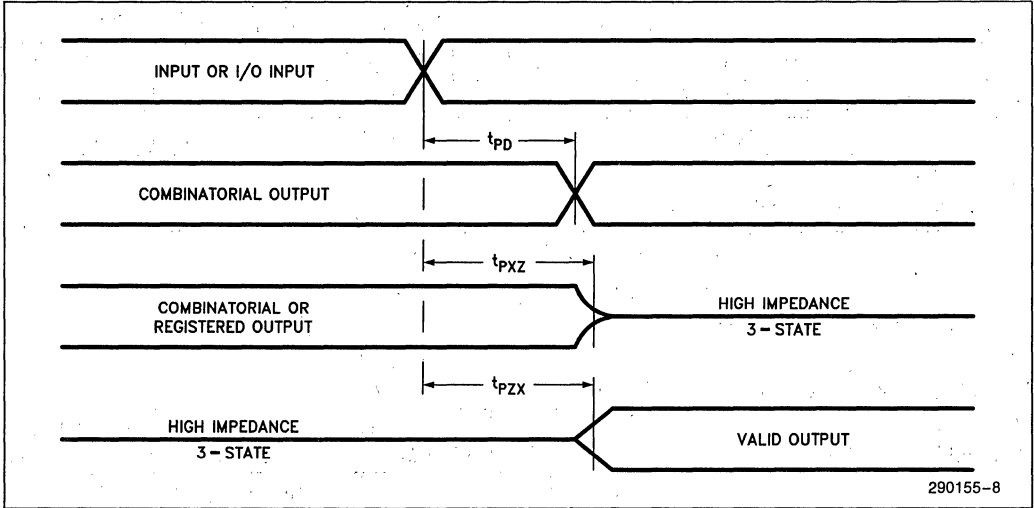
 11.  $t_{PZX}$  and  $t_{PXZ}$  are measured at  $\pm 0.5V$  from steady state voltage as driven by spec. output load.  $t_{PXZ}$  is measured with  $C_L = 5\text{ pF}$ .

**A.C. CHARACTERISTICS**  $T_A = 0^\circ\text{C to } 70^\circ\text{C}$ ,  $V_{CC} = 5V \pm 5\%$ , Turbo Bit On<sup>(10)</sup>
**SYNCHRONOUS CLOCK MODE**

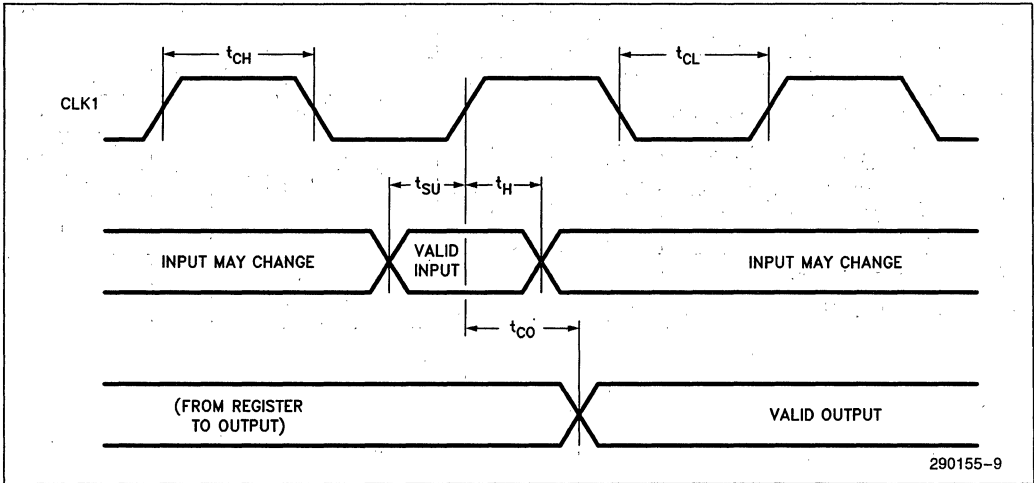
Symbol	Parameter	5C032-25			5C032-30 EP320-1			5C032-35 EP320-2			Non- <sup>(8)</sup> Turbo Mode	Unit
		Min	Typ	Max	Min	Typ	Max	Min	Typ	Max		
$f_{MAX}$	Max. Frequency (Pipelined) $1/t_{SU}$ — No Feedback			50			43.5			40		MHz
$f_{CNT}$	Max. Count Frequency $1/t_{CNT}$ — with Feedback			33.3			28.5			25		MHz
$t_{SU}$	Input Setup Time to CLK	20			23			25			+ 15	ns
$t_H$	I or I/O Hold after CLK High	0			0			0				ns
$t_{CO}$	CLK High to Output Valid			15			17			20		ns
$t_{CNT}$	Register Output Feedback to Register Input — Internal Path	30			35			40			+ 15	ns
$t_{CH}$	CLK High Time	10			11			12				ns
$t_{CL}$	CLK Low Time	10			11			12				ns

SWITCHING WAVEFORMS

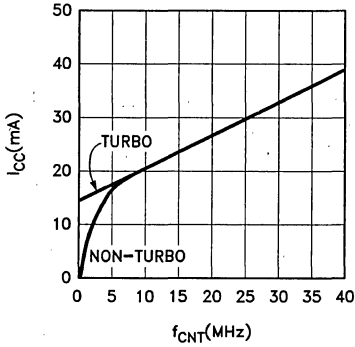
COMBINATORIAL MODE



SYNCHRONOUS CLOCK MODE



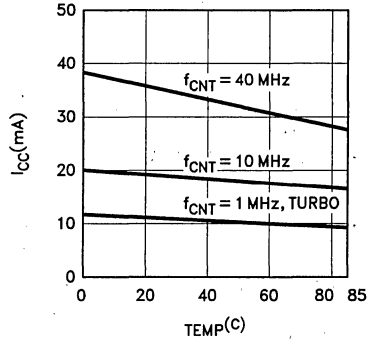
**Current in Relation to Frequency**



Conditions:  $T_A = 0^\circ\text{C}$ ,  $V_{CC} = 5.25\text{V}$

290155-11

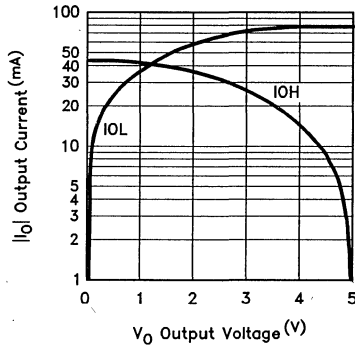
**Current in Relation to Temperature**



Conditions:  $V_{CC} = 5.25\text{V}$

290155-12

**Output Drive Current in Relation to Voltage**



Conditions:  $T_A = +25^\circ\text{C}$ ,  $V_{CC} = 5\text{V}$

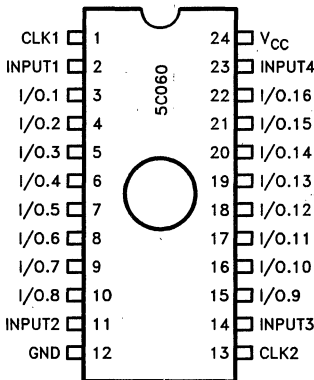
290155-13



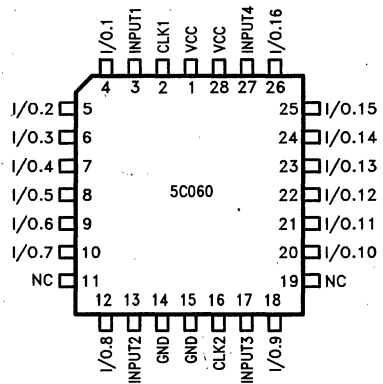


# 5C060 600-GATE CHMOS H-SERIES ERASABLE PROGRAMMABLE LOGIC DEVICE (H-EPLD)

- High Performance LSI Semi-Custom Logic Alternative to Low-End Gate Arrays, TTL, and 74HC SSI and MSI Logic
  - CHMOS EPROM Technology Based. UV Erasable
  - Programmable Clock System with Two Synchronous Clocks as Well as Asynchronous Clocking Option on all Registers
  - Programmable Output Registers. Can be Configured as D, T, SR, or JK Types
  - Programmable Security Bit Allows Total Protection of Proprietary Designs
  - 16 Macrocells with Programmable I/O Architecture; up to 20 Inputs (4 Dedicated, 16 I/O) or 16 Outputs
  - High Speed  $t_{PD}$  (max) 45 ns, 16.67 MHz Performance
  - Low Power; 50  $\mu$ A Typical Standby Current
  - Erasable Array for 100% Generic Testability
  - Small Footprint 24-Pin 0.3" DIP and 28 Pin J-Leaded Chip Carrier Package
- (See Packaging Spec. Order # 231369)
- 100% Compatible with EP600



290194-1



290194-2

**Figure 1. 5C060 Pin Configurations**

The Intel 5C060 H-EPLD (H-series Programmable Logic Device) is capable of implementing over 600 equivalent gates of user-customized logic functions through programming. The device can be used to replace low-end gate arrays, multiple programmable logic arrays and LS TTL and 74HC (CMOS) SSI and MSI logic devices. The 5C060 can also be used as a direct, low-power replacement for most, common 24-pin fuse-based programmable logic devices. With its revolutionary programmable I/O architecture, the device has advanced functional capabilities beyond that of typical programmable logic.

The 5C060 H-EPLD uses CHMOS EPROM (floating gate) cells as logic control elements instead of fuses. The CHMOS EPROM technology reduces power consumption of H-EPLDs to less than 20% of a comparable bipolar device without sacrificing speed performance. In addition, Intel's advanced CHMOS I-E EPROM process technology enables greater

logic densities to be achieved with superior speed and low-power performance over other comparable devices. Intel's H-ELPDs add the benefits of "zero" stand-by power not available on other programmable logic devices. EPROM technology allows these devices to be 100% factory tested by programming and erasing all the EPROM logic control elements.

The erasability of EPLDs introduces the designer to a new concept in hardware design called Modular EPLD Logic Design (MELD). Just as modular software design speeds development time and reduces errors by isolating them to a specific module, the MELD philosophy aids in hardware design. A designer can develop his modular design on the Intel Programmable Logic Development System II (iPLDS II) and test individual modules for functionality. If one of the modules has a design flaw, the designer merely erases the part and starts anew (since the 5C060 is

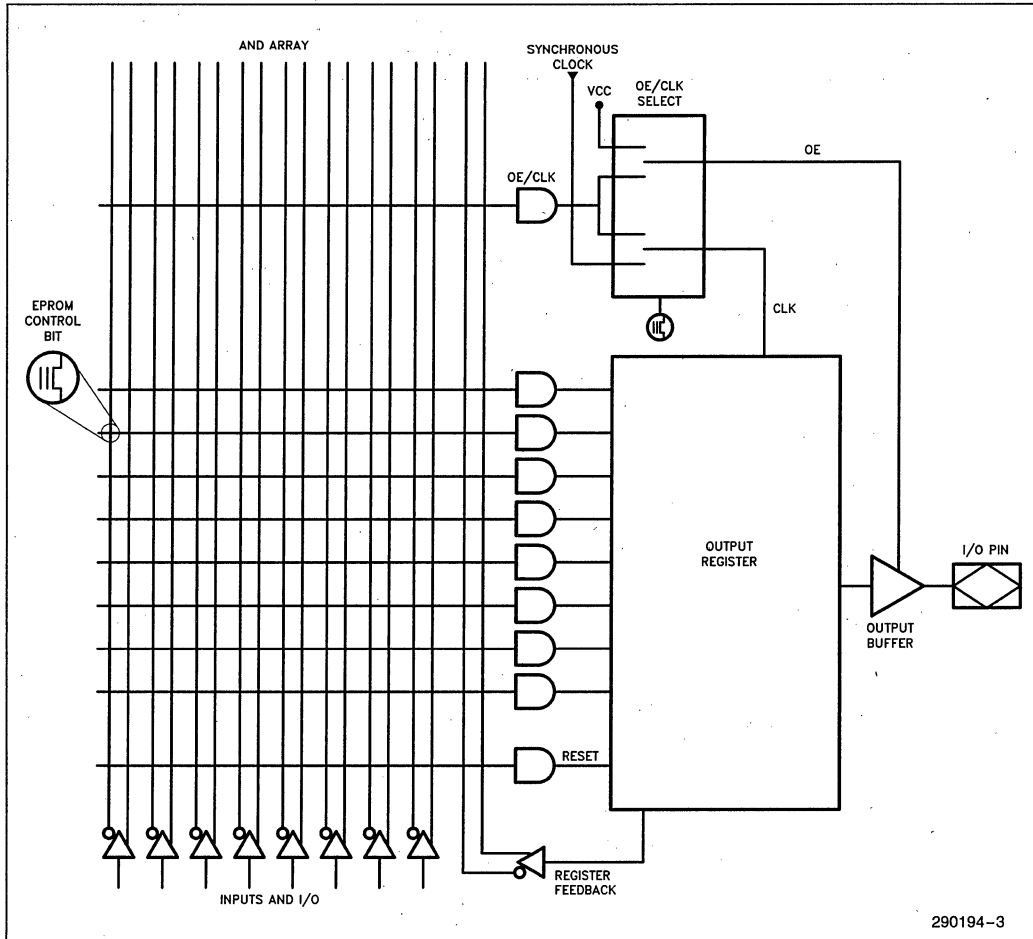


Figure 2. Basic Macrocell Architecture of the 5C060

EPROM-based, there is no waste associated with modular design as there would be in fuse-based PLDs).

The architecture of the 5C060 is based on the "Sum of Products" PLA (Programmable Logic Array) structure with a programmable AND array feeding into a fixed OR array. The device accommodates combinational and sequential logic functions. A proprietary programmable I/O architecture provides individual selection of either combinatorial or registered output and feedback signals all with selectable polarity.

A feature unique to the 5C060 is the ability to individually program the output registers as a D-, T-, SR-, or JK-type Flip-Flop without sacrificing the utilization of programmable AND logic. Additionally, each output register can be individually clocked from any of the input or feedback paths available within the AND array. With these features, a wide variety of logic functions can be simultaneously implemented—all on the same device.

## ARCHITECTURE DESCRIPTION

Externally, the 5C060 has 4 dedicated data input pins, 16 I/O pins which may be configured for input, output, or bidirectional operations, and 2 synchronous clock inputs. The 5C060 is contained in a 24-pin windowed package (0.3 inch wide) or 28-lead J-leaded chip carrier package, and contains 16 programmable registers.

The basic Macrocell architecture for the 5C060 is shown in Figure 2. The 5C060 has 16 of these Macrocells (one for each I/O pin). The Macrocell is organized in the familiar sum-of-products structure with a programmable AND array attached to a fixed OR term. The inputs to the programmable AND array originate from the true and complement signals from each of the dedicated input pins and each of the I/O control blocks. The 40-input AND array of the 5C060 feeds 160 AND gates (product terms) which are distributed among the 16 available Macrocells within that device. The global device architecture is shown in Figure 3.

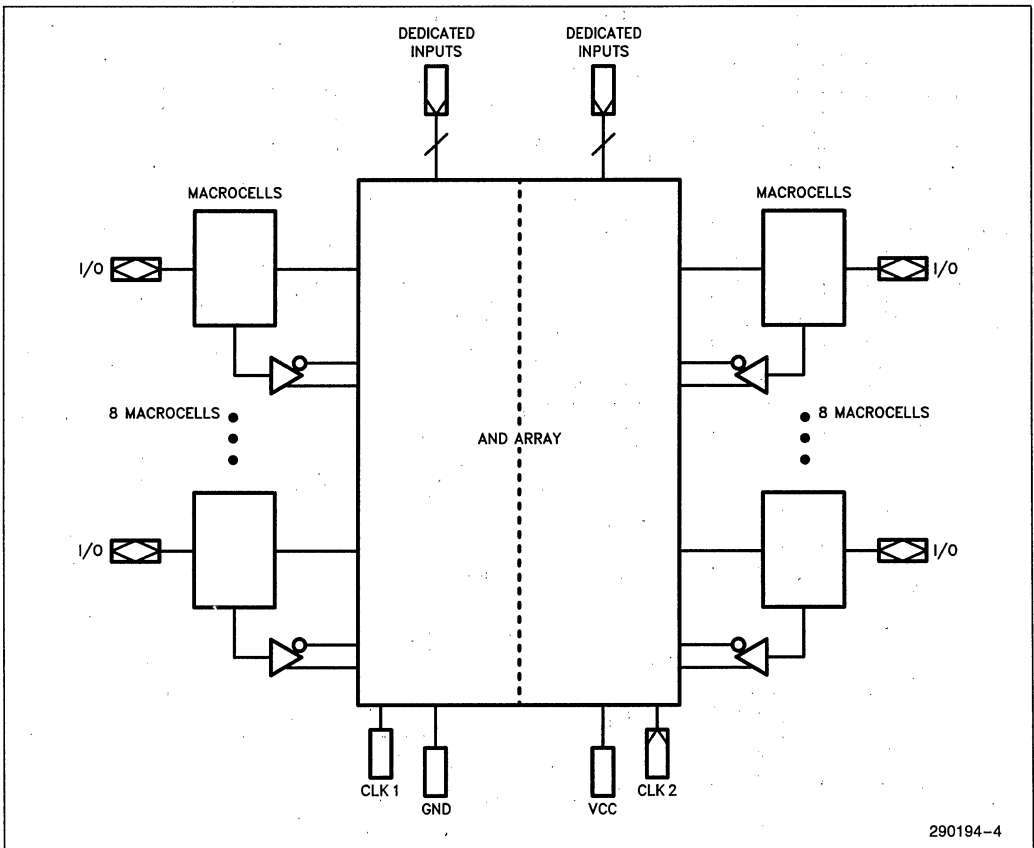


Figure 3. 5C060 Global Architecture

The Macrocells contain ten product terms total. Eight of the ten product terms (AND gates) are dedicated for logic implementation. One product term on each Macrocell is used for RESET control to the output register associated with the Macrocell. The final product term is used for OUTPUT ENABLE/Asynchronous Clock implementation.

Within the AND array, there is an EPROM connection at every intersection of an input signal (true and complement) and a product term to a given Macrocell. Before programming an erased device, every EPROM connection is made at every intersection. But during the programming process, these connections are opened so that only the desired connections remain. Therefore, the true or complement of any input signal can be connected to any product term. If both the true and complement connections of any signal are left intact, a logical false results on the output of the AND gate. However, if both the true and complement connections are open, then a logic "don't care" results on the AND gate. Lastly, if all the inputs of a product term are programmed open, then a logical true results on the output of the AND gate.

The 5C060 has two dedicated clock inputs to provide synchronous clock signals to the internal registers. Each of the clock signals controls half the total registers within the given device. For example, CLK1 provides synchronous clocking to the registers in Macrocells in the left half of the array while CLK2 controls the registers associated with Macrocells in the right half of the array. The advanced I/O architecture allows for any number of the registers to be synchronously clocked (from none to all). Both of the dedicated clock inputs latch the data into a given register when triggered on a positive edge.

## MACROCELL ARCHITECTURE SELECTION

The 5C060 architecture provides each Macrocell with over 50 different possible I/O register configurations. Each I/O pin can be configured for combinatorial or registered output (true or complement) with feedback. In addition, four different types of output registers can be implemented into every I/O pin without any additional logic requirements. The feedback mechanism for each register back into the AND array can be programmed to provide for either registered feedback from the Macrocell or input feedback (treating the pin as an input). Another advantage of the advanced I/O capability of the 5C060 is the ability to individually clock each internal register from asynchronous clock signals.

## Output Enable (OE)/Clock Selection

Two modes of operation are provided by the OE/CLK Select Multiplexer as a part of each Macrocell. One mode provides for three-state buffering of outputs while in the other mode, the outputs are always enabled. The operation of the OE/CLK Select Multiplexer sets the mode within a given Macrocell. Therefore, the output mode can be selected individually on every output. Figure 4 illustrates the two modes of OE/CLK operation.

### MODE 0: THREE-STATE BUFFERING

In Mode 0, the three-state output buffer is controlled by a single product term originating from the AND array. The output is enabled when the product term is a logical true. Conversely, the output appears as high impedance when the product term is a logical false as shown in Table 1. In Mode 0, the Macrocell Flip-Flop is connected to its associated synchronous clock (either CLK1 or CLK2 depending upon the Macrocell's location within the device). Thus, the Macrocell Flip-Flop may be clocked by its respective synchronous clock but its output will not become valid until the output is enabled.

**Table 1. Mode 0 Output Selection**

Product Term	Output Buffer
FALSE	Three-State
TRUE	Enabled

### MODE 1: OUTPUT BUFFER ENABLED

In Mode 1, the Output Buffer is always enabled. In addition, the Macrocell Flip-Flop is connected to the AND array. The Macrocell Flip-Flop may now be triggered from an asynchronous clock signal generated by the AND array logic to the OE/CLK multiplexable term. Mode 1 allows the Macrocell Flip-Flops to be individually clocked from any of the available signals in the AND array. Since both true and complement values appear in the AND array, the Flip-Flop may be configured to trigger on positive or negative clock edges. Gated clock structures can be created since the Flip-Flop clock is created by a product term.

## Invert Select EPROM Bit

The Invert Select EPROM bit is used to invert the product term input into the register. This applies to all inputs including double inputs on the JK and SR registers.

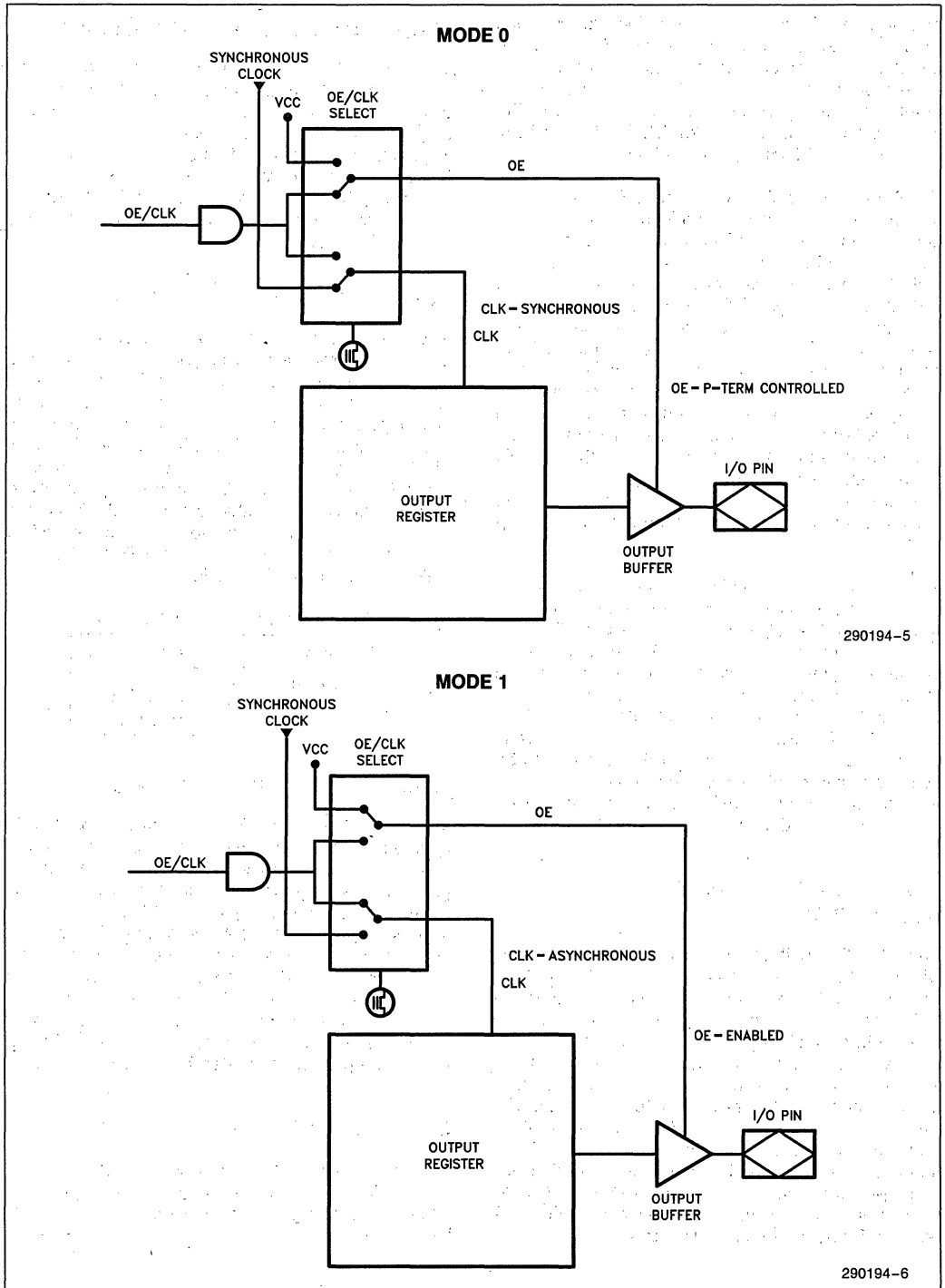


Figure 4. Output Enable/Clock Configuration

## REGISTER SELECTION

The advanced I/O architecture of the 5C060 allows four different register types along with combinatorial output as illustrated in Figure 5a. The register types include a T, D, JK, or SR Flip-Flop and each Macrocell I/O structure may be independently configured. In addition, all registers have an individual asynchronous RESET control from a dedicated product term derived in the AND array. When this dedicated product term is a logical one, the Macrocell register is immediately cleared to a logical zero independent of the register clock. The RESET function occurs automatically on power-up.

### Output Register Configuration

The four different register types shown in Figure 5b-5e are described below.

#### D- or T-type Flip-Flops

When either a D- or T-type Flip-Flop is configured as part of the I/O structure, all eight of the product terms into the Macrocell are ORed together and fed into the register input.

#### JK or SR Registers

When either a JK or SR register is configured, the eight product terms are shared among two OR gates (one for the J or S input and the other for the K or R input). The allocation for these product terms for each of the register inputs is optimized by the iPLDS II development software.

## OUTPUT/FEEDBACK

The Output Select Multiplexer allows for either registered, combinatorial or no output.

The Feedback Select Multiplexer EPROM bit enables registered, I/O (using the pin for bidirectional input or just input), or no feedback to the AND array.

The Feedback Select is also important for building product terms with more than 8 products. The 8-product product term of a Macrocell can be fed back into the AND array and combined with still more signals to create a much larger product term (of more than 8-inputs). In addition, if the feedback product term is not to be output, then the iPLDS II will reserve the associated Macrocell pin and indicate it in the REPORT file. A reserved pin should be left floating (no connect) when assembled onto a circuit board.

Any I/O pin may be configured as a dedicated input by selecting no output and pin feedback through the appropriate multiplexers.

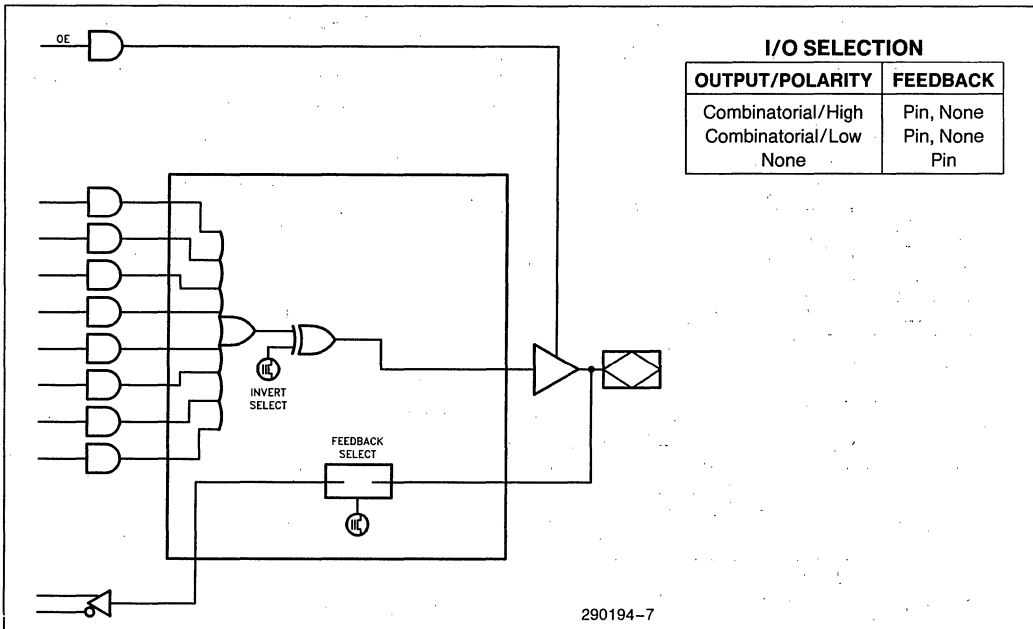


Figure 5a. Combinatorial I/O Configuration

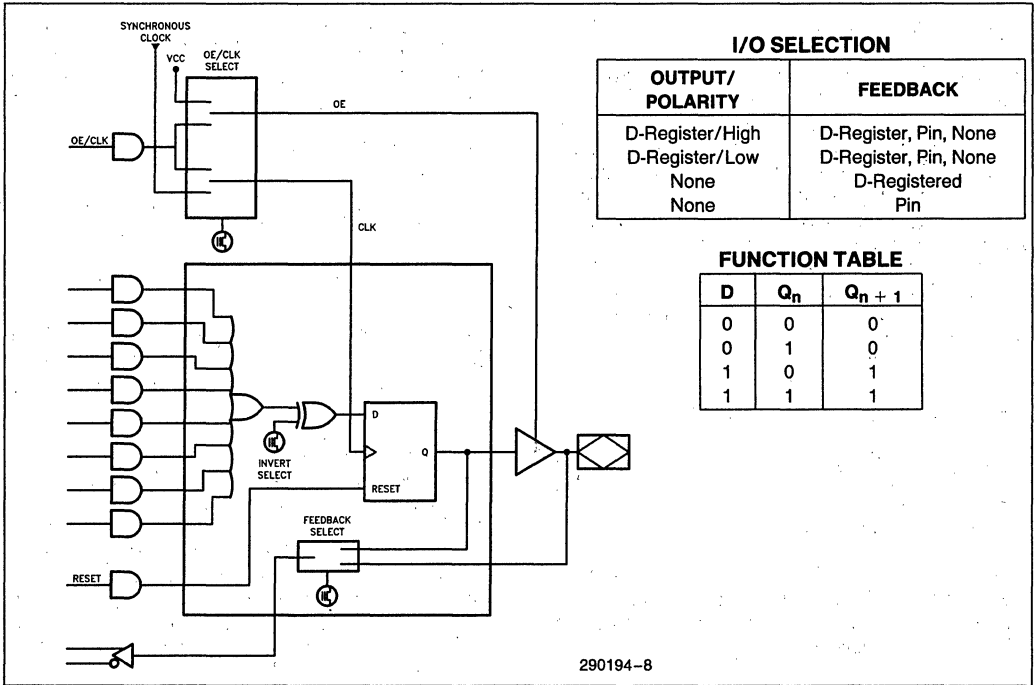


Figure 5b. D-Type Flip-Flop Register Configuration

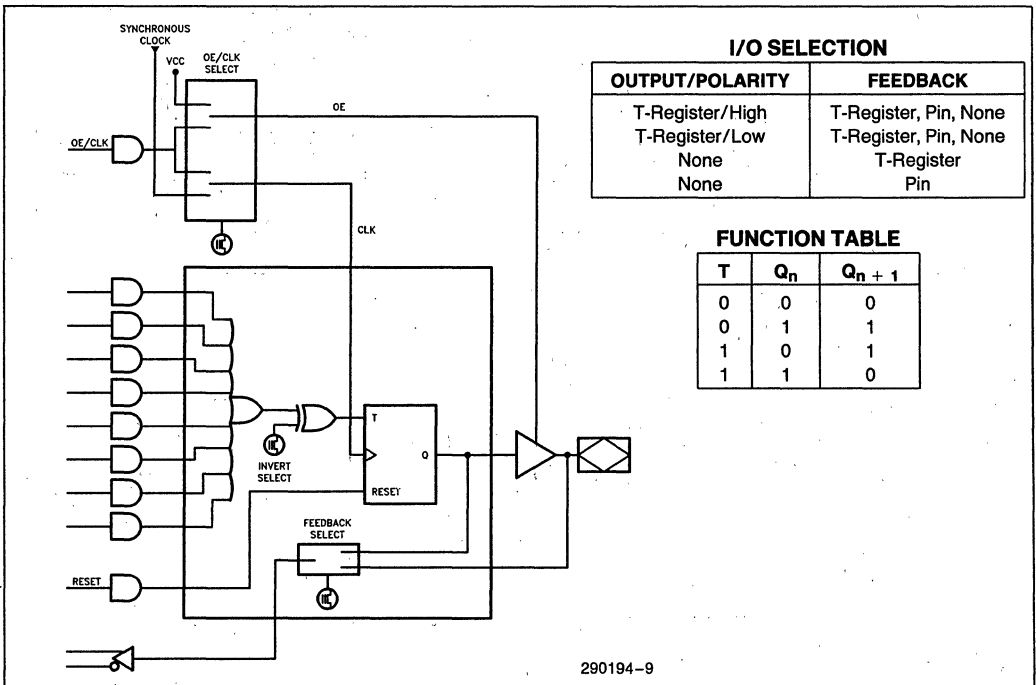


Figure 5c. Toggle Flip-Flop Register Configuration

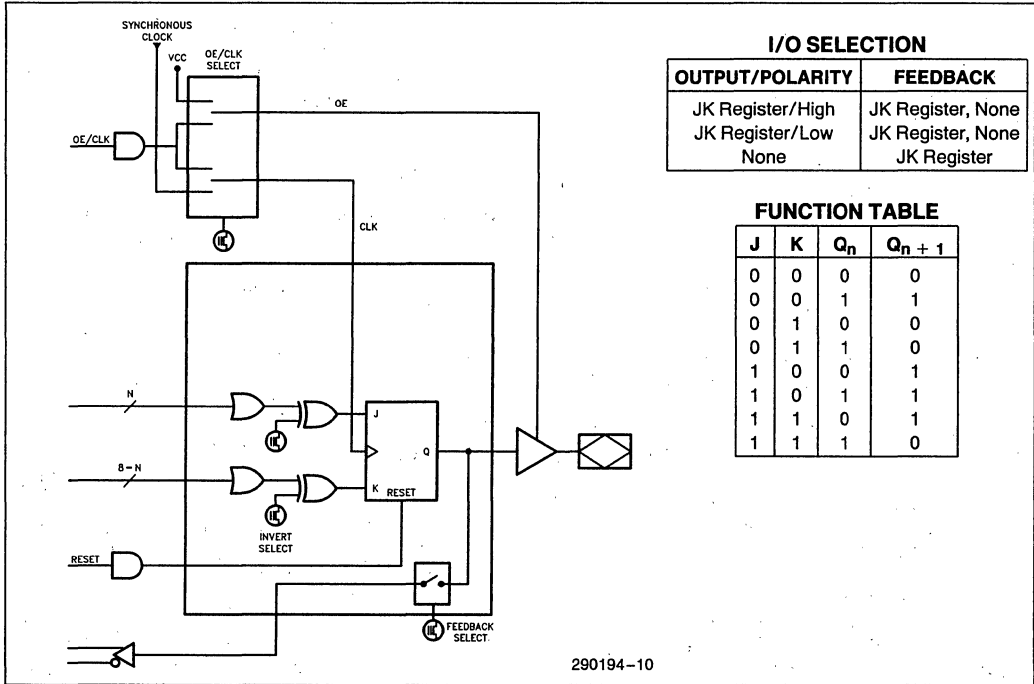


Figure 5d. JK Flip-Flop Register Configuration

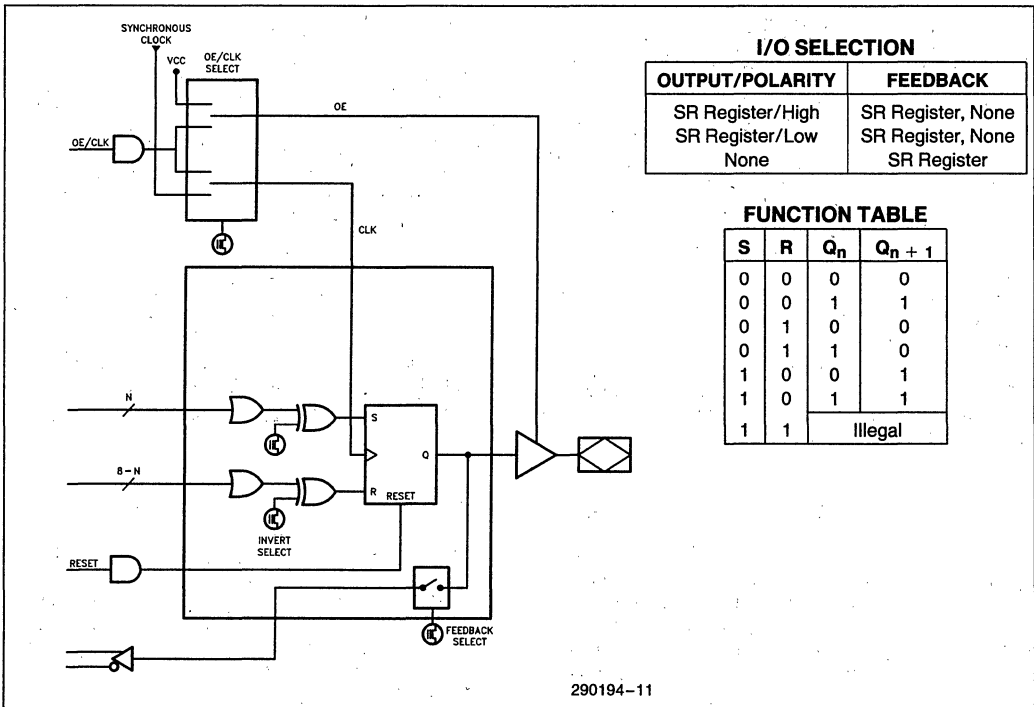


Figure 5e. SR Flip-Flop Register Configuration



## Erased-State Configuration

Prior to programming or after erasing, the I/O structure is configured for combinatorial active low output with input (pin) feedback.

## ERASURE CHARACTERISTICS

Erasure characteristics of the device are such that erasure begins to occur upon exposure to light with wavelengths shorter than approximately 4000Å. It should be noted that sunlight and certain types of fluorescent lamps have wavelengths in the 3000Å–4000Å. Data shows that constant exposure to room level fluorescent lighting could erase the typical device in approximately three years, while it would take approximately one week to cause erasure when exposed to direct sunlight. If the 5C060 is to be exposed to these types of lighting conditions for extended periods of time, conductive opaque labels should be placed over the device window to prevent unintentional erasure.

The recommended erasure procedure for the 5C060 is exposure to shortwave ultraviolet light with a wavelength of 2537Å. The integrated dose (i.e., UV intensity  $\times$  exposure time) for erasure should be a minimum of fifteen (15) Wsec/cm<sup>2</sup>. The erasure time with this dosage is approximately 15 to 20 minutes using an ultraviolet lamp with a 12,000  $\mu$ W/cm<sup>2</sup> power rating. The 5C060 should be placed within one inch of the lamp tubes during erasure. The maximum integrated dose the 5C060 can be exposed to without damage is 7258 Wsec/cm<sup>2</sup> (1 week at 12,000  $\mu$ W/cm<sup>2</sup>). Exposure to high intensity UV light for longer periods may cause permanent damage to the device.

## PROGRAMMING CHARACTERISTICS

Initially, and after erasure, all the EPROM control bits of the 5C060 are connected (in the "1" state). Each of the connected control bits are selectively disconnected by programming the EPROM cells into their "0" state. Programming voltage and waveform specifications are available by request from Intel to support programming of the 5C060.

## intelligent Programming™ Algorithm

The 5C060 supports the intelligent Programming Algorithm which rapidly programs Intel H-ELPDs (and EPROMs) using an efficient and reliable method. The intelligent Programming Algorithm is particularly suited to the production programming environment. This method greatly decreases the overall program-

ming time while programming reliability is ensured as the incremental program margin of each bit is continually monitored to determine when the bit has been successfully programmed.

## FUNCTIONAL TESTING

Since the logical operation of the 5C060 is controlled by EPROM elements, the device is completely testable. Each programmable EPROM bit controlling the internal logic is tested using application-independent test program patterns. After testing, the devices are erased before shipment to customers. No post-programming tests of the EPROM array are required.

The testability and reliability of EPROM-based programmable logic devices is an important feature over similar devices based on fuse technology. Fuse-based programmable logic devices require a user to perform post-programming tests to insure proper programming. These tests must be done at the device level because of the cumulative error effect. For example, a board containing ten devices each possessing a 2% device fallout translates into an 18% fallout at the board level (it should be noted that programming fallout of fuse-based programmable logic devices is typically 2% or higher).

## DESIGN RECOMMENDATIONS

For proper operation, it is recommended that all input and output pins be constrained to the voltage range  $GND < (V_{IN} \text{ or } V_{OUT}) < V_{CC}$ . Unused inputs should be tied to an appropriate logic level (e.g. either  $V_{CC}$  or  $GND$ ) to minimize device power consumption. Reserved pins (as indicated in the logic compiler REPORT file) should be left floating (no connect) so that the pin can attain the appropriate logic level. A power supply decoupling capacitor of at least 0.2  $\mu$ F must be connected directly between  $V_{CC}$  and  $GND$  pins of the device.

As with all CMOS devices, ESD handling procedures should be used with the 5C060 to prevent damage to the device during programming, assembly, and test.

## DESIGN SECURITY

A single EPROM bit provides a programmable design security feature that controls the access to the data programmed into the device. If this bit is set, a proprietary design within the device cannot be copied. This EPROM security bit enables a higher degree of design security than fused-based devices

since programmed data within EPROM cells is invisible even to microscopic evaluation. The EPROM security bit, along with all the other EPROM control bits, will be reset by erasing the device.

**AUTOMATIC STAND-BY MODE**

The 5C060 contains a programmable bit, the Turbo Bit, that optimizes operation for speed or for power savings. When the Turbo Bit is programmed (TURBO = ON), the device is optimized for maximum speed. When the Turbo Bit is not programmed (TURBO = OFF), the device is optimized for power savings by entering standby mode during periods of inactivity.

Figure 6 shows the device entering standby mode approximately 100 ns after the last input transition. When the next input transition is detected, the device returns to active mode. Wakeup time adds an additional 25 ns to the propagation delay through the device as measured from the first input. No delay will occur if an output is dependent on more than one input and the last of the inputs changes after the device has returned to active mode.

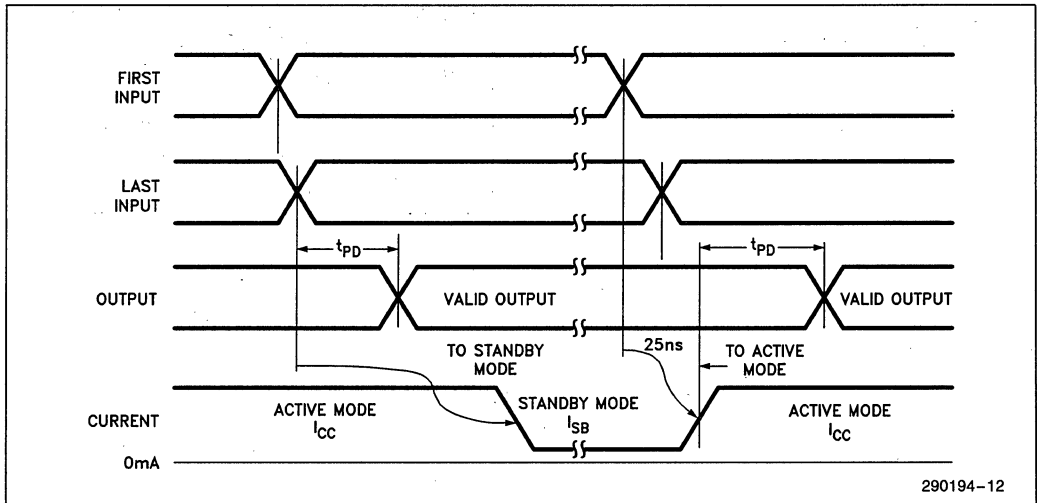
After erasure, the Turbo Bit is unprogrammed (OFF); automatic standby mode is enabled. When the Turbo Bit is programmed (ON), the device never enters standby mode.

**LATCH-UP IMMUNITY**

All of the input, I/O, and clock pins of the 5C060 have been designed to resist latch-up which is inherent in inferior CMOS structures. The 5C060 is designed with Intel's proprietary CHMOS II-E EPROM process. Thus, each of the pins will not experience latch-up with currents up to 100 mA and voltages ranging from -1V to V<sub>CC</sub> + 1V. Furthermore, the programming pin is designed to resist latch-up to the 13.5V maximum device limit.

**INTEL PROGRAMMABLE LOGIC DEVELOPMENT SYSTEM II (IPLDS II)**

iPLDS II provides all the tools needed to design with Intel H-Series EPLDs or compatible devices. In addition to providing development assistance, iPLDS II insulates the user from having to know all the intricate details of EPLD architecture (the machine will optimize a design to benefit from architectural features). It contains comprehensive third generation software that supports four different design entry methods, minimizes logic, does automatic pin assignments and produces the best design fit for the selected EPLD. It is user friendly with guided menus, on-line Help messages and soft key inputs.



**Figure 6. 5C060 Standby and Active Mode Transitions**

In addition, the iPLDS II contains programmer hardware in the form of an iUP-PC Universal Programmer Personal Computer to enable the user to program EPLDs, read and verify programmed devices and also to graphically edit programming files. The software generates industry standard JEDEC object code output files which can be downloaded to other programmers as well.

The iPLDS II has interfaces to popular schematic capture packages to enable designs to be entered using schematics. An integrated schematic entry method is provided by SCHEMA II-PLD, a low-cost schematic capture package that supports EPLD primitives and user-defined macro symbols. SCHEMA II-PLD contains the EPLD Design Manager, which provides a single user interface to both SCHEMA II-PLD and iPLS II software. The other design formats supported are Boolean equation entry and State Machine design entry.

The iPLDS II operates on the IBM† PC/XT, PC/AT, or other compatible machine with the following configuration:

1. At least one floppy disk drive and hard disk drive.
2. MS-DOS†† Operating System Version 3.0 or greater.
3. 512K Memory (640K recommended).
4. Intel iUP-PC Universal Programmer Personal Computer and GUPI Adaptor (supplied with iPLDS II).
5. A color monitor is suggested.

Detailed information on the Intel Programmable Logic Development System II is contained in a separate Intel data sheet. (Order Number: 280168)

†IBM Personal Computer is a registered trademark of International Business Machines Corporation.

††MS-DOS is a registered trademark of Microsoft Corporation.

### ADF PRIMITIVES SUPPORTED

The following ADF primitives are supported by this device:

INP	JOJF
CONF	JONF
COIF	SONF
RONF	SOSF
RORF	TOIF
ROIF	TONF
NORF	TOTF
NOJF	CLKB
NOSF	
NOTF	

### ORDERING INFORMATION

t <sub>pd</sub> (ns)	t <sub>co</sub> (ns)	f <sub>MAX</sub> (MHz)	Order Code	Package	Operating Range
45	22	26	D5C060-45	CERDIP	Commercial
			P5C060-45	PDIP	
			N5C060-45	PLCC	
55	25	23	D5C060-55	CERDIP	Commercial
			P5C060-55	PDIP	
			N5C090-55	PLCC	

**ABSOLUTE MAXIMUM RATINGS\***

Symbol	Parameter	Min	Max	Units
V <sub>CC</sub>	Supply Voltage <sup>(1)</sup>	-2.0	7.0	V
V <sub>PP</sub>	Programming Supply Voltage <sup>(1)</sup>	-2.0	13.5	V
V <sub>I</sub>	DC Input Voltage <sup>(1)(2)</sup>	-0.5	V <sub>CC</sub> + 0.5	V
t <sub>stg</sub>	Storage Temperature	-65	+150	°C
t <sub>amb</sub>	Ambient Temperature <sup>(3)</sup>	-10	+85	°C

\*Notice: Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

**NOTES:**

1. Voltages with respect to ground.
2. Minimum DC input is -0.5V. During transitions, the inputs may undershoot to -2.0V or overshoot to 7.0V for periods less than 20 ns under no load conditions.
3. Under bias. Extended temperature versions are also available.

**RECOMMENDED OPERATING CONDITIONS**

Symbol	Parameter	Min	Max	Unit
V <sub>CC</sub>	Supply Voltage	4.75	5.25	V
V <sub>IN</sub>	Input Voltage	0	V <sub>CC</sub>	V
V <sub>O</sub>	Output Voltage	0	V <sub>CC</sub>	V
T <sub>A</sub>	Operating Temperature	0	+70	°C
t <sub>R</sub> <sup>(4)</sup>	Input Rise Time		500	ns
t <sub>F</sub> <sup>(4)</sup>	Input Fall Time		500	ns

**NOTE:**

4. t<sub>R</sub>, t<sub>F</sub> for CLK is 250 ns max.

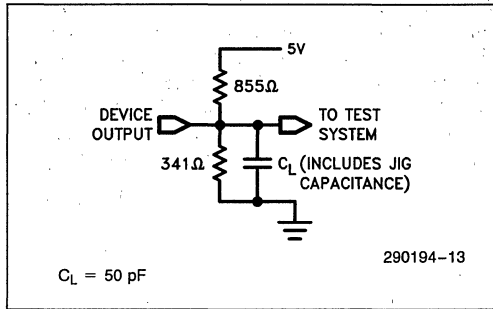
**D.C. CHARACTERISTICS** T<sub>A</sub> = 0°C to 70°C, V<sub>CC</sub> = 5.0V ± 5%

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
V <sub>IH</sub> <sup>(5)</sup>	HIGH Level Input Voltage		2.0		V <sub>CC</sub> + 0.3	V
V <sub>IL</sub> <sup>(5)</sup>	LOW Level Input Voltage		-0.3		0.8	V
V <sub>OH</sub> <sup>(6)</sup>	HIGH Level Output Voltage	I <sub>O</sub> = -4.0 mA DC, V <sub>CC</sub> = Min.	2.4			V
V <sub>OL</sub>	LOW Level Output Voltage	I <sub>O</sub> = 4.0 mA DC, V <sub>CC</sub> = Min.			0.45	V
I <sub>I</sub>	Input Leakage Current	V <sub>CC</sub> = Max., GND < V <sub>IN</sub> < V <sub>CC</sub>			±10.0	μA
I <sub>OZ</sub>	Output Leakage Current	V <sub>CC</sub> = Max., GND < V <sub>OUT</sub> < V <sub>CC</sub>			±10.0	μA
I <sub>SC</sub> <sup>(7)</sup>	Output Short Circuit Current	V <sub>CC</sub> = Max., V <sub>OUT</sub> = 0.5V		20	30	mA
I <sub>SB</sub> <sup>(8)</sup> 5C060	Standby Current (Standby)	V <sub>CC</sub> = Max., V <sub>IN</sub> = V <sub>CC</sub> or GND		50	100	μA
I <sub>CC</sub> 5C060	Power Supply Current (Active) (Turbo Bit Off) Device Prog. as 16-Bit Ctr.	V <sub>CC</sub> = Max., V <sub>IN</sub> = V <sub>CC</sub> or GND No Load, Input Freq. = 1 MHz		10	15	mA

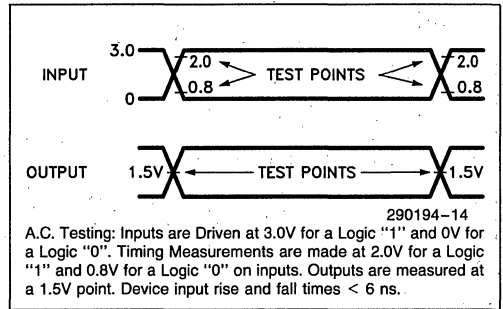
**NOTES:**

5. Absolute values with respect to device GND; all over and undershoots due to system or tester noise are included.
6. I<sub>O</sub> at CMOS levels (3.84V) = -2 mA.
7. Not more than 1 output should be tested at a time. Duration of that test must not exceed 1 second.
8. With Turbo Bit Off, device automatically enters standby mode approximately 100 ns after last input transition.

**A.C. TESTING LOAD CIRCUIT**



**A.C. TESTING INPUT, OUTPUT WAVEFORM**



**CAPACITANCE**

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
C <sub>IN</sub>	Input Capacitance	V <sub>IN</sub> = 0V, f = 1.0 MHz			20	pF
C <sub>OUT</sub>	Output Capacitance	V <sub>OUT</sub> = 0V, f = 1.0 MHz			20	pF
C <sub>CLK</sub>	Clock Pin Capacitance	V <sub>IN</sub> = 0V, f = 1.0 MHz			20	pF
C <sub>VPP</sub>	V <sub>PP</sub> Pin	CLK2 on 5C060			50	pF

**A.C. CHARACTERISTICS** T<sub>A</sub> = 0°C to 70°C, V<sub>CC</sub> = 5V ± 5%, Turbo Bit On<sup>(9)</sup>

Symbol	From	To	Device						Non-(11) Turbo Mode	Unit
			5C060-45 EP600-3			5C060-55 EP600				
			Min	Typ	Max	Min	Typ	Max		
t <sub>PD1</sub>	Input	Comb. Output			43			53	+ 25	ns
t <sub>PD2</sub>	I/O	Comb. Output			45			55	+ 25	ns
t <sub>pZX</sub> <sup>(10)</sup>	I or I/O	Output Enable			45			55	+ 25	ns
t <sub>pXZ</sub> <sup>(10)</sup>	I or I/O	Output Disable			45			55	+ 25	ns
t <sub>CLR</sub>	Asynch. Reset	Q Reset			45			55	+ 25	ns

**NOTES:**

9. Typical Values are at T<sub>A</sub> = 25°C, V<sub>CC</sub> = 5V, Active Mode.

10. t<sub>pZX</sub> and t<sub>pXZ</sub> are measured at ±0.5V from steady state voltage as driven by spec. output load. t<sub>pXZ</sub> is measured with C<sub>L</sub> = 5 pF.

11. If device is operated with Turbo Bit Off (Non-Turbo Mode), increase time by amount shown.

**SYNCHRONOUS CLOCK MODE A.C. CHARACTERISTIC**
 $T_A = 0^\circ\text{C to } 70^\circ\text{C}, V_{CC} = 5.0\text{V} \pm 5\%, \text{ Turbo Bit On}^{(9)}$ 

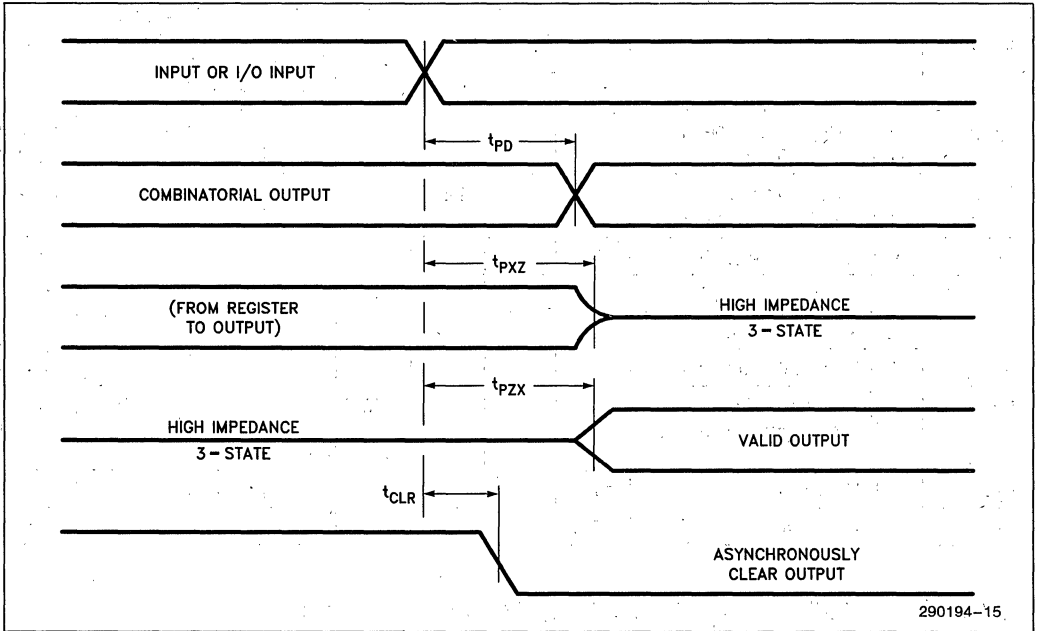
Symbol	Parameter	Device						Non-(11) Turbo Mode	Unit
		5C060-45 EP600-3			5C060-55 EP600				
		Min	Typ	Max	Min	Typ	Max		
$f_{\text{MAX}}$	Max. Frequency (Pipelined) ( $1/t_{\text{SU}}$ —No Feedback)			26.3			23.3		MHz
$f_{\text{CNT}}$	Max. Count Frequency ( $1/t_{\text{CNT}}$ —With Feedback)			22.2			18.2		MHz
$t_{\text{SU1}}$	Input Setup Time to CLK	36			41			+ 25	ns
$t_{\text{SU2}}$	I/O Setup Time to CLK	38			43			+ 25	ns
$t_{\text{H}}$	I or I/O Hold after CLK High	0			0				ns
$t_{\text{CO}}$	CLK High to Output Valid			22			25		ns
$t_{\text{CNT}}$	Register Output Feedback to Register Input—Internal Path	45			55			+ 25	ns
$t_{\text{CH}}$	CLK High Time	17.5			21.5				ns
$t_{\text{CL}}$	CLK Low Time	17.5			21.5				ns

**ASYNCHRONOUS CLOCK MODE A.C. CHARACTERISTICS**
 $T_A = 0^\circ\text{C to } 70^\circ\text{C}, V_{CC} = 5.0\text{V} \pm 5\%, \text{ Turbo Bit On}^{(8)}$ 

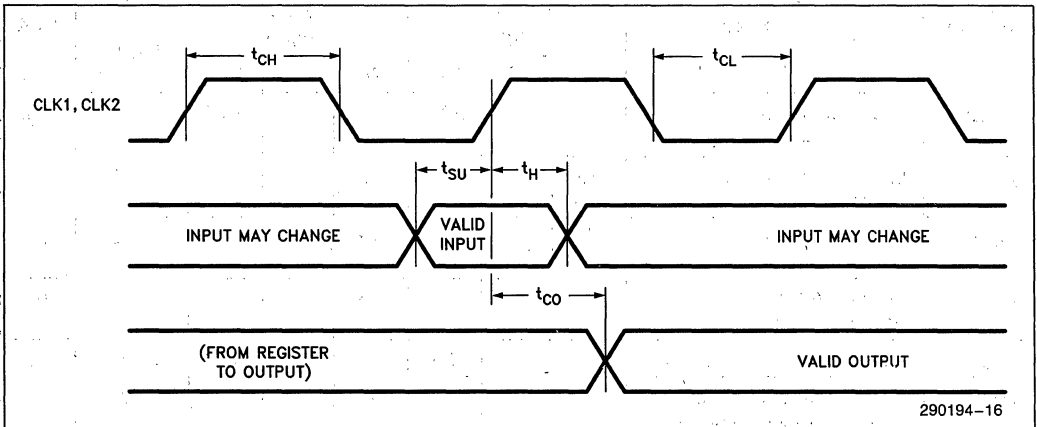
Symbol	Parameter	Device						Non-(11) Turbo Mode	Unit
		5C060-45 EP600-3			5C060-55 EP600				
		Min	Typ	Max	Min	Typ	Max		
$f_{\text{ACNT}}$	Max. Count Frequency ( $1/t_{\text{ACNT}}$ —With Feedback)			22.2			18.2		MHz
$t_{\text{ASU1}}$	Input Setup Time to Asynch. Clock	10			10			+ 25	ns
$t_{\text{ASU2}}$	I/O Setup Time to Asynch. Clock	12			12			+ 25	ns
$t_{\text{AH}}$	Input or I/O Hold After Asynch. Clock	15			15				ns
$t_{\text{ACO}}$	Asynch. CLK to Output Valid			50			58	+ 25	ns
$t_{\text{ACNT}}$	Register Output Feedback to Register Input—Internal Path	45			55			+ 25	ns
$t_{\text{ACH}}$	Asynch. CLK High Time	17.5			21.5			+ 25	ns
$t_{\text{ACL}}$	Asynch. CLK Low Time	17.5			21.5			+ 25	ns

**SWITCHING WAVEFORMS**

**COMBINATORIAL MODE**

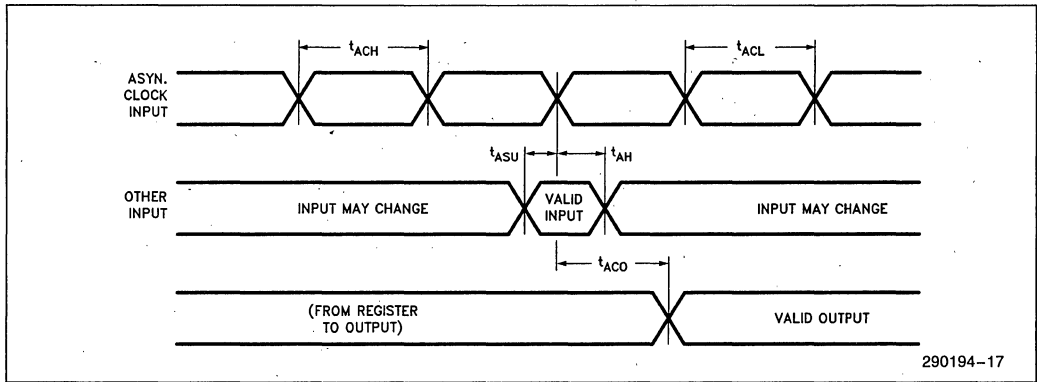


**SYNCHRONOUS CLOCK MODE**



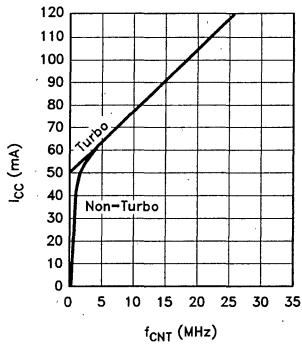
**SWITCHING WAVEFORMS** (Continued)

**ASYNCHRONOUS CLOCK MODE**



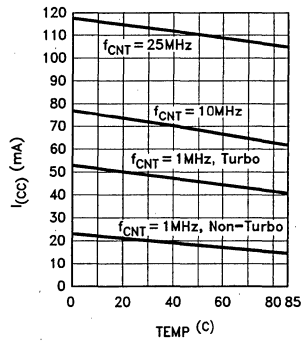
290194-17

**5C060**  
**Current in Relation to Frequency**



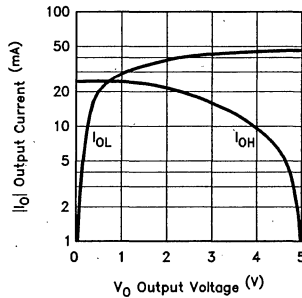
Conditions:  $T_A = 0^\circ\text{C}$ ,  $V_{CC} = 5.25\text{V}$  290194-18

**5C060**  
**Current in Relation to Temperature**



Conditions:  $V_{CC} = 5.25\text{V}$ , TTL inputs 290194-19

**5C060**  
**Output Drive Current in Relation to Voltage**



Conditions:  $T_A = 25^\circ\text{C}$  290194-20

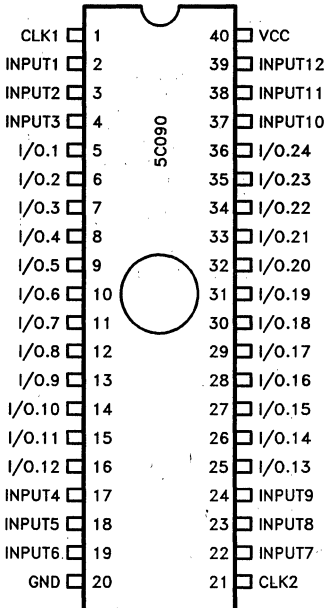




# 5C090 900-GATE CHMOS H-SERIES ERASABLE PROGRAMMABLE LOGIC DEVICE (H-EPLD)

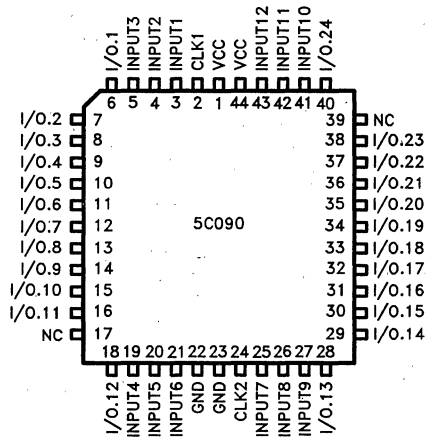
- High Performance LSI Semi-Custom Logic Alternative to Low-End Gate Arrays, TTL, and 74HC SSI and MSI Logic
- High Speed,  $t_{PD}$  (max) 50 ns, 26.3 MHz Pipelined, 20 MHz w/Feedback
- CHMOS EPROM Technology Based. UV Erasable
- Low Power; 50  $\mu$ A Typical Standby Current
- Erasable Array for 100% Generic Testability
- Programmable Clock System with Two Synchronous Clocks as Well as Asynchronous Clocking Option on all Registers
- Programmable Output Registers. Can be Configured as D, T, SR, or JK Types
- Programmable Security Bit Allows Total Protection of Proprietary Designs
- 24 Macrocells with Programmable I/O Architecture; Up to 36 Inputs (12 Dedicated, 24 I/O) or 24 Outputs
- 40-Pin DIP Package for Expanded I/O Capability
- 44-Pin J-Leaded Chip Carrier Package
- 100% Compatible with EP900

(See Packaging Spec., Order Number # 291369)



290195-1

Figure 1. 5C090 Pin Configurations



290195-2

The Intel 5C090 H-EPLD (H-series Programmable Logic Device) is capable of implementing over 900 equivalent gates of user-customized logic functions through programming. The device can be used to replace low-end gate arrays, multiple programmable logic arrays and LS TTL and 74HC (CMOS) SSI and MSI logic devices. With its revolutionary programmable I/O architecture, the device has advanced functional capabilities beyond that of typical programmable logic.

The 5C090 H-EPLD uses CHMOS EPROM (floating gate) cells as logic control elements instead of fuses. The CHMOS EPROM technology reduces power consumption of H-EPLDs to less than 20% of a comparable bipolar device without sacrificing speed performance. In addition, Intel's advanced CHMOS II-E EPROM process technology enables greater logic densities to be achieved with superior speed and low-power performance over other comparable

devices. Intel's H-ELPDs add the benefits of "zero" stand-by power not available on other programmable logic devices. EPROM technology allows these devices to be 100% factory tested by programming and erasing all the EPROM logic control elements.

The erasability of EPLDs introduces the designer to a new concept in hardware design called Modular EPLD Logic Design (MELD). Just as modular software design speeds development time and reduces errors by isolating them to a specific module, the MELD philosophy aids in hardware design. A designer can develop his modular design on the Intel Programmable Logic Development System II (iPLDS II) and test individual modules for functionality. If one of the modules has a design flaw, the designer merely erases the part and starts anew (since the 5C090 is EPROM-based, there is no waste associated with modular design as there would be in fuse-based PLDs).

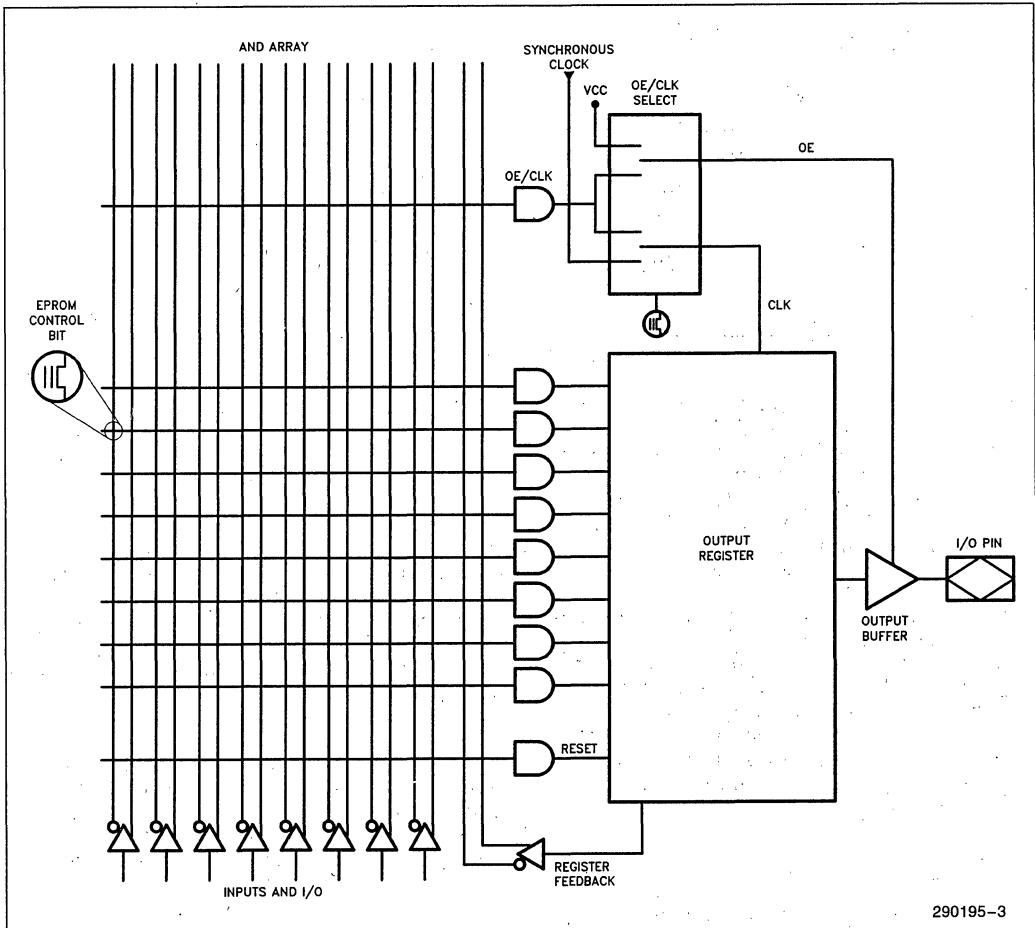


Figure 2. Basic Macrocell Architecture of the 5C090

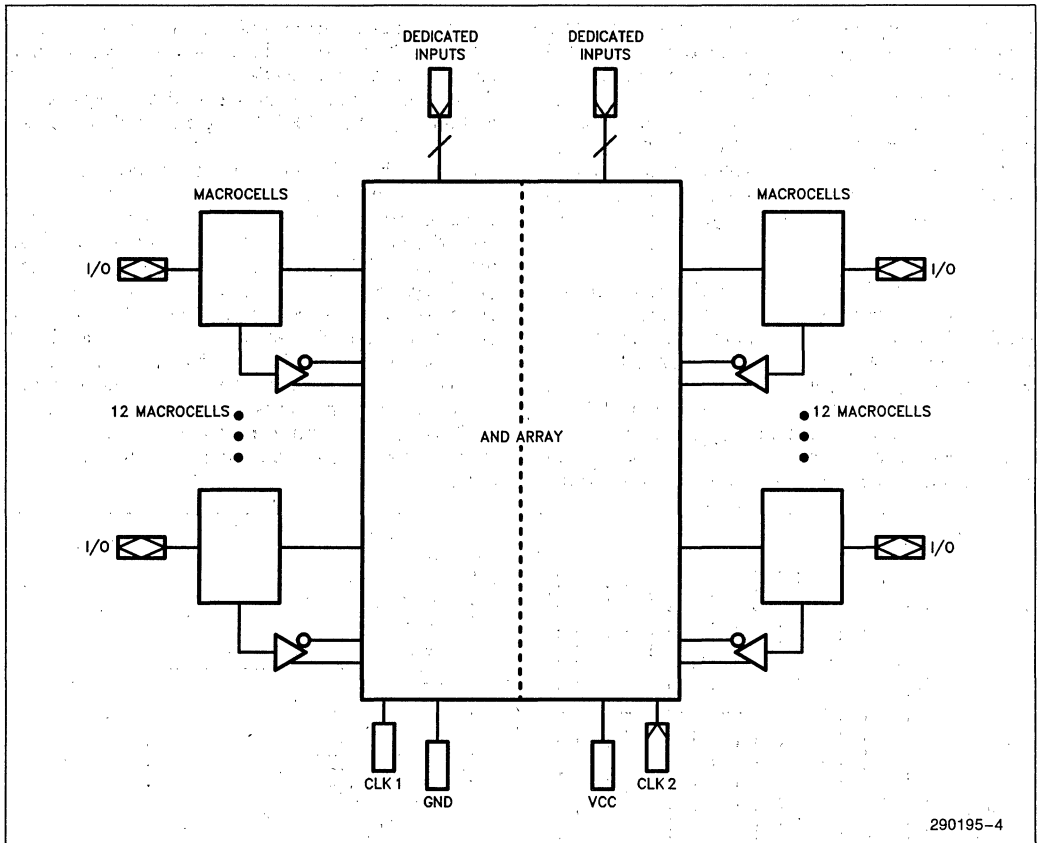


Figure 3. 5C090 Global Architecture

The architecture of the 5C090 is based on the "Sum of Products" PLA (Programmable Logic Array) structure with a programmable AND array feeding into a fixed OR array. The device accommodates combinational and sequential logic functions. A proprietary programmable I/O architecture provides individual selection of either combinational or registered output and feedback signals all with selectable polarity.

A feature unique to the 5C090 is the ability to individually program the output registers as a D-, T-, SR-, or JK-type Flip-Flop without sacrificing the utilization of programmable AND logic. Additionally, each output register can be individually clocked from any of the input or feedback paths available within the AND array. With these features, a wide variety of logic functions can be simultaneously implemented—all on the same device.

## ARCHITECTURE DESCRIPTION

The 5C090 has 12 dedicated inputs, 24 I/O pins which may be configured for input, output, or bidirec-

ditional operations, and 2 synchronous clock inputs. The 5C090 is packaged in a 40-lead windowed ceramic DIP or 44-lead J-leaded chip carrier package and contains 24 programmable registers.

The basic Macrocell architecture for the 5C090 is shown in Figure 2. The 5C090 has 24 of these macrocells (one for each I/O pin). The Macrocell is organized in the familiar sum-of-products structure with a programmable AND array attached to a fixed OR term. The inputs to the programmable AND array originate from the true and complement signals from each of the dedicated input pins and each of the I/O control blocks.

The AND array for the 5C090 has 72 inputs derived from the true and complement signals at the input and I/O pins. The AND array in the 5C090 encompasses 240 product terms which are distributed among the 24 Macrocells. The global device architecture is shown in Figure 3.

The Macrocells contain ten product terms total. Eight of the ten product terms (AND gates) are dedicated for logic implementation. One product term on each Macrocell is used for RESET control to the output register associated with the Macrocell. The final product term is used for OUTPUT ENABLE/Asynchronous Clock implementation.

Within the AND array, there is an EPROM connection at every intersection of an input signal (true and complement) and a product term to a given Macrocell. Before programming an erased device, every EPROM connection is made at every intersection. But during the programming process, these connections are opened so that only the desired connections remain. Therefore, the true or complement of any input signal can be connected to any product term. If both the true and complement connections of any signal are left intact, a logical false results on the output of the AND gate. However, if both the true and complement connections are open, then a logic "don't care" results on the AND gate. Lastly, if all the inputs of a product term are programmed open, then a logical true results on the output of the AND gate.

The 5C090 has two dedicated clock inputs to provide synchronous clock signals to the internal registers. Each of the clock signals controls half the total registers within the given device. For example, CLK1 provides synchronous clocking to the registers in Macrocells in the left half of the array while CLK2 controls the registers associated with Macrocells in the right half of the array. The advanced I/O architecture allows for any number of the registers to be synchronously clocked (from none to all). Both of the dedicated clock inputs latch the data into a given register when triggered on a positive edge.

**MACROCELL ARCHITECTURE SELECTION**

The 5C090 architecture provides each Macrocell with over 50 different possible I/O register configurations. Each I/O pin can be configured for combinatorial or registered output (true or complement) with feedback. In addition, four different types of output registers can be implemented into every I/O pin without any additional logic requirements. The feedback mechanism for each register back into the AND array can be programmed to provide for either registered feedback from the Macrocell or input feedback (treating the pin as an input). Another advantage of the advanced I/O capability of the 5C090 is the ability to individually clock each internal register from asynchronous clock signals.

**Output Enable (OE)/Clock Selection**

Two modes of operation are provided by the OE/CLK Select Multiplexer as a part of each Macrocell. One mode provides for three-state buffering of outputs while in the other mode, the outputs are always enabled. The operation of the OE/CLK Select Multiplexer sets the mode within a given Macrocell. Therefore, the output mode can be selected individually on every output. Figure 4 illustrates the two modes of OE/CLK operation.

**MODE 0: THREE-STATE BUFFERING**

In Mode 0, the three-state output buffer is controlled by a single product term originating from the AND array. The output is enabled when the product term is a logical true. Conversely, the output appears as high impedance when the product term is a logical false as shown in Table 1. In Mode 0, the Macrocell Flip-Flop is connected to its associated synchronous clock (either CLK1 or CLK2 depending upon the Macrocell's location within the device). Thus, the Macrocell Flip-Flop may be clocked by its respective synchronous clock but its output will not become valid until the output is enabled.

**Table 1. Mode 0 Output Selection**

Product Term	Output Buffer
FALSE	Three-State
TRUE	Enabled

**MODE 1: OUTPUT BUFFER ENABLED**

In Mode 1, the Output Buffer is always enabled. In addition, the Macrocell Flip-Flop is connected to the AND array. The Macrocell Flip-Flop may now be triggered from an asynchronous clock signal generated by the AND array logic to the OE/CLK multiplexable term. Mode 1 allows the Macrocell Flip-Flops to be individually clocked from any of the available signals in the AND array. Since both true and complement values appear in the AND array, the Flip-Flop may be configured to trigger on positive or negative clock edges. Gated clock structures can be created since the Flip-Flop clock is created by a product term.

**Invert Select EPROM Bit**

The Invert Select EPROM bit is used to invert the product term input into the register. This applies to all inputs including double inputs on the JK and SR registers.

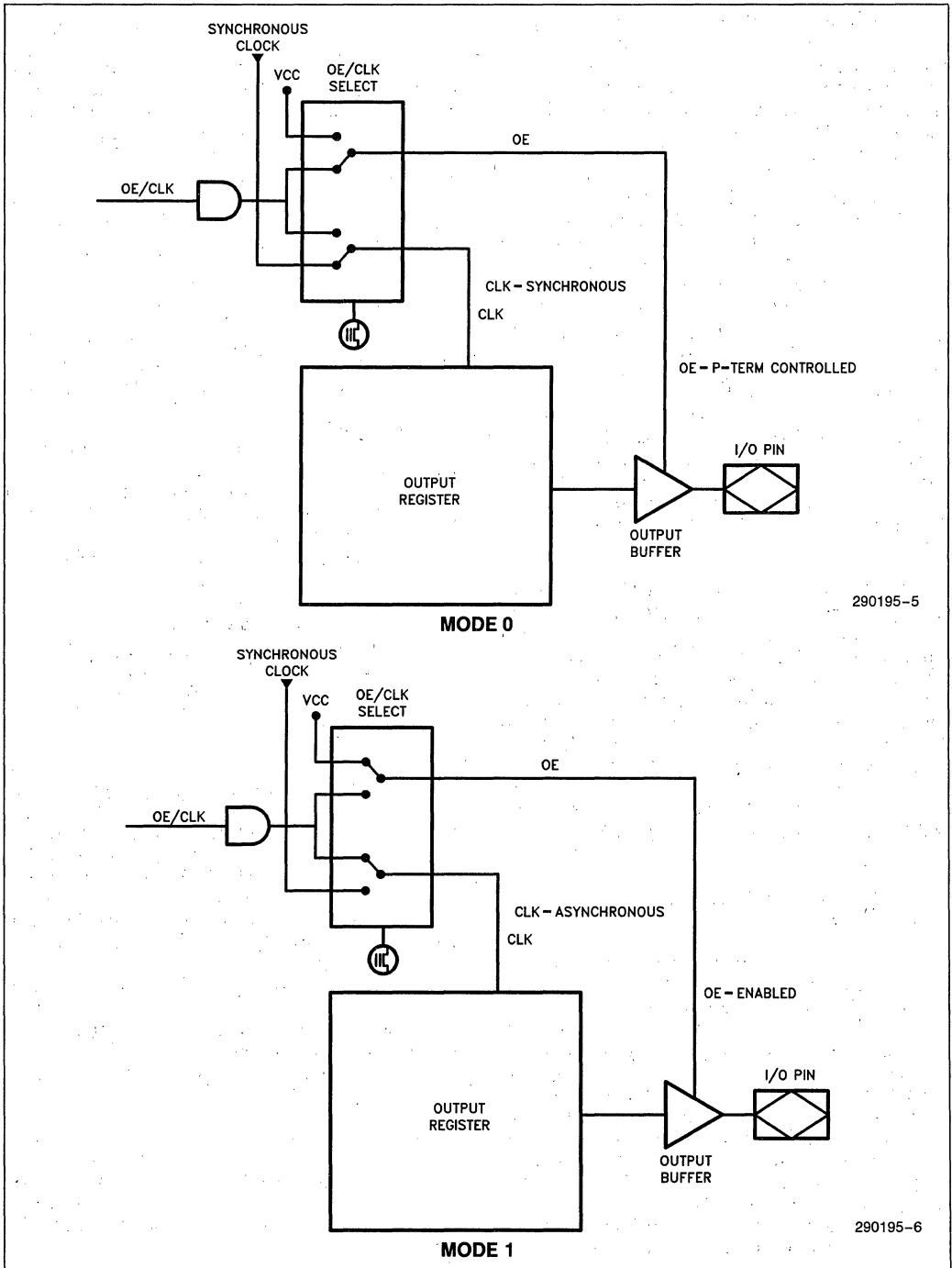


Figure 4. Output Enable/Clock Configuration

## REGISTER SELECTION

The advanced I/O architecture of the 5C090 allows four different register types along with combinatorial output as illustrated in Figure 5a through e. The register types include a T, D, JK, or SR Flip-Flop and each Macrocell I/O structure may be independently configured. In addition, all registers have an individual asynchronous RESET control from a dedicated product term derived in the AND array. When this dedicated product term is a logical one, the Macrocell register is immediately cleared to a logical zero independent of the register clock. The RESET function occurs automatically on power-up.

### Output Register Configuration

The four different register types shown in Figure 5 are described below.

#### D- or T-type Flip-Flops

When either a D- or T-type Flip-Flop is configured as part of the I/O structure, all eight of the product terms into the Macrocell are ORed together and fed into the register input.

#### JK or SR Registers

When either a JK or SR register is configured, the eight product terms are shared among two OR gates (one for the J or S input and the other for the K or R input). The allocation for these product terms for each of the register inputs is optimized by the iPLDS II development software.

## OUTPUT/FEEDBACK

The Output Select Multiplexer allows for either registered, combinatorial or no output.

The Feedback Select Multiplexer EPROM bit enables registered, I/O (using the pin for bidirectional input or just input), or no feedback to the AND array.

The Feedback Select is also important for building product terms with more than 8 products. The 8-product product term of a Macrocell can be fed back into the AND array and combined with still more signals to create a much larger product term (of more than 8-inputs). In addition, if the feedback product term is not to be output, then the iPLDS II will reserve the associated Macrocell pin and indicate it in the REPORT file. A reserved pin should be left floating (no connect) when assembled onto a circuit board.

Any I/O pin may be configured as a dedicated input by selecting no output and pin feedback through the appropriate multiplexers.

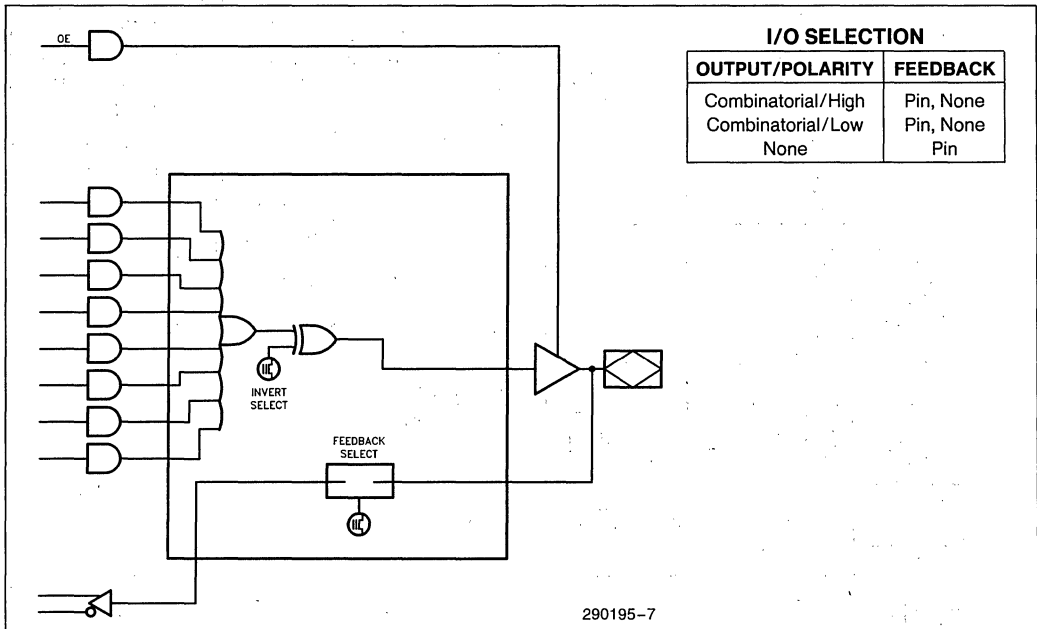


Figure 5a. Combinatorial I/O Configuration

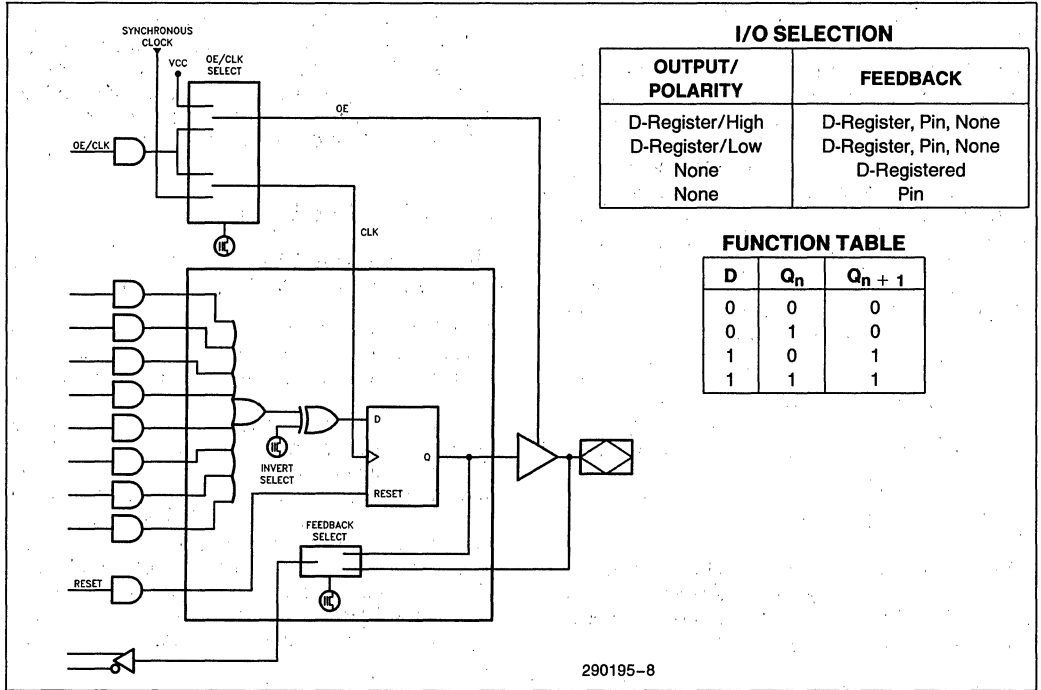


Figure 5b. D-Type Flip-Flop Register Configuration

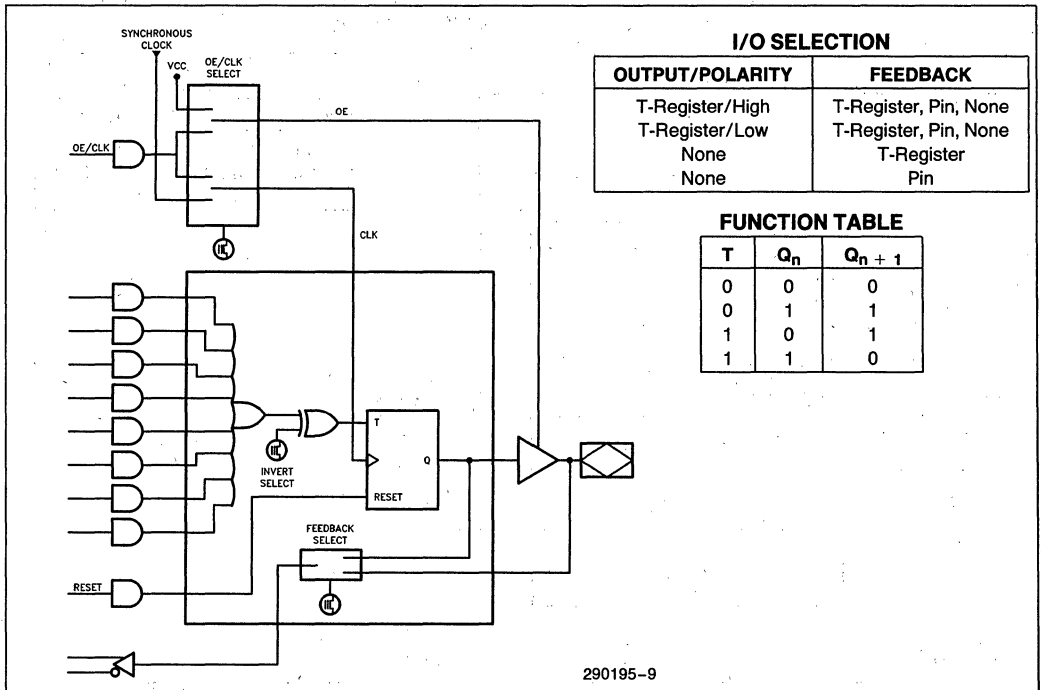


Figure 5c. Toggle Flip-Flop Register Configuration

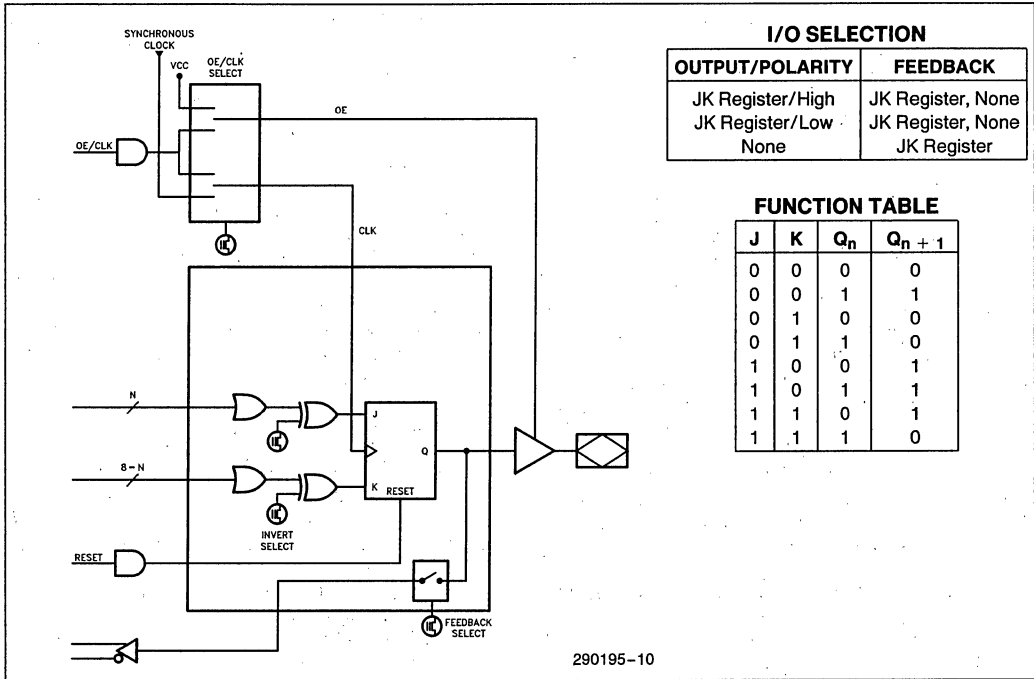


Figure 5d. JK Flip-Flop Register Configuration

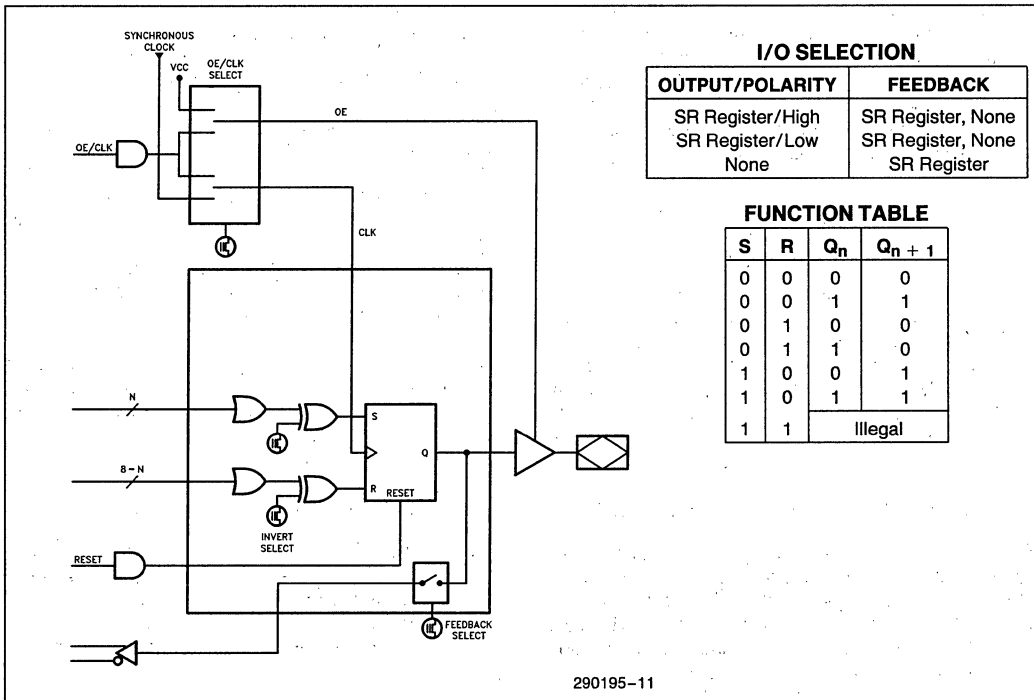


Figure 5e. SR Flip-Flop Register Configuration



## Erased-State Configuration

Prior to programming or after erasing, the I/O structure is configured for combinatorial active low output with input (pin) feedback.

## ERASURE CHARACTERISTICS

Erasure characteristics of the device are such that erasure begins to occur upon exposure to light with wavelengths shorter than approximately 4000Å. It should be noted that sunlight and certain types of fluorescent lamps have wavelengths in the 3000–4000Å. Data shows that constant exposure to room level fluorescent lighting could erase the typical device in approximately three years, while it would take approximately one week to cause erasure when exposed to direct sunlight. If the 5C090 is to be exposed to these types of lighting conditions for extended periods of time, conductive opaque labels should be placed over the device window to prevent unintentional erasure.

The recommended erasure procedure for the 5C090 is exposure to shortwave ultraviolet light with a wavelength of 2537Å. The integrated dose (i.e., UV intensity  $\times$  exposure time) for erasure should be a minimum of fifteen (15) Wsec/cm<sup>2</sup>. The erasure time with this dosage is approximately 15 to 20 minutes using an ultraviolet lamp with a 12,000  $\mu$ W/cm<sup>2</sup> power rating. The 5C090 should be placed within one inch of the lamp tubes during erasure. The maximum integrated dose the 5C090 can be exposed to without damage is 7258 Wsec/cm<sup>2</sup> (1 week at 12,000  $\mu$ W/cm<sup>2</sup>). Exposure to high intensity UV light for longer periods may cause permanent damage to the device.

## PROGRAMMING CHARACTERISTICS

Initially, and after erasure, all the EPROM control bits of the 5C090 are connected (in the "1" state). Each of the connected control bits are selectively disconnected by programming the EPROM cells into their "0" state. Programming voltage and waveform specifications are available by request from Intel to support programming of the 5C090.

## intelligent Programming™ Algorithm

The 5C090 supports the intelligent Programming Algorithm which rapidly programs Intel H-ELPDs (and EPROMs) using an efficient and reliable method. The intelligent Programming Algorithm is particularly suited to the production programming environment. This method greatly decreases the overall program-

ming time while programming reliability is ensured as the incremental program margin of each bit is continually monitored to determine when the bit has been successfully programmed.

## FUNCTIONAL TESTING

Since the logical operation of the 5C090 is controlled by EPROM elements, the device is completely testable. Each programmable EPROM bit controlling the internal logic is tested using application-independent test program patterns. After testing, the devices are erased before shipment to customers. No post-programming tests of the EPROM array are required.

The testability and reliability of EPROM-based programmable logic devices is an important feature over similar devices based on fuse technology. Fuse-based programmable logic devices require a user to perform post-programming tests to insure proper programming. These tests must be done at the device level because of the cumulative error effect. For example, a board containing ten devices each possessing a 2% device fallout translates into an 18% fallout at the board level (it should be noted that programming fallout of fuse-based programmable logic devices is typically 2% or higher).

## DESIGN RECOMMENDATIONS

For proper operation, it is recommended that all input and output pins be constrained to the voltage range  $GND < (V_{IN} \text{ or } V_{OUT}) < V_{CC}$ . Unused inputs should be tied to an appropriate logic level (e.g. either  $V_{CC}$  or  $GND$ ) to minimize device power consumption. Reserved pins (as indicated in the logic compiler REPORT file) should be left floating (no connect) so that the pin can attain the appropriate logic level. A power-supply decoupling capacitor of at least 0.2  $\mu$ F must be connected directly between  $V_{CC}$  and  $GND$  pins of the device.

As with all CMOS devices, ESD handling procedures should be used with the 5C090 to prevent damage to the device during programming, assembly and test.

## DESIGN SECURITY

A single EPROM bit provides a programmable design security feature that controls the access to the data programmed into the device. If this bit is set, a proprietary design within the device cannot be copied. This EPROM security bit enables a higher de-

gree of design security than fused-based devices since programmed data within EPROM cells is invisible even to microscopic evaluation. The EPROM security bit, along with all the other EPROM control bits, will be reset by erasing the device.

**AUTOMATIC STAND-BY MODE**

The 5C090 contains a programmable bit, the Turbo Bit, that optimizes operation for speed or for power savings. When the Turbo Bit is programmed (TURBO = ON), the device is optimized for maximum speed. When the Turbo Bit is not programmed (TURBO = OFF), the device is optimized for power savings by entering standby mode during periods of inactivity.

Figure 6 shows the device entering standby mode approximately 100 ns after the last input transition. When the next input transition is detected, the device returns to active mode. Wakeup time adds an additional 25 ns to the propagation delay through the device as measured from the first input. No delay will occur if an output is dependent on more than one input and the last of the inputs changes after the device has returned to active mode.

After erasure, the Turbo Bit is unprogrammed (OFF); automatic standby mode is enabled. When the Turbo Bit is programmed (ON), the device never enters standby mode.

**LATCH-UP IMMUNITY**

All of the input, I/O, and clock pins of the 5C090 have been designed to resist latch-up which is inherent in inferior CMOS structures. The 5C090 is designed with Intel's proprietary CHMOS II-E EPROM process. Thus, each of the pins will not experience latch-up with currents up to 100 mA and voltages ranging from -1V to V<sub>CC</sub> + 1V. Furthermore, the programming pin is designed to resist latch-up to the 13.5V maximum device limit.

**INTEL PROGRAMMABLE LOGIC DEVELOPMENT SYSTEM II (iPLDS II)**

iPLDS II graphically provides all the tools needed to design with Intel H-Series EPLDs or compatible devices. In addition to providing development assistance, iPLDS II insulates the user from having to know all the intricate details of EPLD architecture (the machine will optimize a design to benefit from architectural features). It contains comprehensive third generation software that supports four different design entry methods, minimizes logic, does automatic pin assignments and produces the best design fit for the selected EPLD. It is user friendly with guided menus, on-line Help messages and soft key inputs.

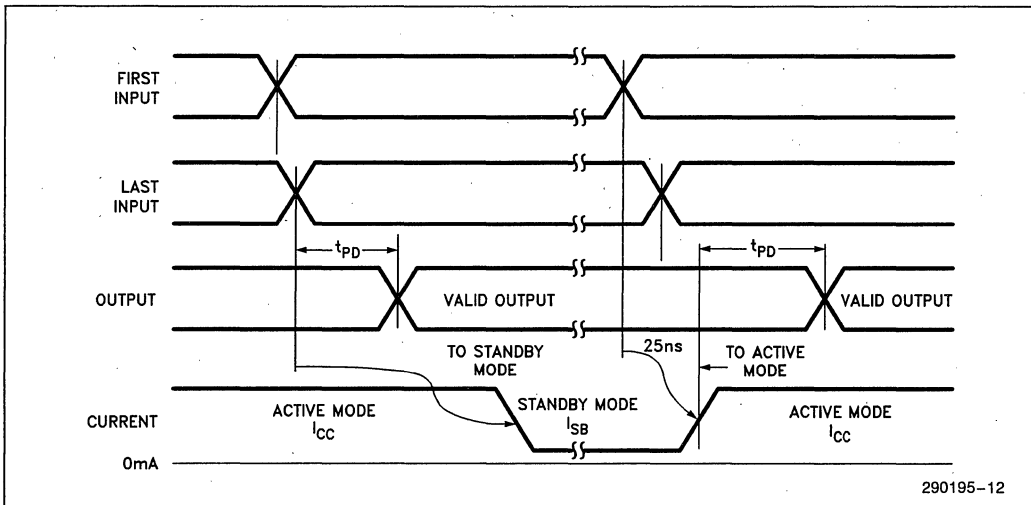


Figure 6. 5C090 Standby and Active Mode Transitions

In addition, the iPLDS II contains programmer hardware in the form of an iUP-PC Universal Programmer Personal Computer to enable the user to program EPLDs, read and verify programmed devices and also to graphically edit programming files. The software generates industry standard JEDEC object code output files which can be downloaded to other programmers as well.

The iPLDS II has interfaces to popular schematic capture packages to enable designs to be entered using schematics. An integrated schematic entry method is provided by SCHEMA II-PLD, a low-cost schematic capture package that supports EPLD primitives and user-defined macro symbols. SCHEMA II-PLD contains the EPLD Design Manager, which provides a single user interface to both SCHEMA II-PLD and iPLS II software. The other design formats supported are Boolean equation entry and State Machine design entry.

The iPLDS II operates on the IBM† PC/XT, PC/AT, or other compatible machine with the following configuration:

1. At least one floppy disk drive and hard disk drive.
2. MS-DOS†† Operating System Version 3.0 or greater.
3. 512K Memory (640K recommended).
4. Intel iUP-PC Universal Programmer Personal Computer and GUPI Adaptor (supplied with iPLDS II).
5. A color monitor is suggested.

Detailed information on the Intel Programmable Logic Development System II is contained in a separate Intel data sheet. (Order Number: 280168)

†IBM Personal Computer is a registered trademark of International Business Machines Corporation.

††MS-DOS is a registered trademark of Microsoft Corporation.

### ADF PRIMITIVES SUPPORTED

The following ADF primitives are supported by this device:

INP	JOJF
CONF	JONF
COIF	SONF
RONF	SOSF
RORF	TOIF
ROIF	TONF
NORF	TOTF
NOJF	CLKB
NOSF	
NOTF	

### ORDERING INFORMATION

t <sub>PD</sub> (ns)	t <sub>CO</sub> (ns)	f <sub>MAX</sub> (MHz)	Order Code	Package	Operating Range
50	23	26.3	D5C090-50	CERDIP	Commercial
			P5C090-50	PDIP	
			CJ5C090-50	JLCC	
			N5C090-50	PLCC	
60	25	21.7	D5C090-60	CERDIP	Commercial
			P5C090-60	PDIP	
			CJ5C090-60	JLCC	
			N5C090-60	PLCC	

**ABSOLUTE MAXIMUM RATINGS\***

Symbol	Parameter	Min	Max	Units
V <sub>CC</sub>	Supply Voltage <sup>(1)</sup>	-2.0	7.0	V
V <sub>PP</sub>	Programming Supply Voltage <sup>(1)</sup>	-2.0	13.5	V
V <sub>I</sub>	DC Input Voltage <sup>(1)(2)</sup>	-0.5	V <sub>CC</sub> +0.5	V
t <sub>stg</sub>	Storage Temperature	-65	+150	°C
t <sub>amb</sub>	Ambient Temperature <sup>(3)</sup>	-10	+85	°C

**NOTES:**

1. Voltages with respect to ground.
2. Minimum DC input is -0.5V. During transitions, the inputs may undershoot to -2.0V or overshoot to 7.0V for periods less than 20 ns under no load conditions.
3. Under bias. Extended temperature versions are also available.

*\*Notice: Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.*

**RECOMMENDED OPERATING CONDITIONS**

Symbol	Parameter	Min	Max	Unit
V <sub>CC</sub>	Supply Voltage	4.75	5.25	V
V <sub>IN</sub>	Input Voltage	0	V <sub>CC</sub>	V
V <sub>O</sub>	Output Voltage	0	V <sub>CC</sub>	V
T <sub>A</sub>	Operating Temperature	0	+70	°C
t <sub>R</sub>	Input Rise Time		500	ns
t <sub>F</sub>	Input Fall Time		500	ns

**NOTE:**

4. t<sub>R</sub>, t<sub>F</sub> for CLK is 250 ns max.

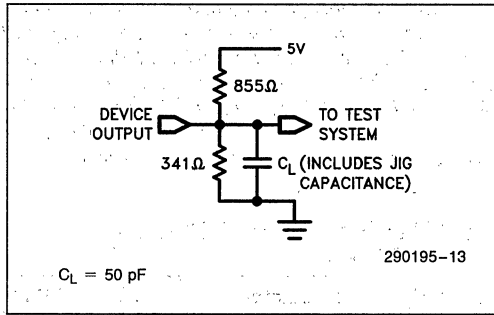
**D.C. CHARACTERISTICS** T<sub>A</sub> = 0°C to 70°C, V<sub>CC</sub> = 5.0V ± 5%

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
V <sub>IH</sub> <sup>(5)</sup>	HIGH Level Input Voltage		2.0		V <sub>CC</sub> + 0.3	V
V <sub>IL</sub> <sup>(5)</sup>	LOW Level Input Voltage		-0.3		0.8	V
V <sub>O</sub> H <sup>(6)</sup>	HIGH Level Output Voltage	I <sub>O</sub> = -4.0 mA DC, V <sub>CC</sub> = Min.	2.4			V
V <sub>O</sub> L	LOW Level Output Voltage	I <sub>O</sub> = 4.0 mA DC, V <sub>CC</sub> = Min.			0.45	V
I <sub>I</sub>	Input Leakage Current	V <sub>CC</sub> = Max., GND < V <sub>IN</sub> < V <sub>CC</sub>			±10.0	μA
I <sub>OZ</sub>	Output Leakage Current	V <sub>CC</sub> = Max., GND < V <sub>OUT</sub> < V <sub>CC</sub>			±10.0	μA
I <sub>SC</sub> <sup>(7)</sup>	Output Short Circuit Current	V <sub>CC</sub> = Max., V <sub>OUT</sub> = 0.5V		20	30	mA
I <sub>SB</sub> <sup>(8)</sup> 5C090	Standby Current (Standby)	V <sub>CC</sub> = Max., V <sub>IN</sub> = V <sub>CC</sub> or GND		50	100	μA
I <sub>CC</sub> 5C090	Power Supply Current (Active) (Turbo Bit Off) Device Prog. as Two 12-Bit Ctrs.	V <sub>CC</sub> = Max., V <sub>IN</sub> = V <sub>CC</sub> or GND No Load, Input Freq. = 1 MHz		15	25	mA

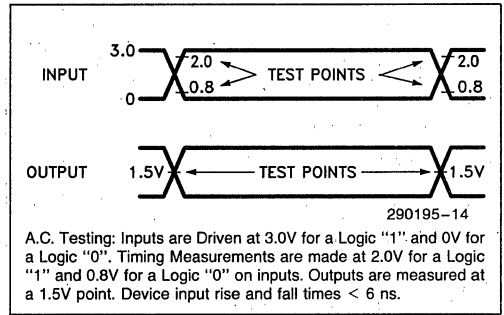
**NOTES:**

5. Absolute values with respect to device GND; all over and undershoots due to system or tester noise are included.
6. I<sub>O</sub> at CMOS levels (3.84V) = -2 mA.
7. Not more than 1 output should be tested at a time. Duration of that test must not exceed 1 second.
8. With Turbo Bit Off, device automatically enters standby mode approximately 100 ns after last input transition.

**A.C. TESTING LOAD CIRCUIT**



**A.C. TESTING INPUT, OUTPUT WAVEFORM**



**CAPACITANCE**

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
C <sub>IN</sub>	Input Capacitance	V <sub>IN</sub> = 0V, f = 1.0 MHz			20	pF
C <sub>OUT</sub>	Output Capacitance	V <sub>OUT</sub> = 0V, f = 1.0 MHz			20	pF
C <sub>CLK</sub>	Clock Pin Capacitance	V <sub>IN</sub> = 0V, f = 1.0 MHz			20	pF
C <sub>VPP</sub>	V <sub>PP</sub> Pin	CLK2 on 5C090			80	pF

**A.C. CHARACTERISTICS** T<sub>A</sub> = 0°C to 70°C, V<sub>CC</sub> = 5V ± 5%, Turbo Bit On<sup>(9)</sup>

Symbol	From	To	Device						Non-(11) Turbo Mode	Unit
			5C090-50 EP900-2			5C090-60 EP900				
			Min	Typ	Max	Min	Typ	Max		
t <sub>PD1</sub>	Input	Comb. Output			45			55	+ 25	ns
t <sub>PD2</sub>	I/O	Comb. Output			50			60	+ 25	ns
t <sub>PZX</sub> <sup>(10)</sup>	I or I/O	Output Enable			50			60	+ 25	ns
t <sub>PXZ</sub> <sup>(10)</sup>	I or I/O	Output Disable			50			60	+ 25	ns
t <sub>CLR</sub>	Asynch. Reset	Q Reset			50			60	+ 25	ns

**NOTES:**

- 9. Typical Values are at T<sub>A</sub> = 25°C, V<sub>CC</sub> = 5V, Active Mode.
- 10. t<sub>PZX</sub> and t<sub>PXZ</sub> are measured at ±0.5V from steady state voltage as driven by spec. output load. t<sub>PXZ</sub> is measured with C<sub>L</sub> = 5 pF.
- 11. If device is operated with Turbo Bit Off (Non-Turbo Mode), increase time by amount shown.

**SYNCHRONOUS CLOCK MODE A.C. CHARACTERISTIC**

$T_A = 0^\circ\text{C to } 70^\circ\text{C}$ ,  $V_{CC} = 5.0\text{V} \pm 5\%$ , Turbo Bit On<sup>(9)</sup>

Symbol	Parameter	Device						Non-(11) Turbo Mode	Unit
		5C090-50 EP900-2			5C090-60 EP900				
		Min	Typ	Max	Min	Typ	Max		
f <sub>MAX</sub>	Max. Frequency (Pipelined) (1/t <sub>SU</sub> —No Feedback)			26.3			21.7		MHz
f <sub>CNT</sub>	Max. Count Frequency (1/t <sub>CNT</sub> —With Feedback)			20			16.7		MHz
t <sub>SU1</sub>	Input Setup Time to CLK	36			43			+ 25	ns
t <sub>SU2</sub>	I/O Setup Time to CLK	38			46			+ 25	ns
t <sub>H</sub>	I or I/O Hold after CLK High	0			0				ns
t <sub>CO</sub>	CLK High to Output Valid			23			25		ns
t <sub>CNT</sub>	Register Output Feedback to Register Input—Internal Path	50			60			+ 25	ns
t <sub>CH</sub>	CLK High Time	17.5			23				ns
t <sub>CL</sub>	CLK Low Time	17.5			23				ns

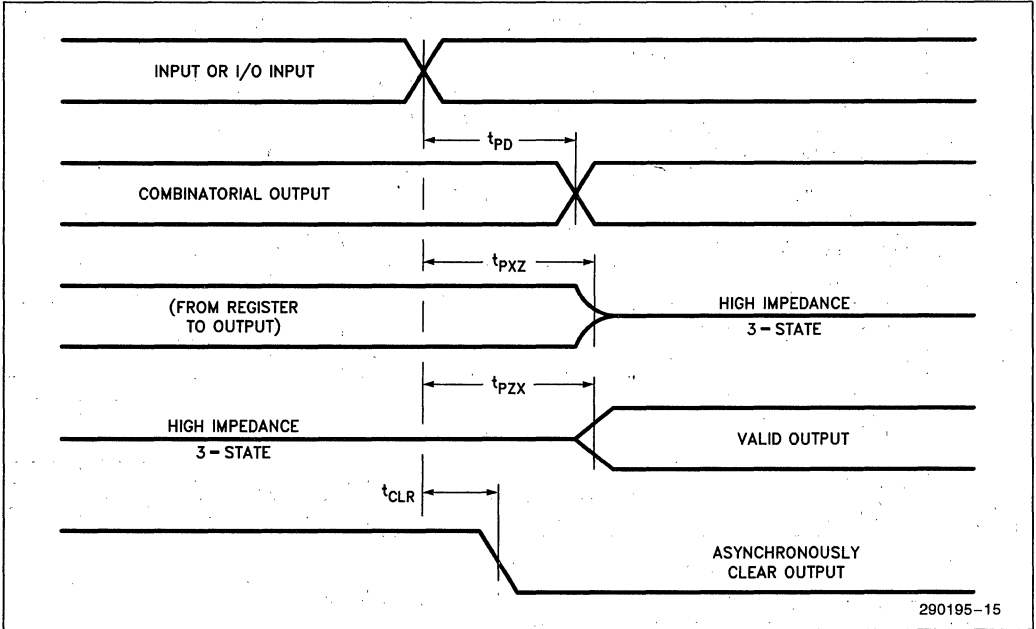
**ASYNCHRONOUS CLOCK MODE A.C. CHARACTERISTICS**

$T_A = 0^\circ\text{C to } 70^\circ\text{C}$ ,  $V_{CC} = 5.0\text{V} \pm 5\%$ , Turbo Bit On<sup>(8)</sup>

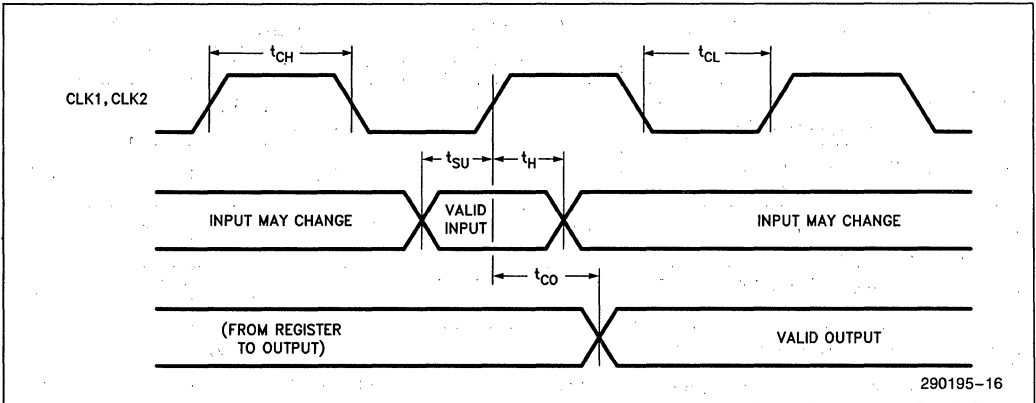
Symbol	Parameter	Device						Non-(11) Turbo Mode	Unit
		5C090-50 EP900-2			5C090-60 EP900				
		Min	Typ	Max	Min	Typ	Max		
f <sub>ACNT</sub>	Max. Count Frequency (1/t <sub>ACNT</sub> —With Feedback)			20			16.7		MHz
t <sub>ASU1</sub>	Input Setup Time to Asynch. Clock	10			10			+ 25	ns
t <sub>ASU2</sub>	I/O Setup Time to Asynch. Clock	13			15			+ 25	ns
t <sub>AH</sub>	Input or I/O Hold After Asynch. Clock	15			15				ns
t <sub>ACO</sub>	Asynch. CLK to Output Valid			48			59	+ 25	ns
t <sub>ACNT</sub>	Register Output Feedback to Register Input—Internal Path	50			60			+ 25	ns
t <sub>ACh</sub>	Asynch. CLK High Time	17.5			23			+ 25	ns
t <sub>ACL</sub>	Asynch. CLK Low Time	17.5			23			+ 25	ns

**SWITCHING WAVEFORMS**

**COMBINATORIAL MODE**

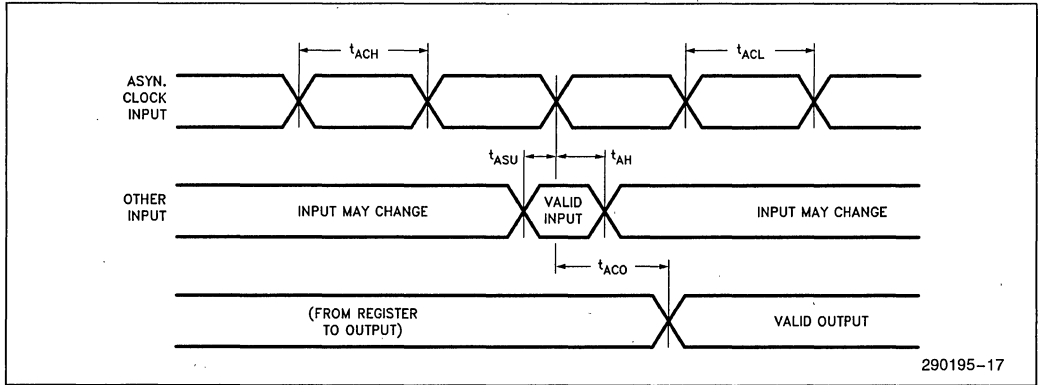


**SYNCHRONOUS CLOCK MODE**

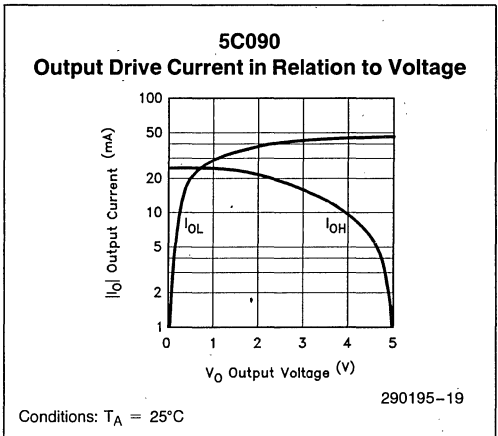
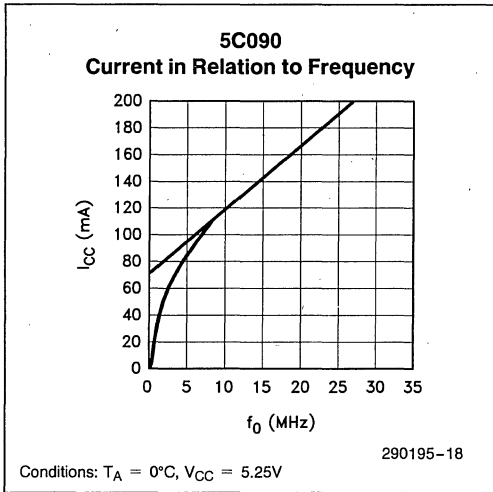


SWITCHING WAVEFORMS (Continued)

ASYNCHRONOUS CLOCK MODE



290195-17







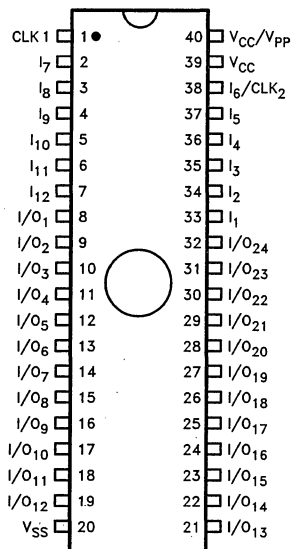


# 5C121 1200 GATE CHMOS H-SERIES ERASABLE PROGRAMMABLE LOGIC DEVICE

- High Performance LSI Semi-Custom Logic Replacement for Gate Arrays and Conventional Fixed Logic
- EPROM Technology Based. UV Erasable
- Programmable Macrocell and I/O Architecture; up to 36 Inputs or 24 Outputs, 28 Macrocells Including 4 Buried Registers
- All Inputs are Latchable with a Programmable Latch Feature
- High Speed  $t_{PD}$  (Max) 50 ns Operating Frequency (Max) 20 MHz
- Low Power; 15 mW Typical Standby Dissipation
- Typical Usable Gate Count of 1200 2-Input NAND Gates
- Advanced Architecture Features Including Programmable Output Polarity (Active High/Low), Register By-Pass and Reset Controls
- Programmable Clock System for Input Latches and Output Registers
- Product-Term Sharing and Local Bus Architecture for Optimized Array Performance
- Compatible with LS TTL and 74HC CMOS Logic
- Register Pre-Load and Erasable Array for 100% Generic Testability
- Programmable "Security Bit" allows total protection of proprietary designs
- Available in a 40-Lead Window Cerdip Package (See Packaging Spec, Order #231369)
- Fully Compatible with EP1210

The Intel 5C121 H-EPLD (H-series Erasable Programmable Logic Device) is an LSI logic circuit that is user customizable through programming. This device can be used to replace gate arrays, multiple programmable logic arrays and LS TTL and 74HC CMOS SSI and MSI logic devices. The logic capacity of the 5C121 is typically equal to 1200 two-input NAND gates.

## Pin Configuration



290098-1

ILLUSTRATIONS COURTESY OF ALTERA CORPORATION.

The 5C121 H-EPLD uses CHMOS\* EPROM (floating gate) cells as logic control elements instead of fuses. Use of Intel's advanced CHMOS IIE EPROM process technology enables greater logic densities to be achieved with superior speed and power performance. The EPROM technology also enables these devices to be 100% factory tested by the programming and the erasure of all the EPROM logic control elements in the device.

The architecture of the 5C121 is based on the 'Sum of Products' PLA (Programmable Logic Array) structure with a programmable AND array feeding into a fixed OR array. Flexibility in accommodating logical functions without the overhead of unnecessary product terms or speed penalties of programmable OR structures is achieved through the provision of a range of OR gate widths as well as through product term sharing. The use of a segmented PLA structure with local and global connectivity allows for further improvements in performance. The 5C121 also contains innovative architectural features that provide extensive Input/Output flexibility.

## ARCHITECTURE DESCRIPTION

The 5C121 H-EPLD has 12 dedicated inputs as well as 24 Input/Output pins. All inputs to the circuit (both dedicated and I/O inputs) may be latched using transparent 7475 type latches. In addition to these 36 input latches, 28 D type registers are also provided.

The internal architecture of the 5C121 H-EPLD is based on 28 macrocells. Each macrocell (see Figure 1) contains a PLA structure (programmable AND array product terms connected to an OR gate) and an I/O architecture control block (with a D Flip-Flop) that can be programmed to create many different output logic structures. This powerful I/O architecture can be configured to support both active-high, active-low, 3-state, open drain and bi-directional data ports all on a 4-bit wide basis. They can also act as inputs on a nibble wide basis with optional input latching.

Macrocells in each half of the circuit are grouped together for I/O architecture programming. Each bank of four macrocells can be further programmed on an individual macrocell basis to generate active high or active low outputs of the logic function from the PLA.

The primary logic array of the 5C121 is segmented into two symmetrical halves that communicate via global bus signals. The main array contains some 15104 programmable elements representing 236

product terms (AND gates) each containing 64 input signals.

The macrocells share a common programmable clock system (described in a later section) that controls clocking of all registers and input latches. The device contains 8 modes of clock operation that allow logic transition to take place on either rising or falling edges of the clock signals.

The device also contains four macrocells whose outputs are not tied to any I/O pin but feed back into the array to create buried state-functions. The feedback path may be either the registered or combinational result of the PLA output. The use of the buried state macrocells provides maximum equivalent logic density without demanding higher pin-count packages that consume valuable board space.

## MACROCELL I/O ARCHITECTURE

The Input/Output architecture of the 5C121 macrocell (see Figure 1) can be programmed using both static and dynamic controls. The static controls remain fixed after the device is programmed whereas the dynamic controls may change state as a result of the signals applied to the device.

The static controls set the inversion logic (i), register by-pass (ii) and input feedback multiplexers (iii). In the latter two cases these controls operate on four macrocells as a bank.

The buried-state registers have simpler controls that determine if the feedback is to be registered or combinational.

The inversion control logic, marked (i) in Figure 1, is achieved by programming the EPROM control bit connected to the same XOR gate as the output from the PLA structure. Programming or erasure of this EPROM element toggles the OR gate output of the PLA between active-high and active-low. The inversion control operates on an individual macrocell basis.

The register by-pass control, marked (ii) in Figure 1 allows the PLA output to either flow through the D Flip-Flop as a registered output or by-pass the Flip-Flop and be a combinational output.

The dynamic controls consist of a programmable input latch-enable as well as reset and output enable product terms. The latch-enable function is common throughout the 5C121 and once chosen, will latch all the inputs. This function is programmed by the clock control block but may also be driven by input signals applied to pin 1 (see clock modes—Table 1).

\*CHMOS is a patented process of Intel Corporation.

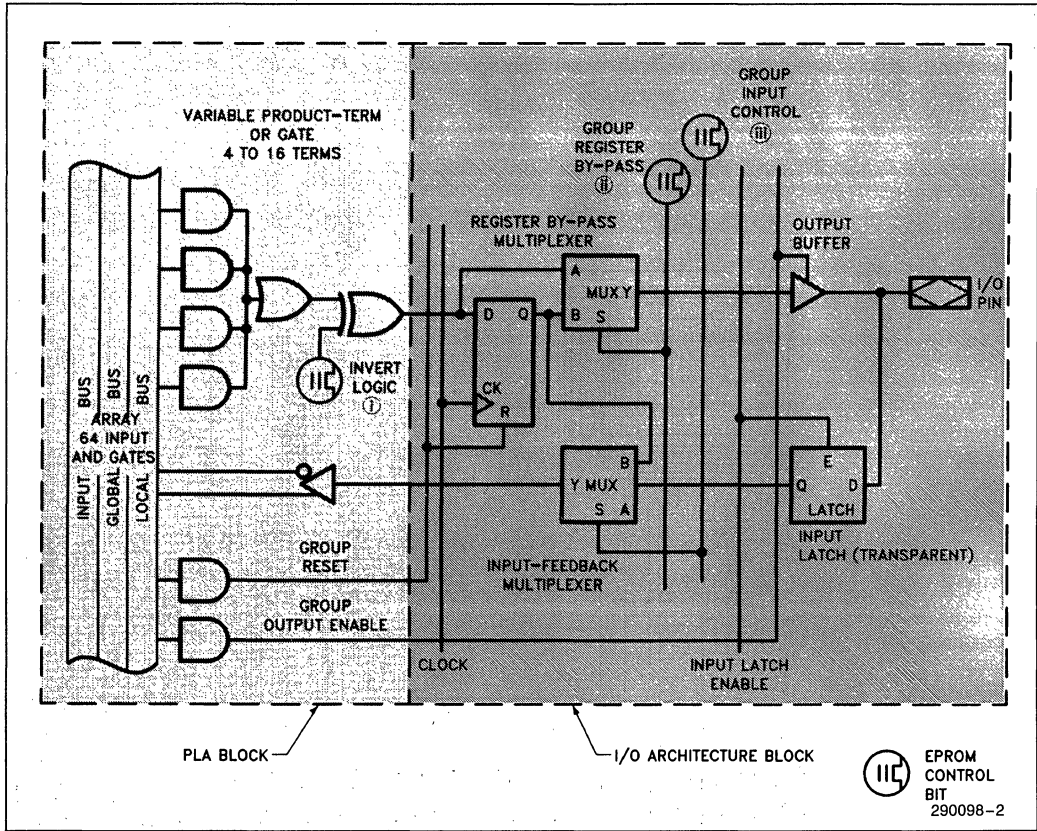


Figure 1. 5C121 Macrocell I/O Architecture

The reset and output-enable controls are logically controlled by single product terms (the logic AND of programmed variables in the array). These terms have control over banks of four macrocells.

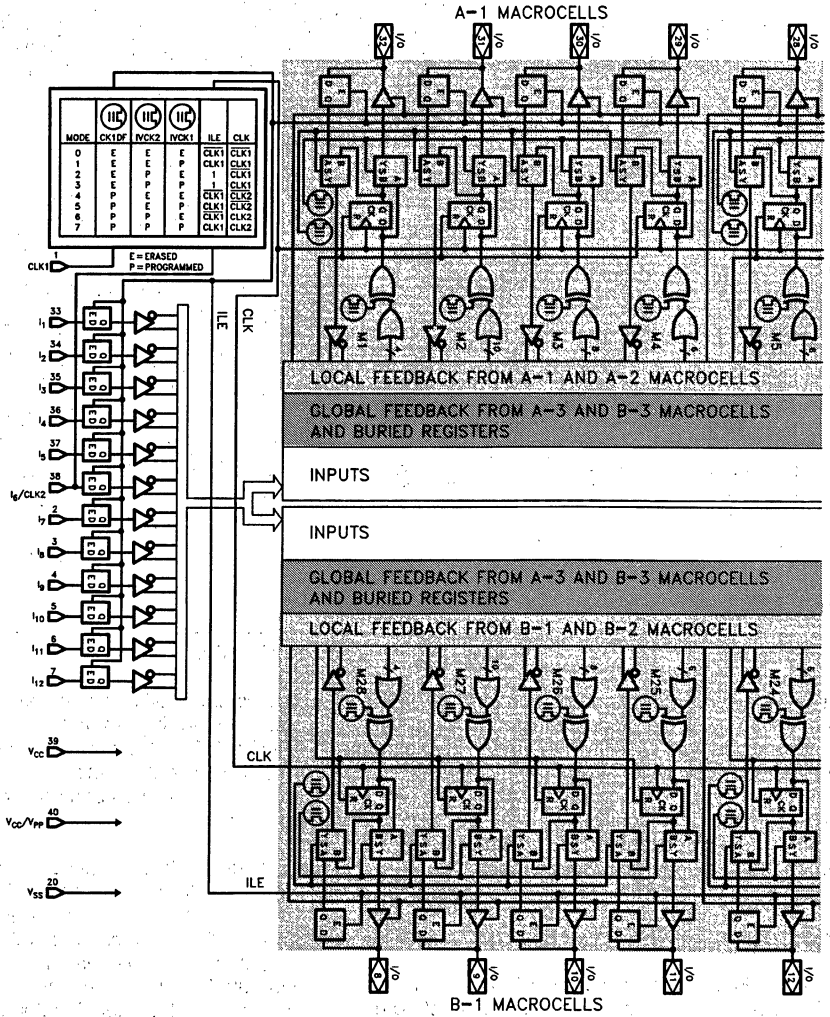
The output-enable control may be used to generate architecture types that include bi-directional, 3-state, open drain, or input only structures.

**INTERNAL BUS STRUCTURE**

The two identical halves of the 5C121 communicate via a series of busses. The local bus structure used

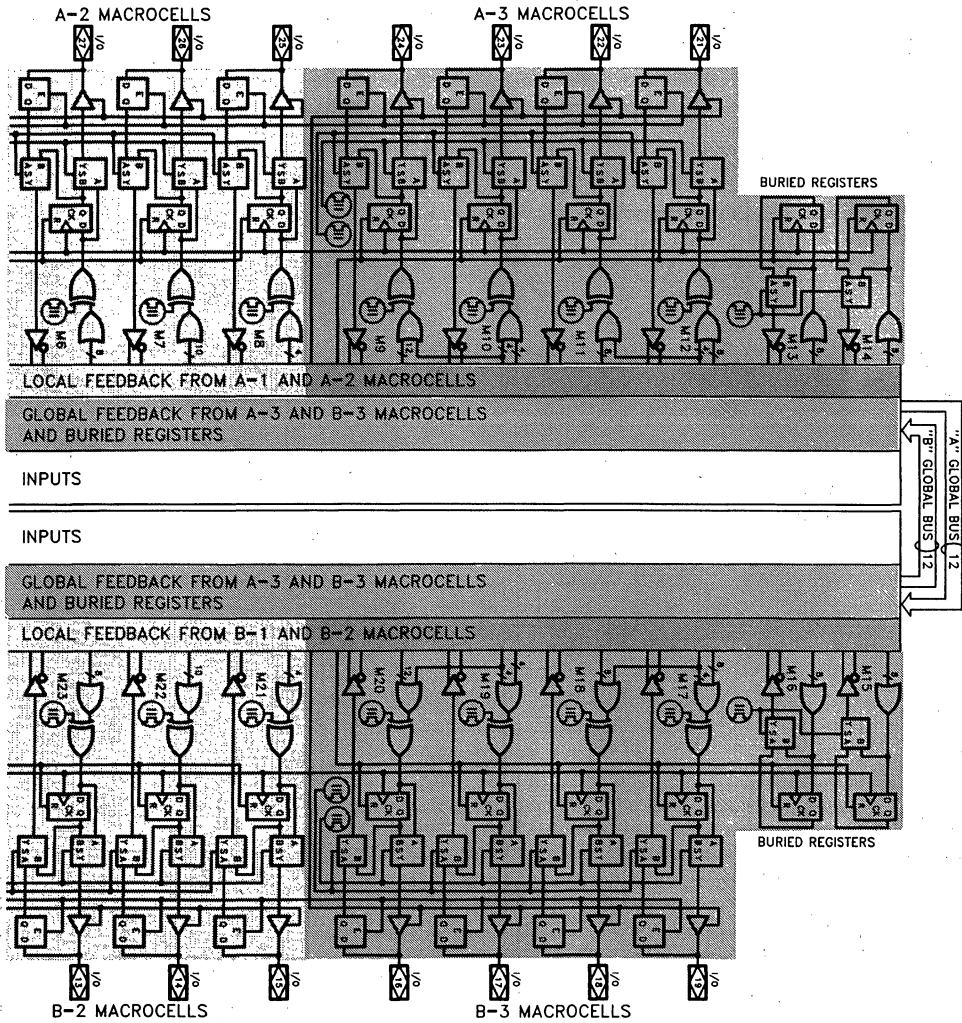
for communication within each half of the chip contains 16 conductors that carry the TRUE and COMPLEMENT of 8 local macrocells. In the block diagram (Figure 2) of the 5C121 the local macrocells are B-1 and B-2 on one half and A-1 and A-2 on the other half.

The global busses (Input bus & Global feedback from A-3 & B-3 macrocells & buried registers) are made up of 48 conductors that span the entire chip. These 48 conductors carry the TRUE and COMPLEMENT of the twelve primary inputs (pins 2 through 7 and 33 through 38), signals from 4 Buried Registers as well as the global outputs of 8 macrocells in groups A-3 and B-3.



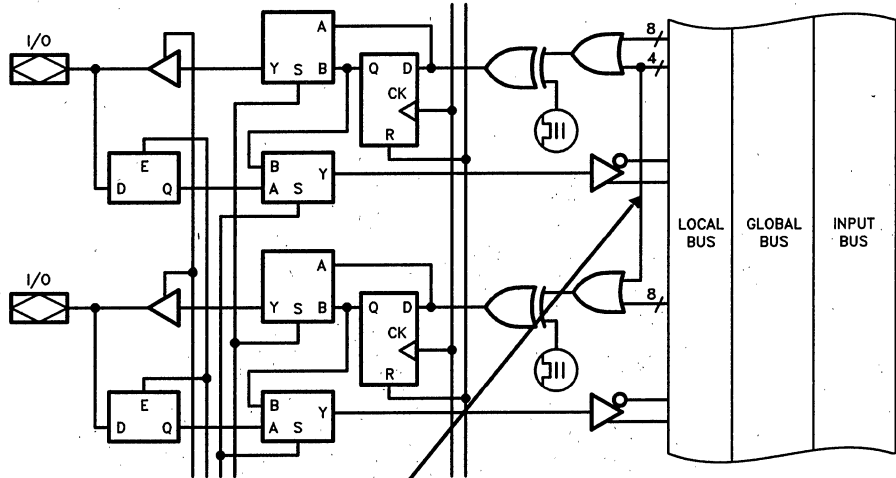
290098-3

Figure 2. 5C121 Block Diagram



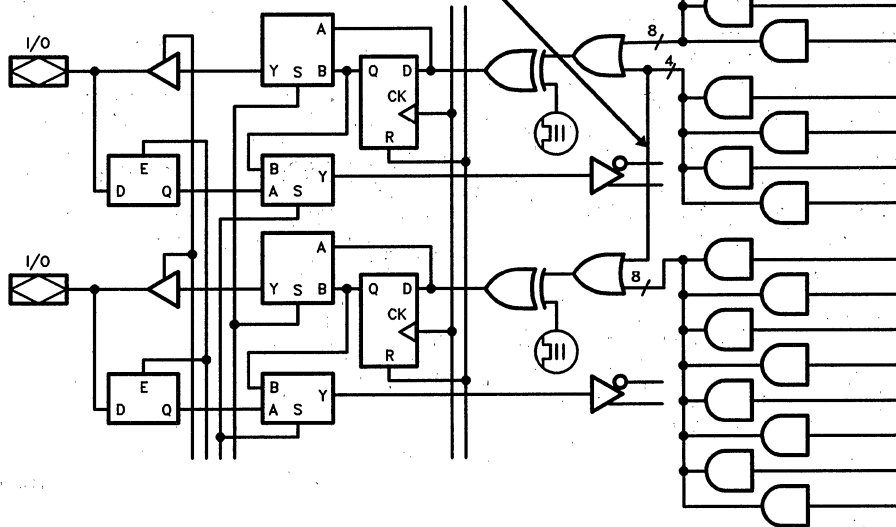
290098-4


Figure 2. 5C121 Block Diagram (Continued)



*In this illustration a small group of 4 product-terms is shared by groups containing 8 product-terms each. This feature is most useful in counter applications where common terms exist in the functions.*

**DETAILED CIRCUIT REPRESENTATION**



 = 64 INPUT AND GATE (ONE PRODUCT TERM)

**Figure 3. Shared Product-Term Circuits**

### SHARED PRODUCT TERMS

Macrocells 9 & 10, 11 & 12, 17 & 18 and 19 & 20 (in groups A-3 and B-3—the macrocells with global feedback) have the facility to share a total of 16 additional product terms. This sharing takes place between pairs of adjacent macrocells. This capability enables, for example, macrocells 9 and 10 to expand to 16 and 8 effective product terms respectively, and for macrocells 11 and 12 both to expand to 12 effective product terms. Figure 3 shows this sharing technique in detail. This facility is primarily of use in state machine and counter applications where common product terms are frequently required among output functions.

### MACROCELL-BUS INTERFACE

As discussed earlier, the macrocells within the 5C121 are interconnected to other macrocells and inputs to the device via three internal data busses.

The product terms span the entire bus structure (local feedback, global feedback and input busses) that

is adjacent to their macrocell (see Figure 4) so that they may produce a logical AND of any of the variables (or their complements) that are present on the busses.

All macrocells have the ability to return data to the local or the global bus. Feedback data may originate from the output of the macrocell or from the I/O pin. Feedback to the global bus communicates throughout the part. Macrocells that feedback to the local bus communicate only to their half of the 5C121. Connections to and from the signal busses are made with EPROM switches that provide the reprogrammable logic capability of the circuit.

Macrocells in groups A-3 and B-3 and the buried registers all have global bus connections while macrocells in groups A-1, A-2 and B-1, B-2 have only local bus connections (see Block Diagram, Figure 2). Advanced features of the Intel Programmable Logic Development System II will, if desired, automatically select an appropriate macrocell to meet both the logic requirements and the connection to an appropriate signal bus to achieve the interconnection to other macrocells.

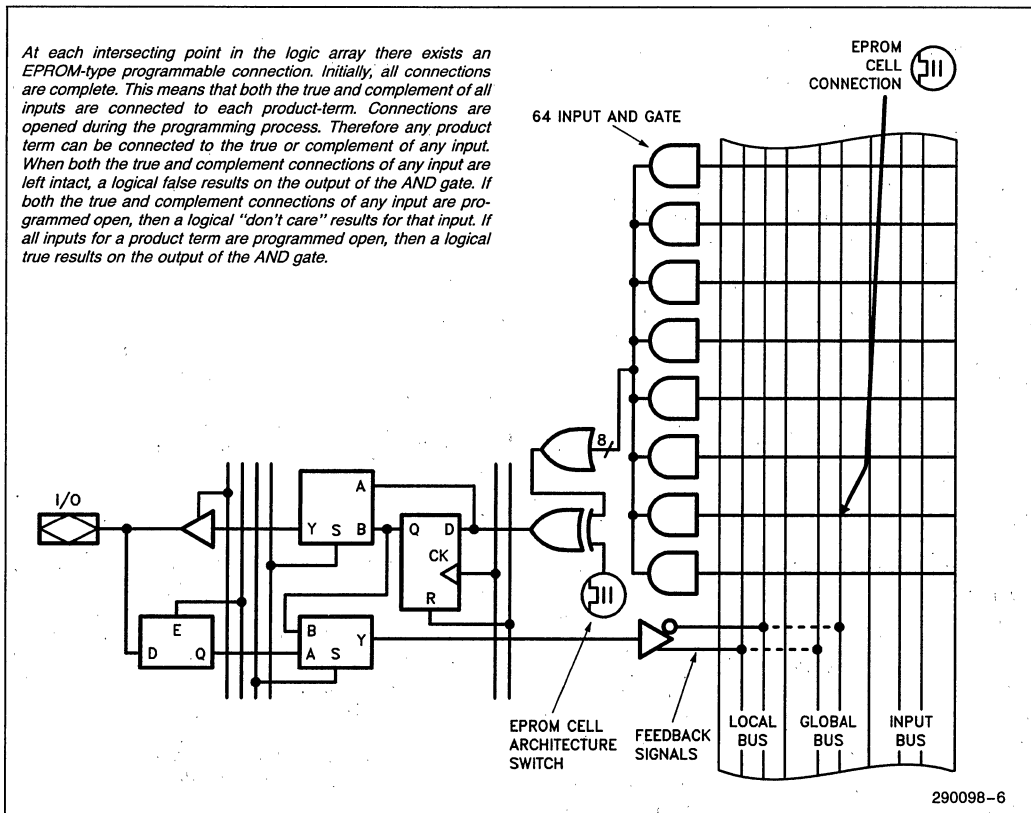


Figure 4. Macrocell-Bus Interface



## CLOCK MODE CONTROL

The 5C121 contains two internal clock data paths that drive the input latches (transparent 7475 type) and the output registers. These clocks may be programmed into one of 8 operating modes (see clock mode Table 1). Figure 1 shows a typical macrocell which is driven by the master clock signal CLK and the input latch-enable signal ILE.

The master clock signal is input via pin 1. If programmed modes 4, 5, 6 & 7 are chosen, a second clock signal is required which is input via pin 38 (see Figure 5). Table 1 shows the operation of each clock programming mode.

If modes 0, 1, 4, 5, 6 or 7 are chosen (i.e. latching of the inputs is required), all inputs, both dedicated and I/O, are latched with the same ILE signal. Data applied to the inputs when CLK1 is low (high) is latched when CLK1 goes high (low) and will stay latched as long as CLK1 stays high (low). Levels shown in parenthesis are for modes 1, 5 & 7 and levels shown outside parenthesis are for modes 0, 4 & 6.

Care is required when using any of the clock modes 4, 5, 6 or 7, that require two input clock signals to ensure that timing hazards are not created.

## ERASURE CHARACTERISTICS

Erasure characteristics of the 5C121 are such that erasure begins to occur upon exposure to light with wavelengths shorter than approximately 4000Å. It should be noted that sunlight and certain types of fluorescent lamps have wavelengths in the 3000–4000Å. Data shows that constant exposure to room level fluorescent lighting could erase the typical 5C121 in approximately three years, while it would take approximately one week to cause erasure when exposed to direct sunlight. If the 5C121 is to be exposed to these types of lighting conditions for extended periods of time, conductive opaque labels should be placed over the window to prevent unintentional erasure.

The recommended erasure procedure for the 5C121 is exposure to shortwave ultraviolet light which has the wavelength of 2537Å. The integrated dose (i.e., UV intensity  $\times$  exposure time) for erasure should be a minimum of fifteen (15) Wsec/cm<sup>2</sup>. The erasure time with this dosage is approximately 15 to 20 minutes using an ultraviolet lamp with a 12,000  $\mu$ W/cm<sup>2</sup> power rating. The 5C121 should be placed within one inch of the lamp tubes during erasure. The maximum integrated dose the 5C121 can be exposed to without damage is 7258 Wsec/cm<sup>2</sup> (1 week @ 12,000  $\mu$ W/cm<sup>2</sup>). Exposure to high intensity UV light for longer periods may cause permanent damage.

## PROGRAMMING CHARACTERISTICS

Initially, and after erasure, all the EPROM control bits of the 5C121 are connected (in the "1" state). Each of the connected control bits are selectively disconnected by programming the EPROM cell into their "0" state. Programming voltage and waveform specifications are available by request from Intel to support programming of the 5C121.

## intelligent Programming™ Algorithm

The 5C121 supports the intelligent Programming Algorithm which rapidly programs Intel H-ELPDs (and EPROMs) using an efficient and reliable method. The intelligent Programming Algorithm is particularly suited to the production programming environment. This method greatly decreases the overall programming time while programming reliability is ensured as the incremental program margin of each bit is continually monitored to determine when the bit has been successfully programmed.

## FUNCTIONAL TESTING

Since the logical operation of the 5C121 is controlled by EPROM elements, the device is completely factory tested. Each programmable EPROM bit controlling the internal logic including the buried state registers are tested using application-independent test program patterns. After testing, the devices are erased before shipment to customers. No post-programming tests of the EPROM array are necessary.















## DESIGN RECOMMENDATIONS

For proper operation it is recommended that input and output pins be constrained to the range  $GND < (V_{IN} \text{ or } V_{OUT}) < V_{CC}$ . Unused inputs should be tied to an appropriate logic level (e.g. either  $V_{CC}$  or  $GND$ ) to minimize device power consumption.

When utilizing a macrocell with an I/O pin connection as a buried macrocell (i.e. just using the macrocell for feedback purposes to other macrocells), its I/O pin is a 'reserved pin'. (The Intel Programmable Logic Development System II will label the pin 'RESERVED' in the utilization report that it generates.) Such an I/O pin will actually be an output pin and should not be grounded. It should be left unconnected such that it can go high or low depending on the state of the macrocell's output.

In normal operation  $V_{CC}/V_{PP}$  (pin 40) should be connected directly to  $V_{CC}$  (pin 39).

**Table 1. Clock Programming** (Key: L = Latched; T = Transparent)

Programmed Mode	Input Signals Are Latched When:	Output Registers Change State When:	Clock Configuration
0	CLK1 (Pin 1)  L T	CLK1 (Pin 1) 	1 Clock
1	CLK1 (Pin 1)  T L	CLK1 (Pin 1) 	1 Clock
2	Inputs Not Latched	CLK1 (Pin 1) 	1 Clock
3	Inputs Not Latched	CLK1 (Pin 1) 	1 Clock
4	CLK1 (Pin 1)  L T	CLK2 (Pin 38) 	2 Clocks
5	CLK1 (Pin 1)  T L	CLK2 (Pin 38) 	2 Clock
6	CLK1 (Pin 1)  L T	CLK2 (Pin 38) 	2 Clocks
7	CLK1 (Pin 1)  T L	CLK2 (Pin 38) 	2 Clocks

As with all CMOS devices, ESD handling procedures should be used with the 5C121 to prevent damage to the device during programming, assembly, and test.

**DESIGN SECURITY**

A single EPROM bit provides a programmable design security feature that controls the access to the data programmed into the device. If this bit is set, a proprietary design within the device cannot be copied. This EPROM security bit enables a higher degree of design security than fused-based devices since programmed data within EPROM cells is invisible even to microscopic evaluation. The EPROM security bit, along with all the other EPROM control bits, will be reset by erasing the device.

**AUTOMATIC STAND-BY MODE**

The 5C121 contains a programmable bit, the Turbo Bit, that optimizes operation for speed or for power savings. When the Turbo Bit is programmed (TURBO = ON), the device is optimized for maximum speed. When the Turbo Bit is not programmed (TURBO = OFF), the device is optimized for power savings by entering standby mode during periods of inactivity.

Figure 6 shows the device entering standby mode approximately 100 ns after the last input transition. When the next input transition is detected, the device returns to active mode. Wakeup time adds an additional 10 ns to the propagation delay through the device as measured from the first input. No delay will occur if an output is dependent on more than one input and the last of the inputs changes after the device has returned to active mode.

After erasure, the Turbo Bit is unprogrammed (OFF); automatic standby mode is enabled. When the Turbo Bit is programmed (ON), the device never enters standby mode.

**LATCH-UP IMMUNITY**

All of the input, I/O, and clock pins of the 5C121 have been designed to resist latch-up which is inherent in inferior CMOS structures. The 5C121 is designed with Intel's proprietary CHMOS II-E EPROM process. Thus, each of the 5C121 pins will not experience latch-up with currents up to 100 mA and voltages ranging from -1V to V<sub>CC</sub> + 1V. Furthermore, the programming pin is designed to resist latch-up to the 13.5V maximum device limit.

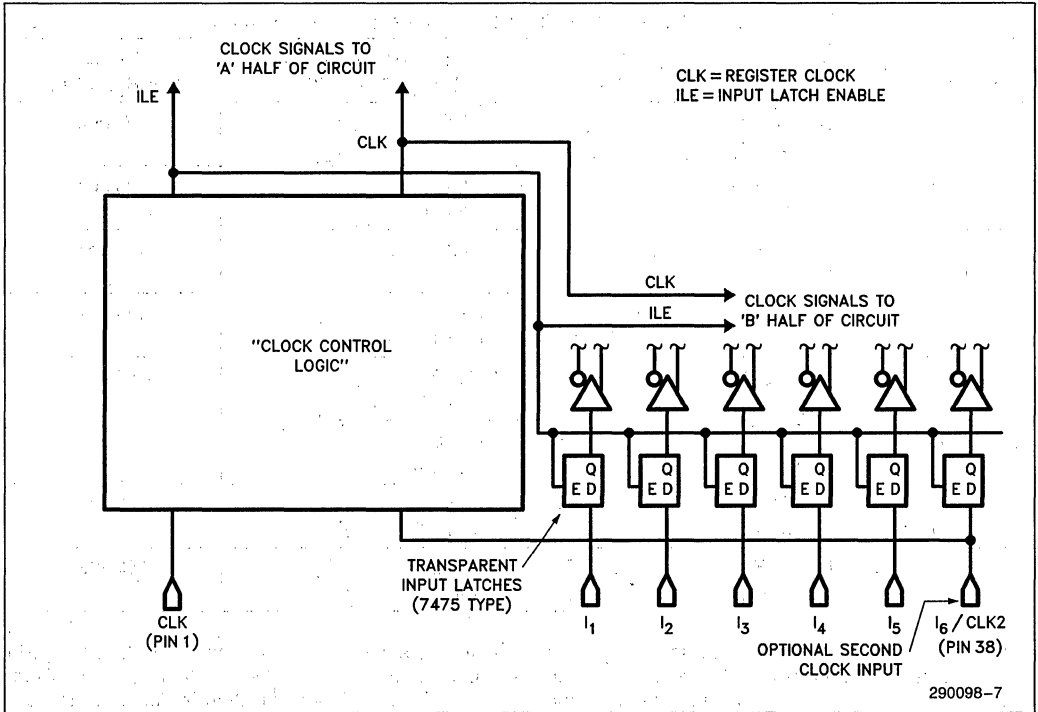


Figure 5. Programmable Clock Control System

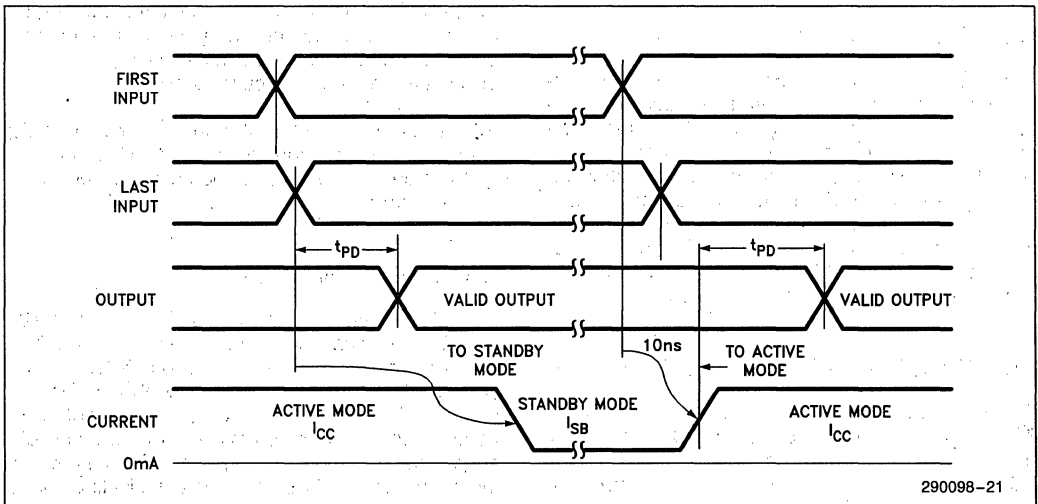


Figure 6. 5C121 Standby Mode and Active Mode Transitions

## Intel Programmable Logic Development System II (iPLDS II)

iPLDS II provides all the tools needed to design with Intel H-Series EPLDs or compatible devices. It contains comprehensive third generation software that supports four different design entry methods, minimizes logic, does automatic pin assignments and produces the best design fit for the selected EPLD. It is user friendly with guided menus, on-line Help messages and soft key inputs.

In addition, the iPLDS II contains programmer hardware in the form of an expansion card for the PC with programming software to enable the user to program EPLDs, read and verify programmed devices and also to graphically edit programming files. The software generates industry standard JEDEC object code output files which can be downloaded to other programmers as well.

The iPLDS II has interfaces to popular schematic capture packages to enable designs to be entered using schematics. An integrated schematic entry method is provided by SCHEMA II-PLD, a low-cost schematic capture package that supports EPLD primitives and user-defined macro symbols. SCHEMA II-PLD contains the EPLD Design Manager, which provides a single user interface to both SCHEMA II-PLD and iPLS II software. The other design entry formats supported are Boolean equation entry and State Machine design entry.

The iPLDS II runs on the IBM<sup>†</sup> PC, PC/XT or PC/AT and other compatible machines with the following configuration:

- (1) At least one floppy disk drive and hard disk drive
- (2) MS-DOS<sup>††</sup> Operating System Version 2.0 or later release

(3) 512K Memory (640K recommended)

(4) Intel iUP-PC Universal Programmer-Personal Computer and GUPI Adaptor (supplied with iPLDS II).

Detailed information on the Intel Programmable Logic Development System II is contained in a separate Intel data sheet (Order Number: 280168).

<sup>†</sup>IBM Personal Computer is a registered trademark of International Business Machine Corporation.

<sup>††</sup>MS-DOS is a registered trademark of Microsoft Corporation.

## ADF PRIMITIVES SUPPORTED

The following ADF primitives are supported by this device:

INP	RONF
LINP	RORF
CONF	ROIF
CORF	ROLF
COIF	NOCF
COLF	NORF

## ORDERING INFORMATION

t <sub>PD</sub> (ns)	t <sub>CO</sub> (ns)	f <sub>MAX</sub> (MHz)	Order Code	Package	Operating Range
55	32	25	D5C121-55	CERDIP	Commercial
65	33	20	D5C121-65	CERDIP	Commercial
90	38	16	D5C121-90	CERDIP	Commercial

**ABSOLUTE MAXIMUM RATINGS\***

Symbol	Parameter	Min	Max	Unit
$V_{CC}$	Supply Voltage <sup>(1)</sup>	-2.0	7.0	V
$V_{PP}$	Programming Supply Voltage <sup>(1)</sup>	-2.0	13.5	V
$V_I$	DC Input Voltage <sup>(1)(2)</sup>	-0.5	$V_{CC} + 0.5$	V
$I_{CC}$	DC $V_{CC}$ Current <sup>(4)</sup>		100	mA
$T_{stg}$	Storage Temperature	-65	+150	°C
$T_{amb}$	Ambient Temperature <sup>(3)</sup>	-10	+85	°C

**NOTES:**

1. Voltages with respect to ground.
2. Minimum DC input is -0.5V. During transitions, the inputs may undershoot to -2.0V or overshoot to 7.0V for periods less than 20 ns under no load conditions.
3. Under bias.
4. With outputs tristated.

**RECOMMENDED OPERATING CONDITIONS**

Symbol	Parameter	Min	Max	Units
$V_{CC}$	Supply Voltage	4.75	5.25	V
$V_I$	Input Voltage	0	$V_{CC}$	V
$V_O$	Output Voltage	0	$V_{CC}$	V
$T_A$	Operating Temperature	0	70	°C
$t_R$	Input Rise Time		500	ns
$t_F$	Input Fall Time		500	ns

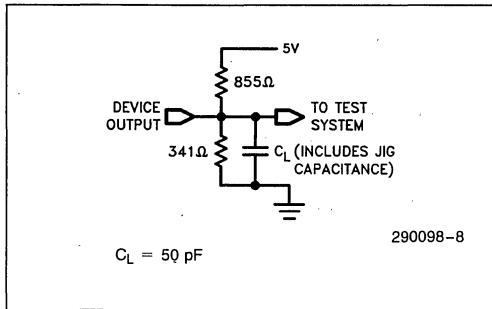
**D.C. CHARACTERISTICS**  $T_A = 0^\circ$  to  $70^\circ\text{C}$ ,  $V_{CC} = 5.0\text{V} \pm 5\%$ 

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
$V_{IH}$	HIGH Level Input Voltage		2.0		$V_{CC} + 0.3$	V
$V_{IL}$	LOW Level Input Voltage		-0.3		0.8	V
$V_{OH}$	HIGH Level Output Voltage	$I_O = -4.0$ mA DC	2.4			V
$V_{OL}$	LOW Level Output Voltage	$I_O = 4.0$ mA DC			0.45	V
$I_I$	Input Leakage Current	$V_I = V_{CC}$ or GND			$\pm 10.0$	$\mu\text{A}$
$I_{OZ}$	3-State Output Off-State Current	$V_O = V_{CC}$ or GND			$\pm 10.0$	$\mu\text{A}$
$I_{OS}$	Output Short Circuit Current	(Note 5)			130	mA
$I_{SB}$	$V_{CC}$ Supply Current (Standby) (Note 6)	$V_I = V_{CC}$ or GND $I_O = 0$			3	mA
					30	
$I_{CC}$	$V_{CC}$ Supply Current (Active)	No Load $f = 10$ MHz			50	mA
					100	

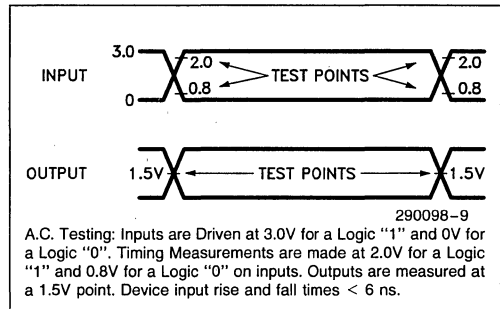
**NOTES:**

5. Output shorted for no more than 1 sec. and no more than one output shorted at a time.  $I_{OS}$  is sampled but not 100% tested.
6. Chip automatically goes into standby mode if logic transitions do not occur. (Approximately 100 ns after last transition.)

**A.C. TESTING LOAD CIRCUIT**



**A.C. TESTING INPUT, OUTPUT WAVEFORM**



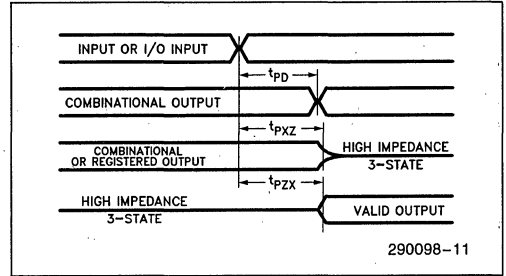
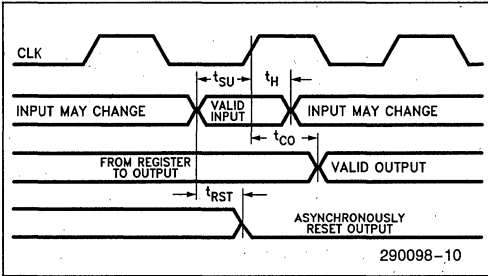
**A.C. CHARACTERISTICS**  $T_A = 0^\circ \text{ to } 70^\circ\text{C}$ ,  $V_{CC} = 5.0V \pm 5\%$

Symbol	Parameter	Device	5C121-55 EP1210-1		5C121-65 EP1210-2		5C121-90 EP1210		Unit
		Conditions	Min	Max	Min	Max	Min	Max	
$t_{PD}$	Non-Registered Input or I/O Input to Non-Registered Output			55	65		90	ns	
$t_{PZX}^{(7)}$	Non-Registered Input or I/O Input to Output Enable	$C_L = 30 \text{ pF}$		50	65		90	ns	
$t_{PXZ}^{(7)}$	Non-Registered Input or I/O Input to Output Disable			50	65		90	ns	
$t_{SU}$	Non-Registered Input or I/O Input to Output Register Setup		40		47		62	ns	
$t_H$	Non-Registered Input or I/O Input to Output Register Hold		0		0		0	ns	
$t_{CH}$	Clock High Time		20		25		30	ns	
$t_{CL}$	Clock Low Time	$C_L = 30 \text{ pF}$	20		25		30	ns	
$t_{CO}$	Clock to Output Delay			32		33		38	ns
$t_{CNT}$	Minimum Clock Period (Register Output Feedback to Register Input—Internal Path)		50		55		75	ns	
$f_{CNT}$	Maximum Frequency ( $1/t_{CNT}$ )			20.0		18.0		13.0	MHz
$f_{MAX}$	Maximum Frequency ( $1/t_{SU}$ )—Pipelined			25.0		21.2		16.1	MHz
$t_{RST}$	Asynchronous Reset Time			50		65		90	ns
$t_{ILS}$	Set Up Time for Latching Inputs		0		0		0	ns	
$t_{ILH}$	Hold Time for Latching Inputs		15		20		25	ns	
$t_{C1C2}$	Minimum Clock 1 to Clock 2 Delay		40		50		65	ns	
$t_{LDFS}$	Input Latch to D-FF Setup Time	Mode 0, 1	40		50		65	ns	
$t_{DFILS}$	D-FF to Input Latch Setup Time		25		30		35	ns	
$t_{P3}$	Minimum Period for a 2-Clock System ( $T_{C1C2} + t_{CO1}$ )		72		83		103	ns	
$f_3$	Maximum Frequency ( $1/t_{P3}$ )			13.8		12.0		9.7	MHz

**NOTE:**

7.  $t_{PZX}$  and  $t_{PXZ}$  are measured at  $\pm 0.5V$  from steady state voltage as driven by spec. output load.  $t_{PXZ}$  is measured with  $C_L = 5 \text{ pF}$ .

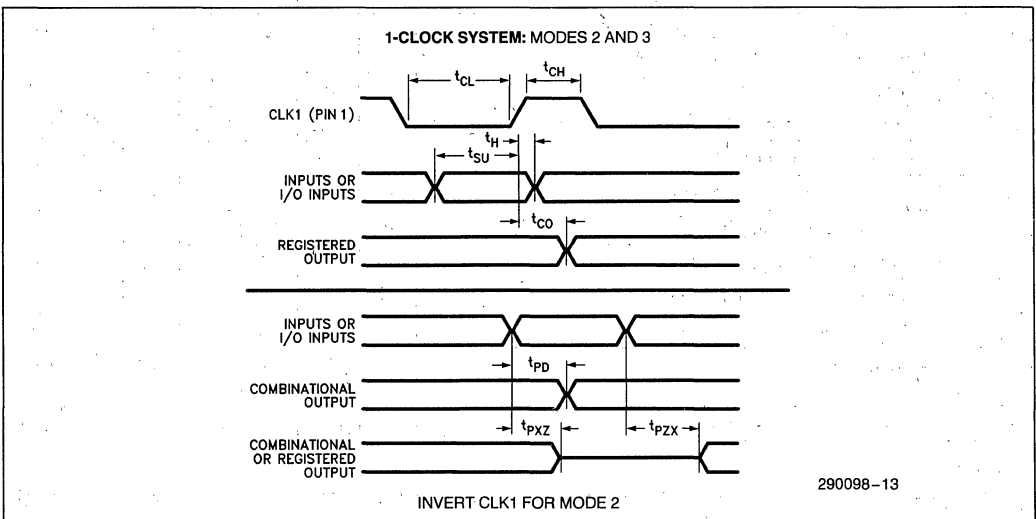
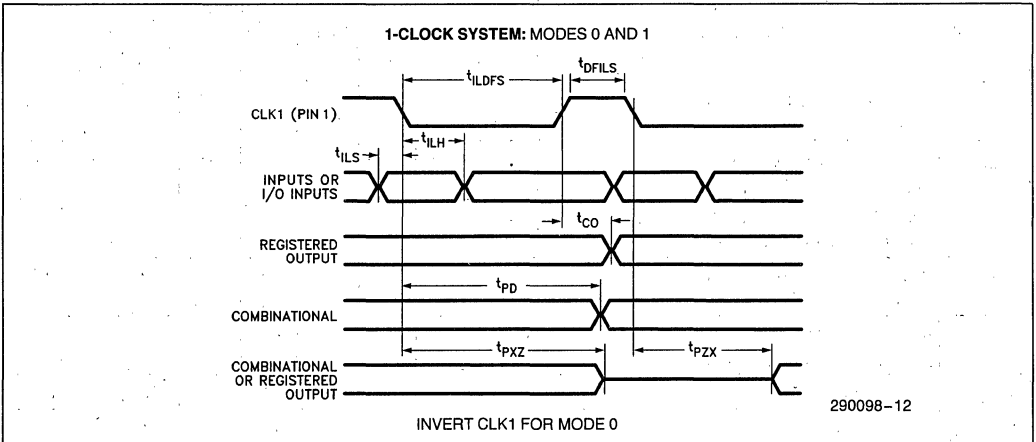
**SWITCHING WAVEFORMS**



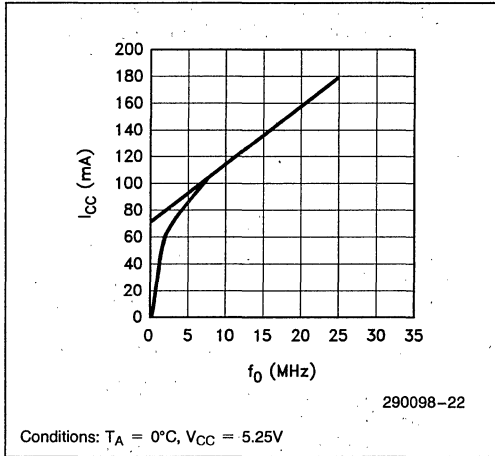
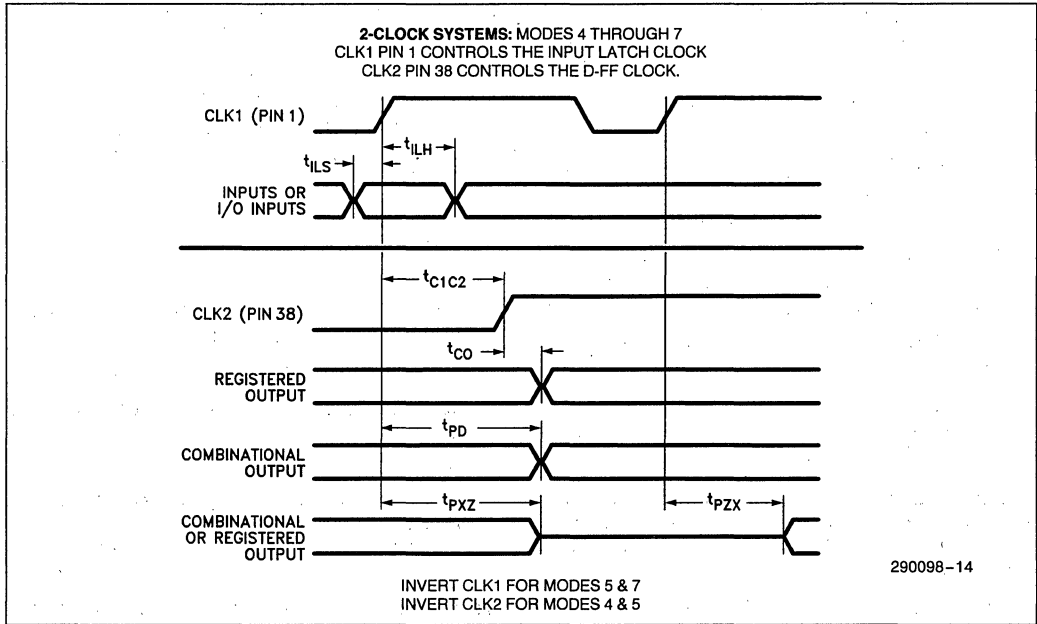
**NOTE:**

Above waveforms shown for clock modes 2 or 3 ( $t_{SU}$  &  $t_H$  are as in modes 2 & 3; no ILE signal is used).

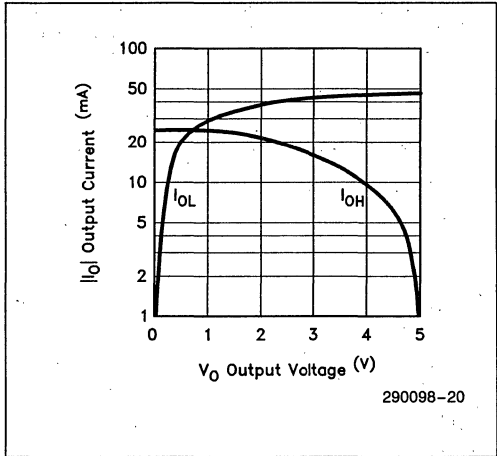
**CLOCK MODES SWITCHING WAVEFORMS**



**CLOCK MODES  
SWITCHING WAVEFORMS** (Continued)



**5C121 Current in Relation to Frequency**



**Output Drive Current in Relation to Voltage**



# 5C180 1800-GATE CHMOS ERASABLE PROGRAMMABLE LOGIC DEVICE

- High Performance LSI Semicustom Logic Replacement for TTL and 74HC SSI and MSI Logic
- CHMOS EPROM Technology-Based UV Erasable
- 48 Macrocells with Programmable I/O Architecture; up to 64 Inputs (16 Dedicated, 48 I/O) or 48 Outputs
- High Speed  $t_{PD}$  (max) 70 ns, Operating Frequency (max) 20.8 MHz (Pipelined), 16.1 MHz (w/Feedback)
- Low Power; 100  $\mu$ W Typical Standby Dissipation
- Programmable "Security Bit" Allows Total Protection of Proprietary Designs
- Dual Feedback Signals Allowing I/O Pins to Be Used for Buried Logic and Dedicated Input
- Programmable Clock System with Four Synchronous Clocks as well as Asynchronous Clocking Option on All Registers
- Programmable Registers. Can Be Configured as D, T, SR or JK Types with Individual Reset Controls
- Register Pre-Load and Erasable Array for 100% Generic Testability
- 100% Compatible with EP1800
- 68-Pin J-Lead Chip Carrier and Pin Grid Array Packages

(See packaging spec., Order #231369)

The Intel 5C180 EPLD (Erasable Programmable Logic Device) is a CHMOS LSI Logic Device capable of integrating 1800 to over 2000 equivalent gates of SSI/MSI logic. This user customizable Logic Device is available in a 68-pin J-Leaded chip carrier or Pin Grid Array package and has the benefits of low power and increased flexibility.

The 5C180 EPLD uses CHMOS EPROM (floating gate) cells as logic control elements instead of fuses. Use of Intel's advanced CHMOS II-E EPROM process technology enables greater logic densities to be achieved with superior speed and power performance. The EPROM technology also enables these devices to be 100% factory tested by the programming and the erasure of all the EPROM logic control elements in the device.

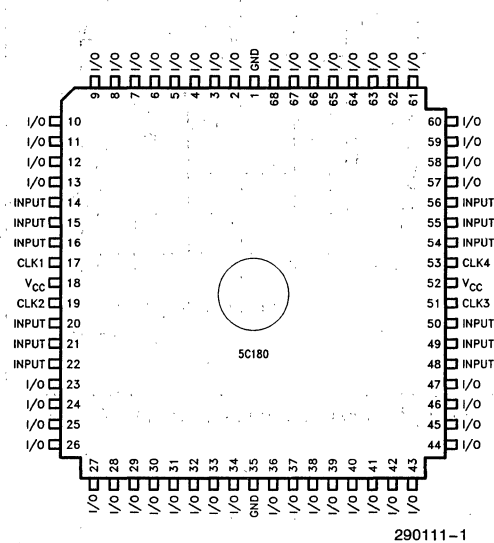


Figure 1. Pin Configuration

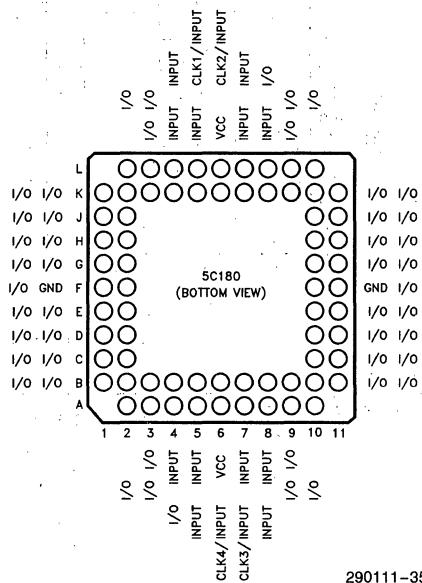


Figure 2. PGA Pin Configuration

The architecture of the 5C180 is based on the "Sum of Products" PLA (Programmable Logic Array) structure with a programmable AND array feeding into a fixed OR array. The 48 macrocells of the 5C180 can be partitioned into 4 identical quadrants each containing 12 macrocells. This device makes use of a segmented PLA structure with local and global bus structures to provide for increased performance and greater device utilization. The 5C180 has unique architectural features that allow programming of all 48 registers to D, T, SR or JK configurations without sacrificing product terms. These registers can be either clocked asynchronously or in banks with four synchronous clocks. In addition, the 16 global macrocells have two independent feedback paths to the array that allow for buried logic implementation together with use of the I/O pin for input functions.

## ARCHITECTURE DESCRIPTION

Externally, the 5C180 provides 12 dedicated data inputs, 4 synchronous clock inputs, and 48 I/O pins which may be individually programmed for input, output, or bi-directional operation.

The Block Diagram is shown in Figure 2 with pin numbers for the JLCC package. Figure 3 shows the device block diagram with pin numbers for the PGA package. The internal architecture is organized in familiar sum-of-products (AND-OR) structure. The 5C180 houses a total of 480 product terms distributed among 48 Macrocells. The basic Macrocell structure is shown in Figure 4. Input and feedback signals are selectively connected to product terms via EPROM cells. The output of the AND array feeds a fixed OR gate to produce sum-of-products logic. The final output may be combinatorial or registered, programmed active high or low. Combinatorial, registered, or pin feedback is also user-defined.

The 5C180 is partitioned into 4 identical quadrants. Each quadrant contains 12 Macrocells. Input signals to the Macrocells come from the 5C180 Local and Global bus structures. These two buses comprise an 88-input AND array for each quadrant. The output of each Macrocell feeds an I/O Architecture Control Block which contains output and feedback selection.

Four dedicated clock inputs provide synchronous clock signals to the 5C180 internal registers. There is one synchronous clock per quadrant. Therefore each clock signal controls a bank of 12 registers. CLK1 may be connected to registers in Macrocells 1-12, CLK2 with Macrocells 13-24, CLK3 with Macrocells 25-36, and CLK4 with Macrocells 37-48. With synchronous clocks, the flip-flops are positive edge triggered. Both true and complement signals for each dedicated clock input may also be used

within the AND array. All 48 internal registers may be individually programmed for synchronous or asynchronous clocking. Asynchronous clocking is possible via a Macrocell product term. Clock inputs not used for synchronous clock signals may be used as global bus inputs.

## Invert Select EPROM Bit

The Invert Select EPROM bit is used to invert the product term input into the register. This applies to all inputs including double inputs on JK and SR registers. The invert option allows the highest possible logic utilization by use of deMorgan logic inversion.

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

## BUS STRUCTURE

Input and feedback signals are connected to each 5C180 Macrocell via a Local and Global Bus. Figure 5 shows the Macrocell-Bus interface for Quadrant D. The Global Bus contains 64 input signals while the Local Bus has 24.

Within the 5C180 Macrocell, the product-terms share the entire bus structure. Therefore, a logical AND of any of the variables (or their complements) that is present on the buses may be produced by each product term.

All quadrants share the same Global Bus. Inputs to the bus come from the true and complement signals of the 12 dedicated data inputs, 4 clock inputs, and the 16 Global Macrocell pin feedback signals.

Each quadrant has its own Local Bus. Inputs to this bus come from the 12 quadrant Macrocells. For the eight Local Macrocells, the signals can be either from the Macrocell internal logic or from the pin. For the four Global Macrocells, the signals come from the Macrocell internal logic only.

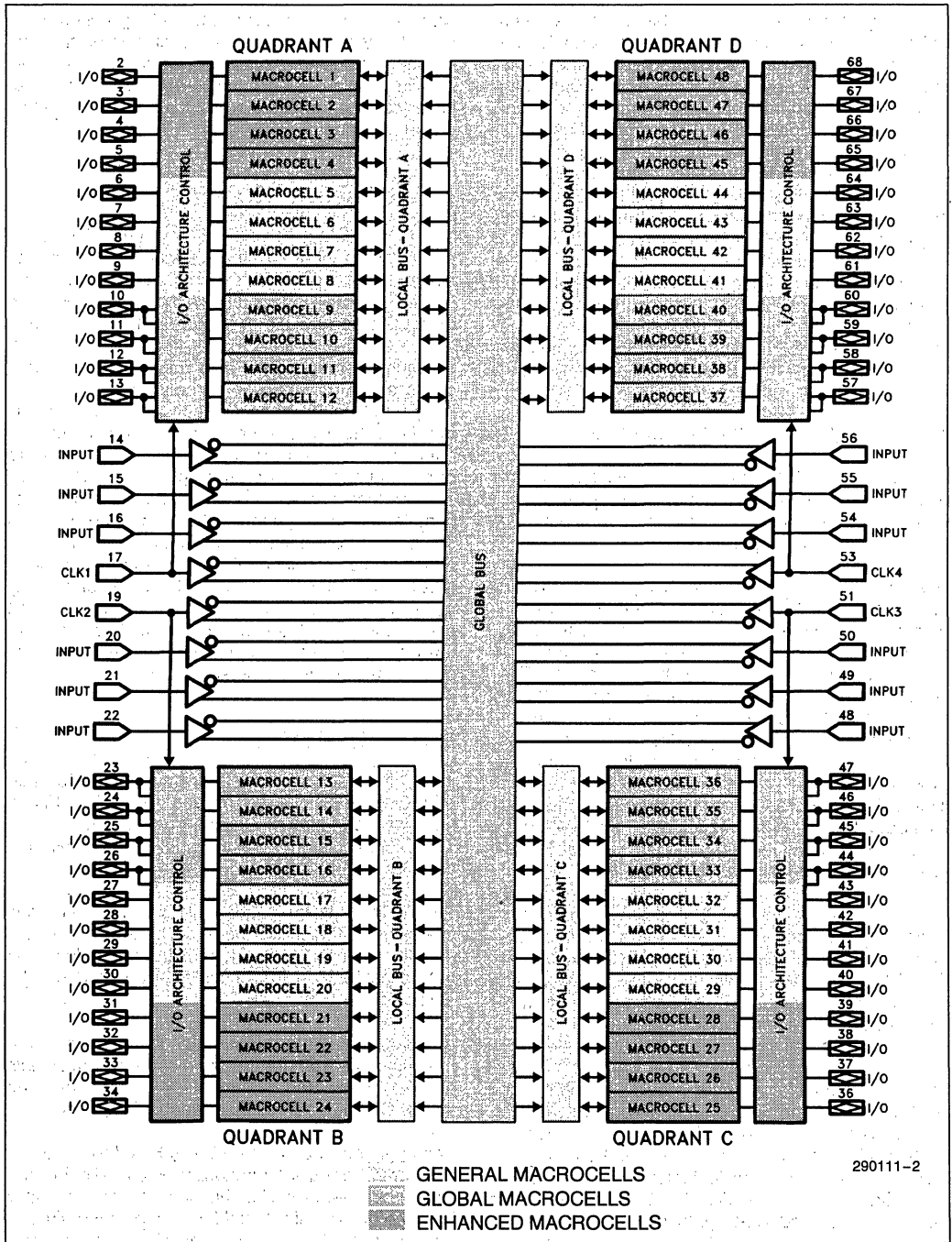


Figure 2. 5C180 Block Diagram—JLCC Package

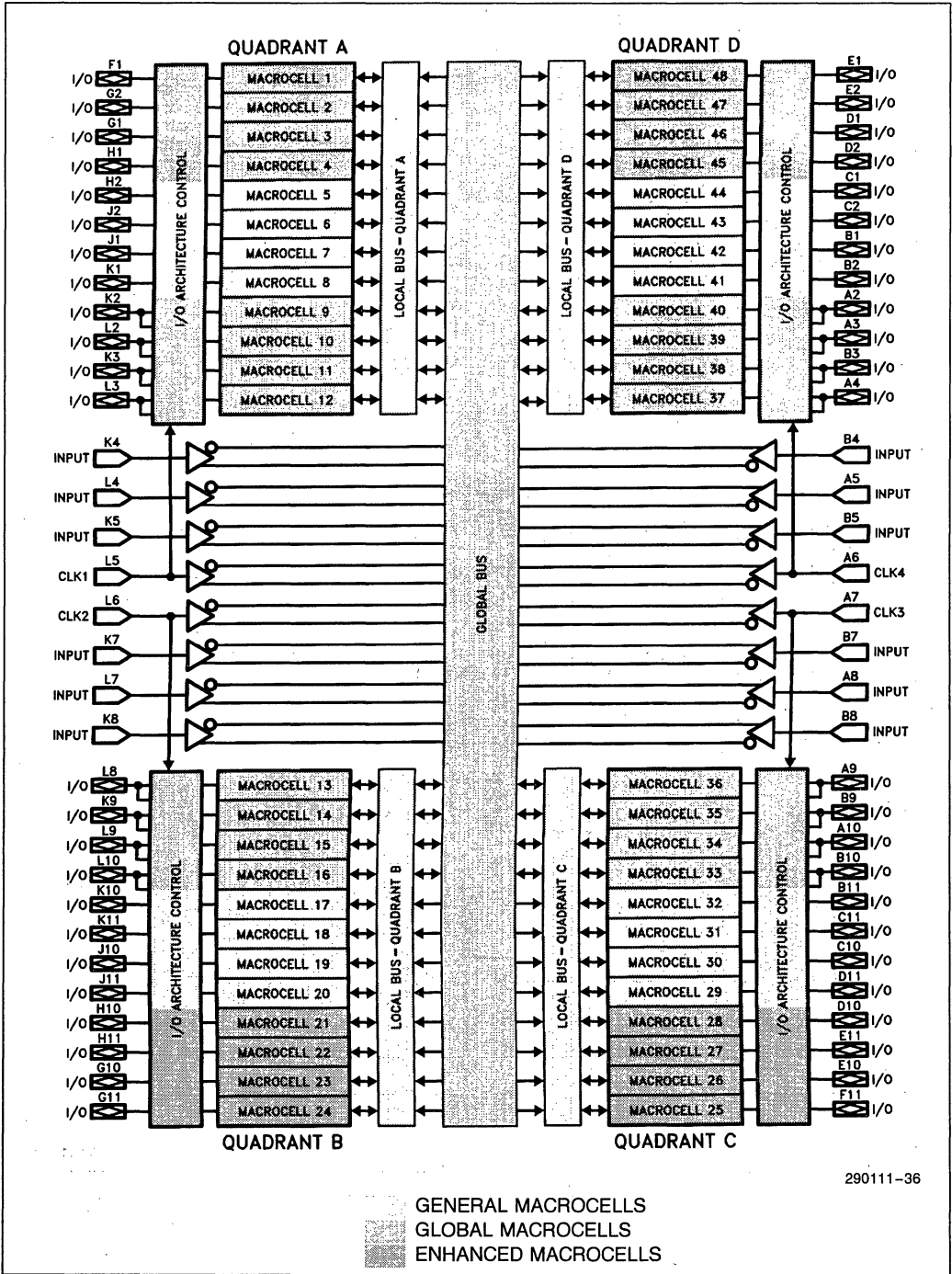


Figure 3. 5C180 Block Diagram—PGA Package

Table 1 summarizes the Macrocell interconnect.

Table 1. Macrocell Interconnect

	Pin #	Macro-cell #	Feedback Structure	Feedback Interconnect
Quad A	2-9 10-13	1-8 9-12	Local Local Global	Quad A Quad A All
Quad B	23-26 27-34	13-16 17-24	Local Global Local	Quad B All Quad B
Quad C	36-43 44-47	25-32 33-36	Local Local Global	Quad C Quad C All
Quad D	57-60 61-68	37-40 41-48	Local Global Local	Quad D All Quad D

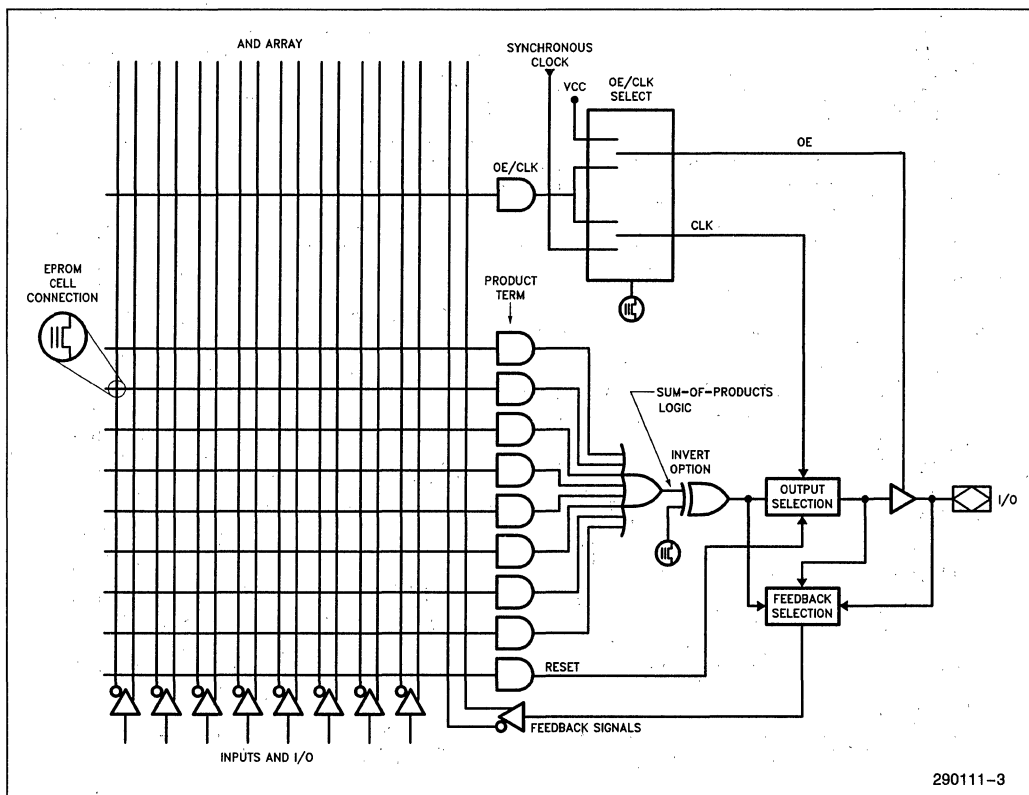
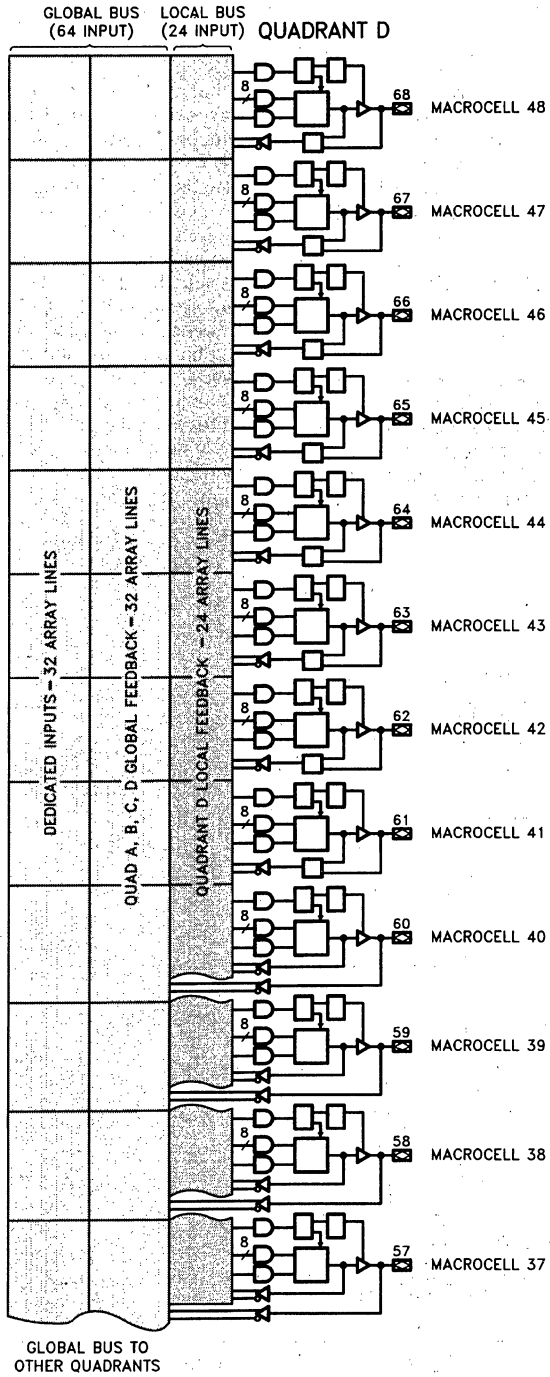


Figure 4. Basic Macrocell Architecture of the 5C180



290111-4

Figure 5. Quadrant "D" Bus Interface

**5C180 MACROCELLS**

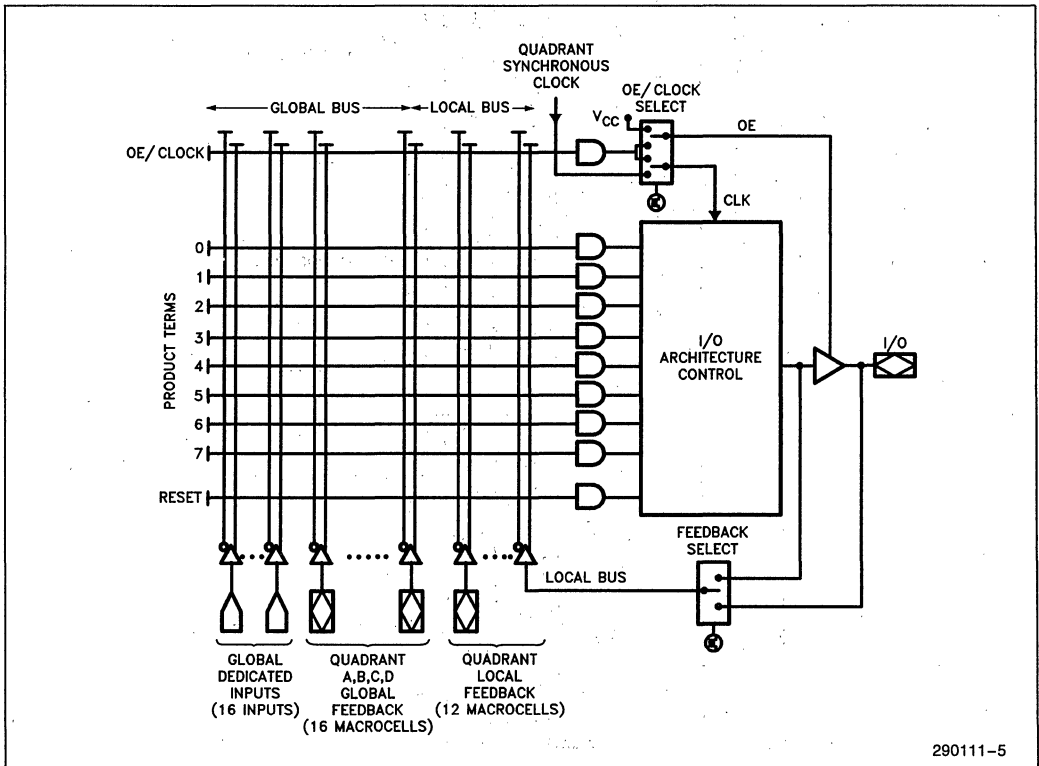
Within each 5C180 quadrant there are two different types of Macrocells; Local Macrocells, Figure 6, and Global Macrocells, Figure 7. Both types share an 88-input AND array and contain a total of ten product terms. Eight product terms are dedicated for logic implementation. One product term is reserved for Asynchronous Clear to the Macrocell register. The remaining product term is used for Output Enable/Asynchronous Clock implementation. Each 5C180 product term represents an 88-input AND gate. The I/O Architecture Control Block provides each Macrocell with both combinatorial and registered I/O configurations.

Local Macrocells provide one feedback path into the AND array. Combinatorial, registered or pin feedback may be selected from the Feedback Select Multiplexer. The selected feedback signal is then routed to the quadrant local bus. Therefore, the Local Macrocell feedback communicates only to Mac-

rocells within the same quadrant. There are a total of 32 Local Macrocells within the 5C180, with eight per quadrant.

Local macrocells are divided into two groups: General Macrocells and Enhanced Macrocells. The Enhanced Macrocells are architecturally identical to the General Macrocells but operate at higher speeds. These speed differences are reflected in the specification tables.

Global Macrocells contain two independent feedback paths to the AND array. Combinatorial or registered feedback is supplied to the local bus and pin feedback is supplied to the global bus. The "dual feedback" capability allows the Macrocell to be used for internal logic functions as well as a dedicated input pin. To obtain this configuration, the output buffer must be disabled. If the Global Macrocell I/O pin is not being used as a dedicated input, the Macrocell logic may be fed back along the global bus allowing routing to any of the 5C180's 48 Macrocells. There are 16 Global Macrocells contained in the 5C180, four per quadrant.



**Figure 6. Local Macrocell Logic Array**

290111-5

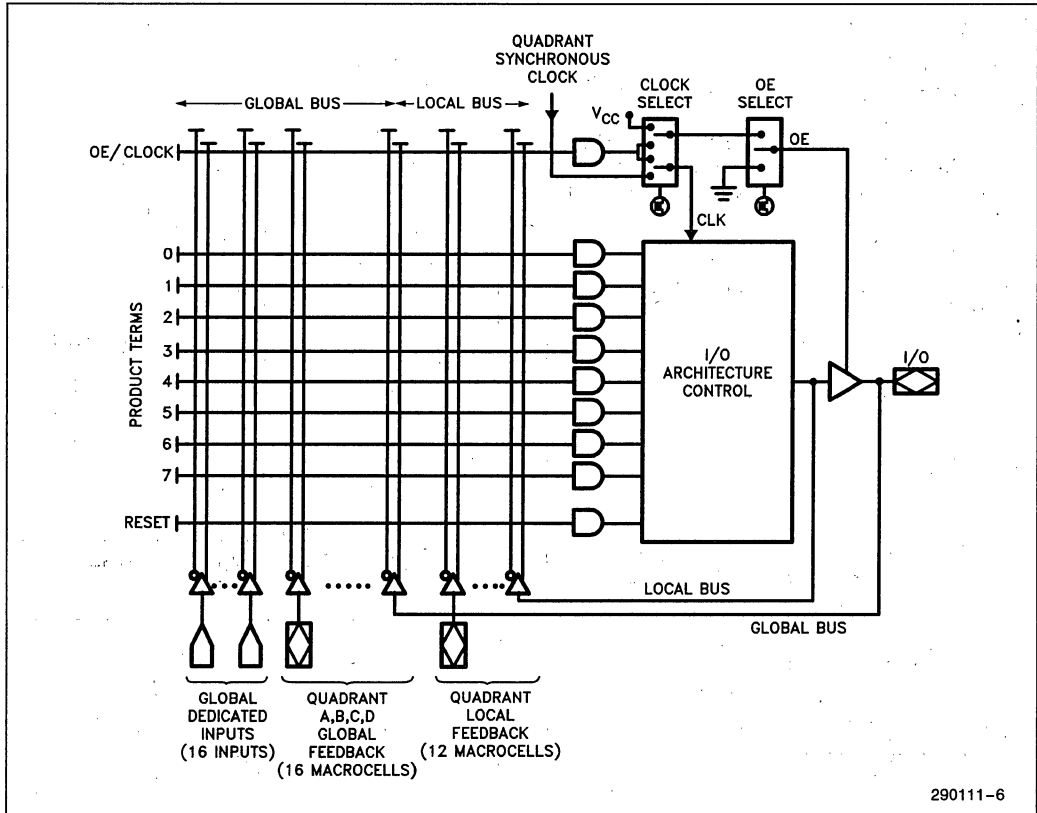


Figure 7. Global Macrocell Logic Array

**MACROCELL LOGIC CONFIGURATIONS**

**Combinatorial Selection**

In the Combinatorial configuration, eight product terms are ORed together to generate the output signal. The Invert Select EPROM bit controls output polarity and the Output Enable buffer is product-term controlled. The Feedback Select allows the user to choose combinatorial, I/O (pin) or no feedback to the respective local and global buses.

**REGISTER SELECTION**

The advanced I/O architecture of the 5C180 allows four different register types along with combinatorial output as illustrated in Figures 8a-8e. The register types include a T, D, JK, or SR Flip-Flop and each Macrocell I/O structure may be independently configured. In addition, all registers have an individual asynchronous RESET control from a dedicated

product term derived in the AND array. When this dedicated product term is a logical one, the Macrocell register is immediately cleared to a logical zero independent of the register clock. The RESET function occurs automatically on power-up.

The four different register types shown in Figures 8b-8e are described below:

**D- or T-type Flip-Flops**

When either a D- or T-type Flip-Flop is configured as part of the I/O structure, all eight of the product terms into the Macrocell are ORed together and fed into the register input.

**JK or SR Registers**

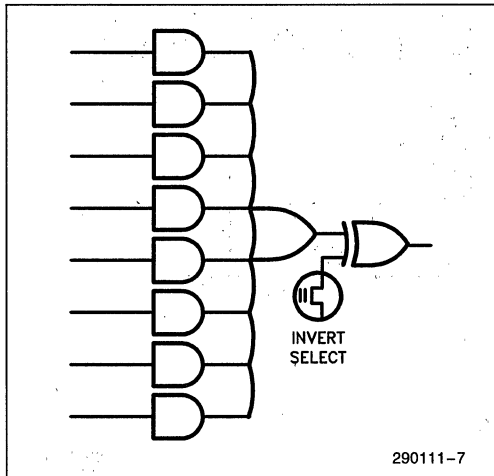
When either a JK or SR register is configured, the eight product terms are shared among two OR gates (one for the J or S input and the other for the K or R input). The allocation for these product terms for each of the register inputs is optimized by the iPLDS II development software.



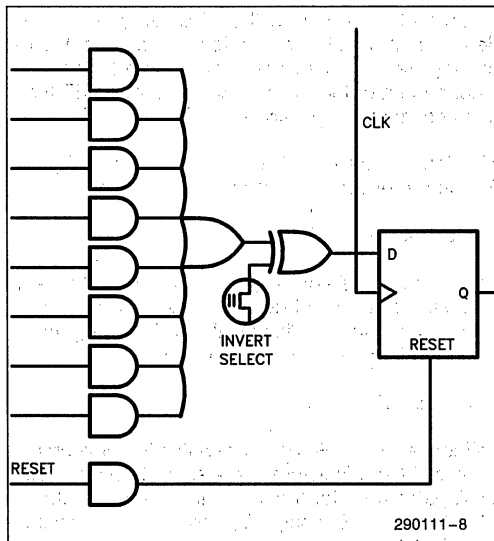
**Buried Logic Selection**

For Global Macrocells, if no output is selected, the logic may be "buried" and the I/O pin can be used as an additional dedicated input. The use of "dual feedback" is accomplished by tri-stating the Output Enable Buffer. Thus, up to 16 additional dedicated inputs may be added without sacrificing the Macrocell internal logic.

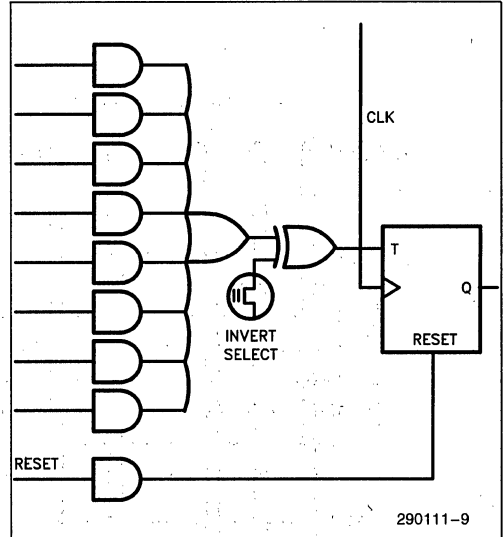
In the erased state, the I/O architecture is configured for combinatorial active low output with I/O (pin) feedback.



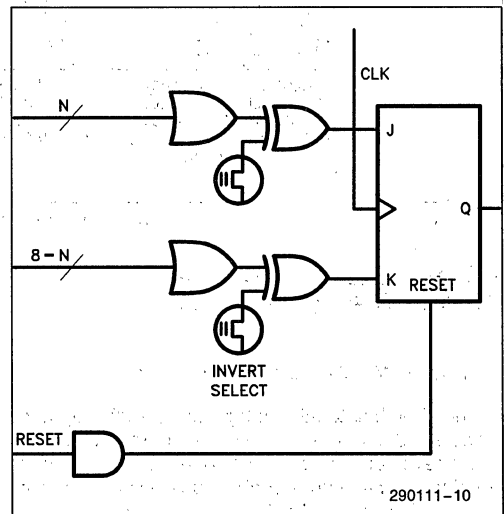
**Figure 8a. Combinatorial I/O Configuration**



**Figure 8b. D-Type Flip-Flop Register Configuration**



**Figure 8c. Toggle Flip-Flop Register Configuration**



**Figure 8d. JK Flip-Flop Register Configuration**

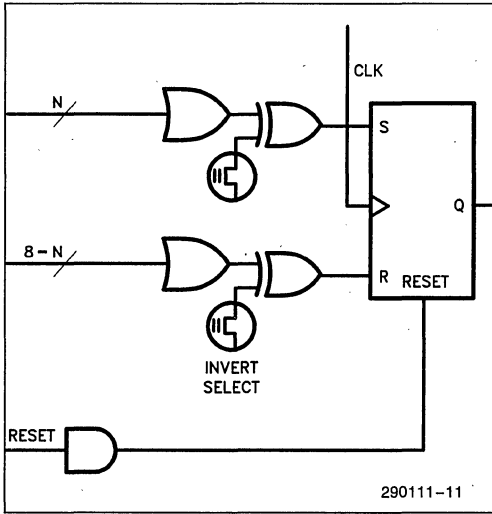


Figure 8e. SR Flip-Flop Register Configuration

**MACROCELL OE/CLK SELECT**

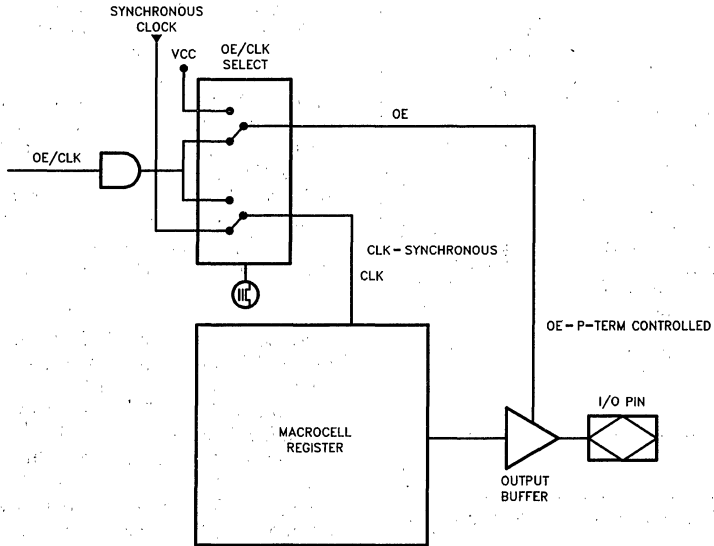
Each 5C180 register may be clocked synchronously or asynchronously. Figure 9a and 9b shows the modes of operation provided by the OE/CLK Select Multiplexers for both Local and Global Macrocells.

The operation of each multiplexer is controlled by EPROM bits and may be individually configured for each 5C180 Macrocell.

In Mode 0, the three-state output buffer is controlled by a single product term. If the output of the AND gate is a logical true then the output buffer is enabled. If a logical false resides on the output of the AND gate then the output buffer is seen as high impedance. In this mode the Macrocell flip-flop may be clocked by its quadrant synchronous clock input. In the erased state, the 5C180 is configured as Mode 0.

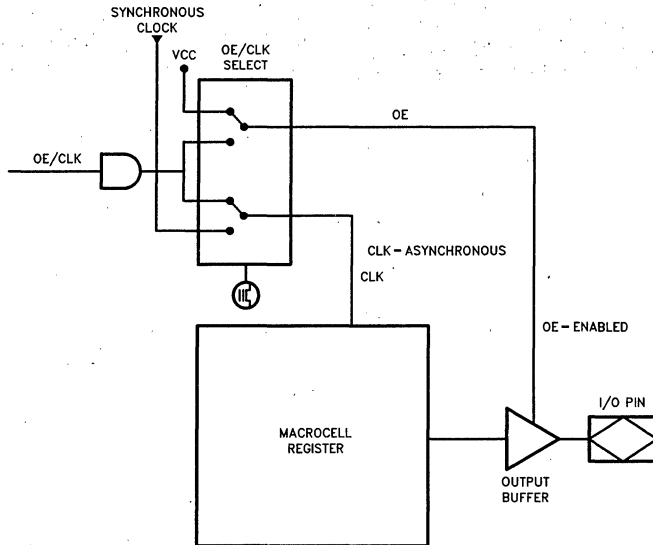
In Mode 1, the Output Buffer is always enabled. The Macrocell flip-flop now may be triggered from an asynchronous clock signal generated by the Macrocell product term. This mode allows individual clocking of flip-flops from any available signal in the quadrant AND array. Because both true and complement signals reside in the AND array, the flip-flops may be configured for positive or negative edge triggered operation. With the clock now controlled by a product term, gate clock structures are also possible.

In Modes 2 and 3, the Output Buffer is always disabled. The Macrocell flip-flop may still be triggered from clock signals generated from the Macrocell product term or asynchronous clocks. This mode is only possible for Global Macrocells.



290111-12

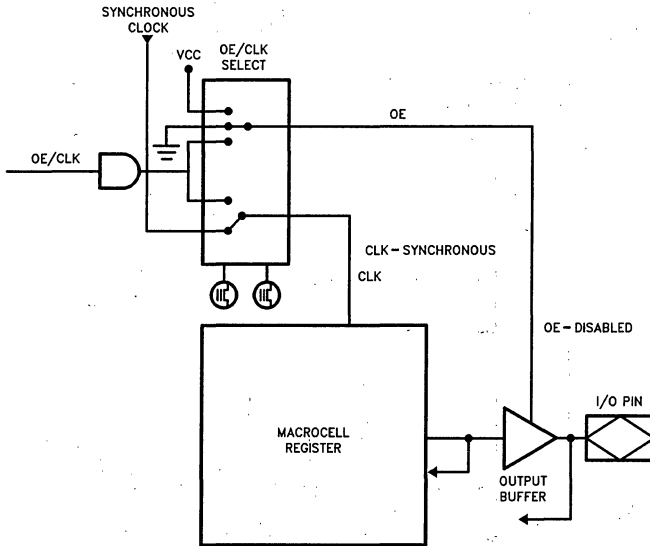
The register is clocked by the quadrant synchronous clock signal which is common to 11 other Macrocells. The output is enabled by the logic from the product term.



290111-13

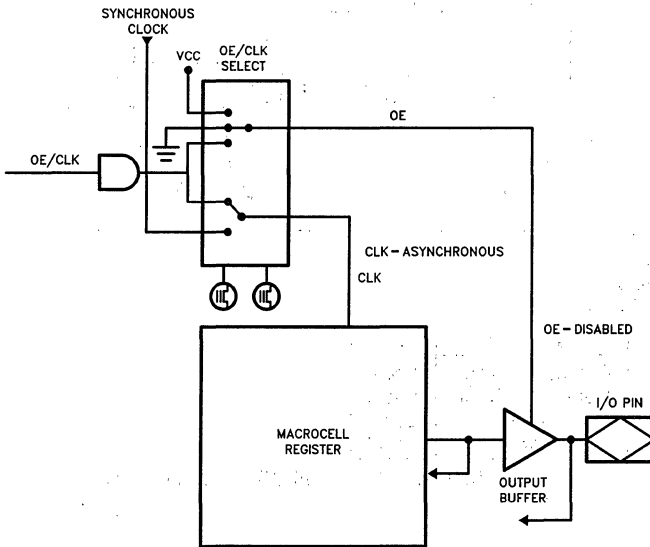
The output is permanently enabled and the register is clocked via the product term. This allows for gated clocks that may be generated from elsewhere in the 5C180.

Figure 9a. Local Macrocell OE/CLK Selection



290111-14

The output is permanently disabled and the register clocked by the quadrant synchronous clock signal. The pin can be used as an input while the register or combinational output can be fed back.



290111-15

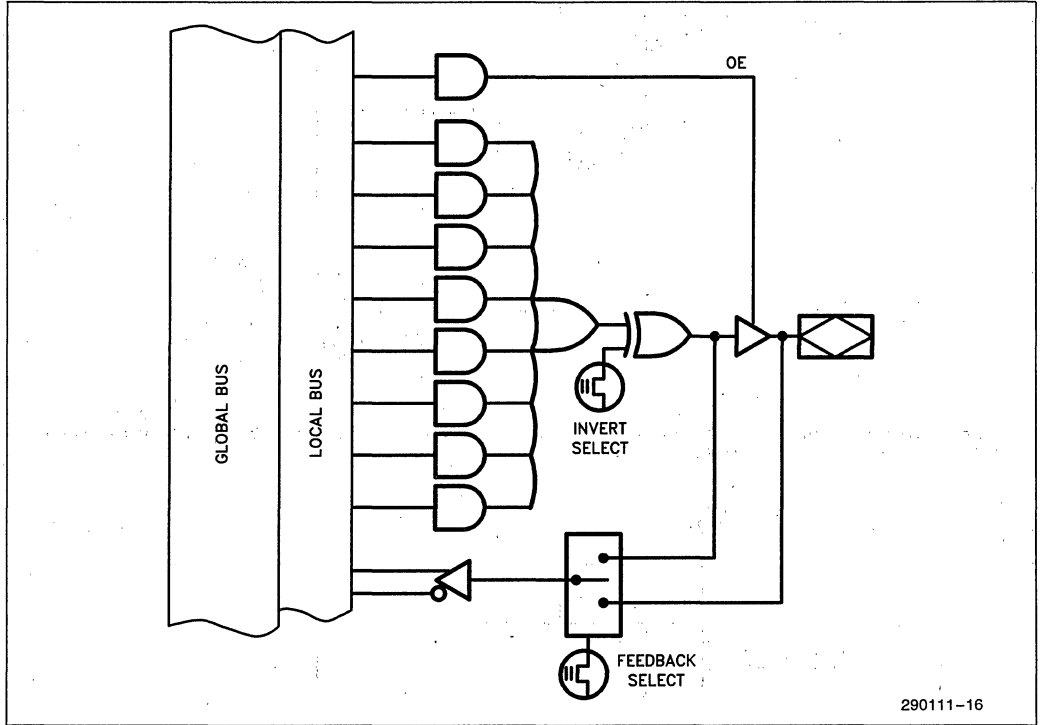
The output is permanently disabled and the register is clocked via the product term. This allows gated clocks that may be generated elsewhere in the 5C180. The pin can be used as an input while the register or combinational output can be fed back.

Figure 9b. Global Macrocell Additional OE/CLK Selection

**MACROCELL LOGIC + I/O CONFIGURATIONS**

The 5C180 Input/Output Architecture provides each Macrocell with over 50 possible I/O configurations.

Figures 10 and 11 show the 5C180 basic I/O configurations for both the Local and Global Macrocells. Along with combinatorial, four register types are available. Each Macrocell may be independently programmed.

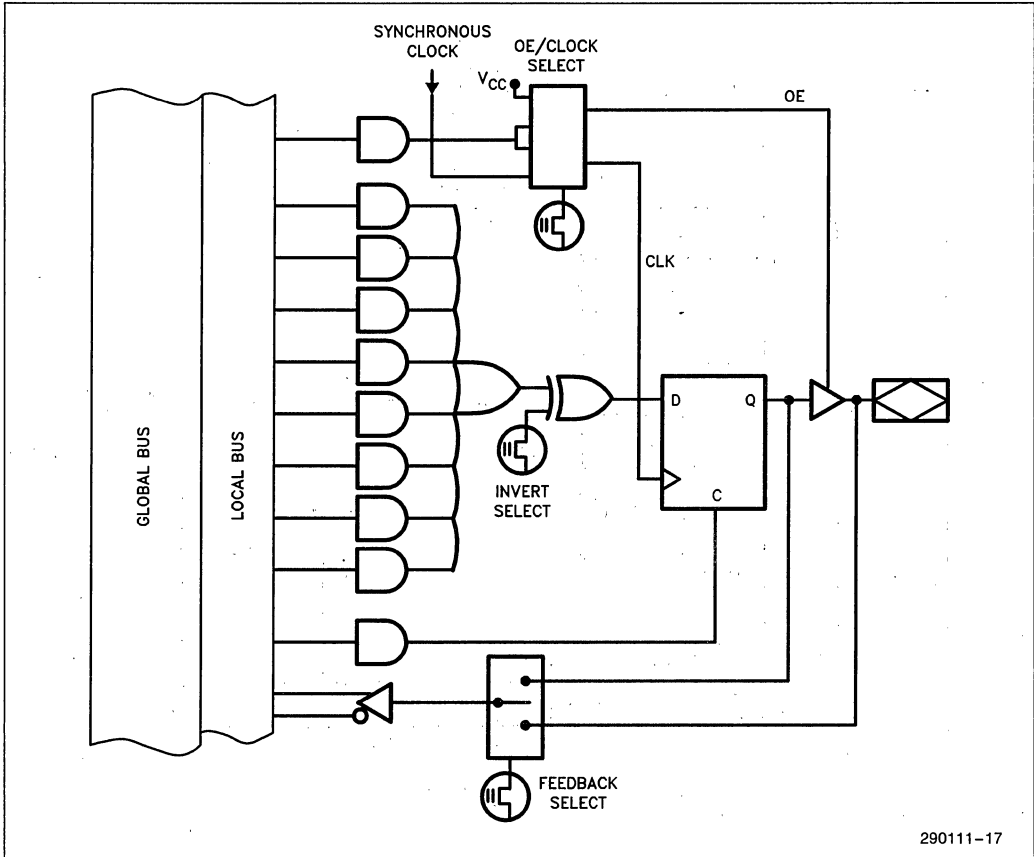


290111-16

**COMBINATORIAL I/O Selection**

Output/Polarity	Feedback	Bus
Combinatorial/High	Comb, Pin, None	Local
Combinatorial/Low	Comb, Pin, None	Local
None	Comb	Local
None	Pin	Local

Figure 10. Local Macrocell I/O Configurations



**D-TYPE FLIP-FLOP**

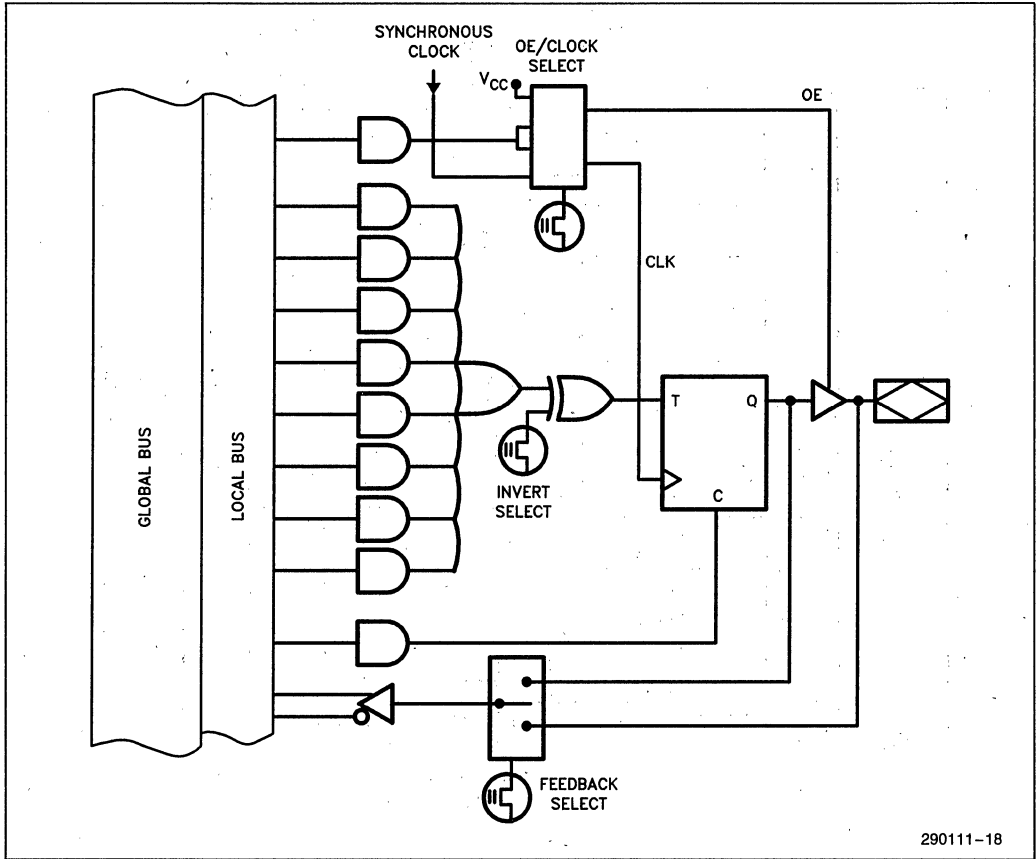
**I/O Selection**

Output/Polarity	Feedback	Bus
D-Register/High	D-Register, Pin, None	Local
D-Register/Low	D-Register, Pin, None	Local
None	D-Register	Local
None	Pin	Local

**Function Table**

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

Figure 10. Local Macrocell I/O Configurations (Continued)



290111-18

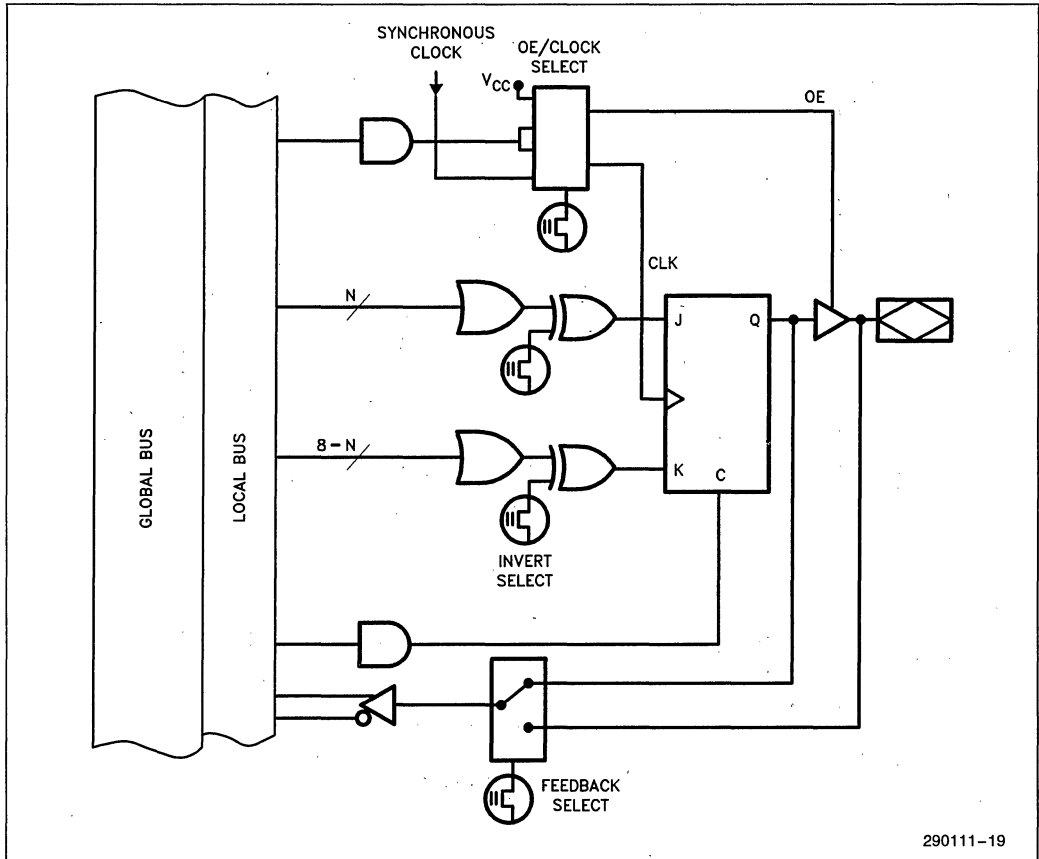
**TOGGLE FLIP-FLOP**  
I/O Selection

Output/Polarity	Feedback	Bus
T-Register/High	T-Register, Pin, None	Local
T-Register/Low	T-Register, Pin, None	Local
None	T-Register	Local
None	Pin	Local

**Function Table**

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

Figure 10. Local Macrocell I/O Configurations (Continued)



290111-19

**JK FLIP-FLOP**

I/O Selection

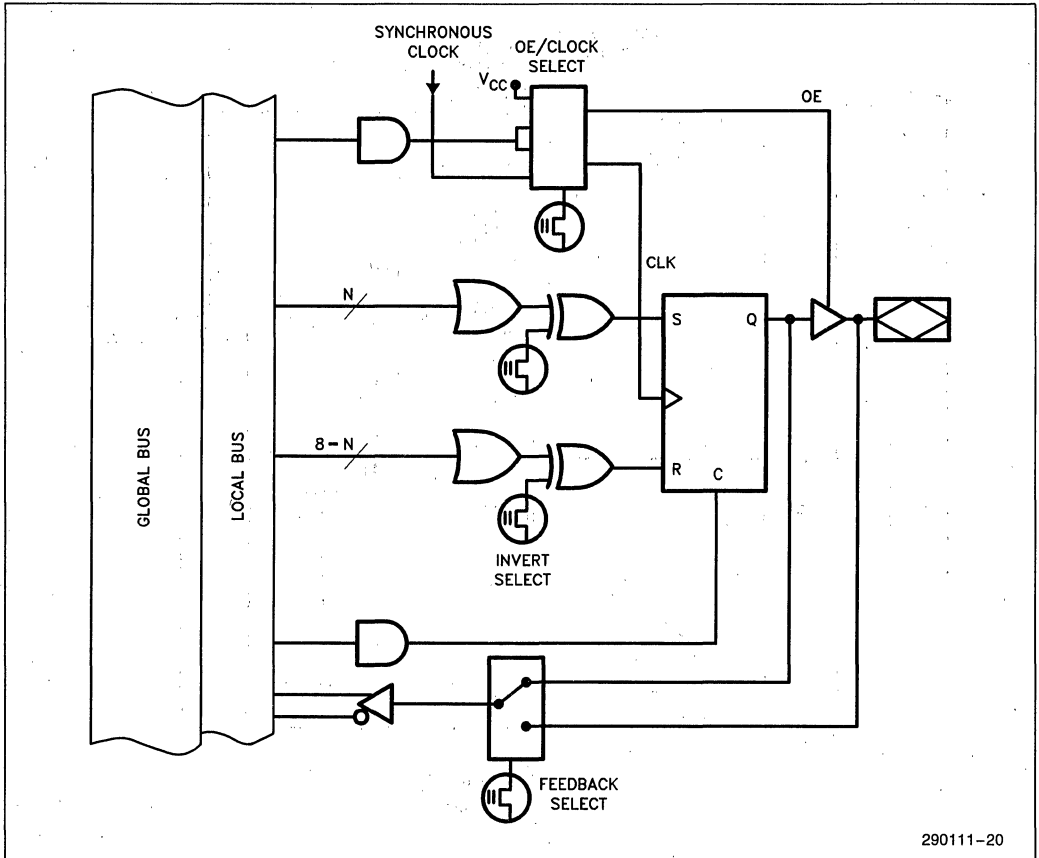
Output/Polarity	Feedback	Bus
JK Register/High	JK Register, None	Local
JK Register/Low	JK Register, None	Local
None	JK Register	Local

**Function Table**

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

Figure 10. Local Macrocell I/O Configurations (Continued)





290111-20

**SR FLIP-FLOP**

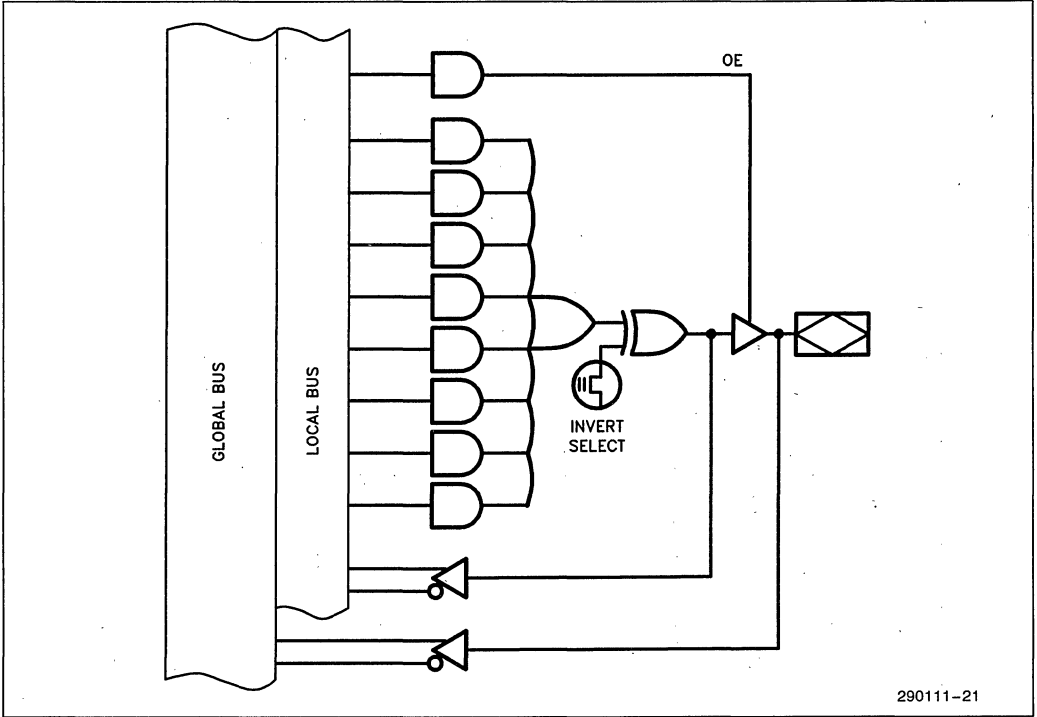
**I/O Selection**

Output/Polarity	Feedback	Bus
SR Register/High	SR Register, None	Local
SR Register/Low	SR Register, None	Local
None	SR Register	Local

**Function Table**

S	R	$Q_n$	$Q_{n+1}$
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1

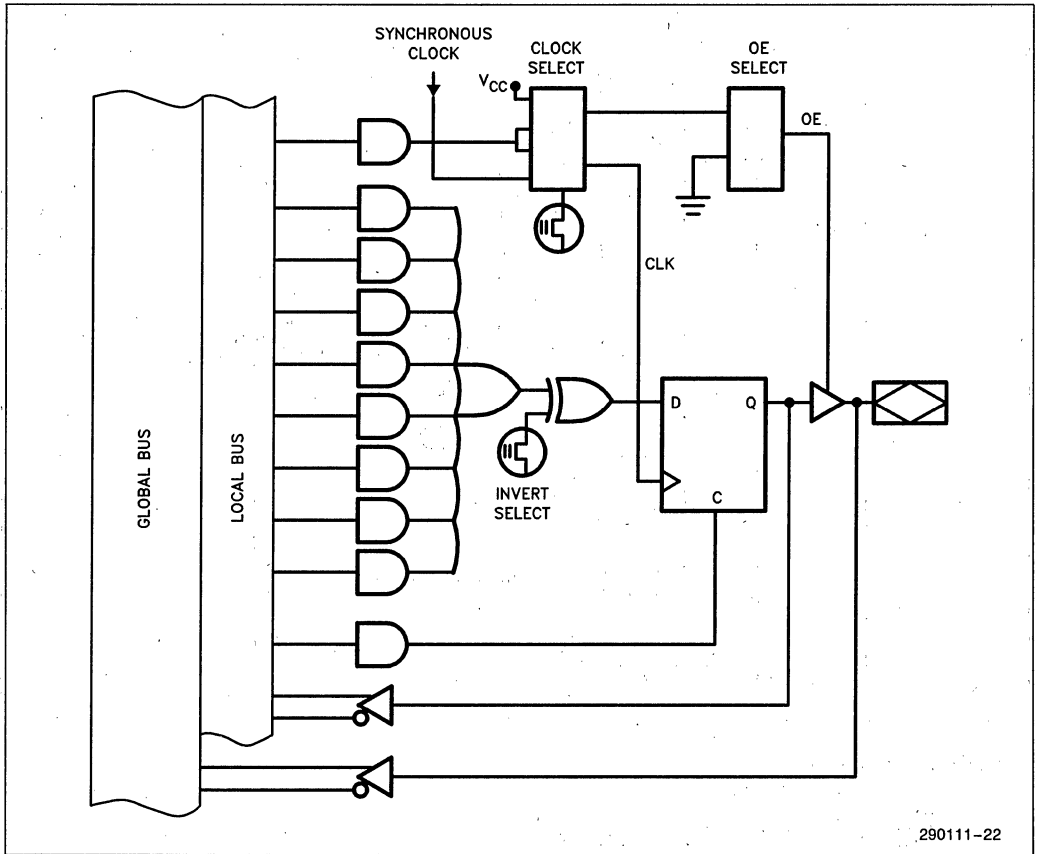
Figure 10. Local Macrocell I/O Configurations (Continued)



**COMBINATORIAL**  
I/O Selection

Output/Polarity	Feedback	Bus
Combinatorial/High	Comb, Pin, None	Local, Global
Combinatorial/Low	Comb, Pin, None	Local, Global
None	Comb	Local, Global
None	Pin	Global
None	Comb/Pin	Local/Global

Figure 11. Global Macrocell I/O Configurations



290111-22

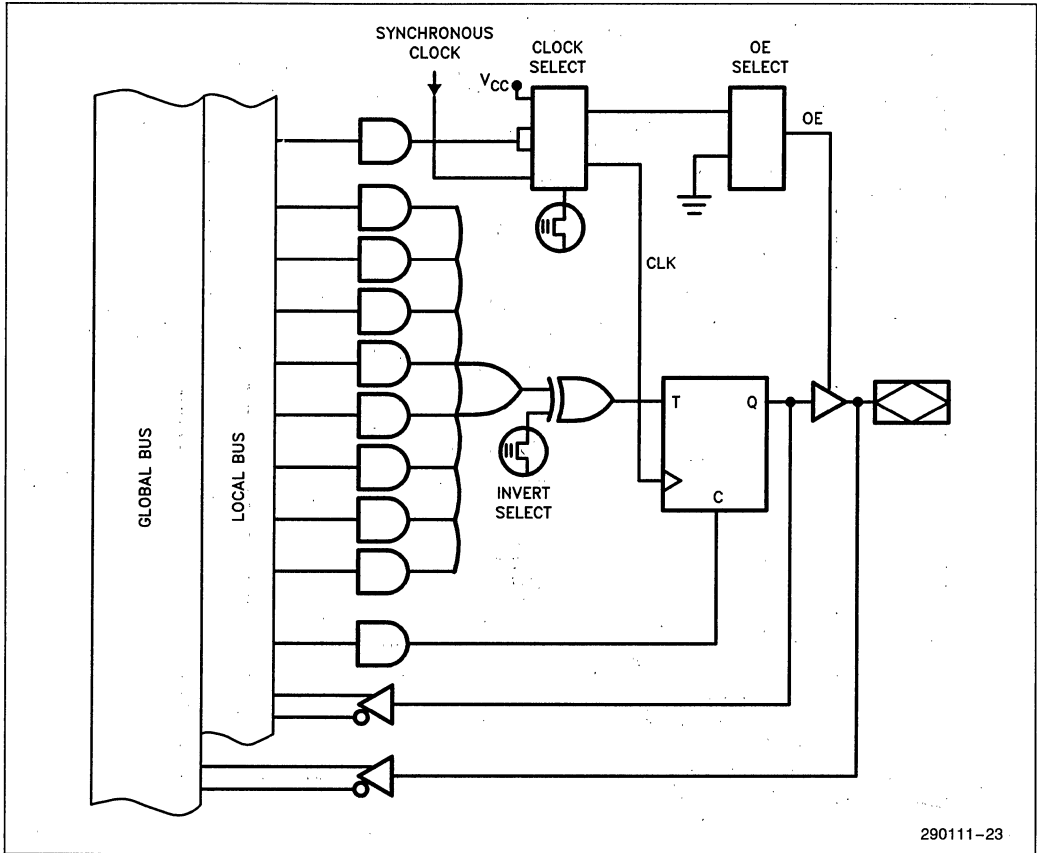
**D-TYPE FLIP-FLOP**  
I/O Selection

Output/Polarity	Feedback	Bus
D-Register/High	D-Register, Pin, None	Local, Global
D-Register/Low	D-Register, Pin, None	Local, Global
None	D-Register	Local, Global
None	Pin	Global
None	D-Register/Pin	Local/Global

**Function Table**

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

Figure 11. Global Macrocell I/O Configurations (Continued)



290111-23

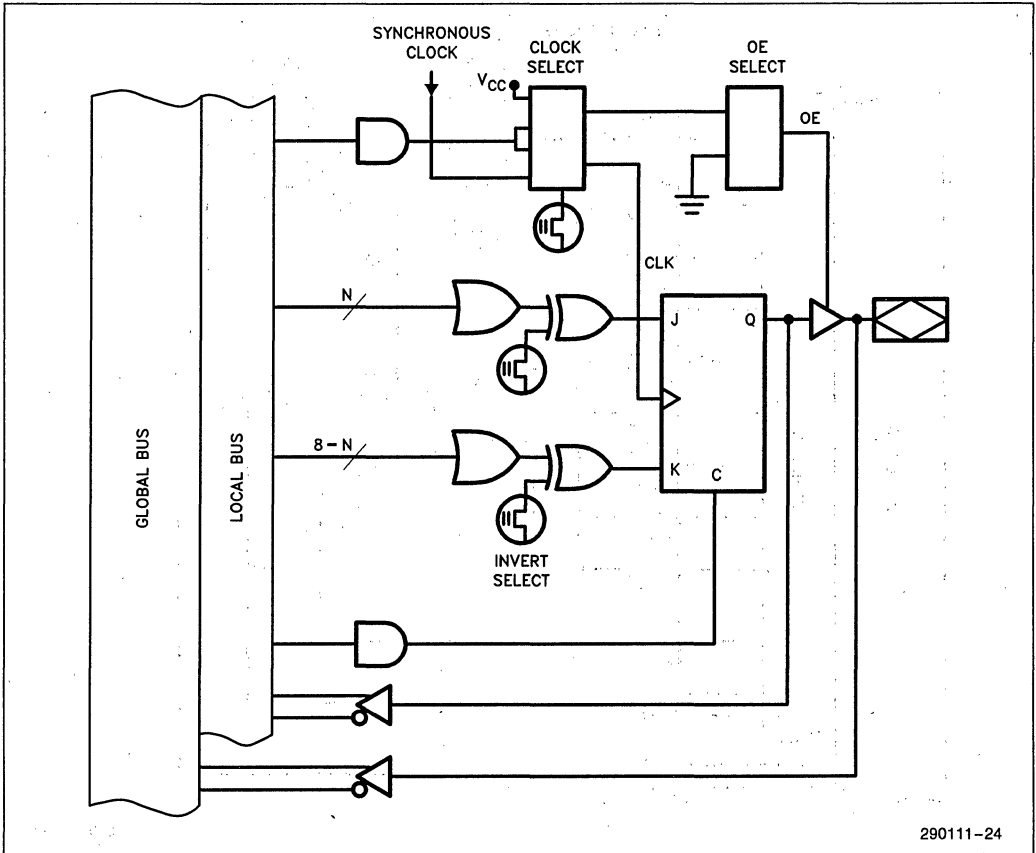
**TOGGLE FLIP-FLOP**  
I/O Selection

Output/Polarity	Feedback	Bus
T-Register/High	T-Register, Pin, None	Local, Global
T-Register/Low	T-Register, Pin, None	Local, Global
None	T-Register	Local, Global
None	Pin	Global
None	T-Register/Pin	Local/Global

**Function Table**

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

Figure 11. Global Macrocell I/O Configurations (Continued)



**JK FLIP-FLOP**

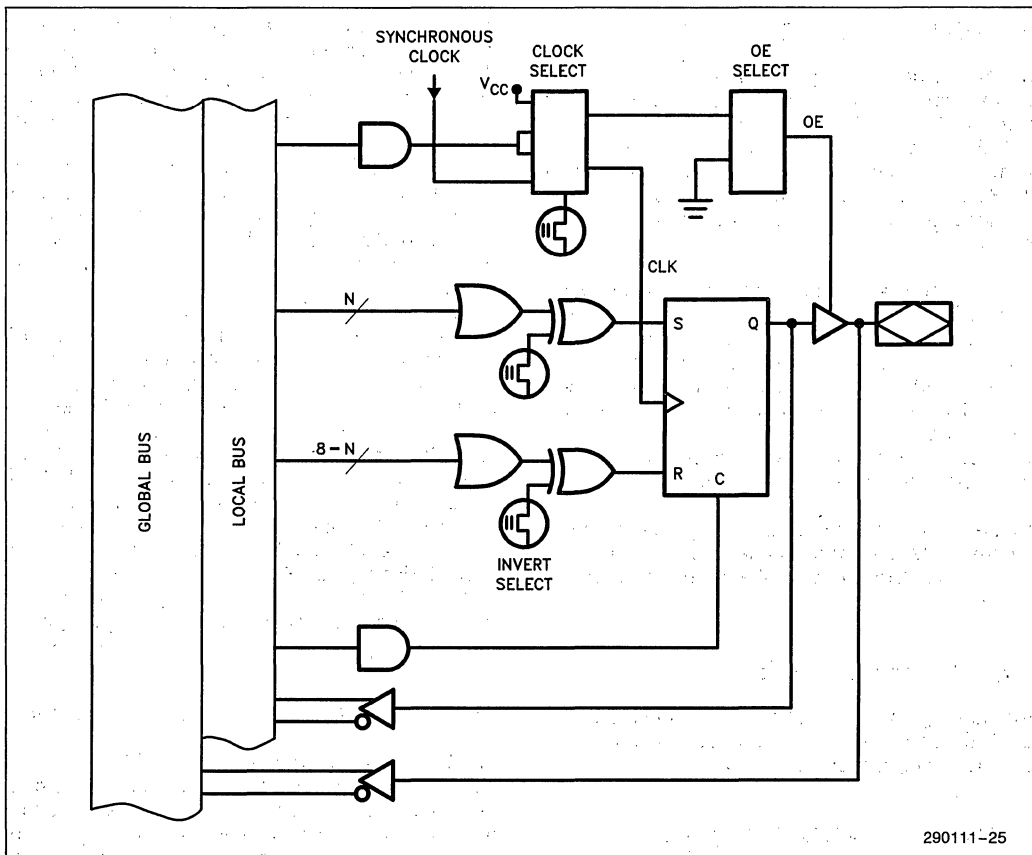
**I/O Selection**

Output/Polarity	Feedback	Bus
JK Register/High	JK Register, None	Local, Global
JK Register/Low	JK Register, None	Local, Global
None	JK Register	Local
None	JK Register/Pin	Local/Global

**Function Table**

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

Figure 11. Global Macrocell I/O Configurations (Continued)



290111-25

**SR FLIP-FLOP**

**I/O Selection**

Output/Polarity	Feedback	Bus
SR Register/High	SR Register, None	Local, Global
SR Register/Low	SR Register, None	Local, Global
None	SR Register	Local
None	SR Register/Pin	Local/Global

**Function Table**

S	R	$Q_n$	$Q_{n+1}$
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1

Figure 11. Global Macrocell I/O Configurations (Continued)

### AUTOMATIC STAND-BY MODE

The 5C180 contains a programmable bit, the Turbo Bit, that optimizes operation for speed or for power savings. When the Turbo Bit is programmed (TURBO = ON), the device is optimized for maximum speed. When the Turbo Bit is not programmed (TURBO = OFF), the device is optimized for power savings by entering standby mode during periods of inactivity.

Figure 12 shows the device entering standby mode approximately 100 ns after the last input transition. When the next input transition is detected, the device returns to active mode. Wakeup time adds an additional 30 ns to the propagation delay through the device as measured from the first input. No delay will occur if an output is dependent on more than one input and the last of the inputs changes after the device has returned to active mode.

After erasure, the Turbo Bit is unprogrammed (OFF); automatic standby mode is enabled. When the Turbo Bit is programmed (ON), the device never enters standby mode.

### Erased-State Configuration

Prior to programming or after erasing, the I/O structure is configured for combinatorial active low output with input (pin) feedback.

### ERASURE CHARACTERISTICS

Erasure characteristics of the 5C180 are such that erasure begins to occur upon exposure to light with

wavelengths shorter than approximately 4000Å. It should be noted that sunlight and certain types of fluorescent lamps have wavelengths in the 3000Å–4000Å range. Data shows that constant exposure to room level fluorescent lighting could erase the typical 5C180 in approximately three years, while it would take approximately one week to cause erasure when exposed to direct sunlight. If the 5C180 is to be exposed to these types of lighting conditions for extended periods of time, conductive opaque labels should be placed over the device window to prevent unintentional erasure.

The recommended erasure procedure for the 5C180 is exposure to shortwave ultraviolet light with a wavelength of 2537Å. The integrated dose (i.e., UV intensity × exposure time) for erasure should be a minimum of fifteen (15) Wsec/cm<sup>2</sup>. The erasure time with this dosage is approximately 15 to 20 minutes using an ultraviolet lamp with a 12,000 μW/cm<sup>2</sup> power rating. The 5C180 should be placed within one inch of the lamp tubes during erasure. The maximum integrated dose the 5C180 can be exposed to without damage is 7258 Wsec/cm<sup>2</sup> (1 week at 12,000 μW/cm<sup>2</sup>). Exposure to high intensity UV light for longer periods may cause permanent damage to the device.

### PROGRAMMING CHARACTERISTICS

Initially, and after erasure, all the EPROM control bits of the 5C180 are connected. Each of the connected control bits are selectively disconnected by programming the EPROM cells into their "on" state. Programming voltage and waveform specifications are available by request from Intel to support programming of the 5C180.

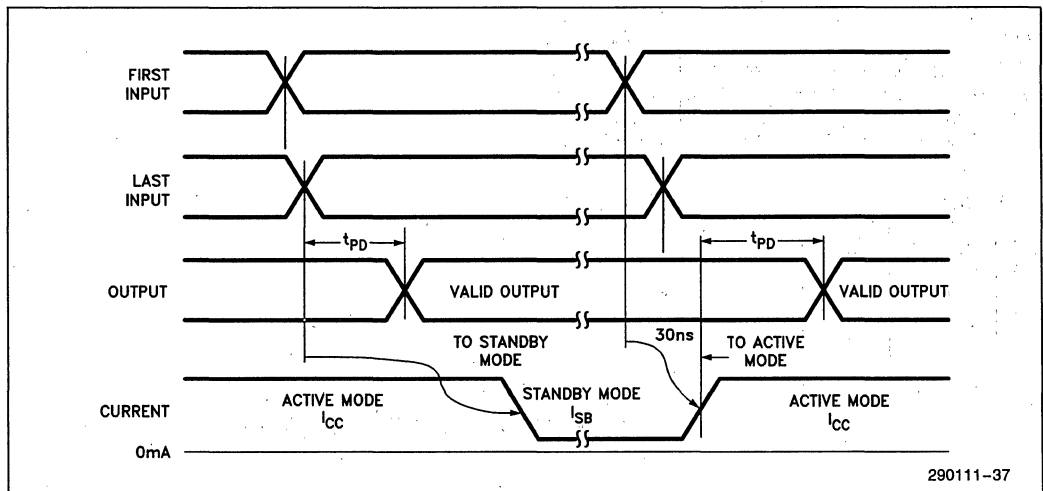


Figure 12. 5C180 Standby and Active Mode Transitions

## intelligent Programming™ Algorithm

The 5C180 supports the intelligent Programming Algorithm which rapidly programs Intel H-ELPDs (and EPROMs) using an efficient and reliable method. The intelligent Programming Algorithm is particularly suited to the production programming environment. This method greatly decreases the overall programming time while programming reliability is ensured as the incremental program margin of each bit is continually monitored to determine when the bit has been successfully programmed.

## FUNCTIONAL TESTING

Since the logical operation of the 5C180 is controlled by EPROM elements, the device is completely testable. Each programmable EPROM bit controlling the internal logic is tested using application-independent test program patterns. After testing, the devices are erased before shipment to customers. No post-programming tests of the EPROM array are required.

The testability and reliability of EPROM-based programmable logic devices is an important feature over similar devices based on fuse technology. Fuse-based programmable logic devices require a use to perform post-programming tests to insure proper programming. These tests must be done at the device level because of the cumulative error effect. For example, a board containing ten devices each possessing a 2% device fallout translates into an 18% fallout at the board level (it should be noted that programming fallout of fuse-based programmable logic devices is typically 2% or higher).

## DESIGN RECOMMENDATIONS

For proper operation, it is recommended that all input and output pins be constrained to the voltage range  $GND < (V_{IN} \text{ or } V_{OUT}) < V_{CC}$ . Unused inputs should be tied to an appropriate logic level (e.g., either  $V_{CC}$  or GND) to minimize device power consumption. Reserved pins (as indicated in the logic compiler REPORT file) should be left floating (no connect) so that the pin can attain the appropriate logic level. A power supply decoupling capacitor of at least 0.2  $\mu\text{f}$  must be connected directly between  $V_{CC}$  and GND.

As with all CMOS devices, ESD handling procedures should be used with this device to prevent damage during programming, assembly, and test.

## DESIGN SECURITY

A single EPROM bit provides a programmable design security feature that controls the access to the data programmed into the device. If this bit is set, a proprietary design within the device cannot be copied. This EPROM security bit enables a higher degree of design security than fused-based devices since programmed data within EPROM cells is invisible even to microscopic evaluation. The EPROM security bit, along with all the other EPROM control bits, will be reset by erasing the device.

## LATCH-UP IMMUNITY

All of the input, I/O, and clock pins of the 5C180 have been designed to resist latch-up which is inherent in inferior CMOS structures. The 5C180 is designed with Intel's proprietary CHMOS II-E EPROM process. Thus, each of the 5C180 pins will not experience latch-up with currents up to 100 mA and voltages ranging from  $-1\text{V}$  to  $V_{CC} + 1\text{V}$ . Furthermore, the programming pin is designed to resist latch-up to the 13.5V maximum device limit.

## INTEL PROGRAMMABLE LOGIC DEVELOPMENT SYSTEM II (iPLDS II)

iPLDS II provides all the tools needed to design with Intel H-Series EPLDs or compatible devices. In addition to providing development assistance, iPLDS II insulates the user from having to know all the intricate details of EPLD architecture (the machine will optimize a design to benefit from architectural features). It contains comprehensive third generation software that supports several different design entry methods, minimizes logic, does automatic pin assignments and produces the best design fit for the selected EPLD. It is user friendly with guided menus, on-line Help messages and soft key inputs.

In addition, the iPLDS II contains programmer hardware in the form of an iUP-PC Universal Programmer-Personal Computer to enable the user to program EPLDs, read and verify programmed devices and also to graphically edit programming files. The software generates industry standard JEDEC object code output files which can be downloaded to other programmers as well.

iPLDS II has interfaces to popular schematic capture packages to enable designs to be entered using schematics. An integrated, schematic entry method is provided by SCHEMA II-PLD, a low-cost schematic capture package that supports EPLD primitives and user-defined macro symbols. SCHEMA II-PLD contains the EPLD Design Manager, which provides a single user interface to both SCHEMA II-PLD and iPLS II software. TTL symbol and macro libraries are



available for SCHEMA II-PLD to simplify the design process. The other design formats supported are Boolean equation entry and State Machine design entry. For additional information on iPLDS II, refer to the iPLDS II Data Sheet, order number: 290134.

iPLDS II operates on the IBM† PC/XT, PC/AT, or other compatible machine with the following configuration:

1. At least one floppy disk drive and hard disk drive.
2. MS-DOS‡ Operating System Version 3.0 or greater.
3. 512K Memory (640K recommended).
4. Intel iUP-PC Universal Programmer-Personal Computer (supplied with iPLDS II).
5. GUPI LOGIC Adaptor.
6. A color monitor is suggested.

† IBM Personal Computer is a registered trademark of International Business Machines Corporation.

‡ MS-DOS is a registered trademark of Microsoft Corporation.

**ADF PRIMITIVES SUPPORTED**

The following ADF primitives are supported by this device:

- |      |      |
|------|------|
| INP  | JOJF |
| CONF | JONF |
| COCF | SONF |
| COIF | SOSF |
| RONF | TOIF |
| RORF | TONF |
| ROIF | TOTF |
| NOCF | CLKB |
| NORF |      |
| NOJF |      |
| NOSF |      |
| NOTF |      |

**ORDERING INFORMATION**

t <sub>PD</sub> (ns)	t <sub>CO</sub> (ns)	f <sub>MAX</sub> (MHz)	Order Code	Package	Operating Range
70	29	20.8	CJ5C180-70	JLCC	Commercial
			N5C180-70	PLCC	
			A5C180-70	PGA	
75	30	19.6	CJ5C180-75	JLCC	Commercial
			N5C180-75	PLCC	
			A5C180-75	PGA	
90	35	16.1	CJ5C180-90	JLCC	Commercial
			N5C180-90	PLCC	
			A5C180-90	PGA	

**ABSOLUTE MAXIMUM RATINGS\***

Symbol	Parameter	Min	Max	Units
V <sub>CC</sub>	Supply Voltage(1)	-2.0	7.0	V
V <sub>PP</sub>	Programming Supply Voltage(1)	-2.0	13.5	V
V <sub>I</sub>	DC Input Voltage(1)(2)	-0.5	V <sub>CC</sub> +0.5	V
t <sub>stg</sub>	Storage Temperature	-65	+150	°C
t <sub>amb</sub>	Ambient Temperature(3)	-10	+85	°C

\*Notice: Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

NOTICE: Specifications contained within the following tables are subject to change.

**NOTES:**

1. Voltages with respect to ground.
2. Minimum DC input is -0.5V. During transitions, the inputs may undershoot to -2.0V or overshoot to 7.0V for periods less than 20 ns under no load conditions.
3. Under bias. Extended temperature versions are also available.

**RECOMMENDED OPERATING CONDITIONS**

Symbol	Parameter	Min	Max	Units
V <sub>CC</sub>	Supply Voltage	4.75	5.25	V
V <sub>IN</sub>	Input Voltage	0	V <sub>CC</sub>	V
V <sub>O</sub>	Output Voltage	0	V <sub>CC</sub>	V
T <sub>A</sub>	Operating Temperature	0	+70	°C
t <sub>R</sub> (4)	Input Rise Time		500	ns
t <sub>F</sub> (4)	Input Fall Time		500	ns

**NOTE:**

4. t<sub>R</sub> and t<sub>F</sub> for clocks is 250 ns.

**D.C. CHARACTERISTICS** T<sub>A</sub> = 0° to +70°C, V<sub>CC</sub> = 5V ±5%

Symbol	Parameter/Test Conditions	Min	Typ	Max	Unit
V <sub>IH</sub> (5)	High Level Input Voltage	2.0		V <sub>CC</sub> + 0.3	V
V <sub>IL</sub> (5)	Low Level Input Voltage	-0.3		0.8	V
V <sub>OH</sub> (6)	High Level Output Voltage I <sub>O</sub> = -4.0 mA D.C., V <sub>CC</sub> = min.	2.4			V
V <sub>OL</sub>	Low Level Output Voltage I <sub>O</sub> = 4.0 mA D.C., V <sub>CC</sub> = min.			0.45	V
I <sub>I</sub>	Input Leakage Current V <sub>CC</sub> = max., GND < V <sub>IN</sub> < V <sub>CC</sub>			± 10	µA
I <sub>OZ</sub>	Output Leakage Current V <sub>CC</sub> = max., GND < V <sub>OUT</sub> < V <sub>CC</sub>			± 10	µA

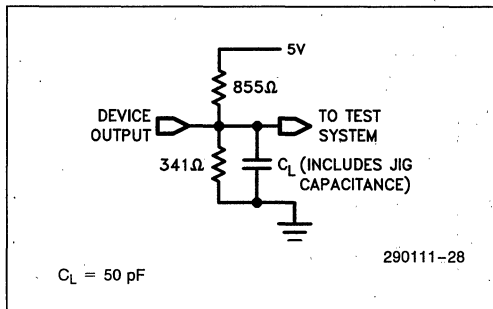
**NOTES:**

5. Absolute values with respect to device GND; all over and undershoots due to system or tester noise are included.
6. I<sub>O</sub> at CMOS levels (3.84 V) = -2 mA
7. Not more than 1 output should be tested at a time. Duration of that test must not exceed 1 second.
8. With Turbo Bit Off, device automatically enters standby mode approximately 100 ns after last input transition.

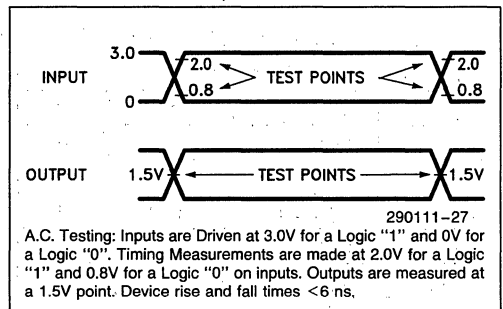
**D.C. CHARACTERISTICS**  $T_A = 0^\circ$  to  $+70^\circ\text{C}$ ,  $V_{CC} = 5V \pm 5\%$  (Continued)

Symbol	Parameter/Test Conditions	Min	Typ	Max	Unit
$I_{SC}^{(7)}$	Output Short Circuit Current $V_{CC} = \text{max.}, V_{OUT} = 0.5V$		20	30	mA
$I_{SB}^{(8)}$	Standby Current $V_{CC} = \text{max.}, V_{IN} = V_{CC}$ or GND, Standby mode		35	150	$\mu\text{A}$
$I_{CC}$	Power Supply Current $V_{CC} = \text{max.}, V_{IN} = V_{CC}$ or GND, No load, Input Freq. = 1 MHz Active mode (Turbo = Off), Device prog. as four 12-bit Ctrs.		30	45	mA

**A.C. TESTING LOAD CIRCUIT**



**A.C. TESTING INPUT, OUTPUT WAVEFORM**



**CAPACITANCE**

Symbol	Parameter	Min	Typ	Max	Unit	Conditions
C <sub>IN</sub>	Input Capacitance			15	pF	V <sub>IN</sub> = 0V, f = 1.0 MHz
C <sub>OUT</sub>	Output Capacitance			15	pF	V <sub>OUT</sub> = 0V, f = 1.0 MHz
C <sub>CLK</sub>	Clock Pin Capacitance			25	pF	V <sub>OUT</sub> = 0V, f = 1.0 MHz
C <sub>VPP</sub>	V <sub>PP</sub> Pin Capacitance			160	pF	CLK2, V <sub>OUT</sub> = 0V, f = 1.0 MHz

**A.C. CHARACTERISTICS** T<sub>A</sub> = 0°C to +70°C, V<sub>CC</sub> = 5V ±5%, Turbo Bit On<sup>(9)</sup>

Symbol	From	To	5C180-70 EP1800-2			5C180-75			5C180-90 EP1800			Non-Turbo Mode <sup>(11)</sup>	Unit
			Min	Typ	Max	Min	Typ	Max	Min	Typ	Max		
t <sub>PD1</sub>	Input <sup>(12)</sup>	Comb. Output			65			70			85	+30	ns
t <sub>PD2</sub>	I/O <sup>(12)</sup>	Comb. Output			70			75			90	+30	ns
t <sub>PD2e</sub>	I/O <sup>(13)</sup>	Comb. Output			65			70			85	+30	ns
t <sub>PZX</sub> <sup>(10)</sup>	I or I/O	Output Enable			70			75			90	+30	ns
t <sub>PXZ</sub> <sup>(10)</sup>	I or I/O	Output Disable			70			75			90	+30	ns
t <sub>CLR</sub>	Asynch. Reset	Q Reset			70			75			90	+30	ns

**NOTES:**

9. Typ. Values are at T<sub>A</sub> = 25°C, V<sub>CC</sub> = 5V, Active Mode.

10. t<sub>PZX</sub> and t<sub>PXZ</sub> are measured at ±0.5V from steady state voltage as driven by spec. output load. t<sub>PXZ</sub> is measured with C<sub>L</sub> = 5 pF.

11. If device is operated with Turbo Bit Off (Non-Turbo Mode), increase time by amount shown.

**SYNCHRONOUS CLOCK MODE A.C. CHARACTERISTICS**

T<sub>A</sub> = 0°C to +70°C, V<sub>CC</sub> = 5V ±5%, Turbo Bit On<sup>(9)</sup>

Symbol	Symbol	5C180-70 EP1800-2			5C180-75			5C180-90 EP1800			Non-Turbo Mode <sup>(11)</sup>	Unit
		Min	Typ	Max	Min	Typ	Max	Min	Typ	Max		
f <sub>MAX</sub>	Max Frequency 1/(t <sub>CH</sub> + t <sub>CL</sub> )—No Feedback			20.8			19.6			16.1		MHz
f <sub>CNT</sub>	Max. Count Frequency 1/t <sub>CNT</sub> —With Feedback			16.1			15.1			12.2		MHz
t <sub>SU1</sub>	Input Setup Time to Clk <sup>(12)</sup>	48			51			62			+30	ns
t <sub>SU2</sub>	I/O Setup Time to Clk <sup>(12)</sup>	53			56			67			+30	ns
t <sub>SU2e</sub>	I/O Setup Time to Clk <sup>(13)</sup>	48			51			62			+30	ns
t <sub>H</sub>	I or I/O Hold after Clk High	0			0			0				ns
t <sub>CO</sub>	Clk High to Output Valid			29			30			35		ns
t <sub>CNT</sub>	Register Output Feedback to Register Input— Internal Path	62			66			82			+30	ns
t <sub>CH</sub>	Clk High Time	24			25			30				ns
t <sub>CL</sub>	Clk Low Time	24			25			30				ns

**ASYNCHRONOUS CLOCK MODE A.C. CHARACTERISTICS**

$T_A = 0^\circ\text{C}$  to  $+70^\circ\text{C}$ ,  $V_{CC} = 5\text{V} \pm 5\%$ , Turbo Bit On<sup>(9)</sup>

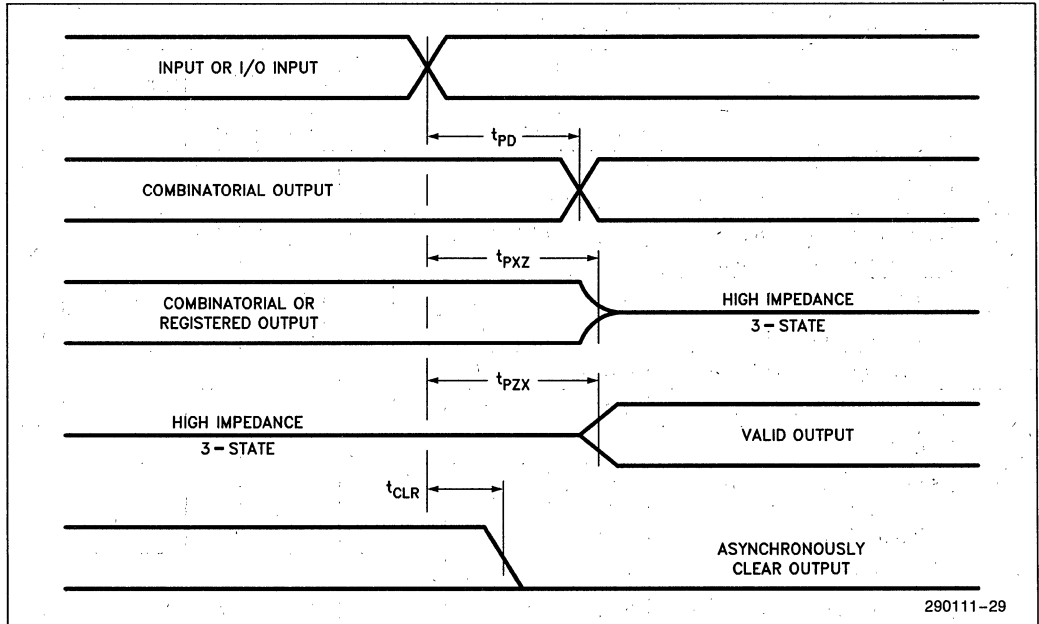
Symbol	Parameter	5C180-70 EP1800-2			5C180-75			5C180-90 EP1800			Non-Turbo Mode <sup>(11)</sup>	Unit
		Min	Typ	Max	Min	Typ	Max	Min	Typ	Max		
$f_{AMAX}$	Max. Frequency $1/(t_{ACH} + t_{ACL})$ —No Feedback			20.8			20			16.6		MHz
$f_{ACNT}$	Max. Frequency $1/t_{ACNT}$ —With Feedback			16.1			15.1			12.2		MHz
$t_{ASU1}$	Input Setup Time to Asynch. Clock <sup>(12)</sup>	17			19			23			+ 30	ns
$t_{ASU2}$	I/O Setup Time to Asynch. Clock <sup>(12)</sup>	22			25			28			+ 30	ns
$t_{AH}$	Input or I/O Hold to Asynch. Clock	30			30			30				ns
$t_{ACO}$	Asynch. Clk to Output Valid			70			75			90		ns
$t_{ACNT}$	Register Output Feedback to Register Input— Internal Path	62			66			82			+ 30	ns
$t_{ACH}$	Asynch. Clk High Time	24			25			30				ns
$t_{ACL}$	Asynch. Clk Low Time	24			25			30				ns

**NOTES:**

- 12. For General and Global Macrocells.
- 13. For Enhanced Macrocells.

**SWITCHING WAVEFORMS**

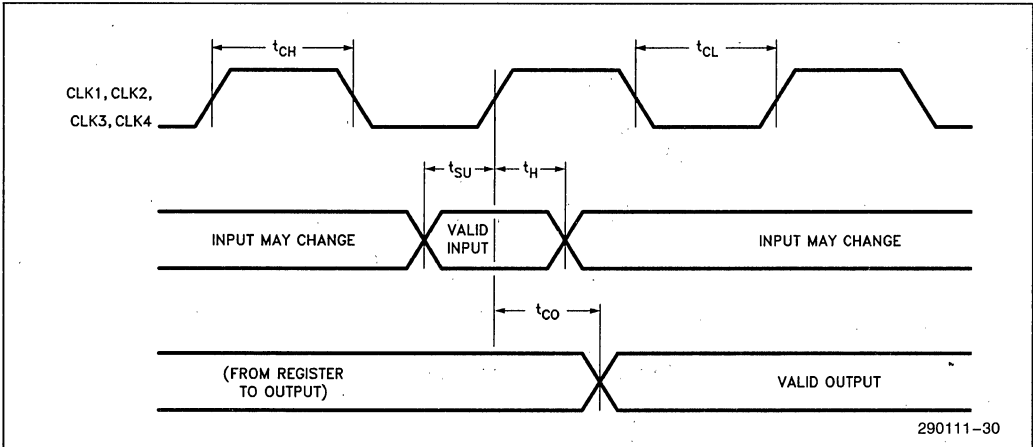
**COMBINATORIAL MODE**



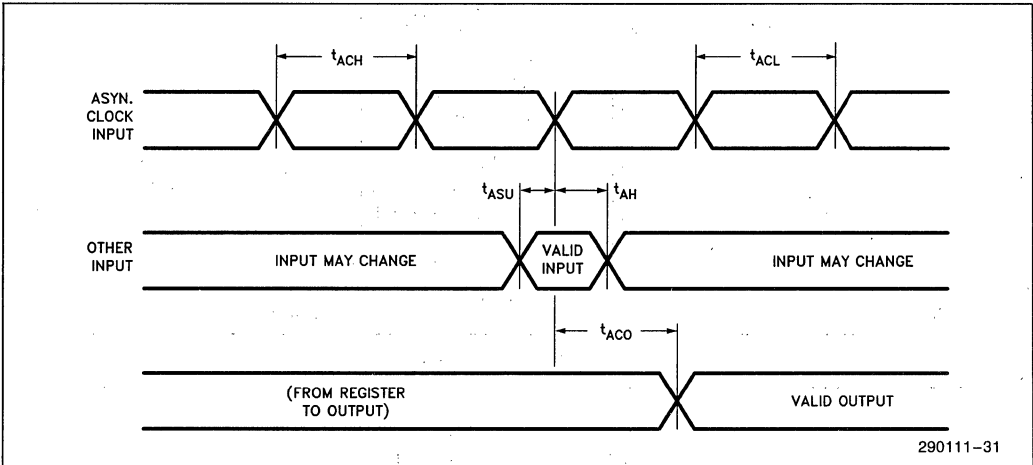
290111-29

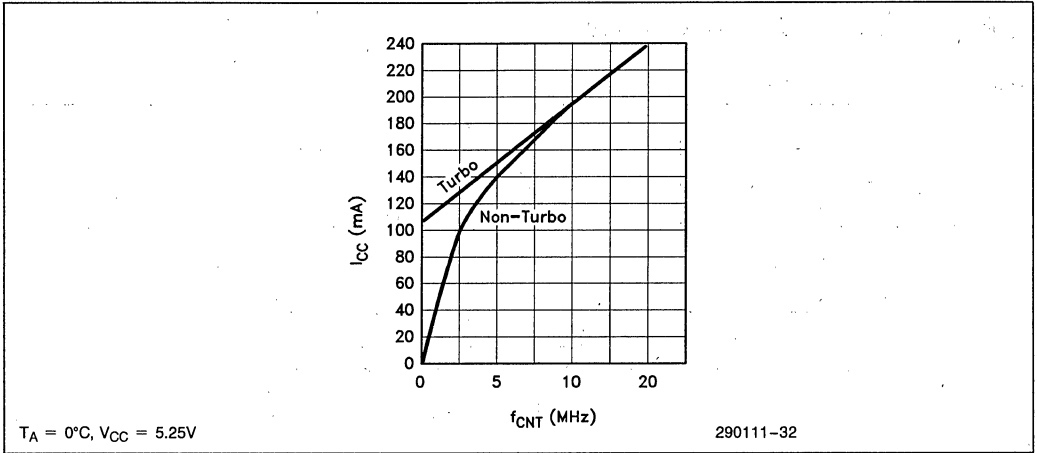
**SWITCHING WAVEFORMS** (Continued)

**SYNCHRONOUS CLOCK MODE**

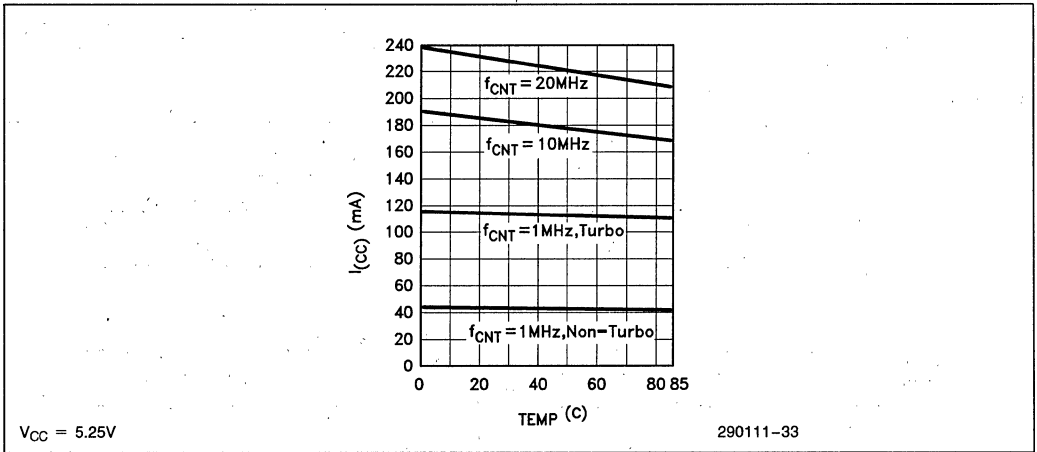


**ASYNCHRONOUS CLOCK MODE**

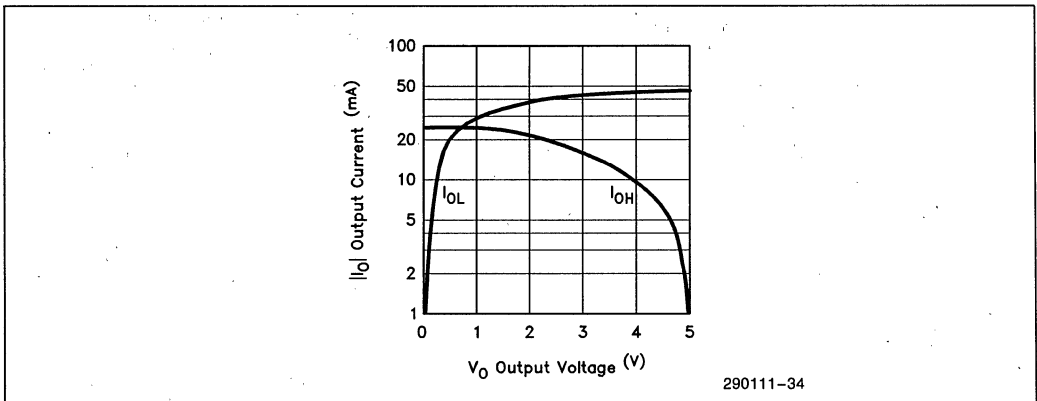




Current in Relation to Frequency



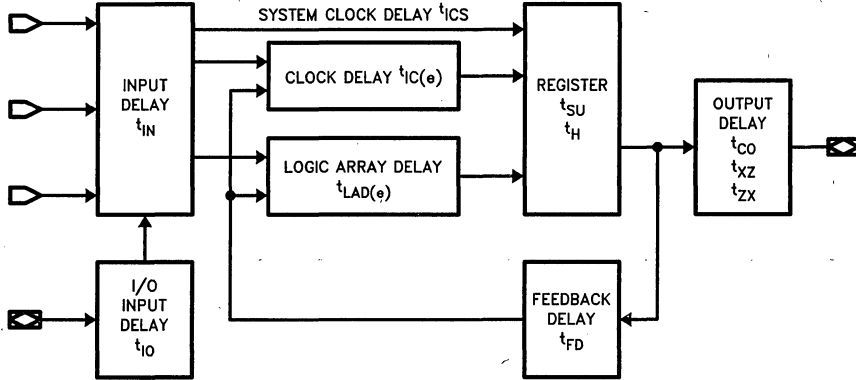
Current in Relation to Temperature



Output Drive Current in Relation to Voltage

### 5C180 INTERNAL TIMING

The following internal timing model and specifications are provided to aid in determining the different timing parameters for all permutations of timing paths through the device. The mnemonics in the table represent *internal parameters* only and should not be confused with external timing parameters shown in previous tables, even though some mnemonics are the same.



290111-38

Symbol	Parameter	5C180-70 EP1800-2		5C180-75		5C180-90 EP1800		Non-Turbo Mode <sup>(11)</sup>		Unit
		Min	Max	Min	Max	Min	Max	Min	Max	
$t_{IN}$	Input Pad and Buffer Delay		10		11		14		0	ns
$t_{IO}$	I/O Input Pad and Buffer Delay		5		5		5		0	ns
$t_{LADe}$	Enhanced Logic Array Delay		35		37		43		30	ns
$t_{LAD}$	Logic Array Delay		40		42		48		30	ns
$t_{OD}$	Output Buffer and Pad Delay		15		17		23		0	ns
$t_{ZX}$	Output Buffer Enable		15		17		23		0	ns
$t_{XZ}$	Output Buffer Disable		15		17		23		0	ns
$t_{SU}$	Register Setup Time	12		13		18		0		ns
$t_{HS}$	Register Hold Time (System Clock)	0		0		0		0		ns
$t_{H}$	Register Hold Time	30		30		30		0		ns
$t_{ICe}$	Enhanced Clock Delay		35		37		43		30	ns
$t_{IC}$	Clock Delay		40		42		48		30	ns
$t_{ICS}$	System Clock Delay		4		4		4		0	ns
$t_{FD}$	Feedback Delay		10		11		16		-30	ns
$t_{CLRe}$	Enhanced Register Clear Time		35		37		43		30	ns
$t_{CLR}$	Register Clear Time		40		42		48		30	ns





**APPLICATION  
BRIEF**

**AB-8**

May 1986

**Implementing Cascaded  
Logic in the 5C121**

**J. R. DONNELL**  
APPLICATIONS ENGINEER  
PROGRAMMABLE LOGIC

Order Number: 292003-001

**PROBLEM**

Designs that utilize numerous levels of cascaded logic often result in excessive product terms when expressed in the sum-of-products form. Although this poses no problem when designing with discrete logic, EPLDs are generally optimized for the sum-of-product form. This stems from the architecture of the basic Macrocell.

Macrocells typically consist of a programmable AND array feeding a fixed width OR gate. In the 5C121, OR gate widths range from four to sixteen inputs. For many applications, sixteen available product terms are sufficient. However, one example where product terms become an issue is cascaded exclusive-OR circuits. Here the number of product terms increase as  $2^n$  where n equals the number of exclusive-OR gates. If the number of product terms exceeds sixteen, the equation will not fit directly in the 5C121.

**SOLUTION**

There is a simple solution to reduce the product term requirements when using cascading XOR (or other) logic. Figure 1 shows a circuit cascading five exclusive ORs. As designed, this circuit expands to 32 product terms when expressed in the minimized sum-of-products form. (This is assuming that signals A thru F are

single product terms themselves.) Figure 2 shows the minimized logic equation file produced by Intel's Logic Optimizing Compiler (iLOC).

An easy solution to fitting this logic into the 5C121 is to cascade three exclusive ORs together and then send the result through a No Output Combinational Feedback primitive (NOCF). This signal can now be cascaded through two more XOR's to get the five total. This circuit is shown in Figure 3. Figure 4 shows the logic equation file for this implementation. Note the reduction in product terms from Figure 2. If the buried registers are available, Intel's iPLDs software will automatically assign the combinational feedback to a buried register thereby saving a pin. This technique can be used for any circuit that generates excessive product terms.

The only penalty in this method is the added delay needed for the feedback path. The worst case  $t_{pd}$  (input to output delay) for the circuit in Figure 3 would be twice the specified  $T_{pd}$  in the 5C121-XX data sheet. Basically the signal must go through the device twice. For the 5C121-90 the  $T_{pd}$  would be 180 ns worst case as implemented in Figure 3.

Figure 5 shows the report file generated by the compiler. In this case the NOCF path was automatically assigned to the buried registers.

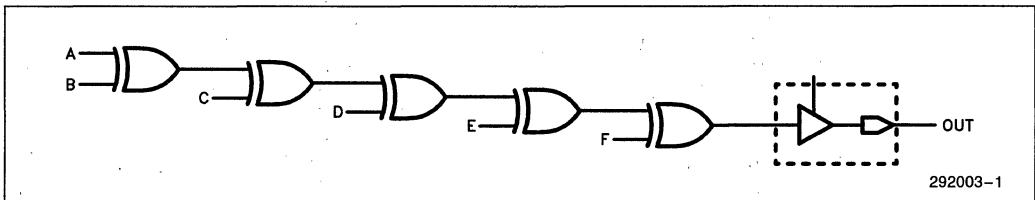


Figure 1. Cascaded Exclusive-ORs

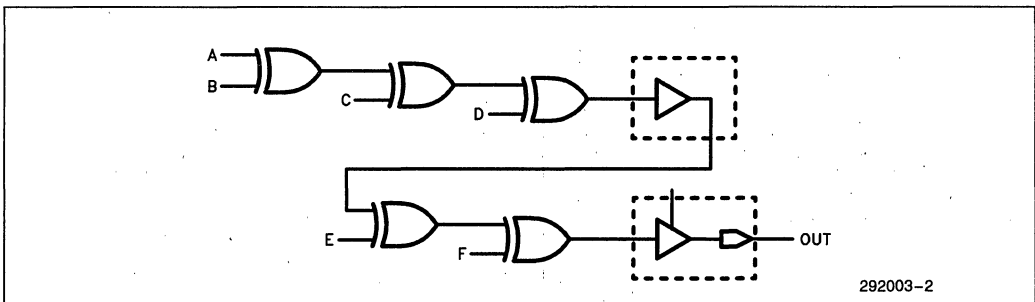


Figure 3. Cascaded Exclusive-ORs using Combinational Feedback



Logic Optimizing Compiler Utilization Report

\*\*\*\*\* Design implemented successfully

JRD  
INTEL  
October 8, 1985

1  
5C121  
CASCADING 5XORS WITH COMBINATIONAL FEEDBACK  
LB Version 3.0, Baseline 17x, 9/26/85

5C121

```

-----
GND -| 1 40|- Vcc
GND -| 2 39|- Vcc
GND -| 3 38|- Ap
GND -| 4 37|- Bp
GND -| 5 36|- Cp
GND -| 6 35|- Dp
GND -| 7 34|- Ep
GND -| 8 33|- Fp
GND -| 9 32|- O
GND -|10 31|- RESERVED
GND -|11 30|- RESERVED
GND -|12 29|- RESERVED
GND -|13 28|- GND
GND -|14 27|- GND
GND -|15 26|- GND
GND -|16 25|- GND
GND -|17 24|- GND
GND -|18 23|- GND
GND -|19 22|- GND
GND -|20 21|- GND
-----

```

\*\*INPUTS\*\*

Name	Pin	Resource	MCell #	PTerms	MCells	Feeds:		
						OE	Clear	Clock
Fp	33	INP	-	-	1	-	-	-
Ep	34	INP	-	-	1	-	-	-
Dp	35	INP	-	-	13	-	-	-
Cp	36	INP	-	-	13	-	-	-
Bp	37	INP	-	-	13	-	-	-
Ap	38	INP	-	-	13	-	-	-

\*\*OUTPUTS\*\*

Name	Pin	Resource	MCell #	PTerms	MCells	Feeds:	
						OE	Clear
O	32	CONF	1	4/ 4	-	-	-

```

**BURIED REGISTERS**

```

Name	Pin	Resource	MCell #	PTerms	MCells	Feeds: OE	Clear
	-	NOCF	13	8/ 8	1	-	-

```

**UNUSED RESOURCES**

```

Name	Pin	Resource	MCell	PTerms
-	1	-	-	-
-	2	-	-	-
-	3	-	-	-
-	4	-	-	-
-	5	-	-	-
-	6	-	-	-
-	7	-	-	-
-	8	-	28	4
-	9	-	27	10
-	10	-	26	8
-	11	-	25	6
-	12	-	24	6
-	13	-	23	8
-	14	-	22	10
-	15	-	21	4
-	16	-	20	12
-	17	-	19	4
-	18	-	18	8
-	19	-	17	8
-	21	-	12	8
-	22	-	11	8
-	23	-	10	4
-	24	-	9	12
-	25	-	8	4
-	26	-	7	10
-	27	-	6	8
-	28	-	5	6
-	29	-	4	6
-	30	-	3	8
-	31	-	2	10
-	NA	-	14	8
-	NA	-	15	8
-	NA	-	16	8

```

**PART UTILIZATION**
18% Pins
7% MacroCells
5% Pterms

```

292003-4

Figure 5. The Utilization Report



# APPLICATION BRIEF

AB-9

May 1986

## **5C121 As A Three And One Half Digit Display Driver**

**THOM BOWNS**  
PROGRAMMABLE LOGIC APPLICATIONS  
INTEL CORPORATION

Order Number: 292006-001

## INTRODUCTION

Described is a method of constructing a multi-digit, seven segment decoder driver with latching capability in a single EPLD. The design is a simple, easily understood method of using the 5C121 as a seven-segment display driver.

This design has many advantages: (1) the ability to update a single digit without disturbing the others, (2) Outputs are latched and retain their data without update from the controlling device(s), (3) Input interfacing is simple and straightforward, using four data inputs, two digit select lines, and a data strobe line.

The display driver interface is therefore not limited to microprocessor applications only (although it can be used with them). Possible applications include a Multimeter display, a clock or timer display, or a simple controller system display.

## PROBLEM

The display driver needs to latch the incoming data at the correct time, route it to the correct digit, and then decode the four bit data into seven-segment output format.

## SOLUTION IN EPLD

A simple solution to the display driver imagined above can be realized in the 5C121 EPLD.

The 5C121 EPLD is organized in groups of Macrocells. Each Macrocell contains a number of multiple input AND gates which are feeding an OR gate. The OR gate feeds a selectable registered output. This output may also be routed back into the array for feedback purposes.

Figure 1 shows a basic block diagram of the three and one half digit display driver. The data is input to a distribution block, which sends the data to one of four seven-segment decoders depending upon the digit selected by the Digit Select inputs. The outputs are updated by strobing the WR input. The data input is in a HEX format and may be in the range of 0 to F HEX (0 to 15 Decimal). Digit select is placed upon the two select lines in a binary format; 0, 1, 2, 3. When data is present on the input lines and a digit is selected, the strobe line may be pulsed high and that output digit is then updated to the numeral suggested by the input data.

Figure 2 illustrates the Boolean equivalents of the design in Figure 1. In the NETWORK section of Figure 2, the inputs and outputs of the design are described.

For instance, the NETWORK equation

$$SSA1, SA1F = RORF (ISA1, WRN, GND, GND, VCC)$$

represents that the output pin for segment "A" of the 1st Seven Segment display (SSA1) results from a Registered Output Registered Feedback (RORF) structure in the EPLD. The feedback signal (SA1F) is the same as the signal output (SSA1). The RORF's D input is driven by the signal ISA1, the clock input is driven by WRN, and reset, preset and output enable signals are tied to their default voltage levels (either GND or VCC).

The EQUATION section of Figure 2 shows how the data distribution and decoding logic works. Equations starting with A-G are generic seven segment display equations. Segment decoding results from the combination of the true or false of the four data inputs (e.g., D0 or !D0).

Equations such as

$$SE1 = (E * WE1) + (SE1F * !WE1)$$

show how the data is distributed. Segment E of display 1 (SE1) is valid (ON) if the "E" decode exists and display 1 is chosen by the address inputs ( $WE1 = !A0 * !A1$ ). It is also valid if it was previously turned on (SE1F) AND seven segment display 1 is not selected (!WE1).

These equations may be entered using LB in the form of a Netlist, or may be entered directly into the ADF by means of a text editor. The ADF is then compiled and programmed into a 5C121 using IPLS.

## SUMMARY

This method of using the 5C121 as a three and one half digit display driver is advantageous in respect to its simple interface, and its ability to hold all other digits stable while one is being updated. Displays with more than three and one half digits may be produced in the 5C121 by using the input latches as data storage and by multiplexing the outputs in a scanning fashion.

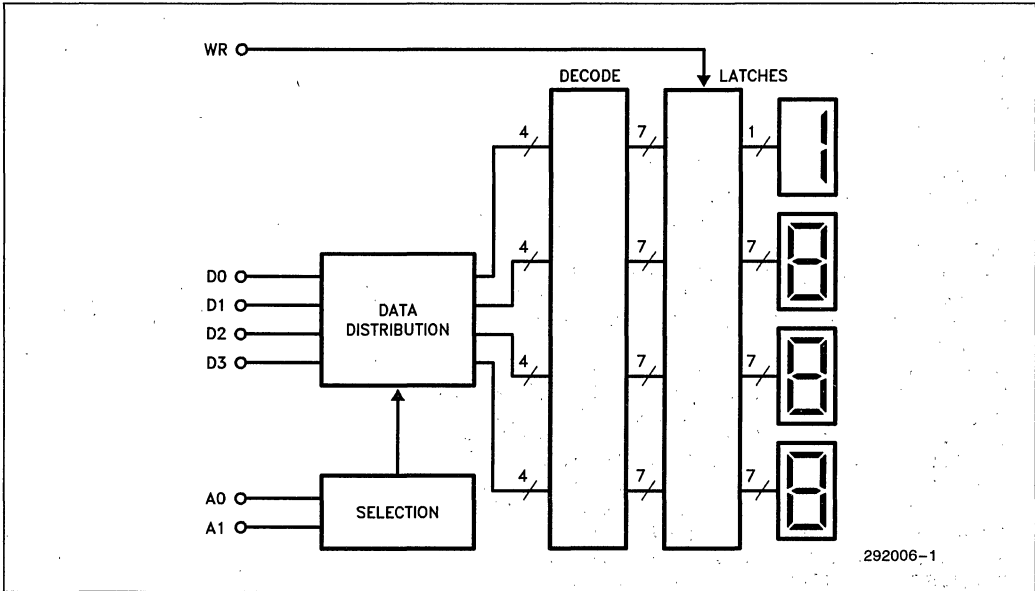


Figure 1. Block Diagram



Thom Bowns  
 Intel  
 October 29, 1985  
 U4  
 1  
 5C121  
 3.5 digit output driver  
 LB Version 3.0, Baseline 17x, 9/26/85  
 PART: 5C121  
 INPUTS: AOp,A1p,D0p,D1p,D2p,D3p,WRp  
 OUTPUTS: SSA1,SSB1,SSC1,SSD1,SSE1,SSF1,SSG1,SSA2,  
 SSB2,SSC2,SSD2,SSE2,SSF2,SSG2,SSA3,SSB3,SSC3,SSD3,SSE3,SSF3,SSG3,SSA4  
 NETWORK:  
 SSA1,SA1F = RORF (ISA1,WRN,GND,GND,VCC)  
 SSB1,SB1F = RORF (ISB1,WRN,GND,GND,VCC)  
 SSC1,SC1F = RORF (ISC1,WRN,GND,GND,VCC)  
 SSD1,SD1F = RORF (ISD1,WRN,GND,GND,VCC)  
 SSE1,SE1F = RORF (SE1,WRN,GND,GND,VCC)  
 SSF1,SF1F = RORF (SF1,WRN,GND,GND,VCC)  
 SSG1,SG1F = RORF (SG1,WRN,GND,GND,VCC)  
 SSA2,SA2F = RORF (SA2,WRN,GND,GND,VCC)  
 SSB2,SB2F = RORF (SB2,WRN,GND,GND,VCC)  
 SSC2,SC2F = RORF (SC2,WRN,GND,GND,VCC)  
 SSD2,SD2F = RORF (SD2,WRN,GND,GND,VCC)  
 SSE2,SE2F = RORF (SE2,WRN,GND,GND,VCC)  
 SSF2,SF2F = RORF (SF2,WRN,GND,GND,VCC)  
 SSG2,SG2F = RORF (SG2,WRN,GND,GND,VCC)  
 SSA3,SA3F = RORF (SA3,WRN,GND,GND,VCC)  
 SSB3,SB3F = RORF (SB3,WRN,GND,GND,VCC)  
 SSC3,SC3F = RORF (SC3,WRN,GND,GND,VCC)  
 SSD3,SD3F = RORF (SD3,WRN,GND,GND,VCC)  
 SSE3,SE3F = RORF (SE3,WRN,GND,GND,VCC)  
 SSF3,SF3F = RORF (SF3,WRN,GND,GND,VCC)  
 SSG3,SG3F = RORF (SG3,WRN,GND,GND,VCC)  
 SSA4,SA4F = RORF (SA4,WRN,GND,GND,VCC)  
 ISA1 = NOCF (SA1)  
 ISB1 = NOCF (SB1)  
 ISC1 = NOCF (SC1)  
 ISD1 = NOCF (SD1)  
 WRN = NOT (WR)  
 WR = INP (WRp)  
 D0 = INP (D0p)  
 D1 = INP (D1p)  
 D2 = INP (D2p)  
 D3 = INP (D3p)  
 A0 = INP (A0p)  
 A1 = INP (A1p)  
 EQUATIONS:  
 A = !D3\*!D2\*!D1\*DO + !D3\*D2\*!D1\*!DO + D3\*!D2\*D1\*DO + D3\*D2\*!D1\*DO;  
 B = !D3\*D2\*!D1\*DO + D2\*D1\*!DO + D3\*D2\*!D1\*!DO + D3\*D1\*DO;  
 C = !D3\*!D2\*D1\*!DO + D3\*D2\*!D1\*!DO + D3\*D2\*D1;  
 D = !D3\*!D2\*!D1\*DO + !D3\*D2\*!D1\*!DO + D2\*D1\*DO + D3\*!D2\*D1\*!DO;  
 E = !D3\*!D2\*DO + !D3\*D2\*!D1 + !D3\*D2\*D1\*DO + D3\*!D2\*!D1\*DO;  
 F = !D3\*!D2\*!D1\*DO + !D3\*!D2\*D1 + !D3\*D2\*D1\*DO + D3\*D2\*!D1\*DO;  
 G = !D3\*!D2\*!D1 + !D3\*D2\*D1\*DO + D3\*D2\*!D1\*!DO;

292006-2

Figure 2. ADF Listing

```
SE1 = (E * WE1)
      + (SE1F * !WE1);
SF1 = (F * WE1)
      + (SF1F * !WE1);
SG1 = (G * WE1)
      + (SG1F * !WE1);
SA2 = (A * WE2)
      + (SA2F * !WE2);
SB2 = (B * WE2)
      + (SB2F * !WE2);
SC2 = (C * WE2)
      + (SC2F * !WE2);
SD2 = (D * WE2)
      + (SD2F * !WE2);
SE2 = (E * WE2)
      + (SE2F * !WE2);
SF2 = (F * WE2)
      + (SF2F * !WE2);
SG2 = (G * WE2)
      + (SG2F * !WE2);
SA3 = (A * WE3)
      + (SA3F * !WE3);
SB3 = (B * WE3)
      + (SB3F * !WE3);
SC3 = (C * WE3)
      + (SC3F * !WE3);
SD3 = (D * WE3)
      + (SD3F * !WE3);
SE3 = (E * WE3)
      + (SE3F * !WE3);
SF3 = (F * WE3)
      + (SF3F * !WE3);
SG3 = (G * WE3)
      + (SG3F * !WE3);
SA1 = (A * WE1)
      + (SA1F * !WE1);
SB1 = (B * WE1)
      + (SB1F * !WE1);
SC1 = (C * WE1)
      + (SC1F * !WE1);
SD1 = (D * WE1)
      + (SD1F * !WE1);
SA4 = ((!D3*!D2*!D1*!D0) * WE4) + (SA4F * !WE4);
WE1 = !AO * !A1;
WE2 = AO * !A1;
WE3 = !AO * A1;
WE4 = AO * A1;
END$
```

292006-3

Figure 2. ADF Listing (Continued)

June 1986

# **Square Pegs in Round Holes—A Fitting Tutorial for the 5C121**

**J. R. DONNELL**  
PROGRAMMABLE LOGIC APPLICATIONS  
INTEL CORPORATION

Order Number: 292014-001

## INTRODUCTION

This application brief explores the various techniques for getting the most out of Intel's line of Erasable Programmable Logic Devices (EPLDs). In many cases, techniques discussed here will not be needed due to the intelligent fitting algorithms built into Intel's Programmable Logic Software (iPLS). As a matter of fact, most designs can be implemented in EPLDs without any knowledge of the device architectures. For complex designs, the designer will still need an in-depth understanding of the target EPLD in order to maximize the EPLD's utility.

This application brief explores fitting techniques for the 5C121, a 1200 gate equivalent CHMOS EPLD. The techniques described here will also apply to any EPLD that supports a similar architecture.

## FITTING

When fitting logic designs into the 5C121 there are two typical scenarios: 1) The 5C121 design has been completed without pin assignments and the compiler warns the user that fitting may be time consuming, and 2) pin assignments have been made and the "\*\*\*\*ERR-FIT ... " message comes up.

Let's look at the first situation.

In general, if the designer does not care what signals get assigned to what pins, the choice can be left to the compiler and the compiler will make pin assignments. For simple designs pin assignments are very easy. However, designs that include a variety of different register types, feedback paths, and product term widths may take a long time for the compiler to fit. When the designer is faced with the message, "Fitting may be time consuming", the compilation should be aborted, and intelligent pin assignments made. NOTE: Control C (^C) may be used to abort a design. The software will not stop immediately because the software does not poll the keyboard until it updates the display. Rebooting the system will also work.

To make intelligent pin assignments, the designer needs a basic understanding of the architecture of the part. For the 5C121 this understanding should include the number of product terms supported in each Macrocell, what Macrocells support local feedback, and what Macrocells support global feedback. This information is easily found in the data sheet. One other point, the Macrocells in the 5C121 are grouped into groups of four. All Macrocells in a group must have the same output type. Therefore, if one output is registered, the other three must also be registered. This means that a combinatorial output could not be put into the same group as a registered output. Output enable (OE) terms are also based on Macrocell grouping. All four Macrocells are driven from the same OE term.

Once the basic 5C121 architecture is understood, intelligent pin assignments can be made. After assigning the pins recompile the design using iPLS.

Compiling the design with pin assignments is a new ball game. This time it is fit or not fit. If the design does not fit, an error like: "\*\*\*\*ERR-FIT—It is not possible to fit the specific pin requests you made" will occur. In most cases, the compiler will also ask if it can remove pin assignments and try its own. If the design has already been attempted without pin assignments, or if specific pin assignments are needed, answer no and isolate the problem.

## ISOLATE THE PROBLEM

The first step towards isolating the problem is to print out a copy of the utilization report (<Filename>.RPT), logic equation file (<Filename>.LEF), and the Advanced Design File (<Filename>.ADF). Next, fill out the 5C121 architecture worksheet included in this application brief. Include the signal name for each pin, the type of output, and the number of product terms needed for each output. All this information is available in the files that were printed earlier. The next step is to identify the conflict.

## CONFLICTS

There are three potential conflicts with pin assignments in the 5C121; incompatible output structures, excessive product terms, and local/global feedback conflicts. Incompatible output structures and excessive product term errors are the easiest to spot.

## INCOMPATIBLE OUTPUT STRUCTURES

As shown in the 5C121 Design Worksheet, the 5C121 is divided into six Macrocell groupings. The data sheet refers to these as the A-1, B-1, A-2, B-2, A-3, and B-3 Macrocells. One requirement of the 5C121 architecture is that Macrocells within the same grouping have the same output structure. This was discussed earlier, but it is worth revisiting. The file titled example 1 in the appendix shows an ADF for a design that contains such an I/O conflict. Following the ADF is a completed 5C121 architecture worksheet with a number of problems. Concentrating on the incompatible output problem on the 5C121 worksheet, notice that pins 31 and 32 belong to the same Macrocell group, and that they are assigned conflicting I/O structures.

The solution to an incompatible output structure conflict may be as simple as reassigning pins. Another option may be to use a different output type for that sig-

nal. This is very dependent on the design. Another option is possible when a Macrocell grouping has been assigned combinatorial output structure, and a registered output needs to be assigned to that same group. A possible solution is to use one of the buried registers configured as a NORF (No Output Registered Feedback) cell to hold the signal, and then send the signal out through a CONF (Combinatorial Output No Feedback) primitive. This output primitive is compatible with the other output primitives in that grouping, and the register output requirement has also been satisfied. The penalty is loss of speed due to the additional feedback path.

## EXCESSIVE PRODUCT TERMS

Excessive product term conflicts are also easy to spot. (A product term consists of a set of signals ANDed together which are separated from other ANDed groups by an OR gate.) Written next to the I/O slot on the 5C121 architecture worksheet is the number of product terms that each Macrocell supports. Match that number with the number of product terms for each output indicated in the logic equation file (LEF). If more product terms are required of an output than are provided, there is a product term conflict. The utilization report also shows the number of product terms used for each signal.

The solution, again, may be as simple as reassigning pins since the 5C121 supports varying product term widths. In fact, the 5C121 supports up to 16 product terms on pins 16 and 24. Note that four of those product terms are shared with the adjacent Macrocell. Sharing means that those signals are common. It is not product term allocation. If the number of product terms exceeds the capability of the device, the design may still fit by splitting up long equations and inserting NOCF (No Output Combinatorial Feedback) primitives. Again the price for using this solution is reduced speed. This technique is covered more thoroughly in AB-8 titled: Implementing Cascaded Logic in the 5C121.

## LOCAL/GLOBAL FEEDBACK

It is possible to encounter one other type of fitting conflict in the 5C121. This occurs when a feedback signal from the A-1 or A-2 Macrocells feeds the B-1 or B-2 Macrocells. The issue is that these Macrocells feed buses that are local to one half of the chip. Therefore, the signal is not physically available to the other side of the device.

The best way to understand the local and global bussing in the 5C121 is to divide the chip in half lengthwise. One side contains the A Macrocells, and the other side

contains the B Macrocells. The two sides are mirror images. Speaking generically now, the -1 and -2 Macrocells feed only local busses; local to their respective side of the device. The -3 Macrocells and the buried registers feed global busses which route signals to both sides of the device. Therefore a feedback signal coming from the A-1 or A-2 group can only feed the A Macrocells, however, a feedback signal from the A-3 group could feed the B-1, B-2, B-3, or the B buried Macrocells. This local/global bussing applies to both feedback and input signals on the I/O pins. All of the dedicated inputs feed the global bus.

Example 1 also shows a simple two bit counter with seven segment driver outputs. The worksheet shows that the counter registers were assigned to pins 27 and 28, while the seven segment outputs were assigned to pins 8 thru 14. The seven segment outputs decode the feedback signals from the counter registers to generate the appropriate digit output, and therefore must have access to those signals. This presents a local/global feedback conflict. If the designer is locked into those specific pin assignments a design workaround is needed.

One solution might be to take the outputs of the counter and externally tie them to dedicated input pins thereby making those signals global. This would work but that solution ends up wasting input pins. A better solution would be to internally route the counter feedback signals through one of the buried registers configured as a NOCF primitive. After passing through the buried register the signals become global. Both the incompatible output solution and this solution are shown in the worksheet, ADF, and utilization report shown as example 2. If we did not need the counter signals externally, it would of been wise to simply use the buried registers to perform the counting function.

One final comment regarding the utilization report. The utilization report shown in example 1 indicates that signals CLK and CNT feed Macrocell 1001 and 1002. These are fictitious Macrocell numbers that the software assigns to requests that cannot be met. In example 1, three requests were unfulfilled: REGOUT, LED1 and LED0. REGOUT was unfulfilled because of incompatible output structures. LED0 and LED1 were unfulfilled because their feedback signals needed to drive the seven segment display outputs. This was impossible because the LED outputs were assigned to a local bus on the opposite side of the device.

The files shown in example 2 fix the LED fitting problems by sending the feedback signals through the buried registers, thereby making them global. In the case of REGOUT, the buried register primitive NORF (No Output Registered Feedback) is used, allowing the output primitive to be combinatorial.

## EXAMPLE 1

### ADF

```

JR Donnell
Intel
April 3, 1986

0
5C121
Fitting example
LB Version 3.0, Baseline 17x, 9/26/85
PART: 5C121
INPUTS: CNT@2,CLK@1
OUTPUTS: LED0@28,LED1@27,REGOUT@32,CONFOUT@31,SEGA@8,
          SEGB@9,SEGC@10,SEGD@11,SEGE@12,SEGF@13,SEGG@14

NETWORK:
LED0,A = RORF (NLED0D,CLK,GND,GND,VCC)
LED1,B = RORF (NLED1D,CLK,GND,GND,VCC)
REGOUT = RONF (NREGOUTD,CLK,GND,GND,VCC)
CONFOUT = CONF (NCONFOUTIN,VCC)
SEGA = CONF (NSEGAIN,VCC)
SEGB = CONF (NSEGBIN,VCC)
SEGC = CONF (NSEGCIN,VCC)
SEGD = CONF (NSEGDIN,VCC)
SEGE = CONF (NSEGEIN,VCC)
SEGF = CONF (NSEGFIN,VCC)
SEGG = CONF (NSEGGIN,VCC)
CLK = INP (CLK)
CNT = INP (CNT)
EQUATIONS:
NSEGGIN = 2
          + 3;
2 = B*/A;
3 = A*B;
NLED1D = /A*/B*CNT
          + /A*B*/CNT
          + A*/B*CNT
          + A*B*/CNT;
NLED0D = /A*B*CNT
          + A*/B*/CNT
          + A*/B*CNT
          + A*B*/CNT;
NSEGFIN = 0;
0 = /B*/A;
NSEGBIN = 0
          + 2;
NSEGDIN = 0
          + 2
          + 3;
NSEGCIN = 0
          + 1
          + 3;
1 = /B*A;
NSEGBIN = 0
          + 1
          + 2
          + 3;
NSEGAIN = 0
          + 2
          + 3;
NCONFOUTIN = A*B;
NREGOUTD = /A*/B;
END$

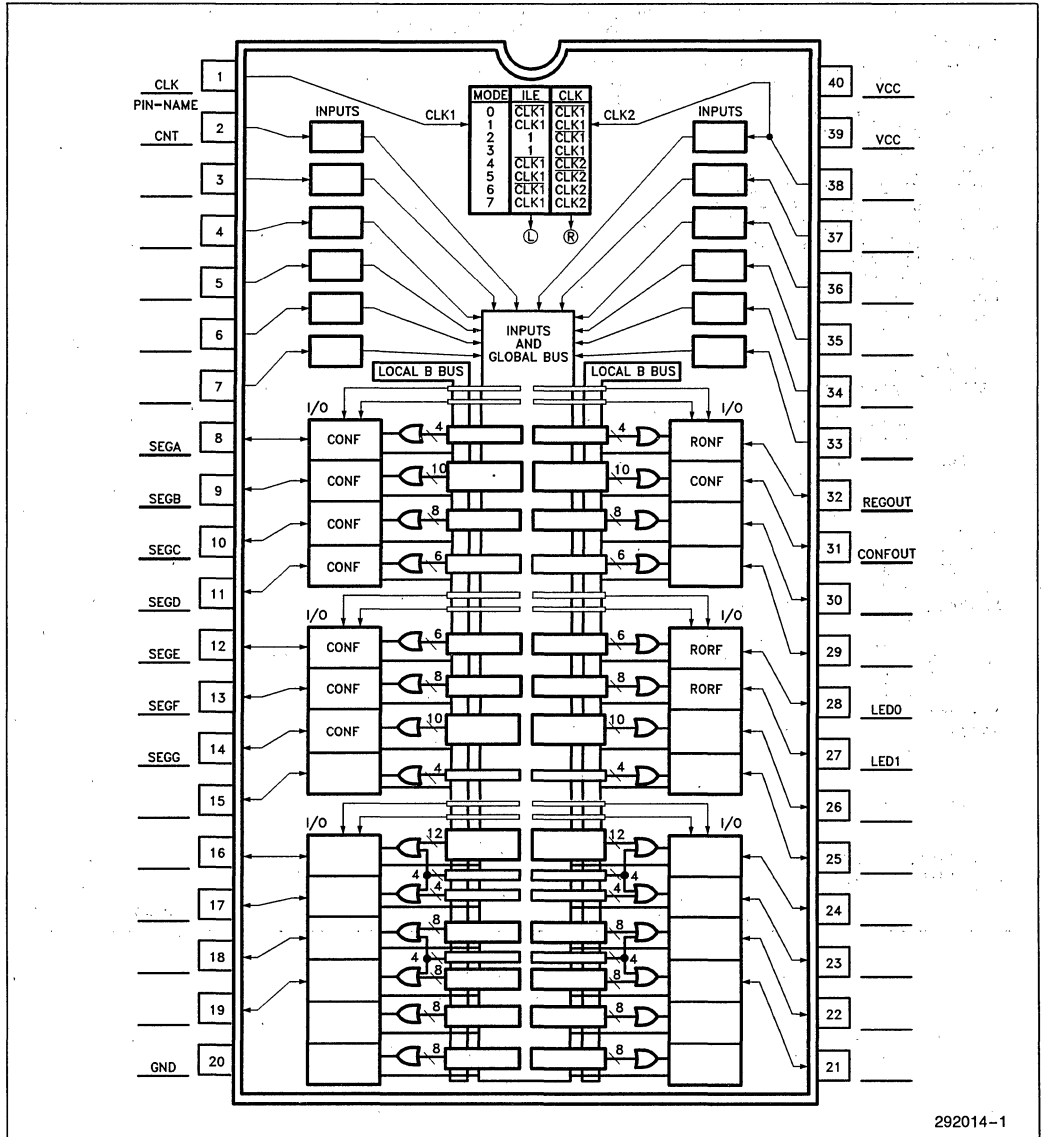
```

292014-2

**SUMMARY**

As programmable logic devices become more dense, signal routing and resource partitioning becomes necessary. In general, these choices are made by the semiconductor manufacture to most efficiently utilize the available logic. In some cases though, these choices make certain designs more difficult to implement in a given device. Intelligent software, a basic knowledge of the device architecture, and a little experience in fitting techniques will always make the job easier.

**EXAMPLE 1 (Continued)**  
5C121 Design Worksheet



EXAMPLE 1 (Continued)

Logic Optimizing Compiler Utilization Report

\*\*\*\* Unable to implement design

JR Donnell  
Intel  
April 3, 1986

0  
5C121  
Fitting example  
LB Version 3.0, Baseline 17x, 9/26/85

```

          5C121
          - - -
CLK - : 1  40:- Vcc
CNT - : 2  39:- Vcc
GND - : 3  38:- GND
GND - : 4  37:- GND
GND - : 5  36:- GND
GND - : 6  35:- GND
GND - : 7  34:- GND
SEGA - : 8  33:- GND
SEGB - : 9  32:- RESERVED
SEGC -:10  31:- CONFOUT
SEGD -:11  30:- RESERVED
SEGE -:12  29:- RESERVED
SEGF -:13  28:- GND
SEGG -:14  27:- GND
RESERVED -:15  26:- GND
GND -:16  25:- GND
GND -:17  24:- GND
GND -:18  23:- GND
GND -:19  22:- GND
GND -:20  21:- GND
          - - -
    
```

\*\*INPUTS\*\*

Name	Pin	Resource	MCell #	PTerms	MCells	Feeds:		
						OE	Clear	Clock
CLK	1	INP	-	-	-	-	-	Reg
CNT	2	INP	-	-	1001 1002	-	-	-

\*\*OUTPUTS\*\*

Name	Pin	Resource	MCell #	PTerms	MCells	Feeds:	
						OE	Clear
SEGA	8	CONF	28	2/ 4	-	-	-
SEGB	9	CONF	27	2/10	-	-	-
SEGC	10	CONF	26	2/ 8	-	-	-
SEGD	11	CONF	25	2/ 6	-	-	-



EXAMPLE 1 (Continued)

SEGE	12	CONF	24	1/ 6	-	-	-
SEGF	13	CONF	23	1/ 8	-	-	-
SEGG	14	CONF	22	1/10	-	-	-
CONFOUT	31	CONF	2	1/10	-	-	-

\*\*UNFULFILLED REQUESTS\*\*

\*\*OUTPUTS\*\*

Name	Pin	Resource	MCell #	PTerms	MCells	Feeds:	
						OE	Clear
REGOUT	-	RONF	1000	1	-	-	-
LED1	-	RORF	1001	2	2	-	-
					22		
					23		
					25		
					26		
					28		
					1000		
					1001		
					1002		
LEDO	-	RORF	1002	3	2	-	-
					23		
					24		
					25		
					26		
					27		
					28		
					1000		
					1002		

\*\*UNUSED RESOURCES\*\*

Name	Pin	Resource	MCell	PTerms
-	3	-	-	-
-	4	-	-	-
-	5	-	-	-
-	6	-	-	-
-	7	-	-	-
-	15	-	21	4
-	16	-	20	12
-	17	-	19	4
-	18	-	18	8
-	19	-	17	8
-	21	-	12	8
-	22	-	11	8
-	23	-	10	4
-	24	-	9	12
-	25	-	8	4
-	26	-	7	10
-	27	-	6	8
-	28	-	5	6
-	29	-	4	6
-	30	-	3	8
-	32	-	1	4
-	33	-	-	-
-	34	-	-	-
-	35	-	-	-
-	36	-	-	-
-	37	-	-	-
-	38	-	-	-
-	NA	-	13	8
-	NA	-	14	8
-	NA	-	15	8
-	NA	-	16	8

292014-4

292014-5

## EXAMPLE 2

### ADF

```

JR Donnell
Intel
April 3, 1986

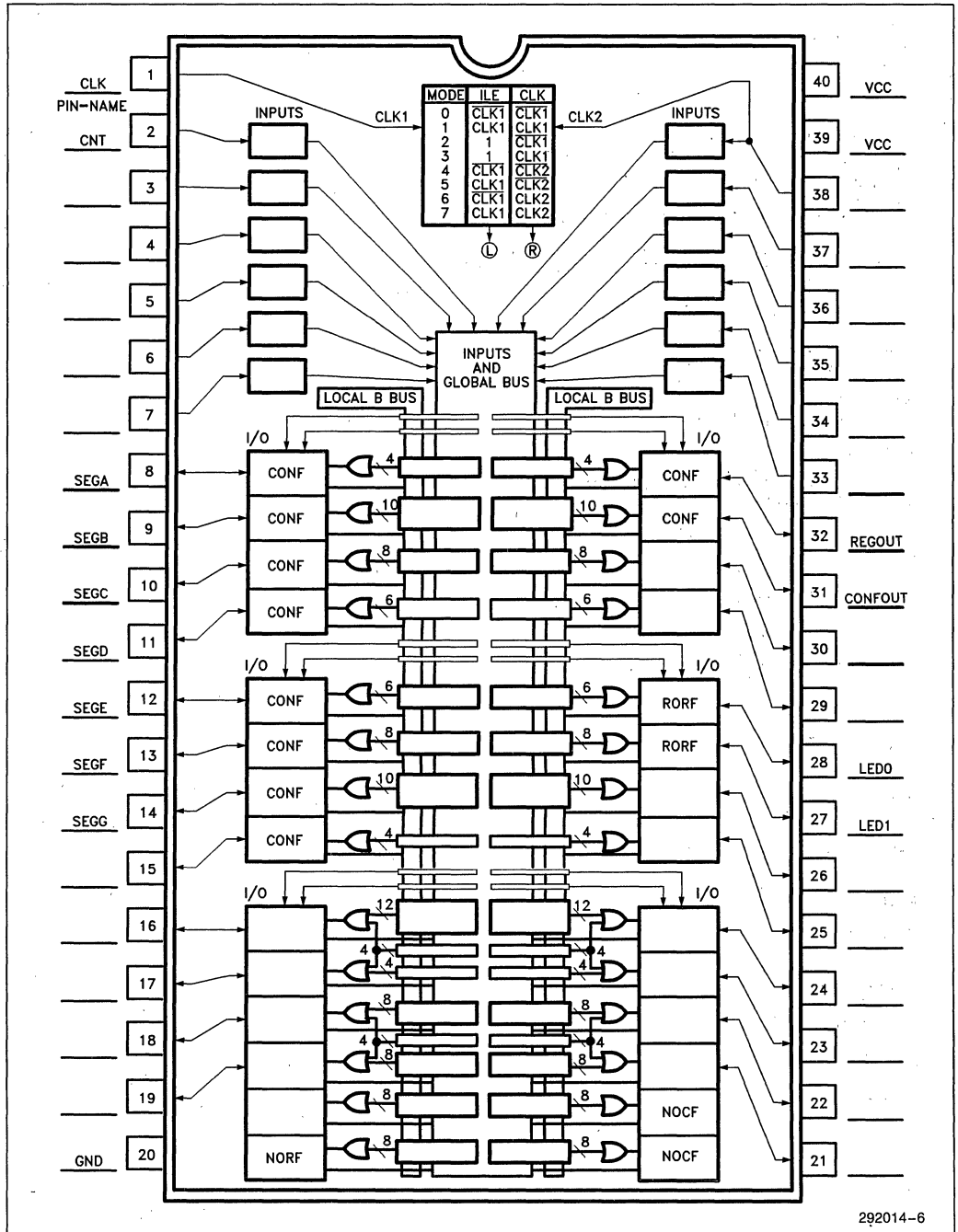
0
5C121
Fitting example
LB Version 3.0, Baseline 17x, 9/26/85
PART: 5C121
INPUTS: CNT@2, CLK@1
OUTPUTS: LED0@28, LED1@27, REGOUT@32, CONFOUT@31, SEGA@8,
        SEGB@9, SEGC@10, SEGD@11, SEGE@12, SEGF@13, SEGG@14

NETWORK:
LED0, NATONOCF = RORF (NLED0D, CLK, GND, GND, VCC)
LED1, NBTONOCF = RORF (NLED1D, CLK, GND, GND, VCC)
REGOUT = CONF (NREGOUTIN, VCC)
CONFOUT = CONF (NCONFOUTIN, VCC)
SEGA = CONF (NSEGAIN, VCC)
SEGB = CONF (NSEGBIN, VCC)
SEGC = CONF (NSEGCIN, VCC)
SEGD = CONF (NSEGDIN, VCC)
SEGE = CONF (NSEGIN, VCC)
SEGF = CONF (NSEGFIN, VCC)
SEGG = CONF (NSEGGIN, VCC)
A = NOCF (NATONOCF)
CLK = INP (CLK)
B = NOCF (NBTONOCF)
NREGOUTIN = NORF (NREGOUTD, CLK, GND, GND)
CNT = INP (CNT)
EQUATIONS:
NLED0D = /A*B*CNT
        + A*/B*/CNT
        + A*/B*/CNT
        + A*B*/CNT;
NLED1D = /A*/B*/CNT
        + /A*B*/CNT
        + A*/B*/CNT
        + A*B*/CNT;
NCONFOUTIN = A*B;
NSEGAIN = 0
        + 2
        + 3;
NSEGBIN = 0
        + 1
        + 2
        + 3;
NSEGCIN = 0
        + 1
        + 3;
NSEGDIN = 0
        + 2
        + 3;
NSEGIN = 0
        + 2;
NSEGFIN = 0;
NSEGGIN = 2
        + 3;
NREGOUTD = /A*/B;
2 = B*/A;
3 = A*B;
0 = /B*/A;
1 = /B*A;
END$

```

292014-7

**EXAMPLE 2** (Continued)  
5C121 Design Worksheet



**EXAMPLE 2 (Continued)**

Logic Optimizing Compiler Utilization Report

\*\*\*\* Design implemented successfully

JR Donnell  
Intel  
April 3, 1986

0  
5C121  
Fitting example  
LB Version 3.0, Baseline 17x, 9/26/85

5C121

```

    CLK -- 1  40:- Vcc
    CNT -- 2  39:- Vcc
    GND -- 3  38:- GND
    GND -- 4  37:- GND
    GND -- 5  36:- GND
    GND -- 6  35:- GND
    GND -- 7  34:- GND
    SEGA -- 8  33:- GND
    SEGB -- 9  32:- HEGOUT
    SEGC --10  31:- CONFOUT
    SEGD --11  30:- RESERVED
    SEGE --12  29:- RESERVED
    SEGF --13  28:- LED0
    SEGG --14  27:- LED1
    RESERVED --15 26:- RESERVED
    GND --16  25:- RESERVED
    GND --17  24:- GND
    GND --18  23:- GND
    GND --19  22:- GND
    GND --20  21:- GND
  
```

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

Name	Pin	Resource	MCell #	PTerms	MCells	Feeds:		
						OE	Clear	Clock
CLK	1	INP	-	-	-	-	-	Reg
CNT	2	INP	-	-	5 6	-	-	-

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

Name	Pin	Resource	MCell #	PTerms	MCells	Feeds:	
						OE	Clear
SEGA	8	CONF	28	2/ 4	-	-	-
SEGB	9	CONF	27	2/10	-	-	-
SEGC	10	CONF	26	2/ 8	-	-	-
SEGD	11	CONF	25	2/ 6	-	-	-

292014-8

**EXAMPLE 2 (Continued)**

SEGE	12	CONF	24	1/ 6	-	-	-
SEGF	13	CONF	23	1/ 8	-	-	-
SEGG	14	CONF	22	1/10	-	-	-
LED1	27	RORF	6	2/ 8	13	-	-
LED0	28	RORF	5	3/ 6	14	-	-
CONFOUT	31	CONF	2	1/10	-	-	-
REGOUT	32	CONF	1	1/ 4	-	-	-

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

Name	Pin	Resource	MCell #	PTerms	MCells	Feeds:	OE	Clear
-	-	NOCF	13	1/ 8	2	-	-	-
					5			
					6			
					15			
					22			
					23			
					25			
					26			
					28			
-	-	NOCF	14	1/ 8	2	-	-	-
					5			
					15			
					23			
					24			
					25			
					26			
					27			
					28			
-	-	NORF	15	1/ 8	1	-	-	-

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

Name	Pin	Resource	MCell	PTerms
-	3	-	-	-
-	4	-	-	-
-	5	-	-	-
-	6	-	-	-
-	7	-	-	-
-	15	-	21	4
-	16	-	20	12
-	17	-	19	4
-	18	-	18	8
-	19	-	17	8
-	21	-	12	8
-	22	-	11	8
-	23	-	10	4
-	24	-	9	12
-	25	-	8	4
-	26	-	7	10
-	29	-	4	6

**EXAMPLE 2** (Continued)

-	30	-	3	8
-	33	-	-	-
-	34	-	-	-
-	35	-	-	-
-	36	-	-	-
-	37	-	-	-
-	38	-	-	-
-	NA	-	16	8

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

35% Pins  
50% MacroCells  
10% Pterms

292014-10



**APPLICATION  
BRIEF**

**AB-11**

February 1987

**16-Bit Binary Counter  
Implementation  
Using the 5C060 EPLD**

**KARL-HEINZ WEIGL  
INTEL CORPORATION  
MUNICH, GERMANY**

Order Number: 292015-002

## INTRODUCTION

System designers often use programmable logic devices to implement counters. Use of PLA devices lets the user build customized counters to suit individual applications. In most cases such counters are not available, 'off-the-shelf' SSI/MSI devices. In other applications, the PLA implementation allows the designer to squeeze the counter function along with other 'glue' tasks into a single PLA, with the attendant higher integration benefits.

Use of traditional 20-pin and 24-pin PLAs, however, does not allow for the construction of large counters having greater than 10 significant bits. This is because these traditional PLAs have register and product term restrictions (even the larger bipolar PLAs have only 8 to 10 registers and less than 8 product terms per register). In contrast, the 5C060 24-pin erasable programmable logic device (EPLD) contains 16 registers that are programmable as 'D', 'T', 'RS' or 'JK' types. These 16 programmable registers enable the construction of Up/Down counters with up to 16 significant bits.

This application brief details the implementation of a 16-bit binary counter in the 5C060 EPLD. The design also demonstrates efficient counter construction utilizing toggle flip-flops (T-FF) that allows for minimum product term utilization.

## DESIGN OBJECTIVE

The objective of the design is to implement a counter with the following features: (i) 16-bit binary count, (ii) toggle flip-flops, (iii) asynchronous clear, (iv) RUN/STOP function and (v) UP/DOWN function. The function table is shown in Figure 1.

RESET	UP/DOWN	RUN/STOP	Function
X	X	0	Inhibit Counting
0	0	1	Count Down
0	1	1	Count Up
1	X	X	Reset All Outputs to 'LOW'

Figure 1

## TOGGLE FLIP-FLOPS

Counters can be most effectively implemented in PLA architectures using toggle flip-flops. This is because counters constructed with 'D' type flip-flops require an additional product term for every successive significant bit, whereas toggle flip-flop implementation requires only one product term per significant bit. Thus, the toggle flip-flop counter design is more miserly in product term consumption than the 'D' register design. Since product term minimization is the key element to maximizing PLA utilization, the T-FF counter design is more efficient. The truth table for the toggle flip-flop is shown in Fig. 2.

T	Q(N)	Q (N + 1)
0	0	0
0	1	1
1	0	1
1	1	0

Figure 2

## SOLUTION

The 16-bit binary counter function was implemented in the 5C060 EPLD using the Intel Programmable Logic Development System (iPLDS). The equations for the 16-bit binary counter with the RESET, UP/DOWN and RUN/STOP functions are shown in the 'EQUATIONS' section of the LEF (Fig. 4). The pinout of the 5C060 with the implemented counter is shown in the RPT file (Utilization Report) Fig. 5. This RPT file also shows, under the 'OUTPUTS' section, that in each macrocell only one out of 8 product terms is used. In contrast the same 16-bit counter designed using 'D' type flip-flops would have required more than 16 product terms for the last significant bit.



```

INTEL CORPORATION
JAN. 15, 1987
1
1.0
5C060
BINARY 16-BIT UP/DOWN COUNTER WITH RUN/STOP AND ASYNCH. RESET USING T-FF

LB Version 4.01, Baseline 27.1 4/9/86
OPTIONS: TURBO=ON
PART: 5C060
INPUTS: RS,CLOCK,RESET,UD
OUTPUTS: Q0,Q1,Q2,Q3,Q4,Q5,Q6,Q7,Q8,Q9,QA,QB,QC,QD,QE,QF
NETWORK:
Q0,Q0F = TOTF (Q0T,CLK,CLR,GND,VCC)
Q1,Q1F = TOTF (Q1T,CLK,CLR,GND,VCC)
Q2,Q2F = TOTF (Q2T,CLK,CLR,GND,VCC)
Q3,Q3F = TOTF (Q3T,CLK,CLR,GND,VCC)
Q4,Q4F = TOTF (Q4T,CLK,CLR,GND,VCC)
Q5,Q5F = TOTF (Q5T,CLK,CLR,GND,VCC)
Q6,Q6F = TOTF (Q6T,CLK,CLR,GND,VCC)
Q7,Q7F = TOTF (Q7T,CLK,CLR,GND,VCC)
Q8,Q8F = TOTF (Q8T,CLK,CLR,GND,VCC)
Q9,Q9F = TOTF (Q9T,CLK,CLR,GND,VCC)
QA,QAF = TOTF (QAT,CLK,CLR,GND,VCC)
QB,QBF = TOTF (QBT,CLK,CLR,GND,VCC)
QC,QCF = TOTF (QCT,CLK,CLR,GND,VCC)
QD,QDF = TOTF (QDT,CLK,CLR,GND,VCC)
QE,QEF = TOTF (QET,CLK,CLR,GND,VCC)
QF = TONF (QFT,CLK,CLR,GND,VCC)
Q0T = OR (Q0U,Q0D)
CLK = INP (CLOCK)
CLR = INP (RESET)
Q1T = OR (Q1U,Q1D)
Q2T = OR (Q2U,Q2D)
Q3T = OR (Q3U,Q3D)
Q4T = OR (Q4U,Q4D)
Q5T = OR (Q5U,Q5D)
Q6T = OR (Q6U,Q6D)
Q7T = OR (Q7U,Q7D)
Q8T = OR (Q8U,Q8D)
Q9T = OR (Q9U,Q9D)
QAT = OR (QAU,QAD)
QBT = OR (QBU,QBD)
QCT = OR (QCU,QCD)
QDT = OR (QDU,QDD)
QET = OR (QEU,QED)
QFT = OR (QFU,QFD)
RS = INP (RS)
UD = INP (UD)
NUD = NOT (UD)
Q0U = AND (UD,RS)

```

292015-1

Figure 3. Example .ADF

```

Q1U = AND (UD, Q0F, Q0U)
Q2U = AND (UD, Q1F, Q1U)
Q3U = AND (UD, Q2F, Q2U)
Q4U = AND (UD, Q3F, Q3U)
Q5U = AND (UD, Q4F, Q4U)
Q6U = AND (UD, Q5F, Q5U)
Q7U = AND (UD, Q6F, Q6U)
Q8U = AND (UD, Q7F, Q7U)
Q9U = AND (UD, Q8F, Q8U)
QAU = AND (UD, Q9F, Q9U)
QBU = AND (UD, QAF, QAU)
QCU = AND (UD, QBF, QBU)
QDU = AND (UD, QCF, QCU)
QEU = AND (UD, QDF, QDU)
QFU = AND (UD, QEF, QEU)
NQ0F = NOT (Q0F)
NQ1F = NOT (Q1F)
NQ2F = NOT (Q2F)
NQ3F = NOT (Q3F)
NQ4F = NOT (Q4F)
NQ5F = NOT (Q5F)
NQ6F = NOT (Q6F)
NQ7F = NOT (Q7F)
NQ8F = NOT (Q8F)
NQ9F = NOT (Q9F)
NQAF = NOT (QAF)
NQBF = NOT (QBF)
NQCF = NOT (QCF)
NQDF = NOT (QDF)
NQEF = NOT (QEF)
Q0D = AND (NUD, RS)
Q1D = AND (NUD, NQ0F, Q0D)
Q2D = AND (NUD, NQ1F, Q1D)
Q3D = AND (NUD, NQ2F, Q2D)
Q4D = AND (NUD, NQ3F, Q3D)
Q5D = AND (NUD, NQ4F, Q4D)
Q6D = AND (NUD, NQ5F, Q5D)
Q7D = AND (NUD, NQ6F, Q6D)
Q8D = AND (NUD, NQ7F, Q7D)
Q9D = AND (NUD, NQ8F, Q8D)
QAD = AND (NUD, NQ9F, Q9D)
QBD = AND (NUD, NQAF, QAD)
QCD = AND (NUD, NQBF, QBD)
QDD = AND (NUD, NQCF, QCD)
QED = AND (NUD, NQDF, QED)
QFD = AND (NUD, NQEF, QED)
ENDS

```

292015-2

Figure 3. Example .ADF (Continued)

INTEL CORPORATION  
JAN. 15, 1987

1  
1.0  
5C060

BINARY 16-BIT UP/DOWN COUNTER WITH RUN/STOP AND ASYNCH. RESET USING T-FF

LEB Version 4.01, Baseline 27.1 4/9/86

LEF Version 4.01 Baseline 22.2 2/4/86

OPTIONS: TURBO=ON

PART:

5C060

INPUTS:

RS, CLOCK, RESET, UD

OUTPUTS:

Q0, Q1, Q2, Q3, Q4, Q5, Q6, Q7, Q8, Q9, QA, QB, QC, QD, QE, QF

NETWORK:

CLK = INP(CLOCK)

RS = INP(RS)

CLR = INP(RESET)

UD = INP(UD)

Q0, Q0F = TOTF(Q0T, CLK, CLR, GND, VCC)

Q1, Q1F = TOTF(Q1T, CLK, CLR, GND, VCC)

Q2, Q2F = TOTF(Q2T, CLK, CLR, GND, VCC)

Q3, Q3F = TOTF(Q3T, CLK, CLR, GND, VCC)

Q4, Q4F = TOTF(Q4T, CLK, CLR, GND, VCC)

Q5, Q5F = TOTF(Q5T, CLK, CLR, GND, VCC)

Q6, Q6F = TOTF(Q6T, CLK, CLR, GND, VCC)

Q7, Q7F = TOTF(Q7T, CLK, CLR, GND, VCC)

Q8, Q8F = TOTF(Q8T, CLK, CLR, GND, VCC)

Q9, Q9F = TOTF(Q9T, CLK, CLR, GND, VCC)

QA, QAF = TOTF(QAT, CLK, CLR, GND, VCC)

QB, QBF = TOTF(QBT, CLK, CLR, GND, VCC)

QC, QCF = TOTF(QCT, CLK, CLR, GND, VCC)

QD, QDF = TOTF(QDT, CLK, CLR, GND, VCC)

QE, QEF = TOTF(QET, CLK, CLR, GND, VCC)

QF = TONF(QFT, CLK, CLR, GND, VCC)

EQUATIONS:

QFT = UD' \* QEF' \* QDF' \* QCF' \* QBF' \* QAF' \* Q9F' \* Q8F' \* Q7F' \* Q6F' \* Q5F' \* Q4F' \* Q3F' \* Q2F' \* Q1F' \* Q0F' \* RS

+ UD \* QEF \* QDF \* QCF \* QBF \* QAF \* Q9F \* Q8F \* Q7F \* Q6F \* Q5F \* Q4F \* Q3F \* Q2F \* Q1F \* Q0F \* RS;

QET = UD' \* QDF' \* QCF' \* QBF' \* QAF' \* Q9F' \* Q8F' \* Q7F' \* Q6F' \* Q5F' \* Q4F' \* Q3F' \* Q2F' \* Q1F' \* Q0F' \* RS

+ UD \* QDF \* QCF \* QBF \* QAF \* Q9F \* Q8F \* Q7F \* Q6F \* Q5F \* Q4F \* Q3F \* Q2F \* Q1F \* Q0F \* RS;

QDT = UD' \* QCF' \* QBF' \* QAF' \* Q9F' \* Q8F' \* Q7F' \* Q6F' \* Q5F' \* Q4F' \* Q3F' \* Q2F' \* Q1F' \* Q0F' \* RS

+ UD \* QCF \* QBF \* QAF \* Q9F \* Q8F \* Q7F \* Q6F \* Q5F \* Q4F \* Q3F \* Q2F \* Q1F \* Q0F \* RS;

292015-3

Figure 4. Example .LEF

```

QCT = UD' * QBF' * QAF' * Q9F' * Q8F' * Q7F' * Q6F' * Q5F' * Q4F' * Q3F' *
      Q2F' * Q1F' * Q0F' * RS
+ UD * QBF * QAF * Q9F * Q8F * Q7F * Q6F * Q5F * Q4F * Q3F * Q2F *
  Q1F * Q0F * RS;

QBT = UD' * QAF' * Q9F' * Q8F' * Q7F' * Q6F' * Q5F' * Q4F' * Q3F' * Q2F' *
      Q1F' * Q0F' * RS
+ UD * QAF * Q9F * Q8F * Q7F * Q6F * Q5F * Q4F * Q3F * Q2F * Q1F *
  Q0F * RS;

QAT = UD' * Q9F' * Q8F' * Q7F' * Q6F' * Q5F' * Q4F' * Q3F' * Q2F' * Q1F' *
      Q0F' * RS
+ UD * Q9F * Q8F * Q7F * Q6F * Q5F * Q4F * Q3F * Q2F * Q1F * Q0F *
  RS;

Q9T = UD' * Q8F' * Q7F' * Q6F' * Q5F' * Q4F' * Q3F' * Q2F' * Q1F' * Q0F' *
      RS
+ UD * Q8F * Q7F * Q6F * Q5F * Q4F * Q3F * Q2F * Q1F * Q0F * RS;

Q8T = UD' * Q7F' * Q6F' * Q5F' * Q4F' * Q3F' * Q2F' * Q1F' * Q0F' * RS
+ UD * Q7F * Q6F * Q5F * Q4F * Q3F * Q2F * Q1F * Q0F * RS;

Q7T = UD' * Q6F' * Q5F' * Q4F' * Q3F' * Q2F' * Q1F' * Q0F' * RS
+ UD * Q6F * Q5F * Q4F * Q3F * Q2F * Q1F * Q0F * RS;

Q6T = UD' * Q5F' * Q4F' * Q3F' * Q2F' * Q1F' * Q0F' * RS
+ UD * Q5F * Q4F * Q3F * Q2F * Q1F * Q0F * RS;

Q5T = UD' * Q4F' * Q3F' * Q2F' * Q1F' * Q0F' * RS
+ UD * Q4F * Q3F * Q2F * Q1F * Q0F * RS;

Q4T = UD' * Q3F' * Q2F' * Q1F' * Q0F' * RS
+ UD * Q3F * Q2F * Q1F * Q0F * RS;

Q3T = UD' * Q2F' * Q1F' * Q0F' * RS
+ UD * Q2F * Q1F * Q0F * RS;

Q2T = UD' * Q1F' * Q0F' * RS
+ UD * Q1F * Q0F * RS;

Q1T = UD' * Q0F' * RS
+ UD * Q0F * RS;

Q0T = RS;

```

ENDS

292015-4

Figure 4. Example .LEF (Continued)

Logic Optimizing Compiler Utilization Report  
 FIT Version 4.01 Baseline 27.1 4/9/86

\*\*\*\*\* Design implemented successfully

\*\*\*\* NOTE: Connect signal CLOCK to pin 1 AND pin 13.

INTEL CORPORATION

JAN. 15, 1987

1

1.0

5C060

BINARY 16-BIT UP/DOWN COUNTER WITH RUN/STOP AND ASYNCH. RESET USING T-FF

LB Version 4.01, Baseline 27.1 4/9/86

OPTIONS: TURBO=ON

```

      5C060
CLOCK -- 1  24:- Vcc
GND    -- 2  23:- RS
Q7     -- 3  22:- QF
Q6     -- 4  21:- QE
Q5     -- 5  20:- QD
Q4     -- 6  19:- QC
Q3     -- 7  18:- QB
Q2     -- 8  17:- QA
Q1     -- 9  16:- Q9
Q0    --10  15:- Q8
UD    --11  14:- RESET
GND    --12  13:- CLOCK
  
```

\*\*INPUTS\*\*

Name	Pin	Resource	MCell #	PTerms	MCells	Feeds:		
						OE	Clear	Clock
CLOCK	1	INP	-	-	-	-	-	CLK1 CLK2
UD	11	INP	-	-	1 2 3 4 5 6 7 8 9 10 11 12 13 14 15	-	-	-
GND	12	GND	-	-	-	-	-	-
CLOCK	13	INP	-	-	-	-	-	CLK1 CLK2
RESET	14	INP	-	-	-	-	1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16	-

Figure 5. Example .RPT File

```

RS  23  INP  -  -  1  -  -  -
      2
      3
      4
      5
      6
      7
      8
      9
     10
     11
     12
     13
     14
     15
     16

Vcc 24  Vcc  -  -  -  1  -  -
      2
      3
      4
      5
      6
      7
      8
      9
     10
     11
     12
     13
     14
     15
     16

```

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

Name	Pin	Resource	MCell #	PTerms	MCells	Feeds:		
						OE	Clear	Clock
Q7	3	TOTF	9	2/ 8	1	-	-	-
					2			
					3			
					4			
					5			
					6			
					7			
					8			
Q6	4	TOTF	10	2/ 8	1	-	-	-
					2			
					3			
					4			
					5			
					6			
					7			
					8			
Q5	5	TOTF	11	2/ 8	1	-	-	-
					2			
					3			
					4			
					5			
					6			
					7			
					8			
Q4	6	TOTF	12	2/ 8	1	-	-	-
					2			
					3			
					4			
					5			
					6			
					7			
					8			
9								
10								
11								

Figure 5. Example .RPT File (Continued)

Q3	7	TOTF	13	2/ 8	1	-	-	-
					2			
					3			
					4			
					5			
					6			
					7			
					8			
					9			
					10			
					11			
					12			
Q2	8	TOTF	14	2/ 8	1	-	-	-
					2			
					3			
					4			
					5			
					6			
					7			
					8			
					9			
					10			
					11			
					12			
					13			
Q1	9	TOTF	15	2/ 8	1	-	-	-
					2			
					3			
					4			
					5			
					6			
					7			
					8			
					9			
					10			
					11			
					12			
					13			
					14			
Q0	10	TOTF	16	1/ 8	1	-	-	-
					2			
					3			
					4			
					5			
					6			
					7			
					8			
					9			
					10			
					11			
					12			
					13			
					14			
					15			
Q8	15	TOTF	8	2/ 8	1	-	-	-
					2			
					3			
					4			
					5			
					6			
					7			
Q9	16	TOTF	7	2/ 8	1	-	-	-
					2			
					3			
					4			
					5			
					6			
QA	17	TOTF	6	2/ 8	1	-	-	-
					2			
					3			
					4			
					5			

292015-7

Figure 5. Example .RPT File (Continued)

QB	18	TOTF	5	2/ 8	1 2 3 4	-	-	-
QC	19	TOTF	4	2/ 8	1 2 3	-	-	-
QD	20	TOTF	3	2/ 8	1 2	-	-	-
QE	21	TOTF	2	2/ 8	1	-	-	-
QF	22	TONF	1	2/ 8	-	-	-	-
<b>**UNUSED RESOURCES**</b>								
	Name	Pin	Resource	MCell	PTerms			
		2	-	-	-			
<b>**PART UTILIZATION**</b>								
95%	Pins							
100%	MacroCells							
24%	Pterms							

292015-8

Figure 5. Example .RPT File (Continued)



October 1988

**Designing a Mailbox Memory for  
Two 80C31 Microcontrollers Using  
EPLDs**

**K. WEIGL & J. STAHL  
INTEL CORPORATION  
MUNICH, GERMANY**

Order Number: 292016-003

## INTRODUCTION

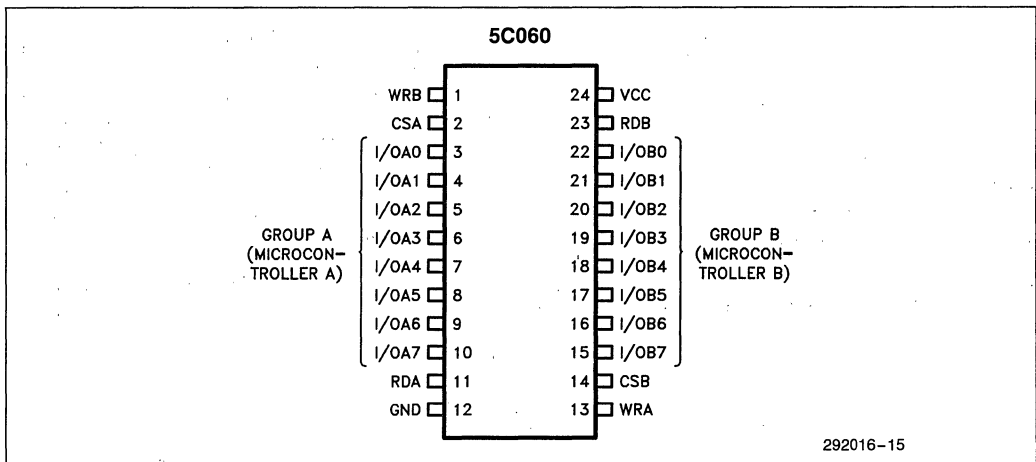
Very often, complex systems involve two or more microcontrollers to fulfill the requirements defined by a given objective. Since the nature of microcontrollers does not allow for easy dual-port memory design (no "READY" input; no "HOLD/HLDA" interface; port-oriented I/O etc.), design engineers are faced with the problem of interchanging information (data and status) between those microcontrollers. This application brief describes the design of a mailbox for exchanging information between two 80C31s, using a 5C060 H-EPLD as a "back-to-back" register, and a 5C031 H-EPLD as an arbitration vehicle to control the actions of the CPUs.

## THE 5C060 MAILBOX

In this application, the 16 macrocells of the 5C060 are grouped into two sets of 8 so called "ROIF" (register output with input feedback) primitives to implement the two 8 bit bus interfaces needed. The grouping is done according to the following picture.

The 5C060 allows for independent clocking of 8 macrocells on each side of the chip, the two clock inputs are used to clock data from the microcontroller bus into the chip. To read the data written into the mailbox by one of the controllers, the RDA- (controller A is reading) or RDB- (controller B is reading) line must be pulled low by activating the read command (/RD). In order to avoid spurious read-cycles, the /RD commands from both microcontrollers are logically "ORed" together with an active high CS-signal (Chip Select) inside the 5C060. The CS-signal for both ports is derived from address line A15. Therefore, whenever A15 becomes a logic "1" (true), the mailbox is activated and ready to take or submit data.

Address range for the mailbox: F000 Hex to FFFF Hex  
(Upper 12 kbyte)



## THE 5C031 "MAILBOX CONTROLLER"

To keep the two microcontrollers informed about the status of their mailbox, the 5C031 is programmed to supply the following signals to both controllers:

/OBFA: "OUTPUT BUFFER FULL" FOR MC A

/OBFB: "OUTPUT BUFFER FULL" FOR MC B

/IBEA: "INPUT BUFFER EMPTY" FOR MC A

/IBEB: "INPUT BUFFER EMPTY" FOR MC B

/INTA: INTERRUPT TO MC A

/INTB: INTERRUPT TO MC B

The next section will discuss the meanings of these signals in more detail.

**Output Buffer Full:** This flag is set whenever the controller writes into its own output buffer. The flag remains valid, until the second controller has read the data. The flag is automatically reset to its inactive state when this read cycle is accomplished.

---

### NOTE:

Both controllers can access (read or write) the mailbox simultaneously.

---

**Input Buffer Empty:** This flag indicates that there is no message in the mailbox. The flag will become inactive as soon as one microcontroller places a message for the other one (or vice versa).

Example: /IBEA remains "LOW" until microcontroller B places a message for controller A into the mailbox for A. /IBEA will go "HIGH" as soon as controller B has accomplished its write cycle, and will not go "LOW" again until microcontroller A has read the message.

**Interrupt:** The 5C031 is programmed to supply interrupts to both microcontrollers involved, on one of the following events.

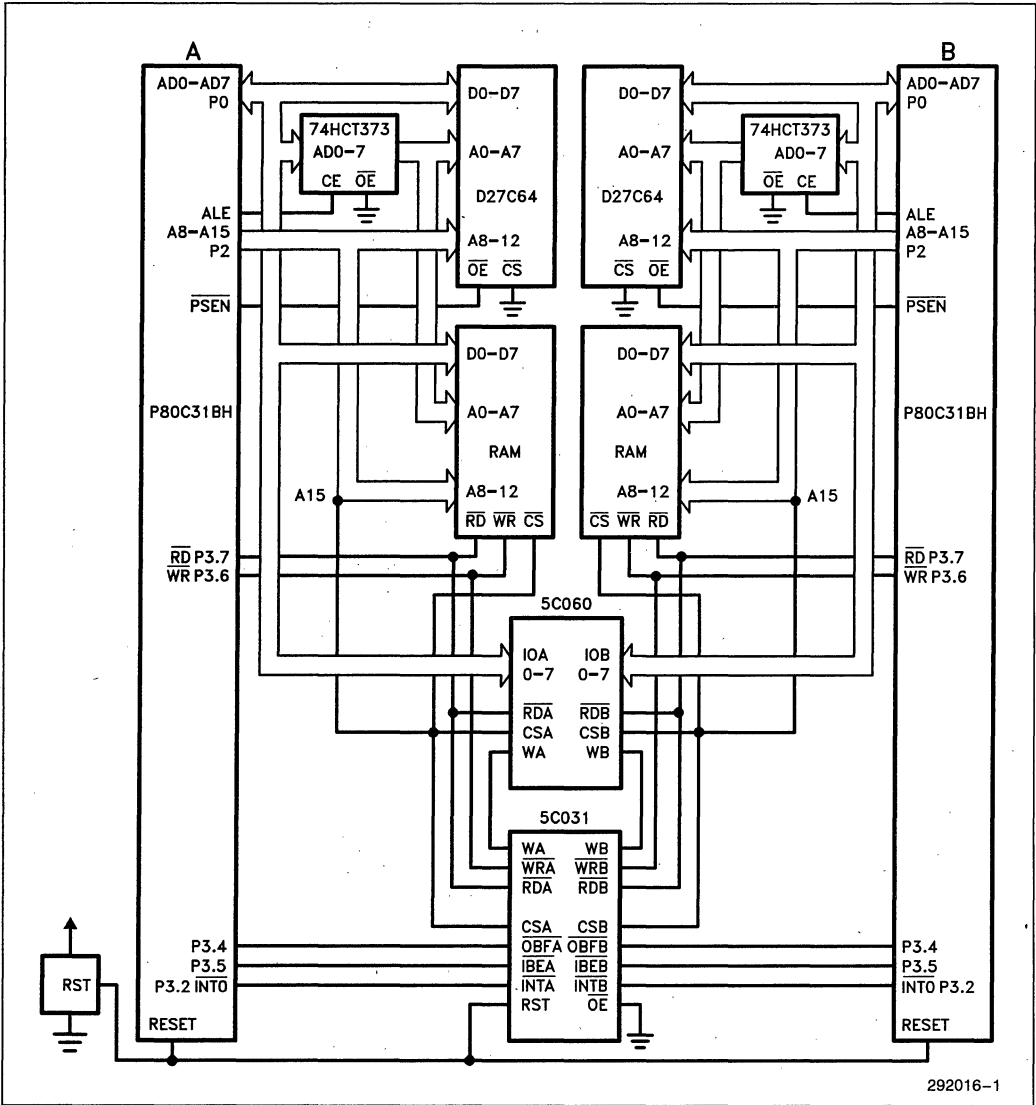
1. The /OBF flag of the opposite microcontroller becomes active; e.g. if controller A is placing a message for controller B, controller B receives an interrupt the same time as /OBFA becomes valid or vice versa.

2. The /IBE flag of the opposite microcontroller goes active, indicating that this controller has received the message; e.g. if controller B reads the message stored by controller A, its /IBEB flag goes active and controller receives an interrupt indicating that the buffer is empty.

The signals described above are necessary to accomplish a secure handshake without overwriting messages accidentally. In addition to that, the 5C031 is issuing the actual write commands for the two register sets inside the 5C060. The /WRA and /WRB signals are results of logical "AND" functions between the appropriate CS- and /WR signals from the microcontrollers. Therefore, spurious write cycles are unlikely to happen.

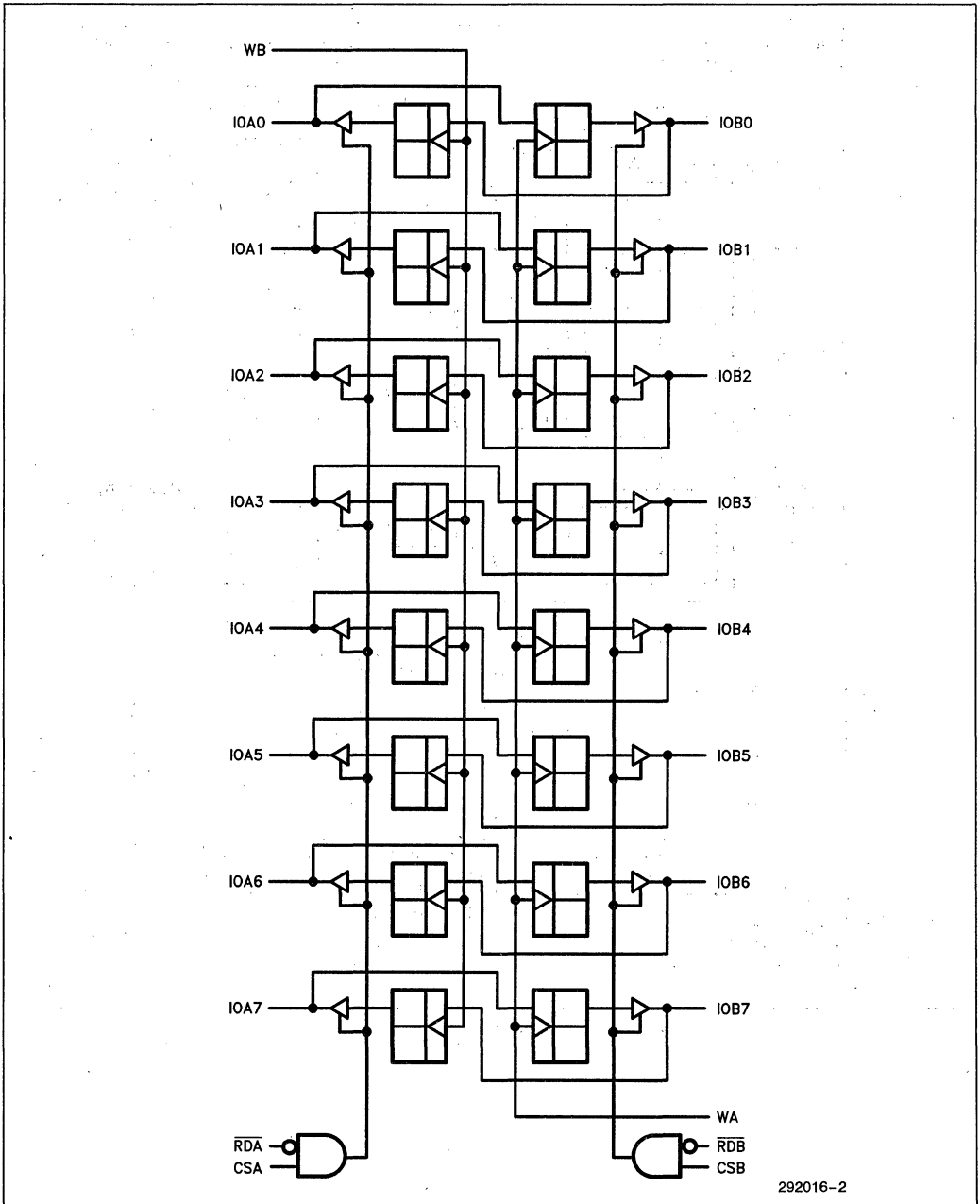
### NOTE:

This design can also be efficiently implemented in a single 5CBIC EPLD.



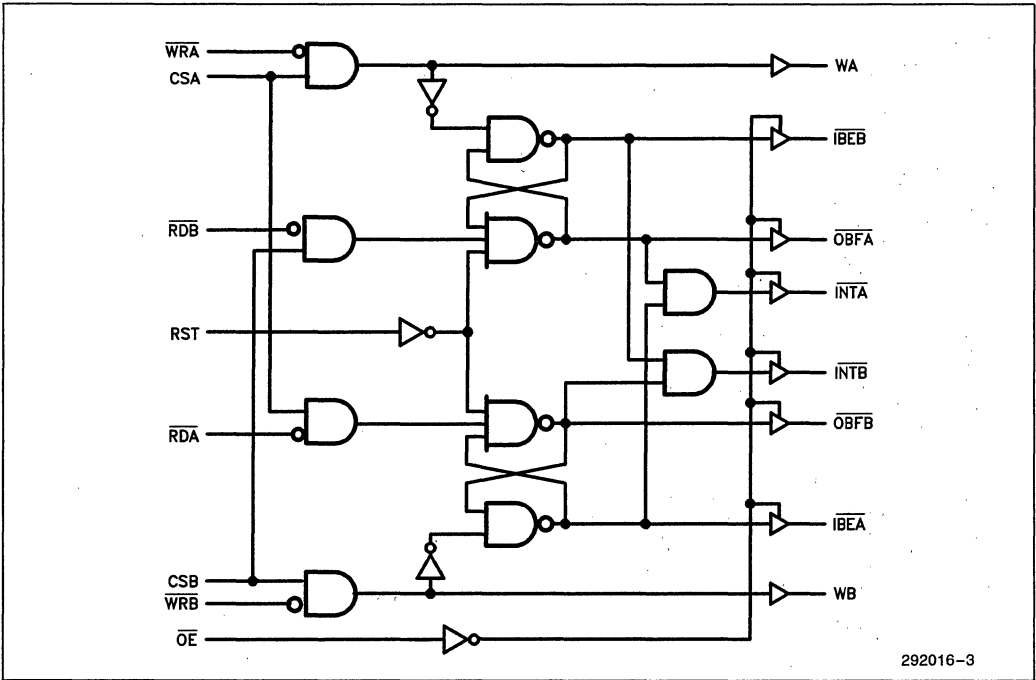
Block Diagram

5C060 "BACK TO BACK REGISTER"



292016-2

5C031 "MAIL BOX CONTROLLER"



292016-3

## 5C060 REGISTER ADF

```

JUERG STAHL
INTEL ZUERICH
March 27, 1986
80C31 MAILBOX MEMORY USING 5C060 / 5C031
1
5C060

*****
** EXAMPLE .ADF **
*****

LB Version 3.0, Baseline 17x, 9/26/85
PART: 5C060
INPUTS: WB@1, CSA@2, CSB@14, nRDA@11, nRDB@23, WA@13
OUTPUTS: IOB7@15, IOA7@10, IOB6@16, IOA6@9,
          IOB5@17, IOA5@8, IOB4@18, IOA4@7,
          IOB3@19, IOA3@6, IOB2@20, IOA2@5,
          IOB1@21, IOA1@4, IOB0@22, IOA0@3

NETWORK:
IOB7,DB7 = ROIF (DA7,WAC,GND,GND,RDBC)
IOA7,DA7 = ROIF (DB7,WBC,GND,GND,RDAC)
IOB6,DB6 = ROIF (DA6,WAC,GND,GND,RDBC)
IOA6,DA6 = ROIF (DB6,WBC,GND,GND,RDAC)
IOB5,DB5 = ROIF (DA5,WAC,GND,GND,RDBC)
IOA5,DA5 = ROIF (DB5,WBC,GND,GND,RDAC)
IOB4,DB4 = ROIF (DA4,WAC,GND,GND,RDBC)
IOA4,DA4 = ROIF (DB4,WBC,GND,GND,RDAC)
IOB3,DB3 = ROIF (DA3,WAC,GND,GND,RDBC)
IOA3,DA3 = ROIF (DB3,WBC,GND,GND,RDAC)
IOB2,DB2 = ROIF (DA2,WAC,GND,GND,RDBC)
IOA2,DA2 = ROIF (DB2,WBC,GND,GND,RDAC)
IOB1,DB1 = ROIF (DA1,WAC,GND,GND,RDBC)
IOA1,DA1 = ROIF (DB1,WBC,GND,GND,RDAC)
IOB0,DB0 = ROIF (DA0,WAC,GND,GND,RDBC)
IOA0,DA0 = ROIF (DB0,WBC,GND,GND,RDAC)

WAC = INP (WA)
RDBC = AND(CSBI,RDBI)
WBC = INP (WB)
RDAC = AND(CSAI,RDAI)
CSBI = INP (CSB)
nRDBI = INP(nRDB)
nRDAI = INP(nRDA)
CSAI = INP(CSA)
RDAI = NOT(nRDAI)
RDBI = NOT(nRDBI)

END$

```

292016-4

## 5C060 REGISTER LEF

```

JUERG STAHL
INTEL ZUERICH
March 27, 1986
80C31 MAILBOX MEMORY USING 5C060 / 5C031
1
5C060

*****
** EXAMPLE .LEF **
*****

LB Version 3.0, Baseline 17x, 9/26/85
LEF Version 1.0 Baseline 1.5i 02 Feb 1987
PART:
    5C060
INPUTS:
    WB01, CSA02, CSB014, nRDA011, nRDB023, WA013
OUTPUTS:
    IOB7015, IOA7010, IOB6016, IOA6009, IOB5017, IOA5008, IOB4018, IOA4007,
    IOB3019, IOA3006, IOB2020, IOA2005, IOB1021, IOA1004, IOB0022, IOA0003
NETWORK:
    WBC = INP(WB)
    WAC = INP(WA)
    CSAI = INP(CSA)
    CSBI = INP(CSB)
    nRDAI = INP(nRDA)
    nRDBI = INP(nRDB)
    IOB7, DB7 = ROIF(DA7, WAC, GND, GND, RDAC)
    IOA7, DA7 = ROIF(DB7, WBC, GND, GND, RDAC)
    IOB6, DB6 = ROIF(DA6, WAC, GND, GND, RDAC)
    IOA6, DA6 = ROIF(DB6, WBC, GND, GND, RDAC)
    IOB5, DB5 = ROIF(DA5, WAC, GND, GND, RDAC)
    IOA5, DA5 = ROIF(DB5, WBC, GND, GND, RDAC)
    IOB4, DB4 = ROIF(DA4, WAC, GND, GND, RDAC)
    IOA4, DA4 = ROIF(DB4, WBC, GND, GND, RDAC)
    IOB3, DB3 = ROIF(DA3, WAC, GND, GND, RDAC)
    IOA3, DA3 = ROIF(DB3, WBC, GND, GND, RDAC)
    IOB2, DB2 = ROIF(DA2, WAC, GND, GND, RDAC)
    IOA2, DA2 = ROIF(DB2, WBC, GND, GND, RDAC)
    IOB1, DB1 = ROIF(DA1, WAC, GND, GND, RDAC)
    IOA1, DA1 = ROIF(DB1, WBC, GND, GND, RDAC)
    IOB0, DB0 = ROIF(DA0, WAC, GND, GND, RDAC)
    IOA0, DA0 = ROIF(DB0, WBC, GND, GND, RDAC)
EQUATIONS:
    RDAC = CSAI * nRDAI';
    RDAC = CSBI * nRDBI';
END$

```

292016-5



5C060 REGISTER UTILIZATION REPORT

Logic Optimizing Compiler Utilization Report  
 FIT Version 1.0 Baseline 1.0i 2/6/87

\*\*\*\* Design implemented successfully

JUERG STAHL  
 INTEL ZUERICH  
 March 27, 1986  
 80C31 MAILBOX MEMORY USING 5C060 / 5C031  
 1  
 5C060

\*\*\*\*\*  
 \*\* EXAMPLE .RPT FILE \*\*  
 \*\*\*\*\*

LB Version 3.0, Baseline 17x, 9/26/85

5C060

```

WB --: 1  24:- Vcc
CSA --: 2  23:- nRDB
IOA0 --: 3  22:- IOB0
IOA1 --: 4  21:- IOB1
IOA2 --: 5  20:- IOB2
IOA3 --: 6  19:- IOB3
IOA4 --: 7  18:- IOB4
IOA5 --: 8  17:- IOB5
IOA6 --: 9  16:- IOB6
IOA7 --:10  15:- IOB7
nRDA --:11  14:- CSB
GND --:12  13:- WA
  
```

\*\*INPUTS\*\*

Name	Pin	Resource	MCell #	PTerms	MCells	Feeds:			
						OE	Clear	Clock	
WB	1	INP	-	-	-	-	-	CLK1	
CSA	2	INP	-	-	-	9			
						10			
						11			
						12			
						13			
						14			
						15			
						16			
						nRDA	11	INP	-
10									
11									
12									
13									
14									
15									
16									
GND	12	GND	-	-	-				
							2		
							3		
							4		
							5		
							6		
							7		
							8		
							9		

5C060 REGISTER UTILIZATION REPORT (Continued)

										10	11	12	13	14	15	16	
WA	13	INP	-	-	-	-	-	-	-	-	-	-	-	-	-	-	CLK2
CSB	14	INP	-	-	-	-	-	-	-	1	-	-	-	-	-	-	-
										2							
										3							
										4							
										5							
										6							
										7							
										8							
nRDB	23	INP	-	-	-	-	-	-	-	1	-	-	-	-	-	-	-
										2							
										3							
										4							
										5							
										6							
										7							
										8							
Vcc	24	Vcc	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
<b>**OUTPUTS**</b>																	
										Feeds:							
Name	Pin	Resource	MCell #	PTerms	MCells	OE	Clear	Clock									
IOA0	3	ROIF	9	1/ 8	1	-	-	-									
IOA1	4	ROIF	10	1/ 8	2	-	-	-									
IOA2	5	ROIF	11	1/ 8	3	-	-	-									
IOA3	6	ROIF	12	1/ 8	4	-	-	-									
IOA4	7	ROIF	13	1/ 8	5	-	-	-									
IOA5	8	ROIF	14	1/ 8	6	-	-	-									
IOA6	9	ROIF	15	1/ 8	7	-	-	-									
IOA7	10	ROIF	16	1/ 8	8	-	-	-									
IOB7	15	ROIF	8	1/ 8	16	-	-	-									
IOB6	16	ROIF	7	1/ 8	15	-	-	-									
IOB5	17	ROIF	6	1/ 8	14	-	-	-									
IOB4	18	ROIF	5	1/ 8	13	-	-	-									
IOB3	19	ROIF	4	1/ 8	12	-	-	-									
IOB2	20	ROIF	3	1/ 8	11	-	-	-									
IOB1	21	ROIF	2	1/ 8	10	-	-	-									
IOB0	22	ROIF	1	1/ 8	9	-	-	-									
292016-7																	
All Resources used																	
<b>**PART UTILIZATION**</b>																	
100%	Pins																
100%	MacroCells																
12%	PTerms																
292016-8																	

## 5C031 ARBITER ADF

JUERG STAHL  
 INTEL ZUERICH  
 March 28, 1986  
 80C31 MAILBOX MEMORY USING 5C060 / 5C031  
 2  
 5C031

\*\*\*\*\*  
 \*\* EXAMPLE .ADF \*\*  
 \*\*\*\*\*

LB Version 3.0, Baseline 17x, 9/26/85

PART: 5C031

INPUTS: RST, nWRA, nRDB, CSA, nRDA, nWRB, CSB, nOE

OUTPUTS: WA, nOBFA, nIBEB, nINTA, nINTB, nOBFB, nIBEA, WB

NETWORK:

nWRA = INP(nWRA)

nRDB = INP(nRDB)

RST = INP(RST)

CSA = INP(CSA)

nRDA = INP(nRDA)

nWRB = INP(nWRB)

CSB = INP(CSB)

nOE = INP(nOE)

WRA = NOT(nWRA)

WRB = NOT(nWRB)

RDA = NOT(nRDA)

RDB = NOT(nRDB)

OE = NOT(nOE)

nRST = NOT(RST)

WA = CONF(WAd, VCC)

WAd = AND(CSA, WRA)

WB = CONF(WBd, VCC)

WBd = AND(CSB, WRB)

nRB = NAND(RDB, CSB)

nRA = NAND(RDA, CSA)

nWAd = NOT(WAd)

nWBd = NOT(WBd)

nOBFA, nOBFA = COCF(nOBFA, OE)

nOBFB, nOBFB = COCF(nOBFB, OE)

nIBEA, nIBEA = COCF(nIBEA, OE)

nIBEB, nIBEB = COCF(nIBEB, OE)

nINTA = CONF(nINTA, OE)

nINTB = CONF(nINTB, OE)

nINTAd = AND(nOBFA, nIBEA)

nINTBd = AND(nOBFB, nIBEB)

nOBFBd = NAND(nRA, nIBEA, nRST)

nOBFA, nOBFA = NAND(nRB, nIBEB, nRST)

nIBEBd = NAND(nWAd, nOBFA)

nIBEA, nIBEA = NAND(nWBd, nOBFB)

END\$

292016-9

## 5C031 ARBITER LEF

```

JUERG STAHL
INTEL ZUERICH
March 28, 1986
80C31 MAILBOX MEMORY USING 5C060 / 5C031
2
5C031

```

```

*****
** EXAMPLE .LEF **
*****

```

```

LB Version 3.0, Baseline 17x, 9/26/85
LEF Version 1.0 Baseline 1.5i 02 Feb 1987
PART:

```

```

5C031

```

## INPUTS:

```

RST, nWRA, nRDB, CSA, nRDA, nWRB, CSB, nOE

```

## OUTPUTS:

```

WA, nOBFA, nIBEB, nINTA, nINTB, nOBFB, nIBEA, WB

```

## NETWORK:

```

RST = INP(RST)
nWRA = INP(nWRA)
nRDB = INP(nRDB)
CSA = INP(CSA)
nRDA = INP(nRDA)
nWRB = INP(nWRB)
CSB = INP(CSB)
nOE = INP(nOE)
WA = CONF(WAd, VCC)
nOBFA, nOBFA = COCF(nOBFAAd, OE)
nIBEB, nIBEB = COCF(nIBEBBd, OE)
nINTA = CONF(nINTAd, OE)
nINTB = CONF(nINTBd, OE)
nOBFB, nOBFB = COCF(nOBFBd, OE)
nIBEA, nIBEA = COCF(nIBEAAd, OE)
WB = CONF(WBd, VCC)

```

## EQUATIONS:

```

WBd = CSB * nWRB';

nIBEAAd = CSB * nWRB'
+ nOBFB';

nOBFBd = (nIBEA * RST' * CSA'
+ nIBEA * RST' * nRDA)';

nINTBd = nOBFB * nIBEB;

nINTAd = nOBFA * nIBEA;

nIBEBBd = CSA * nWRA'
+ nOBFA';

OE = nOE';

nOBFAAd = (nIBEB * RST' * CSB'
+ nIBEB * RST' * nRDB)';

WAd = CSA * nWRA';

```

```

END$

```

```

292016-10

```

5C031 ARBITER LEF (Continued)

Logic Optimizing Compiler Utilization Report  
 FIT Version 1.0 Baseline 1.0i 2/6/87

\*\*\*\*\* Design implemented successfully

JUERG STAHL  
 INTEL ZUERICH  
 March 28, 1986  
 80C31 MAILBOX MEMORY USING 5C060 / 5C031  
 2  
 5C031

\*\*\*\*\*  
 \*\* EXAMPLE .RPT FILE \*\*  
 \*\*\*\*\*

LB Version 3.0, Baseline 17x, 9/26/85

5C031

```

GND --: 1 20:- Vcc
GND --: 2 19:- WB
nOE --: 3 18:- WA
CSB --: 4 17:- nOBFB
nWRB --: 5 16:- nINTB
nRDA --: 6 15:- nINTA
CSA --: 7 14:- nIBBB
nRDB --: 8 13:- nOBFA
nWRA --: 9 12:- nIBEA
GND --:10 11:- RST
  
```

\*\*INPUTS\*\*

Name	Pin	Resource	MCell #	PTerms	MCells	Feeds:		
						OE	Clear	Preset
nOE	3	INP	-	-	-	3	-	-
						4		
						5		
						6		
						7		
						8		
CSB	4	INP	-	-	1	-	-	-
					7			
					8			
nWRB	5	INP	-	-	1	-	-	-
					8			
nRDA	6	INP	-	-	3	-	-	-
CSA	7	INP	-	-	2	-	-	-
					3			
					6			
nRDB	8	INP	-	-	7	-	-	-
nWRA	9	INP	-	-	2	-	-	-
					6			
GND	10	GND	-	-	-	-	-	-
RST	11	INP	-	-	3	-	-	-
					7			
Vcc	20	Vcc	-	-	-	1	-	-
						2		

5C031 ARBITER UTILIZATION REPORT

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

Name	Pin	Resource	MCell #	PTerms	MCells	Feeds: OE	Clear	Preset
nIBEA	12	COCF	8	2/ 8	3 5	-	-	-
nOBFA	13	COCF	7	2/ 8	5 6	-	-	-
nIBEB	14	COCF	6	2/ 8	4 7	-	-	-
nINTA	15	CONF	5	1/ 8	-	-	-	-
nINTB	16	CONF	4	1/ 8	-	-	-	-
nOBFB	17	COCF	3	2/ 8	4 8	-	-	-
WA	18	CONF	2	1/ 8	-	-	-	-
WB	19	CONF	1	1/ 8	-	-	-	-

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

Name	Pin	Resource	MCell	PTerms
-	1	-	-	-
-	2	-	-	-

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

88% Pins  
 100% MacroCells  
 18% Pterms

October 1988

# **Atypical Latch/Register Construction in EPLDs**

**THOM BOWNS**  
PROGRAMMABLE LOGIC APPLICATIONS  
INTEL CORPORATION

Order Number: 292031-003

**ATYPICAL LATCH/REGISTER CONSTRUCTION IN EPLDs**

Though Intel's EPLDs include many of the typical latch and register types, some logic designs require register or latch configurations not directly supported in the current EPLDs. In many cases these register and latch configurations can be generated using the logic array and combinational feedback. A "latch" is defined as a level-triggered, flow-through type such as the 74373, and a "register" is defined as an edge-triggered flip-flop such as the 7474.

This application brief will detail the construction of a D-type latch, an RS latch and a D flip-flop using combinational logic and feedback. Also discussed is the construction of an RS flip-flop, a JK flip-flop and a T flip-flop using registered logic and feedback.

The RS latch is the simplest latch configuration. The equations for it are as follows:  $QB = !(Q + S)$ ,  $Q = !(QB + R)$  where Q is the output of one NOR gate, and QB is the output of the other (Note: as a convention

in this Ap brief, the "!" operator is used to signify inversion). The schematic of the RS latch is shown in Figure 1a.

Since cross coupled logic is not supported in EPLDs, we must convert the equation to a single term with feedback.

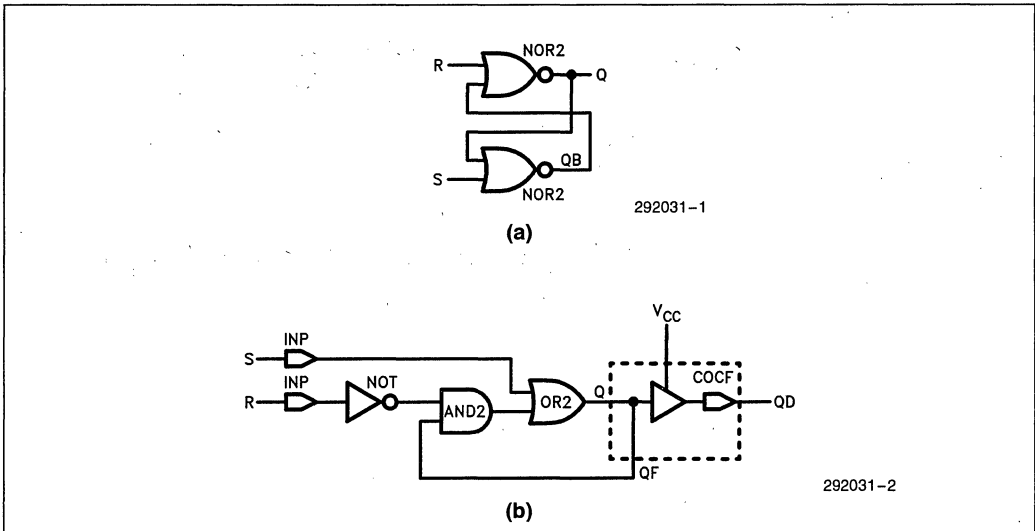
$$QD, QF = COCF(Q, VCC)$$

$$Q = S + !R * QF;$$

where QF is the feedback from Q output.

This circuit can be implemented in an EPLD macrocell. Where combinational feedback is not supported, I/O feedback will suffice. The schematic of this implementation is shown in Figure 1b.

With the RS latch, the inputs are normally low. A logical one on S sets Q to 1, and a one on R resets Q to a 0. Logical ones on both inputs simultaneously cause the output to remain at a high level since S takes precedence over R in this implementation.



**Figure 1. RS Latch Implementation In a) Discrete Gates and b) EPLD Logic**



Another latch is the 74373 type, or D latch. This latch works by either enabling input data to appear at the output, or by holding the output to the last input data state. Its equation is this:  $QB = !((!D * E) * Q)$ ,  $Q = !((D * E) * QB)$ . Again, Q is the output of one NAND gate, and QB is the output of the other. Figure 2a shows this version of the design.

$$QD, QF = COCF(Q, VCC)$$

$$Q = D * E + !E * QF;$$

QF is the feedback from the COCF. In this circuit, when E is high, data flows through transparently. When E is brought low, data is latched. When using input feedback, care must be taken when tri-stating the output as data will no longer be latched. The EPLD implementation is given in Figure 2b.

Again, we must convert to an EPLD-type equation and schematic:

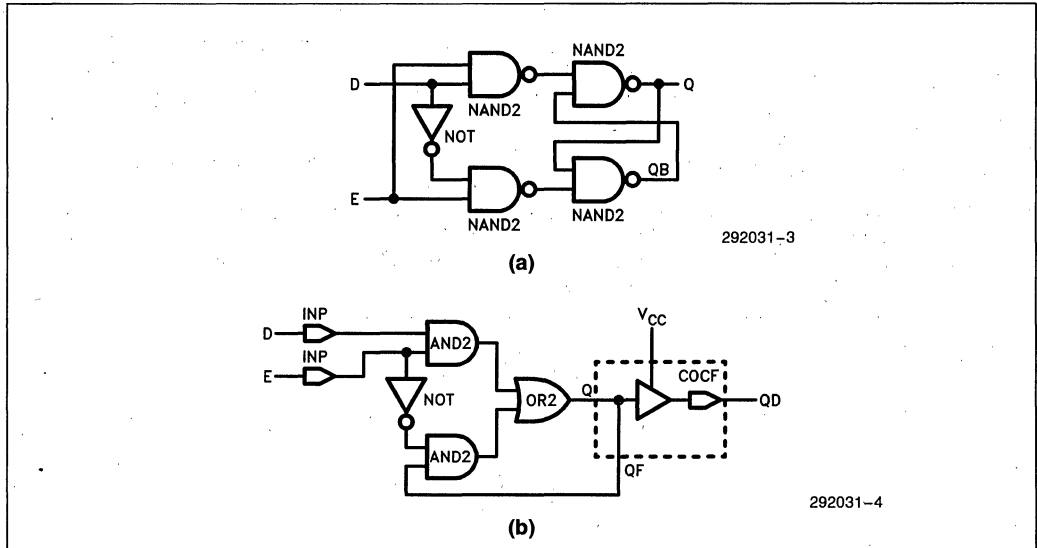


Figure 2. Implementation of a D Type Latch Using a) Discrete Gates and b) EPLD Logic

This latch can be cascaded with a second latch to produce an edge triggered, master/slave D flip-flop, using combinational logic. The flip-flop is a solution to using asynchronous clocking, preset and clear functions when they aren't supported. Also, if an I/O conflict exists within a macrocell group when using registered logic, this design will fit since it uses combinational logic. Figure 3 shows the schematic for this design.

This design does consume two macrocells, but in many cases, that isn't a problem.

The boolean equation of the D flip-flop is this:

$$\begin{aligned}
 QD, QF &= COCF(Q, VCC) \\
 YF &= NOCF(Y) \\
 Y &= D * !CLOCK + YF * CLOCK; \\
 Q &= YF * CLOCK + QF * !CLOCK;
 \end{aligned}$$

Q is the flip-flop output and Y is the first latch output. Data is latched in to the second latch on the low-going edge of clock, and is clocked out to Q on the high-going edge of clock.

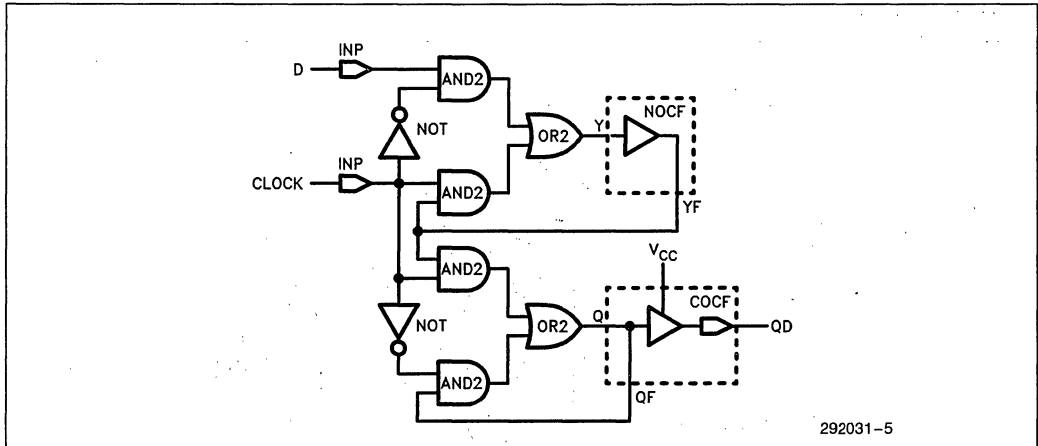


Figure 3. Combinational Logic Implementation of a D Flip-Flop

Preset and clear can be added into the equations as well:

$$QD, QF = COCF(Q, VCC)$$

$$YF = NOCF(Y)$$

$$Y = D * ICLOCK + YF * CLOCK;$$

$$Q = YF * CLOCK * !(CLEAR TERM) + (PRESET TERM) + QF * ICLOCK * !(CLEAR TERM);$$

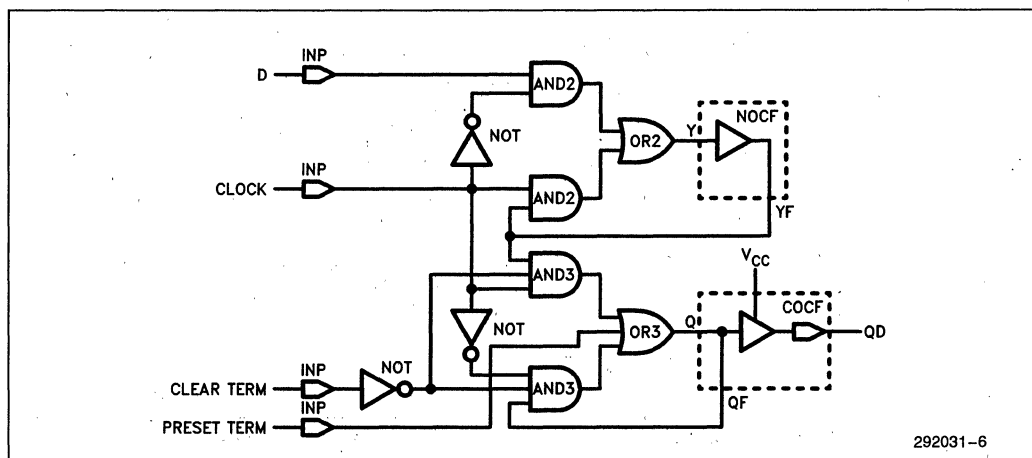
When the PRESET TERM is logically true, Q is asynchronously set to 1.

When the CLEAR TERM is logically true, Q is asynchronously cleared to 0.

The PRESET TERM takes priority over the CLEAR TERM.

This schematic is shown in Figure 4.

Due to the nature of the design, input delays plus array delays plus feedback delays must be added and used to determine a maximum operating frequency. In this example,  $t_{IN} + t_{AD} + t_{CF} + t_{AD} = 113$  ns for a -65 5C121, leaving a maximum frequency of 8.8 MHz.



292031-6

Figure 4. D Flip-Flop with Added Preset and Clear Terms

Other useful workarounds involve D registers and logic in constructing RS, JK and T flip-flops, for use in EPLDs not supporting these configurations. The RS flip-flop is simply the RS latch discussed earlier coupled to registered feedback.

$$QD, QF = RORF(Q, CLOCK, GND, GND, VCC)$$

$$Q = S + QF * !R;$$

Normally, S and R will remain low. When S is brought high, QD will become 1 on the next clock trigger edge. When R is brought high, QD will become 0 on the next clock trigger edge. The schematic is given in Figure 5.

The JK flip-flop is another useful and easily implemented register:

$$QD, QF = RORF(Q, CLOCK, GND, GND, VCC)$$

$$Q = J * !QF + !K * QF$$

When J = K = 1, QD toggles to opposite state on next clock trigger. When J = K = 0, QD remains the same. When J does not equal K, QD will follow J on next clock trigger. The schematic is shown in Figure 6.

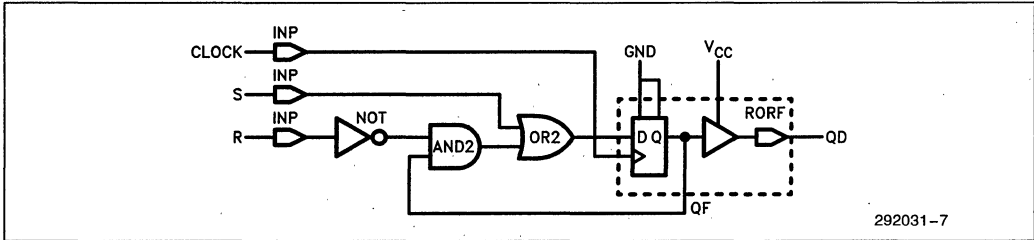


Figure 5. EPLD Implementation of an RS Flip-Flop

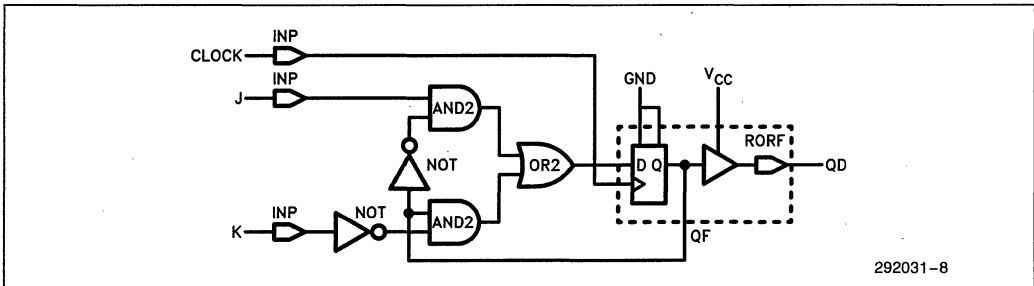


Figure 6. EPLD Implementation of a JK Flip-Flop

The T flip-flop is also easily constructed:

$$QD, QF = \text{RORF}(Q, \text{CLOCK}, \text{GND}, \text{GND}, \text{VCC})$$

$$Q = T * !QF + !T * QF;$$

When T is high, QD will toggle to opposite state on next trigger. When T is low, QD will remain the same. Figure 7 shows the T flip-flop design schematic.

Each of these designs uses a minimum number of p-terms; adding p-terms is possible to the limit of the macrocell being used. It is possible to substitute an entire logical expression for each input listed (except

register clock), as long as the minimized logic equations resulting do not exceed the macrocells p-term count.

For example, consider using the J-K register. Setting  $J = A * B * C + D$  and setting  $K = E * !F * !G + H + I$  then the minimized p-term count will expand from two p-terms to five p-terms, which would still be okay within a macrocell with more than five p-terms.

Using logic gates and combinational or registered feedback, one can easily implement many types of latches and registers. Regardless of the EPLD type, there exists the resources to implement any of the discussed circuitry.

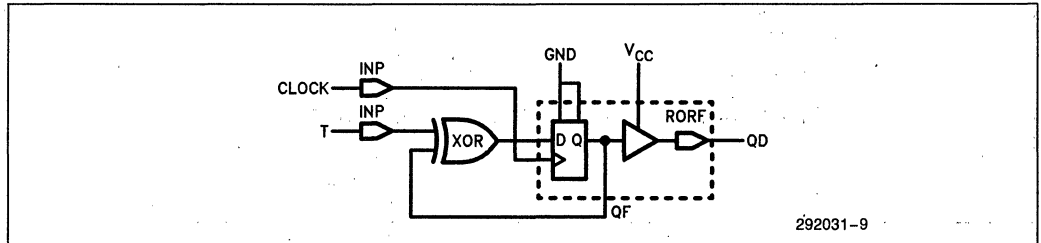


Figure 7. Implementation of a T Flip-Flop

October 1988

**5C032-25 vs. 16V8-25:  
A Device Comparison**

**DANIEL E. SMITH  
LILYAS KOUMIS**  
PROGRAMMABLE LOGIC APPLICATIONS  
INTEL CORPORATION

Order Number: 292051-001

## INTRODUCTION

This application brief compares the Intel 5C032-25 EPLD with the Lattice 16V8-25 GAL\*, showing how the 5C032 is superior to the 16V8 for low-power CMOS PLD applications. The compatibility between the two devices is high enough that the 5C032-25 can be dropped directly into the 16V8-25 socket for the majority of applications. Areas where the 5C032 is not compatible are also noted. Information in the brief is based on the Intel 5C032 Data Sheet (order number: 290155-002 or later) and the Lattice 16V8 Data Sheet (undated).

The comparison is divided into the following areas:

- Technology
- Architecture
- Specifications
- Development Support

## TECHNOLOGY

The 5C032 is produced on Intel's CHMOS EPROM process and is, therefore, UV erasable. The 16V8 is produced on a CMOS EEPROM process and is electrically erasable. Because neither device will typically be erased and reprogrammed in-circuit, this difference is negligible. The fuse patterns for the two devices are different. Therefore, the JEDEC files are not compatible.

## ARCHITECTURE

Architecturally, the 5C032 is a superset of the 16V8. Any architectural configuration supported by the 16V8 can be implemented in the 5C032. There are a number of configurations, however, supported by the 5C032 that cannot be implemented in the 16V8 architecture.

As shown in Figure 1, both the 5C032 and 16V8 are 20-pin devices with 8 I/O macrocells. The two devices are pin compatible. All inputs and I/Os are on the same pins. Macrocells in the devices support registered and combinatorial modes. (Refer to the discussions on "Inputs" and "Macrocells" later in this brief.)

The major architectural difference between the 16V8 and the 5C032 lies in flexibility. During programming, the 16V8 uses 10 bits to internally configure all 8 macrocells. 1 bit (SYN) is a global "register/combinatorial" mode bit. A second bit (AO) is also a global bit that controls an OE mux. These two bits provide global selection of modes but limit the independent control of macrocells. Each macrocell has an individual configuration bit (ACn) to give macrocells some independent control. In contrast, the 5C032 provides 2 bits per macrocell to independently configure each macrocell (16 bits total). This gives the 5C032 greater flexibility than the 16V8. Another difference concerns the state of macrocell registers on power-up. 5C032 registers are low on power-up, while the 16V8 registers are high. This difference may be important in some applications.

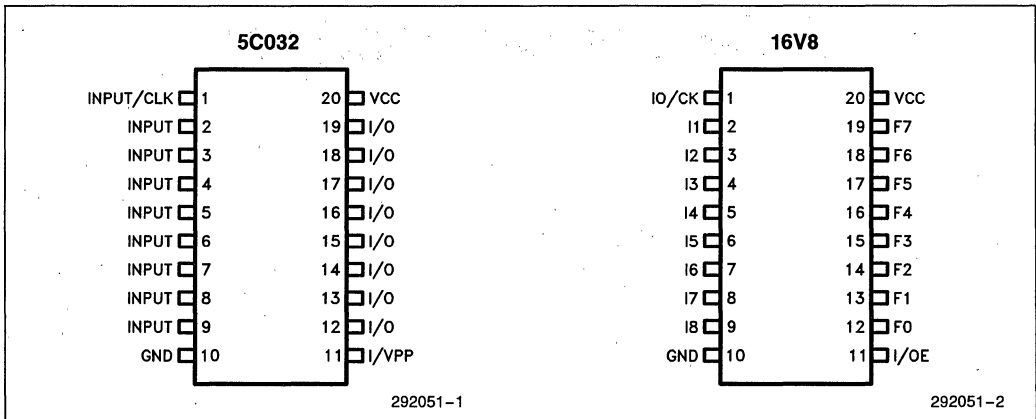


Figure 1. 5C032 and 16V8 Pinouts

\*GAL is a registered trademark of Lattice Semiconductor Corporation.

## Inputs

The 5C032 has 9 dedicated inputs and one CLK/INP pin. The 16V8 has 8 dedicated inputs, one global CLK/INP pin, and one global OE/INP pin. The CLK inputs are both on pin 1. The global OE on the 16V8 (pin 11) corresponds to an input on the 5C032. This pin can be used on the 5C032 as a global OE. On the 5C032, however, *any* input can function as the global OE. The 16V8 does not provide this flexibility.

On both devices, the CLK pin can function as an input to the logic array when implementing combinatorial logic only (no registers). Pin 11 (OE/INP on the 16V8) can also be used as a dedicated input in combinatorial mode. Pin 11 on the 5C032 can be used as an input in both registered and combinatorial mode.

## Macrocells

Each 16V8 macrocell is fed by 8 p-terms. One of the eight p-terms can be used to control the OE signal for combinatorial macrocells. When this is done, only 7 p-terms remain as inputs to the macrocell. Depending on the configuration, the OE can also be tied to VCC or GND, or can be globally driven by pin 11.

5C032 macrocells are fed by 8 p-terms. A ninth p-term is provided for independent OE control. Thus the 5C032 macrocell can implement equations with more p-terms than the 16V8. All options are available independently for all macrocells, which makes the device more flexible than the 16V8.

The 16V8 is placed in registered or combinatorial mode by a global architecture bit. Registered mode means pin 1 is global CLOCK and pin 11 is global OE. Registered macrocells cannot use product terms to independently enable outputs; only the global OE can be used. Macrocells can be configured as combinatorial outputs when the device is in registered mode, but only 7 p-terms are available as macrocell inputs. Buried registers can be emulated on a global basis by disabling the global OE and using the feedbacks only, but buried registers cannot be mixed with output registers in the same design.

In the 5C032, registers are selected on a macrocell-by-macrocell basis. Any supported configuration can be implemented on any other macrocell. Independent OE p-terms are available with registers (see Figure 2). A global OE can be implemented by programming all OE p-terms the same. Buried registers can also be selected on a macrocell basis. These differences make the 5C032 much more flexible than the 16V8.

## I/O Configurations

Table 1 shows the configurations supported for both devices. Note that most 16V8 macrocell configurations have some restriction on use.

**Table 1. 5C032/16V8 Configurations**

5C032	16V8 (Comb.)	16V8 (Reg.)
Input	Input	Input
Input on unused Macrocell	Input on unused Macrocell	Input on unused Macrocell
Comb. Out (no feedback)	Comb. Out (no feedback—OE = VCC)	Comb. Out (no feedback—7 p-terms)
Comb. Out (input feedback)	Comb. Out (input feedback—7 p-terms)	Comb. Out (input feedback—7 p-terms)
Register (with feedback—p-term controlled OE)	n/a	Register (with feedback—global OE only)
Register (no feedback—p-term controlled OE)	n/a	Register (no feedback—global OE only)
Buried Register (any register)	n/a	Buried Register (global only)

p-terms = Product terms

n/a = not available

Table 2 summarizes the architecture comparison:

**Table 2. 5C032/16V8 Architecture Comparison**

Device Feature	5C032	16V8
# of Pins	20	20
Dedicated Inputs	9	8/9
Total Inputs	16	16
Macrocells	8	8
Synch. Clocks	1	1
Logic P-terms/Macrocell	8	8/7(1)
OE P-Terms/Macrocell	1	0/1(2)
Global OE	1 (3)	1
Device Erase	UV	Electrical
Register Output State On Power-Up	low	high

1. When using a p-term to drive the OE signal for a macrocell, the 16V8 can only use 7 p-terms as macrocell inputs.
2. 16V8 registers must use the global OE signal. Macrocells programmed for combinatorial mode can use a p-term. In contrast, the 5C032 provides a p-term for all macrocells in all configurations.
3. Global OE is implemented on 5C032 by driving all OE p-terms by pin 11.



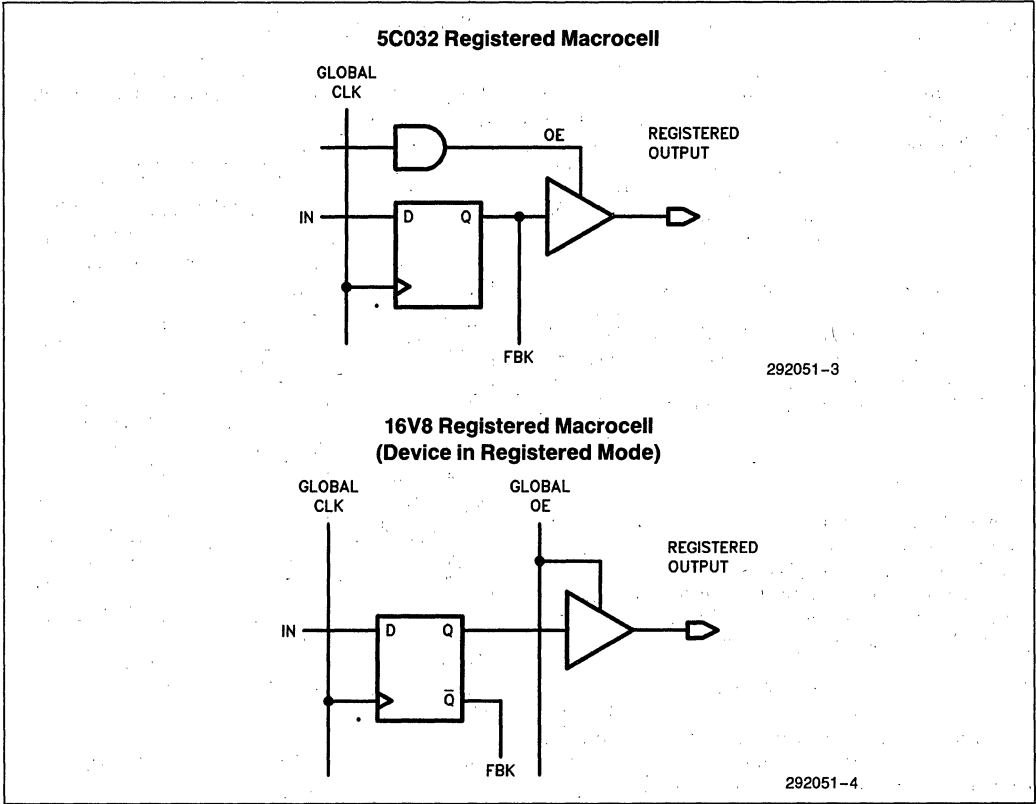


Figure 2. 5C032 and 16V8 Registers

**SPECIFICATIONS**

The following tables describe differences between the 5C032 and the half-power 16V8 in three different tables: (1) Absolute Maximum Ratings, (2) D.C. Characteristics, and (3) A.C. Characteristics.

**D.C. Characteristic Differences**

The Intel 5C032-25 meets or exceeds all but two 16V8 D.C. specifications (short circuit current and  $I_{OL}$ ). Due to the advanced CMOS technology, the Intel 5C032 consumes one-third the power of the half-power 16V8. It also consumes almost three orders of magnitude less

standby power than the half-power 16V8. For low power applications where the output drive current requirements are 4mA or less, the Intel 5C032 is an ideal replacement for the 16V8. The 16V8, with outputs capable of sinking up to 16mA, is better suited to applications that require higher current sink.

**A.C. Characteristics Differences**

The Intel 5C032-25 meets all but one 16V8 A.C. specification (Output Enable/Disable). Thus the Intel 5C032 is an ideal replacement for the 16V8 in most applications.

**Absolute Maximum Rating Differences**

Parameter	5C032-25			16V8-25			Units
	Symbol	Min	Max	Symbol	Min	Max	
Supply Voltage	$V_{CC}$	-2.0	7.0	$V_{CC}$	-0.5	7.0	V
Storage Temp.	$T_{stg}$	-65	+150	$T_{stg}$	-65	+125	C
Ambient Temp.	$T_{amb}$	-10	+85	$T_A$	0	+75	C

**D.C. Characteristics(1)**

Parameter	5C032-25			16V8-25			Units
	Symbol	Min	Max	Symbol	Min	Max	
Supply Current	$I_{CC}$		30	$I_{CC}$		90	mA
Short Circuit Current	$I_{SC}$		-10	$I_{OS}$		-130	mA
Standby Current	$I_{SB}$		0.1	$I_{SB}$		70	mA
Output Low Voltage	$V_{OL}$		0.45 $I_{OL}=4$	$V_{OL}$		0.5 $I_{OL}=16$	V mA
Output High Voltage	$V_{OH}$	2.4 $I_{OH}=-4$		$V_{OH}$	2.4 $I_{OH}=-3.2$		V mA
Input High Voltage	$V_{IH}$	2.0	$V_{CC}+0.3$	$V_{IH}$	2.0	$V_{CC}+1$	V

1. All D.C. Characteristics are compared to the Half-Power GAL 16V8. A Comparison to the Full-Power GAL 16V8 would show that power consumption is twice that of the Half-Power GAL 16V8.

**A.C. Characteristics**

Parameter	5C032-25			16V8-25			Units
	Symbol	Min	Max	Symbol	Min	Max	
Input to Active Out	t <sub>PD</sub>		25	t <sub>DVQV1</sub>		25	ns
P-term Enable to Out Enable	t <sub>PZX</sub>		25	t <sub>DVQV2</sub>		25	ns
P-term Disable to Out Disable	t <sub>PXZ</sub>		25	t <sub>DVQZ2</sub>		25	ns
OE-pin Enable to Out Enable	t <sub>PZX</sub>		25	t <sub>GHQZ2</sub>		20	ns
OE-pin Disable to Out Disable	t <sub>PXZ</sub>		25	t <sub>GHQV</sub>		20	ns
Clock High to Output Valid	t <sub>CO</sub>		15	t <sub>CHQV</sub>		15	ns
Input Setup Time	t <sub>SU</sub>	20		t <sub>DVCH</sub>	20		ns
Input Hold Time	t <sub>H</sub>	0		t <sub>CHDX</sub>	0		ns
Clock Low	t <sub>CL</sub>	10		t <sub>CHCL</sub>	15		ns
Clock High	t <sub>CH</sub>	10		t <sub>CLCH</sub>	15		ns
Register Output Fdbk to Register Input (Internal)	t <sub>CNT</sub>	30		(1)			ns
Max Count Frequency	f <sub>CNT</sub>		33.3	(2)		33.3	MHz

1. Lattice does not specify this parameter. Intel specifies this parameter strictly for calculation of f<sub>CNT</sub>. f<sub>CNT</sub> is the count frequency associated with designs that use feedback signals, e.g., counters.

2. Lattice does not specify an equivalent. However, this value can be determined using either "register output feedback to register input" delay, or the "clock period", whichever is the larger. Since "register output feedback to register input" delay is unknown, the indicated frequency value assumes the clock period (t<sub>CHCL</sub> + t<sub>CLCH</sub> = 30 ns) is the larger of the two parameters.

**DEVELOPMENT SUPPORT**

Both the 5C032 and the 16V8 are supported by ABEL and can be programmed on the Data I/O LOGICPAK and UNISITE programmers. The 5C032 is also supported by iPLDS II (Intel Programmable Logic Development System) using the PCCP PC-based programmer and by iPLS II (Intel Programmable Logic Software) using either the PC-based programmer or the iUP-200A/201A Programmer. The 16V8 is also supported on the Data I/O Model 60 programmer.

**SUMMARY**

The 5C032-25 provides a low-power upgrade to the 16V8-25 for most applications. If your application requires higher density devices, or fast programmable devices for specific applications, contact your local Intel sales office.



**APPLICATION  
NOTE**

**AP-271**

April 1986

# **Applying The 5C121 Architecture**

**JIM DONNELL**  
PROGRAMMABLE LOGIC APPLICATIONS  
INTEL CORPORATION

Order Number: 292008-001

## INTRODUCTION

Intel's 5C121 Erasable Programmable Logic Device represents a new breed in the world of programmable logic. With gate densities approaching those of gate arrays and a reconfigurable architecture, the logic designer is freed from choosing between scores of generic programmable logic to perhaps find an acceptable match for his or her design needs. Adding to the list of benefits is the fact that the 5C121 is erasable. Now sections of the design can actually be programmed and tested in the device — without sacrificing a part to the circular file. In addition, there is no longer a need to generate test vectors to qualify the programming of the parts. EPLDs are erasable and therefore 100% testable at the factory.

## OBJECTIVE

The purpose of this application note is to demonstrate the architectural options of the 5C121 by designing a digital crosspoint switch. Conceptually, a digital crosspoint switch switches data from any input to any output. Figure 1 shows a block diagram of a byte-wide crosspoint switch.

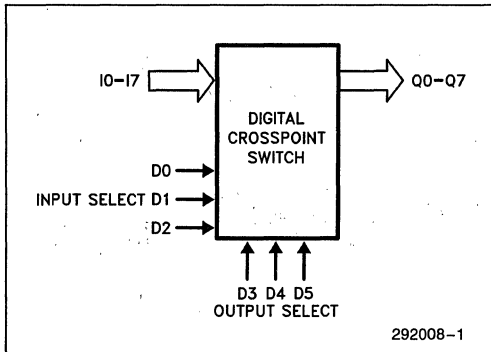


Figure 1. Functional Diagram of a Digital Crosspoint Switch

This design will employ features such as: registered output with registered feedback, combinational feedback, input latches, buried registers, and dual clock options. The digital crosspoint switch in this design can route data from one of eight inputs to one of eight outputs in a single clock cycle. Options for holding the deselected outputs at previous levels, latching inputs, and fitting considerations are explored.

## THE BASIC ARCHITECTURE

The 5C121 contains 28 Macrocells, 12 dedicated inputs, 24 programmable I/O lines, and two clocks input pins. Inputs may be flow through, or latched on the rising or falling edge of either clock. Output options

include registered or combinational output. In addition, each output may be fed back into the array in both the true and complement version. For a more complete description of the 5C121 architecture the reader is referred to the 5C121 data sheet.

## COMBINATIONAL FEEDBACK

Feedback in logic designs is used for a variety of reasons. Combinational feedback in the 5C121 is often used to reduce the number of product terms feeding one Macrocell. Though the 5C121 has Macrocells that can accept up to 16 product terms, all Macrocells are not that wide.

Let's look at an example. Equation 1 represents one of the eight Boolean expressions necessary to implement a digital crosspoint switch. Logically, this expression selects one of eight input signals (I0-I7), and routes that signal to Q0. Data bits D0, D1, and D2 select one of the eight input lines. In this case, data bits !D3, !D4, and !D5 select output Q0. (The exclamation point is used to indicate a logical complement of the signal.) Equations for Q1 through Q7 are very similar and will be discussed later.

$$\begin{aligned}
 Q0 = & ( I0 \times !D2 \times !D1 \times !D0 \\
 & + I1 \times !D2 \times !D1 \times D0 \\
 & + I2 \times !D2 \times D1 \times !D0 \\
 & + I3 \times !D2 \times D1 \times D0 \\
 & + I4 \times D2 \times !D1 \times !D0 \\
 & + I5 \times D2 \times !D1 \times D0 \\
 & + I6 \times D2 \times D1 \times !D0 \\
 & + I7 \times D2 \times D1 \times D0 ) \times !D5 \times !D4 \times !D3; \quad (1)
 \end{aligned}$$

$$\begin{aligned}
 SELECTEQ = & I0 \times !D2 \times !D1 \times !D0 \\
 & + I1 \times !D2 \times !D1 \times D0 \\
 & + I2 \times !D2 \times D1 \times !D0 \\
 & + I3 \times !D2 \times D1 \times D0 \\
 & + I4 \times D2 \times !D1 \times !D0 \\
 & + I5 \times D2 \times !D1 \times D0 \\
 & + I6 \times D2 \times D1 \times !D0 \\
 & + I7 \times D2 \times D1 \times D0; \quad (2)
 \end{aligned}$$

Equation 2 contains the terms that will be common to all eight output equations. Both equations in this case contain eight product terms. By treating equation 2 as one common signal and routing that signal through combinational feedback, we can reduce the number of product terms in equations Q0 thru Q7 to one p-term each. The advantage is that the outputs can now be placed in any of the 24 I/O Macrocells available in the 5C121. In addition, the 5C121 contains four buried registers. (Buried registers have no output and are used solely for feedback.) If a buried register is available, iPLDs (Intel's Programmable Logic Development System) will automatically assign the No Output — Combinational Feedback function to a buried register. This increases the flexibility for pin assignments and makes

**COMBINATIONAL FEEDBACK**

(Continued)

p-terms available in case a design change is needed. Equations 3 thru 10 reflect this improvement.

- Q0 = SELECTEQ × ID5 × ID4 × ID3; (3)
- Q1 = SELECTEQ × ID5 × ID4 × D3; (4)
- Q2 = SELECTEQ × ID5 × D4 × ID3; (5)
- Q3 = SELECTEQ × ID5 × D4 × D3; (6)
- Q4 = SELECTEQ × D5 × ID4 × ID3; (7)
- Q5 = SELECTEQ × D5 × ID4 × D3; (8)
- Q6 = SELECTEQ × D5 × D4 × ID3; (9)
- Q7 = SELECTEQ × D5 × D4 × D3; (10)

$$Q1 = \text{SELECTEQ} \times \text{ID5} \times \text{ID4} \times \text{D3} + !( \text{ID5} \times \text{ID4} \times \text{D3} ) \times \text{Q1-fdbk}; \quad (12)$$

$$Q2 = \text{SELECTEQ} \times \text{ID5} \times \text{D4} \times \text{ID3} + !( \text{ID5} \times \text{D4} \times \text{ID3} ) \times \text{Q2-fdbk}; \quad (13)$$

$$Q3 = \text{SELECTEQ} \times \text{ID5} \times \text{D4} \times \text{D3} + !( \text{ID5} \times \text{D4} \times \text{D3} ) \times \text{Q3-fdbk}; \quad (14)$$

$$Q4 = \text{SELECTEQ} \times \text{D5} \times \text{ID4} \times \text{D3} + !( \text{D5} \times \text{ID4} \times \text{ID3} ) \times \text{Q4-fdbk}; \quad (15)$$

$$Q5 = \text{SELECTEQ} \times \text{D5} \times \text{ID4} \times \text{D3} + !( \text{D5} \times \text{ID4} \times \text{D3} ) \times \text{Q5-fdbk}; \quad (16)$$

$$Q6 = \text{SELECTEQ} \times \text{D5} \times \text{D4} \times \text{ID3} + !( \text{D5} \times \text{D4} \times \text{ID3} ) \times \text{Q6-fdbk}; \quad (17)$$

$$Q7 = \text{SELECTEQ} \times \text{D5} \times \text{DR} \times \text{D3} + !( \text{D5} \times \text{D4} \times \text{DE} ) \times \text{Q7-fdbk}; \quad (18)$$

Equations 11 thru 18 are all that are necessary to implement a digital crosspoint switch with the output hold feature. Each equation contains only four product terms when written in the expanded form and could therefore fit into any Macrocell in the 5C121. The appendix contains the report and ADF files generated by the iPLDs software.

**REGISTERED FEEDBACK**

Registered feedback is also employed in a variety of applications such as counters and state machines. In this particular example, the registered feedback signal can be used to hold the deselected outputs of the switch at their previous level until that output is selected again. This is accomplished by simply "ANDing" the feedback signal with the inversion of the output select signal. The result is then "ORed" with the equation for the given output. Holding the previous output might be useful in control applications or when interfacing to slow peripherals. Equations 11 thru 18 are the result.

$$Q0 = \text{SELECTEQ} \times \text{ID5} \times \text{ID4} \times \text{ID3} + !( \text{D5} \times \text{ID4} \times \text{ID3} ) \times \text{Q0-fdbk}; \quad (11)$$

**TIMING ANALYSIS**

Figure 2 shows the internal delay paths associated with this design in the 5C121. The frequency at which the 5C121 may be clocked can be determined by examining the internal delay elements of the 5C121. These include the input delay (Tin), two array delays (Tad), and the combinational feedback delay (Tcf). Table 1 gives the simulation data for each of these paths in a 5C121-50.

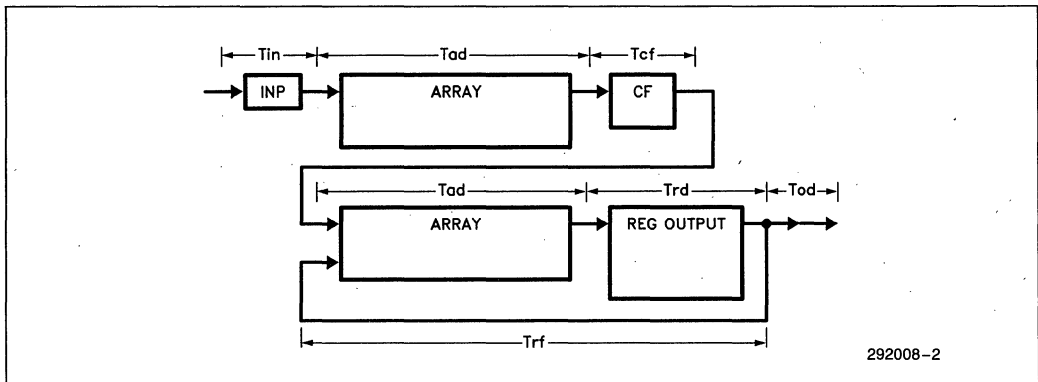


Figure 2. Crosspoint Delay Path

292008-2

**TIMING ANALYSIS** (Continued)

**Table 1. 5C121-50 Simulation Data**

Model Parameter	Delay (ns)
Tad	38
Trd	7
Tod	8
Tin	10
Tic	8
Trf	5
Tcf	5

The sum of the delays before the register input equal the set-up time  $T_{su}$  with reference to the internal clock. By subtracting the input clock delay  $T_{ic}$  we shift the reference to the external clock pin. The set-up time with reference to external signals is shown in equation 19. Inverting this signal yields the maximum clock frequency,  $f_{max}$ . The maximum clock frequency is shown in equation 20.

$$T_{su} = T_{in} + 2T_{ad} + T_{cf} - T_{ic}; \tag{19}$$

$$f_{max} = 1 / T_{su} \tag{20}$$

Therefore, this configuration of the 5C121-50 could be clocked at 10 MHz, allowing a data transfer rate of 10 Mbits/second. By paralleling six 5C121s together, eight

bits could be switched per cycle. Figure 3 shows the timing diagram for this configuration of the 5C121 digital crosspoint switch. Included in the appendix is the Advanced Design File (ADF), Logic Equation File (LEF), and Utilization report generated by Intel's Programmable Logic Software (iPLS) for this design.

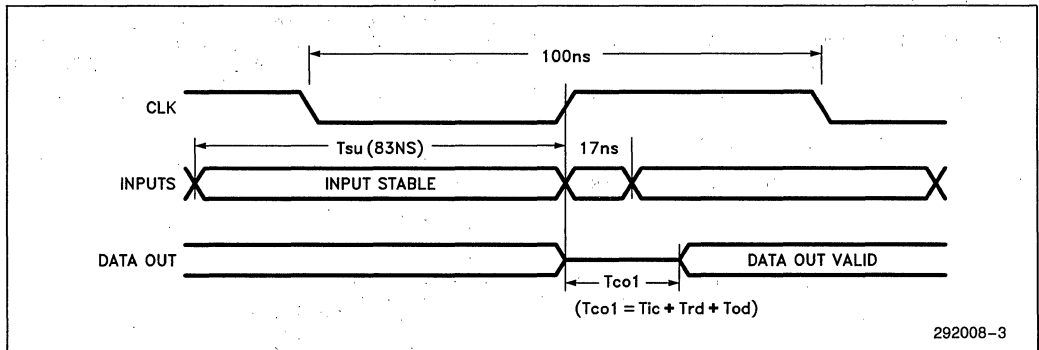
**INPUT LATCHES**

One point must be raised about Figure 3. Notice that the time allowed for external data set-up is only 17 ns. Therefore, 17 ns after the rising edge of the clock, data must be stable and remain stable at the input pins until the next clock pulse. In most systems this would be a very stringent requirement. Fortunately the 5C121 has the ability to latch the data at the input pins with 7475 type transparent latches. Employing this feature eases the data set-up requirement as shown in Figure 4.

**SUMMARY**

The flexible architecture of the 5C121 gives the designer a variety of options for input and output configurations. Inputs may be latched to ease system timing requirements. Outputs may be clocked for synchronous systems or fed directly out as asynchronous signals. Feedback can be used to reduce product term requirements, to save present state information for state machines and counters, or simply to hold deselected outputs as shown in this example. Imagine the possibilities.

J. R. Donnell  
PLDO Applications



**Figure 3. Crosspoint Timing Diagram**

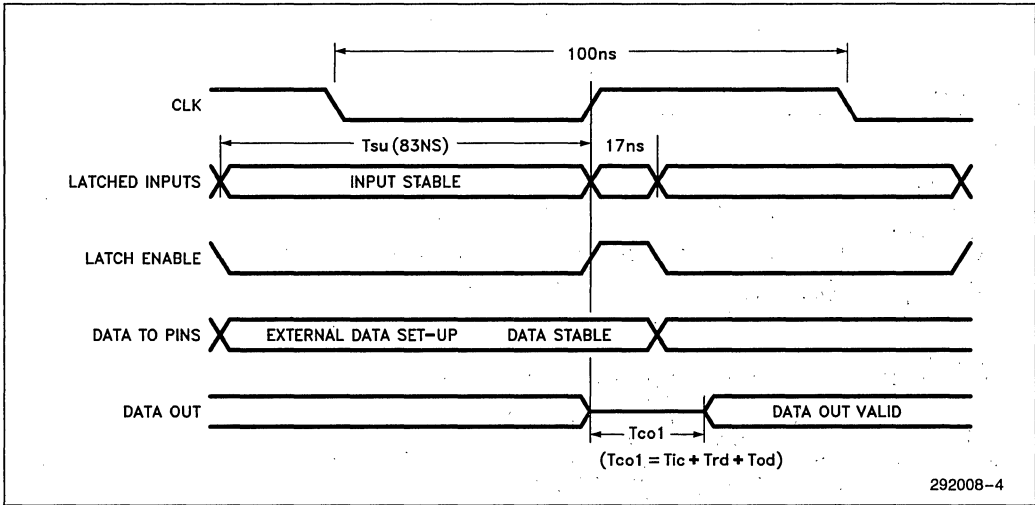


Figure 4. Crosspoint Timing Diagram with Input Latches



## APPENDIX

## ADF File

```

0
5C121
Digital Crosspoint Switch
LB Version 3.0, Baseline 17x, 9/26/85
PART: 5C121
INPUTS: I00@37, I01@36, I02@35, I03@34, I04@8, I05@9, I06@10, I07@11, I10@33, I11@32
        ; I12@31, I13@30, I14@29, I15@28, I16@27, I17@26, CLK@38, D0@2, D1@3, D2@4, D3@5
        , D4@6, D5@7, ILE@1
OUTPUTS: Q00@12, Q01@13, Q02@14, Q03@15, Q04@16, Q05@17, Q06@18, Q07@19, Q10@24, Q11@23
        , Q12@22, Q13@21

NETWORK:
Q00, Q00FBK = RORF (Q00D, CLK, GND, GND, VCC) % BIT 0 OUTPUTS %
Q01, Q01FBK = RORF (Q01D, CLK, GND, GND, VCC)
Q02, Q02FBK = RORF (Q02D, CLK, GND, GND, VCC)
Q03, Q03FBK = RORF (Q03D, CLK, GND, GND, VCC)
Q04, Q04FBK = RORF (Q04D, CLK, GND, GND, VCC)
Q05, Q05FBK = RORF (Q05D, CLK, GND, GND, VCC)
Q06, Q06FBK = RORF (Q06D, CLK, GND, GND, VCC)
Q07, Q07FBK = RORF (Q07D, CLK, GND, GND, VCC)
Q10, Q10FBK = RORF (Q10D, CLK, GND, GND, VCC) % 4 OF THE 8, BIT 0 OUTPUTS%
Q11, Q11FBK = RORF (Q11D, CLK, GND, GND, VCC)
Q12, Q12FBK = RORF (Q12D, CLK, GND, GND, VCC)
Q13, Q13FBK = RORF (Q13D, CLK, GND, GND, VCC)
CLK = INP (CLK)
D5 = LIMP (D5, ILE) % OUTPUT SELECT CONTROL BITS %
ILE = INP (ILE)
D4 = LIMP (D4, ILE)
D3 = LIMP (D3, ILE)
D2 = LIMP (D2, ILE) % INPUT SELECT CONTROL BITS %
D1 = LIMP (D1, ILE)
D0 = LIMP (D0, ILE)
I00 = LIMP (I00, ILE)
I01 = LIMP (I01, ILE)
I02 = LIMP (I02, ILE)
I03 = LIMP (I03, ILE)
I04 = LIMP (I04, ILE)
I05 = LIMP (I05, ILE)
I06 = LIMP (I06, ILE)
I07 = LIMP (I07, ILE)
I10 = LIMP (I10, ILE) % INPUTS FOR BIT 1 SWITCH %
I11 = LIMP (I11, ILE)
I12 = LIMP (I12, ILE)
I13 = LIMP (I13, ILE)
I14 = LIMP (I14, ILE)
I15 = LIMP (I15, ILE)
I16 = LIMP (I16, ILE)
I17 = LIMP (I17, ILE)
SELECTEQ0F = NOCF (SELECTEQ0)
SELECTEQ1F = NOCF (SELECTEQ1)
EQUATIONS:
Q00D = SELECTEQ0F*!D5*!D4*!D3
      + !(D5*!D4*!D3)*Q00FBK;
Q01D = SELECTEQ0F*!D5*!D4* D3
      + !(D5*!D4* D3)*Q01FBK;
Q02D = SELECTEQ0F*!D5* D4*!D3
      + !(D5* D4*!D3)*Q02FBK;
Q03D = SELECTEQ0F*!D5* D4* D3
      + !(D5* D4* D3)*Q03FBK;
Q04D = SELECTEQ0F* D5*!D4*!D3
      + !( D5*!D4*!D3)*Q04FBK;
Q05D = SELECTEQ0F* D5*!D4* D3

```

292008-5

## ADF File (Continued)

```
      + !( D5*!D4* D3)*Q05FBK;
Q06D = SELECTEQ0F* D5* D4*!D3
      + !( D5* D4*!D3)*Q06FBK;
Q07D = SELECTEQ0F* D5* D4* D3
      + !( D5* D4* D3)*Q07FBK;
Q10D = SELECTEQ1F*!D5*!D4*!D3
      + !(!D5*!D4*!D3)*Q10FBK;
Q11D = SELECTEQ1F*!D5*!D4* D3
      + !(!D5*!D4* D3)*Q11FBK;
Q12D = SELECTEQ1F*!D5* D4*!D3
      + !(!D5* D4*!D3)*Q12FBK;
Q13D = SELECTEQ1F*!D5* D4* D3
      + !(!D5* D4* D3)*Q13FBK;
SELECTEQ0 = I00*!D2*!D1*!D0      % COMMON EQUATION FOR BIT 0 %
      + I01*!D2*!D1*D0
      + I02*!D2*D1*!D0
      + I03*!D2*D1*D0
      + I04*D2*!D1*!D0
      + I05*D2*!D1*D0
      + I06*D2*D1*!D0
      + I07*D2*D1*D0;
SELECTEQ1 = I10*!D2*!D1*!D0      % COMMON EQUATION FOR BIT 1 %
      + I11*!D2*!D1*D0
      + I12*!D2*D1*!D0
      + I13*!D2*D1*D0
      + I14*D2*!D1*!D0
      + I15*D2*!D1*D0
      + I16*D2*D1*!D0
      + I17*D2*D1*D0;
END$
```

292008-6

## LEF File

JR Donnell  
Intel  
January 24, 1986

0  
5C121  
Digital Crosspoint Switch  
LB Version 3.0, Baseline 17x, 9/26/85  
PART:

5C121

## INPUTS:

I00e37, I01e36, I02e35, I03e34, I04e8, I05e9, I06e10, I07e11, I10e33,  
I11e32, I12e31, I13e30, I14e29, I15e28, I16e27, I17e26, CLKe38, D0e2,  
D1e3, D2e4, D3e5, D4e6, D5e7, ILEe1

## OUTPUTS:

Q00e12, Q01e13, Q02e14, Q03e15, Q04e16, Q05e17, Q06e18, Q07e19, Q10e24,  
Q11e23, Q12e22, Q13e21

## NETWORK:

CLK = INP(CLK)  
ILE = INP(ILE)  
I00 = LIMP(I00, ILE)  
I01 = LIMP(I01, ILE)  
I02 = LIMP(I02, ILE)  
I03 = LIMP(I03, ILE)  
I04 = LIMP(I04, ILE)  
I05 = LIMP(I05, ILE)  
I06 = LIMP(I06, ILE)  
I07 = LIMP(I07, ILE)  
I10 = LIMP(I10, ILE)  
I11 = LIMP(I11, ILE)  
I12 = LIMP(I12, ILE)  
I13 = LIMP(I13, ILE)  
I14 = LIMP(I14, ILE)  
I15 = LIMP(I15, ILE)  
I16 = LIMP(I16, ILE)  
I17 = LIMP(I17, ILE)  
D0 = LIMP(D0, ILE)  
D1 = LIMP(D1, ILE)  
D2 = LIMP(D2, ILE)  
D3 = LIMP(D3, ILE)  
D4 = LIMP(D4, ILE)  
D5 = LIMP(D5, ILE)  
Q00, Q00FBK = RORF(Q00D, CLK, GND, GND, VCC)  
Q01, Q01FBK = RORF(Q01D, CLK, GND, GND, VCC)  
Q02, Q02FBK = RORF(Q02D, CLK, GND, GND, VCC)  
Q03, Q03FBK = RORF(Q03D, CLK, GND, GND, VCC)  
Q04, Q04FBK = RORF(Q04D, CLK, GND, GND, VCC)  
Q05, Q05FBK = RORF(Q05D, CLK, GND, GND, VCC)  
Q06, Q06FBK = RORF(Q06D, CLK, GND, GND, VCC)  
Q07, Q07FBK = RORF(Q07D, CLK, GND, GND, VCC)  
Q10, Q10FBK = RORF(Q10D, CLK, GND, GND, VCC)  
Q11, Q11FBK = RORF(Q11D, CLK, GND, GND, VCC)  
Q12, Q12FBK = RORF(Q12D, CLK, GND, GND, VCC)  
Q13, Q13FBK = RORF(Q13D, CLK, GND, GND, VCC)  
SELECTEQ0F = NOCF(SELECTEQ0)  
SELECTEQ1F = NOCF(SELECTEQ1)

## EQUATIONS:

SELECTEQ1 = I10 \* D2' \* D1' \* D0'  
+ D2 \* D1' \* D0' \* I14  
+ D2' \* D1 \* D0' \* I12  
+ D2' \* D1' \* D0 \* I11  
+ D2 \* D1 \* D0' \* I16  
+ D2 \* D1' \* D0 \* I15  
+ D2' \* D1 \* D0 \* I13

## LEF File (Continued)

```

+ D2 * D1 * D0 * I17;

SELECTEQ0 = I00 * D2' * D1' * D0'
+ D2 * D1' * D0' * I04
+ D2' * D1 * D0' * I02
+ D2' * D1' * D0 * I01
+ D2 * D1 * D0' * I06
+ D2 * D1' * D0 * I05
+ D2' * D1 * D0 * I03
+ D2 * D1 * D0 * I07;

Q13D = D3' * Q13FBK
+ D4' * Q13FBK
+ D5 * Q13FBK
+ SELECTEQ1F * D5' * D4 * D3;

Q12D = D4' * Q12FBK
+ D3 * Q12FBK
+ D5 * Q12FBK
+ SELECTEQ1F * D5' * D4 * D3';

Q11D = D3' * Q11FBK
+ D4 * Q11FBK
+ D5 * Q11FBK
+ SELECTEQ1F * D5' * D4' * D3;

Q10D = D3 * Q10FBK
+ D4 * Q10FBK
+ D5 * Q10FBK
+ SELECTEQ1F * D5' * D4' * D3';

Q07D = D3' * Q07FBK
+ D4' * Q07FBK
+ D5' * Q07FBK
+ SELECTEQ0F * D5 * D4 * D3;

Q06D = D4' * Q06FBK
+ D5' * Q06FBK
+ D3 * Q06FBK
+ SELECTEQ0F * D5 * D4 * D3';

Q05D = D3' * Q05FBK
+ D5' * Q05FBK
+ D4 * Q05FBK
+ SELECTEQ0F * D5 * D4' * D3;

Q04D = D5' * Q04FBK
+ D3 * Q04FBK
+ D4 * Q04FBK
+ SELECTEQ0F * D5 * D4' * D3';

Q03D = D3' * Q03FBK
+ D4' * Q03FBK
+ D5 * Q03FBK
+ SELECTEQ0F * D5' * D4 * D3;

Q02D = D4' * Q02FBK
+ D3 * Q02FBK
+ D5 * Q02FBK
+ SELECTEQ0F * D5' * D4 * D3';

Q01D = D3' * Q01FBK
+ D4 * Q01FBK
+ D5 * Q01FBK
+ SELECTEQ0F * D5' * D4' * D3;

Q00D = D3 * Q00FBK
+ D4 * Q00FBK
+ D5 * Q00FBK
+ SELECTEQ0F * D5' * D4' * D3';

```

292008-13

ENDS

292008-14

RPT File

Logic Optimizing Compiler Utilization Report

\*\*\*\* Design implemented successfully

JR Donnell  
Intel  
January 24, 1986

0  
5C121  
Digital Crosspoint Switch  
LB Version 3.0, Baseline 17x, 9/26/85

5C121

ILE	1	40	Vcc
D0	2	39	Vcc
D1	3	38	CLK
D2	4	37	I00
D3	5	36	I01
D4	6	35	I02
D5	7	34	I03
I04	8	33	I10
I05	9	32	I11
I06	10	31	I12
I07	11	30	I13
Q00	12	29	I14
Q01	13	28	I15
Q02	14	27	I16
Q03	15	26	I17
Q04	16	25	GND
Q05	17	24	Q10
Q06	18	23	Q11
Q07	19	22	Q12
GND	20	21	Q13

\*\*INPUTS\*\*

Name	Pin	Resource	MCell #	PTerms	MCells	Feeds: OE	Clear	Clock
ILE	1	INP	-	-	-	-	-	Latch
D0	2	LINP	-	-	13 15	-	-	-
D1	3	LINP	-	-	13 15	-	-	-
D2	4	LINP	-	-	13 15	-	-	-
D3	5	LINP	-	-	9 10 11 12 17 18 19 20 21	-	-	-

RPT File (Continued)

					22			
					23			
					24			
D4	6	LINP	-	-	9	-	-	-
					10			
					11			
					12			
					17			
					18			
					19			
					20			
					21			
					22			
					23			
					24			
D5	7	LINP	-	-	9	-	-	-
					10			
					11			
					12			
					17			
					18			
					19			
					20			
					21			
					22			
					23			
					24			
I04	8	LINP	28	0/ 4	15	-	-	-
I05	9	LINP	27	0/10	15	-	-	-
I06	10	LINP	26	0/ 8	15	-	-	-
I07	11	LINP	25	0/ 6	15	-	-	-
I17	26	LINP	7	0/10	13	-	-	-
I16	27	LINP	6	0/ 8	13	-	-	-
I15	28	LINP	5	0/ 6	13	-	-	-
I14	29	LINP	4	0/ 6	13	-	-	-
I13	30	LINP	3	0/ 8	13	-	-	-
I12	31	LINP	2	0/10	13	-	-	-
I11	32	LINP	1	0/ 4	13	-	-	-
I10	33	LINP	-	-	13	-	-	-
I03	34	LINP	-	-	15	-	-	-
I02	35	LINP	-	-	15	-	-	-
I01	36	LINP	-	-	15	-	-	-
I00	37	LINP	-	-	15	-	-	-
CLK	38	INP	-	-	-	-	-	Reg

RPT File (Continued)

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

Name	Pin	Resource	MCell #	PTerms	MCells	Feeds:	
						OE	Clear
Q00	12	RORF	24	4/ 6	24	-	-
Q01	13	RORF	23	4/ 8	23	-	-
Q02	14	RORF	22	4/10	22	-	-
Q03	15	RORF	21	4/ 4	21	-	-
Q04	16	RORF	20	4/12	20	-	-
Q05	17	RORF	19	4/ 4	19	-	-
Q06	18	RORF	18	4/ 8	18	-	-
Q07	19	RORF	17	4/ 8	17	-	-
Q13	21	RORF	12	4/ 8	12	-	-
Q12	22	RORF	11	4/ 8	11	-	-
Q11	23	RORF	10	4/ 4	10	-	-
Q10	24	RORF	9	4/12	9	-	-

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

Name	Pin	Resource	MCell #	PTerms	MCells	Feeds:	
						OE	Clear
	-	NOCF	13	8/ 8	9	-	-
					10		
					11		
					12		
	-	NOCF	15	8/ 8	17	-	-
					18		
					19		
					20		
					21		
					22		
					23		
					24		

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

Name	Pin	Resource	MCell	PTerms
-	25	-	8	4
-	NA	-	14	8
-	NA	-	16	8

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

97% Pins  
 89% MacroCells  
 30% Pterms



**APPLICATION  
NOTE**

**AP-272**

June 1986

**The 5C060  
Unification of a CHMOS System**

**J. R. DONNELL**  
PROGRAMMABLE LOGIC APPLICATIONS  
INTEL CORPORATION

Order Number: 292009-003



**INTRODUCTION**

From an outside glance, the world of computers and microprocessors seems filled with dedicated ICs that fulfill a variety of system needs. Upon closer inspection we find that designers must still reach into their bag of random logic to link together all of the parts of the system. It seems a shame to stuff a board full of high powered peripherals and still have portions of that board wasted on decoders, latches, and other miscellaneous random logic.

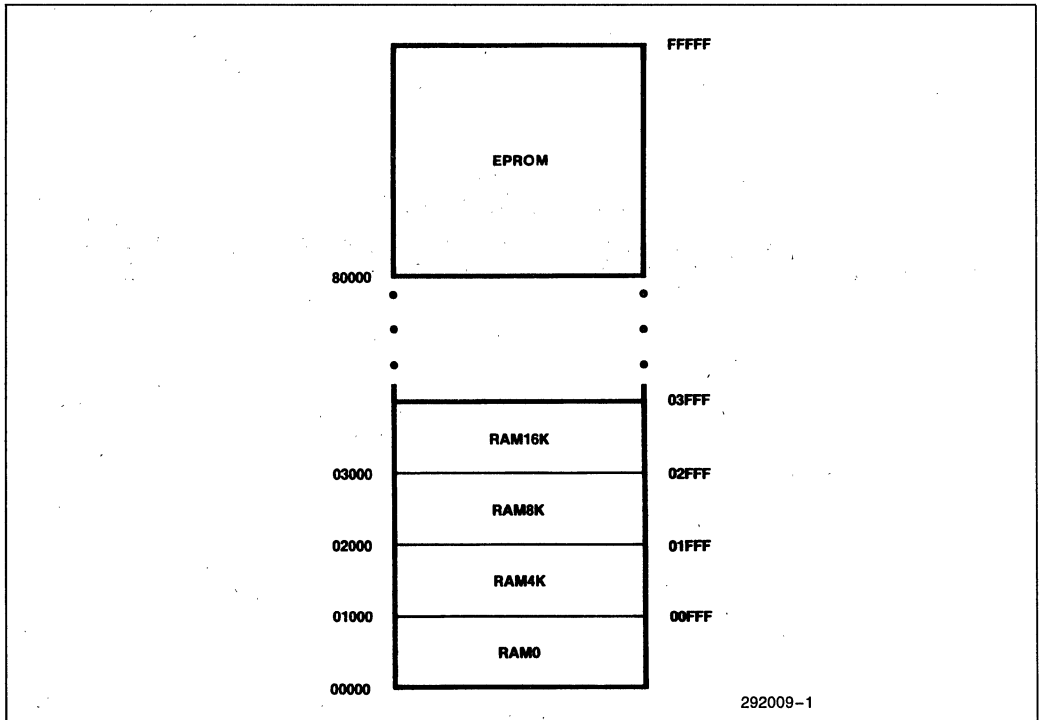
True, programmable logic has been around a long time. But that logic is somewhat rigid in form, one time programmable, and can also double as space heaters. These devices are totally unacceptable for a CMOS system. What is needed is a flexible PLA architecture, erasability for prototyping, and CMOS for low power. In addition, for this particular application the device must perform from static operation to 10 MHz.

**OBJECTIVE**

This application note covers the design of three separate circuits for Intel's CHMOS Design Kit. The functions performed by the 5C060 are: Memory decoding, wait state generation, and the power down circuitry for the 80C88 system clock.

**MEMORY DECODING**

The system in question supports one 32K bank of EPROM memory, and four banks of 4K static RAM. Figure 1 shows the memory map of this system. Address lines A19, A13, and A12 will be used to decode the address space. PWR\_DWN and S2\_MIO serve as enables. In addition, to avoid data bus contention signals memory read (MRDC) and advanced memory write (AMWC) are decoded along with the address lines for RAM chip selects. This is necessary for devices without output enables (OE) on multiplexed address/data busses.



**Figure 1. 80C88 Memory Map**

Figure 2 shows a discrete implementation of the chip select decoding logic.

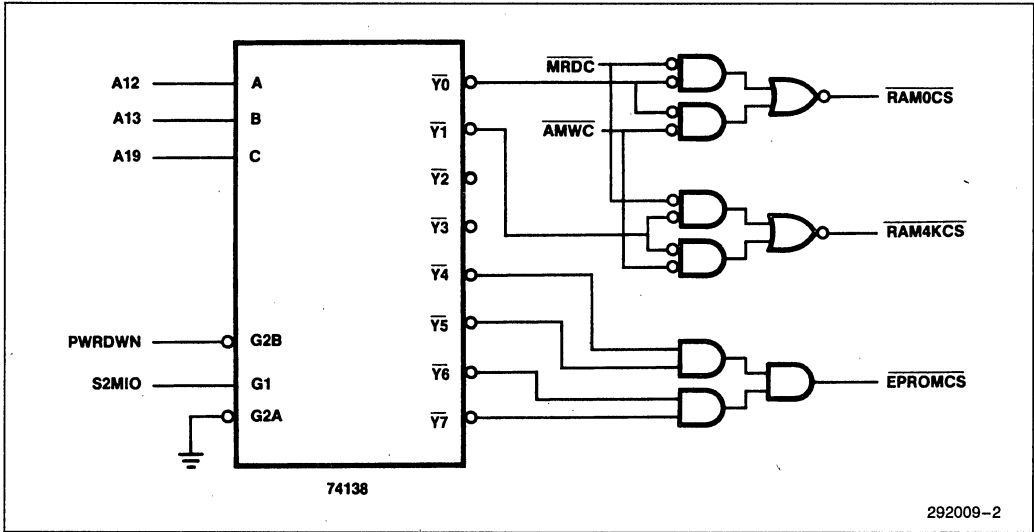


Figure 2. Discrete Decoding Logic Solution

Several options for entering this design are available through Intel's Programmable Logic Development System (iPLDS). (For a more complete description of iPLDS the reader is referred to the iPLDS data sheet.) The design entry vehicle chosen for this application note is the Logic Builder. (Logic Builder is an interactive netlist method of design entry especially suited to Boolean equation entry and entry from existing schematics.) Several reasons are behind this decision. First, the Logic Builder software is included in iPLDS. In addition, Logic Builder entry is very fast, the designer may choose either netlist entry or Boolean equations, and finally, the Logic Builder software makes additions and corrections of design very easy.

Using Logic Builder, the first step for this design is to determine the equations for the 3 to 8 decoder shown in Figure 2. These equations are simply the decoding of the address lines ANDed with the enable signal. Equations 0 thru 8 implement the decoding function of Figure 2.

- /Y0 = /A19\*/A13\*/A12\*ENABLE; (0)
- /Y1 = /A19\*/A13\*A12\*ENABLE; (1)
- /Y2 = /A19\*A13\*/A12\*ENABLE; (2)
- /Y3 = /A19\*A13\*A12\*ENABLE; (3)
- /Y4 = A19\*/A13\*/A12\*ENABLE; (4)
- /Y5 = A19\*/A13\*A12\*ENABLE; (5)
- /Y6 = A19\*A13\*/A12\*ENABLE; (6)
- /Y7 = A19\*A13\*A12\*ENABLE; (7)
- ENABLE = /PWRDWN\*S2MIO; (8)

Armed with this knowledge it becomes trivial to enter the circuit of Figure 2 into Logic Builder. Included in the Appendix is the Advanced Design File (ADF) created by Logic Builder for this circuit (ADF-1). Typically the ADF would now be submitted to the Logic Optimizing Compiler (LOC) for Boolean minimization and design fitting. In this case we have used only a small portion of the logic available in the 5C060 so let us continue with the wait state generator and power down circuitry.

### Power Down

Since this design is based on the 80C88 we can actually stop the system clock for extended periods of time and power back up as if nothing had occurred. The circuit to achieve this power down is shown in Figure 3.

As long as the PWRDWN signal is low the 82C84 clock output is OR'ed with a logical zero from the PWRDWN flip-flop. As a result the 82C84 drives the 80C88 system clock. If PWRDWN goes HIGH, the rising edge of the next 82C84 clock will set the output of the PWRDWN flip-flop HIGH inhibiting the fall of the next clock cycle. The 80C88 system clock will remain HIGH until PWRDWN goes LOW and the PWRDWN flip-flop is clocked from the 82C84 clock. Using this configuration we avoid partial clock cycles for the 80C88 system clock.

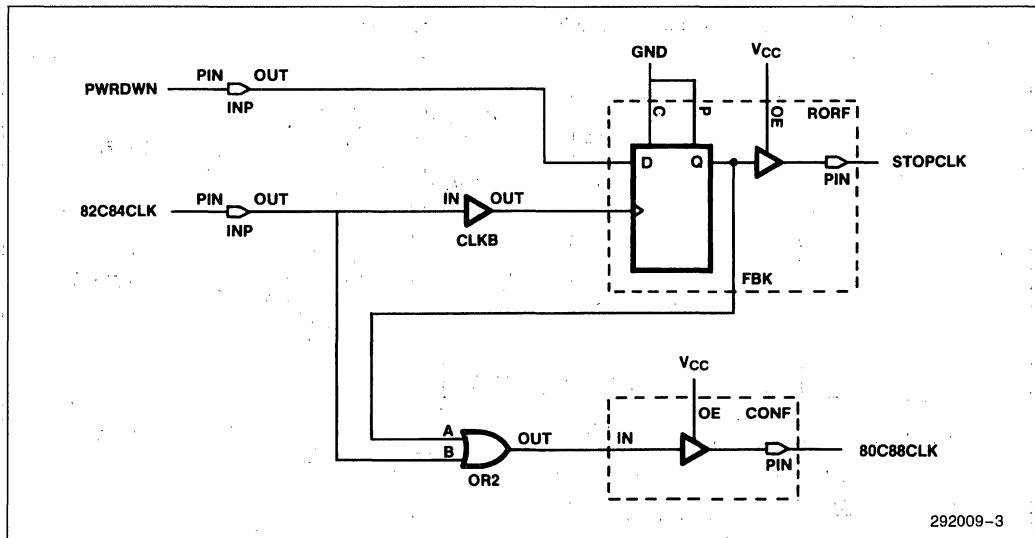


Figure 3. 80C88 Power Down Circuit

Again, entering this circuit into Logic Builder is trivial. In fact it can be added directly to the decoder circuit shown above. The ADF file for this addition is shown in the appendix under ADF-2.

## Wait States

The majority of memory and peripheral devices which fail to operate at the maximum CPU frequency typically do not require more than one wait state. The circuit shown in Figure 4 is an example of a simple wait state generator. The circuit operation is as follows. Given that a memory location requiring a wait state has been selected, ALE in conjunction with /WAITCS will clear the flip-flop—driving the 82C84RDY line high low. The 82C84 samples the RDY line during T2 of the 80C88 bus cycle, and in this case detects a wait state. The rising edge of T2 then clocks the 82C84RDY line high thereby inserting only one wait state.

Once again, adding this circuit to the existing decoder and power down design is simple. The final ADF file is given in the appendix under ADF-3. Once the final design has been completed the ADF is submitted to the Logic Optimizing Compiler. LOC compiles the design, performs Boolean minimization, and fits the design into the target EPLD. In addition, LOC produces two files. The JEDEC programming file, the Logic Equation File

(LEF), and the Utilization Report. These are also included in the appendix for each step in this design process.

## LOC FILES

### The JEDEC File

The JEDEC file is analogous to the object code file that is used to program EPROMs. This file is used by the Logic Programming Software (LPS) to program Intel's EPLDs.

### The LEF File

The LEF file is an optional file produced by the compiler. The LEF file contains the minimized Boolean equations which resulted from the original ADF. Some interesting points can be raised concerning the LEF file. Looking at LEF-3, first recall that the EPROM chip select was a function of A19, A13, A12, and the enable signals. It turns out that after minimization the EPROM chip select depends only on A19 and the enable signals (/PWRDWN and S2MIO). This is shown in the LEF file. One other point, the initial wait state circuitry employed a JK flip-flop. The compiler automatically minimized this circuit into a D-type flip-flop with feedback achieving the same functionality.

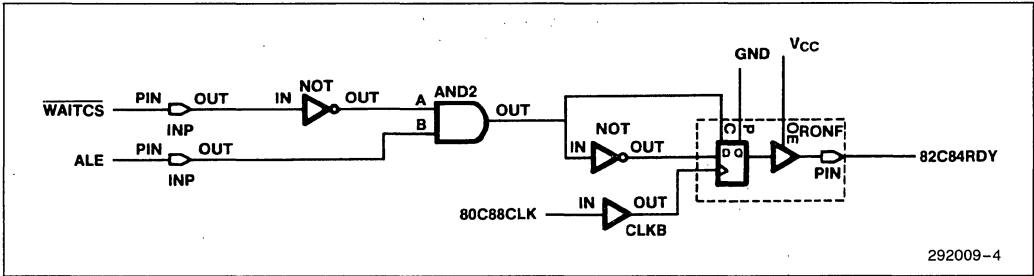


Figure 4. Single Wait State Generator for the 80C88

### The Utilization Report

Finally, the Utilization Report contains the pin-out for the design, information about the architectural layout of the design, and a percent utilization for pins, macrocells, and product terms. Examining the utilization report for this design we find that two of the sixteen macrocells are still available. We could therefore add more functionality in the same 24 pin package. Possible additions would be more memory decoding, invalid memory detection, additional wait state generators, etc. One point should be raised: The circuitry designed in this applications note is relatively simple compared to the complex logic functions that could be implemented in the 5C060.

### SUMMARY

The designs shown in this applications note are typical requirements of any microprocessor system. The 5C060 provided a single chip solution to bind together the primary elements of that system. Few other types of programmable logic could implement the same logic in a single package. None could do it in CMOS erasable logic. The 5C060 has room for more.

## APPENDIX

## ADF-1

```

JR Donnell
Intel
January 31, 1986
5C060
0
5C060
Decoder for 80C88 system - 16K RAM and upper 512K EPROM
IR Version 3.0, Baseline 17x, 9/26/85
PART: 5C060
INPUTS: A19, A13, A12, PWRDWN, S2MIO, AMWC, MRDC
OUTPUTS: RAMOCS, RAM4KCS, RAM8KCS, RAM16KCS, RPROMCS
NETWORK:
RAMOCS = CONF (RAMOCS, VCC)
RAM4KCS = CONF (RAM4KCS, VCC)
RAM8KCS = CONF (RAM8KCS, VCC)
RAM16KCS = CONF (RAM16KCS, VCC)
RPROMCS = CONF (RPROMCS, VCC)
A19 = INP (A19)
A13 = INP (A13)
A12 = INP (A12)
PWRDWN = INP (PWRDWN)
S2MIO = INP (S2MIO)
MRDC = INP (MRDC)
AMWC = INP (AMWC)
EQUATIONS:
RAM8KCS = ((MRDC*Y2
+ /AMWC*Y2):
RAM16KCS = ((MRDC*Y3
+ /AMWC*Y3):
RPROMCS = ((Y7
+ /Y6
+ /Y5
+ /Y4):
Y7 = ((A19*A13*A12*RNABLE):
Y6 = ((A19*A13*/A12*ENABLE):
Y5 = ((A19*/A13*A12*RNABLE):
Y4 = ((A19*/A13*/A12*ENABLE):
RNABLE = /PWRDWN*S2MIO:
Y3 = ((A19*A13*A12*ENABLE):
Y2 = ((A19*A13*/A12*RNABLE):
RAM4KCS = ((MRDC*Y1
+ /AMWC*Y1):
Y1 = ((A19*/A13*A12*ENABLE):
RAMOCS = ((MRDC*Y0
+ /AMWC*Y0):
Y0 = ((A19*/A13*/A12*RNABLE):
END$

```

292009-5

## ADF-2

```

JR Donnell
Intel
January 31, 1986
5C060
0
5C060
Decoder for 80C88 system - 16K RAM and upper 512K EPROM
Plus power down circuit
LB Version 3.0, Baseline 17x, 9/26/85
PART: 5C060
INPUTS: A19, A13, A12, PWRDWN, S2MIO, AMWC, MRDC, R2C84CLK
OUTPUTS: RAM0CS, RAM4KCS, RAM8KCS, RAM16KCS, EPROMCS, STOPCLK, R0C88CLK
NETWORK:
RAM0CS = CONF (RAM0CS, VCC)
RAM4KCS = CONF (RAM4KCS, VCC)
RAM8KCS = CONF (RAM8KCS, VCC)
RAM16KCS = CONF (RAM16KCS, VCC)
EPROMCS = CONF (EPROMCS, VCC)
STOPCLK, STOPCLKF = RORF (PWRDWN, R2C84CLK, GND, GND, VCC)
R0C88CLK = CONF (R0C88CLK, VCC)
PWRDWN = INP (PWRDWN)
R2C84CLKB = CLKB (R2C84CLK)
R0C88CLK = OR (STOPCLKF, R2C84CLK)
R2C84CLK = INP (R2C84CLK)
A19 = INP (A19)
A13 = INP (A13)
A12 = INP (A12)
S2MIO = INP (S2MIO)
MRDC = INP (MRDC)
AMWC = INP (AMWC)
EQUATIONS:
RAM0CS = //(MRDC*Y0
+ /AMWC*Y0);
RAM4KCS = //(MRDC*Y1
+ /AMWC*Y1);
RAM8KCS = //(MRDC*Y2
+ /AMWC*Y2);
RAM16KCS = //(MRDC*Y3
+ /AMWC*Y3);
EPROMCS = //(Y7
+ /Y6
+ /Y5
+ /Y4);
Y0 = //(A19*/A13*/A12*ENABLE);
Y1 = //(A19*/A13*/A12*ENABLE);
Y2 = //(A19*/A13*/A12*ENABLE);
Y3 = //(A19*/A13*/A12*ENABLE);
Y7 = //(A19*/A13*/A12*ENABLE);
Y6 = //(A19*/A13*/A12*ENABLE);
Y5 = //(A19*/A13*/A12*ENABLE);
Y4 = //(A19*/A13*/A12*ENABLE);
ENABLE = /PWRDWN*S2MIO;
RND$

```

292009-6

JR Donnell  
 Intel  
 January 31, 1986  
 5C060  
 0  
 5C060  
 Decoder for 80C88 system - 16K RAM and upper 512K EPROM  
 Plus power down circuit  
 Plus wait state circuit  
 IR Version 3.0. Baseline 17x. 9/26/85  
 PART: 5C060  
 INPUTS: A19, A13, A12, PWRDWN, S2MIO, AMWC, MRDC, R2C84CLK, ALE, WAITCS  
 OUTPUTS: RAM0CS, RAM4KCS, RAM8KCS, RAM16KCS, RPRMCS, STOPCLK, R0C88CLK, R2C84RDY  
 NETWORK:  
 RAM0CS = CONF (RAM0CS, VCC)  
 RAM4KCS = CONF (RAM4KCS, VCC)  
 RAM8KCS = CONF (RAM8KCS, VCC)  
 RAM16KCS = CONF (RAM16KCS, VCC)  
 EPROMCS = CONF (EPROMCS, VCC)  
 STOPCLK, STOPCLKF = RORF (PWRDWN, R2C84CLKR, GND, GND, VCC)  
 R0C88CLK, R0C88CLKF = COIF (R0C88CLK, VCC)  
 R2C84RDY = RONF (R2C84RDYD, R0C88CLKR, R2C84RDYD, GND, VCC)  
 PWRDWN = INP (PWRDWN)  
 R2C84CLKR = CLKR (R2C84CLK)  
 R0C88CLK = OR (STOPCLKF, R2C84CLK)  
 R2C84CLK = INP (R2C84CLK)  
 A19 = INP (A19)  
 A13 = INP (A13)  
 A12 = INP (A12)  
 S2MIO = INP (S2MIO)  
 MRDC = INP (MRDC)  
 AMWC = INP (AMWC)  
 R0C88CLKB = CLKB (R0C88CLKF)  
 WAITCS = INP (WAITCS)  
 ALE = INP (ALE)  
 EQUATIONS:  
 RAM0CS =  $((/MRDC*Y0$   
           + /AMWC\*Y0);  
 RAM4KCS =  $((/MRDC*Y1$   
           + /AMWC\*Y1);  
 RAM8KCS =  $((/MRDC*Y2$   
           + /AMWC\*Y2);  
 RAM16KCS =  $((/MRDC*Y3$   
           + /AMWC\*Y3);  
 RPRMCS =  $((/Y7$   
           + /Y6  
           + /Y5  
           + /Y4);  
 Y0 =  $((/A19*/A13*/A12*ENABLE);$   
 Y1 =  $((/A19*/A13*/A12*RNABLE);$   
 Y2 =  $((/A19*/A13*/A12*ENABLE);$   
 Y3 =  $((/A19*/A13*/A12*RNABLE);$   
 Y7 =  $((/A19*/A13*/A12*ENABLE);$   
 Y6 =  $((/A19*/A13*/A12*RNABLE);$   
 Y5 =  $((/A19*/A13*/A12*ENABLE);$   
 Y4 =  $((/A19*/A13*/A12*RNABLE);$   
 ENABLE = /PWRDWN\*S2MIO;  
 R2C84RDYD = -/R2C84RDYD;  
 R2C84RDYD = /WAITCS\*ALE;  
 RND\$

JR Donnell  
 Intel  
 January 31, 1986  
 5C060  
 0  
 5C060

## LEF-3

Decoder for 80C88 system - 16K RAM and upper 512K EPROM  
 Plus power down circuit  
 Plus wait state circuit  
 LB Version 3.0, Baseline 17x, 9/26/85  
 PART:

5C060

INPUTS: A19, A13, A12, PWRDWN, S2MIO, AMWC, MRDC, R2C84CLK, ALE, WAITCS

OUTPUTS: RAMOCS, RAM4KCS, RAM8KCS, RAM16KCS, EPROMCS, STOPCLK, R0C88CLK, R2C84RDY

NETWORK:

A19 = INP(A19)  
 A13 = INP(A13)  
 A12 = INP(A12)  
 PWRDWN = INP(PWRDWN)  
 S2MIO = INP(S2MIO)  
 AMWC = INP(AMWC)  
 MRDC = INP(MRDC)  
 R2C84CLK = INP(R2C84CLK)  
 ALE = INP(ALE)  
 WAITCS = INP(WAITCS)  
 RAMOCS = CONF(RAMOCS, VCC)  
 RAM4KCS = CONF(RAM4KCS, VCC)  
 RAM8KCS = CONF(RAM8KCS, VCC)  
 RAM16KCS = CONF(RAM16KCS, VCC)  
 EPROMCS = CONF(EPROMCS, VCC)  
 ..SG000D = CLKB(R2C84CLK)  
 STOPCLK, STOPCLKF = RORF(PWRDWN, ..SG000D, GND, GND, VCC)  
 R0C88CLK, R0C88CLKF = COIF(R0C88CLK, VCC)  
 ..SG001D = CLKB(R0C88CLKB)  
 R2C84RDY = R0NF(R2C84RDYD, ..SG001D, R2C84RDYC, GND, VCC)

EQUATIONS:

R2C84RDYC = WAITCS' \* ALE;  
 ..SG001D = R0C88CLKF;  
 R2C84RDYD = (WAITCS' \* ALE)';  
 R0C88CLK = (STOPCLKF' \* R2C84CLK)';  
 ..SG000D = R2C84CLK;  
 RPROMCS = (A19 \* PWRDWN' \* S2MIO)';  
 RAM16KCS = MRDC \* AMWC  
 + A19' \* A13 \* A12 \* PWRDWN' \* S2MIO;  
 RAM8KCS = MRDC \* AMWC  
 + A19' \* A13 \* A12' \* PWRDWN' \* S2MIO;  
 RAM4KCS = MRDC \* AMWC  
 + A19' \* A13' \* A12 \* PWRDWN' \* S2MIO;  
 RAMOCS = MRDC \* AMWC  
 + A19' \* A13' \* A12' \* PWRDWN' \* S2MIO;

RND\$

292009-8



RPT-3

Tecio Optimizing Compiler Utilization Report

\*\*\*\* Design implemented successfully

JR Donnell  
 Intel  
 January 31, 1986  
 5C060  
 0  
 5C060  
 Decoder for 80C88 system - 16K RAM and upper 512K EPROM  
 Plus power down circuit  
 Plus wait state circuit  
 LR Version 3.0. Baseline 17x. 9/26/85

5C060

```

GND -- 1 24:- Vcc
PWRDWN -- 2 23:- A19
GND -- 3 22:- STOPCLK
GND -- 4 21:- R2C84RDY
WAITCS -- 5 20:- 80C88CLK
ALB -- 6 19:- RPR0MCS
R2C84CLK -- 7 18:- RAM16KCS
MRDC -- 8 17:- RAM8KCS
AMWC -- 9 16:- RAM4KCS
S2MTO -- 10 15:- RAM0CS
A12 -- 11 14:- A13
GND -- 12 13:- GND
  
```

\*\*INPUTS\*\*

Name	Pin	Resource	MCell #	PTerms	MCells	Feeds:		
						OE	Clear	Clock
PWRDWN	2	INP	-	-	1 4 5 6 7 8	-	-	-
WAITCS	5	TNP	11	0/ 8	2	-	-2	-
ALB	6	INP	12	0/ 8	2	-	2	-
R2C84CLK	7	TNP	13	0/ 8	3	-	-	1
MRDC	8	INP	14	0/ 8	5 6 7 8	-	-	-
AMWC	9	TNP	15	0/ 8	5 6 7 8	-	-	-
S2MTO	10	TNP	16	0/ 8	4 5	-	-	-

						6			
						7			
						8			
A12	11	TNP	-	-		5	-	-	-
						6			
						7			
						8			
A13	14	TNP	-	-		5	-	-	-
						6			
						7			
						8			
A19	23	TNP	-	-		4	-	-	-
						5			
						6			
						7			
						8			

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

Name	Pin	Resource	MCell #	PTerms	MCells	Feeds:		
						OR	Clear	Clock
RAMOCS	15	CONF	8	2/ 8	-	-	-	-
RAM4KCS	16	CONF	7	2/ 8	-	-	-	-
RAM8KCS	17	CONF	6	2/ 8	-	-	-	-
RAM16KCS	18	CONF	5	2/ 8	-	-	-	-
EPROMCS	19	CONF	4	1/ 8	-	-	-	-
ROCRCCLK	20	COTF	3	1/ 8	-	-	-	2
R2CB4RDY	21	RONFA	2	1/ 8	-	-	-	-
STOPCLK	22	RORFA	1	1/ 8	3	-	-	-

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

Name	Pin	Resource	MCell	PTerms
-	1	-	-	-
-	3	-	9	8
-	4	-	10	8
-	13	-	-	-

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

81% Pins  
 87% MacroCells  
 9% Pterms



**APPLICATION  
NOTE**

**AP-276**

June 1986

**Implementing a CMOS Bus  
Arbiter/Controller in the  
5C060 EPLD**

**DANIEL E. SMITH**  
APPLICATIONS ENGINEERING  
INTEL CORPORATION

Order Number: 292012-001

## INTRODUCTION

This application note shows how to implement a CMOS Bus Arbiter/Controller in an Intel 5C060 EPLD (Erasable Programmable Logic Device). The note includes a brief overview of a similar circuit implemented with typical PLA devices, a more detailed discussion of the 5C060 implementation, and a summary.

The bus priority resolution and arbitration scheme selected for the circuit is that used by the industry-standard MULTIBUS I interface. Operation and timing for the MULTIBUS I interface is well understood by most engineers and is described in readily available Intel publications. Thus, a description of the MULTIBUS I interface is not included here. The bus arbiter/controller functions shown here support both serial and parallel priority resolution between bus masters. Timing is equivalent to MULTIBUS I specifications. Electrical specifications for both the PLA and EPLD approaches vary from MULTIBUS I standards. Neither of the two circuits discussed here provide the full current sink capability for all MULTIBUS I signals. Because the EPLD implementation is designed for CMOS systems, however, this requirement is not relevant for the 5C060 implementation.

## PLA APPROACH

The functional equivalent of a MULTIBUS I arbiter/controller can be implemented in two 20-pin PLA-type devices as shown in Figures 1 and 2. (Figure 1 shows the logic for the arbiter device. Figure 2 shows the logic for the controller and the connections to the arbiter.) Figure 3 shows the arbiter list file as an example of PLA-type files. Two different 20-pin PLA devices are required to implement the arbiter and controller functions, a 16R4-type device and a 16L8-type device.

Implementation of logic devices in PLA-type devices, such as those shown here, has proven to be quite beneficial. Development time and cost is much less than for custom silicon device designs. The two PLA-type devices take up less board space than a discrete TTL implementation of the same functions. In addition, the two raw devices can also be used for different functions in other products, thereby reducing inventory costs. As a result of these factors (and others), use of PLA-type devices has grown substantially in recent years.

With the increased density and flexibility of EPLD devices over typical PLA-type devices, even greater space, inventory, and cost savings can be obtained by using EPLDs. The following section shows an implementation of the same arbiter/controller functions in a single 24-pin 5C060 EPLD device.

## 5C060 IMPLEMENTATION

The equivalent functions for both the MULTIBUS I arbiter and controller fit inside a single 5C060 EPLD device. The 5C060 device is available in a 24-pin 0.3" DIP package. Figures 4 and 5 show logic diagrams for the arbiter and controller functions. When compared with the PLA implementation, some differences in the design are immediately apparent. These differences result from the characteristics of the EPLD macrocell or from corrections to the circuit used in Figures 1 and 2.

The major change resulting from the EPLD macrocell structure concerns the EPLD output buffers. Since output buffers from macrocells are non-inverting (PLA-type devices typically contain inverting buffers), signals enter the buffers in the same logic orientation from which they are to appear at the output. The logic for the EPLD (shown in Figures 4 and 5) incorporates this change.

Some errors in the PLA-type implementation have also been corrected in the EPLD design. These changes are as follows:

- The  $M/\overline{IO}$  input to the MRDC/ and MWTC/ gates is inverted.  $M/\overline{IO}$  distinguishes between memory and I/O cycles. The PLA-type implementation does not use this signal properly; the PLA-type controller generates read or write commands to both memory and I/O at the same time, which can result in contention between memory and I/O during bus transfers.
- BPRO/ is gated by BPRN/ in the EPLD design. When using serial priority resolution, this allows the highest priority arbiter to prevent all other masters from controlling the bus. (In the PLA design, BPRO/ is enabled/disabled only by a local request. Higher priority arbiters cannot disable all other arbiters. This can result in contention between bus masters. By gating BPRO/ with BPRN/ in the EPLD design, this source of bus contention is prevented.)

Figure 6 shows the list file for the arbiter/controller device. Figure 7 shows the report file produced by the iPLDS software. This file contains a pinout diagram of the final programmed device and provides a resource usage map for the device.

Most of the input and output signals are self-explanatory to those familiar with Intel processors and the MULTIBUS I interface. The XREQ input is the bus transfer request signal from the address decode logic. The BUSY/ and CBRQ/ outputs are bi-directional, simulated open-collector outputs. These outputs use the iPLDS 5C060 (Combinational-Output I/O-Feedback) primitive in the list file. The BUSY/ signal serves to illustrate this use of EPLD outputs.

A pull-up resistor is used externally (i.e., on the back-plane) to hold **BUSY/** high when no arbiter is in control of the bus. When the arbiter is granted control of the bus, **AEN** is clocked high, which enables the output of the **BUSY/** driver. Since the input to the **BUSY/** driver is low during normal operation (**RESET/** inverted), the enabled driver pulls **BUSY/** low to signal other arbiters that the bus is in use. When the arbiter is finished using the bus, **AEN** goes low to disable the **BUSY/** driver (three-state output). The pull-up resistor pulls **BUSY/** high to signal other arbiters that the bus is free for use if needed.

Note that **BUSY/** is also routed into the bus grant logic as input **BSI**. **BSI** prevents the arbiter from taking control of the bus (and driving **BUSY/** low) when some other arbiter already has control of the bus. Thus only one arbiter may pull **BUSY/** low at any one time.

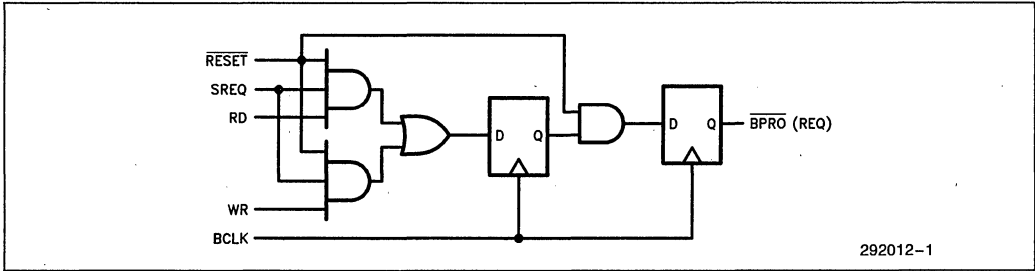
The one difference between standard **MULTIBUS I** logic levels and the **EPLD** implementation described here relates to the **BCLK/** signal. **MULTIBUS I** bus arbitration uses the negative-going edge of **BCLK/** to synchronize events. All 5C060 flip-flops, however, clock on the positive-going edge of **BCLK/**. If all bus masters in the system use the same arbiter implementation, this poses no problem. Otherwise, an external inverter is required for the **BCLK/** input.

## COMPARISON/SUMMARY

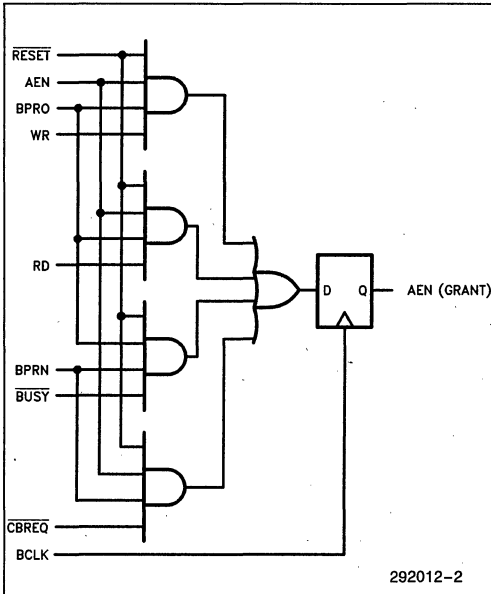
Both the **PLA** and **EPLD** implementations of the bus arbiter/controller result in a lower device count than a discrete logic circuit. Lower device count means less p.c. board space, fewer assembly steps, and fewer device interconnects. Both **PLA** and **EPLD** implementations are quicker and less expensive to develop than a custom gate array or dedicated silicon device.

In contrast to the **PLA** approach, however, the **EPLD** implementation requires only a single device, while the **PLA** approach requires two different devices. Thus the **EPLD** approach results in twice the cost savings (inventory and assembly) and half the programming activity to produce the device. Fewer device interconnects also means greater reliability. In addition, programmed **EPLD** devices can be erased and reprogrammed for a different application if needed, a feature not available with **PLAs**.

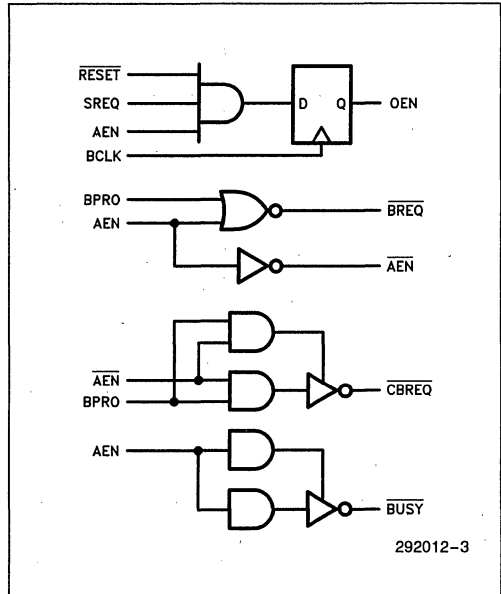
Overall, the greater flexibility, and the incremental design, manufacturing, and cost advantages of **EPLD** devices make them ideal for many applications where **PLA** devices would otherwise be used.



A) Request Synchronizer

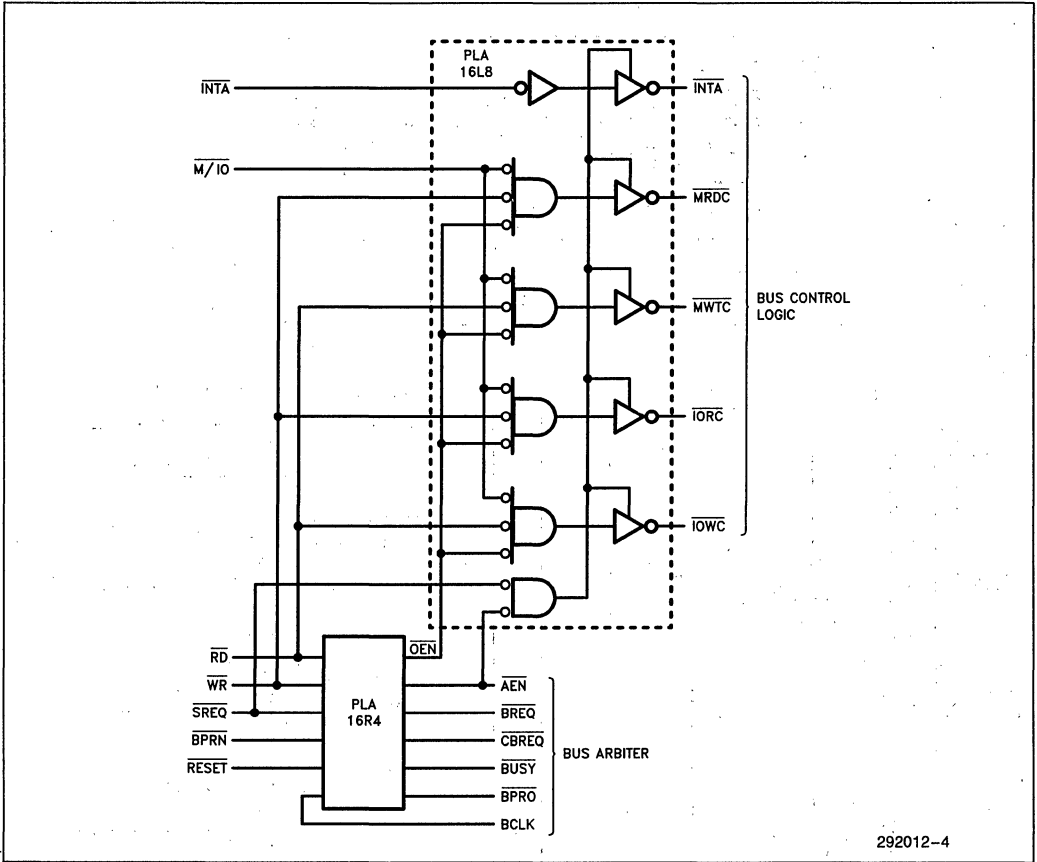


B) Grant/Access Logic



C) Bus Transfer Control

Figure 1. PLA Approach to a Bus Arbiter



292012-4

Figure 2. Bus Controller with Arbiter Connected

```

PLA16R4
ARB001
MULTIBUS I ARBITER
SOME SYSTEM COMPANY
PLA DESIGN FILE
D. E. ENGR. 1/1/85
BCLK /WR /RD /SREQ /RESET /BPRN NC NC NC GND
/E /CBREQ /BUSY /SYNC /BPRO /AEN /OEN /BREQ NC VCC

SYNC := /RESET*SREQ*WR +
       /RESET*SREQ*RD

BPRO := /RESET*SYNC

AEN := /RESET* AEN*BPRO*WR +
       /RESET* AEN*BPRO*RD +
       /RESET*BPRO*BPRN*/BUSY +
       /RESET* AEN*BPRN*/CBREQ

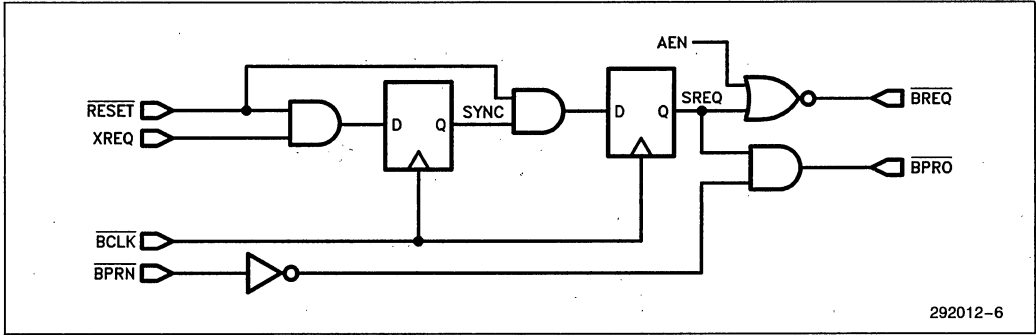
OEN := /RESET*SREQ*AEN

IF(BPRO*/AEN) CBREQ = BPRO*/AEN
IF(AEN) BUSY = AEN

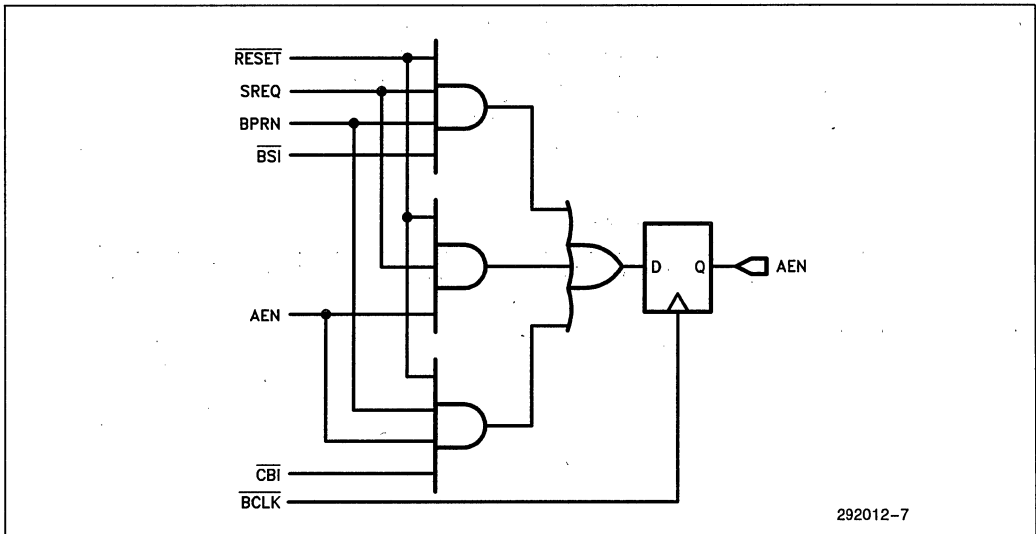
BREQ = BPRO +
      AEN
    
```

292012-5

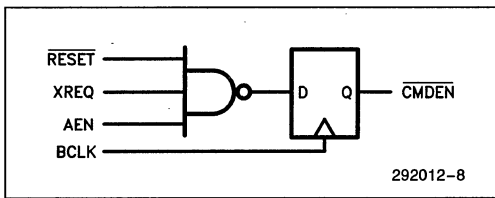
Figure 3. List File for PLA Arbiter



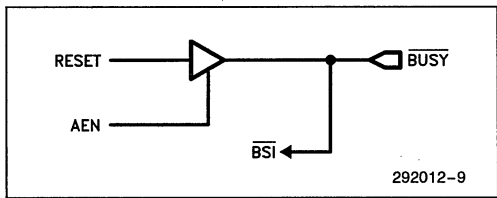
A) Request



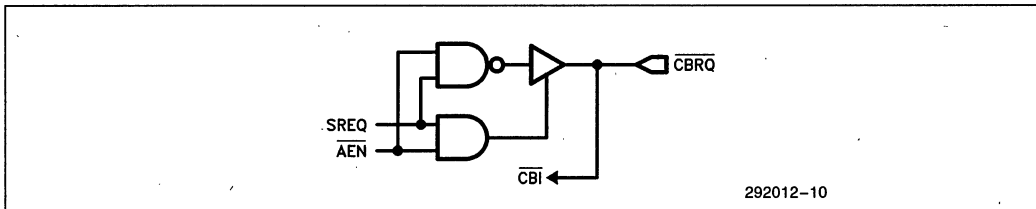
B) Grant



C) Command Enable



D) Busy



E) CBRQ

Figure 4. Logic Diagram of Bus Arbiter Functions



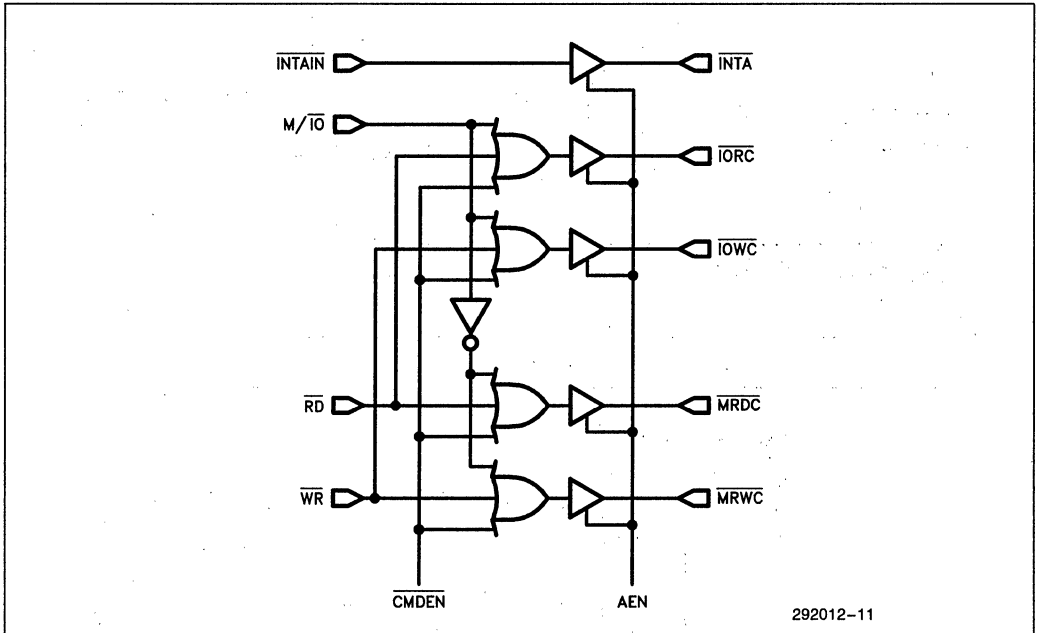


Figure 5. Logic Diagram of Bus Controller Functions

DANIEL E. SMITH  
 INTEL CORPORATION  
 MARCH 27, 1986  
 VERSION 1.1  
 REV. A  
 5C060  
 CMOS BUS ARBITER/CONTROLLER

PART: 5C060  
 INPUTS: BCLK, XREQ, RESET, BPRN, MIO, RD, WR, INTAIN  
 OUTPUTS: BPRO, AEN, BREQ, CBRQ, BUSY, INTA, MRDC, MWTC, IORC, IOWC

NETWORK:

BCLK	= INP (BCLK)	%BUS CLOCK INPUT%
INTAIN	= INP (INTAIN)	%INT. ACK. INPUT%
XREQ	= INP (XREQ)	%SYSTEM REQUEST INPUT%
RESET	= INP (RESET)	%RESET INPUT%
BPRN	= INP (BPRN)	%BUS PRIORITY INPUT%
MIO	= INP (MIO)	%MEMORY/IO INPUT%
RD	= INP (RD)	%READ INPUT%
WR	= INP (WR)	%WRITE INPUT%
BPRO	= CONF (BPROc, VCC)	%BUS PRIORITY OUTPUT%
AEN, AEN	= NORF (AENd, BCLK, GND, GND, VCC)	%ADDRESS ENABLE (GRANT)%
BREQ	= CONF (BREQc, VCC)	%BUS REQUEST%
CBRQ, CBI	= COIF (CBRQc1, CBRQc2)	%CBRQ/ -- SIMULATED O.C.%
BUSY, BSI	= COIF (BUSYc, AEN)	%BUSY/ -- SIMULATED O.C.%
INTA	= CONF (INTAIN, AEN)	%INT. ACK. OUTPUT%
MRDC	= CONF (MRDCc, AEN)	%MEMORY READ COMMAND%
MWTC	= CONF (MWTCc, AEN)	%MEMORY WRITE COMMAND%
IORC	= CONF (IORCc, AEN)	%I/O READ COMMAND%
IOWC	= CONF (IOWCc, AEN)	%I/O WRITE COMMAND%
SREQ	= NORF (SREQd, BCLK, GND, GND)	%VALID BUS REQUEST%
SYNC	= NORF (SYNcd, BCLK, GND, GND)	%SYNCHRONIZED REQUEST%
CMDEN	= NORF (CMDEND, BCLK, GND, GND)	%COMMAND ENABLE%

EQUATIONS:

292012-12

BPROc = (SREQ \* /BPRN);  
 AENd = RESET \* SREQ \* /BPRN \* BSI +  
 RESET \* SREQ \* AEN +  
 RESET \* /BPRN \* AEN \* CBI;  
 BREQc = /(SREQ + AEN);  
 BUSYc = /RESET;  
 CBRQc1 = /(SREQ \* /AEN);  
 CBRQc2 = SREQ \* /AEN;  
 MRDCc = /MIO + RD + CMDEN;  
 MWTCc = /MIO + WR + CMDEN;  
 IORCc = MIO + RD + CMDEN;  
 IOWCc = MIO + WR + CMDEN;  
 SREQd = RESET \* SYNC;  
 SYNcd = RESET \* XREQ;  
 CMDEND = /(RESET \* XREQ \* AEN);

END\$

292012-13

Figure 6. iPLDS Network List File

Logic Optimizing Compiler Utilization Report

\*\*\*\* Design implemented successfully

DANIEL E. SMITH  
 INTEL CORPORATION  
 MARCH 27, 1986  
 VERSION 1.1  
 REV. A  
 5C060  
 CMOS BUS ARBITER/CONTROLLER

5C060

BCLK	-:	1	24:-	Vcc
MIO	-:	2	23:-	XREQ
RESERVED	-:	3	22:-	INTA
RESERVED	-:	4	21:-	IOWC
RESERVED	-:	5	20:-	IORC
AEN	-:	6	19:-	MWTC
BPRO	-:	7	18:-	MRDC
INTAIN	-:	8	17:-	BUSY
WR	-:	9	16:-	CBRQ
RD	-:	10	15:-	BREQ
BPRN	-:	11	14:-	RESET
GND	-:	12	13:-	GND

\*\*INPUTS\*\*

Name	Pin	Resource	MCell #	PTerms	Feeds:			
					MCells	OR	Clear	Clock
BCLK	1	INP	-	-	-	-	-	CLK1
MIO	2	INP	-	-	2	-	-	-
					3			
					4			
					5			
INTAIN	8	INP	14	0/ 8	1	-	-	-
WR	9	INP	15	0/ 8	2	-	-	-
					4			
RD	10	INP	16	0/ 8	3	-	-	-
					5			
BPRN	11	INP	-	-	12	-	-	-
					13			
RESET	14	INP	-	-	6	-	-	-
					9			
					10			
					11			
					12			
XREQ	23	INP	-	-	9	-	-	-
					10			

292012-14

Figure 7. iPLDS Report File

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

Name	Pin	Resource	MCell #	PTerms	MCells	Feeds:		Clear	Clock
						OE			
AEN	6	RORF	12	3/ 8	7	-7	-	-	-
						8	1		
						9	2		
						12	3		
							4		
							6		
BPRO	7	CONF	13	1/ 8	-	-	-	-	
BREQ	15	CONF	8	1/ 8	-	-	-	-	
CBRQ	16	COIF	7	1/ 8	12	-	-	-	
BUSY	17	COIF	6	1/ 8	12	-	-	-	
MRDC	18	CONF	5	1/ 8	-	-	-	-	
MWTC	19	CONF	4	1/ 8	-	-	-	-	
IORC	20	CONF	3	1/ 8	-	-	-	-	
IOWC	21	CONF	2	1/ 8	-	-	-	-	
INTA	22	CONF	1	1/ 8	-	-	-	-	

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

Name	Pin	Resource	MCell #	PTerms	MCells	Feeds:		Clear	Clock
						OE			
	3	NORF	9	1/ 8	2	-	-	-	-
						3			
						4			
						5			
	4	NORF	10	1/ 8	11	-	-	-	
	5	NORF	11	1/ 8	7	7	-	-	
					8				
					12				
					13				

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

Name	Pin	Resource	MCell	PTerms
-	13	-	-	-

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

95% Pins  
 100% MacroCells  
 11% Pterms

Figure 7. iPLDS Report File (Continued)

January 1987

**EPLDs, PLAs and TTL  
Comparing the “Hidden Costs”  
in Production**

**PEDRO VARGAS**  
PROGRAMMABLE LOGIC APPLICATIONS  
INTEL CORPORATION

Order Number: 292030-001

**INTRODUCTION**

When comparing logic alternatives, too often the outcome is dominated by the piece price of the components. A side by side comparison based on component costs only, may give the appearance that EPLDs are cost prohibitive. However, when the overall cost of manufacturing a system is considered, the higher integration of EPLDs proves to be a cost-effective solution.

**OBJECTIVE**

This application note examines the total costs associated with designing, prototyping, and manufacturing a system. Once these costs have been examined, a comparison is made between EPLDs and other logic alternatives. By being aware of these additional costs, the engineer can make a more accurate cost comparison as a design is begun.

**COSTS DEFINED**

Costs can be difficult to pinpoint, let alone measure. However, with a bit of examination, we can break down costs into the following categories;

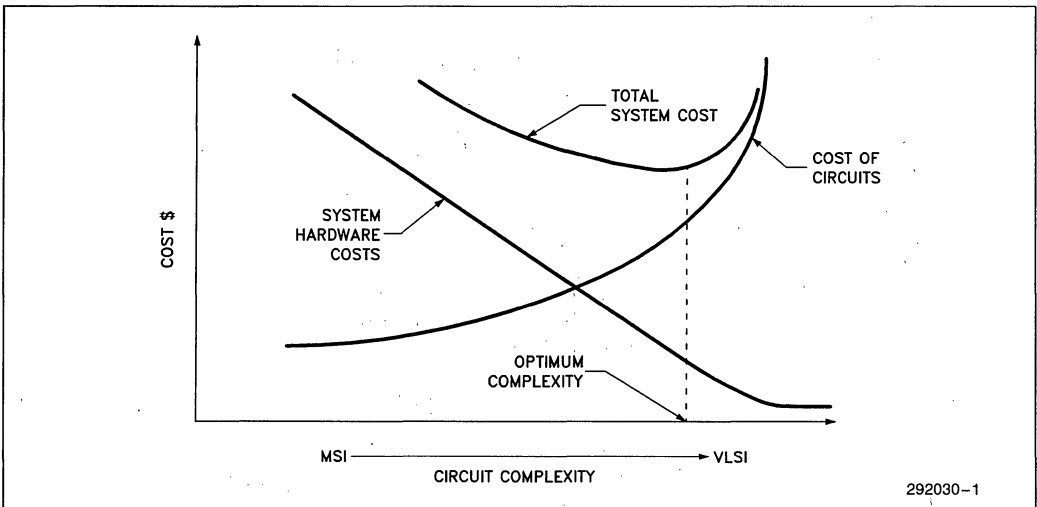
- Design costs — the cost of conceiving a product

- Prototype costs — first implementation of the product idea
- Production costs — volume manufacturing of the product

Usually, the brunt of the cost for the first two categories is dismissed as NRE (non recurring expense). The effect of these costs on the overall project is examined later, let's look at the third category. Production costs, can be further broken down into;

- Component costs — the cost of the parts per board
- Inspection costs — labor costs for receiving the parts
- Inventory costs — the cost for storing, handling and dispensing the parts
- PCB fabrication — the cost for labor and equipment used in building a board
- Integration costs — the cost of harnesses, enclosures, nuts and bolts etc.

It's important to understand how the cost of a product is affected not only by the cost of the ICs used, but also by the other costs listed above. Figure 1 is a graph which shows this relationship.



**Figure 1. Optimizing Circuit Complexity**

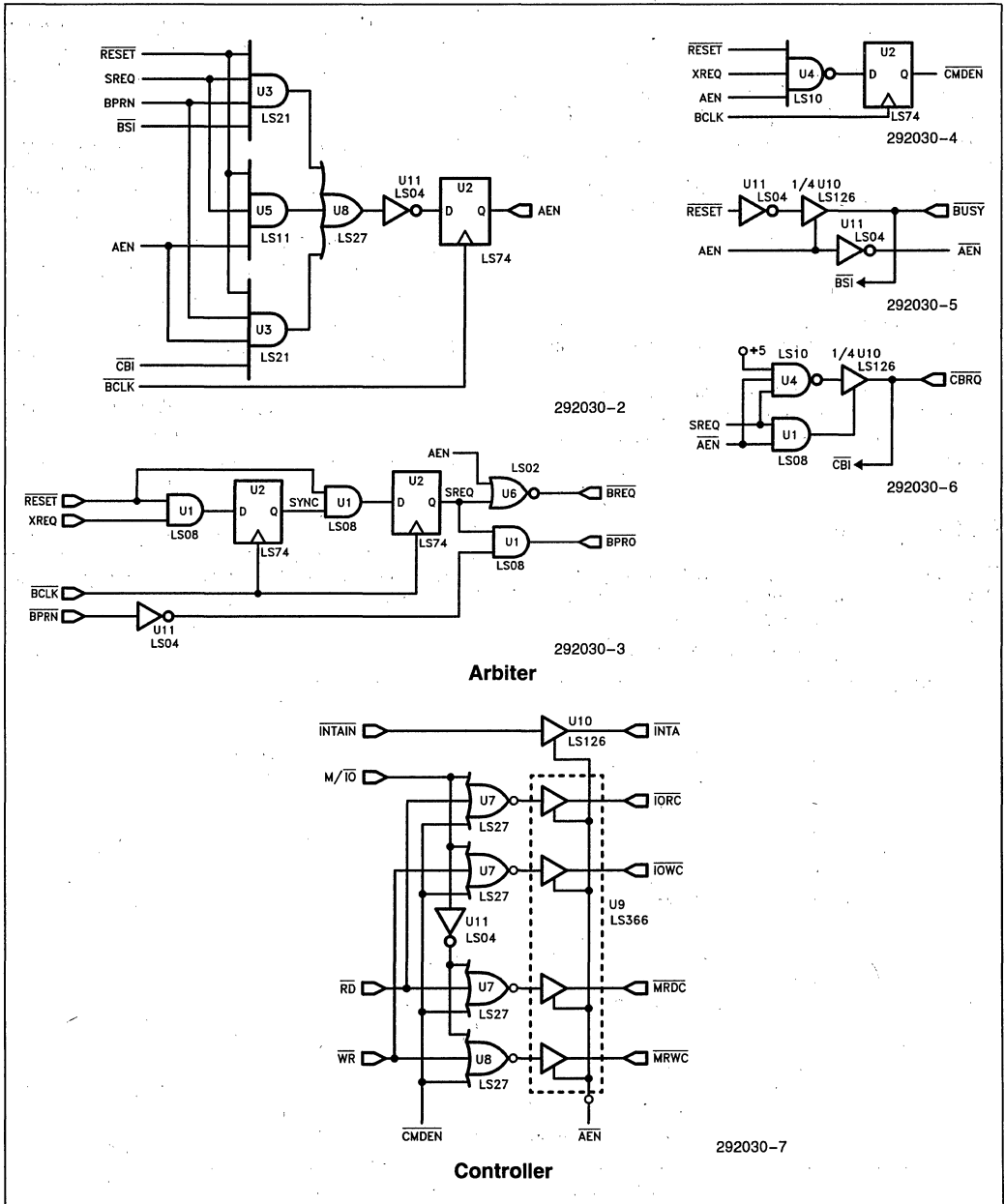


Figure 2. MULTIBUS Arbiter/Controller-TTL Implementation

The graph shows that as the density of the components used in a system progresses from SSI to VLSI, the cost for these devices increases. This isn't surprising, denser chips cost more to make. At the same time, by using denser devices, system hardware cost decreases. This is shown by the center line, which encompasses all the costs listed above. The bathtub curve above these shows the effect that denser ICs has on a system. That is, by using higher integration ICs, more functions are removed from the board. This in turn reduces the cost of the system in labor and parts costs.

A cost-effective product is one that uses the most efficient logic for the application. It's important to note that use of the least expensive component may not translate into system cost savings.

PAL\* is a registered trademark of Monolithic Memories Inc.

### ARBITER CIRCUIT

Let's explore costs in more detail with an example. The example used here is the circuit of Figure 2, a MULTIBUS® I arbiter/controller. The circuit is used by bus masters arbitrating for control of the bus. Our implementation comparison contrasts TTL, PAL\*, and EPLD solutions.

### Implementation Requirements

The TTL implementation is typical of many board level designs in the sense that it relies on inexpensive LSTTL. Figure 2 shows that the implementation is composed of standard logic gates and D-latches. The component list in Table 1 shows the circuit breakdown in more detail. [20]

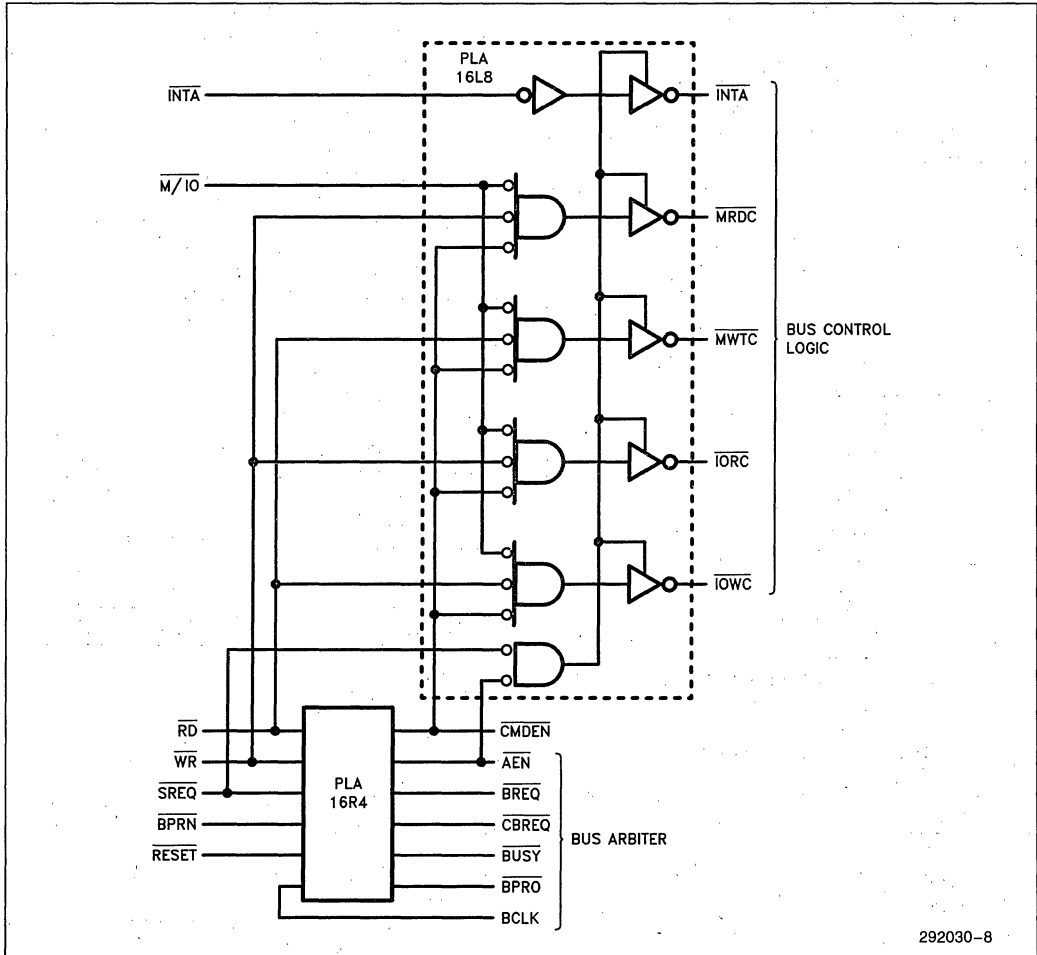


Figure 3. MULTIBUS Arbiter/Controller-PAL Implementation

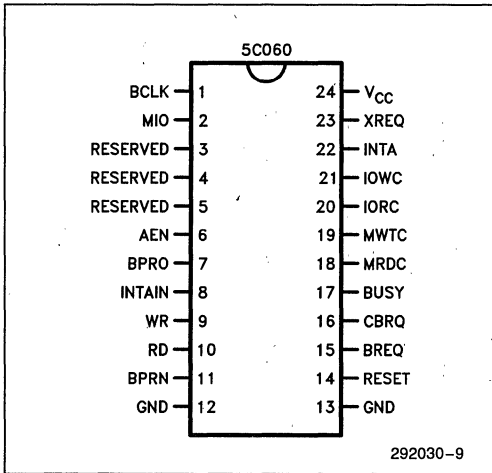


**Table 1. Arbiter/Controller TTL Component List**

IC	Type	DIP	I <sub>CC</sub> (mA)	Area (in <sup>2</sup> )	Cost \$
U1	LS08	14 PIN	8.8	0.21	0.18
U2	LS74	14 PIN	8	0.21	0.24
U3	LS21	14 PIN	4.4	0.21	0.22
U4	LS10	14 PIN	3.3	0.21	0.16
U5	LS11	14 PIN	6.6	0.21	0.22
U6	LS02	14 PIN	5.4	0.21	0.17
U7	LS27	14 PIN	6.8	0.21	0.23
U8	LS27	14 PIN	6.8	0.21	0.23
U9	LS366	16 PIN	21	0.24	0.39
U10	LS126	14 PIN	22	0.21	0.39
U11	LS04	14 PIN	6.6	0.21	0.16

The PAL version of the circuit is shown in Figure 3. Two PALs are used due to the requirement of registered outputs on several of the signals.<sup>[20]</sup>

The complete circuit can also be designed in one 5C060 EPLD (Figure 4).<sup>[18]</sup> Looking at the three figures quickly points out the amount of circuit board space required by each version. The three implementations are compared side by side in Table 2.



**Figure 4. MULTIBUS Arbiter/Controller-EPLD Implementation**

**Table 2. Implementation Results for Arbiter/Controller**

Item	TTL	PLA	EPLD
IC Count	11	2	1
Pin Count	156	40	24
Interconn	36	7	0
Area	2.34	0.6	0.36
I <sub>CC</sub> (mA)	100	240	15
P <sub>wr</sub> (mW)	500	1,200	75

- IC Count — The total chip count
- Pin Count — The total number of IC pins
- Interconnections — The traces required to connect logic gates together
- Area (inches-square)— The sum of the area of all ICs
- I<sub>CC</sub> (mA) — The current consumed while active
- P<sub>wr</sub> (mW) — Total power consumption at 5 VDC.

**Production Costs**

Earlier, we noted that production costs consist of many variables. Usually, these variables are lumped together under the term "hidden cost". Although hidden costs are kept in mind by engineers, lack of tangible figures usually precludes their use in detailed cost breakdowns. For this reason, several manufacturers and consulting firms have come up with typical costs per IC and per pin.

For example, SOURCE III (San Jose, CA) reports in one of their studies that the manufacturing cost of a system translates to about 0.35 cents per IC pin. ICE Corporation (Scottsdale, AZ) and EDN magazine concur that the inserted cost of an IC is about \$2 dollars. DATAQUEST also published a cost of about \$2 to \$4 per IC. While the data seems to be consistent, most engineers want to see for themselves how figures like these might be arrived at. The next sections provide insight into this process.

**COMPONENTS**

The cost of the component is the easiest value to obtain. A quick call to a distributor or (at worst) a scan through the back of BYTE magazine (for TTL) gives us this cost. Table 3 shows the breakdown of component costs for each version of our MULTIBUS I circuit.

**Table 3. Average Component Costs**

Package	TTL	PLA	EPLD
DIP14	\$0.25		
DIP16	\$0.35		
DIP20	\$0.55	\$1.50	
DIP24		\$2.90	\$6.00

The price of TTL has changed very little for the last few years<sup>[24]</sup> while EPLDs are dropping in price tremendously. PALs have also leveled off in pricing. Why? Figure 5 shows the life cycle curve of IC products used by the semiconductor industry. From the curve we see that TTL is in the stable range and prices are not likely to drop much more. PALs are also maturing and approaching a stable pricing range. EPLDs however, are in a growth area and historically this is

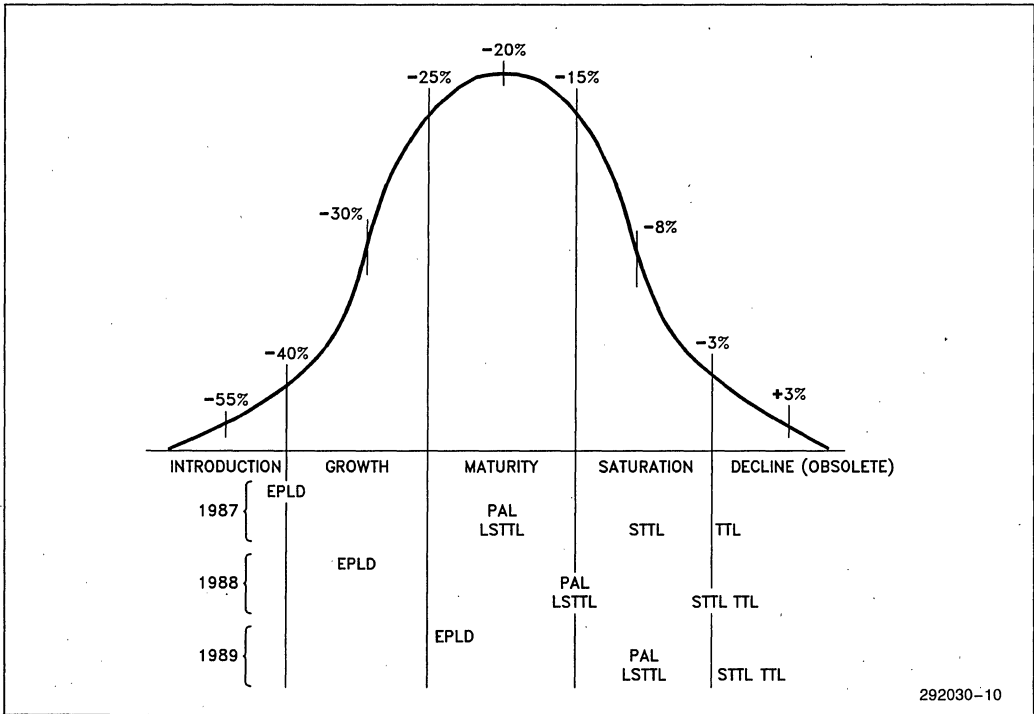


Figure 5. Typical Price Changes Through Semiconductor Product Life Cycle

where the heaviest pricing pressure is. This means that while EPLDs might be expensive (per part) right now, it's not out of the question to expect a 30% per year price reduction as the process is honed and perfected. In other words, it's also important to consider the price of a component at the projected production date, not just at design time.

Life cycle position is also important in understanding the gate cost that is associated with programmable logic devices like PALs and EPLDs. This relationship is shown in Figure 6. The curves translate our observation that newer devices have steeper price cuts during their introduction phase. The PAL curve shows that the cost per gate is leveling off due to the maturity of the device. In contrast, the EPLD is in the growth region, and based on the traditional price reductions, shows a cost per gate that intersects and bypasses the PAL curve.

**INCOMING INSPECTION**

For most companies, incoming inspection is more than taking the parts and putting them on the shelf. Most have visual checking as well as some form of IC testing. The variables here are, what amount of human intervention is needed, are automatic handlers needed, are "go/no go" tests or "binning" done automatically? The typical scenario means that components are graded and tested individually, and then placed into one of several bins or kitted. Because the operators handle a large variety of pinned devices (resistors, capacitors, ICs), the cost can be distributed on a per pin basis. Many companies use a penny per pin for this cost. [16]

Inspection cost = \$0.01 per pin

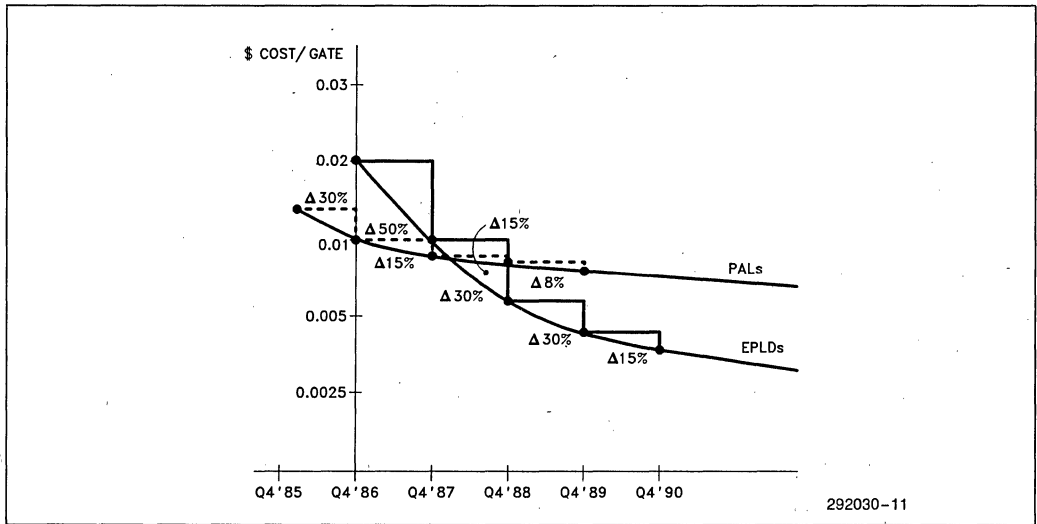


Figure 6. Projected Cost Per Gate

**INVENTORY**

While most engineers agree that reducing parts count on their board makes the cost of inventory less, they usually attribute this to the reduction in component costs alone. In reality, the overhead of carrying inventory is made up of the following factors; [21]

- Cost of the component
- Cost of storage
- Maintenance costs
- Data processing
- Usage
- Taxes insurance and interest
- Turnover rate

The American Production and Inventory Control Society (APICS) reports that since 1973 the median cost of carrying inventory has been about 25% of total production costs. They also note that the largest contributing factors are the cost of materials handling storage, and data processing. For simplicity, let's limit our inventory cost to these items.

$$\text{Inventory cost} = \text{storage} + \text{maintenance} + \text{processing}$$

Depending on the locale of a company, the cost of storage can vary greatly. However, this cost is charged on a square foot per year basis. Lets assume a conservative figure of \$20 dollars and distribute this among the ICs in our example circuit.

$$\text{storage} = [\text{Total IC area (sq. ft.)} \times \$20] / \text{IC count}$$

Maintenance refers to the cost of handling, counting, marking, and auditing each IC. Each production manager has their own way of keeping tabs on this. One way is to charge on a per part basis. A review from several production oriented journals cites \$0.3 cents as the typical handling charge for 16 pin devices. [23]

$$\text{Maintenance} = \$0.03 \text{ per 16 pin part.}$$

Processing [21] usually entails a parts log that tracks each part by manufacturer, cost, second source etc. Also, monthly shortage reports are quite common as are quarterly orders and audits. Limiting this cost to paper only, at one sheet of paper per week, per year, at a cost of a penny per part type;

$$\text{Processing} = \$0.52 \text{ per part type per year}$$

**PCB FABRICATION**

The cost of manufacturing (cutting, etching, drilling) a circuit board seems to vary around two pricing methods. Some fab houses charge on a square inch basis. Others base their price on a gut feeling based on previous jobs. The square inch method is the most common.

Items of interest in evaluating PCB costs are, number of ICs, number of traces and vias, and in general, the complexity of the board. Traces that are smaller than 10 mils require extra care in etching. Depending on complexity, and additional charge might be added to the area cost. This charge covers material loss in case of low etch yields. Yield is directly dependent on the number of ICs on a board. In other words, more ICs mean more holes, tighter traces, and a greater chance of losing some boards in their processing. The average going

rate is \$0.20 cents per inch for double-sided boards. The price increases by about 40% for every two layers. This extra charge, however is too subjective to consider in our comparison.

$$\text{PCB Fab} = [\$0.20 \times \text{total IC area (sq. inch)}] / \text{IC count}$$

**Traces**

There is a real cost involved with traces, which doesn't surface until later in the production cycle or on a later board revision. A technical paper presented at the 1984 international Test Conference<sup>[1]</sup> estimates that the cost of a trace on a board is ten to thirty times that of one made in silicon. The cost of traces is taken up by:

- Increased drilling (more traces = more vias = more holes)
- Lower PCB yield (smaller mill lines drop the board yield)
- Increased risk of trace to trace shorts (lower reliability)
- More expensive artwork mods (it costs more to move traces around on a board)
- More expensive PCB mods (cost of cuts, jumpers, and rework)

In our circuit example, an extra trace is that which is unnecessary in contrasting implementations. For example, referring to Figure 2, of all the traces required to connect/RESET in the TTL implementation, only one will be required for the EPLD and PAL circuit (the input); the others won't be needed.

For our comparison, let's take the median value of twenty as our multiplying factor. Since a silicon trace costs an order of magnitude less than an EPLD gate (\$0.01), the resulting cost of a PCB trace is;

$$(\$0.01/10) \times 20 = \$0.02 \text{ cents per trace}$$

$$\text{Trace cost} = [\text{total trace count} \times \$0.02] / \text{IC count}$$

**ASSEMBLY**

The cost of assembling a board is largely dependent on labor charges and capital. Assembly consists of lead forming, component insertion, and soldering. The labor charge is hourly and varies between domestic and off-shore assembly houses. While machines can certainly do lead cutting, crimping, and insertion, human intervention is still an expensive presence. Assembly costs can be charged on a per board or per chip basis. The latter is more appropriate for our comparison. The average charge (domestically) is about \$0.10 per IC.

$$\text{Assembly} = \$0.10 \text{ per 16 pin part}$$

One important result of using high integration parts like EPLDs is that the assembly procedures (manual or automatic) go smoother. This is due to fewer parts being handled, and less overheating of the equipment. Overall, the industry reports less insertion faults (parts stuffed wrong) as denser ICs are used and as insertion equipment matures with them.

**TEST**

Test strategies can vary, but the typical test flow for a board<sup>[3]</sup> is shown in Figure 7. The process is basically taking a board through increasing complexity levels of testing. For example, ATE might be a bed of nails fixture that catches 60 percent of the faults. Test bed is usually a backplane with all boards known good except for the one under test. System test is the final integration of all the boards that were tested individually.

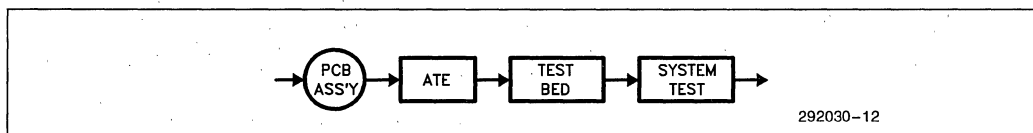


Figure 7. Typical Test Flow

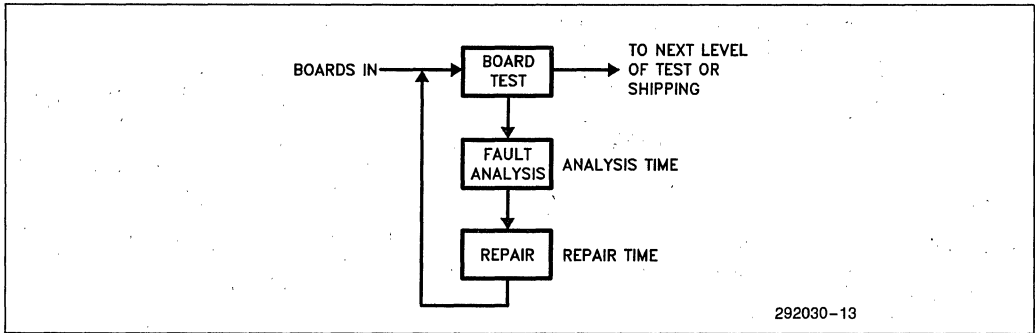


Figure 8. Typical Test and Repair Loop

292030-13

Errors can occur at any step of the test flow; each time this happens, a test loop is initiated. This loop is depicted in Figure 8. The cost for testing a device depends on the cost of the equipment, depreciation, the labor rate, and other factors that are company dependent. There are several ways to reduce test costs, but the best way is to reduce the probability of errors occurring. There is no question that as the number of ICs increases, so does the probability of error.

With all things considered, the industry reports a nominal test cost of about \$0.15 per IC.<sup>[27][28]</sup>

$$\text{Test cost} = \$0.15 \text{ per } 16 \text{ pin IC}$$

**REWORK**

The cost of rework is best understood by considering the cause of errors in more detail. Errors are typically caused by poor board quality, inadequate solder process, tolerance of insertion, and of course, bad chips. Table 4 shows the average board fault spectrum. The figures are a conclusion reached by EVALUATION ENGINEERING magazine<sup>[10]</sup> as to what the industry is currently seeing. The table shows that the majority of board errors is due to solder shorts. These errors are the result of traces or IC holes being too close, which is what happens on densely populated boards.

Table 4. Average Board Fault Spectrum

Tolerance	20%
Shorts	40%
Insertion	30%
Bad Parts	10%

Of all the material costs associated with rework, the main cost is the time spent on a repair. Considering that it takes approximately two minutes to desolder,

insert, resolder, and clean a component pin<sup>[9]</sup>, one can see that more ICs on a board directly affect cost. Repair times also increase dramatically on multi-layer boards that might have been doubled sided if denser logic was used.

For our comparison, let's assume that our test equipment is 95% efficient in finding solder faults on the first pass (no loop). This leaves 5% of the faults that go undetected and eventually must be found and repaired. The estimated cost per pin based on a \$6.00 hourly wage and the two minute repair time is approximately \$0.02 cents.

$$\text{Rework} = [\$0.02 \times \text{total pin count}] / \text{IC count}$$

It is important to note that the probability of errors is based on a Poisson distribution<sup>[8]</sup> that increases exponentially with the number of pins and components. This distribution is used in wave solder processing to correct for solder errors. Mathematically this is expressed as:

$$P = \frac{e^{-np}(np)^x}{X!}$$

- where; P = The probability that a defect will occur
- n = The number of components
- p = The fraction defective
- x = The actual number of defects

This means that the TTL and PAL version of the arbiter have a higher probability of error than the EPLD version. However, to make our comparison easier, let's simplify this to more of a linear relation. For each implementation, the rework cost per IC is calculated by;

$$\text{Rework cost} = [(\text{total pin count}) \times (5\%) \times (\$0.02 \text{ cents})] / \text{IC count}$$

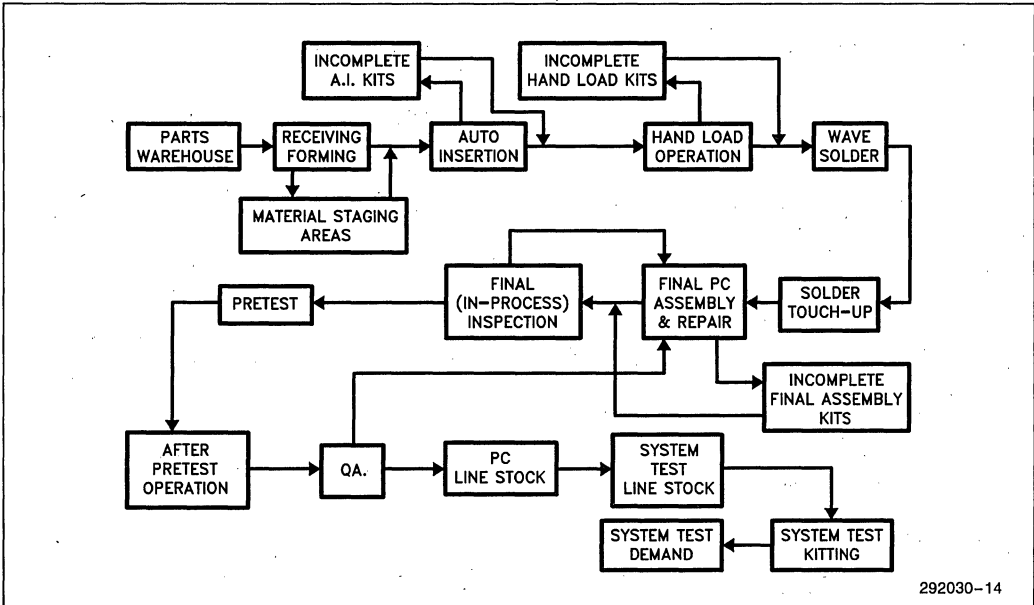


Figure 9. Example of a Production Line

292030-14

**QUALITY CONTROL**

In most production operations, boards go through several steps of quality inspection. The bare board might be inspected after preliminary tests and after system tests. Although 100% inspection should theoretically eliminate all errors, in real life this rarely happens. The main reason for this is the complexity of the production and rework loops as shown in Figure 9.

Quality control's purpose is to remove defective products and either junk them or rework them, neither of which is cost effective. The best approach is to design the quality in, not fix it in. One way to design in quality is by reducing the possibility of errors and increasing the reliability of a product. This is one of the primary advantages of dense logic (like EPLDs and PALs) over TTL.

A survey conducted by *CIRCUITS MANUFACTURING* magazine<sup>[8]</sup> yielded the cost of \$10 to \$50 dollars to inspect, find, and repair a defect on a board. They summarized that the actual cost of inspection is about \$0.004 for each hole on a board. With this in mind, let us assume a 100% inspection of our arbiter circuit for each implementation. This means that each pin (and every trace via) will have to be looked at. The calculation for this is;

$$QC\ cost = (total\ pin\ count \times \$0.004) / IC\ count$$

**POWER SUPPLY**

Price for 5V, single output, switching power supplies as advertised by several vendors is \$1.00 per watt. The calculation for determining power supply costs in our comparison is:

$$Power\ cost = [(5VDC \times I_{CC} (mA)) \times \$1.00\ per\ watt] / IC\ count$$

**Additional Costs**

In addition to the more obvious costs, there are several other items that contribute to the "hidden cost" of a system.

**PROGRAMMING LOSS**

Because PALs are a one time programmable type of device, full testing can't be done on them without destroying the user's fuses. For this reason PALs have a published programming loss of 2%<sup>[20]</sup>. The cost for this is:

$$Programming\ loss = (PAL\ IC\ count \times 0.02) \times PAL\ cost\ per\ IC$$

EPLDs, because they are based on EPROM cells, can be programmed for different patterns, fully tested before customer delivery, and then erased. The result is a near 100% percent programming yield<sup>[22]</sup>.

**PROGRAMMING FEE**

Programming fee is the cost of programming a device. While many companies have in-house programmers, it is quite common for programming to be done by the distributor. In some cases, and at low volumes, the programming may be done free of charge. However, at larger volumes a programming charge is not uncommon. The charge varies with volume, programmer availability and in general, your state of affairs with the distributor. The cost for programming EPLDs and PALs is the same per device and averages about \$0.25 cents.

$$\text{Programming fee} = \$0.25 \text{ cents}$$

**SAFETY STOCK**

Although this particular item was not mentioned in the inventory section, it plays a very important role in the production world. Safety stock<sup>[21]</sup> is extra ICs ordered to cover for unexpected events. Unexpected here might be a large unforeseen customer order or simply a bad batch of parts.

While industry seems to strive for the optimum JIT (just in time) production<sup>[14][16]</sup>, which stresses minimal inventory until needed, it's not unusual for production managers to carry a five to ten percent inventory buffer depending on the cost of the part. In most cases, the larger expensive parts like microprocessors, peripheral controllers, and other LSI devices are safety stocked in smaller quantities.

Let's assume that the safety stock is to be a maximum of 10%. Five percent might be used to cover for the unexpected occurrences, and five for WIP (work in process) modifications. Since all parts have the same probability of unexpected events we can assign that percentage equally. Justifying the second 5% depends on the IC technology itself. For instance, WIP modifications usually require cuts and jumpers on TTL, therefore it's unnecessary to order the additional 5%. In process modifications to an EPLD are done simply by reprogramming it, here again there is no need for the additional 5%. PALs however cannot be cut and jumpered (internally) nor can they be reprogrammed. Also, there is the possibility that "on the shelf" PALs will be programmed in advance, therefore a WIP mod that impacts their function means that those parts must be obsolete (junked). In this case, an additional 5% is justifiable.

Let us assume that the production manager reduces safety stock by a moderate amount, let's say 3%. In a case like this, usually the larger more expensive parts are curtailed first. Since EPLDs provide good coverage for work in progress and because they are more expensive by comparison, we can reduce the total safety stock to 2% and not compromise our safety margin. Because TTL is inexpensive it tends to suffer more of the "gunshot" approach in testing<sup>[7]</sup>. This means that the usage rate is greater because production technicians tend to replace TTL parts with more liberty. For this reason let's leave the TTL safety stock as it stands. PALs could be reduced, but faced with the fact that the programming yield is 2% and that internal modifications can't be made, the production manager might decide not to change the safety stock for PALs. These results are shown in Table 5.

**Table 5. Safety Stock**

	TTL	PAL	EPLD
Unexpected Events	5%	5%	2%
WIP MODS	0	5%	0
Total	5%	10%	2%

The safety stock calculation for each implementation is:

$$\text{Safety stock} = (\% \text{ of stock} \times \text{IC type} \times \text{IC type cost}) / \text{IC count}$$

**DE-COUPLING CAPACITORS**

While adding caps solves many problems due to system noise, it also increases the cost of PCB layout, PCB fab, and adds an additional burden on all of our other costs. For a TTL system, a good de-coupling rule of thumb is to use one 0.01  $\mu\text{f}$  per each synchronous driven gate and at least 0.1  $\mu\text{f}$  per 20 gates regardless of synchronicity. Engineers recognize the need for decoupling and usually take it a step further by using one capacitor per IC. Most boards reflect this practice, which, in itself is very good. However, the addition of all these caps is definitely measurable, in both component and systems cost.

The average cost of a ceramic capacitor in moderate quantities is about half a cent. For our comparison we will follow the accepted practice and de-couple each TTL, PAL, and EPLD device. Our capacitor cost is then:

$$\text{De-coupling cost} = \$0.005 \times \text{IC count}$$

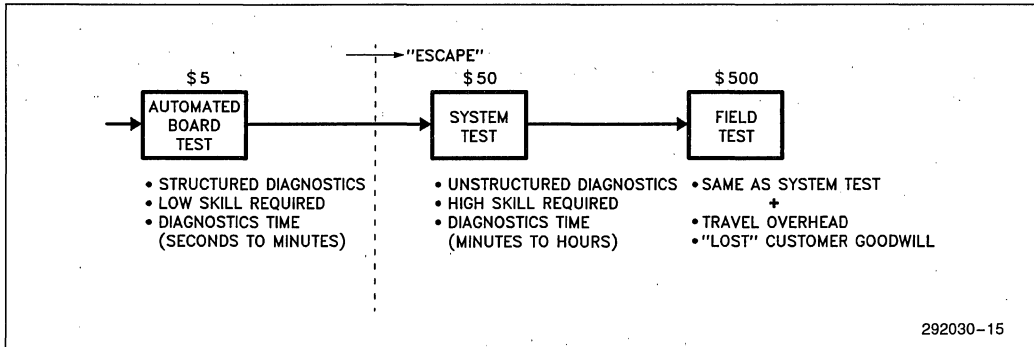


Figure 10. Escape Costs

**Other Costs To Consider**

Eventually, some place toward the end of a production line, a board becomes part of a system. At this point it is housed in an enclosure and all the necessary cabling is done. Even here, however, the impact of using a particular IC technology can still be felt.

**DEFECT ESCAPES**

One very significant item that the test community acknowledges is the cost of "escapes"[4]. "Escape" is defined as a fault that goes through the early stages of board test undetected. Figure 10 shows the escape relationship. An industry rule of thumb states that the cost to detect a fault increases by an order of magnitude at each stage. This means that if it costs \$5 to find a fault at the board test level, that same fault might cost \$50 at the system level and \$500 at the field level. An important relationship to remember, is that the number of faults per board increases logarithmically, as the number of components on the board increases[6]. The cost of an "escape" is difficult to quantify, but generally, a board with a higher component count has a greater cost[2][8].

**CABLES/WIRING HARNESS**

When the number of components or the power requirements of a system are reduced, a reduction in cables and wiring is usually expected. The cost savings here is either in the elimination of cables (because more functions are condensed into an IC) or the reduction of cable gauge or length (because less power is required, in the case of EPLDs). Also, fewer cables means fewer cable ties, connector pins, and mounting hardware. While this is a subjective figure, let's assume that the distributed cost of system cables is \$0.25 per IC.

$$\text{Cable cost} = \$0.25 \times \text{IC count}$$

**ENCLOSURE**

Certain applications require reduced packaging or enclosure size. In industrial control for example, each line might require a complete system to monitor it's operation. In a case like this, a large bulky box full of boards might not be appropriate. A good example of the benefits that high integration logic provide enclosures, is the third market versions of the popular PC. Many of these companies have fully compatible versions that fit on a single board. EPLDs and PALs are capable of providing a cost savings in this respect. However, while PALs approach the density requirements, their large power needs render them counterproductive to the low power specs of small systems. TTL is just not as effective as either PALs or EPLDs.

For our comparison let us assume the cost of enclosure per chip is \$0.75. The calculation is:

$$\text{Enclosure cost} = \$0.75 \times \text{IC count}$$

Table 6 shows the cable and enclosure costs for the MULTIBUS I circuit. Although the results are based on assumed values, we can see that a larger IC count influences the burdened cost of the system. Our final comparison will not use these figures, but they should be considered.

**Table 6. Other Production Costs for Multibus I Circuit**

	TTL	PLA	EPLD
Wiring/harness	\$2.750	\$0.500	\$0.250
Enclosure	\$8.250	\$1.500	\$0.750



**Arbiter Circuit Conclusion**

A compilation of the cost variables for our comparison is shown in Table 7a and 7b. Because the cost may differ for each company, the comparison calculations

were done on a Lotus 1-2-3 worksheet that the individual engineer can modify with their specific values. The worksheet is available, and can be downloaded from the Intel EPLD bulletin board. Table 8 shows our calculation results for three years of production.

Inventory:		Costs	
Incoming insp. (\$/pin)		\$0.010	
Storage (\$/sq.ft./yr)		\$20.000	
Maintenance (\$/part)		\$0.030	
Processing (\$/part type/yr)		\$0.520	
Safety stock (%)		2%	
Manufacturing:		Costs	
PCB fab. (\$/sq.in.)		\$0.200	
Assembly (\$/part)		\$0.100	
Test (\$/part)		\$0.150	
Rework (\$/pin)		\$0.020	
QC (\$/pin)		\$0.004	
Power (\$/watt)		\$1.000	
Interconn		\$0.020	
Program (\$/part)		\$0.250	
Caps. (each)		\$0.005	
(a)			
Integrated Circuits			
Component Count:			
Package	TTL	PLA	EPLD
DIP14	10		
DIP16	1		
DIP20	0	2	
DIP24			1
		ICs	Types
		TTL	10
		PLA	2
		EPLD	1
Circuit Requirements:		I <sub>CC</sub> (max)	Interconnects
TTL circuit (total mA).		100	36
PLA circuit (total mA).		240	7
EPLD circuit (total mA).		15	0
(b)			

**Tables 7a and b. Multibus Arbiter/Controller Cost Variables**

**Table 8. MULTIBUS I Arbiter/Controller Production Costs**

<b>AVERAGE COMPONENT COST</b>									
	<b>Year 1</b>			<b>Year 2</b>			<b>Year 3</b>		
<b>Package</b>	<b>TTL</b>	<b>PLA</b>	<b>EPLD</b>	<b>TTL</b>	<b>PLA</b>	<b>EPLD</b>	<b>TTL</b>	<b>PLA</b>	<b>EPLD</b>
DIP14	\$0.25			\$0.20			\$0.19		
DIP16	\$0.35			\$0.30			\$0.27		
DIP20	\$0.55	\$2.00		\$0.38	\$1.70		\$0.35	\$1.56	
DIP24			\$6.00			\$4.20			\$2.90
<b>PRODUCTION COSTS</b>									
	<b>Year 1</b>			<b>Year 2</b>			<b>Year 3</b>		
<b>Item (costs per part)</b>	<b>TTL</b>	<b>PLA</b>	<b>EPLD</b>	<b>TTL</b>	<b>PLA</b>	<b>EPLD</b>	<b>TTL</b>	<b>PLA</b>	<b>EPLD</b>
Components	\$0.259	\$2.000	\$6.000	\$0.209	\$1.700	\$4.200	\$0.197	\$1.560	\$2.900
Incoming Insp.	\$0.142	\$0.200	\$0.240	\$0.142	\$0.200	\$0.240	\$0.142	\$0.200	\$0.240
Inventory									
Maintenance	\$0.027	\$0.038	\$0.045	\$0.027	\$0.038	\$0.045	\$0.027	\$0.038	\$0.045
Storage	\$0.030	\$0.042	\$0.050	\$0.030	\$0.042	\$0.050	\$0.030	\$0.042	\$0.050
Processing	\$0.473	\$0.520	\$0.520	\$0.473	\$0.520	\$0.520	\$0.473	\$0.520	\$0.520
Printed Circuit Board									
Fabrication	\$0.043	\$0.060	\$0.072	\$0.043	\$0.060	\$0.072	\$0.043	\$0.060	\$0.072
Trace costs	\$0.065	\$0.070	\$0.000	\$0.065	\$0.070	\$0.000	\$0.065	\$0.070	\$0.000
Assembly	\$0.089	\$0.125	\$0.150	\$0.089	\$0.125	\$0.150	\$0.089	\$0.125	\$0.150
Board test	\$0.150	\$0.150	\$0.150	\$0.150	\$0.150	\$0.150	\$0.150	\$0.150	\$0.150
Rework	\$0.014	\$0.020	\$0.024	\$0.014	\$0.020	\$0.024	\$0.014	\$0.020	\$0.024
QC	\$0.057	\$0.080	\$0.096	\$0.057	\$0.080	\$0.096	\$0.057	\$0.080	\$0.096
Power Supply	\$0.045	\$0.600	\$0.075	\$0.045	\$0.600	\$0.075	\$0.045	\$0.600	\$0.075
Total Cost/Part	\$1.393	\$3.904	\$7.422	\$1.343	\$3.604	\$5.622	\$1.331	\$3.464	\$4.322
Total Cost/System	\$15.321	\$7.808	\$7.422	\$14.771	\$7.208	\$5.622	\$14.641	\$6.928	\$4.322
Additional Costs/System									
Programming loss	\$0.000	\$0.080	\$0.000	\$0.000	\$0.068	\$0.000	\$0.000	\$0.062	\$0.000
Safety stock	\$0.143	\$0.400	\$0.120	\$0.115	\$0.340	\$0.084	\$0.109	\$0.312	\$0.058
Programming fee	\$0.000	\$0.500	\$0.250	\$0.000	\$0.500	\$0.250	\$0.000	\$0.500	\$0.250
De-coupling caps	\$0.055	\$0.010	\$0.005	\$0.055	\$0.010	\$0.005	\$0.055	\$0.010	\$0.005
True mfg. cost/system	\$15.518	\$8.798	\$7.797	\$14.941	\$8.126	\$5.961	\$14.804	\$7.813	\$4.635

The comparison in component costs shows that the EPLD costs more than either a TTL or PAL IC. As costs are added, the figures for TTL and PALs begin to approach the cost of an EPLD. These are shown on the line labeled "Total cost/part".

The "Total cost/system" line shows the actual cost when all the ICs are considered. For the first year, the TTL version is the more expensive implementation, and the EPLD numbers look very favorable.

The "True mfg. cost/system" line results after additional costs are figured in. Here we see that the first year, the EPLD version already provides a \$1 savings over the PAL version, and that the cost of the TTL implementation is very high. Also, the inserted cost per IC at this point is, \$1.15 for TTL, \$2.40 for PAL and \$1.80 for the EPLD. This is in line with the inserted costs that we mentioned earlier.

The production costs for two additional years shows that the decreasing price of EPLDs (based on the curve of Figure 5) will continue to provide costs savings as production ramps up in quantities.

In terms of functional benefits, the EPLD implementation is the most beneficial because;

- The chip count has gone down, one EPLD has replaced 11 TTL ICs in one implementation, and 2 PALs in the other, reducing the cost and time of:
  - board layout
  - board fab
  - assembly
  - rework
- The reliability of the board has increased. Fewer components translates into less probability of error.
- Modifications are easier to make. Instead of cuts and jumpers (for TTL), or throwing away a PAL, a change is re-programmed.
- The need for de-coupling caps is reduced. All those individual ICs are eliminated and in some cases the distributed capacitance of the board may be enough de-coupling.
- Power supply requirements are small. The active current requirements are much smaller with EPLDs. This in turn reduces the need for large power supplies and fans.
- Cable requirements and enclosure benefits have been improved. Since EPLDs provide better integration over TTL and PALs, the size of the system will be smaller. This translates into fewer boards and cables.
- Inventory is reduced. One EPLD replaces many TTL devices. Also, "on the shelf" programmed EPLDs can be reused in a pinch, PALs can't.

Less expense and probability of "escapes". The time and cost of finding and fixing escape problems is re-

duced to one reprogrammable IC. In the field, this translates into less "down time" for the customer and a higher level of customer "goodwill" for the OEM.

Allows capability for customized hardware. Specific customer requirements can be implemented. Also, DIP switches and configuration jumpers may not be necessary in many cases, since configurations can be programmed into the EPLD.

## Development Costs

As mentioned earlier, the costs of development are usually dismissed as NRE. One reason for this is the difficulty in pegging down these costs. However, while money might be expendable at this stage, time is usually critical. Time saved at the front end can make a difference in beating the competition to market. The following topics are presented for consideration. No costs are assigned to them.

### RESEARCH

The amount of time spent researching components, component sources, and technical data can be very large. Designs done with a large IC count require more research and analysis time. Higher integration devices require learning curve time, but, in the long run this tends to reduce research time, especially in future designs.

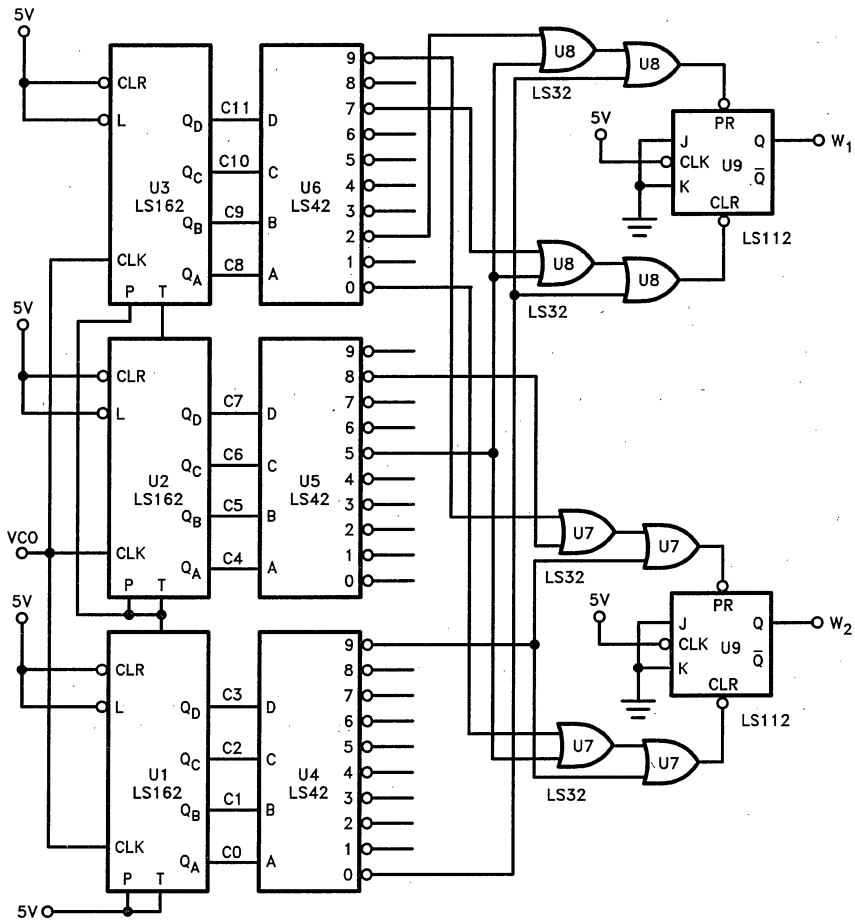
### PROTOTYPING

For most companies, prototypes are three to five level wire wrap boards built by inhouse technicians or outside contractors. During prototype fab, a certain amount of work has to be done to each IC. Part of this work is, adding bypass caps, labeling chips, and lead forming. In smaller companies, the board might be hand wrapped. Larger companies might use an automatic wrapper. Once the board is wrapped, a continuity check is done on each wire net to insure connections and minimize shorts.

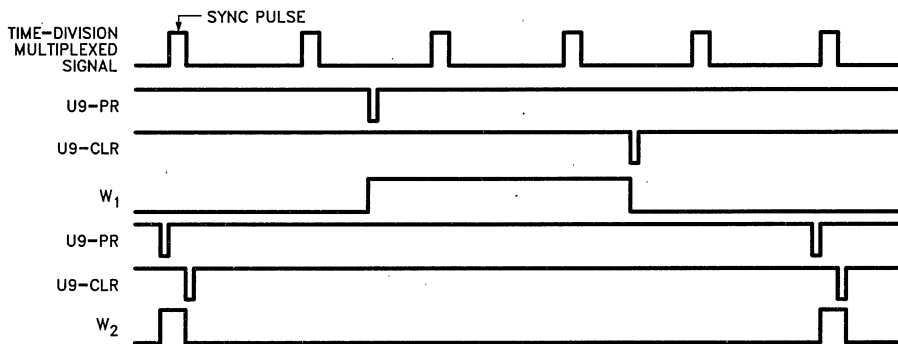
The turn around time for a proto-board is one to two weeks and can be shortened by paying a premium price. An alternate way of shortening this time is to simplify the board by using denser ICs.

### DEBUGGING

Fixing bugs on a proto-board involves unwrapping and wrapping connections, as well as replacing ICs. Making mods on a TTL board is very time consuming and error prone due to the large numbers of wires. Making mods with PALs is expensive since the part usually has to be junked. EPLDs in contrast, are re-programmable and lend themselves to all the revisions that are common in the early design stages.



292030-16



292030-17

Figure 11. Time Window Generator, TTL Circuit

## PCB LAYOUT

Artwork quotes are based on several factors. These are, board size, number of 16-pin chip equivalents, pad count, and the chip to board packing ratio. The chip equivalents are calculated by taking the total lead count (ICs and discretes) and dividing by 16. Pad count is the number of holes in the board. The packing ratio determines how much room an IC has around it. This is critical because space is needed to place sockets, vias, and trace bends. Currently, most service bureaus consider 0.75 square inches per IC to be the minimum packing density. This figure applies to DIPs only, other packages like SMT (Surface Mount Technology) will improve on this. However, for standard DIPs anything less than this might push the board into a multi-layer.

During schematic evaluation, the bureau doesn't usually charge for traces directly. Because they can't foresee the exact count, and they don't have time to count them on the sheets, they make a judgment based on previous jobs. If the board appears to be tight, their autorouter (CAD based) won't be as efficient, and more hand layout will have to be done. However, as more CAD based service bureaus integrate schematic capture front ends, the cost of traces and vias will be more visible.

Because the evaluation is subjective, the final cost varies, and is a combination of charges. However, because pad count can be determined easily, the overall price is usually gauged against a pad price.

## WINDOW CIRCUIT

### Background Information

In applications that involve time-division multiplexing, it is useful to have a circuit that windows a specific area of the bit stream [27]. The circuit of Figure 11 is a TTL implementation of such a circuit. The idea is to count time slots from a known reference and at a certain decode, set and clear a latch. The output of the latch is the time window, which might be used for further gating in other parts of the circuit. The TTL parts list is detailed in Table 9.

The PAL alternative of Figure 12 is comprised of two 16L8s and one 16R4. While the component count has been reduced from nine to three, there are still fourteen extra interconnections.

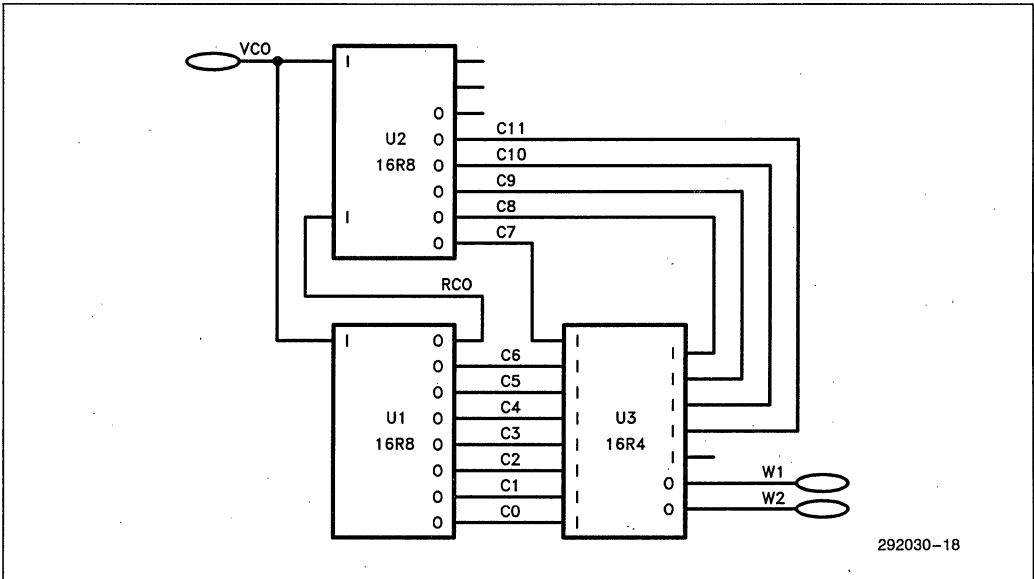
One 5C060 is needed to integrate the complete circuit. Fourteen out of the sixteen EPLD macrocells are used, and external traces are only the three I/O pins as shown in Figure 13.

### Production Costs

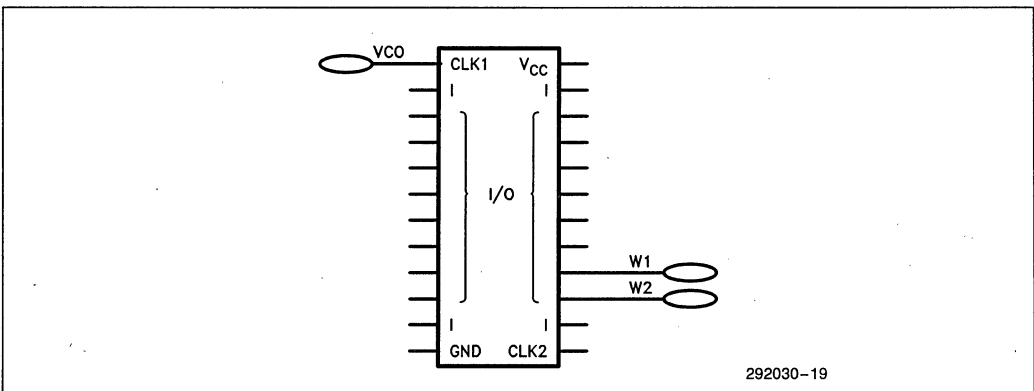
The production variables for the window circuit are shown in Table 10a and 10b, and the production costs in Table 11. The comparison shows three years of system costs for each implementation.

**Table 9. TTL Component List for Window Generator**

IC	Type	DIP	I <sub>cc</sub> (mA)	Area(In <sup>2</sup> )	\$
U1	LS162	16	32	.24	.49
U2	LS162	16	32	.24	.49
U3	LS162	16	32	.24	.49
U4	LS42	16	13	.24	.39
U5	LS42	16	13	.24	.39
U6	LS42	14	13	.24	.39
U7	LS32	14	9.8	.21	.18
U8	LS32	14	9.8	.21	.18
U9	LS112	14	6	.21	.29



**Figure 12. Time Window Generator, PAL Circuit**



**Figure 13. Time Window Generator, EPLD Circuit**

<b>Inventory:</b>		<u><b>Costs</b></u>									
Incoming insp. (\$/pin)		\$0.010									
Storage (\$/sq.ft./yr)		\$20.000									
Maintenance (\$/part)		\$0.030									
Processing (\$/part type/yr)		\$0.520									
Safety stock (%)		2%									
<b>Manufacturing:</b>		<u><b>Costs</b></u>									
PCB fab. (\$/sq.in.)		\$0.200									
Assembly (\$/part)		\$0.100									
Test (\$/part)		\$0.150									
Rework (\$/pin)		\$0.020									
QC (\$/pin)		\$0.004									
Power (\$/watt)		\$1.000									
Interconn		\$0.020									
Program (\$/part)		\$0.250									
Caps. (each)		\$0.005									
(a)											
<b>Integrated Circuits</b>											
<b>Component Count:</b>											
<b>Package</b>	<b>TTL</b>	<b>PLA</b>	<b>EPLD</b>								
DIP14	3										
DIP16	6										
DIP20		3									
DIP24			1								
<table border="1" style="display: inline-table; border-collapse: collapse;"> <thead> <tr> <th style="padding: 2px;">ICs</th> <th style="padding: 2px;">Types</th> </tr> </thead> <tbody> <tr> <td style="padding: 2px;">TTL</td> <td style="text-align: center; padding: 2px;">4</td> </tr> <tr> <td style="padding: 2px;">PLA</td> <td style="text-align: center; padding: 2px;">2</td> </tr> <tr> <td style="padding: 2px;">EPLD</td> <td style="text-align: center; padding: 2px;">1</td> </tr> </tbody> </table>				ICs	Types	TTL	4	PLA	2	EPLD	1
ICs	Types										
TTL	4										
PLA	2										
EPLD	1										
<b>Circuit Requirements:</b>		<u><b>Icc (max)</b></u>	<u><b>Interconnects</b></u>								
TTL circuit (total mA).		160	52								
PLA circuit (total mA).		360	14								
EPLD circuit (total mA).		15	0								
(b)											

**Tables 10a and b. Window Circuit Cost Variables**

**Table 11. Window Circuit Production Costs**

<b>AVERAGE COMPONENT COST</b>									
<b>Package</b>	<b>Year 1</b>			<b>Year 2</b>			<b>Year 3</b>		
	<b>TTL</b>	<b>PLA</b>	<b>EPLD</b>	<b>TTL</b>	<b>PLA</b>	<b>EPLD</b>	<b>TTL</b>	<b>PLA</b>	<b>EPLD</b>
DIP14	\$0.22			\$0.19			\$0.17		
DIP16	\$0.44			\$0.37			\$0.26		
DIP20		\$2.00			\$1.70			\$1.56	
DIP24			\$6.00			\$4.20			\$2.90
<b>PRODUCTION COSTS</b>									
<b>Item (costs per part)</b>	<b>Year 1</b>			<b>Year 2</b>			<b>Year 3</b>		
	<b>TTL</b>	<b>PLA</b>	<b>EPLD</b>	<b>TTL</b>	<b>PLA</b>	<b>EPLD</b>	<b>TTL</b>	<b>PLA</b>	<b>EPLD</b>
Components	\$0.367	\$2.000	\$6.000	\$0.310	\$1.700	\$4.200	\$0.230	\$1.560	\$2.900
Incoming Insp.	\$0.153	\$0.200	\$0.240	\$0.153	\$0.200	\$0.240	\$0.153	\$0.200	\$0.240
Inventory									
Maintenance	\$0.029	\$0.038	\$0.045	\$0.029	\$0.038	\$0.045	\$0.029	\$0.038	\$0.045
Storage	\$0.032	\$0.042	\$0.050	\$0.032	\$0.042	\$0.050	\$0.032	\$0.042	\$0.050
Processing	\$0.231	\$0.347	\$0.520	\$0.231	\$0.347	\$0.520	\$0.231	\$0.347	\$0.520
Printed Circuit Board									
Fabrication	\$0.046	\$0.060	\$0.072	\$0.046	\$0.060	\$0.072	\$0.046	\$0.060	\$0.072
Trace costs	\$0.116	\$0.093	\$0.000	\$0.116	\$0.093	\$0.000	\$0.116	\$0.093	\$0.000
Assembly	\$0.096	\$0.125	\$0.150	\$0.096	\$0.125	\$0.150	\$0.096	\$0.125	\$0.150
Board test	\$0.150	\$0.150	\$0.150	\$0.150	\$0.150	\$0.150	\$0.150	\$0.150	\$0.150
Rework	\$0.015	\$0.020	\$0.024	\$0.015	\$0.020	\$0.024	\$0.015	\$0.020	\$0.024
QC	\$0.061	\$0.080	\$0.096	\$0.061	\$0.080	\$0.096	\$0.061	\$0.080	\$0.096
Power Supply	\$0.089	\$0.600	\$0.075	\$0.089	\$0.600	\$0.075	\$0.089	\$0.600	\$0.075
<b>Total Cost/Part</b>	<b>\$1.385</b>	<b>\$3.754</b>	<b>\$7.422</b>	<b>\$1.328</b>	<b>\$3.454</b>	<b>\$5.622</b>	<b>\$1.248</b>	<b>\$3.314</b>	<b>\$4.322</b>
<b>Total Cost/System</b>	<b>\$12.463</b>	<b>\$11.263</b>	<b>\$7.422</b>	<b>\$11.953</b>	<b>\$10.363</b>	<b>\$5.622</b>	<b>\$11.233</b>	<b>\$9.943</b>	<b>\$4.322</b>
<b>Additional Costs/System</b>									
Programming loss	\$0.000	\$0.120	\$0.000	\$0.000	\$0.102	\$0.000	\$0.000	\$0.094	\$0.000
Safety stock	\$0.165	\$0.600	\$0.120	\$0.140	\$0.510	\$0.084	\$0.104	\$0.468	\$0.058
Programming fee	\$0.000	\$0.750	\$0.250	\$0.000	\$0.750	\$0.250	\$0.000	\$0.750	\$0.250
De-coupling caps	\$0.045	\$0.015	\$0.005	\$0.045	\$0.015	\$0.005	\$0.045	\$0.015	\$0.005
<b>True mfg. cost/system</b>	<b>\$12.673</b>	<b>\$12.748</b>	<b>\$7.797</b>	<b>\$12.137</b>	<b>\$11.740</b>	<b>\$5.961</b>	<b>\$11.381</b>	<b>\$11.269</b>	<b>\$4.635</b>



The production costs again show that the system cost for the first year is better with EPLDs. The two consecutive years show that the declining price of EPLDs make them an excellent candidate for systems that will ramp up production at that time.

## Window Circuit Conclusion

The TTL version of the circuit was implemented with MSI counters and decoders. As a result, the PAL implementation was bound by the number of count bits and had to be programmed into two PALs. In circuits like this, it is useful to rewire the decode for different counts depending on the application. The PAL implementation allows this by incorporating the decode and output latches into one IC.

The EPLD implementation tackles the MSI integration quite easily and also provides the capability to reprogram the decoder. Since the counter and output latches consist of fourteen registered outputs, the sixteen macrocells of the 5C060 easily accommodate the needed functions.

## SUMMARY

We have examined the hidden costs of production and how they differ for several logic alternatives. By examining these costs, we have shown that while an EPLD is presently a more expensive part, its level of integration reduces system costs and improves reliability. The following items should be considered when evaluating logic alternatives:

- system cost is determined by more than component cost
- system cost and reliability is influenced by the type and amount of components used
- semiconductors have a life cycle that determines their present price at design, and at production time

In summary, when all system costs are considered, EPLDs can provide cost savings to the design and production of most board designs.

## REFERENCES

1. The Future Is Now: Extending CAE into test of custom VLSI.  
Robert S. Broughton, Tektronix.  
Michael G. Brashler, Tektronix.  
IEEE International Test Conference Proceedings, 1984
2. Reducing The Cost of Quality Through Test Data Management.  
Paul N. Manikas, GenRad Inc.  
Stephen G. Eichenlaub, Harvard University.  
IEEE International Test Conference Proceedings, 1983
3. A Quantitative Analysis Of The Trade-offs Between Higher Capital Investment and Higher Yield In PCB Testing.  
Mark A. Myers, Teradyne Inc.  
IEEE International Test Conference Proceedings, 1984
4. An Analysis Of The Cost And Quality Impact Of LSI/VLSI Technology On PCB Test Strategies.  
Mark A. Myers, Teradyne Inc.  
IEEE International Test Conference Proceedings, 1983
5. IC Quality Control By The User.  
Roger Dunn, Xerox Corp.  
IEEE International Test Conference Proceedings, 1983
6. An Analysis Of The Economics of Self Test.  
P. Varma, University of Manchester.  
A. P. Ambler, University of Manchester.  
K. Baker, GEC Research Labs.  
IEEE International Test Conference Proceedings, 1984
7. In Circuit Testability Factors: Shoot With A Rifle.  
Douglas W. Raymond, Zehntel Production Services.  
IEEE International Test Conference Proceedings, 1984
8. Seven Steps To Zero Defects.  
D. W. Rudd, AT&T Technologies.  
Circuits Manufacturing, June 1986
9. Rework Forum  
Donald Ford, Senior Editor.  
Circuits Manufacturing, September 1986
10. Manufacturing Defect Analyzers: Annual Round-up.  
Evaluation Engineering magazine, August 1986
11. Assembly: Automation Makes It Better.  
Roland W. Roy and Gordon Weeks, Andover Controls.  
Circuits Manufacturing, February 1986
12. Shrinking Lines Squeeze Processes.  
Jerry Murray, West Coast Editor.  
Circuits Manufacturing, September 1986
13. Ribbon Cable for Reliable Interconnections.  
Bennett W. Brachman, Xport Trading Inc.  
Electronic Packaging and Production magazine, July 1986

14. TQC and JIT: Partners In Production.  
Rick Walleigh, Hewlett Packard.  
Circuits Manufacturing, February 1986
15. Automated Handling/Sorting: Multisite Development Moves to Back Burner.  
Evaluation Engineering magazine, May 1986
16. Software Charts The Course of Component Testing.  
Ronald Pound, Editor.  
Electronic Packaging and Production magazine, June 1986
17. Complexity, PLDs Drive The Market.  
Evaluation Engineering magazine, July 1986
18. Intel User Defined Logic Handbook.  
Intel Corp. 1986
19. VLSI Semicustom Design Guide.  
CMP Publications, Summer 1986
20. AMD Programmable Array Logic Handbook.  
Advanced Micro Devices, 1984
21. Handbook Of Industrial Engineering.  
Gavriel Salvendy, Editor, Purdue University  
John Wiley & Sons Publications
22. Components Quality/Reliability Handbook.  
Intel Corporation.
23. The Cost Edge.  
DM DATA Corp.  
Scottsdale, AZ
24. Semiconductor Purchasing Strategies Integrated Circuits Engineering Corp.  
Scottsdale, AZ
25. Status 1986 Integrated Circuits Engineering Corp.  
Scottsdale, AZ
26. EDN Semicustom Design Series  
EDN Magazine, 1985
27. EDN Design Ideas  
EDN Magazine, 1985



**APPLICATION  
NOTE**

**AP-321**

November 1988

**Fitting the 5C180**

**TODD KOELLING**  
PROGRAMMABLE LOGIC APPLICATIONS  
INTEL CORPORATION

Order Number: 292053-001

## INTRODUCTION

In many ways, fitting the 5C180 is like climbing a mountain. Just when what appears to be the summit is reached, another summit is revealed behind it. This may occur several times before the actual summit is surmounted.

Likewise, fitting a 5C180 may have several false summits. Just when one has conquered what appears to be the "problem", another problem often appears behind it. This may occur several times before the design fitting is complete.

This application note addresses the problems that can be encountered when trying to fit a 5C180 and offers suggestions on how to get past them. The key to the climb is examining what resources are still available after the software\* complains that a particular resource is not available.

## SUMMIT NUMBER ONE: PIN ESTIMATE

Before keying in the design, it is best to estimate the I/O pin requirements. This is done by counting the total number of inputs to the device and outputs from the device.

**PROBLEM:** Not enough Input Pins

**HELP:** Run all synchronous clocks through Clock Buffers (CLKBs). Shared clocks may use the same CLKB output which may result in reduction from 4 CLK input pins to 1 CLK input pin (see Figures 1a &

\*PLS II ver. 1.1 or later is ESSENTIAL for 5C180 designs as the fitting algorithm was significantly improved with this release.

\*PLS II ver. 1.5 or later is HIGHLY RECOMMENDED as the error messages and Utilization Report Files were significantly enhanced with this release.

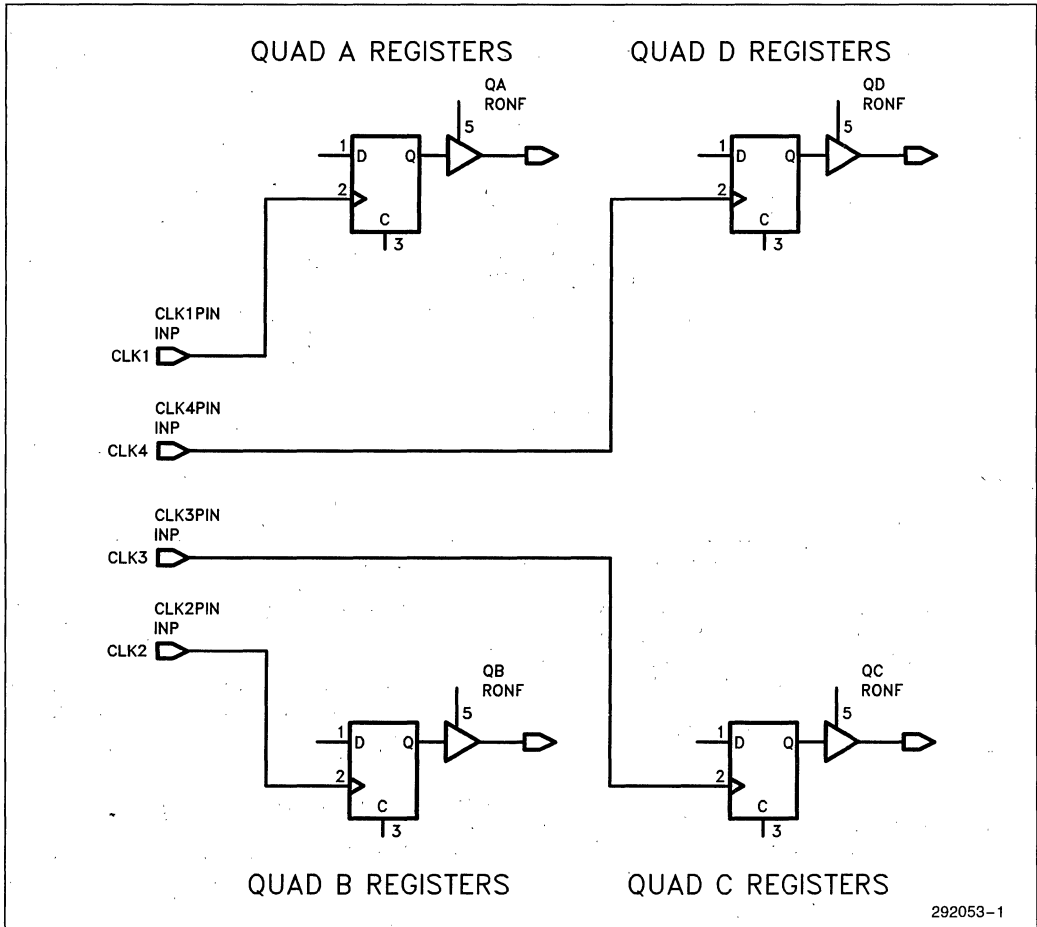


Figure 1a. Summit One—Input Clocks Before

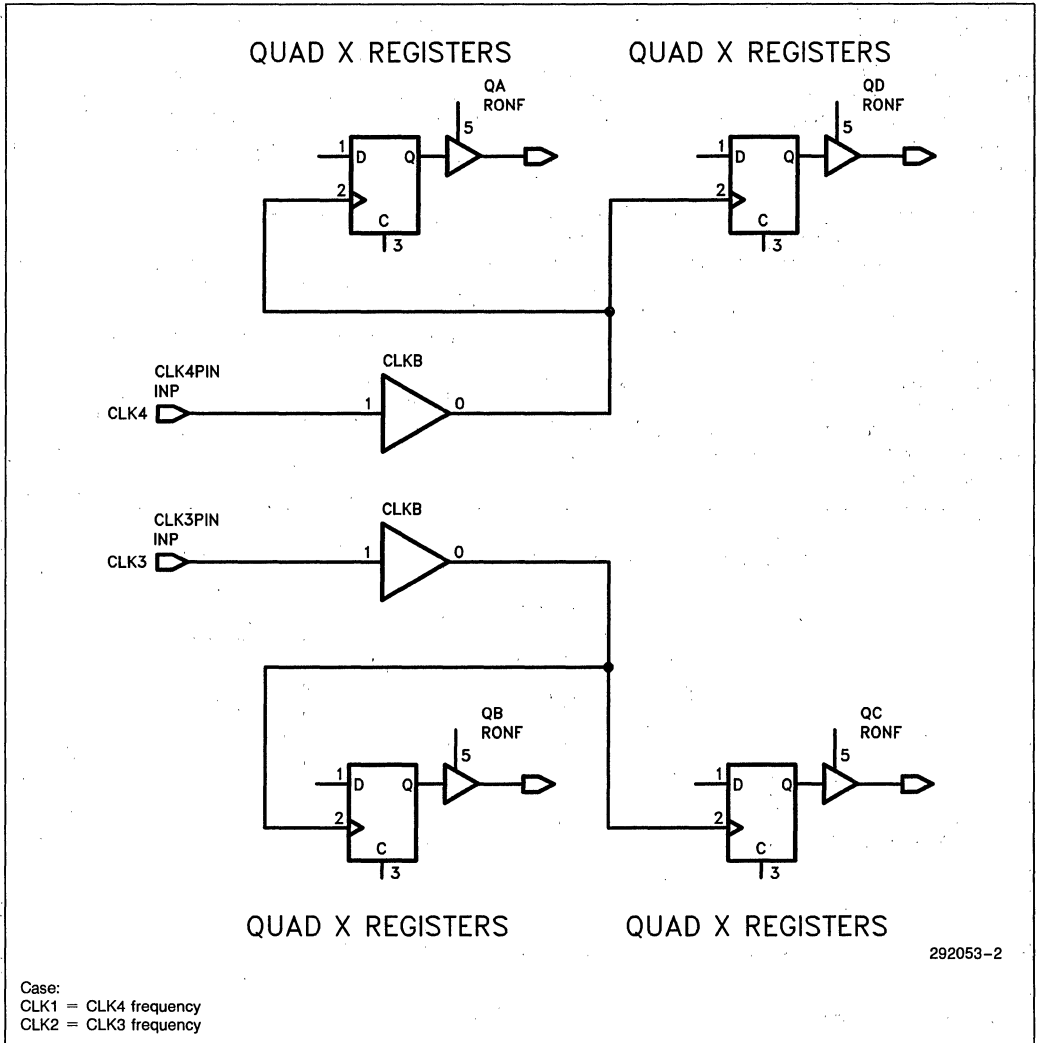


Figure 1b. Summit Two—Input Clocks After

1b). This also frees the registers that the clock feeds from the synchronous clock pin quadrant, increasing the chance of fitting later on. **THIS PRACTICE IS RECOMMENDED FOR ALL DESIGNS.**

**PENALTY:** Input setup time is shortened. (See Synchronous vs. Asynchronous A.C. Characteristics in Data Sheet).

If clock buffering cannot solve the problem, the design must be repartitioned to reduce the number of input pins. Repartitioning is explained in the next section.

**SUMMIT NUMBER TWO: MACROCELL ESTIMATE**

If the I/O pin requirements can be met, the next step is to consider the macrocell requirements. The total macrocell count can be estimated by counting the number of outputs plus the number of internal registers.

**PROBLEM:** Not enough macrocells

**REPARTITIONING:** Unless the fundamentals of the design can be changed, this error means that the design

must be repartitioned. This is done by removing part of the circuitry and placing it in a second device such as a 5C060 or 5C090. The 5C060 and 5C090 are recommended since their architectures (and therefore their ADFs) are nearly identical to those of the 5C180 (the NOCF and COCF primitives are the only exceptions).

Portions of the 5C180 ADF can be easily transferred into one of the smaller devices or the smaller device ADFs can be transferred back to the 5C180 if sufficient room is freed up later on. **IT IS RECOMMENDED THAT FOUR OR FIVE UNUSED MACROCELLS BE LEFT IN THE 5C180 FOR USE BY LATER STAGES.**

### SUMMIT NUMBER THREE: SUCCESSFUL TRANSLATION

With the design entered, the next summit is successful translation.

**ERROR:** \*\*\*ERR-MAC-No macrofunction for: ...

**EXPLANATION:** The Macro Expander Module cannot find a macro for a network element.

**FIX:** Make sure correct search path is available for macro libraries. Check for typo or syntax error. If using schematic capture, make certain that only valid EPLD library symbols were entered.

**ERROR:** Any "\*\*\*\*ERROR-XLT-..."

**EXPLANATION:** The Translator found a problem with the way the design was entered. These errors are basically syntax errors which violate ADF format. It may be a simple typo, missing parenthesis or missing semicolon. Remember that the iPLS II LOC does differentiate between upper and lower case letters. If using schematic capture, make sure that all device inputs and outputs have pin symbols and that all the pins and wires are properly labeled.

**FIX:** Refer to your iPLS II manual or call the EPLD Hotline, 1-800-323-EPLD, for help on the tough ones.

### SUMMIT NUMBER FOUR: REGISTER CLOCK INPUTS

**ERROR:** \*\*\*ERROR-XLT-Clock input must be driven by INP or CLKB

**EXPLANATION:** The clock for a flip-flop must be driven synchronously by a direct quadrant clock pin input (INP) or asynchronously through a Clock Buffer (CLKB). This problem occurs when an equation or gate logic is connected directly to the register clock input.

**FIX:** In order to tell the LOC software that the clock for a flip-flop will be driven by an equation or gate logic, a Clock Buffer (CLKB) must be placed between the equation or logic and the register clock input for each register that is asynchronously clocked.

### SUMMIT NUMBER FIVE: ASYNCHRONOUS CLOCKS AND OUTPUT ENABLES

**ERROR:** \*\*\*ERROR-XLT-OE with asynchronous clock not allowed

**EXPLANATION:** Asynchronous clock and output enable can't be used at the same time in the same macrocell. The 5C180 basic macrocell architecture, Figure 2, shows why. A single p-term is shared between the asynchronous clock and the output enable. This means that both switches in the diagram can be up or both switches can be down. By trying to use a p-term output enable with an asynchronous clock, the top switch would have to be down while the bottom switch is up. This cannot be done as then the register would be clocked and enabled with the same signal.

**WORKAROUND:** To get around this problem, one of the signals must be routed through another macrocell (see Figures 3a-b). The clock could be generated in another macrocell, sent out to a pin, then sent back in on the synchronous clock pin. Alternately, in a first macrocell the register is placed as an asynchronously clocked NORF. In a second macrocell, the register feedback is sent out to a pin using a CONF enabled by the desired enable signal.

**PENALTIES:** Routing the clock through a separate macrocell and back in offers slightly better performance—since the synchronous clock to output time is faster than a second macrocell delay, but this implementation uses a lot of resources—three pins and two macrocells. The second method, routing the feedback from the register back and controlling the output enable in a second macrocell is more straightforward and uses less resources.

### SUMMIT NUMBER SIX: GREATER THAN ONE PRODUCT-TERM REGISTER CONTROLS

**ERRORS:** \*\*\*INFO-FIT- Eqn. too big, 4/-1 PTerm(s), on OE signal OE3

\*\*\*INFO-FIT- Illegal inversion of CLEAR input (CLR1)

**EXPLANATION:** As shown in the basic 5C180 macrocell architecture, Figure 2, only one product term (multiple input AND gate) is available for the register

clock, clear, and output enable. This means that any control resource containing an OR gate following Boolean minimization will not fit. Likewise, any control resource requiring an invert will not fit either. To find the offending signal, LOOK AT THE EQUATIONS SECTION OF THE LOGIC EQUATION FILE (.LEF).

**WORKAROUND:** Once the offending signal has been located, it must be routed through another macrocell using an NOCF primitive (see Figure 4a-b). If the control signal is a clock, then a clock buffer (CLKB) must also be added.

**PENALTY:** Unless a trick explained below can be used, this routing results in the use of an additional macrocell and a doubling of the signal propagation delay.

**Clr Fitting Trick**

**PROBLEM:** Register clear input breaks 1 p-term resource limit

**TRICK:** If register has D input of either VCC or GND, substitute SR Flip-Flop.

**EXPLANATION:** D-type EPLD register has only 1 AND gate feeding CLR; SR Flip-Flop utilizes logic array for CLR input allowing a max of 8 AND gates (p-terms) for the CLR resource.

**PENALTIES:** SR Flip-Flop is synchronously clocked. D register has asynchronous clear.

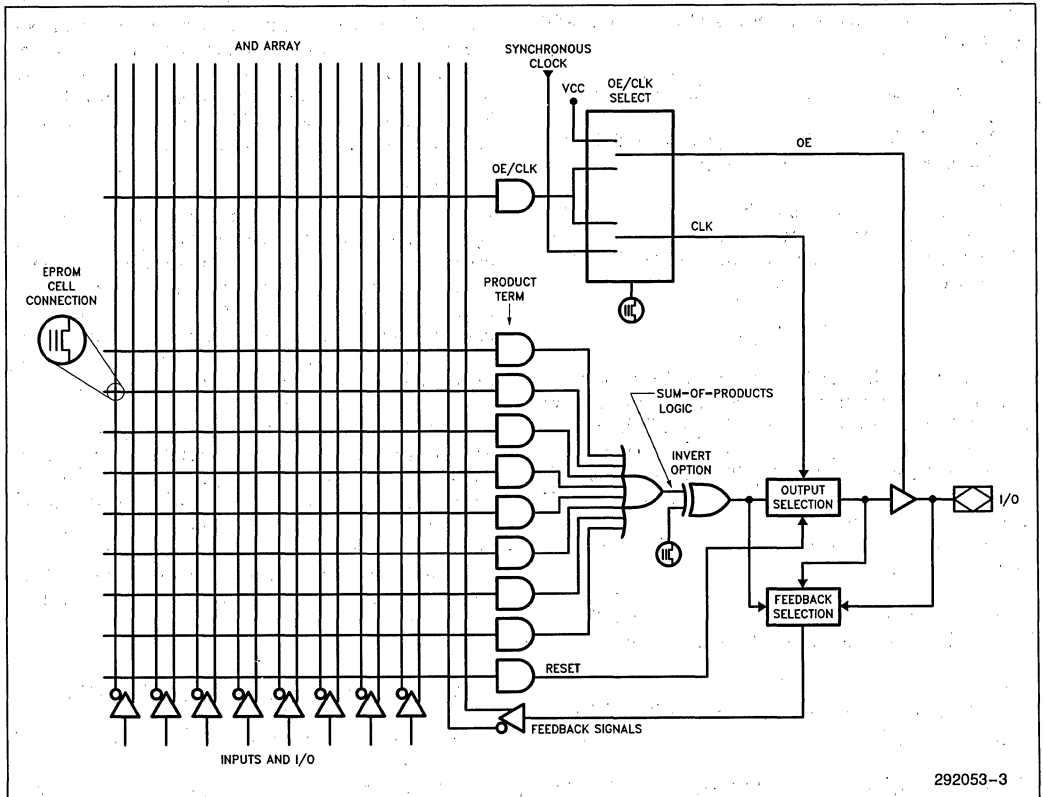


Figure 2. Basic Macrocell Architecture of the 5C180

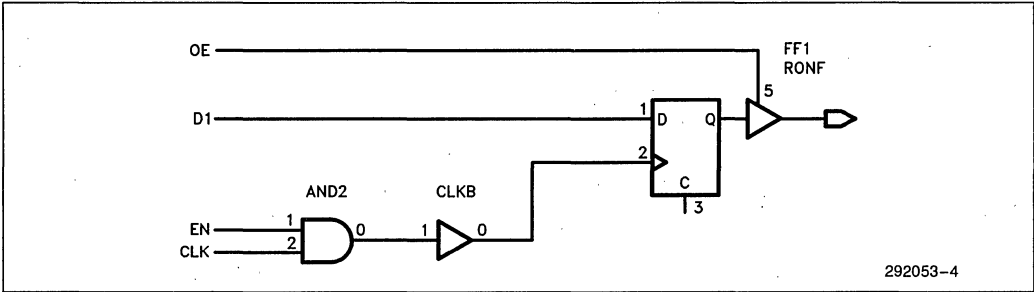


Figure 3a. Summit Five—Asynchronous Clock and OE Before

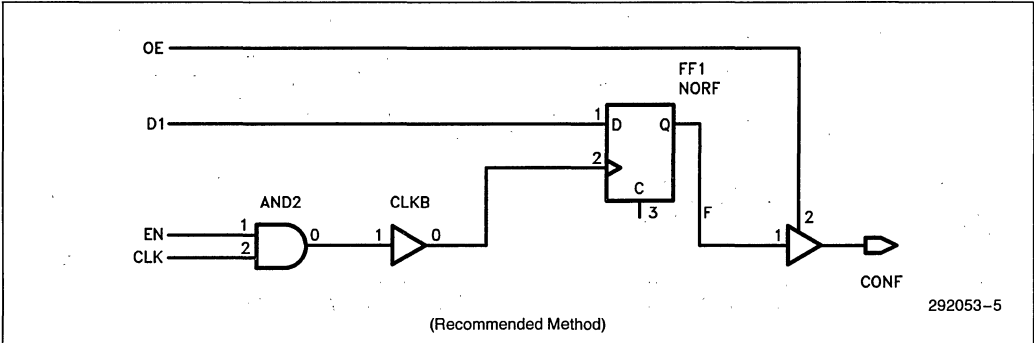


Figure 3b. Summit Five—Asynchronous Clock and OE After

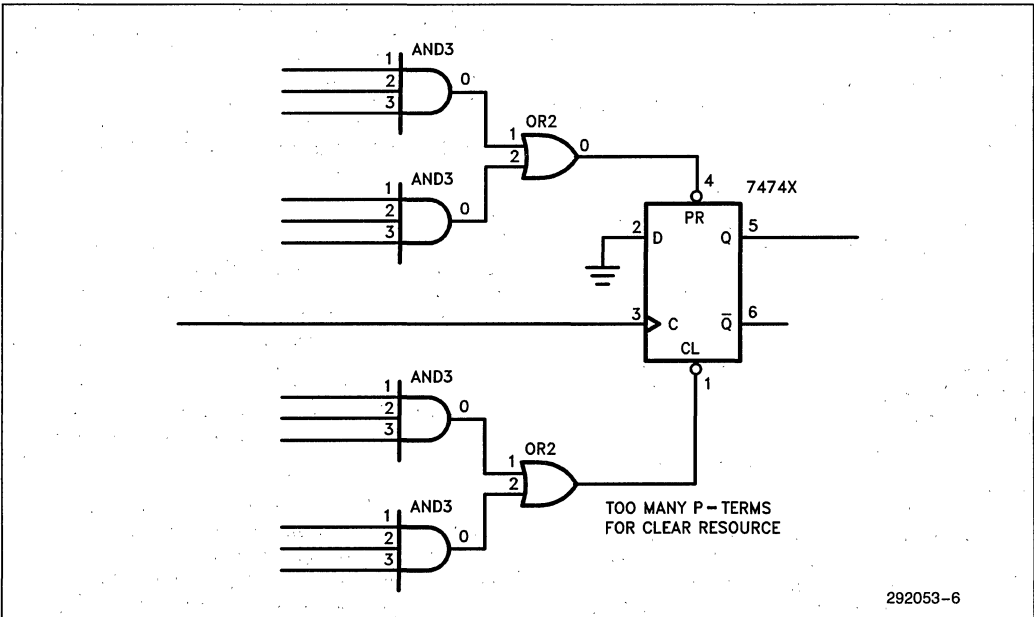


Figure 4a. Summit Six—Too Many P-Terms on Control - Clear



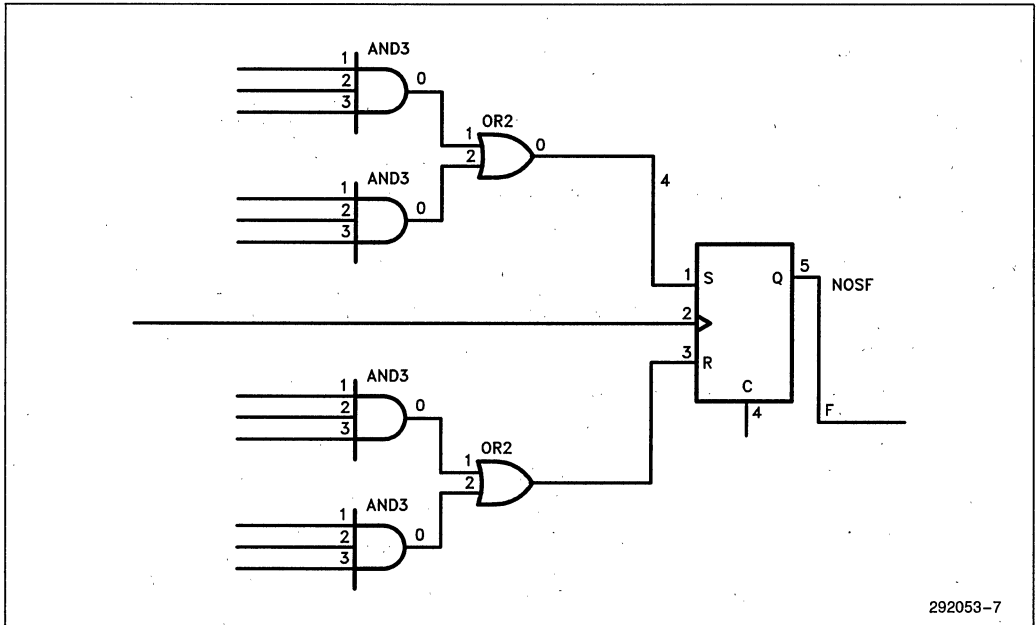


Figure 4b. Summit Six—SR Flip-Flop Equivalent Implementation

## OE Fitting Trick

**PROBLEM:** Output enable on an equation or combinational equation exceeds 1 p-term resource limit.

**TRICK:** If a low output rather than a tri-state output can be tolerated, the signal can be gated rather than tristated.

**EXPLANATION:** Run the OE and the equation through an AND gate before going to a pin. The output of the pin will only follow the equation when the enable is active, otherwise it will be zero.

**PENALTY:** Forced low rather than tri-state output.

## SUMMIT NUMBER SEVEN: NOT ENOUGH P-TERMS FOR AN EQUATION

**ERROR:** \*\*\*INFO-FIT- Too many PTerms to fit in any MCell: 10/8 for EQN.

**EXPLANATION:** Since the 5C180 has a maximum of eight product terms per macrocell, there's a chance that this number may be exceeded by the requirements of an equation. If so, the equation is cited by the LOC and can be examined by looking at the EQUATIONS section of the .LEF.

**WORKAROUND:** The workaround for this situation may already be in place! If any portion of the logic (or equation) is routed into a NOCF or CONF elsewhere in the design, that feedback can be taken and routed into the equation (see Figure 5a-b). This means a single feedback node—rather than several nodes will now feed the equation and thereby reduce the p-term count. (If the feedback is to be taken from a CONF primitive, the CONF must be changed to a COCF or COIF to make the feedback available.)

If part of the logic or equation is not routed into a NOCF or CONF elsewhere in the design, then part of the equation must be routed through a NOCF, COCF, or COIF primitive. A NOCF is recommended as it does not use a pin if placed in a global macrocell. If several equations are in violation of the eight p-term maximum, try to choose a group of logic that is common to all of the equations. In this manner, the p-term count for several equations can be brought down with the use of single extra macrocell, rather than the use of a macrocell for each equation.

**PENALTY:** Any time a portion of an output signal must be routed through another macrocell a speed penalty is incurred (roughly one propagation delay). If an already existing macrocell can be found, then there is no architectural penalty. If a new one must be created, then another macrocell is added to the total macrocell count.

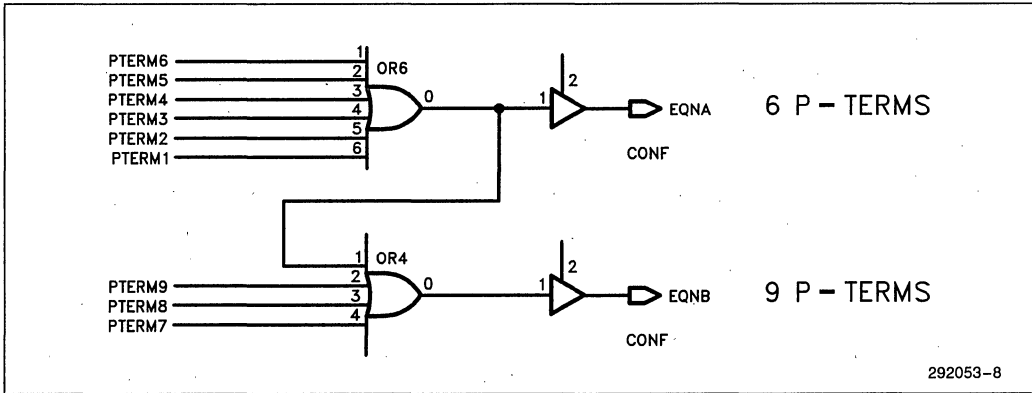


Figure 5a. Summit Seven—Too Many P-Term Equation Before

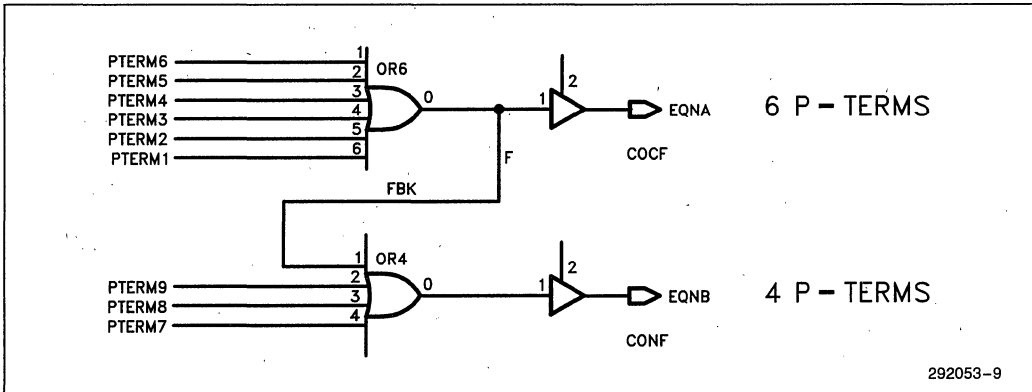


Figure 5b. Summit Seven—Too Many P-Term Equation After

**SUMMIT EIGHT: MACROCELL RESOURCES EXCEEDED**

**ERROR: \*\*\*INFO-FIT-** Design requires too many macrocells

**EXPLANATION:** If this error didn't occur at the beginning, there's a good chance summits five, six or seven will push the macrocell count over the limit. (Remember that the macrocell count includes not only the outputs, but also the buried resources such as NOCFs, NORFs and NOTFs). To find out exactly how many macrocells the design requires, LOOK AT THE NETWORK: SECTION OF THE LOGIC EQUATION FILE (.LEF). The inputs list in the LEF will list both the outputs and all the buried resources required by the design. If the count exceeds 48, then too many macrocells are required.

**FIX:** Repartition. The same applies if the number of input pins is exceeded.

**THE FINAL ASCENT: NOT ENOUGH GLOBAL FEEDBACK!**

Congratulations! If you have made it this far, you have demonstrated courage, intelligence and tenacity beyond that of the average climber. You will soon be rewarded, but first there is one more obstacle to be overcome. Welcome to the North Face of local/global feedback!

**A Word About Local/Global Feedback**

First of all, why does local/global feedback exist? The answer can be found in the graph shown in Figure 6. The propagation delay versus array size is shown for the 5C060/090/180 family. As the number of inputs into the array increases, the propagation delay increases...exponentially. If all the inputs and feedback were made global, the 5C180 would have 136 inputs feeding each array (remember that both true and complement polarities must be fed into the array of a PLD architec-

ture). This would have put the 5C180 Tpd in the 250 - 300 ns range! By making eight macrocells local for four quadrants, the number of array inputs was dropped to 88 and the Tpd subsequently decreased to 75 ns.

The tradeoff to the local/global routing scheme is more difficult design routing. With the help of the iPLS II and a couple of tricks, however, most designs can still be fit.

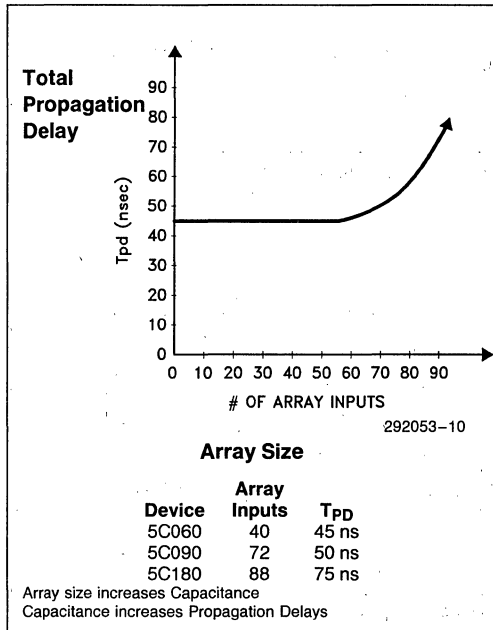


Figure 6. Propagation Delay vs. Array Size for the 5C060/090/180 Family

**A Few Notes**

The global/local macrocell assignments are shown in Figure 7. Please note that:

1. Dedicated input pins are GLOBAL.
2. Global macrocell I/O pin are GLOBAL.
3. Global macrocell internal feedback paths are LOCAL.

4. Local macrocell pin/feedback paths are LOCAL.

where GLOBAL means that the signal feeds all macrocells and LOCAL means that the signal only feeds the macrocells in its quadrant.

**Clock Input Pins**

The clock input pins feed the global bus like the regular inputs, except the synchronous register input connection is dedicated to a particular quadrant. Thus, each clock input can be used as a logic input in all quadrants or a clock input in its own quadrant. To be used as a register clock input in a quadrant outside its own, however, it must be tapped from the global bus via an asynchronous clock buffer (CLKB).

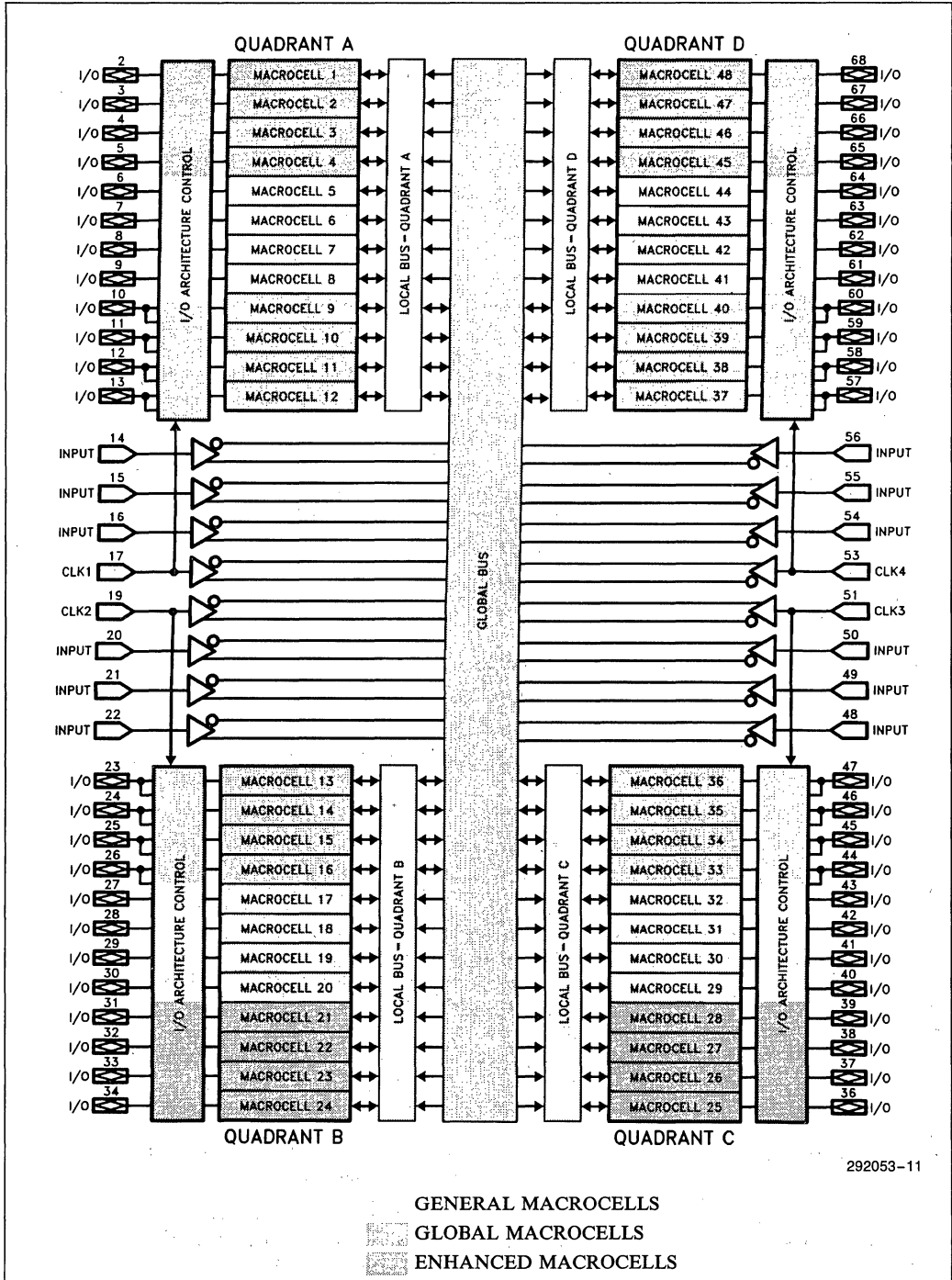
**Global Macrocell Feedback**

The feedback path is local for GLOBAL macrocells while the I/O input is global for all GLOBAL macrocells. Thus, changing the feedback of a register or combinational equation from a standard feedback to I/O pin feedback path will change the routing from local to global. The iPLS II LOC automatically recognizes and performs this through a process called "promoting". With the promotion process, global routing can be obtained on signals that would otherwise remain local.

\*\*\*INFO-FIT- Promoted "TEQNF" from NOCF to COIF

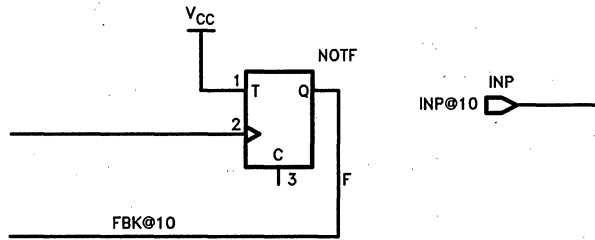
**Burying a Register in a Global Cell**

Because the global macrocells have separate register and I/O pin feedback paths, it is possible to "bury" a register or equation by disabling the output buffer and still use the pin as an input. The iPLS II LOC automatically assigns an input to the pin of a buried register macrocell if it is necessary and possible. Such assignments are documented in the Utilization Report File (.RPT). If manual assignment is desired, it may be performed by placing the input pin assignment in the ADF INPUTS: list and assigning the buried register feedback to the same pin in the OUTPUTS: list (Figure 8). Registers or equations can only be buried on global macrocells, since local macrocells only have one feedback path that is used for either the register or the pin feedback.



292053-11

Figure 7. 5C180 Block Diagram



292053-12

**Buried Register Pin Assignment in ADF**

Intel  
 PLDO Apps  
 July 27, 1988  
 5C180 Buried Reg Pin Assignments

PART: 5C180  
 INPUTS: A@15, B@10, CLK@17 % Assign input B to pin 10 %  
 OUTPUTS: FBK@10 % Assign buried reg feedback FBK %  
 % to pin 10 (GLOBAL macrocell 9) %

**NETWORK:**

A = INP(A) % Inputs %  
 B = INP(B)  
 CLK = INP(CLK)

FBK = NORF(IN,CLK,GND,GND) % Buried Register %

**EQUATIONS:**

IN = A \* B \* FBK; % Register Input Equation %

END\$

**Figure 8. Assigning Buried Reg in Schematic**

**Two Global Fitting Tricks**

If the LOC is unable to fit the design, there are a couple of manual tricks that may help:

**PROBLEM: NOT ENOUGH GLOBAL FEEDBACK**

**RESOURCES AVAILABLE: EXTRA MACROCELLS**

**TRICK:** Duplicate the macrocell logic that needs to be global in two (or more) regions with appropriate renaming (see Figure 9).

**EXPLANATION:** This makes the signal available in two regions via two local macrocells rather than one which can't be global.

**PENALTIES:** There may be a slight timing discrepancy between the two macrocells for combinational logic, but any discrepancy will be small (< 2 ns).

**PROBLEM: NOT ENOUGH GLOBAL FEEDBACK**

**RESOURCES AVAILABLE: EXTRA INPUT PINS**

**TRICK:** Send out the signal that needs to be global and externally connect it to one of the input pins.

**EXPLANATION:** Inputs feed the global bus, making the signal available in all quadrants.

**PENALTIES:** An output buffer plus input buffer minus feedback delay is added (approximately 25 ns). An external connection must be made on the board.

**NOTE:**

For the previous tricks, look at the Utilization Report (.RPT) file. The "Interconnect Cross Reference" is particularly useful for examining the routing requirements of the design.

If the previous tricks cannot be done (see Figure 11) and scrutinization of the Interconnect Cross Reference reveals no other way to achieve the desired routing, repartitioning is necessary. That is, place a chunk of interconnected logic into a 5C060 or 5C090 and go back to the start.

**CONCLUSION**

Fitting the 5C180 is a process with many stages. One difficulty may hide the next and fixing one problem will sometimes uncover another. Equipped with the iPLS II LOC and a few tricks, however, fitting can be accomplished.

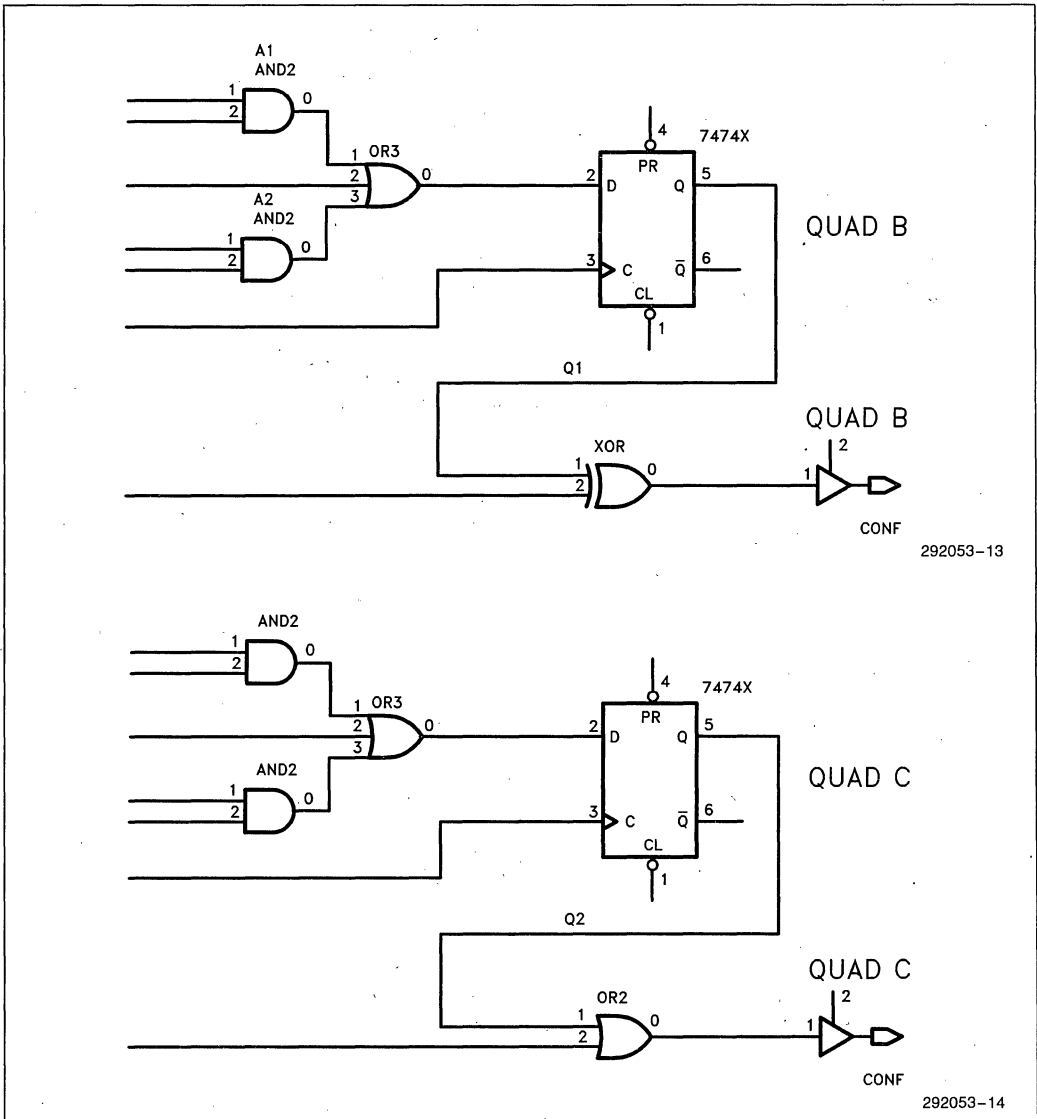


Figure 9. Not Enough Global Feedback Extra Macrocells—Fit

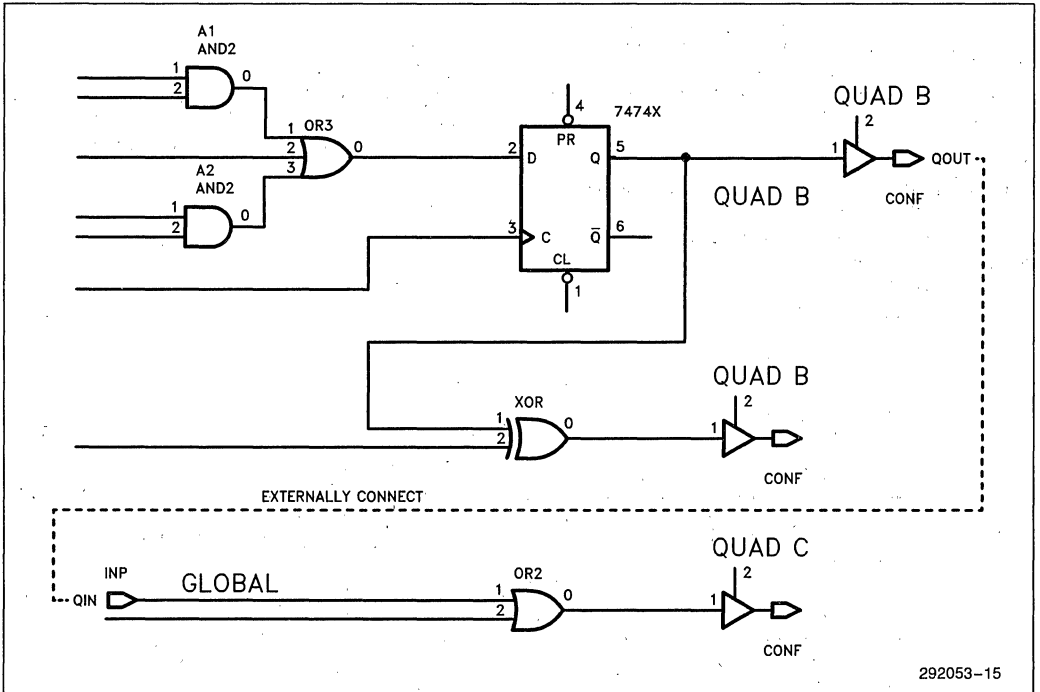


Figure 10. Not Enough Global Feedback Extra Inputs—Fit

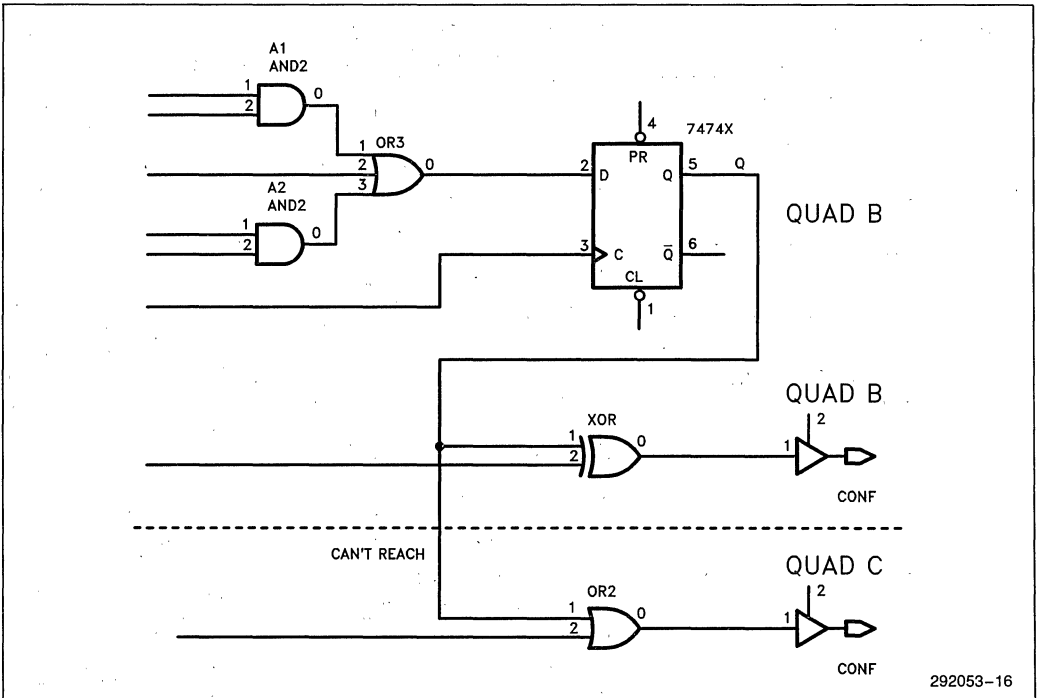


Figure 11. Not Enough Global Feedback—No Fit



September 1988

# **5C180 vs. EP1800: A Comparison of Device Specifications**

**LILIYAS KOUMIS**  
PROGRAMMABLE LOGIC APPLICATIONS  
INTEL CORPORATION

Order Number: 294006-001



## INTRODUCTION

This engineering report compares the Intel 5C180 EPLD with the Altera EP1800 EPLD showing how the specifications for the two devices relate to one another. Because Intel and Altera use a different methodology for specifying parameters for this device, the most significant hurdle to overcome when performing a comparison is to correlate the different specs. That correlation is performed here in table format.

*In summary, the Intel parts meet or exceed the specifications for the equivalent Altera parts.* The equivalent Intel/Altera devices are shown below. All numbers are based on the most current data sheet specs. (Intel 5C180 Data Sheet, order number 290111-005. Altera 1988 Data Book.)

Intel	Altera
5C180-70	EP1800-2
5C180-75	EP1800-3
5C180-90	EP1800

The tables that follow compare each spec. listed in the Intel 5C180 data sheet for each of the three versions of the device. A description of the parameter is listed, followed by the Intel mnemonic and the Intel spec. The

equivalent Altera value is then listed. The formula used to determine the Altera value is provided on a second line to aid in correlating Altera's internal timing numbers to Intel's external numbers. Intel specifies device parameters (i.e., input pin to output pin), while Altera specifies internal timing paths (i.e., input pad delay, logic array delay, etc.). Intel's specifications reflect numbers that can be measured, rather than internal numbers that must be estimated from external measurements. Altera's internal timing specifications appear at the top of each page.

Note that a new spec. has been added to many of the parameters. This new spec. is "enhanced output". Enhanced outputs are macrocells 1 through 4, 21 through 24, 25 through 28, and 45 through 48. Enhanced macrocells are 5 ns faster than the standard macrocells.

Intel guarantees the specifications of the 5C180 devices listed in the 5C180 Data Sheet. Our Manufacturing group conducts extensive testing of the devices with appropriate guardbands to guarantee all published values under worst case conditions. This testing ensures proper operation across widely divergent applications. Also, every Intel product must pass an extensive qualification program before it is released to the marketplace. Strict quality controls and monitors are applied during the qualification and manufacturing processes.

**Intel 5C180-70 vs. Altera EP1800-2**

**COMBINATORIAL MODE (5C180-70 VS. EP1800-2)**

Parameter	INTEL			ALTERA			Units
	Symbol	Min	Max	Symbol	Min	Max	
I/O pin pad & buffer delay	(1)			t <sub>IO</sub>		5	ns
INPUT pin pad & buffer delay	(1)			t <sub>IN</sub>		10	ns
Logic array delay	(1)			t <sub>LAD</sub>		40	ns
Enhanced logic array delay	(1)			t <sub>LADe</sub>		35	ns
Output buffer and pad delay	(1)			t <sub>OD</sub>		15	ns
Output buffer enable	(1)			t <sub>ZX</sub>		15	ns
Output buffer disable	(1)			t <sub>XZ</sub>		15	ns
Clock Delay (Asynch.)	(1)			t <sub>IC</sub>		40	ns
Enhanced Clock Delay (Asynch.)	(1)			t <sub>ICe</sub>		35	ns
INPUT pin to comb. output (t <sub>IN</sub> + t <sub>LAD</sub> + t <sub>OD</sub> )	t <sub>PD1</sub>		65	t <sub>PD1</sub>		65	ns
INPUT pin to enhanced comb. output (t <sub>IN</sub> + t <sub>LADe</sub> + t <sub>OD</sub> )	t <sub>PD1e</sub>		60	t <sub>PD1e</sub>		60	ns
I/O pin to comb. output (t <sub>IO</sub> + t <sub>IN</sub> + t <sub>LAD</sub> + t <sub>OD</sub> )	t <sub>PD2</sub>		70	t <sub>PD2</sub>		70	ns
I/O pin to enhanced comb. output (t <sub>IO</sub> + t <sub>IN</sub> + t <sub>LADe</sub> + t <sub>OD</sub> )	t <sub>PD2e</sub>		65	t <sub>PD2e</sub>		65	ns
INPUT to output enable (t <sub>IN</sub> + t <sub>LAD</sub> + t <sub>ZX</sub> )	t <sub>PZX1</sub>		65	(2)		65	ns
INPUT pin to enhanced output enable (t <sub>IN</sub> + t <sub>LADe</sub> + t <sub>ZX</sub> )	t <sub>PZX1e</sub>		60	(2)		60	ns
I/O to output enable (t <sub>IO</sub> + t <sub>IN</sub> + t <sub>LAD</sub> + t <sub>ZX</sub> )	t <sub>PZX2</sub>		70	(2)		70	ns
I/O pin to enhanced output enable (t <sub>IO</sub> + t <sub>IN</sub> + t <sub>LADe</sub> + t <sub>ZX</sub> )	t <sub>PZX2e</sub>		65	(2)		65	ns
INPUT to output disable (t <sub>IN</sub> + t <sub>LAD</sub> + t <sub>XZ</sub> )	t <sub>PXZ1</sub>		65	(2)		65	ns
INPUT pin to enhanced output disable (t <sub>IN</sub> + t <sub>LADe</sub> + t <sub>XZ</sub> )	t <sub>PXZ1e</sub>		60	(2)		60	ns
I/O to output disable (t <sub>IO</sub> + t <sub>IN</sub> + t <sub>LAD</sub> + t <sub>XZ</sub> )	t <sub>PXZ2</sub>		70	(2)		70	ns
I/O pin to enhanced output disable (t <sub>IO</sub> + t <sub>IN</sub> + t <sub>LADe</sub> + t <sub>XZ</sub> )	t <sub>PXZ2e</sub>		65	(2)		65	ns
Asynchronous Clear (t <sub>IO</sub> + t <sub>IN</sub> + t <sub>IC</sub> + t <sub>OD</sub> )	t <sub>CLR</sub>		70	(2)		70	ns

**NOTES:**

1. Intel does not spec internal timings of the device.
2. Altera does not spec. in 1988 handbook, these are calculated values based on formula given.

**SYNCHRONOUS MODE (5C180-70 VS. EP1800-2)**

Parameter	INTEL			ALTERA			Units
	Symbol	Min	Max	Symbol	Min	Max	
Internal register setup time	(1)			$t_{SU}$	12		ns
I/O pin pad & buffer delay	(1)			$t_{IO}$		5	ns
INPUT pin pad & buffer delay	(1)			$t_{IN}$		10	ns
OUTPUT buffer and pad delay	(1)			$t_{OD}$		15	ns
Logic array delay	(1)			$t_{LAD}$		40	ns
Enhanced logic array delay	(1)			$t_{LADe}$		35	ns
System clock delay	(1)			$t_{ICS}$		4	ns
Feedback delay	(1)			$t_{FD}$		10	ns
Max. Frequency (no fdbk) (1/[INPUT pin setup to CLKx])	$f_{MAX}$		20.8	$f_{MAX}$		20.8	MHz
Max. Count Frequency (with fdbk) (1/ $t_{CNT}$ )	$f_{CNT}$		16.1	$f_{CNT}$		16.1	MHz
INPUT pin setup to CLKx <sup>(3)</sup> ( $t_{IN} + t_{LAD} + t_{SU} - t_{IN} - t_{ICS}$ )	$t_{SU1}$	48		(2)	48		ns
I/O pin setup to CLKx <sup>(3)</sup> ( $t_{IO} + t_{IN} + t_{LAD} + t_{SU} - t_{IN} - t_{ICS}$ )	$t_{SU2}$	53		(2)	53		ns
INPUT pin setup to CLKx <sup>(4)</sup> ( $t_{IN} + t_{LADe} + t_{SU} - t_{IN} - t_{ICS}$ )	$t_{SU1e}$	43		(2)	43		ns
I/O pin setup to CLKx <sup>(4)</sup> ( $t_{IO} + t_{IN} + t_{LADe} + t_{SU} - t_{IN} - t_{ICS}$ )	$t_{SU2e}$	48		(2)	48		ns
Clock High to Output Valid ( $t_{IN} + t_{ICS} + t_{OD}$ )	$t_{CO}$		28	(2)		29	ns
Register output feedback to register input-internal path ( $t_{FD} + t_{LAD} + t_{SU}$ )	$t_{CNT}$	62		$t_{CNT}$	62		ns
Clock High Time	$t_{CH}$	24		$t_{CH}$	24		ns
Clock Low Time	$t_{CL}$	24		$t_{CL}$	24		ns

**NOTES:**

1. Intel does not spec internal timings on the device.
2. Altera does not spec. in 1988 handbook, these are calculated values based on formula given.
3. For global and standard macrocells.
4. For enhanced macrocells.

**Intel 5C180-75 vs. Altera EP1800-3**
**COMBINATORIAL MODE (5C180-75 VS. EP1800-3)**

Parameter	INTEL			ALTERA			Units
	Symbol	Min	Max	Symbol	Min	Max	
I/O pin pad & buffer delay	(1)			$t_{IO}$		5	ns
INPUT pin pad & buffer delay	(1)			$t_{IN}$		12	ns
Logic array delay	(1)			$t_{LAD}$		44	ns
Enhanced logic array delay	(1)			$t_{LADe}$		39	ns
Output buffer and pad delay	(1)			$t_{OD}$		19	ns
Output buffer enable	(1)			$t_{ZX}$		19	ns
Output buffer disable	(1)			$t_{XZ}$		19	ns
Clock Delay (Asynch.)	(1)			$t_{IC}$		44	ns
Enhanced Clock Delay (Asynch.)	(1)			$t_{ICe}$		39	ns
INPUT pin to comb. output ( $t_{IN} + t_{LAD} + t_{OD}$ )	$t_{PD1}$		70	$t_{PD1}$		75	ns
INPUT pin to enhanced comb. output ( $t_{IN} + t_{LADe} + t_{OD}$ )	$t_{PD1e}$		65	$t_{PD1e}$		70	ns
I/O pin to comb. output ( $t_{IO} + t_{IN} + t_{LAD} + t_{OD}$ )	$t_{PD2}$		75	$t_{PD2}$		80	ns
I/O pin to enhanced comb. output ( $t_{IO} + t_{IN} + t_{LADe} + t_{OD}$ )	$t_{PD2e}$		70	$t_{PD2e}$		75	ns
INPUT to output enable ( $t_{IN} + t_{LAD} + t_{ZX}$ )	$t_{PZX1}$		70	(2)		75	ns
INPUT pin to enhanced output enable ( $t_{IN} + t_{LADe} + t_{ZX}$ )	$t_{PZX1e}$		65	(2)		70	ns
I/O to output enable ( $t_{IO} + t_{IN} + t_{LAD} + t_{ZX}$ )	$t_{PZX2}$		75	(2)		80	ns
I/O pin to enhanced output enable ( $t_{IO} + t_{IN} + t_{LADe} + t_{ZX}$ )	$t_{PZX2e}$		70	(2)		75	ns
INPUT to output disable ( $t_{IN} + t_{LAD} + t_{XZ}$ )	$t_{PXZ1}$		70	(2)		75	ns
INPUT pin to enhanced output disable ( $t_{IN} + t_{LADe} + t_{XZ}$ )	$t_{PXZ1e}$		65	(2)		70	ns
I/O to output disable ( $t_{IO} + t_{IN} + t_{LAD} + t_{XZ}$ )	$t_{PXZ2}$		75	(2)		80	ns
I/O pin to enhanced output disable ( $t_{IO} + t_{IN} + t_{LADe} + t_{XZ}$ )	$t_{PXZ2e}$		70	(2)		75	ns
Asynchronous Clear ( $t_{IO} + t_{IN} + t_{IC} + t_{OD}$ )	$t_{CLR}$		75	(2)		80	ns

**NOTES:**

- Intel does not spec internal timings of the device.
- Altera does not spec. in 1988 handbook, these are calculated values based on formula given.

**SYNCHRONOUS MODE (5C180-75 VS. EP1800-3)**

Parameter	INTEL			ALTERA			Units
	Symbol	Min	Max	Symbol	Min	Max	
Internal register setup time	(1)			$t_{SU}$	14		ns
I/O pin pad & buffer delay	(1)			$t_{IO}$		5	ns
INPUT pin pad & buffer delay	(1)			$t_{IN}$		12	ns
OUTPUT buffer and pad delay	(1)			$t_{OD}$		19	ns
Logic array delay	(1)			$t_{LAD}$		44	ns
Enhanced logic array delay	(1)			$t_{LADe}$		39	ns
System clock delay	(1)			$t_{ICS}$		4	ns
Feedback delay	(1)			$t_{FD}$		14	ns
Max. Frequency (no fdbk) (1/[INPUT pin setup to CLKx])	$f_{MAX}$		19.6	$f_{MAX}$		18.5	MHz
Max. Count Frequency (with fdbk) (1/ $t_{CNT}$ )	$f_{CNT}$		15.1	$f_{CNT}$		13.8	MHz
INPUT pin setup to CLKx <sup>(3)</sup> ( $t_{IN} + t_{LAD} + t_{SU} - t_{IN} - t_{ICS}$ )	$t_{SU1}$	51		(2)	54		ns
I/O pin setup to CLKx <sup>(3)</sup> ( $t_{IO} + t_{IN} + t_{LAD} + t_{SU} - t_{IN} - t_{ICS}$ )	$t_{SU2}$	56		(2)	59		ns
INPUT pin setup to CLKx <sup>(4)</sup> ( $t_{IN} + t_{LADe} + t_{SU} - t_{IN} - t_{ICS}$ )	$t_{SU1e}$	46		(2)	49		ns
I/O pin setup to CLKx <sup>(4)</sup> ( $t_{IO} + t_{IN} + t_{LADe} + t_{SU} - t_{IN} - t_{ICS}$ )	$t_{SU2e}$	51		(2)	54		ns
Clock High to Output Valid ( $t_{IN} + t_{ICS} + t_{OD}$ )	$t_{CO}$		30	(2)		35	ns
Register output feedback to register input-internal path ( $t_{FD} + t_{LAD} + t_{SU}$ )	$t_{CNT}$	66		$t_{CNT}$	72		ns
Clock High Time	$t_{CH}$	25		$t_{CH}$	27		ns
Clock Low Time	$t_{CL}$	25		$t_{CL}$	27		ns

**NOTES:**

1. Intel does not spec internal timings on the device.
2. Altera does not spec. in 1988 handbook, these are calculated values based on formula given.
3. For global and standard macrocells.
4. For enhanced macrocells.

**Intel 5C180-90 vs. Altera EP1800**
**COMBINATORIAL MODE (5C180-90 VS. EP1800)**

Parameter	INTEL			ALTERA			Units
	Symbol	Min	Max	Symbol	Min	Max	
I/O pin pad & buffer delay	(1)			$t_{IO}$		5	ns
INPUT pin pad & buffer delay	(1)			$t_{IN}$		14	ns
Logic array delay	(1)			$t_{LAD}$		48	ns
Enhanced logic array delay	(1)			$t_{LADe}$		43	ns
Output buffer and pad delay	(1)			$t_{OD}$		23	ns
Output buffer enable	(1)			$t_{ZX}$		23	ns
Output buffer disable	(1)			$t_{XZ}$		23	ns
Clock Delay (Asynch.)	(1)			$t_{IC}$		48	ns
Enhanced Clock Delay (Asynch.)	(1)			$t_{ICe}$		43	ns
INPUT pin to comb. output ( $t_{IN} + t_{LAD} + t_{OD}$ )	$t_{PD1}$		85	$t_{PD1}$		85	ns
INPUT pin to enhanced comb. output ( $t_{IN} + t_{LADe} + t_{OD}$ )	$t_{PD1e}$		80	$t_{PD1e}$		80	ns
I/O pin to comb. output ( $t_{IO} + t_{IN} + t_{LAD} + t_{OD}$ )	$t_{PD2}$		90	$t_{PD2}$		90	ns
I/O pin to enhanced comb. output ( $t_{IO} + t_{IN} + t_{LADe} + t_{OD}$ )	$t_{PD2e}$		85	$t_{PD2e}$		85	ns
INPUT to output enable ( $t_{IN} + t_{LAD} + t_{ZX}$ )	$t_{PZX1}$		85	(2)		85	ns
INPUT pin to enhanced output enable ( $t_{IN} + t_{LADe} + t_{ZX}$ )	$t_{PZX1e}$		80	(2)		80	ns
I/O to output enable ( $t_{IO} + t_{IN} + t_{LAD} + t_{ZX}$ )	$t_{PZX2}$		90	(2)		90	ns
I/O pin to enhanced output enable ( $t_{IO} + t_{IN} + t_{LADe} + t_{ZX}$ )	$t_{PZX2e}$		85	(2)		85	ns
INPUT to output disable ( $t_{IN} + t_{LAD} + t_{XZ}$ )	$t_{PXZ1}$		85	(2)		85	ns
INPUT pin to enhanced output disable ( $t_{IN} + t_{LADe} + t_{XZ}$ )	$t_{PXZ1e}$		80	(2)		80	ns
I/O to output disable ( $t_{IO} + t_{IN} + t_{LAD} + t_{XZ}$ )	$t_{PXZ2}$		90	(2)		90	ns
I/O pin to enhanced output disable ( $t_{IO} + t_{IN} + t_{LADe} + t_{XZ}$ )	$t_{PXZ2e}$		85	(2)		85	ns
Asynchronous Clear ( $t_{IO} + t_{IN} + t_{IC} + t_{OD}$ )	$t_{CLR}$		90	(2)		90	ns

**NOTES:**

1. Intel does not spec internal timings of the device.
2. Altera does not spec. in 1988 handbook, these are calculated values based on formula given.

**SYNCHRONOUS MODE (5C180-90 VS. EP1800)**

Parameter	INTEL			ALTERA			Units
	Symbol	Min	Max	Symbol	Min	Max	
Internal register setup time	(1)			t <sub>SU</sub>	18		ns
I/O pin pad & buffer delay	(1)			t <sub>IO</sub>	5		ns
INPUT pin pad & buffer delay	(1)			t <sub>IN</sub>	14		ns
OUTPUT buffer and pad delay	(1)			t <sub>OD</sub>	23		ns
Logic array delay	(1)			t <sub>LAD</sub>	48		ns
Enhanced logic array delay	(1)			t <sub>LADe</sub>	43		ns
System clock delay	(1)			t <sub>ICS</sub>	4		ns
Feedback delay	(1)			t <sub>FD</sub>	16		ns
Max. Frequency (no fdbk) (1/[INPUT pin setup to CLKx])	f <sub>MAX</sub>		16.1	f <sub>MAX</sub>		16.1	MHz
Max. Count Frequency (with fdbk) (1/t <sub>CNT</sub> )	f <sub>CNT</sub>		12.2	f <sub>CNT</sub>		12.2	MHz
INPUT pin setup to CLKx <sup>(3)</sup> (t <sub>IN</sub> + t <sub>LAD</sub> + t <sub>SU</sub> - t <sub>IN</sub> - t <sub>ICS</sub> )	t <sub>SU1</sub>	62		(2)	62		ns
I/O pin setup to CLKx <sup>(3)</sup> (t <sub>IO</sub> + t <sub>IN</sub> + t <sub>LAD</sub> + t <sub>SU</sub> - t <sub>IN</sub> - t <sub>ICS</sub> )	t <sub>SU2</sub>	67		(2)	67		ns
INPUT pin setup to CLKx <sup>(4)</sup> (t <sub>IN</sub> + t <sub>LADe</sub> + t <sub>SU</sub> - t <sub>IN</sub> - t <sub>ICS</sub> )	t <sub>SU1e</sub>	57		(2)	57		ns
I/O pin setup to CLKx <sup>(4)</sup> (t <sub>IO</sub> + t <sub>IN</sub> + t <sub>LADe</sub> + t <sub>SU</sub> - t <sub>IN</sub> - t <sub>ICS</sub> )	t <sub>SU2e</sub>	62		(2)	62		ns
Clock High to Output Valid (t <sub>IN</sub> + t <sub>ICS</sub> + t <sub>OD</sub> )	t <sub>CO</sub>		35	(2)		41	ns
Register output feedback to register input-internal path (t <sub>FD</sub> + t <sub>LAD</sub> + t <sub>SU</sub> )	t <sub>CNT</sub>	82		t <sub>CNT</sub>	82		ns
Clock High Time	t <sub>CH</sub>	30		t <sub>CH</sub>	30		ns
Clock Low Time	t <sub>CL</sub>	30		t <sub>CL</sub>	30		ns

**NOTES:**

1. Intel does not spec internal timings on the device.
2. Altera does not spec. in 1988 handbook, these are calculated values based on formula given.
3. For global and standard macrocells.
4. For enhanced macrocells.

## Techniques for Modular EPLD Logic Design

Lawrence Palley  
PLDO Product Marketing Manager  
Intel Corporation  
151 Blue Ravine Road  
Folsom, CA 95630

### INTRODUCTION

Advances in both programmable logic devices and the tools used to configure them now enable new design techniques for custom logic applications. New high capacity flexible architected EPLDs (erasable and electrically programmable logic devices) allow for complete single chip integration of one or more logic configurations. Additionally, development tools make use of these capabilities by providing alternatives for design input, high speed logic compilation and minimization, heuristic logic fitting into EPLD devices, and superior reporting documentation. Designers can take advantage of these advances with a new Modular EPLD logic design (MELD) technique, to accelerate their product development.

### ADVANCES IN EPLDs

Traditional PLDs relied on Boolean equation entry and compilation methods for combinatorial function implementation. The primary applications were as SSI/MSI replacements for implementing decode "glue" in microprocessor based systems. PLDs came in bipolar versions with total logic content under 400 gates of equivalent logic. Tools to develop the programmable logic implementation of a function didn't require a high degree of sophistication - the devices for which they were optimizing designs had relatively little logic and little logic flexibility.

Newer EPLDs incorporate several features which broaden their application base. Besides their low power CMOS technology, they incorporate individually configurable register and I/O logic for each macrocell. Devices such as the 5C060 incorporate 16 macrocells with registers programmable into D, or JK configurations. Each register is also programmably configured to be clocked by synchronous or asynchronous clocks. Additionally, outputs and feedback paths for each pin can be combinatorial or registered. The combination of this level of flexibility and gate counts of some devices exceeding 1200, EPLDs have moved programmable logic well past simple combinatorial functions.

To make optimum use of the new EPLD device technology, design tools needed to improve to allow more freedom of design input, better

logic optimization for maximum device utilization, and improved reporting documentation. Intel's programmable logic development system provides these improvements. Input methods include the choice of (and combination of) schematic, netlist, state machine, or Boolean entry. Besides Boolean equation minimization, the optimizer program optimally matches I/O and register resources required by the design with what's available in EPLD devices. It then reports on how the logic entry was reduced, which resources were required, and how the design was placed in a given device. Resources still available in the devices or not able to fit in to the device are also documented.

### MELD

Making use of both the advances in EPLD devices and their development tools, engineers can now design hardware (logic) in much the same way as software is developed. This new design technique called MELD (Modular EPLD Logic Design) is shown in Figure 1. Design Entry, in any of the typical engineering formats, is entered on a development station (in this case a personal computer). Using EPLD development system software, the design is then compiled for EPLD implementation. Object code or (in the case of an EPLD) a JEDEC 1's and 0's file are the result. The unique capability of EPLDs is to test a part of a partitioned design in silicon, erase the EPLD, test the next design, and finally to merge the designs together. This powerful logic design methodology allows for the partitioning of a complex logic function into smaller sub-functions that can be individually designed and debugged using the design tools and the erasability feature of EPLDs. After the individual modules are proved to be functional as desired, they can be combined on the same EPLD, allowing for higher integration and its attendant benefits.



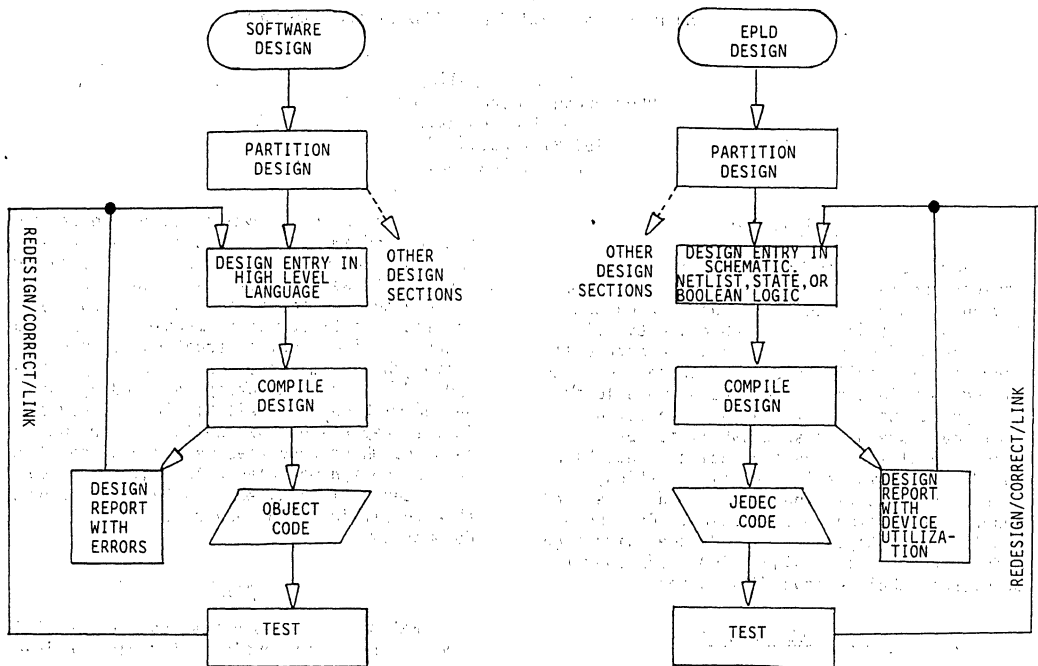


Figure 1. EPLD design process compared with software design process

NOTE: The MELD technique would involve this design process shown above to be implemented for different sub-modules and combining of the sub-functions into the completed high-integration EPLD design.

#### A MELD Example:

This Modular EPLD logic design (MELD) methodology is now illustrated with an example. The example shown here is a design which implements a BCD-counter which is encoded into a seven-segment display.

Figure 2 shows the design of a BCD-counter designed using state machine entry. This design was compiled (Figure 3) and individually tested in-circuit.

Figure 4 shows a design for implementing the seven-segment display shown in Figure 5. It uses Boolean design methods, although not yet optimized. This design has been tested out in several previous designs. An additional section called "LINK EQUATIONS" is now used to connect the BCD-counter with the seven-segment display.

The two design files, BCD-Counter and SEGEQS, are now compiled together in the LOC (Logic Optimizing Compiler) of the Intel Programmable Logic Development System (Figure 6) to yield the combined file, BCD-Counter, of Figure 7.

When implemented in the 5C121 EPLD, the utilization report of Figure 8 results: It shows a pinout designated by the compiler, the routing of inputs, the source of outputs, unused device resources, and some figure of merit about how the design used 5C121 resources. This data can be used to test the device to as feedback for new design inputs. An example of such an input would be to assign signals to 5C121 pins so that PCB layout is simple.

FIGURE 2

LSP  
 INTEL  
 February 7, 1986  
 0  
 0  
 5C121  
 BCD COUNTER  
 LB Version 3.0, Baseline 17x, 9/26/85  
 PART: 5C121  
 INPUTS: CLK,ENABLE,RESET  
 OUTPUTS: BCD0,BCD1,BCD2,BCD3  
 MACHINE:BCD\_COUNTER

CLOCK: CLK

STATES: [BCD3 BCD2 BCD1 BCD0]

S0	[	0	0	0	0	]
S1	[	0	0	0	1	]
S2	[	0	0	1	0	]
S3	[	0	0	1	1	]
S4	[	0	1	0	0	]
S5	[	0	1	0	1	]
S6	[	0	1	1	0	]
S7	[	0	1	1	1	]
S8	[	1	0	0	0	]
S9	[	1	0	0	1	]

%TRANSITIONS%

S0:  
 IF RESET THEN S0  
 IF ENABLE THEN S1

S1:  
 IF RESET THEN S0  
 IF ENABLE THEN S2

S2:  
 IF RESET THEN S0  
 IF ENABLE THEN S3

S3:  
 IF RESET THEN S0  
 IF ENABLE THEN S4

S4:  
 IF RESET THEN S0  
 IF ENABLE THEN S5

S5:  
 IF RESET THEN S0  
 IF ENABLE THEN S6

S6:  
 IF RESET THEN S0  
 IF ENABLE THEN S7

S7:  
 IF RESET THEN S0  
 IF ENABLE THEN S8

S8:  
 IF RESET THEN S0  
 IF ENABLE THEN S9

S9:  
 IF RESET THEN S0  
 IF ENABLE THEN S0

END\$

FIGURE 3

```

LSP
INTEL
February 7, 1986
0
0
5C121
BCD COUNTER
LB Version 3.0, Baseline 17x, 9/26/85SMV Version 1.0 Baseline 1.3 85/12/13 00:12:5

PART: 5C121

INPUTS:
CLK, ENABLE, RESET

OUTPUTS:
BCD0, BCD1, BCD2, BCD3

NETWORK:
CLK = INP(CLK)
ENABLE = INP(ENABLE)
RESET = INP(RESET)
%
I/O's for State Machine "BCD_COUNTER"
%
BCD3, BCD3 = RORF(BCD3.d, CLK, GND, GND, VCC)
BCD2, BCD2 = RORF(BCD2.d, CLK, GND, GND, VCC)
BCD1, BCD1 = RORF(BCD1.d, CLK, GND, GND, VCC)
BCD0, BCD0 = RORF(BCD0.d, CLK, GND, GND, VCC)

EQUATIONS:
%
Boolean Equations for State Machine "BCD_COUNTER"
%
%
Current State Equations for "BCD_COUNTER"
%
S0 = BCD3'*BCD2'*BCD1'*BCD0';
S1 = BCD3'*BCD2'*BCD1'*BCD0;
S2 = BCD3'*BCD2'*BCD1*BCD0';
S3 = BCD3'*BCD2'*BCD1*BCD0;
S4 = BCD3'*BCD2*BCD1'*BCD0';
S5 = BCD3'*BCD2*BCD1'*BCD0;
S6 = BCD3'*BCD2*BCD1*BCD0';
S7 = BCD3'*BCD2*BCD1*BCD0;
S8 = BCD3*BCD2'*BCD1'*BCD0';
S9 = BCD3*BCD2'*BCD1'*BCD0;
%
SV Defining Equations for State Machine "BCD_COUNTER"
%
BCD3.d = S8.n
        + S9.n;
BCD2.d = S4.n
        + S5.n
        + S6.n
        + S7.n;
BCD1.d = S2.n
        + S3.n
        + S6.n
        + S7.n;
BCD0.d = S1.n
        + S3.n
        + S5.n
        + S7.n
        + S9.n;
%

```

FIGURE 3 (CONTINUED)

Next State Equations for State Machine "BCD\_COUNTER"

```

%
S1.n = S1 * (ENABLE)' * (RESET)'
      + S0 * ENABLE * (RESET)';
S2.n = S2 * (ENABLE)' * (RESET)'
      + S1 * ENABLE * (RESET)';
S3.n = S3 * (ENABLE)' * (RESET)'
      + S2 * ENABLE * (RESET)';
S4.n = S4 * (ENABLE)' * (RESET)'
      + S3 * ENABLE * (RESET)';
S5.n = S5 * (ENABLE)' * (RESET)'
      + S4 * ENABLE * (RESET)';
S6.n = S6 * (ENABLE)' * (RESET)'
      + S5 * ENABLE * (RESET)';
S7.n = S7 * (ENABLE)' * (RESET)'
      + S6 * ENABLE * (RESET)';
S8.n = S8 * (ENABLE)' * (RESET)'
      + S7 * ENABLE * (RESET)';
S9.n = S9 * (ENABLE)' * (RESET)'
      + S8 * ENABLE * (RESET)';

```

END\$

LSP  
INTEL  
February 7, 1986  
0  
0

FIGURE 4

5C121  
SEVEN SEGMENT DECODERS FOR BCD COUNTER  
LB Version 3.0, Baseline 17x, 9/26/85  
PART: 5C121  
INPUTS:  
OUTPUTS: SEGA, SEGB, SEGC, SEGD, SEGE, SEGF, SEGG  
NETWORK:  
SEGA = CONF (SEGA, VCC)  
SEGB = CONF (SEGB, VCC)  
SEGC = CONF (SEGC, VCC)  
SEGD = CONF (SEGD, VCC)  
SEGE = CONF (SEGE, VCC)  
SEGF = CONF (SEGF, VCC)  
SEGG = CONF (SEGG, VCC)  
EQUATIONS:  
SEGA = 0 + 2 + 3 + 5 + 7 + 8 + 9;  
  
SEGB = 0 + 1 + 2 + 3 + 4 + 6 + 7 + 8 + 9;  
  
SEGC = 0 + 1 + 3 + 4 + 5 + 6 + 7 + 8 + 9;  
  
SEGD = 0 + 2 + 3 + 5 + 6 + 8;  
SEGE = 0 + 2 + 6 + 8;  
SEGF = 0 + 4 + 5 + 6 + 8 + 9;  
SEGG = 2 + 3 + 4 + 5 + 6 + 8 + 9;  
  
0 = /D3\*/D2\*/D1\*/D0;  
1 = /D3\*/D2\*/D1\*D0;  
2 = /D3\*/D2\* D1\*/D0;  
3 = /D3\*/D2\* D1\*D0;  
4 = /D3\* D2\*/D1\*/D0;  
5 = /D3\* D2\*/D1\*D0;  
6 = /D3\* D2\* D1\*/D0;  
7 = /D3\* D2\* D1\*D0;  
8 = D3\*/D2\*/D1\*/D0;  
9 = D3\*/D2\*/D1\*D0;  
%LINK EQUATIONS %  
  
D0 = BCD0;  
D1 = BCD1;  
D2 = BCD2;  
D3 = BCD3;  
  
END\$

FIGURE 6  
Intel Programmable Logic Software

LOC Menu  
F1 Help  
F2 iPLS Menu  
F3 Input Format           ADF  
F4 File Name             A:BCD A:SEGEQS  
F5 Minimization         Yes  
F6 Inversion Control    No  
F7 LEF Analysis         Yes

FIGURE 7

LSP  
INTEL

February 7, 1986

0  
0  
5C121  
BCD COUNTER  
LB Version 3.0, Baseline 17x, 9/26/85SMV Version 1.0 Baseline 1.3 85/12/13 00:12:5

PART:

5C121

INPUTS:

CLK, ENABLE, RESET

OUTPUTS:

BCD0, BCD1, BCD2, BCD3, SEGA, SEGB, SEGC, SEGD, SEGE, SEGF, SEGG

NETWORK:

CLK = INP(CLK)  
ENABLE = INP(ENABLE)  
RESET = INP(RESET)  
BCD0, BCD0 = RORF(BCD0.d, CLK, GND, GND, VCC)  
BCD1, BCD1 = RORF(BCD1.d, CLK, GND, GND, VCC)  
BCD2, BCD2 = RORF(BCD2.d, CLK, GND, GND, VCC)  
BCD3, BCD3 = RORF(BCD3.d, CLK, GND, GND, VCC)  
SEGA = CONF(SEGA, VCC)  
SEGB = CONF(SEGB, VCC)  
SEGC = CONF(SEGC, VCC)  
SEGD = CONF(SEGD, VCC)  
SEGE = CONF(SEGE, VCC)  
SEGF = CONF(SEGF, VCC)  
SEGG = CONF(SEGG, VCC)

EQUATIONS:

SEGG = BCD1 \* BCD3' \* BCD2'  
+ BCD1' \* BCD3' \* BCD2  
+ BCD1' \* BCD3 \* BCD2'  
+ BCD1 \* BCD3' \* BCD0';

SEGF = BCD3' \* BCD1' \* BCD0'  
+ BCD3' \* BCD2 \* BCD1'  
+ BCD3 \* BCD2' \* BCD1'  
+ BCD3' \* BCD2 \* BCD0';

SEGE = BCD2' \* BCD1' \* BCD0'  
+ BCD3' \* BCD1 \* BCD0';

SEGD = BCD2' \* BCD1' \* BCD0'  
+ BCD3' \* BCD2' \* BCD1  
+ BCD3' \* BCD1 \* BCD0'  
+ BCD3' \* BCD2 \* BCD1' \* BCD0;

SEGC = BCD2' \* BCD1'  
+ BCD3' \* BCD2  
+ BCD3' \* BCD0;

SEGB = BCD2' \* BCD1'  
+ BCD3' \* BCD0'  
+ BCD3' \* BCD1;

SEGA = BCD3' \* BCD2' \* BCD0'  
+ BCD3 \* BCD2' \* BCD1'  
+ BCD3' \* BCD2' \* BCD1  
+ BCD3' \* BCD2 \* BCD0;

BCD3.d = BCD3 \* BCD2' \* BCD1' \* BCD0' \* RESET'  
+ BCD3 \* BCD2' \* BCD1' \* ENABLE' \* RESET'  
+ BCD3' \* BCD2 \* BCD1 \* BCD0 \* ENABLE \* RESET';

FIGURE 7 (CONTINUED)

```

BCD2.d = BCD3' * BCD2 * BCD0' * RESET'
+ BCD3' * BCD2 * BCD1' * RESET'
+ BCD3' * BCD2 * ENABLE' * RESET'
+ BCD3' * BCD2' * BCD1 * BCD0 * ENABLE * RESET';

BCD1.d = BCD3' * BCD1 * BCD0' * RESET'
+ BCD3' * BCD1 * ENABLE' * RESET'
+ BCD3' * BCD1' * BCD0 * ENABLE * RESET';

BCD0.d = BCD3' * BCD0' * ENABLE * RESET'
+ BCD3' * BCD0 * ENABLE' * RESET'
+ BCD2' * BCD1' * BCD0 * ENABLE * RESET'
+ BCD2' * BCD1' * BCD0' * ENABLE * RESET';

```

END\$

FIGURE 8

Logic Optimizing Compiler Utilization Report

\*\*\*\*\* Design implemented successfully

LSP

INTEL

February 7, 1986

0

0

5C121

BCD COUNTER

LB Version 3.0, Baseline 17x, 9/26/85SMV Version 1.0 Baseline 1.3 85/12/13 00:12:5

5C121

```

--- --
CLK -: 1 40:- Vcc
GND -: 2 39:- Vcc
GND -: 3 38:- ENABLE
GND -: 4 37:- RESET
GND -: 5 36:- GND
GND -: 6 35:- GND
GND -: 7 34:- GND
SEGD -: 8 33:- GND
RESERVED -: 9 32:- SEGG
RESERVED -:10 31:- RESERVED
RESERVED -:11 30:- RESERVED
SEGA -:12 29:- SEGC
RESERVED -:13 28:- SEGB
RESERVED -:14 27:- RESERVED
SEGE -:15 26:- RESERVED
RESERVED -:16 25:- SEGF
BCD2 -:17 24:- RESERVED
RESERVED -:18 23:- BCD3
RESERVED -:19 22:- BCD1
GND -:20 21:- BCD0
--- --

```

FIGURE 8 (CONTINUED)

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

Name	Pin	Resource	MCell #	PTerms	MCells	Feeds:		
						OE	Clear	Clock
CLK	1	INP	-	-	-	-	-	Reg
RESET	37	INP	-	-	10 11 12 19	-	-	-
ENABLE	38	INP	-	-	10 11 12 19	-	-	-

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

Name	Pin	Resource	MCell #	PTerms	MCells	Feeds:		
						OE	Clear	
SEGD	8	CONF	28	4/ 4	-	-	-	
SEGA	12	CONF	24	4/ 6	-	-	-	
SEGE	15	CONF	21	2/ 4	-	-	-	
BCD2	17	RORF	19	4/ 4	1 4 5 8 10 12 19 21 24 28	-	-	
BCD0	21	RORF	12	4/ 8	1 4 5 8 10 11 12 19 21 24 28	-	-	
BCD1	22	RORF	11	3/ 8	1 4 5 8 10 11 12 19 21 24 28	-	-	



FIGURE 8 (CONTINUED)

BCD3	23	RORF	10	3/ 4	1	-	-
					4		
					5		
					8		
					10		
					11		
					12		
					19		
					21		
					24		
					28		
SEGF	25	CONF	8	4/ 4	-	-	-
SEGB	28	CONF	5	3/ 6	-	-	-
SEGC	29	CONF	4	3/ 6	-	-	-
SEGG	32	CONF	1	4/ 4	-	-	-

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

Name	Pin	Resource	MCell	PTerms
-	2	-	-	-
-	3	-	-	-
-	4	-	-	-
-	5	-	-	-
-	6	-	-	-
-	7	-	-	-
-	9	-	27	10
-	10	-	26	8
-	11	-	25	6
-	13	-	23	8
-	14	-	22	10
-	16	-	20	12
-	18	-	18	8
-	19	-	17	8
-	24	-	9	12
-	26	-	7	10
-	27	-	6	8
-	30	-	3	8
-	31	-	2	10
-	33	-	-	-
-	34	-	-	-
-	35	-	-	-
-	36	-	-	-
-	NA	-	13	8
-	NA	-	14	8
-	NA	-	15	8
-	NA	-	16	8

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

37% Pins  
 39% MacroCells  
 18% Pterms

CONCLUSIONS

The complete design took less than an hour to enter, compile, and link with EPLDs. The ability to partition designs, then individually implement those designs in the logic design

entry of choice, and finally to link designs together is a new design method only available with advances in programmable logic and their design tools. By taking advantage of these capabilities, designers can bring logic implementations to market faster and with a high degree of integration.

# Crosspoint Switch: A PLD Approach

by Jim Donnell, Intel Corp.

**E**rasable programmable logic devices (EPLDs) combine the gate densities of low-end gate arrays with the short development time and low cost of EPROMs. This merging of technologies produces a device with features suited to a wide range of digital applications. In contrast to the long development times (and higher costs) for gate arrays, EPLDs require minimal frontend design time. In just a few hours, EPLD designs can be developed, modified and verified. Also, core elements from one EPLD design can be incorporated into new designs as quickly as standard software subroutines from one program can be modified and used in other programs.

The design of a digital crosspoint switch using an Intel 5C121 EPLD illustrates these features. *Digital Design* implemented a crosspoint switch in a gate array last year (see *Digital Design*, January through March, 1985). Applications that require a data transfer from one of several inputs to one of several outputs frequently use a digital crosspoint switch. Using the 5C121 EPLD, Intel Corp. (Santa Clara, CA) designed three different configurations of a crosspoint switch.

Offered in a 40-pin package that provides up to 36 inputs or 24 outputs, the 5C121 supports up to 28 macrocells (including four buried registers) and 236 product terms (p-terms). Logic density in the 5C121 is the equivalent of 1,200 usable NAND gates. Maximum power requirements are 100 mA active and 30 mA standby with TTL input levels. With CMOS input levels, a 5C121 requires 50 mA active and 3 mA standby.

Two major parameters determine the complexity and configuration of a digital crosspoint switch: the number of possible switching locations for each bit (inputs and outputs), and the number of bits transferred in one clock pulse (word width). The availability of I/O pins, macrocells and p-terms for a given EPLD

device dictates the number of switches that can be designed into a single device.

## Configuration 1

The first circuit (Figure 1) considered is a digital crosspoint switch with eight inputs and a 3-bit word width. This switch transfers a 3-bit word coming from one of eight sources to a particular output. The number of devices "OR-tied" to each output pin determines the number of outputs. Selecting one of eight data inputs from each of the three channels (A0 to A7, B0 to B7 and C0 to C7), the switch routes that data to a single output (QA, QB and QC). Each output can be OR-tied to more than one

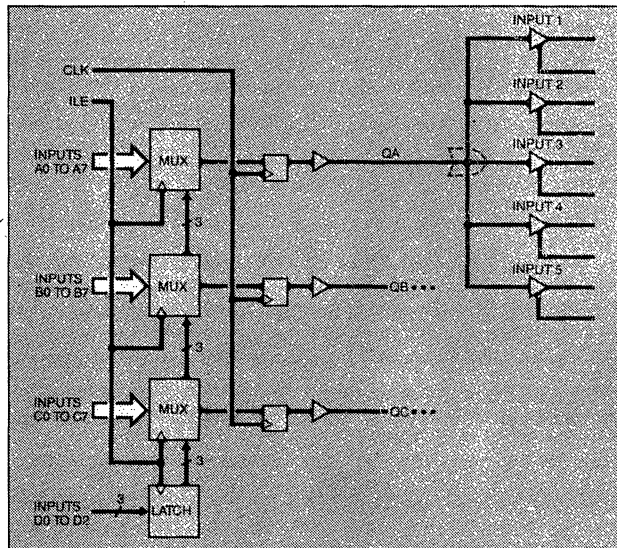


Figure 1: Configuration 1 uses a three-channel eight-to-one multiplexer circuit with latching inputs. Each output can drive multiple, individually selected inputs to complete the digital crosspoint switch. By connecting inputs to the EPLD outputs in an "OR-tied" configuration, with only one input enabled at any time, the multiplexer circuit becomes a crosspoint switch.

three-state input to complete the switch (only one input can be enabled at a time). Three additional control bits (D0 to D2) select one of the eight different inputs. All three channels operate in parallel. Separate input and output clocks allow a high data rate and relax input set-up and hold times. Input data for all three channels, along with the three select bits, are latched by ILE. Data at the inputs can change state after being latched and data is clocked out of the switch by CLK.

Equation 1 shows the Boolean expression for a single channel in the sum-of-products form. (See Table 1 for all equations.) The Boolean expression for the remaining two channels is similar: the designer need only change the A in the equations to a B or C.

**Timing Analysis**

The internal delay paths determine the circuit's maximum operating frequency (fmax). In this configuration there is an input delay (Tin), an array delay (Tad), a register delay (Trd) and an output delay (Tod). The fmax is a function of the signals that must settle at the input of the output register before the rising edge of the clock. In this case, signals propagate only through the input latches and the array. Therefore, the data must be valid at the inputs Tin + Tad just nanoseconds before the rising edge of the internal clock signal (CLK). However, because of the inherent delay of the CLK signal, this reference must be shifted to the rising edge of the external clock signal by subtracting the internal clock delay (Tic). The external data set-up time (Tsu) is shown in Equation 2. Inverting this time requirement yields the maximum operating frequency.

As the output flip-flops are clocked, data propagates through the register to the output pin. With reference to the external clock pin, data becomes valid at the outputs Tic + Trd + Tod nanoseconds after the rising edge of the clock. Figure 2 shows the timing requirements for this circuit, including the input latch signal.

Using a 5C121-50 (50-nsec propagation delay), data can be sent through this switch configuration at 25 Mbits/sec. This transfer rate remains independent of the word width. Since one 5C121 EPLD in this configuration can simultaneously transfer three bits of information, three 5C121's are required to transfer a byte of data during each clock cycle. This configuration of a digital crosspoint switch uses 86% of the 40 pins, 71% of the macrocells and 11% of the available p-terms in the 5C121 EPLD.

**Configuration 2**

The second circuit (Figure 3) also selects one of eight inputs (I0 to I7), but this time data is routed to one of eight different

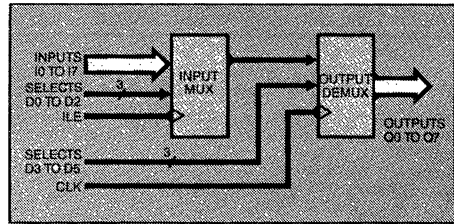


Figure 3: Configuration 2 uses a single-bit eight-input/eight-output digital crosspoint switch. Designers can implement this for either optimal package count (see Figure 4) or for optimal speed (see Figure 5).

outputs (Q0 to Q7). Six control bits are required for each transfer: three to select the input path (D0 to D2); three to select the output path (D3 to D5). By selecting a single output path and clocking all output registers simultaneously, deselected outputs are automatically cleared. This is useful for designs where only the most current data is needed. Equation 4 is the common equation to select one of eight input paths. Equations 5 to 12 complete the Boolean equations for this example.

The previous equations would contain eight product terms if they were written in expanded form. However, by treating SELECTEQ as one signal, each equation contains only one product term. Both options are available in the 5C121. But, there

**In contrast to the long development times for gate arrays, EPLDs require minimal frontend design time.**

are advantages and disadvantages to the two methods. If SELECTEQ is implemented as one signal through a combinational feedback option, one and one-half crosspoint switches can be implemented in one 5C121 (Figure 4). The trade-off is faster speed for low chip count. By design, only 18 macrocells in the 5C121 can support eight product terms. On the other hand, selecting the combinational option reduces the p-terms but introduces an additional input mux delay.

Figure 4 shows that an input signal must pass through four delays before reaching the input to the flip-flop. Again, subtracting the input clock delay to shift the reference point yields Equation 13 for the set-up time. Inverting Tsu gives the maximum operating frequency. In this configuration, data can be clocked through at 12 Mbits/sec. This layout utilizes 97% of the available pins, 89% of the available macrocells and 13% of the product terms. Six 5C121s would be required to implement a byte-wide switch with this layout.

If the combinational feedback option is not used, there are eight output equations, each containing eight product terms. Assigning these equations to the macrocells that support eight p-terms shows that only a single, one-of-eight select line digital crosspoint switch fits into one 5C121. Thus, the design requires eight 5C121s to complete a byte-wide

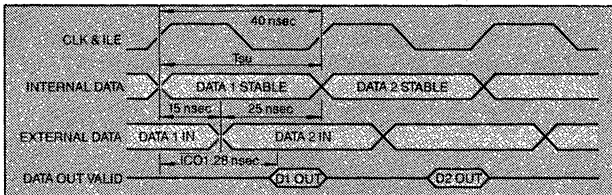


Figure 2: A 40-nsec internal set-up time (prior to clocking data through the output flip-flop) marks Configuration 1. Data clocked into all eight input latches at the rising edge of one ILE/CLK cycle is selected and clocked out of the output flip-flop on the next rising edge of ILE/CLK.

parallel transfer. Since the signal paths are identical to Configuration 1, the same timing analysis applies here.

This layout (Figure 5) utilizes 65% of the pins, 39% of the macrocells and 30% of the p-terms. Though the utilization numbers are lower for this example, the actual available pins and macrocells in the 5C121 are higher than initially visible. Since macrocells in the 5C121 are organized into groups of four, when one output structure in a macrocell group is defined the other three must be of the same structure. Many times, this results in unused pins being labeled "RESERVED" in the utilization report.

**Configuration 3**

The final circuit (Figure 6) again uses eight inputs (I0 to I7) and eight outputs (Q0 to Q7), though this time the deselected outputs "remember" their previously selected state. With the 5C121's register feedback option, deselected outputs can hold the last data bit sent to that output. New data appears when the output is selected again.

Equations 14 to 22 express the Boolean terms necessary to implement this hold feature in the digital crosspoint switch. Note that each output is now a function of both the present inputs and the previous output (Qnfbk), which implements the registered feedback. Data bits D3, D4 and D5 determine which data bit will pass to the output. Again, the number of p-terms dictates the use of combinational feedback, as in Configuration 2.

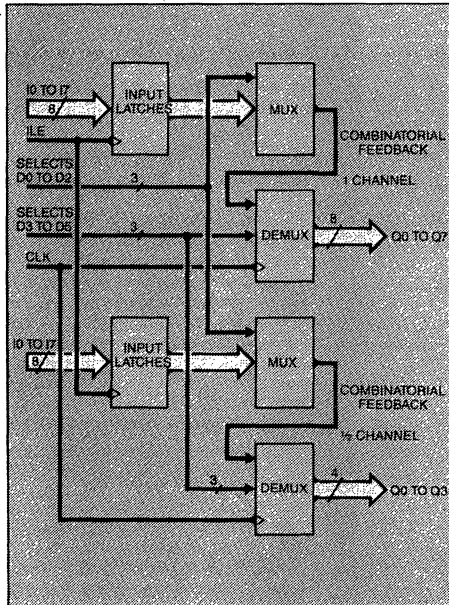


Figure 4: Configuration 2 features a low package count layout. Note that one and one-half switches fit into each 5C121 EPLD. This configuration uses combinational feedbacks to simplify the logic equations, thus eliminating the requirement for eight product terms per output.

Table 1. Equations for All Configurations	
$QA = A0 \cdot D2 \cdot D1 \cdot D0 + A1 \cdot D2 \cdot D1 \cdot D0 + A2 \cdot D2 \cdot D1 \cdot D0 + A3 \cdot D2 \cdot D1 \cdot D0 + A4 \cdot D2 \cdot D1 \cdot D0 + A5 \cdot D2 \cdot D1 \cdot D0 + A6 \cdot D2 \cdot D1 \cdot D0 + A7 \cdot D2 \cdot D1 \cdot D0;$	(1)
$Tsu = Tin + Tsd - Ttc$	(2)
$fmax = 1/(Tin + Tsd - Ttc)$	(3)
$SELECTEQ = 10 \cdot D2 \cdot D1 \cdot D0 + 11 \cdot D2 \cdot D1 \cdot D0 + 12 \cdot D2 \cdot D1 \cdot D0 + 13 \cdot D2 \cdot D1 \cdot D0 + 14 \cdot D2 \cdot D1 \cdot D0 + 15 \cdot D2 \cdot D1 \cdot D0 + 16 \cdot D2 \cdot D1 \cdot D0 + 17 \cdot D2 \cdot D1 \cdot D0;$	(4)
$Q0 = D5 \cdot D4 \cdot D3 \cdot SELECTEQ;$	(5)
$Q1 = D5 \cdot D4 \cdot D3 \cdot SELECTEQ;$	(6)
$Q2 = D5 \cdot D4 \cdot D3 \cdot SELECTEQ;$	(7)
$Q3 = D5 \cdot D4 \cdot D3 \cdot SELECTEQ;$	(8)
$Q4 = D5 \cdot D4 \cdot D3 \cdot SELECTEQ;$	(9)
$Q5 = D5 \cdot D4 \cdot D3 \cdot SELECTEQ;$	(10)
$Q6 = D5 \cdot D4 \cdot D3 \cdot SELECTEQ;$	(11)
$Q7 = D5 \cdot D4 \cdot D3 \cdot SELECTEQ;$	(12)
$Tsu = Tin + 2(Tsd) + Tcf - Ttc$	(13)
$SELECTEQ = 10 \cdot D2 \cdot D1 \cdot D0 + 11 \cdot D2 \cdot D1 \cdot D0 + 12 \cdot D2 \cdot D1 \cdot D0 + 13 \cdot D2 \cdot D1 \cdot D0 + 14 \cdot D2 \cdot D1 \cdot D0 + 15 \cdot D2 \cdot D1 \cdot D0 + 16 \cdot D2 \cdot D1 \cdot D0 + 17 \cdot D2 \cdot D1 \cdot D0;$	(14)
$Q0 = D5 \cdot D4 \cdot D3 \cdot SELECTEQ + D5 \cdot D4 \cdot D3 \cdot Q0fbk;$	(15)
$Q1 = D5 \cdot D4 \cdot D3 \cdot SELECTEQ + D5 \cdot D4 \cdot D3 \cdot Q1fbk;$	(16)
$Q2 = D5 \cdot D4 \cdot D3 \cdot SELECTEQ + D5 \cdot D4 \cdot D3 \cdot Q2fbk;$	(17)
$Q3 = D5 \cdot D4 \cdot D3 \cdot SELECTEQ + D5 \cdot D4 \cdot D3 \cdot Q3fbk;$	(18)
$Q4 = D5 \cdot D4 \cdot D3 \cdot SELECTEQ + D5 \cdot D4 \cdot D3 \cdot Q4fbk;$	(19)
$Q5 = D5 \cdot D4 \cdot D3 \cdot SELECTEQ + D5 \cdot D4 \cdot D3 \cdot Q5fbk;$	(20)
$Q6 = D5 \cdot D4 \cdot D3 \cdot SELECTEQ + D5 \cdot D4 \cdot D3 \cdot Q6fbk;$	(21)
$Q7 = D5 \cdot D4 \cdot D3 \cdot SELECTEQ + D5 \cdot D4 \cdot D3 \cdot Q7fbk;$	(22)

**Timing Analysis**

This configuration's timing analysis is similar to Configuration 2's combinational feedback analysis, with the exception of a register feedback delay (Trf). Trf is the time that the data is present at the output of the flip-flop to the time that data is available to the array.

The total delay associated with the registered feedback consists of the Trd, the Trf and the Tsd. Data from the flip-flop output reaches the input in about 50 nsec. The delay associated with data coming from the input pins is the same as that of Configuration 2 with combinational feedback – approximately 83 nsec. Using this as the clock period, there is ample time to implement the register feedback without affecting the cycle time. In this configuration, data could be clocked through at 12 Mbits/sec.

Combinational feedback reduces the p-term requirement to two p-terms per equation. This allows one and one-half crosspoint switches to fit into one 5C121. The design utilizes 64% of the available pins, 42% of the macrocells and 11% of the product terms. Six devices would be required to implement a byte-wide switch.

All of the configurations function differently, and no one configuration is optimum for all applications. A designer can customize a device to meet the needs of an application, whether those needs include higher speed or lower chip count. A second device can be quickly developed for a different application. Designers are no longer restricted to a single device type that must be adapted to an application with additional logic devices.

An original design can be developed in an afternoon. Additional devices derived from an original design can be developed in a few hours. Also, the ability to erase an EPLD and reprogram it allows design errors to be corrected immediately. Instead of several weeks delay with gate arrays, a designer using EPLDs can have working silicon devices in one day.

Both the flexibility and short design times associated with EPLDs make them a good choice for applications that benefit

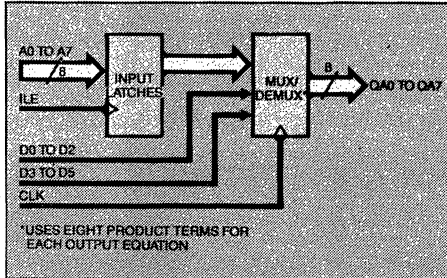


Figure 5: This circuit (Configuration 2 optimized for speed) combines the multiplexer and demultiplexer functions for each channel in a single array. Since each output equation uses eight product terms, only one switching channel can fit into each 5C121 package.

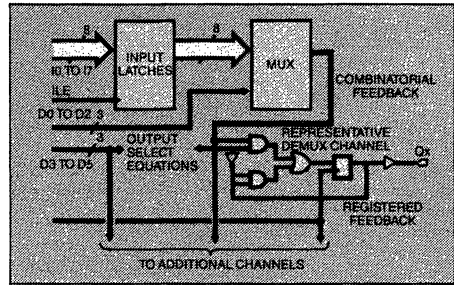


Figure 6: Configuration 3 shows the use of registered feedback to allow deselected outputs to retain their previously selected data. The logic for a representative channel is shown. As with Configuration 2, this configuration can be optimized for package count or speed.

from custom silicon devices. Today, EPLDs offer designers the densities and configuration flexibility of gate arrays, along with the short development time and cost associated with EPROMs.



# A Programmable Logic Mailbox for 80C31 Microcontrollers

Karlheinz Weigl and Jim Donnell, Intel Corp., Frankfurt, West Germany, and Folsom, CA

This article describes the implementation of a semi-intelligent interface between two 80C31 microcontrollers, using a mailbox protocol. Applications for an interface such as the one described here are often found in industrial control areas where multiple microcontrollers are used to accomplish a given task. Due to the architecture of the microcontroller (i.e., no READY input; no HOLD/HLDA interface; port-oriented I/O; etc.), exchanging data and status between these devices becomes a cumbersome task. Given this directive, it becomes the designer's task to develop a multi-port memory interface that allows for zero wait-state operation (i.e., no READY signal required), that electrically isolates the microcontroller buses, and that permits asynchronous access. Synchronization would result in the generation of wait states.

We realize the logic necessary to implement the desired functions in two erasable programmable logic devices (EPLDs). One device, the 5C031, contains roughly the equivalent of 300 2-input NAND gates, while the other EPLD, the 5C060, can implement designs with up to approximately 600 gates.

## The Mailbox Principle And its Implementation

In a mailbox memory system, the microcontrollers exchange information as bytes of data written to or read from a mailbox register. Control logic permits simultaneous access to the mailbox, thus eliminating the need for arbitration between the microcontrollers. Implementing the data exchange in this form achieves most of the design criteria given above.

Avoiding bus arbitration together with the short propagation delays of the

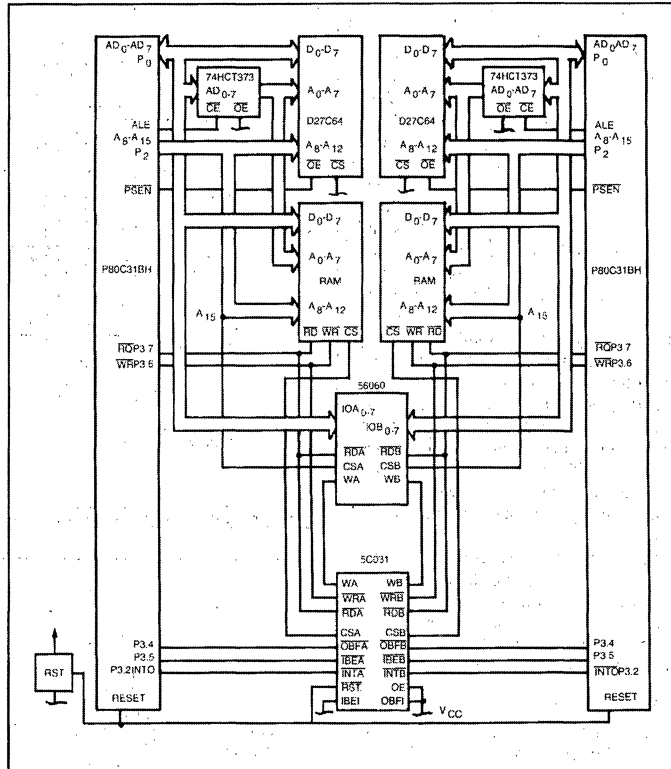


FIGURE 1. Schematic of mailbox memory system.

EPLDs provides zero wait-state operation of the data exchange. Electrical isolation of the address and data buses is achieved by using the high-impedance output capability of the 5C060. Simultaneous, asynchronous access is achieved by separating the RD and WR strobes issued by each microcontroller.

With a mailbox memory system, there

is an obvious need for some type of communication protocol to confirm the reception of a message, or the presence of data in the mailbox. In addition, the read and write logic must be defined such that simultaneous access to the mailbox is permitted. In order to segment the task, the design will be approached in terms of two separate mod-

ules: the mailbox (memory section), and the the control logic (protocol).

To begin the design of the memory section, it is first helpful to identify the resources required for the design. The mailbox requires a total of 16 memory storage registers (two bytes of data), tri-state output control, and two separate clock lines used to write the memory registers.

The 5C060 EPLD was chosen to implement the memory section. This device contains 16 programmable register groups that may be configured to operate as JK-, RS-, D-, and T-type flip-flops. Each register group feeds a bi-directional input/output pin, which may be tri-stated via an output-enable product term. These I/O pins may also serve as data inputs when the register output is tri-stated. This feature forms the basis of the read-signal logic required in the design. Write logic can be accomplished through the two synchronous clock inputs provided in the 5C060. Each synchronous clock drives a set of eight registers in the device. The operation of the memory section of the mailbox memory may now be solidified.

As shown in Figure 1, the two micro-controllers are separated into controller A and controller B. Register group A (signals IOA0 to IOA7) serves as an input buffer to microcontroller A. This buffer receives information from microcontroller B's data bus. The write control for register group A comes from microcontroller B.

Again, referring to Figure 1, it can be seen that register group B serves as an output buffer to microcontroller B. This buffer gets information from microcontroller A and is therefore write-controlled by microcontroller A.

### Data Transfer

In order to read data from the mailbox, the microcontroller must initiate a read cycle addressing the mailbox. The read signal (RDA for microcontroller A, RDB for microcontroller B) enables the tri-state outputs of the 5C060, revealing the appropriate data. Spurious read cycles are avoided by logically combining the read signal with a chip select signal (CSA or CSB) within the chip. The example shown utilizes address bit A15 as the

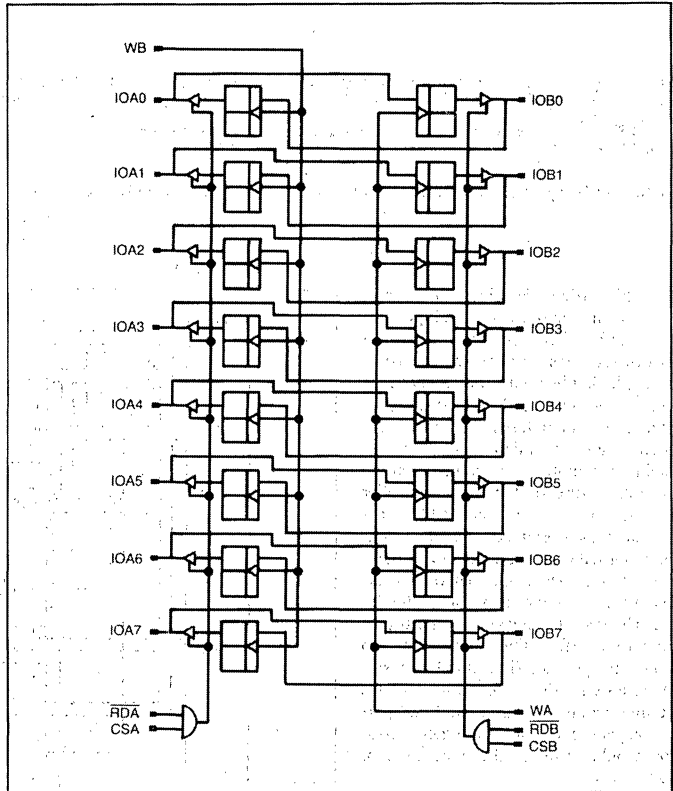


FIGURE 2. Schematic of register interface.

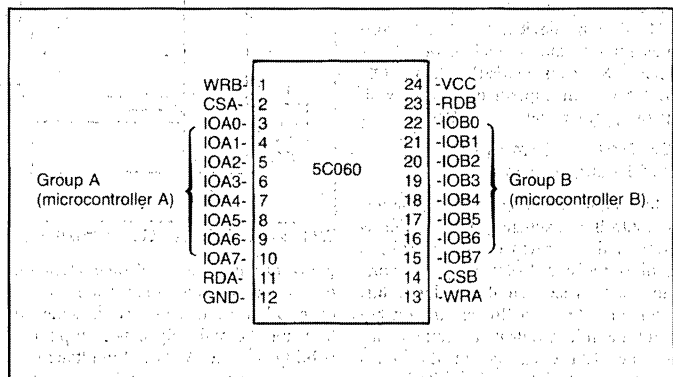


FIGURE 3. Pin-out for register interface.

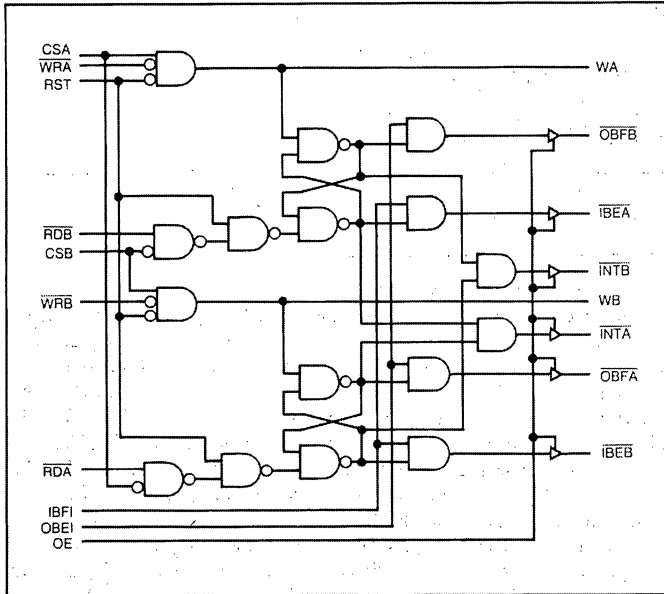


FIGURE 4. Schematic of control logic.

chip-select signal, thereby reserving the upper 32K bytes of memory space for the mailbox.

**Protocol Control Logic**

Having defined the memory section of the mailbox, we next must orchestrate the control logic. To guarantee reliable data transfers, both microcontrollers need feedback about the status of their respective input and output buffers.

In order to achieve a maximum data transfer rate, an interrupt-driven protocol may be used. The signals necessary to achieve the transfer protocol are:

- OBFA (A's output buffer full)
- OBFB (B's output buffer full)
- IBEA (A's input buffer empty)
- IBEB (B's input buffer empty)
- INTA (A's data ready interrupt)
- INTB (B's data ready interrupt)

Further definitions of the control signals can be made as follows.

**Output Buffer Full.** This flag is set whenever a controller writes to the mailbox. The flag remains valid until the second controller has read the data. The

flag is reset when the recipient controller reads the data from the mailbox.

**Input Buffer Empty.** This flag indicates that there is no message in the mailbox and that the mailbox can be written without corrupting the data. This flag is set whenever a controller reads data from the mailbox. The flag remains set until data is placed in the mailbox.

**Interrupt.** The 5C031 is programmed to supply interrupts to both microcontrollers involved, when either one of two events occurs. First, the recipient microcontroller receives an interrupt when its OBF flag goes active. This signals the recipient that data is available in the mailbox. Secondly, the originator microcontroller receives an interrupt when data placed by that microcontroller in the mailbox has been received by the recipient microcontroller. This interrupt indicates that data has been received and that it is safe to write data to the mailbox.

The signals described above form the basis for clean and efficient data transfer between the two microcontrollers. The transfer time is limited only to the over-

head of the interrupt service routines. The 5C060 can accept data at clock rates in excess of 20 MHz.

**Programming the EPLDs**

Figures 2 and 3 show the schematics and pin-out for the memory section, and Figure 4 is a schematic of the protocol sections in the mailbox memory. Using Intel's Programmable Logic Development System, these schematics can be transformed with ease into the logic equations that represent the desired function. The development system accepts a variety of entry methods, including schematic, netlist, state machine, and text file entry.

Once the design has been entered, the file is submitted to the Logic Optimizing Compiler (LOC), which performs an optional Boolean minimization, including De Morgan's inversion, and logically fits the design into the target EPLD.

The development system generates three output files. The Logic Equation File (LEF) contains the result of the minimization process, the Utilization Report File (RPT) contains the final device pin-out, information about the internal logic routing, and a percent utilization for pins, macrocells, and product terms. Finally, the JEDEC file (JED) contains the device programming information required to program the EPLDs. These files are available from the authors.

Programming of the EPLDs is accomplished through Intel's Logic Programming Software (LPS) and the iUP-PC programming hardware. Designs also may be logically simulated through the use of Intel's FSIM software.

**Summary**

Applications such as industrial automation often require communication between multiple microcontrollers. Unfortunately this communication is hampered by the port orientation and lack of bus control signals within the microcontroller environment. One solution—as presented here—is the mailbox memory. The mailbox memory serves as an effective method for transferring data between microcontrollers, while the flexibility of the EPLDs serves as an effective way to implement the mailbox itself. □



## DESIGN APPLICATIONS

ELECTRONIC DESIGN EXCLUSIVE

# Regain lost I/O ports with erasable PLDs

Daniel E. Smith and Thomas B. Bowns

Intel Corp., 1900 Prairie City Rd., Folsom, CA 95630; (916) 351-2747.

As a means for reconstructing or regaining microcontroller I/O ports lost to memory expansion, erasable programmable logic devices, or EPLDs, contain all the necessary functions. In fact, EPLDs perform more functions than most programmable logic arrays, and offer the additional benefits of EPROM-like erasability, the low power consumption of CMOS technology, and gate densities near those of low-end gate arrays.

Lost I/O ports can be externally reconstructed

with standard SSI packages. EPLDs, however, supply an alternative that reduces the external approach's impact on power and space consumption.

The computing power of one-chip microcontrollers plays a role in many applications. But the growing

complexity of these devices, as designers shift from 8- to 16-bit controllers, has strained their I/O capacity.

A typical 8-bit microcontroller in a 40-pin package contains a 4- to 8-kbyte program memory and 32 I/O pins, usually grouped into 8-bit ports. The 16-bit devices contain 8-kbyte memories and up to 40 I/O pins in packages that range from 48 to 68 pins. The possible number of ports falls short for some complex tasks in switching circuits, robotics, and automotive systems.

The I/O shortage is aggravated when the chip's internal program memory is too small for a given task. While tacking on external memory is easy enough, the addition consumes I/O pins.

Although some details vary, the basic techniques for reconstructing these lost I/O ports with EPLDs are the same for most microcontrollers. An example describes a 5C121 EPLD and an 8096 16-bit microcontroller, noting details specific to the micro-

controller.

These techniques not only reconstruct ports on any available microcontroller, but they also are suited to adding new ports. For the 8096, the designer can add two new ports, 5 and 6, by changing to 1FFC-1FFF the hexadecimal address range in which the external memory is deselected. The new ports create a system with 56 I/O signals. The tradeoffs of this addition are the board space needed for two more EPLDs and two more bytes of reserved memory space at 1FFC and 1FFD.

The first consideration in reconstructing a port is the microcontroller's fixed-memory and I/O address map. In the 8096, memory-address ranges 0 to FF and 2000-3FFF contain on-chip registers, interrupt vectors, factory test code, and program memory. Expansion memory can go into the 100 to 1FFD range, a capacity of 8k bytes minus the first 256 and the last 2 bytes, and into the 4000 to FFFF range, another 8 kbytes.

The microcontroller has five 8-bit ports, three of which (0 to 2) are dedicated to I/O functions. Ports 3 and 4, however, are memory-mapped to 1FFE and 1FFF, respectively. These two ports reside right above the lower section of expansion memory space. (Other microcontrollers have the same functions, but their address ranges may vary.)

External memory, therefore, connects to the pins reserved for ports 3 and 4, eliminating them as general I/O ports. Reclamation of these ports calls for external latches and decode logic that disables the external memory and enables the latches at 1FFE and 1FFF. This logic decodes signals  $A_0$  and Byte High Enable, BHE, to select ports 3 and 4. The ports are selected either separately for 8-bit data transfers or together for 16-bit transfers.

The microcontroller multiplexes address and data on signal lines  $AD_0$  to  $AD_{15}$ . As a result, Address Latch Enable, ALE, must latch the address as each bus cycle starts and keep it there for the cycle duration. Then the lines can transfer data throughout the cycle. Because  $\overline{BHE}$  has the same timing as

**Erasable PLDs cut the space and power usually needed to reconstruct I/O ports. They can even build new ports, adding to a chip's capabilities.**

## DESIGN APPLICATIONS ■ Erasable PLDs restore ports

the address, ALE must also latch  $\overline{\text{BHE}}$ .

Reconstruction of both ports without EPLDs requires 14 SSI packages if the high-current sink capability of open-collector drivers is needed. If not, nine packages will do.

Besides the address-decoding logic, the input ports need octal latches. The outputs contain octal latches, but inverting buffers are also needed. If the output does include open-collector drivers, the designer must add another set of inverting buffers to compensate for the drivers' inversion of the signal. In addition, a discrete flip-flop latches  $\overline{\text{BHE}}$ , and discrete gates decode the port selection and RD and WR signals.

On the other hand, reconstructing ports with EPLDs requires no logic outside of the EPLDs themselves (Fig. 1). Each device decodes its respective memory-mapped address, and one device disables the external memory at both 1FFE and 1FFF.

The EPLDs can sink 4 mA, which puts them in the same range as an SSI version without open-collector drivers. The designer can add open-collector drivers if a higher-current sink is needed.

The design process leading to port reconstruction be-

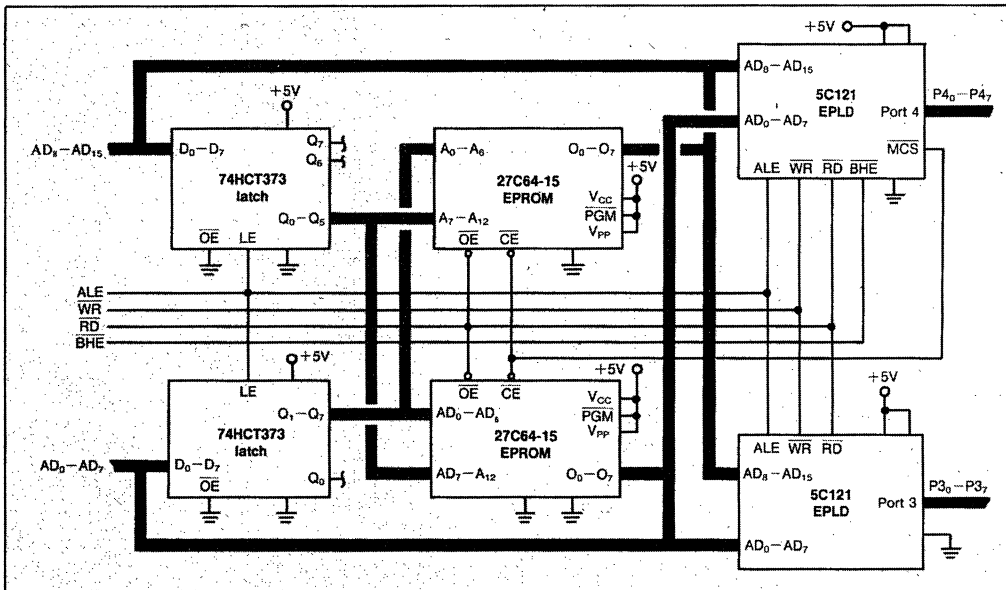
gins with defining the functions required of the EPLD and then creating a design file that can be translated into a Jedec file. Next, the designer programs the EPLD and tests the final circuit. Software can automate much of this procedure.

The first step is to list the functions the EPLD must perform. Then the designer identifies which EPLD feature best satisfies that need, because as with SSI logic, the device can accomplish its task in different ways.

In general, a device reconstructing a port must latch and decode address information from a multiplexed bus. The chip then produces an internal port-selection signal and an external memory-selection signal; the latter in address range 1FFE-1FFF. Moreover, the device acts as a bidirectional data path and decodes the RD and WR signals, routing the data with the port-selection signal (Fig. 2).

Drawing a schematic diagram of the EPLD helps isolate the circuit into functional blocks. In the example, combinatorial logic and three latches do the decoding at port 3.

Address lines  $\text{AD}_1$  through  $\text{AD}_{11}$  pass through an AND gate and are latched as  $\text{LAD}_A$ . Address lines  $\text{A}_{12}$



1. Two erasable programmable logic devices contain all the logic required to reconstruct ports 3 and 4 of an 8096 microcontroller. The two latches and two EPROMs comprise the external memory.

and inverted signals  $AD_{13}$  through  $AD_{15}$  pass through an AND gate and are latched as  $LAD_B$ . These two latched signals pass through another AND gate to create the Memory Disable Signal, MDS, which deactivates the EPROMs. Combined with  $LAD_0$  (address signal  $AD_0$  inverted and latched),  $LAD_A$  and  $LAD_B$  generate the port-selection signal.

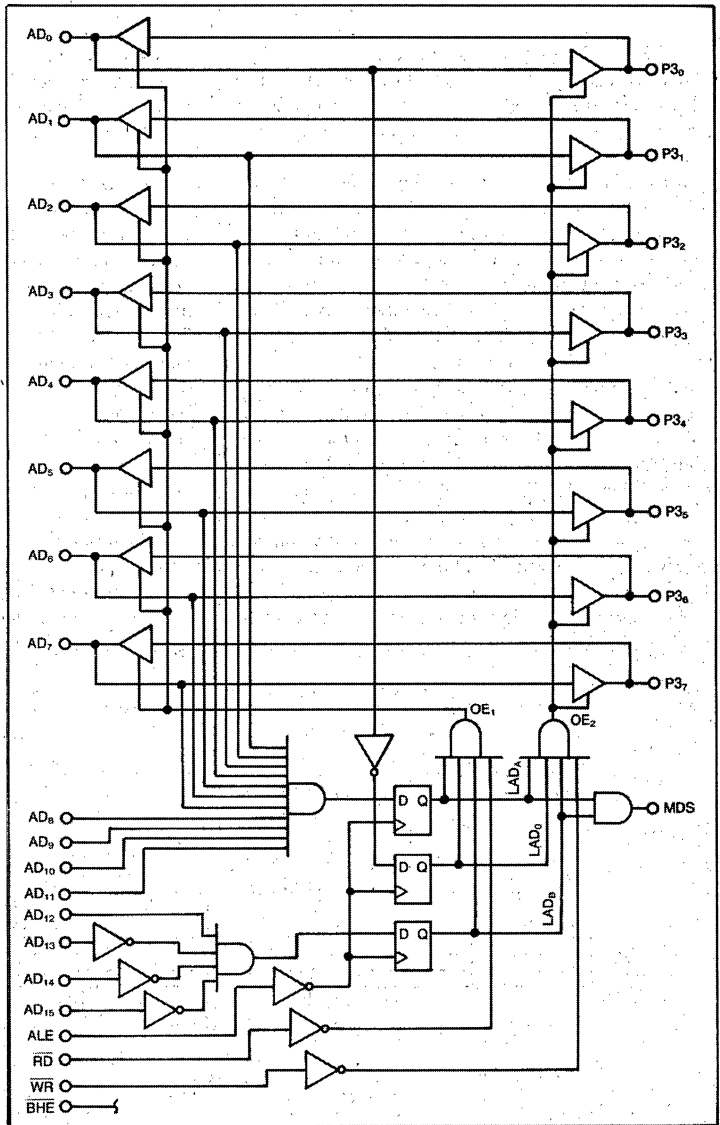
#### PARALLEL FORMAT SAVES TIME

The EPLD decodes and latches signals  $AD_1$  through  $AD_{11}$  and  $AD_{12}$  through  $AD_{15}$  in parallel to minimize the time between address setup and ALE going low. An inverted ALE clocks the latches, which also store decoded addresses while the microcontroller transfers data over the bus.

Two combinatorial-output, internal feedback (COIF) primitives create a double-feedback loop with all output enables to the microcontroller bus controlled by  $OE_1$ , which is active during read operations. Output enables on the I/O side of the EPLD are controlled by  $OE_2$ , which is active during write operations. Thus data is valid at the inputs or outputs only while the appropriate command,  $\overline{RD}$  or  $\overline{WR}$ , is active.

If the application calls for latched outputs, the designer can create them from logic on the EPLD. One configuration is a D-type latch activated by the trailing edge of  $\overline{WR}$  (Fig. 3). In this circuit, the outputs are always enabled, except during reads, when they are placed in a high-impedance state. The Reset signal clears the outputs to a logic 0 during initialization.

The fourth port's schematic varies little from that of the third. Because port 4 handles data transfers on the micro-



**DESIGN APPLICATIONS ■ Erasable PLDs restore ports**

controller's high byte, the data path connects to AD<sub>8</sub> through AD<sub>15</sub>. The BHE signal replaces AD<sub>0</sub> and becomes LAD, which combines with LAD<sub>A</sub> and LAD<sub>B</sub> to select the correct port.

A microcontroller with a different address map or bus interface may require some variations in address decode logic. The basic techniques for regaining I/O ports with EPLDs, however, remain the same.

**DESIGN FILE CREATED**

The next step in the port-reconstruction process is to create from the schematic diagram a design file that can be automatically converted to a Jedec file by Intel's Programmable Logic Software II (iPLSII) program. Four types of inputs are acceptable: a net list file, Boolean equations, state variables, and files from any of several schematic-entry packages that run on personal comput-

ers. The designer can write a net list file with a word-processing program in a nondocument mode, but an easier way is to work with iPLS II's Logic Builder.

The Logic Builder prompts the user for the information it needs. After establishing the file with some background information, the program asks for lists of all the input and output pin names (the user can assign a name to a specific pin number). Next come the internal assignments and connections, and finally, the logic equations needed.

The designer must list all the COIFs that form the bidirectional data path. For example, the entries that create the data line between AD<sub>0</sub> and P3<sub>0</sub> (see Fig. 2 again) are as follows:

$$AD_0, AD_0 = COIF(P3_0, OE_1)$$

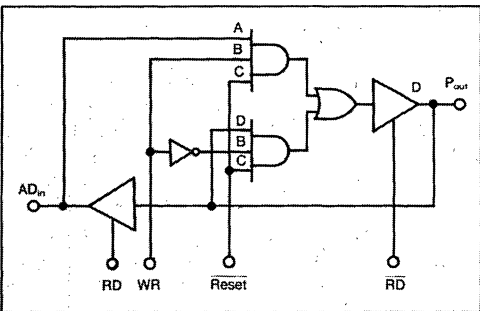
$$P3_0, P3_0 = COIF(AD_0, OE_2)$$

The iPLS II program contains a logic-optimizing compiler that translates the schematic's net list, or other suitable input, into a Jedec programming file. The compiler, which is selected from the program's main menu, optimizes the logic equations and assigns I/O pins and other EPLD resources.

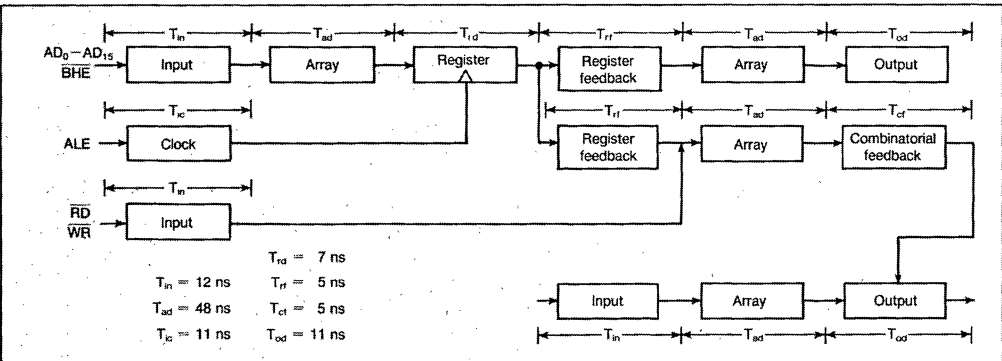
**ERROR MESSAGES POINT OUT PROBLEM**

The program's outputs are the programming file and a device report file that shows the pinout of the programmed device and describes the use of the device's resources. If the compiler cannot translate the file, error messages indicate the design-file entry that caused the problem.

Programming the EPLD is very similar to programming EPROMs. The designer connects an EPLD programming module to the workstation, inserts an unprogrammed device into the socket, and calls up the



3. If a designer needs latched outputs, they can be built without additional logic. This D-type flip-flop is made of logic elements contained in the EPLD.



4. A block diagram of an EPLD's internal delays shows how users can determine the maximum delay for each signal path and, as a result, the port's maximum operating frequency.

programming menu. The menu asks for the device's type and the Jedec file name, and the system then programs and verifies the chip.

Considering how straightforward the port-reconstruction functions are, the best test of the programmed EPLD is to plug it into a circuit and see if it works. An EPROM-based microcontroller with some simple read and write routines to exercise the device works well. The designer can also use an in-circuit emulator for the microcontroller, if one is available.

Any bugs can be fixed quickly. To correct a bug the user erases the EPLD file and changes the design file, which then can be recompiled and the device reprogrammed.

A timing analysis confirms the EPLD's compatibility with different microcontroller clock speeds. The analysis amounts to adding the internal delays for paths through the EPLD and comparing these path delays to the microcontroller's timing requirements.

The three paths of interest are Address Setup to ALE, which must take no longer than 116 ns for an 8096 operating at 6 MHz; and no longer than 50 ns at 10 MHz. Other maximum values are: Data Valid From RD, 358 ns and 230 ns; and Data Valid Before Write, 272 ns and 130 ns.

A block diagram of the specific device with each internal delay is needed for the timing analysis. For the example circuit, the Address Setup to ALE delay for the port 3 EPLD is 49 ns (Fig. 4). This value, achieved by decoding and latching AD<sub>1</sub> to AD<sub>11</sub> in parallel with AD<sub>12</sub> to AD<sub>15</sub>, just meets the maximum delay at 10 MHz.

The delay for Data Valid From RD is the sum of delays in the enable path and the data path, or 136 ns. The delay path for the write operations is shorter: It is that for the enable path added to 41 ns for the data path (after eliminating a 30-ns overlap in enable and data timing), or 106 ns. Both are well within limits. □

*Daniel E. Smith, a senior technical writer at Intel, has also worked in microcomputer-systems testing and written manuals for microprocessors, development software, and bubble memories. He has a BA in history from San Jose University and an MA in biblical studies from the Graduate Theological Union/Jesuit School of Theology in Berkeley, Calif.*

*Thomas B. Bowns is an application engineer for Intel's EPLD operation. He also has worked as a technician on the company's EPROM line. Bowns studied digital and microwave electronics at American River College in Carmichael, Calif.*

---

# Advanced Architecture

## EPLDs

---

**3**



# 5AC312 1-MICRON CHMOS ERASABLE PROGRAMMABLE LOGIC DEVICE

- High Performance LSI Semi-Custom Logic Alternative for Low-End Gate Arrays, TTL, and 74HC- or 74HCT SSI and MSI Logic
  - High Speed tpd (max) 25 ns, 50 MHz Performance Pipelined, 33 MHz with Feedback
  - 12 Macrocells with Programmable I/O Architecture; Up To 22 Inputs (10 Dedicated, 12 I/O) or 12 Outputs
  - 8 Programmable Inputs Individually Configurable as Latches, Registers or Flow-Through
  - Software-Supported Product Term Allocation between Adjacent Macrocells
  - Programmable Output Registers Configurable as D, T, JK, or SR Types
  - Dual Feedback on All Macrocells for Buried Registers with Bidirectional I/O
  - 2 Product Terms on All Macrocell Control Signals
  - CHMOS III-E EPROM Technology based; UV-Erasable
  - UV Erasable Array for 100% Generic Testability
  - Programmable Security Bit Allows 100% Protection of Proprietary Designs
  - Programmable Low-Power Option for Standby Operation; 100  $\mu$ A Typical Standby Current
  - Available in 24-Pin 0.3" DIP and 28-Pin PLCC Packages
- (See Packaging Spec., Order Number #231369)

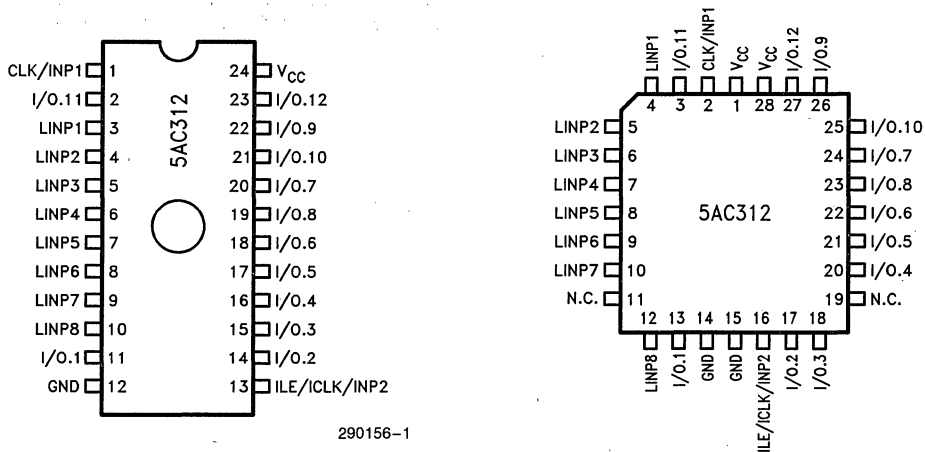


Figure 1. Pin Configurations



## INTRODUCTION

The Intel 5AC312 CHMOS EPLD (Erasable Programmable Logic Device) represents an innovative approach to overcoming the primary limitations of standard PLDs. Due to a proprietary I/O architecture and macrocell structure, the 5AC312 is capable of implementing high performance logic functions more effectively than previously possible. It can be used as an alternative to low-end gate arrays, multiple programmable logic devices or LS-, HC- or HCT SSI and MSI logic devices. Input and macrocell features for the 5AC312 are a superset of features offered by other PLD-type products.

The 5AC312 uses advanced CHMOS EPROM cells as logic control elements instead of poly-silicon fuses. This technology allows the 5AC312 to operate at levels necessary in high performance systems while significantly reducing the power consumption. Its programmable stand-by function reduces power consumption to almost "zero" in applications where a slight speed loss is traded for power savings.

## ARCHITECTURE DESCRIPTION

The architecture of the 5AC312 is based on the familiar "Sum-Of-Products" programmable AND, fixed OR structure, though the 5AC312 macrocell contains a number of significant functional enhancements. This device can implement both combinational and sequential logic functions through

a highly flexible macrocell and I/O structure. The 5AC312 has been designed to effectively implement both combinational-register and register-combinational-register forms of logic to easily accommodate state machine designs.

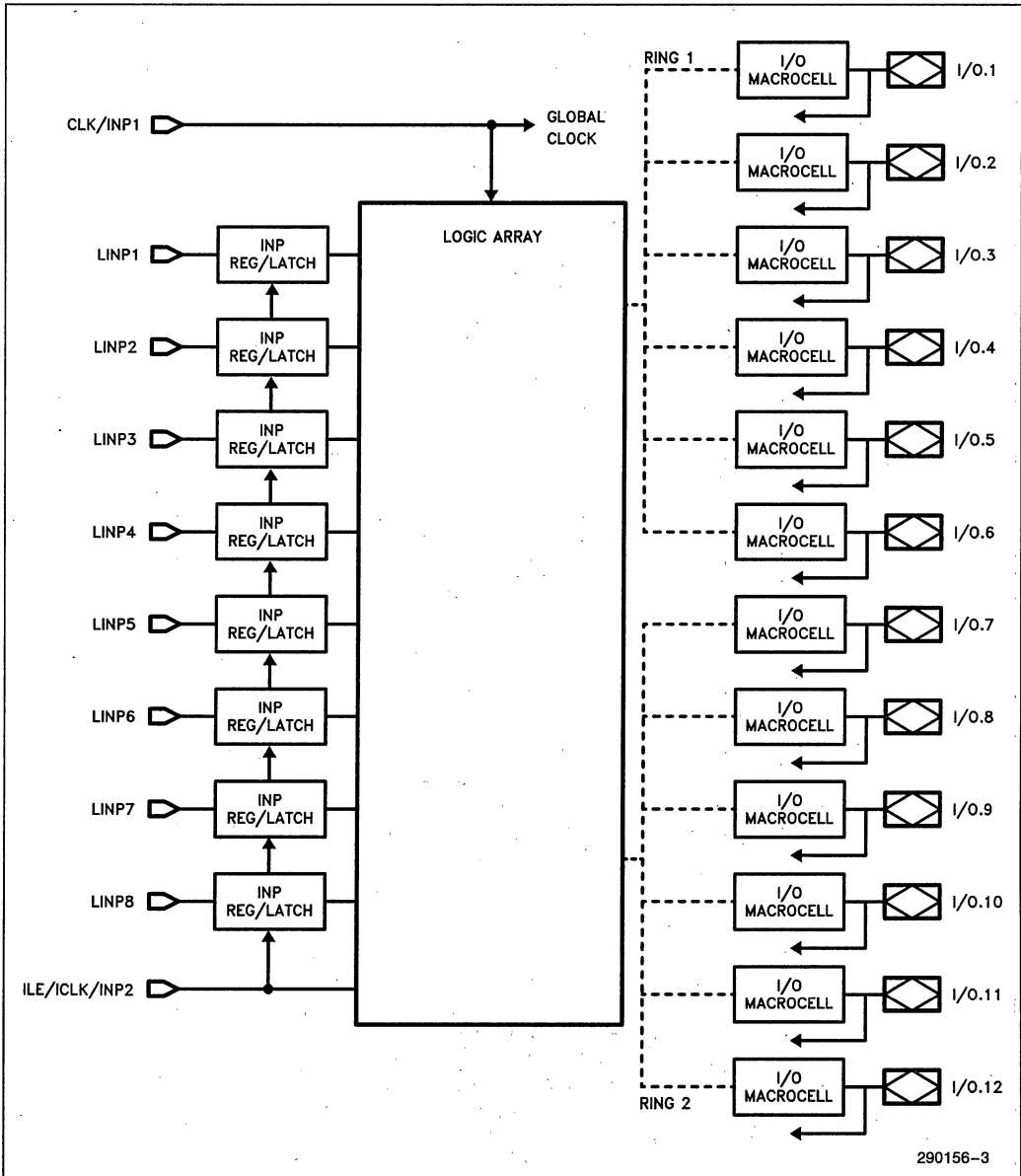
Figure 2 shows a global view of the 5AC312 architecture. The 5AC312 contains a total of 12 I/O macrocells, 8 user-programmable input structures, and 2 additional inputs that can be programmed to serve as either combinatorial inputs or clock inputs. Each of the eight inputs can be individually configured as a latch, register, or flow-through input. Input latches/registers can be synchronously or asynchronously clocked.

Each macrocell is further sub-divided into 16 Product Terms with 8 Product Terms dedicated to the control signals OE, PRESET, ASYNCH. CLK and CLEAR, and 8 Product Terms available for the general data array (see Figure 3).

The basic macrocell architecture of the 5AC312 includes a user-programmable AND array and a user-configurable OR array. The inputs to the programmable AND array originate from the true and complement signals from the programmable input structure, the dedicated inputs, and the 24 feedback paths from the 12 I/O macrocells.

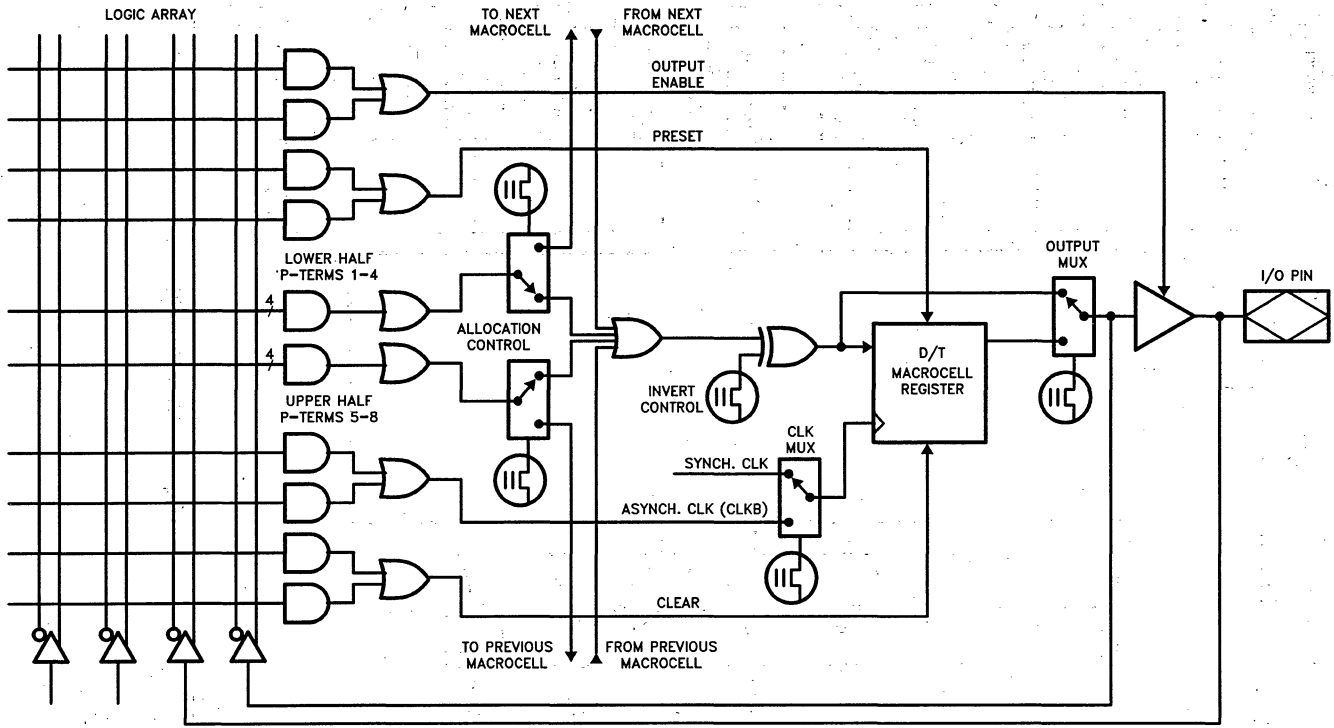
### Programmable Input Structure

Figure 4 shows a block diagram of the 5AC312 input architecture. This device contains 8 user-program-



290156-3

Figure 2. 5AC312 Architecture



290156-4

Figure 3. 5AC312 Basic Macrocell Structure

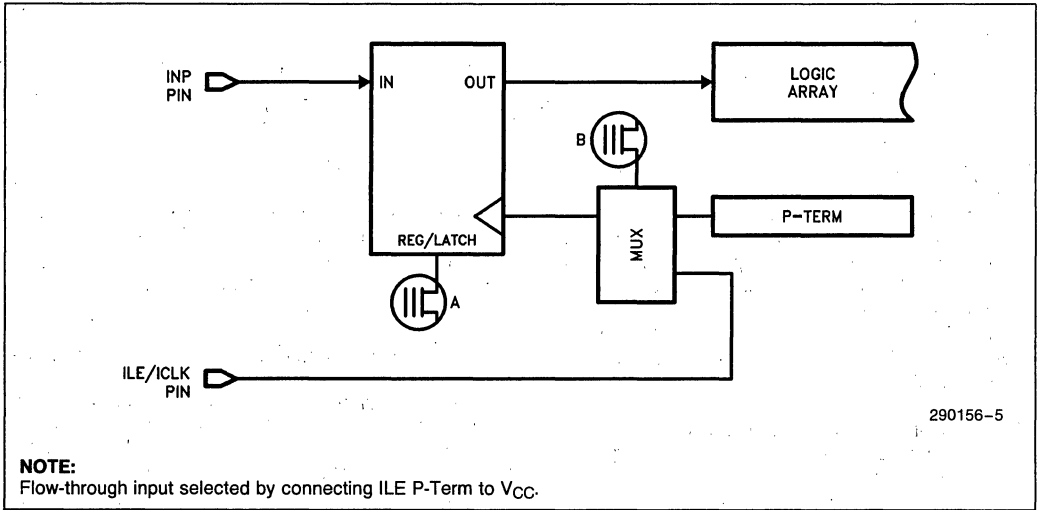


Figure 4. 5AC312 Input Structure

mable input structures that can be individually configured to work in one of five modes:

- Input register (D-register), synchronous operation
- Input register (D-register), asynchronous operation
- Input latch (D-latch), synchronous operation
- Input latch (D-latch), asynchronous operation
- Flow-through input

The configuration is accomplished through the programming of EPROM architecture control bits by the logic compiler and programmer software. If synchronous operation is chosen, the ILE/ICLK/INP becomes an ILE/ICLK (Input Latch Enable) input global to all input latch/register structures. For asynchronous operation, ILE/ICLK/INP can be used as a normal input (flow-through input) to the device while a separate Product Term in the control array is used

to derive an input clock signal for the input structure. Because the clock signal for each input structure can be individually selected, a mix between synchronously and asynchronously clocked input structures is also possible.

Table 1 shows the input latch/register function table with respect to the synchronous ILE/ICLK input.

Table 1. 5AC312 Input Latch/Register Functions

Input Type	ILE/ICLK	D	Q
Latch	H	H	H
Latch	H	L	L
Latch	L	X	Qn
D-FF	↓	H	H
D-FF	↓	L	L
Flow-Through	X	H	H
Flow-Through	X	L	L

H = HIGH Level L = LOW Level X = Don't Care

## Macrocell Array

Each of 12 macrocells in the 5AC312 contains 8 p-terms (Product Terms) to support logic functions. These 8 p-terms are subdivided into 2 groups each containing 4 p-terms. This grouping of p-terms supports the proprietary p-term allocation scheme.

Each macrocell can be configured as a D, T, RS, or JK register. The 8 p-terms for control functions are organized so that 2 p-terms support *each of the four* control signals. Control signals in the 5AC312 are: Output Enable (OE), asynchronous I/O register pre-set (PRESET), asynchronous clock for I/O registers (ASYNCH. CLK), and asynchronous I/O register reset (CLEAR).

CLK is a global clock signal that can be used to synchronously clock any or all macrocell registers. It can be used as an input to the logic array at the same time as a macrocell clock. When CLK is not used as a synchronous clock, it functions only as a dedicated input to the logic array.

## Combinatorial Configuration

The macrocell register can be bypassed to implement combinatorial logic functions. When configured to provide combinatorial logic, only the OE control signal is used.

## Invert Select Bit

An invert select EPROM bit is used to invert the product term input into each macrocell register, including double inputs on JK and SR registers. This invert option allows the highest possible logic utilization by use of DeMorgan's logic inversion.

## Product Term Allocation

Product Term allocation is defined as taking logic resources (p-terms) away from macrocells where they are not used to support demand for more than 8 Product Terms in other areas of the chip. In the 5AC312, this allocation can occur in increments of 4 p-terms between adjacent macrocells.

The 12 macrocells available in the 5AC312 are grouped into two "rings" with 6 macrocells per ring. Product Terms can be allocated in a "shift register" mode inside a ring; allocation of Product Terms between the rings is not supported. The two rings are shown in Figure 2 and listed in Table 2.

## Example:

The logic function in macrocell 4 requires 16 p-terms. In this case, the iPLS II software allocates 4 p-terms from the previous macrocell in Ring 1 (macrocell 3) and 4 p-terms from the next macrocell in Ring 1 (macrocell 5) to accumulate a total of 16 p-terms (8 + 4 + 4). This implementation leaves macrocells 3 and 5 with a remainder of 4 p-terms each. These remaining p-terms in macrocells 3 and 5 can also be allocated away to or can be supplemented with p-terms from their respective previous/next macrocells in Ring 1.

Applying this scheme to the 5AC312 it becomes clear that any macrocell inside the device can support logic functions requiring between 0 and 16 Product Terms. Product Terms allocated away from a macrocell do not affect that macrocell's output structure. If all Product Terms are allocated "away" from a macrocell, the input to that macrocell's I/O control block is tied to GND. This polarity can be changed by programming the invert select EPROM bit. The I/O register as well as all secondary controls to this I/O control block are still available and can be used if needed.

The Product Term allocation scheme described above is automatically supported by iPLDS II V2.0 and is transparent to the user. Users can still use explicit pin assignments, but should assign pins in a way that does not conflict with p-term allocation.

Table 2. Product Term Allocation Rings

Ring 1			Ring 2		
Current Macro-cell	Next Macro-cell	Previous Macro-cell	Current Macro-cell	Next Macro-cell	Previous Macro-cell
1	2	6	7	8	12
2	3	1	8	9	7
3	4	2	9	10	8
4	5	3	10	11	9
5	6	4	11	12	10
6	1	5	12	7	11

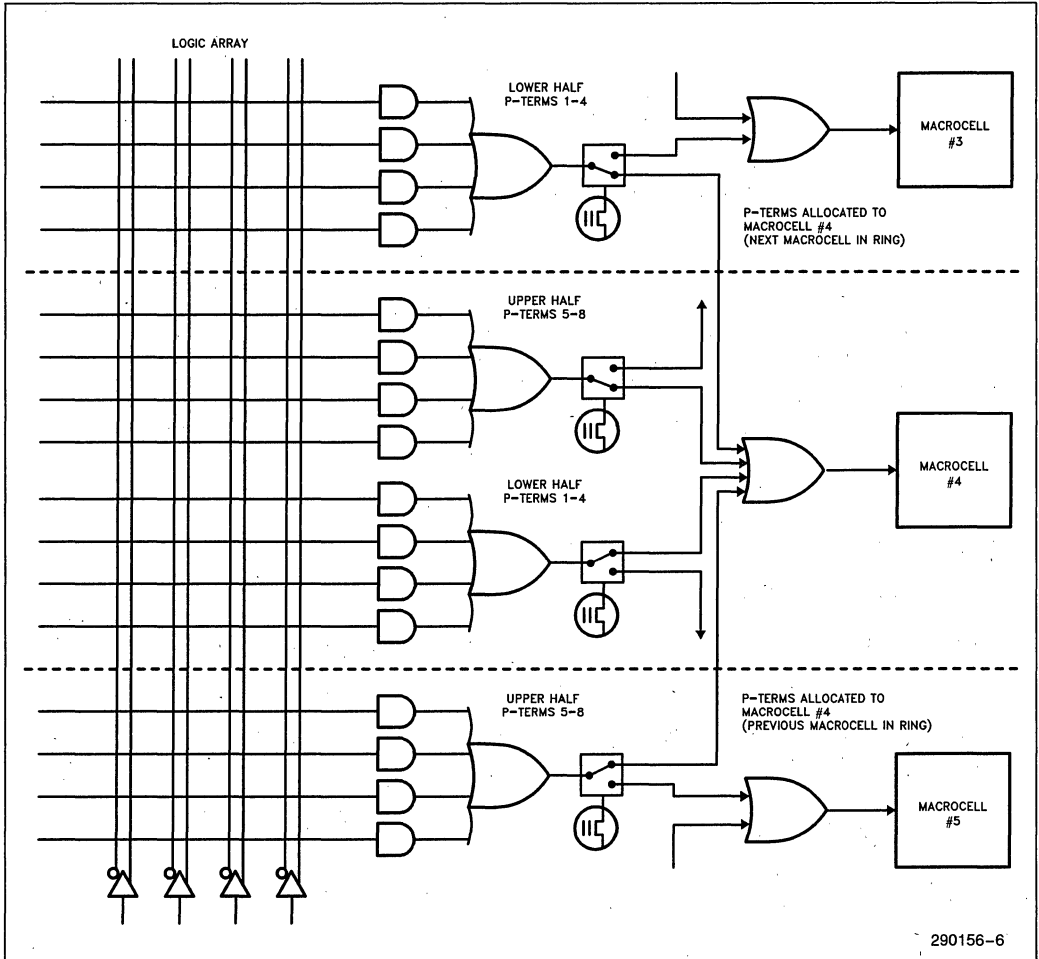


Figure 5. Product Term Allocation (8 + 4 + 4)

### Macrocell I/O Control Block

Each macrocell in the 5AC312 has the ability to implement D, T, SR, and JK registered outputs as well as combinatorial outputs. The asynchronous set and reset inputs to each macrocell register allows implementation of true SR Flip-Flops. Registered outputs may be clocked from the synchronous CLK/INP1 pin or asynchronously clocked by the 2 Product Terms available for ASYNCH. CLK. The 5AC312 also features separate input and feedback paths (dual feedback) on all macrocell I/O control blocks. This enables the designer to utilize input pins when the associated macrocells have been assigned a no output with buried feedback attribute. Multiplexed I/O is accomplished by controlling the output buffer associated with each macrocell using the 2 Product Terms set aside for implementing an OE function.

### Power-On Characteristics

The Macrocell registers of the 5AC312 will experience a reset to their inactive state (logic low) upon  $V_{CC}$  power-up. Using the PRESET function available to each macrocell, any particular register preset can be achieved after power-up. 5AC312 inputs and outputs begin responding within 10  $\mu s$  (6  $\mu s$  typical) after  $V_{CC}$  power-up or after a power-loss/power-up sequence. Input registers are not reset on power-up and are indeterminate. Input latches reflect the state of the input pins on power-up.

### Automatic Standby Mode

The 5AC312 contains a programmable bit, the Turbo Bit, that optimizes operation for speed or for power

savings. When the Turbo Bit is programmed (TURBO = ON), the device is optimized for maximum speed. When the Turbo Bit is not programmed (TURBO = OFF), the device is optimized for power savings by entering standby mode during periods of inactivity.

Figure 6 shows the device entering standby mode approximately 100 ns after the last input transition. When the next input transition is detected, the device returns to active mode. Wakeup time adds an additional 20 ns to the propagation delay through the device as measured from the first input. No delay will occur if an output is dependent on more than one input and the last of the inputs changes after the device has returned to active mode.

After erasure, the Turbo Bit is unprogrammed (OFF); automatic standby mode is enabled. When the Turbo Bit is programmed (ON), the device never enters standby mode.

### intelligent Programming™ Algorithm

The 5AC312 supports the intelligent Programming algorithm which rapidly programs Intel H-EPLDs, EPROMs and Microcontrollers while maintaining a high degree of reliability. It is particularly suited for production programming environments. This method greatly decreases the overall programming time while programming reliability is ensured as the incremental program margin of each bit has been verified in the programming process. (Programming information for the 5AC312 is available from Intel by request.)

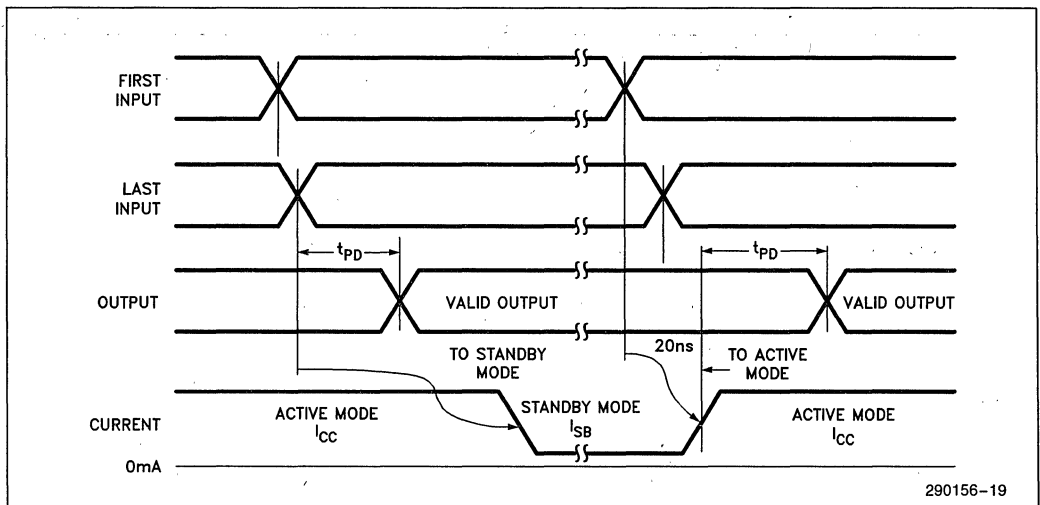


Figure 6. 5AC312 Standby and Active Mode Transitions

## ERASED STATE CONFIGURATION

After erasure and prior to programming, all macro-cells are configured as combinatorial, inverted outputs with output buffers three-stated. Inputs are configured as synchronous registers.

## ERASURE CHARACTERISTICS

Erasure time for the 5AC312 is 1 hour at 12,000  $\mu\text{W}/\text{cm}^2$  with a 2537Å UV lamp.

Erasure characteristics of the device are such that erasure begins to occur upon exposure to light with wavelengths shorter than approximately 4000Å. It should be noted that sunlight and certain types of fluorescent lamps have wavelengths in the 3000Å–4000Å range. Data shows that constant exposure to room level fluorescent lighting could erase the typical 5AC312 in approximately six years, while it would take approximately two weeks to erase the device when exposed to direct sunlight. If the device is to be exposed to these lighting conditions for extended periods of time, conductive opaque labels should be placed over the device window to prevent unintentional erasure.

The recommended erasure procedure for the 5AC312 is exposure to shortwave ultraviolet light with a wavelength of 2537Å. The integrated dose (i.e., UV intensity  $\times$  exposure time) for erasure should be a minimum of forty (40) Wsec/cm<sup>2</sup>.

The erasure time with this dosage is approximately 1 hour using an ultraviolet lamp with a 12,000  $\mu\text{W}/\text{cm}^2$  power rating. The device should be placed within 1 inch of the lamp tubes during exposure. The maximum integrated dose the 5AC312 can be exposed to without damage is 7258 Wsec/cm<sup>2</sup> (1 week at 12,000  $\mu\text{W}/\text{cm}^2$ ). Exposure to high intensity UV light for longer periods may cause permanent damage to the device.

## DESIGN SECURITY

A Security Bit provides a programmable security option to protect the data programmed in the device. Once this bit is set during programming, subsequent attempts to read the device architecture information are prevented. This method provides a higher degree of design security than fuse-based devices, since programmed EPROM cells are invisible even to microscopic examination. The Security Bit (also called the Verify Protect Bit), along with all the other EPROM cells, is reset by erasing the device.

## LATCH-UP IMMUNITY

All of the input, I/O, and clock pins of the device have been designed to resist latch-up which is inherent in inferior CMOS structures. The 5AC312 is designed with Intel's proprietary 1-micron CHMOS EPROM process. Thus, each of the pins will not experience latch-up with currents up to 100 mA and voltages ranging from  $-0.5\text{V}$  to  $V_{\text{CC}} + 0.5\text{V}$ . The programming pin is designed to resist latch-up to the 13.5 maximum device limit.

## DESIGN RECOMMENDATIONS

For proper operation, it is recommended that all input and output pins be constrained to the voltage range  $(\text{GND} < (V_{\text{IN}} \text{ or } V_{\text{OUT}}) < V_{\text{CC}})$ . All unused inputs should be tied to an appropriate logic level to minimize power consumption (do not leave them floating). A power supply decoupling capacitor of at least 0.2  $\mu\text{F}$  must be connected directly between each  $V_{\text{CC}}$  and GND pin.

As with all CMOS devices, ESD handling procedures should be used with the 5AC312 to prevent damage to the device during programming, assembly, and test.

## FUNCTIONAL TESTING

Since the logical operation of the 5AC312 is controlled by EPROM elements, the device is completely testable during the manufacturing process. Each programmable EPROM bit controlling the internal logic is tested using application-independent test patterns. EPROM cells in the 5AC312 are 100% tested for programming and erase. After testing, the devices are erased before shipments to the customers. No post-programming tests of the EPROM array are required.

The testability and reliability of EPROM-based programmable logic devices are important features over similar devices based on fuse technology. Fuse-based programmable logic devices require a user to perform post-programming tests to insure device functionality. During the manufacturing process, tests on these parts can only be performed in very restricted manners to prevent pre-programming of the array.



**INTEL PROGRAMMABLE LOGIC DEVELOPMENT SYSTEM II (iPLDS II)**

Release 2.0 of iPLDS II provides all the tools needed to design with the 5AC312 EPLD. In addition to providing development assistance, iPLDS II insulates the user from knowing the intricate details of EPLD architecture (the machine will optimize a design to benefit from architectural features). It contains comprehensive third generation software that supports four different design entry methods, minimizes logic, does automatic pin assignments and produces the best design fit for the selected EPLD. It is user friendly with guided menus, on-line Help messages and soft key inputs.

In addition, the iPLDS II contains programmer hardware in the form of an iUP-PC Universal Programmer-Personal Computer to enable the user to program EPLDs, read and verify programmed devices and also to graphically edit programming files. The software generates industry standard JEDEC object code output files which can be downloaded to other programmers as well.

The iPLDS II has interfaces to popular schematic capture packages to enable designs to be entered using schematics. A more integrated schematic entry method is provided by SCHEMA II-PLD, a low-cost schematic capture package that supports EPLD primitives and user-defined macro symbols. SCHEMA II-PLD contains the EPLD Design Manager, which provides a single user interface to both SCHEMA II-PLD and iPLS II software. The other design formats supported are Boolean equation entry and State Machine design entry.

The iPLDS operates on the IBM† PC/XT, PC/AT, or other compatible machine with the following configuration:

1. At least one floppy disk drive and hard disk drive.
2. MS-DOS†† Operating System Version 3.0 or greater.
3. 512K Memory (640K recommended).
4. Intel iUP-PC Universal Programmer-Personal Computer and GUPI Adaptor (supplied with iPLDS II)
5. A color monitor is suggested.

Detailed information on the Intel Programmable Logic Development System II is contained in a separate Intel data sheet. (Order Number: 280168)

†IBM Personal Computer is a registered trademark of International Business Machines Corporation.

††MS-DOS is a registered trademark of Microsoft Corporation.

**ADF PRIMITIVES SUPPORTED**

The following ADF primitives are supported by this device:

INP	NOTF
LINP	JOJF
RINP	JONF
CONF	SONF
COCF	SOSF
COIF	TOIF
RONF	TONF
ROIF	TOTF
RORF	CLKB
NOCF	LINB
NORF	
NOJF	
NOSF	

**ORDERING INFORMATION**

t <sub>PD</sub> (ns)	t <sub>CO</sub> (ns)	f <sub>MAX</sub> (MHz)	Order Code	Package	Operating Range
25	15	50	D5AC312-25	CERDIP	Commercial
			P5AC312-25	PDIP	
			N5AC312-25	PLCC	
30	18	40	D5AC312-30	CERDIP	Commercial
			P5AC312-30	PDIP	
			N5AC312-30	PLCC	
35	20	40	D5AC312-35	CERDIP	Commercial
			P5AC312-35	PDIP	
			N5AC312-35	PLCC	

**ABSOLUTE MAXIMUM RATINGS\***

Supply Voltage ( $V_{CC}$ ) (1) .....	-2.0V to +7.0V
Programming Supply Voltage ( $V_{pp}$ ) (1) .....	-2.0V to +13.5V
D.C. Input Voltage ( $V_I$ )(1, 2) ...	-0.5V to $V_{CC} + 0.5V$
Storage Temperature ( $T_{stg}$ ) .....	-65°C to +150°C
Ambient Temperature ( $T_{amb}$ ) (3) ..	-10°C to +85°C

\*Notice: Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

**NOTICE:** Specifications contained within the following tables are subject to change.

**NOTES:**

1. Voltages with respect to GND.
2. Minimum D.C. input is -0.5V. During transitions, the inputs may undershoot to -2.0V or overshoot to +7V for periods of less than 20 ns under no load conditions.
3. Under bias. Extended temperature range versions are available.

**RECOMMENDED OPERATING CONDITIONS**

Symbol	Parameter	Min	Max	Unit
$V_{CC}$	Supply Voltage	4.75	5.25	V
$V_{IN}$	Input Voltage	0	$V_{CC}$	V
$V_O$	Output Voltage	0	$V_{CC}$	V
$T_A$	Operating Temperature	0	+70	°C
$t_R$	Input Rise Time		500	ns
$t_F$	Input Fall Time		500	ns

**D.C. CHARACTERISTICS**  $T_A = 0^\circ\text{C to } +70^\circ\text{C}$ ,  $V_{CC} = 5.0V \pm 5\%$ 

Symbol	Parameter	Min	Typ	Max	Unit	Test Conditions
$V_{IH}^{(4)}$	High Level Input Voltage	2.0		$V_{CC} + 0.3$	V	
$V_{IL}^{(4)}$	Low Level Input Voltage	-0.3		0.8	V	
$V_{OH}^{(5)}$	High Level Output Voltage	2.4			V	$I_O = -4.0\text{ mA D.C.}, V_{CC} = \text{min.}$
$V_{OL}$	Low Level Output Voltage			0.45	V	$I_O = 8.0\text{ mA D.C.}, V_{CC} = \text{min.}$
$I_I$	Input Leakage Current			$\pm 10$	$\mu\text{A}$	$V_{CC} = \text{max.}, \text{GND} < V_{IN} < V_{CC}$
$I_{OZ}$	Output Leakage Current			$\pm 10$	$\mu\text{A}$	$V_{CC} = \text{max.}, \text{GND} < V_{OUT} < V_{CC}$
$I_{SC}^{(6)}$	Output Short Circuit Current	-30		-90	mA	$V_{CC} = \text{max.}, V_{OUT} = 0.5V$
$I_{SB}^{(7)}$	Standby Current		100	150	$\mu\text{A}$	$V_{CC} = \text{max.}, V_{IN} = V_{CC} \text{ or GND, Standby Mode}$
$I_{CC}^{(8)}$	Power Supply Current		10		mA	$V_{CC} = \text{max.}, V_{IN} = V_{CC} \text{ or GND, No Load, Input Freq.} = 1\text{ MHz Active Mode (Turbo} = \text{Off), Device Prog. as 12-Bit Ctr.}$

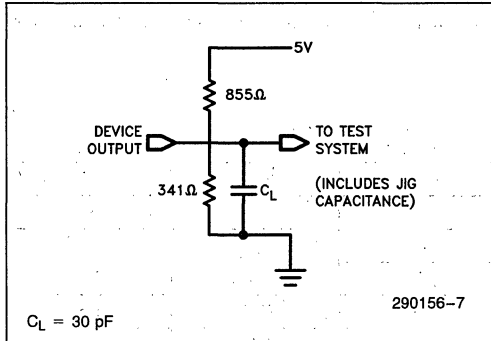
**NOTES:**

4. Absolute values with respect to device GND; all over and undershoots due to system or tester noise are included. Do not attempt to test these values without suitable equipment.
5.  $I_O$  at CMOS levels (3.84V) = -2 mA.
6. Not more than 1 output should be tested at a time. Duration of that test must not exceed 1 second.
7. With Turbo Bit Off, device automatically enters standby mode approximately 100 ns after last input transition.
8. See graph at end of data sheet for  $I_{CC}$  vs. frequency.

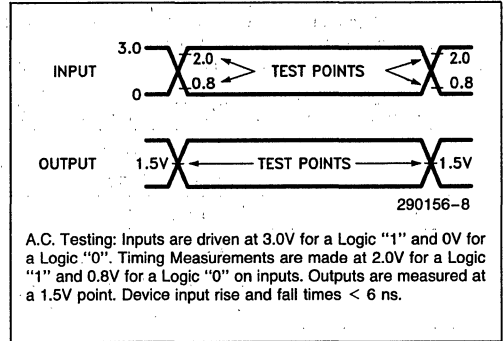
**CAPACITANCE**

Symbol	Parameter	Min	Typ	Max	Unit	Conditions
C <sub>IN</sub>	Input Capacitance			8	pF	V <sub>IN</sub> = 0V, f = 1.0 MHz
C <sub>OUT</sub>	I/O Capacitance			15	pF	V <sub>OUT</sub> = 0V, f = 1.0 MHz
C <sub>CLK</sub>	ILE/ICLK/INP2 Capacitance			12	pF	V <sub>IN</sub> = 0V, f = 1.0 MHz
C <sub>VPP</sub>	V <sub>PP</sub> Pin (CLK/INP1)			25	pF	V <sub>IN</sub> = 0V, f = 1.0 MHz

**A.C. TESTING LOAD CIRCUIT**



**A.C. TESTING INPUT, OUTPUT WAVEFORM**



**A.C. CHARACTERISTICS** T<sub>A</sub> = 0°C to +70°C, V<sub>CC</sub> = 5.0V ± 5%, Turbo Bit "On"<sup>(9)</sup>

Symbol	From	To	5AC312-25			5AC312-30			5AC312-35			Non-(11) Turbo Mode	Unit
			Min	Typ	Max	Min	Typ	Max	Min	Typ	Max		
t <sub>PD1</sub>	Input	Comb. Output		20	25		25	30		30	35	+20	ns
t <sub>PD2</sub>	I/O	Comb. Output		20	25		25	30		30	35	+20	ns
t <sub>PZX</sub> <sup>(10)</sup>	I or I/O	Output Enable		20	25		25	30		30	35	+20	ns
t <sub>PXZ</sub> <sup>(10)</sup>	I or I/O	Output Disable		20	25		25	30		30	35	+20	ns
t <sub>CLR</sub>	Asynch. Reset	Q Reset		20	25		25	30		30	35	+20	ns
t <sub>SET</sub>	Asynch. Set	Q Set		20	25		25	30		30	35	+20	ns

**NOTES:**

9. Typical values are at T<sub>A</sub> = 25°C, V<sub>CC</sub> = 5V, Active Mode.

10. t<sub>PZX</sub> and t<sub>PXZ</sub> are measured at ±0.5V from steady-state voltage as driven by spec. output load. t<sub>PXZ</sub> is measured with C<sub>L</sub> = 5 pF.

11. If device is operated with Turbo Bit Off (Non-Turbo Mode), increase time by amount shown.

**SYNCHRONOUS CLOCK MODE (MACROCELLS) A.C. CHARACTERISTICS**

T<sub>A</sub> = 0°C to +70°C, V<sub>CC</sub> = 5.0V ±5%, Turbo Bit On<sup>(8)</sup>

Symbol	Parameter	5AC312-25			5AC312-30			5AC312-35			Non-(11) Turbo Mode	Unit
		Min	Typ	Max	Min	Typ	Max	Min	Typ	Max		
f <sub>MAX</sub>	Max. Frequency (Pipelined) 1/t <sub>SU</sub> —No Feedback		66	50		50	40		50	40		MHz
f <sub>CNT</sub>	Max. Count Frequency 1/t <sub>CNT</sub> —with Feedback		40	33		35	30		28.5	25		MHz
t <sub>SU1</sub>	Input Setup Time to CLK	20	15		25	20		25	20		+ 20	ns
t <sub>SU2</sub>	I/O Setup Time to CLK	20	15		25	20		25	20		+ 20	ns
t <sub>H</sub>	I or I/O Hold after CLK High	0			0			0				ns
t <sub>CO</sub>	CLK High to Output Valid		10	15		12	18		15	20		ns
t <sub>CNT</sub>	Macrocell Output Feedback to Macrocell Input—Internal Path	30	25		35	30		40	35		+ 20	ns
t <sub>CH</sub>	CLK High Time	10			12.5			12.5				ns
t <sub>CL</sub>	CLK Low Time	10			12.5			12.5				ns

**SYNCHRONOUS CLOCK MODE (INPUT STRUCTURE) A.C. CHARACTERISTICS**

T<sub>A</sub> = 0°C to +70°C, V<sub>CC</sub> = 5.0V ±5%, Turbo Bit On<sup>(8)</sup>

Symbol	Parameter	5AC312-25			5AC312-30			5AC312-35			Non-(11) Turbo Mode	Unit
		Min	Typ	Max	Min	Typ	Max	Min	Typ	Max		
f <sub>MAXI</sub>	Max. Frequency		50	40		40	33		33	28.5		MHz
t <sub>SUIR</sub>	Input Register/Latch Setup Time before ILE/ICLK	5			5			5				ns
t <sub>PLI</sub> <sup>(12)</sup>	Minimum Input Clock Period		20	25		25	30		25	30	+ 20	ns
t <sub>HI</sub>	I Hold after ICLK/ILE ↓	7			10			12				ns
t <sub>COI</sub>	ICLK ↓ to Comb. Output		30	35		35	40		35	40	+ 20	ns
t <sub>EOI</sub>	ILE ↑ to Comb. Output		30	35		35	40		35	40	+ 20	ns
t <sub>CHI</sub>	ILE/ICLK High Time	10			12.5			12.5				ns
t <sub>CLI</sub>	ILE/ICLK Low Time	10			12.5			12.5				ns

**NOTE:**

12. t<sub>PLI</sub> = Input signal through registers/latch to macrocell register input.

**ASYNCHRONOUS CLOCK MODE A.C. CHARACTERISTICS**

T<sub>A</sub> = 0°C to +70°C, V<sub>CC</sub> = 5.0V ±5%, Turbo Bit On<sup>(8)</sup>

Symbol	Parameter	5AC312-25			5AC312-30			5AC312-35			Non-(10) Turbo Mode	Unit
		Min	Typ	Max	Min	Typ	Max	Min	Typ	Max		
<b>INPUT STRUCTURE</b>												
f <sub>AMAXI</sub>	Max. Frequency Input Register 1/(t <sub>ACLI</sub> + t <sub>ACHI</sub> )			50			40			40		MHz
t <sub>ASUI</sub>	Input Register/Latch Setup Time to Asynch. ILE/ICLK	0			0			0			+20	ns
t <sub>AHI</sub>	Input Register/Latch Hold after Asynch. ILE/ICLK	20	14		25	20		30	25			ns
t <sub>ACOI</sub>	Asynch. ICLK to Comb. Output		40	48		45	55		50	60	+20	ns
t <sub>AEOI</sub>	Asynch. ILE ↑ to Comb. Output		40	48		45	55		50	60	+20	ns
t <sub>ACHI</sub>	Asynch. ICLK High Time	10			12.5			12.5				ns
t <sub>ACLI</sub>	Asynch. ICLK Low Time	10			12.5			12.5				ns
<b>MACROCELLS</b>												
f <sub>AMAX</sub>	Max. Frequency (Pipelined) 1/(t <sub>ACL</sub> + t <sub>ACH</sub> )—No Feedback			50			40			40		MHz
f <sub>ACNT</sub>	Max. Frequency 1/t <sub>ACNT</sub> —with Feedback		40	33		35	30		28.5	25		MHz
t <sub>ASU1</sub>	Input Setup Time to Asynch. Clock	10			12			15			+20	ns
t <sub>ASU2</sub>	I/O Setup Time to Asynch. Clock	10			12			15			+20	ns
t <sub>AH</sub>	Input or I/O Hold after Asynch. Clock	5	0		5	0		5	0			ns
t <sub>ACO</sub>	Asynch. CLK to Output Valid		20	25		25	30		30	35	+20	ns
t <sub>ACNT</sub>	Register Output Feedback to Register Input— Internal Path	30	25		35	30		40	35		+20	ns
t <sub>ACH</sub>	Asynch. CLK High Time	10			12.5			12.5				ns
t <sub>ACL</sub>	Asynch. CLK Low Time	10			12.5			12.5				ns

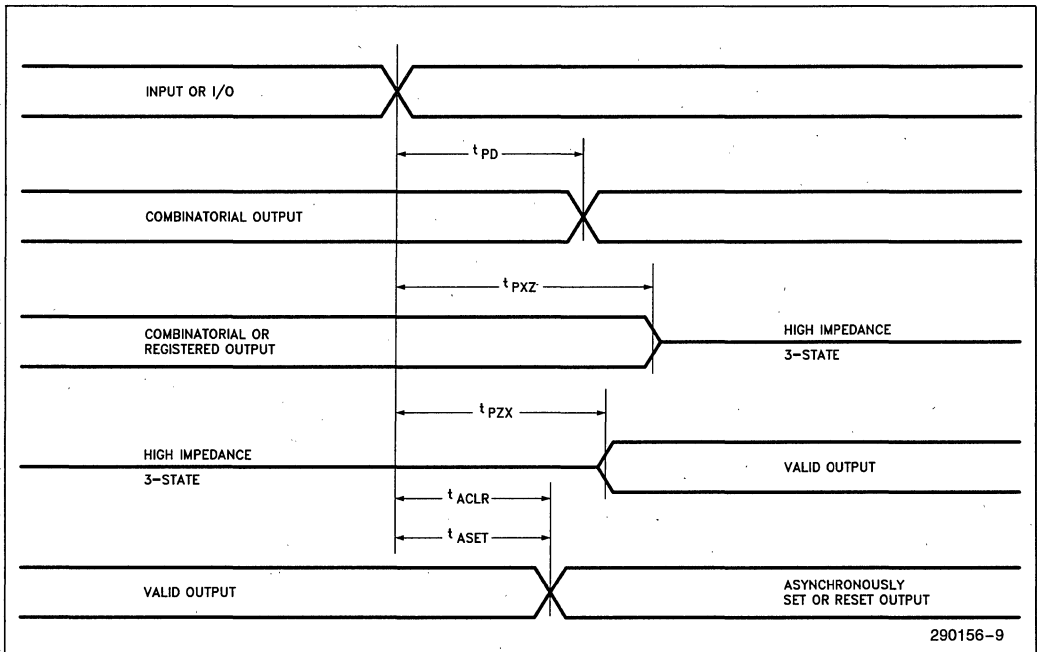
**INPUT-CLOCK-TO-MACROCELL-CLOCK A.C. CHARACTERISTICS**

T<sub>A</sub> = 0°C to +70°C, V<sub>CC</sub> = 5.0V ±5%, Turbo Bit On<sup>(8)</sup>

Symbol	Parameter	5AC312-25			5AC312-30			5AC312-35			Non-(10) Turbo Mode	Unit
		Min	Typ	Max	Min	Typ	Max	Min	Typ	Max		
t <sub>C1C2</sub>	Synchronous ILE/ICLK to Synchronous Macrocell CLK	25			30			35			+20	ns
	Synchronous ILE/ICLK to Asynchronous Macrocell CLK	15			18			20			+20	ns
	Asynchronous ILE/ICLK to Synchronous Macrocell CLK	35			40			45			+20	ns
	Asynchronous ILE/ICLK to Asynchronous Macrocell CLK	25			35			40			+20	ns

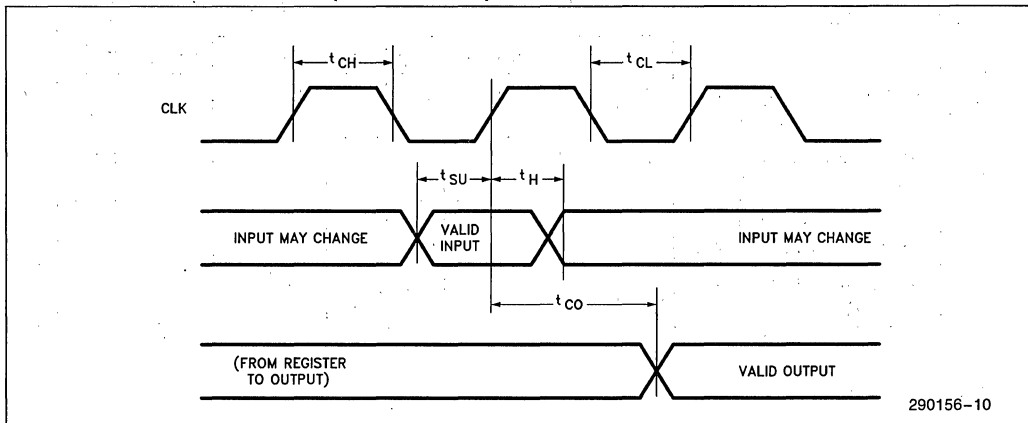
**SWITCHING WAVEFORMS**

**COMBINATORIAL MODE**

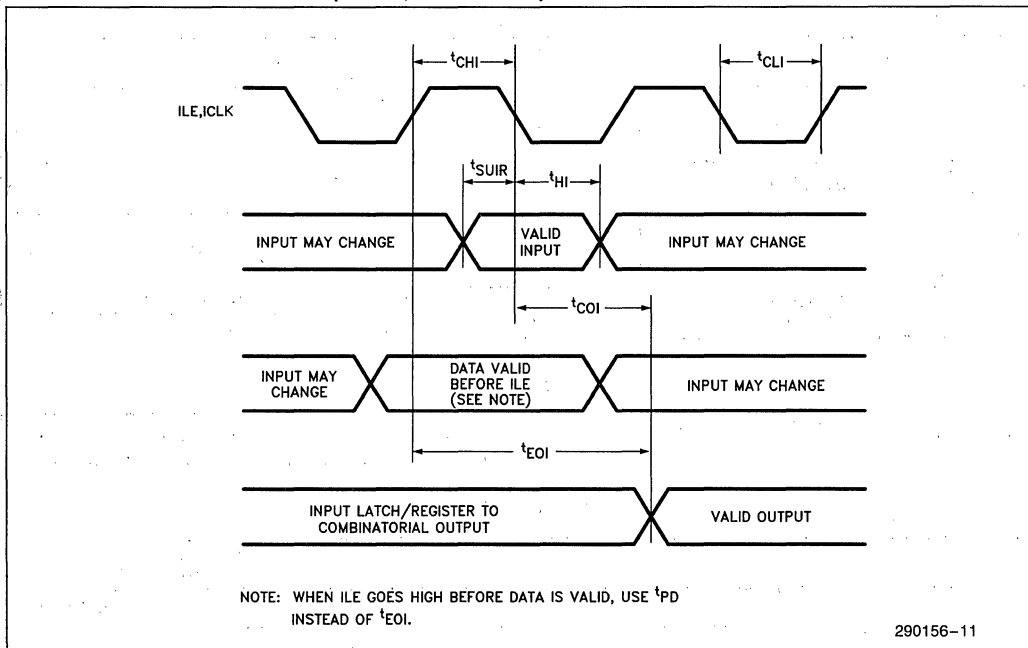


**SWITCHING WAVEFORMS** (Continued)

**SYNCHRONOUS CLOCK MODE (MACROCELLS)**

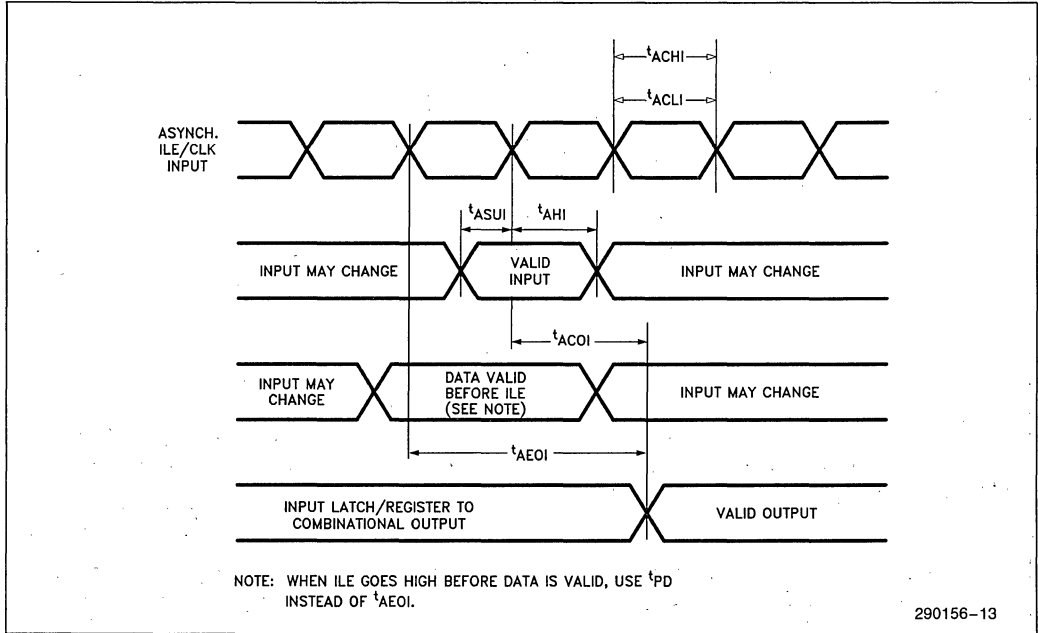


**SYNCHRONOUS CLOCK MODE (INPUT STRUCTURE)**

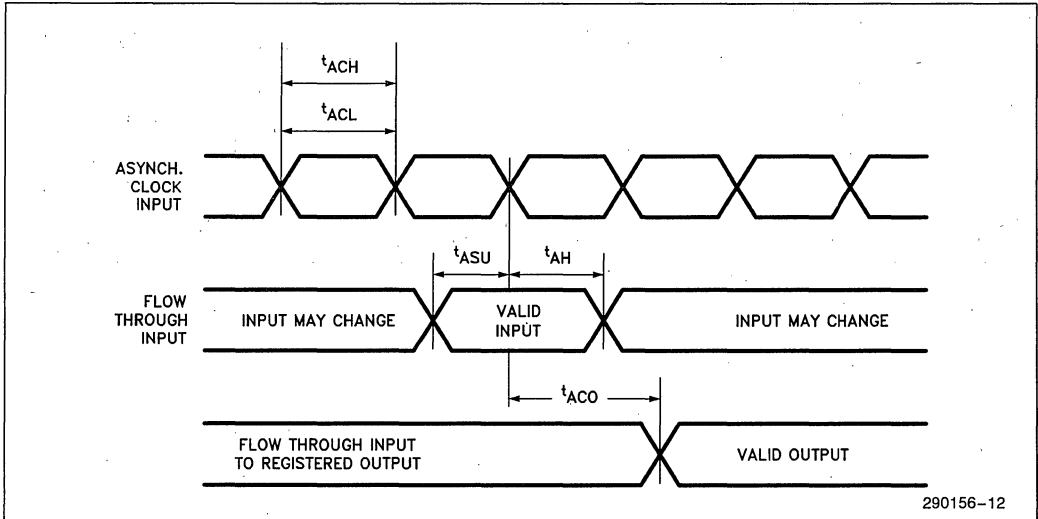


SWITCHING WAVEFORMS (Continued)

ASYNCHRONOUS CLOCK MODE (INPUT STRUCTURE)



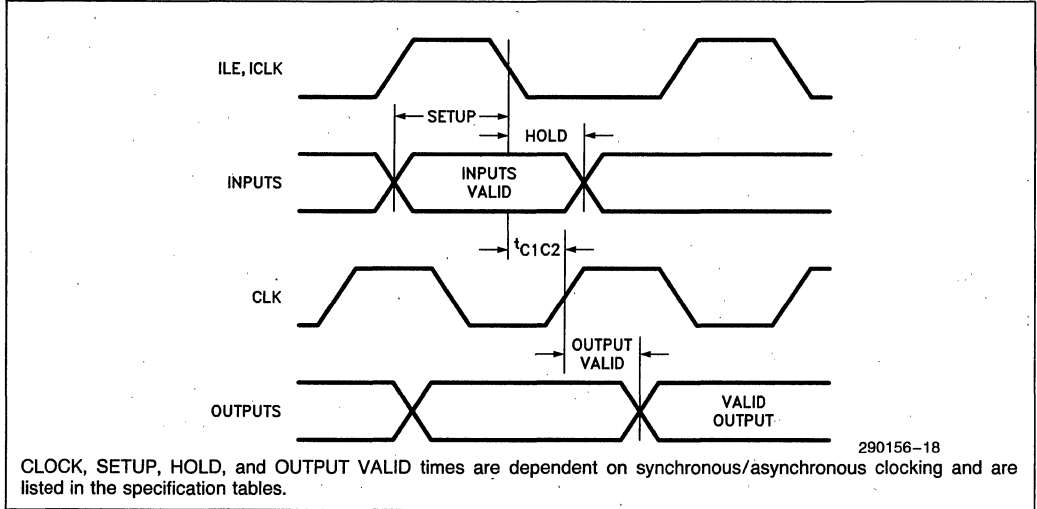
ASYNCHRONOUS CLOCK MODE (MACROCELLS)



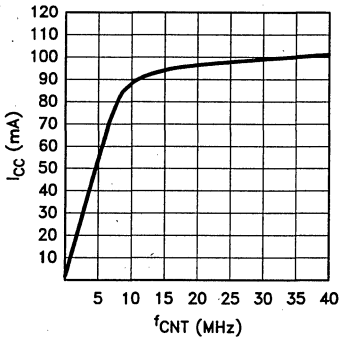


**SWITCHING WAVEFORMS** (Continued)

**INPUT CLOCK-TO-MACROCELL CLOCK TIMING (CLOCKED PIPELINED DATA)**

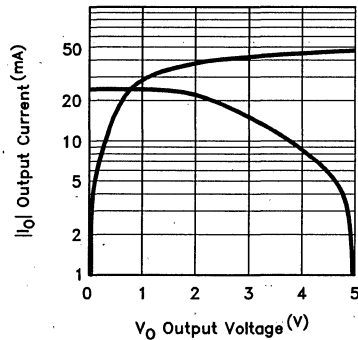


**Current in Relation to Frequency**



Conditions:  $T_A = 0^\circ\text{C}$ ,  $V_{CC} = 5.25\text{V}$  290156-20

**Output Drive Current in Relation to Voltage**

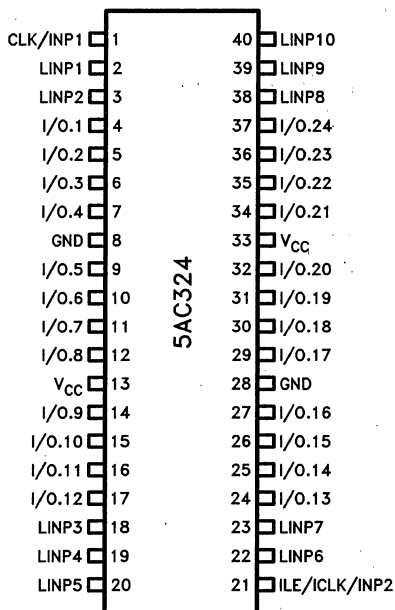


Conditions:  $T_A = +25^\circ\text{C}$  290156-16

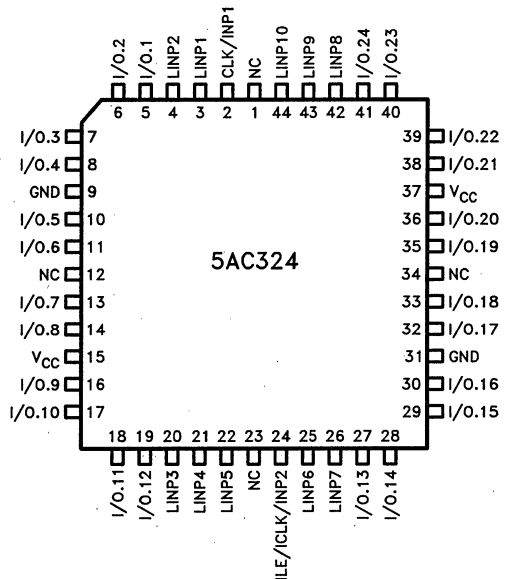
## 5AC324 1-MICRON CHMOS EPLD

- High-Performance LSI Semi-Custom Logic Alternative to Low-end Gate Arrays, TTL, and 74HC SSI and MSI Logic
- High Speed  $t_{pd}$  (max) 35 ns, 40 MHz Performance Pipelined, 25 MHz w/ Feedback
- 24 Macrocells with Programmable I/O Architecture; 10 Programmable Inputs; 1 Dedicated Input or Global CLK Pin; 1 Dedicated Input or Global ILE/ICLK Pin
- Programmable Inputs Configurable as Latches, Registers, or Flow-Through
- Software-Supported Product Term Allocation Between Adjacent Macrocells
- Dual Feedback on All Macrocells for Implementing Buried Registers with Bidirectional I/O
- 2 Product Terms on All Macrocell Control Signals
- Programmable Output Registers Configurable as D, T, JK, or SR Types
- Programmable Low-Power Option for "Stand-by" Operation; 150  $\mu$ A Typical Standby Current
- UV Eraseable EPROM Technology. 100% Generically Testable EPROM Logic Control Array
- Programmable Security Bit Allows 100% Protection of Proprietary Designs
- (Proposed) JEDEC Pinout
- Available in 40-pin DIP and 44-pin J-Leaded Chip Carrier Package (Ceramic and Plastic)

(See Packaging Spec., Order Number #231369)



290160-1



290160-2

Figure 1. 5AC324 Pinout Diagrams

**INTRODUCTION**

The Intel 5AC324 CHMOS EPLD (Erasable Programmable Logic Device) is a high integration device that overcomes the primary limitations of standard PLDs. Due to a proprietary I/O architecture and macrocell structure, the 5AC324 is capable of implementing high performance logic functions more effectively than previously possible. The 5AC324 can be used as an alternative to low-end gate arrays, multiple programmable logic devices, or LS-, HC-, or HCT SSI and MSI logic devices. Input and macrocell features for the 5AC324 are a superset of features offered on other PLD-type products.

The 5AC324 uses advanced CHMOS EPROM cells as logic control elements instead of poly-silicon fuses. This technology allows the device to operate at levels necessary in high performance systems while significantly reducing power consumption. Its programmable standby mode reduces power to near zero in applications where a slight speed loss is traded for power savings.

**ARCHITECTURE DESCRIPTION**

The architecture of the 5AC324 is based on the familiar "Sum-Of-Products" programmable AND, fixed OR structure. This structure is then surrounded by powerful, programmable macrocells and inputs. The 5AC324 can implement both combinatorial and sequential logic functions through a highly flexible macrocell and I/O structure. The architecture of the device supports both combinatorial-register and register-combinatorial-register forms of logic to easily accommodate state machine designs.

Figure 2 shows a global view of the 5AC324 architecture. The 5AC324 contains a total of 24 I/O programmable macrocells, 10 programmable input structures, and two clock inputs that can be programmed to function either as combinatorial inputs or clock inputs for the input structures and macrocells.

Each of the ten programmable inputs can be individually configured as a latch, register or flow-through

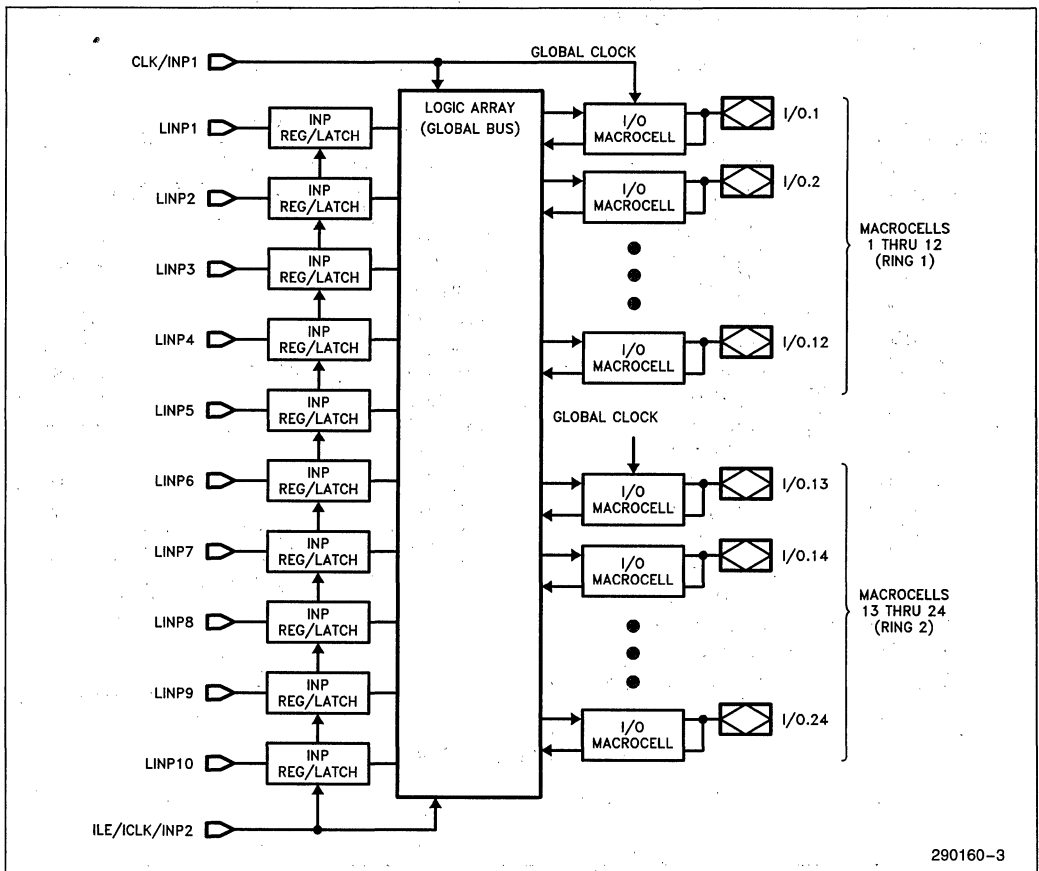


Figure 2. 5AC324 Global Architecture

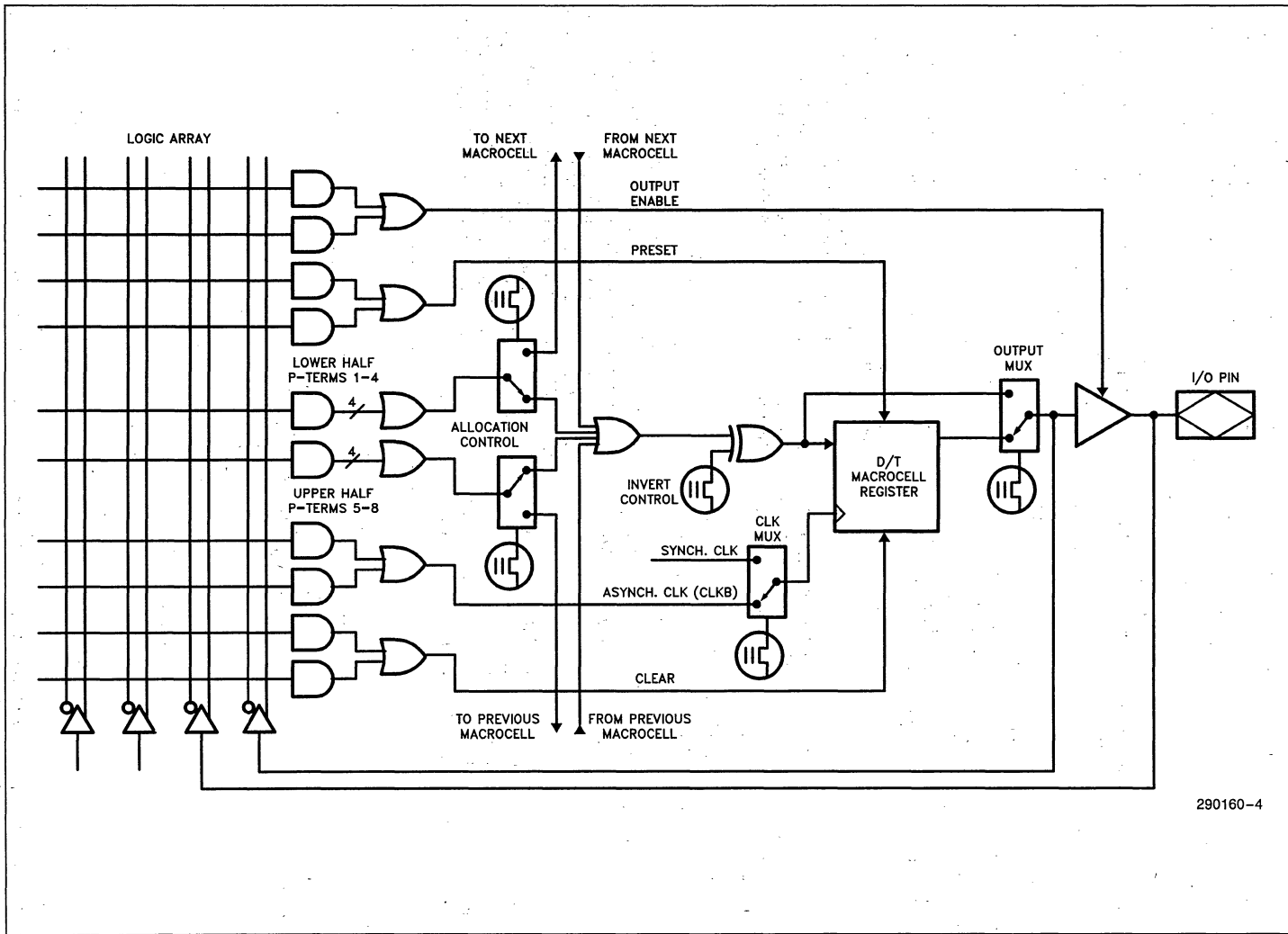


Figure 3. 5AC324 Macrocell Architecture

input. Input latches/registers can be synchronously or asynchronously clocked.

Figure 3 shows the basic architecture of each of the 24 macrocells in the 5AC324. Each macrocell contains 16 p-terms (product terms), with 8 p-terms available for the global array and 8 p-terms dedicated to the four control signals: OE, PRESET, CLEAR, and ASYNCH. CLK. The 8 p-terms from the logic array are organized as a user-programmable AND array and a user-configurable OR array. The inputs to the AND array originate from the true and complement signals from the programmable input structure, the dedicated inputs, and the 48 feedback paths from the 24 I/O macrocells to the global bus. This global bus simplifies designing with the device by eliminating the need to partition a circuit to fit into a local/global internal bus structure.

## INPUTS

Figure 4 shows a block diagram of the 5AC324 input structure. The device contains 10 user-programmable inputs that can be individually configured to operate in one of five modes:

- input register (D-register), synchronously clocked
- input register (D-register), asynchronously clocked
- input latch, (D-latch), synchronously clocked
- input latch, (D-latch), asynchronously clocked
- Flow-through input

Configuration is accomplished through the programming of EPROM architecture control bits via the logic compiler and programmer software. If synchronous operation is selected, the ILE/ICLK pin is used as a global latch/clock to all input latch/register

structures. For asynchronous operation, a separate product term in the array is used to derive the ILE/ICLK signal for each input structure. Because the clock signal for each programmable input can be individually selected, a mix between synchronously and asynchronously clocked inputs is possible. Software can configure each input structure as a flow-through input by selecting a latch and tying the ILE p-term to VCC. When ILE/ICLK is not used as a latch/clock, it functions as a dedicated input to the logic array. Data is latched/clocked on the falling edge of ILE/ICLK (synchronous mode).

## MACROCELLS

Each of the 24 macrocells in the device contains 8 p-terms to support logic functions and 8 p-terms for control signals. The 8 p-terms for logic functions are subdivided into 2 groups, each with 4 p-terms. This grouping of p-terms supports the proprietary p-term allocation scheme in the 5AC324. Each macrocell also provides dual feedbacks to the logic array, which results in more efficient macrocell/pin usage than possible with single feedbacks.

## Register Configuration

Each macrocell can be configured as a D, T, RS, or JK register. The 8 p-terms for control functions are organized so that 2 p-terms support each of the 4 control signals: Output Enable (OE), asynchronous I/O preset (PRESET), asynchronous I/O reset (CLEAR), and asynchronous I/O register clock (ASYNCH. CLK). Availability of 2 p-terms per control signal is another feature that increases the efficiency of the device by reducing the need to use intermediate macrocells sometimes needed to implement control functions.

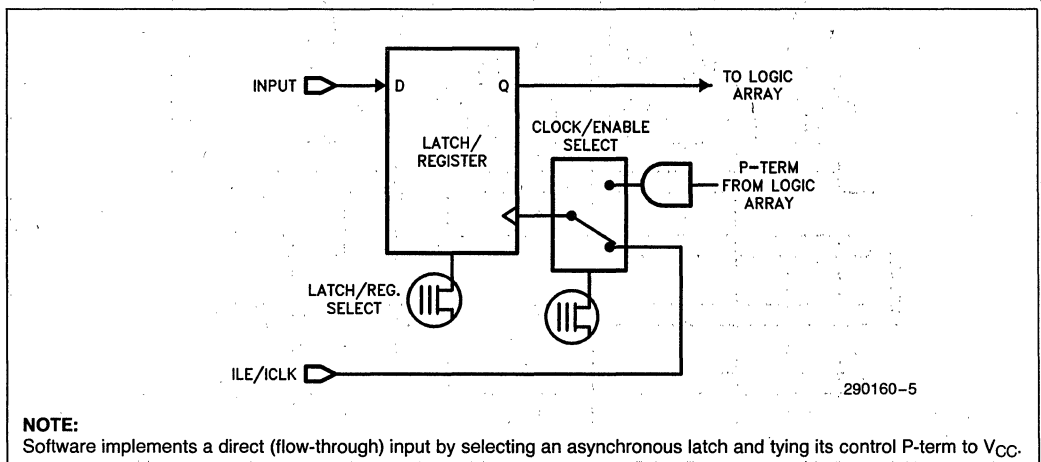


Figure 4. 5AC324 Programmable Input Structure

CLK is a global clock signal that can be used to synchronously clock any or all macrocell registers. When CLK is not used as a synchronous clock, it functions as a dedicated input to the logic array.

## Combinatorial Configuration

The macrocell register can be bypassed to implement combinatorial logic functions. When configured to provide combinatorial logic, only the OE control signal is used.

## Invert Select Bit

An invert select EPROM bit is used to invert the product term input into each macrocell register, including double inputs on JK and SR registers. This invert option allows the highest possible logic utilization by use of DeMorgan's logic inversion.

## LOGIC ARRAY

Each intersecting point in the logic array contains a programmable EPROM connection. Initially (erased state), all connections are complete, i.e., both true and complement states of all signals are connected to each p-term.

Connections are opened during programming. When both the true and complement connections exist, a logical false results on the output of the AND gate. If both the true and complement connections of a signal are programmed "open", then a logic "don't care" results for that signal. If all connections for a p-term are programmed open, then a logical true results on the output of the AND gate.

## PRODUCT TERM ALLOCATION

Product Term (p-term) allocation is defined as taking logic resources (p-terms) from macrocells where they are not used to support demand for additional p-terms in other macrocells. In the 5AC324, p-term allocation can occur in increments of 4 p-terms between adjacent macrocells. The 5AC324 includes 2 rings of 12 macrocells each. P-term groups from one macrocell can be allocated to the adjacent macrocell in the ring. P-term allocation between the two rings is not supported.

## EXAMPLE:

Figure 5 shows a p-term allocation example. In this example, the logic function in macrocell 4 requires 16 p-terms. In this case, software allocates 4 p-terms from the previous macrocell in Ring 1 (macrocell 5) and 4 p-terms from the next macrocell (macrocell 3) to accumulate a total of 16 p-terms ( $8 + 4 + 4$ ). This implementation leaves macrocells 3 and 5 with a remainder of 4 p-terms. These remaining p-terms can also be allocated away to, or supplemented with p-terms from, their adjacent macrocells in Ring 1 (macrocells 2 and 6).

With this scheme, any macrocell inside the device can support logic functions requiring between 0 and 16 p-terms. P-terms allocated away do not affect that macrocell's output structure. The input to the macrocell can be tied to VCC or GND, even when all p-terms have been allocated away. Thus the register and all control signals are still available for use if needed.

Figure 6 shows adjacent macrocells in the 5AC324. Table 1 shows the previous and next macrocells for each macrocell in the device, along with the corresponding allocation ring. P-term allocation is implemented automatically in the development software and is transparent to the user. Users can still use explicit pin assignment, but should assign pins in a way that does not conflict with p-term allocation.

Software support allows the control signals on macrocells to be used to implement simple logic functions even when all the input p-terms have been allocated to adjacent macrocells.

## DUAL-FEEDBACK/BURIED LOGIC

Macrocell output can be fed back to the logic array on either one of the two feedback paths. If the pin feedback is used (connected after the output buffer), bidirectional I/O can be implemented. If the internal feedback path is used to implement a buried register or buried logic function, the pin feedback is still available for use as an input. The availability of dual feedbacks on the 5AC324 enhances resource efficiency over single feedback devices.

## AUTOMATIC STAND-BY MODE

The 5AC324 contains a programmable bit, the Turbo Bit, that optimizes operation for speed or for power savings. When the Turbo Bit is programmed (TURBO = ON), the device is optimized for maximum

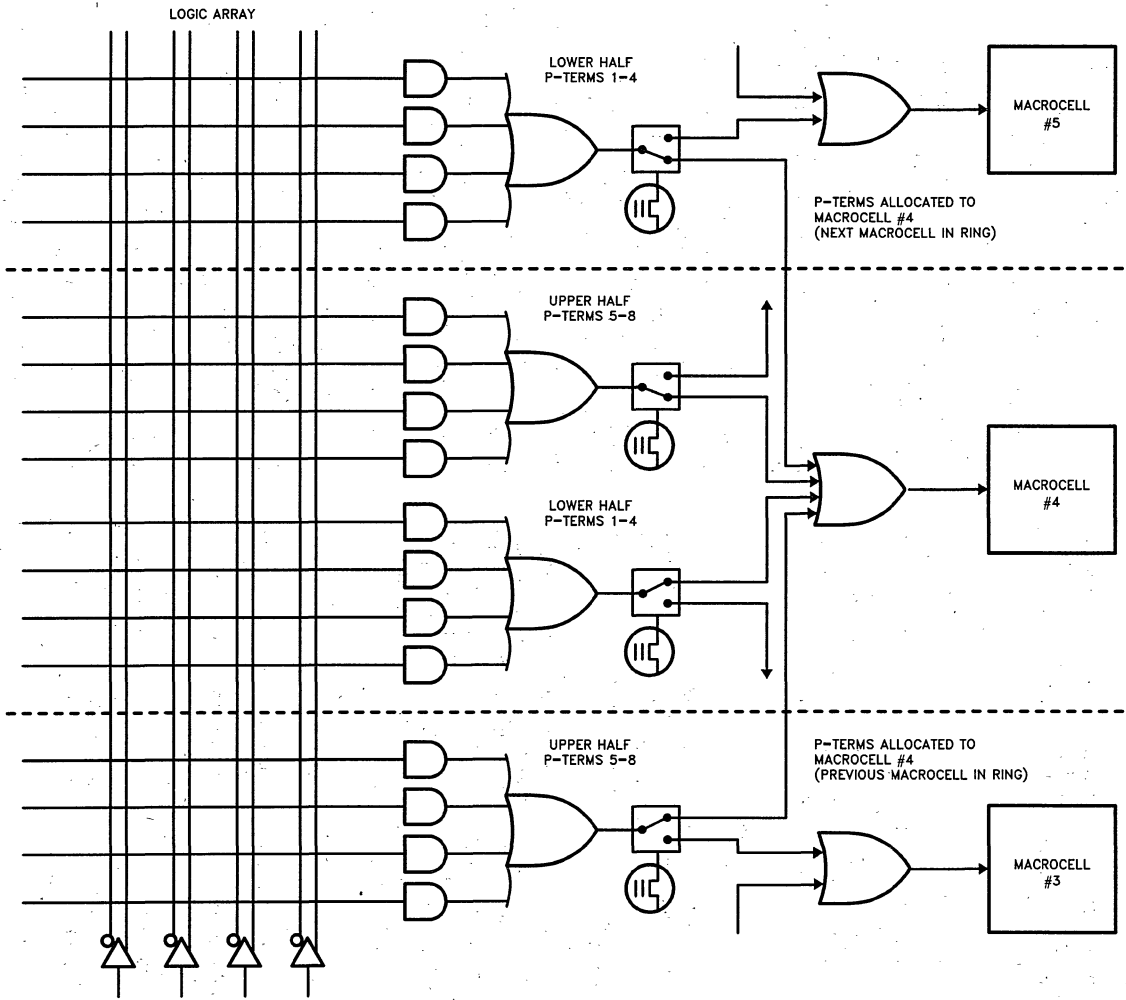


Figure 5. P-Term Allocation Example (8 + 4 + 4)

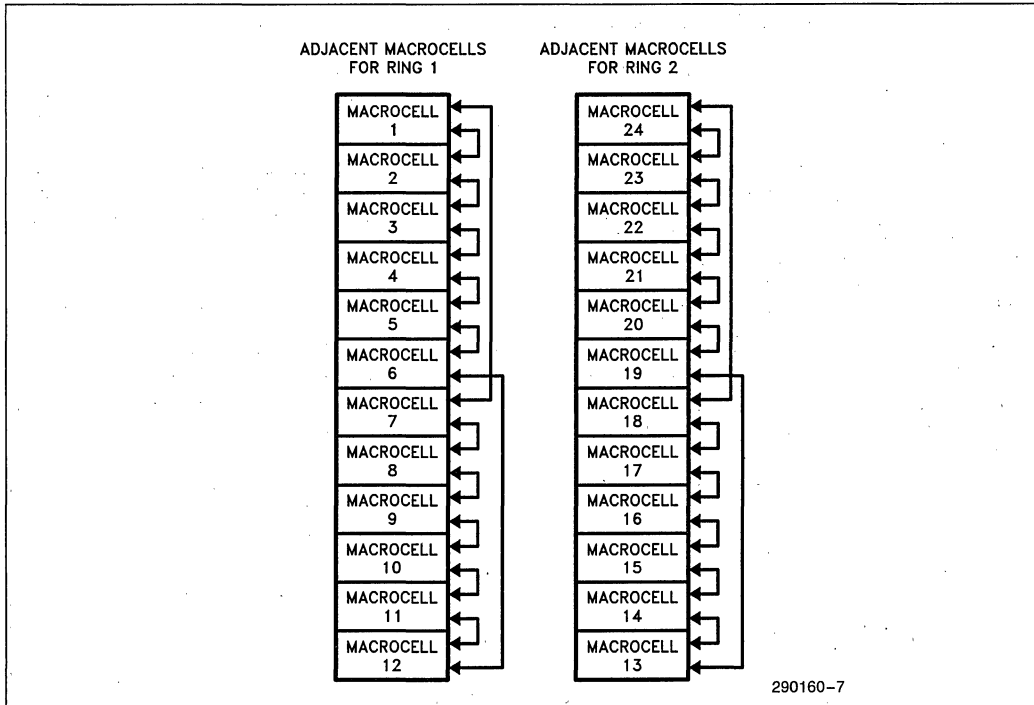


Figure 6. 5AC324 Adjacent Macrocell

Table 1. Product Term Allocation Rings

RING 1			RING 2		
Current Macrocell	Next Macrocell	Previous Macrocell	Current Macrocell	Next Macrocell	Previous Macrocell
1	7	2	13	19	14
2	1	3	14	13	15
3	2	4	15	14	16
4	3	5	16	15	17
5	4	6	17	16	18
6	5	12	18	17	24
7	8	1	19	20	13
8	9	7	20	21	19
9	10	8	21	22	20
10	11	9	22	23	21
11	12	10	23	24	22
12	6	11	24	18	23



speed. When the Turbo Bit is not programmed (TURBO = OFF), the device is optimized for power savings by entering standby mode during periods of inactivity.

Figure 7 shows the device entering standby mode approximately 100 ns after the last input transition. When the next input transition is detected, the device returns to active mode. Wakeup time adds an additional 15 ns to the propagation delay through the device as measured from the first input. No delay will occur if an output is dependent on more than one input and the last of the inputs changes after the device has returned to active mode.

After erasure, the Turbo Bit is unprogrammed (OFF); automatic standby mode is enabled. When the Turbo Bit is programmed (ON), the device never enters standby mode.

**POWER-ON CHARACTERISTICS**

On  $V_{CC}$  power-up, the 5AC324 registers are reset to a logic low. Input latch/register output (to the logic array) are also set to a logic low. 5AC324 inputs and outputs begin responding approximately 20  $\mu$ S after  $V_{CC}$  power-up or after a power-loss/power-up sequence. After power-up, macrocells can be preset to a logic high via the PRESET control signal for each macrocell.

**ERASED STATE CONFIGURATION**

After erasure and prior to programming, all macrocells are configured as combinatorial outputs with output buffers three-stated. Inputs are configured as synchronous registers.

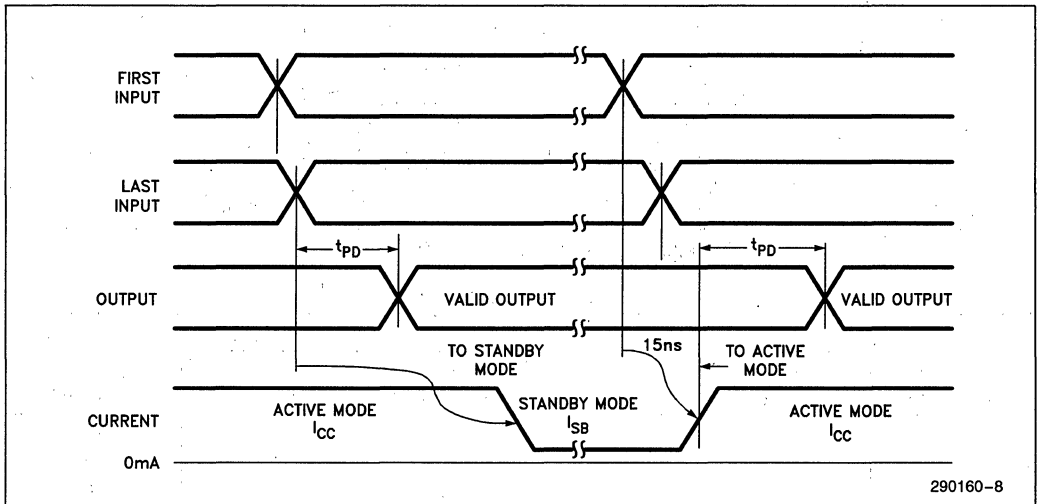


Figure 7. 5AC324 Standby and Active Mode Transitions

## ERASURE CHARACTERISTICS

Erasure time for the 5AC324 is 1 hour at 12,000 mW/cm<sup>2</sup> with a 2537Å UV lamp.

Erasure characteristics of the device are such that erasure begins to occur upon exposure to light with wavelengths shorter than approximately 4000Å. It should be noted that sunlight and certain types of fluorescent lamps have wavelengths in the 3000Å-4000Å range. Data shows that constant exposure to room level fluorescent lighting could erase the typical 5AC324 in approximately six years, while it would take approximately two weeks to erase the device when exposed to direct sunlight. If the device is to be exposed to these lighting conditions for extended periods of time, conductive opaque labels should be placed over the device window to prevent unintentional erasure.

The recommended erasure procedure for the 5AC324 is exposure to shortwave ultraviolet light with a wavelength of 2537Å. The integrated dose (i.e., UV intensity x exposure time) for erasure should be a minimum of fifteen (15) Wsec/cm<sup>2</sup>. The erasure time with this dosage is approximately 1 hour using an ultraviolet lamp with a 12,000 mW/cm<sup>2</sup> power rating. The device should be placed within 1 inch of the lamp tubes during exposure. The maximum integrated dose the 5AC324 can be exposed to without damage is 7258 Wsec/cm<sup>2</sup> (1 week at 12,000 μW/cm<sup>2</sup>). Exposure to high intensity UV light for longer periods may cause permanent damage to the device.

## intelligent Programming™ Algorithm

The 5AC324 supports the intelligent Programming Algorithm, which rapidly programs Intel EPLDs, and many of Intel's microcontrollers and EPROMs while maintaining a high degree of reliability. It is particularly suited for production programming environments. This method decreases the overall programming time while programming reliability is ensured as the incremental programming margin of each bit has been verified during programming. Programming voltage and waveform specifications are available by request from Intel to support programming the device.

## DESIGN SECURITY

A Security Bit provides a programmable security option to protect the data programmed in the device. Once this bit is set during programming, subsequent

attempts to read the device architecture information are prevented. This method provides a higher degree of design security than fused-based devices, since programmed EPROM cells are invisible even to microscopic examination. The Security Bit (also called the Verify Protect Bit), along with all the other EPROM cells, is reset by erasing the device.

## LATCH-UP IMMUNITY

All of the input, I/O, and clock pins of the device have been designed to resist latch-up which is inherent in inferior CMOS structures. The 5AC324 is designed with Intel's proprietary 1-micron CHMOS EPROM process. Thus, each of the pins will not experience latch-up with currents up to 100 mA and voltages ranging from -0.5V to V<sub>CC</sub> + 0.5V. The programming pin is designed to resist latch-up to the 13.5V maximum device limit.

## DESIGN RECOMMENDATIONS

For proper operation, it is recommended that all input and output pins be constrained to the voltage range GND < (V<sub>IN</sub> or V<sub>OUT</sub>) < V<sub>CC</sub>. All unused inputs should be tied to an appropriate logic level to minimize power consumption (do not leave them floating). A power supply decoupling capacitor of at least 0.2μF must be connected directly between each V<sub>CC</sub> and GND pin.

As with all CMOS devices, ESD handling procedures should be used with the 5AC324 to prevent damage to the device during programming, assembly, and test.

## FUNCTIONAL TESTING

Since the logical operation of the 5AC324 is controlled by EPROM elements, the device is completely testable during the manufacturing process. Each programmable EPROM bit controlling the internal logic is tested using application independent test patterns. EPROM cells in the device are 100% tested for programming and erasure. After testing, the devices are erased before shipments to the customers. No post-programming tests of the EPROM array are required.

The testability and reliability of EPROM-based programmable logic devices is an important feature over similar devices based on fuse technology. Fuse-based programmable logic devices require a user to perform post-programming tests to insure

device functionality. During the manufacturing process, tests on fuse-based parts can only be performed in very restricted ways in order to avoid pre-programming the array.

## DESIGN SOFTWARE

Contact your local Intel sales office for evaluation software to get you started with 5AC324 designs. The evaluation software is a proprietary version of iPLS II (Intel Programmable Logic Software II) that

will compile 5AC324 designs and produce a Logic Equation File (LEF) and a Report File. No JEDEC file is produced.

Full iPLS II support (including JEDEC generation capability) is provided by Version 2.0 of iPLS II, which will be available during the second half of 1988. iPLS II includes the LOC (Logic Optimizing Compiler), and LPS (Logic Programming Software).

## ORDERING INFORMATION

$t_{PD}$ (ns)	$t_{CO}$ (ns)	$f_{MAX}$ (MHz)	Order Code	Package	Operating Range
35	20	40	N5AC324-35	PLCC	Commercial
			P5AC324-35	PDIP	
			CJ5AC324-35	J LEAD CHIP CARRIER	
			D5AC324-35	CERDIP	
40	25	33	N5AC324-40	PLCC	Commercial
			P5AC324-40	PDIP	
			CJ5AC324-40	J LEAD CHIP CARRIER	
			D5AC324-40	CERDIP	

**ABSOLUTE MAXIMUM RATINGS\***

Supply Voltage ( $V_{CC}$ )<sup>(1)</sup> ..... -2.0V to +7.0V  
 Programming Supply  
 Voltage ( $V_{PP}$ )<sup>(1)</sup> ..... -2.0V to +13.5V  
 D.C. Input Voltage ( $V_i$ )<sup>(1,2)</sup> .... -0.5V to  $V_{CC} + 0.5V$   
 Storage Temperature ( $T_{sig}$ ) ..... -65°C to +150°C  
 Ambient Temperature ( $T_{amb}$ )<sup>(3)</sup> ..... -10°C to +85°C

\*Notice: Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

**NOTES:**

1. Voltage with respect to GND.
2. Minimum D.C. input is -0.5V. During transitions, the inputs may undershoot to -2.0V for periods of less than 20 ns under no load conditions.
3. Under bias. Extended Temperature versions are also available.

**NOTICE:** Specifications contained within the following tables are subject to change.

**RECOMMENDED OPERATING CONDITIONS**

Symbol	Parameter	Min	Max	Unit
$V_{CC}$	Supply Voltage	4.75	5.25	V
$V_{IN}$	Input Voltage	0	$V_{CC}$	V
$V_O$	Output Voltage	0	$V_{CC}$	V
$T_A$	Operating Temperature	0	+70	°C
$t_R$	Input Rise Time		500	ns
$t_F$	Input Fall Time		500	ns

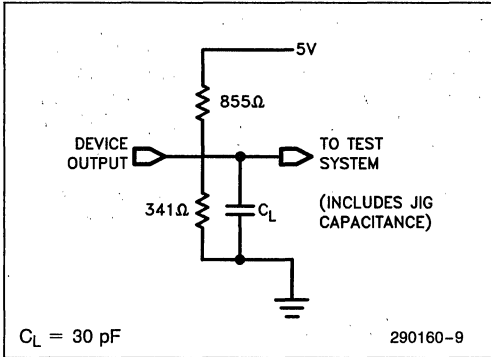
**D.C. CHARACTERISTICS** ( $T_A = 0^\circ C$  to  $+70^\circ C$ ,  $V_{CC} = 5.0V \pm 5\%$ )

Symbol	Parameter	Min	Typ	Max	Unit	Test Conditions
$V_{IH}^{(4)}$	High Level Input Voltage	2.0		$V_{CC} + 0.3$	V	
$V_{IL}^{(4)}$	Low Level Input Voltage	-0.3		0.8		
$V_{OH}^{(5)}$	High Level Output Voltage	2.4			V	$I_O = -4.0$ mA D.C., $V_{CC} = \text{min.}$
$V_{OL}$	Low Level Output Voltage			0.45	V	$I_O = 4.0$ mA D.C., $V_{CC} = \text{min.}$
$I_i$	Input Leakage Current			$\pm 10$	$\mu A$	$V_{CC} = \text{max.},$ $GND < V_{IN} < V_{CC}$
$I_{OZ}$	Output Leakage Current			$\pm 10$	$\mu A$	$V_{CC} = \text{max.},$ $GND < V_{OUT} < V_{CC}$
$I_{SC}^{(6)}$	Output Short Circuit Current	-30		-90	mA	$V_{CC} = \text{max.}, V_{OUT} = 0.5V$
$I_{SB}^{(7)}$	Standby Current		150		$\mu A$	$V_{CC} = \text{max.}, V_{IN} = V_{CC}$ or GND, Standby Mode
$I_{CC}$	Power Supply Current		50		mA	$V_{CC} = \text{max.}, V_{IN} = V_{CC}$ or GND, No Load, $f_{IN} = 1$ MHz, Active Mode (Turbo Off), Device Prog. as Two 12-Bit Counters

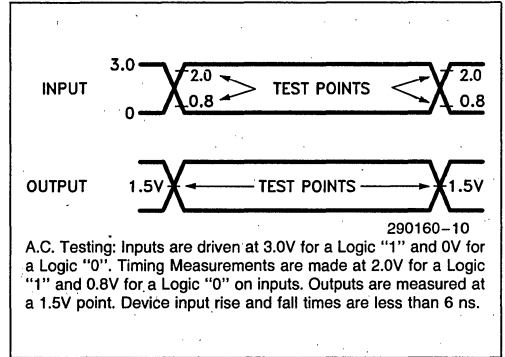
**NOTES:**

4. Absolute values with respect to device GND; all over and undershoots due to system or tester noise are included.
5.  $I_O$  at CMOS levels (3.84V) = -2 mA.
6. Not more than 1 output should be tested at a time. Duration of that test should not exceed 1 second.
7. With Turbo Bit Off, device automatically enters standby mode approximately 100 ns after last input transition.

**A.C. TESTING LOAD CIRCUIT**



**A.C. TESTING INPUT, OUTPUT WAVEFORM**



**CAPACITANCE**

Symbol	Parameter	Min	Typ	Max	Unit	Conditions
$C_{IN}$	Input Capacitance		5	20	pF	$V_{IN} = 0V, f = 1.0 \text{ MHz}$
$C_{OUT}$	Output Capacitance		10	20	pF	$V_{OUT} = 0V, f = 1.0 \text{ MHz}$
$C_{CLK}$	Clock Pin Capacitance		10	20	pF	$V_{OUT} = 0V, f = 1.0 \text{ MHz}$
$C_{VPP}$	$V_{PP}$ Pin Capacitance		20	40	pF	$V_{PP}$ on LIN3

**COMBINATORIAL MODE A.C. CHARACTERISTICS**
 $(T_A = 0^\circ\text{C to } +70^\circ\text{C}, V_{CC} = 5.0\text{V} \pm 5\%, \text{ Turbo Bit On})^{(8)}$ 

Symbol	Parameter	5AC324-35			5AC324-40			Non-Turbo <sup>(9)</sup> Mode	Unit
		Min	Typ	Max	Min	Typ	Max		
$t_{PD}$	Input or I/O to Output		30	35		35	40	+ 15	ns
$t_{PZX}^{(10)}$	Input or I/O to Output Enable		30	35		35	40	+ 15	ns
$t_{PXZ}^{(10)}$	Input or I/O to Output Disable		30	35		35	40	+ 15	ns
$t_{CLR}$	Asynch. Reset to Q Clear		30	35		35	40	+ 15	ns
$t_{SET}$	Asynch. Set to Q Set		30	35		35	40	+ 15	ns

**NOTES:**

8. Typical values are at  $T_A = +25^\circ\text{C}$ ,  $V_{CC} = 5\text{V}$ , Active Mode.

9. If device is operated with Turbo bit Off (Non-Turbo Mode), increase time by amount shown.

10.  $t_{PZX}$  and  $t_{PXZ}$  measured at  $\pm 0.5\text{V}$  from steady-state voltage as driven by spec. output load.  $t_{PXZ}$  measured with  $C_L = 5\text{ pF}$ .

**SYNCHRONOUS CLOCK MODE (MACROCELLS) A.C. CHARACTERISTICS**
 $(T_A = 0^\circ\text{C to } +70^\circ\text{C}, V_{CC} = 5.0\text{V} \pm 5\%, \text{ Turbo Bit On})^{(8)}$ 

Symbol	Parameter	5AC324-35			5AC324-40			Non-Turbo <sup>(9)</sup> Mode	Unit
		Min	Typ	Max	Min	Typ	Max		
$f_{MAX}$	Maximum Frequency (1/ $t_{SU}$ ) No Feedback		50	40		40	33.3	(11)	MHz
$f_{CNT}$	Maximum Frequency (1/ $t_{CNT}$ ) With Feedback		28.5	25		25	22.2	(11)	MHz
$t_{SU1}$	Input Setup Time to CLK $\uparrow$	25	20		30	25		+ 15	ns
$t_{SU2}$	I/O Setup Time to CLK $\uparrow$	25	20		30	25		+ 15	ns
$t_H$	Input or I/O Hold Time from CLK $\uparrow$	0			0				ns
$t_{CO}$	CLK $\uparrow$ to Output Valid		15	20		20	25	+ 15	ns
$t_{CNT}$	Register Output Feedback to Register Input—Internal Path	40	35		45	40		+ 15	ns
$t_{CH}$	Clock High Time	12.5			15			+ 15	ns
$t_{CL}$	Clock Low Time	12.5			15			+ 15	ns
$t_{CW}$	Minimum Clock Width	25			30			+ 15	ns

**NOTE:**

11. Recalculate frequency according to expression at left of table.

**SYNCHRONOUS CLOCK MODE (INPUT STRUCTURE) A.C. CHARACTERISTICS**(T<sub>A</sub> = 0°C to +70°C, V<sub>CC</sub> = 5.0V ±5%, Turbo Bit On)<sup>(8)</sup>

Symbol	Parameter	5AC324-35			5AC324-40			Non-Turbo <sup>(9)</sup> Mode	Unit
		Min.	Typ	Max	Min	Typ	Max		
f <sub>MAXI</sub>	Maximum Frequency (1/t <sub>CWI</sub> )		50	40		40	33.3	(11)	MHz
t <sub>SUIR</sub>	Input Register Setup Time Before ICLK ↓	5			5				ns
t <sub>ESUI</sub>	Input Latch Setup Time Before ILE ↑	5			5				ns
t <sub>COI</sub>	ICLK ↓ to Comb. Output		30	35		40	45	+ 15	ns
t <sub>HI</sub>	Input Hold after ICLK/ILE ↓	5			5				ns
t <sub>EOI</sub>	ILE ↑ to Comb. Output		35	40		45	50	+ 15	ns
t <sub>CHI</sub>	ILE/ICLK High Time	12.5			15			+ 15	ns
t <sub>CLI</sub>	ILE/ICLK Low Time	12.5			15			+ 15	ns
t <sub>CWI</sub>	Minimum Input Clock Width	25			30			+ 15	ns

**ASYNCHRONOUS CLOCK MODE (MACROCELLS) A.C. CHARACTERISTICS**(T<sub>A</sub> = 0°C to +70°C, V<sub>CC</sub> = 5.0V ±5%, Turbo Bit On)<sup>(8)</sup>

Symbol	Parameter	5AC324-35			5AC324-40			Non-Turbo <sup>(9)</sup> Mode	Unit
		Min	Typ	Max	Min	Typ	Max		
f <sub>AMAX</sub>	Max. Frequency (1/t <sub>ACL</sub> + t <sub>ACH</sub> ) No Feedback		20	16.6		16.5	14.2	(11)	MHz
f <sub>ACNT</sub>	Max. Frequency (1/t <sub>ACNT</sub> ) With Feedback		15.3	14.2		14.2	13.3	+ 15	MHz
t <sub>ASU1</sub>	Input Setup Time to Asynch. CLK	10			12.5			+ 15	ns
t <sub>ASU2</sub>	I/O Setup Time to Asynch. CLK	10			12.5			+ 15	ns
t <sub>AH</sub>	Input or I/O Hold Time from Asynch. CLK	30	25		35	40		+ 15	ns
t <sub>ACO</sub>	Asynch. CLK to Output Valid		45	50		50	55	+ 15	ns
t <sub>ACNT</sub>	Asynch. Output Feedback to Register Input - Internal Path	70	65		75	70		+ 15	ns
t <sub>ACH</sub>	Asynch. CLK High Time	30			35			+ 15	ns
t <sub>ACL</sub>	Asynch. CLK Low Time	30			35			+ 15	ns
t <sub>ACW</sub>	Asynch. CLK Width	60			70			+ 15	ns

**ASYNCHRONOUS CLOCK MODE (INPUT STRUCTURE) A.C. CHARACTERISTICS**

(T<sub>A</sub> = 0°C to +70°C, V<sub>CC</sub> = 5.0V ±5%, Turbo Bit On)<sup>(8)</sup>

Symbol	Parameter	5AC324-35			5AC324-40			Non-Turbo <sup>(9)</sup> Mode	Unit
		Min	Typ	Max	Min	Typ	Max		
f <sub>AMAXI</sub>	Maximum Frequency Input Register (1/t <sub>ACWI</sub> )		25	22.2		23	20	(11)	MHz
t <sub>ASUIR</sub>	Input Register Setup Time Before Asynch. ICLK	0			0				ns
t <sub>AESUI</sub>	Input Latch Setup Time Before Asynch. ILE	0			0				ns
t <sub>ACOI</sub>	Asynch. ICLK to Comb. Output		50	55		55	60	+ 15	ns
t <sub>AHI</sub>	Input Hold after Asynch. ICLK/ILE	20			25				ns
t <sub>AEOI</sub>	Asynch. ILE to Comb. Output		35	40		45	50	+ 15	ns
t <sub>ACHI</sub>	Asynch. ILE/ICLK High Time	22.5			25			+ 15	ns
t <sub>ACLI</sub>	Asynch. ILE/ICLK Low Time	22.5			25			+ 15	ns
t <sub>ACWI</sub>	Minimum Input Clock Width	45			50			+ 15	ns

**INPUT-CLOCK-TO-MACROCELL-CLOCK A.C. CHARACTERISTICS**

(T<sub>A</sub> = 0°C to +70°C, V<sub>CC</sub> = 5.0V ±5%, Turbo Bit On)<sup>(8)</sup>

Symbol	Parameter	5AC324-35			5AC324-40			Non-Turbo <sup>(9)</sup> Mode	Unit
		Min	Typ	Max	Min	Typ	Max		
t <sub>C1C2</sub> <sup>(12)</sup>	Synchronous ILE/ICLK Synchronous Macrocell CLK	30			35			+ 15	ns
	Synchronous ILE/ICLK Asynchronous Macrocell CLK	10			20			+ 15	ns
	Asynchronous ILE/ICLK Synchronous Macrocell CLK	45			55			+ 15	ns
	Asynchronous ILE/CLK Asynchronous Macrocell CLK	30			40			+ 15	ns

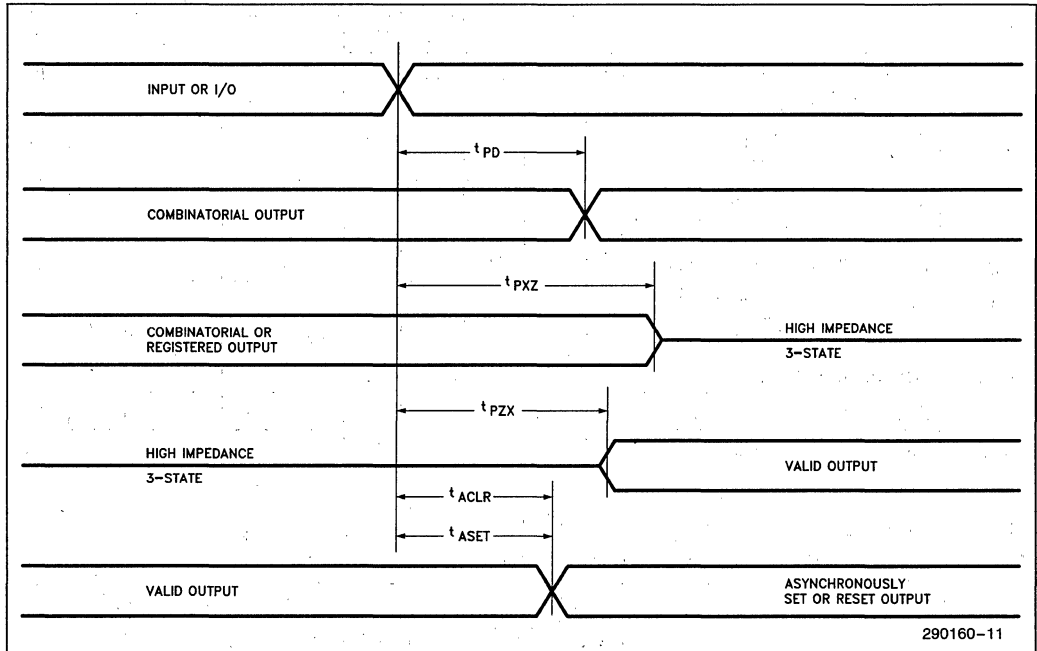
**NOTE:**

12. Times for SETUP, HOLD, and OUTPUT VALID are shown in previous tables.

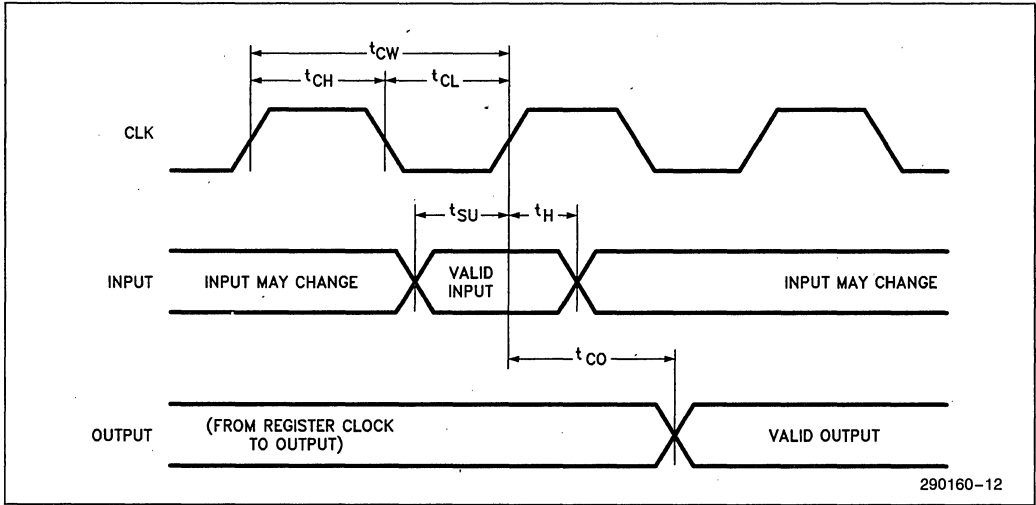


SWITCHING WAVEFORMS

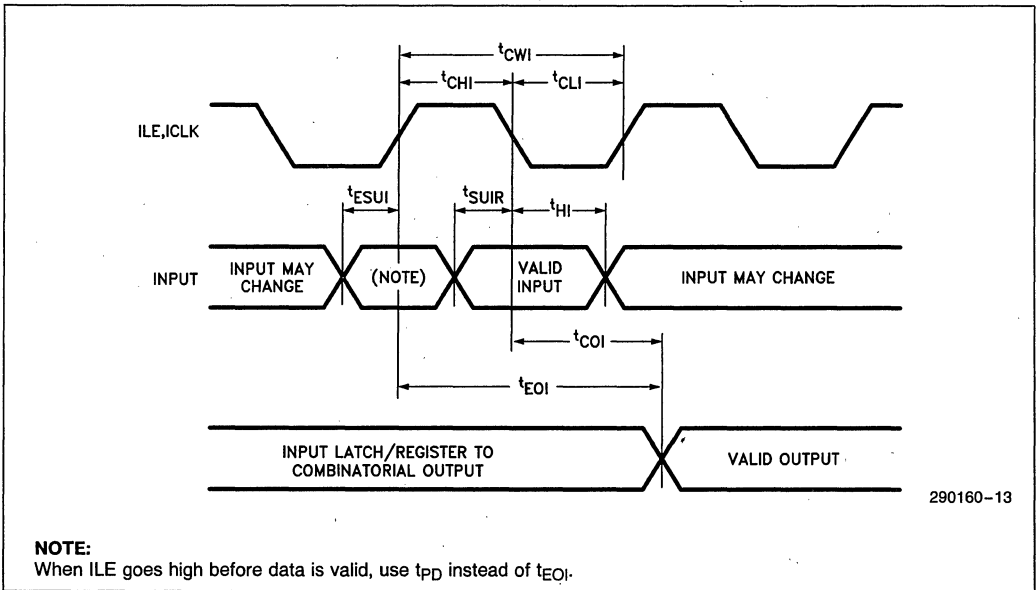
COMBINATORIAL MODE



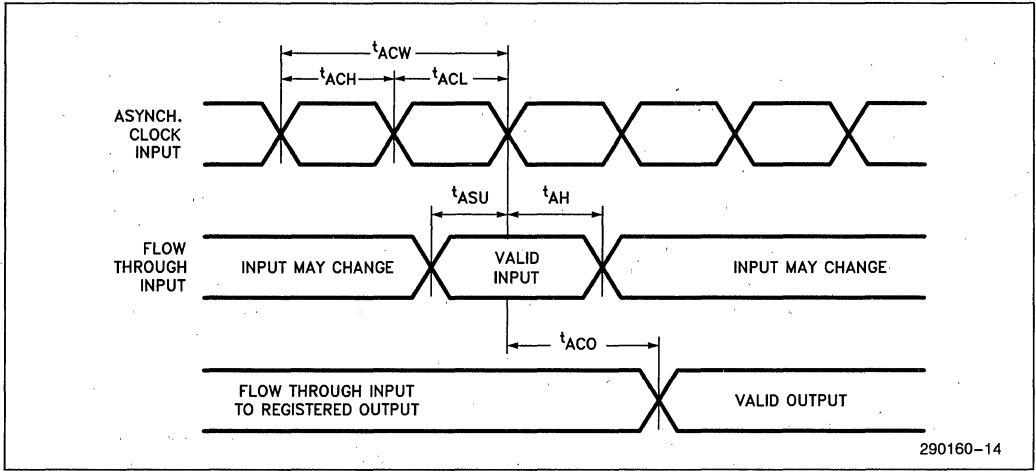
**SYNCHRONOUS CLOCK MODE (MACROCELLS)**



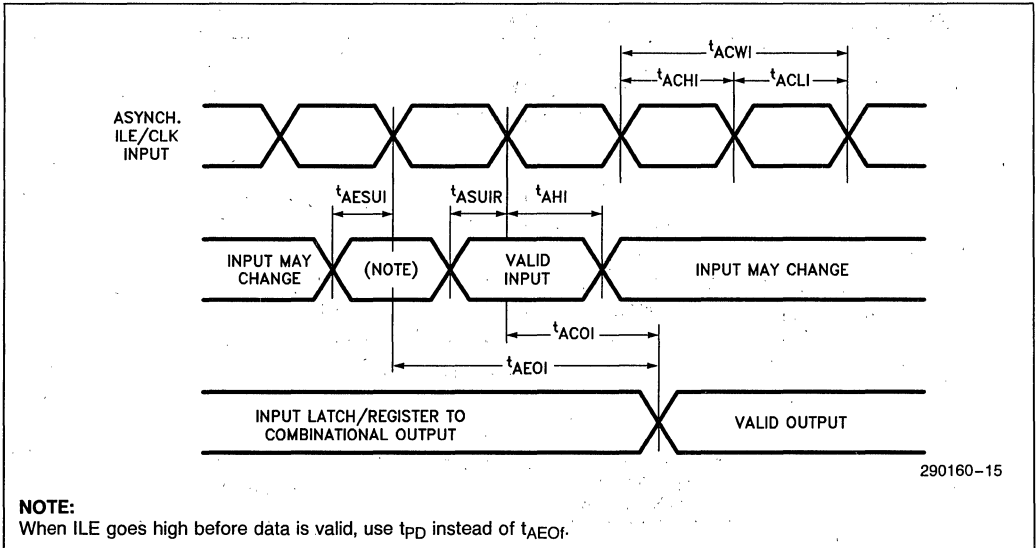
**SYNCHRONOUS CLOCK MODE (INPUT STRUCTURE)**



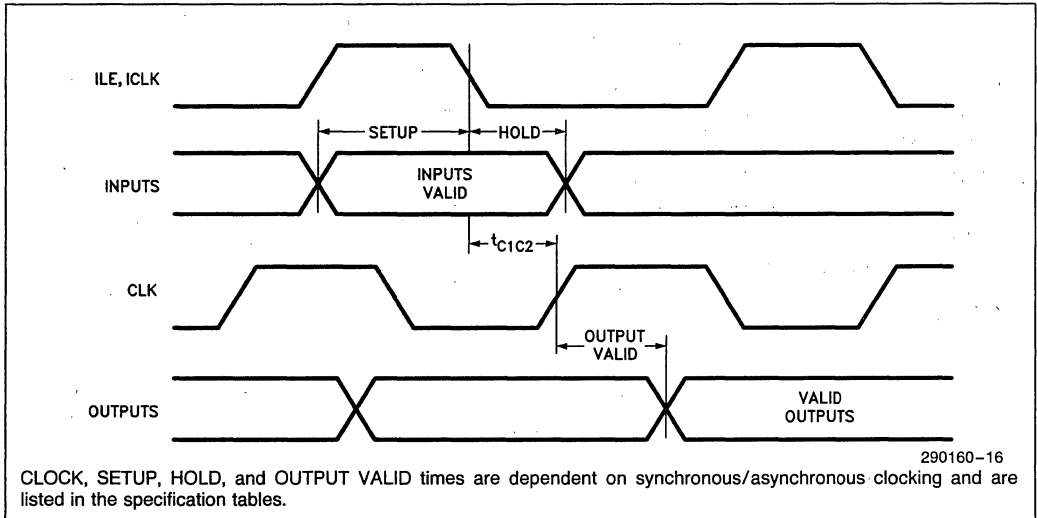
**ASYNCHRONOUS CLOCK MODE (MACROCELLS)**



**ASYNCHRONOUS CLOCK MODE (INPUT STRUCTURE)**

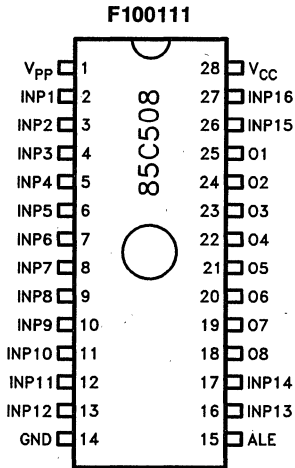


**INPUT-CLOCK-TO-MACROCELL CLOCK TIMING (CLOCKED PIPELINED DATA)**

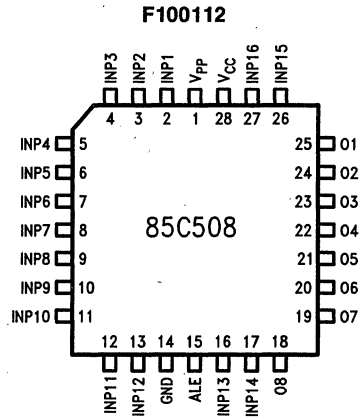


# 85C508 FAST 1-MICRON CHMOS DECODER/LATCH EPLD

- High Performance Programmable Logic Device for High-Speed Microprocessor-to-Memory Decode
  - Upgrade Alternative to Fast Bipolar PLAs and Fast MSI Logic
  - Extremely High Speed— $t_{PD}$  7.5 ns (max), 133.3 MHz (max),  $t_{EO}$  5 ns (max)
- 16 Dedicated Inputs for Address/Data Bus Decoding; 8 Latched Outputs; 1 Global Latch Enable
  - 100% Generically Testable Logic Array
  - Available in 28-pin 300-mil CERDIP and PDIP Packages and in PLCC Package  
(See Packaging Spec., Order Number #231369)



290175-1



290175-2

**Figure 1. 85C508 Pinout Diagrams**

## INTRODUCTION

The Intel 85C508 1-micron CHMOS EPLD (Erasable Programmable Logic Device) is designed to support the speeds required in fast microprocessor to memory paths. The sixteen inputs, p-term array, and eight output latches in the 85C508 provide address and data bus decoding and latching. The 85C508 takes full advantage of the lightning speed of Intel's 1-micron CHMOS technology. The 85C508 can be used as an upgrade to fast bipolar PLDs, and to fast AL, ALS, HC, or HCT SSI and MSI logic devices.

The 85C508 uses advanced EPROM cells as architecture and logic array storage elements instead of poly-silicon fuses. Coupled with Intel's proprietary CHMOS technology, the result is a device that offers a fast 7.5 ns  $t_{PD}$  in flow-through mode and a  $t_{EO}$  of 5 ns in latch mode. The inherent speed of the device makes the 85C508 ideally suited for bus decoding applications with Intel's 80386 microprocessor and 80960 embedded controller families.

## ARCHITECTURE DESCRIPTION

The architecture of the 85C508 is designed for high-speed performance, with dedicated inputs feeding a logic array. Outputs from the logic array feed the fast output latches. All output latches are controlled by the global ALE (Address Latch Enable) signal. Figure 2 shows the global architecture of the 85C508.

The input to each latch is a single NAND p-term that can be connected to the true or complement state of the dedicated inputs. All input signals are available to all eight macrocells.

Each intersecting point in the logic array is connected or not connected based on the value programmed in the EPROM array. Initially (EPROM erased state), no connections exist between any p-term and any input. Connections can be made by programming the appropriate EPROM cells. True and complement connections cannot exist at the same time. Since p-terms are implemented as NANDs, a true condition on a p-term drives the output low.

## POWER-ON CHARACTERISTICS

On  $V_{CC}$  power-up, the 85C508 latches respond to the values on the input signals. No logic high/low state is guaranteed at power up. 85C508 inputs and outputs begin responding approximately 5  $\mu$ s after  $V_{CC}$  power-up or after a power-loss/power-up sequence.

## ERASURE CHARACTERISTICS

Erasure time for the 85C508 is 1 hour at 12,000  $\mu$ Wsec/cm<sup>2</sup> with a 2537Å UV lamp.

Erasure characteristics of the device are such that erasure begins to occur upon exposure to light with wavelengths shorter than approximately 400Å. It should be noted that sunlight and certain types of fluorescent lamps have wavelengths in the 3000Å–4000Å range. Data shows that constant exposure to room level fluorescent lighting could erase the typical 85C508 in approximately six years, while it would take approximately two weeks to erase the device when exposed to direct sunlight. If the device is to be exposed to these lighting conditions for extended periods of time, conductive opaque labels should be placed over the device window to prevent unintentional erasure.

The recommended erasure procedure for the 85C508 is exposure to shortwave ultraviolet light with a wavelength of 2537Å. The integrated dose (i.e., UV intensity x exposure time) for erasure should be a minimum of fifteen (15) Wsec/cm<sup>2</sup>. The erasure time with this dosage is approximately 1 hour using an ultraviolet lamp with a 12,000  $\mu$ W/cm<sup>2</sup> power rating. The device should be placed within 1 inch of the lamp tubes during exposure. The maximum integrated dose the 85C508 can be exposed to without damage is 7258 Wsec/cm<sup>2</sup> (1 week at 12,000  $\mu$ W/cm<sup>2</sup>). Exposure to high intensity UV light for longer periods may cause permanent damage to the device.

## LATCH-UP IMMUNITY

All of the input, output, and clock pins of the device have been designed to resist latch-up which is inherent in inferior CMOS structures. The 85C508 is designed with Intel's proprietary 1-micron CHMOS EPROM process. Thus, each of the pins will not experience latch-up with currents up to 100 mA and voltages ranging from -0.5V to  $V_{CC} + 0.5V$ . The programming pin is designed to resist latch-up to the 13.5V maximum device limit.

## DESIGN RECOMMENDATIONS

For proper operation, it is recommended that all input and output pins be constrained to the voltage range  $GND < (V_{IN} \text{ or } V_{OUT}) < V_{CC}$ . All unused inputs should be tied to an appropriate logic level to minimize power consumption (do not leave them floating). A power supply decoupling capacitor of at least 0.2  $\mu$ F must be connected directly between each  $V_{CC}$  and GND pin.

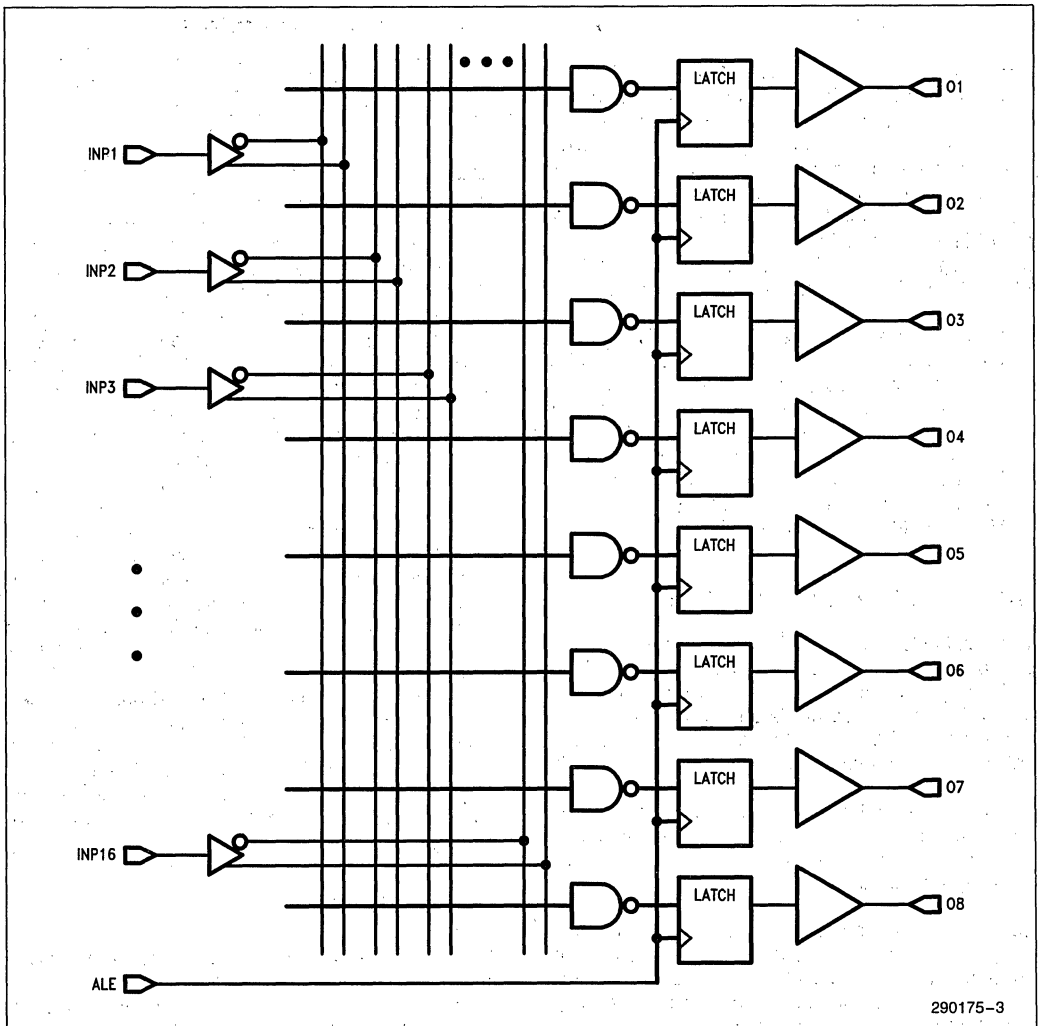


Figure 2. 85C508 Global Architecture

290175-3

As with all CMOS devices, ESD handling procedures should be used with the 85C508 to prevent damage to the device during programming, assembly, and test.

**FUNCTIONAL TESTING**

Since the logical operation of the 85C508 is controlled by EPROM elements, the device is completely testable during the manufacturing process. Each programmable EPROM bit controlling the internal logic is tested using application independent test patterns. EPROM cells in the device are 100% tested for programming and erasure. After testing, the devices are erased before shipments to the customers. No post-programming tests of the EPROM array are required.

The testability and reliability of EPROM-based programmable logic devices is an important feature over similar devices based on fuse technology. Fuse-based programmable logic devices require a user to perform post-programming tests to insure device functionality. During the manufacturing process, tests on fuse-based parts can only be performed in very restricted ways in order to avoid pre-programming the array.

**DESIGN SOFTWARE**

Full software support is provided by version 2.0 of iPLS II (Intel Programmable Logic Software II). That version includes the LOC (Logic Optimizing Compiler), LPS (Logic Programming Software), and Macro Librarian.

For detailed information on iPLS II, refer to the iPLDS II Data Sheet, order number: 290134.

**ORDERING INFORMATION**

t <sub>PD</sub> (ns)	t <sub>EO</sub> (ns)	f <sub>max</sub> (MHz)	Order Code	Package	Operating Range
*7.5	5	133.3	N85C508-7	PLCC	Commercial
			D85C508-7	CERDIP	
			P85C508-7	PDIP	
10	6	100	N85C508-10	PLCC	Commercial
			D85C508-10	CERDIP	
			P85C508-10	PDIP	
15	10	66.5	N85C508-15	PLCC	Commercial
			D85C508-15	CERDIP	
			P85C508-15	PDIP	

**\*NOTE:**  
Under development.



**ABSOLUTE MAXIMUM RATINGS\***

- Supply Voltage ( $V_{CC}$ )<sup>(1)</sup> ..... -2.0V to +7.0V
- Programming Supply Voltage ( $V_{PP}$ )<sup>(1)</sup> ..... -2.0V to +13.5V
- D.C. Input Voltage ( $V_I$ )<sup>(1, 2)</sup> ... -0.5V to  $V_{CC} + 0.5V$
- Storage Temperature ( $T_{stg}$ ) ..... -65°C to +150°C
- Ambient Temperature ( $T_{amb}$ )<sup>(3)</sup> ... -10°C to +85°C

**NOTES:**

1. Voltages with respect to GND.
2. Minimum D.C. input is -0.5V. During transitions, the inputs may undershoot to -2.0V or overshoot to 7.0V for periods of less than 20 ns under no load conditions.
3. Under bias. Extended Temperature versions are also available.

*\*Notice: Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.*

*NOTICE: Specifications contained within the following tables are subject to change.*

**RECOMMENDED OPERATING CONDITIONS**

Symbol	Parameter	Min	Max	Units
$V_{CC}$	Supply Voltage	4.75	5.25	V
$V_{IN}$	Input Voltage	0	$V_{CC}$	V
$V_O$	Output Voltage	0	$V_{CC}$	V
$T_A$	Operating Temperature	0	+70	°C
$t_R$	Input Rise Time		500	ns
$t_F$	Input Fall Time		500	ns

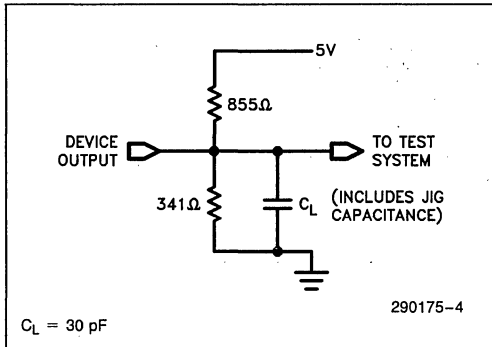
**D.C. CHARACTERISTICS** ( $T_A = 0^\circ\text{C}$  to  $+70^\circ\text{C}$ ,  $V_{CC} = 5.0V \pm 5\%$ )

Symbol	Parameter	Conditions	Min	Typ	Max	Units
$V_{IH}^{(4)}$	High Level Input Voltage		2.0		$V_{CC} + 0.3$	V
$V_{IL}^{(4)}$	Low Level Input Voltage		-0.3		0.8	V
$V_{OH}$	High Level Output Voltage	$I_O = -4.0$ mA D.C., $V_{CC} = \text{min}$	2.4			V
$V_{OL}$	Low Level Output Voltage	$I_O = 4.0$ mA D.C., $V_{CC} = \text{min}$			0.45	V
$I_I$	Input Leakage Current	$V_{CC} = \text{max.}$ , $GND < V_{IN} < V_{CC}$			$\pm 10$	$\mu\text{A}$
$I_{OZ}$	Output Leakage Current	$V_{CC} = \text{max.}$ , $GND < V_{OUT} < V_{CC}$			$\pm 10$	$\mu\text{A}$
$I_{SC}^{(5)}$	Output Short Circuit Current	$V_{CC} = \text{max.}$ , $V_{OUT} = 0.5V$	-30		-90	mA
$I_{CC}$	Power Supply Current	$V_{CC} = \text{max.}$ , $V_{IN} = V_{CC}$ or GND, No Load, $f_{IN} = 50$ MHz, Device Prog. as 16-Bit Address Decoder		30		mA

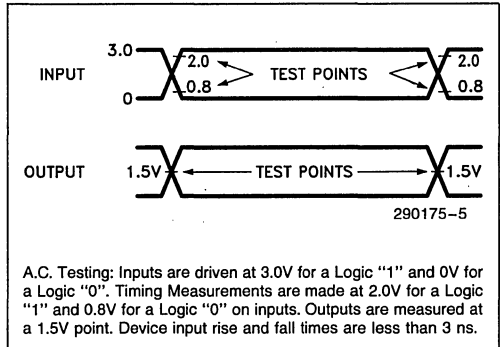
**NOTES:**

4. Absolute values with respect to device GND; all over and undershoots due to system or tester noise are included. Do not attempt to test these values without suitable equipment.
5. Not more than 1 output should be tested at a time. Duration of that test should not exceed 1 second.

**A.C. TESTING LOAD CIRCUIT**



**A.C. TESTING INPUT, OUTPUT WAVEFORM**



A.C. Testing: Inputs are driven at 3.0V for a Logic "1" and 0V for a Logic "0". Timing Measurements are made at 2.0V for a Logic "1" and 0.8V for a Logic "0" on inputs. Outputs are measured at a 1.5V point. Device input rise and fall times are less than 3 ns.

**CAPACITANCE**  $T_A = 0^\circ\text{C to } +70^\circ\text{C}; V_{CC} = 5.0\text{V} \pm 5\%$

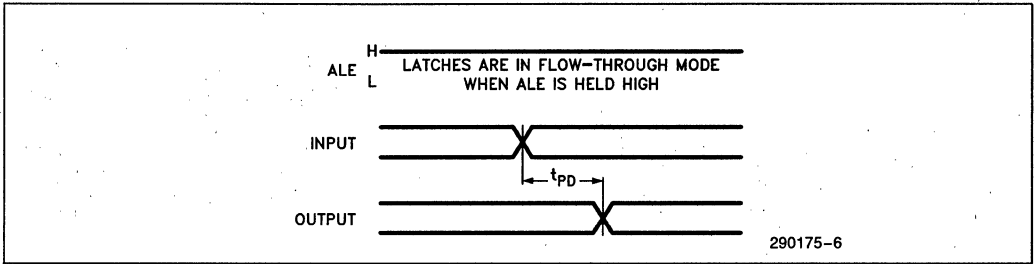
Symbol	Parameter	Conditions	Min	Typ	Max	Units
$C_{IN}$	Input Capacitance	$V_{IN} = 0\text{V}, f = 1.0\text{ MHz}$		6	10	pF
$C_{OUT}$	Output Capacitance	$V_{OUT} = 0\text{V}, f = 1.0\text{ MHz}$		6	10	pF
$C_{CLK}$	ALE Capacitance	$V_{OUT} = 0\text{V}, f = 1.0\text{ MHz}$		6	10	pF
$C_{VPP}$	$V_{PP}$ Pin Capacitance	$V_{PP}$ on Pin 1		20	40	pF

**A.C. CHARACTERISTICS**  $T_A = 0^\circ\text{C to } +70^\circ\text{C}, V_{CC} = 5.0\text{V} \pm 5\%$

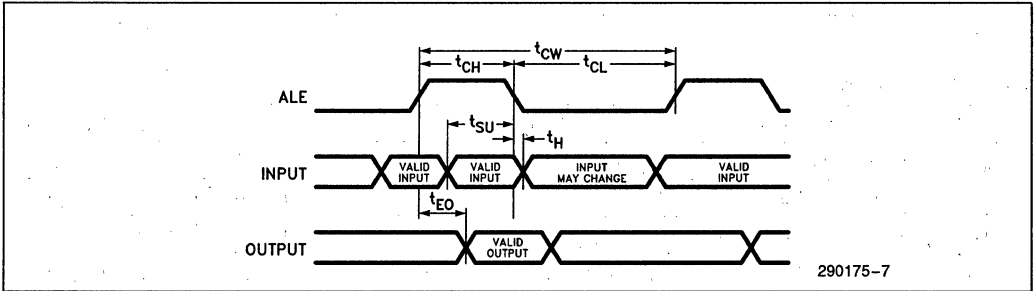
Symbol	Parameter	*85C508-7			85C508-10			85C508-15			Units
		Min	Typ	Max	Min	Typ	Max	Min	Typ	Max	
$t_{PD}$	Propagation Delay (Flow-Through Mode)			7.5		8	10		13	15	ns
$f_{max}$	Maximum Frequency ( $1/t_{CW}$ )			133.3		112	100		90	82.5	MHz
$t_{EO}$	Output Valid from ALE $\uparrow$			5		5	6		8	10	ns
$t_{SU}$	Input Setup Time to ALE $\downarrow$					7	5		10	8	ns
$t_H$	Input Hold from ALE $\downarrow$					-3			-3		ns
$t_{CH}$	ALE High Time					5			7.5		ns
$t_{CL}$	ALE Low Time					5			7.5		ns
$t_{CW}$	ALE Clock Width					10			15		ns

\*NOTE:  
Under development.

**FLOW-THROUGH MODE**



**LATCH MODE**





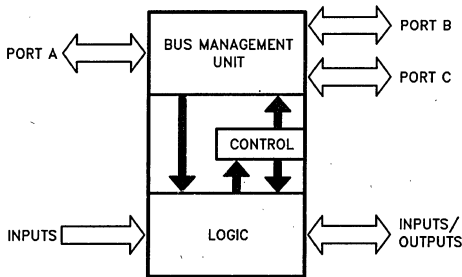
# 5CBIC PROGRAMMABLE BUS INTERFACE CONTROLLER

- Higher Integration Alternative to Transceivers, Latches, Multiplexers and PAL\* Functions
- Applications Include Dual Port Control, Multiplexed Bus Interface, DRAM Control and Similar Functions
- Port-Oriented Bus Management Unit Supports:
  - 3-Way Asynchronous Data Transfer on Byte-Wide Buses
  - Programmable Option of Latched or Real Time Data
  - True or Complement Data Path
- Macrocell-Based Programmable Logic Unit Provides:
  - Variable Input and Output Architecture
- On-Chip Controls for the Bus Management Unit
- Up to Eight Buried Registers
- Programmable Registers can be Configured as Positive Edge-Triggered D-, J-K, R-S or T- Types
- Asynchronous Preset and Clear on All Registers
- Option of Latched Inputs
- Low Power: 75  $\mu$ A Typical Standby
- CHMOS EPROM Technology Based:
  - Max Bus Port Drive Capability: 16 mA
  - Typical Data Transfer Delay Between Ports = 45 ns
- Available in 44-Lead Package  
(See Packaging Spec., Order # 231369)

The Intel 5CBIC is useful in implementing bus interfacing logic functions that have traditionally been done using SSI/MSI TTL components. Core bus functions are provided that can be customized using EPROM bits for specific applications. Control logic can also be implemented through a sum of products architecture that is included in this 44-lead package. Such levels of integration are realized utilizing the benefits of Intel's advanced CHMOSII-E process.

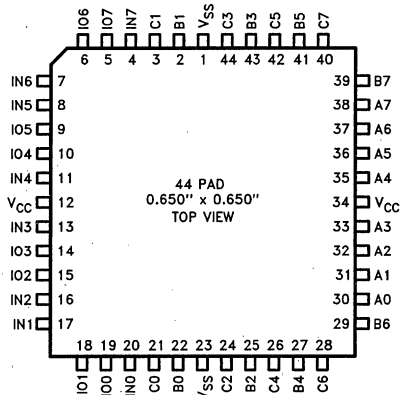
This general purpose architecture is supported by iPLDS II, Intel's Programmable Logic Development System, to develop the design and program the devices. Several methods of entry facilitate the design resulting in shorter completion times.

\*PAL is a trademark of Monolithic Memories, Inc.



290126-1

Figure 1. Block Diagram



290126-2

Figure 2. Lead Configuration

## FUNCTIONAL DESCRIPTION

As the name suggests, this programmable bus interface controller offers a high integration solution to design problems involving data transfer on bus lines and the logic needed to control these transfers. This integration directly translates into savings in board space and lower system cost for equivalent functions implemented using conventional SSI/MSI components.

Present in the port-oriented 5CBIC are two functional blocks that enable complex bus functions to be realized: the Bus Management Unit (BMU) and the Programmable Logic Unit (PLU). These two units communicate with each other through the input and the feedback buses. A control section shown in Figure 3 steers signals from the PLU to the two units through the control bus.

## ARCHITECTURE DESCRIPTION

The innovative architecture of the 5CBIC incorporates a port-oriented approach for bus interface con-

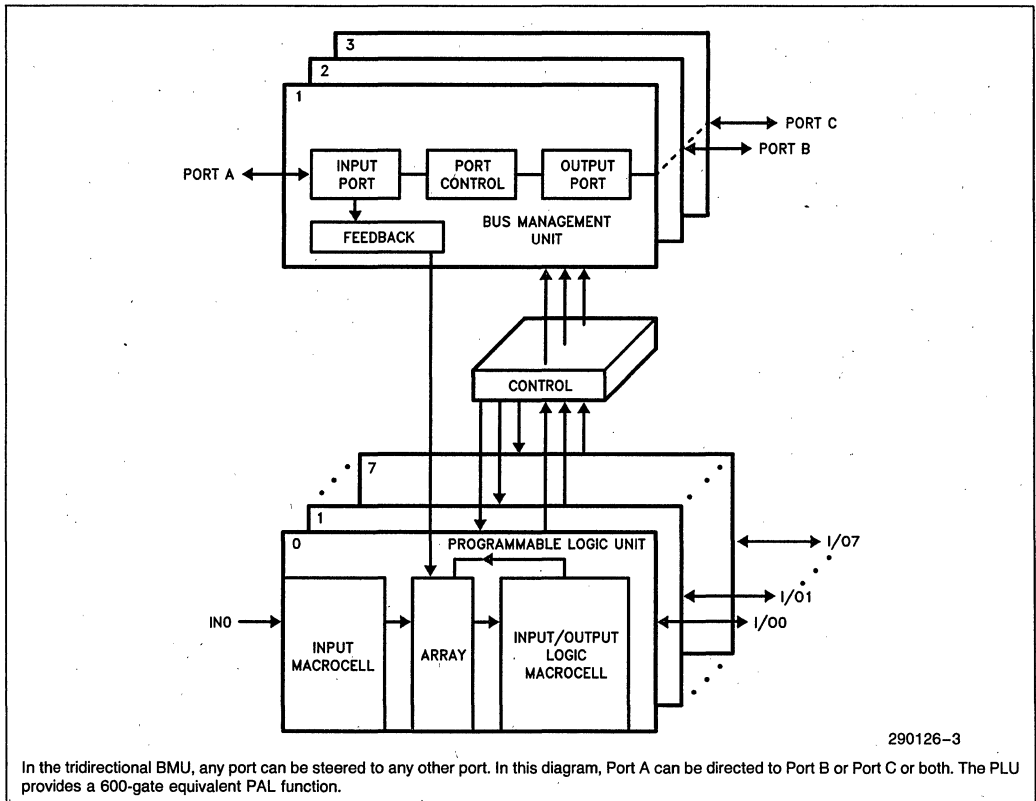
trol is illustrated in Figure 5. The Bus Management Unit (BMU) and the Programmable Logic Unit (PLU) interface to the feedback and the control busses. The macrocells in the PLU feed the input bus.

### Bus Management Unit (BMU)

The Bus Management Unit (BMU) comprises three ports: PA, PB and PC (Figure 4a). Each of these ports is bidirectional and 8 bits wide. Data can be routed from any port to any other port.

Data into any port can be user-selected to be latched by a port Latch Enable signal, (LE). Routing of latched or unlatched data between ports is achieved using a combination of EPROM architecture and dynamic control signals defined by the user. Data out of any port can be programmed to have an inverted sense through EPROM architecture control (INV).

Each bidirectional port can be dynamically configured as an input or an output depending on the control signals OEA, OEB and OEC. Latched data from



In the tridirectional BMU, any port can be steered to any other port. In this diagram, Port A can be directed to Port B or Port C or both. The PLU provides a 600-gate equivalent PAL function.

Figure 3. Functional Blocks in the 5CBIC

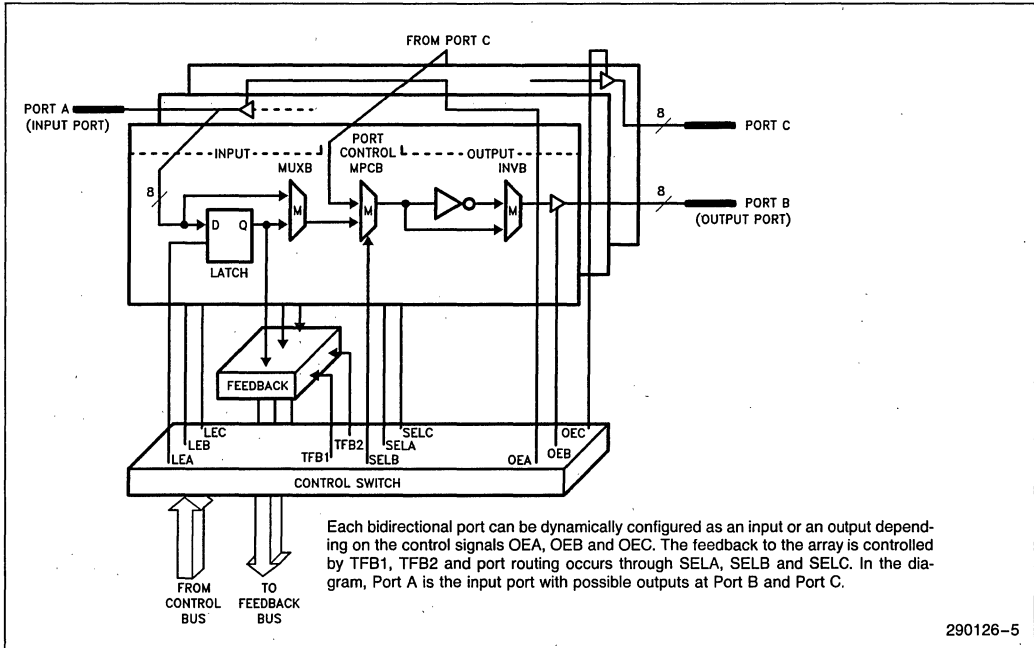


Figure 4a. Bus Management Unit Block Diagram

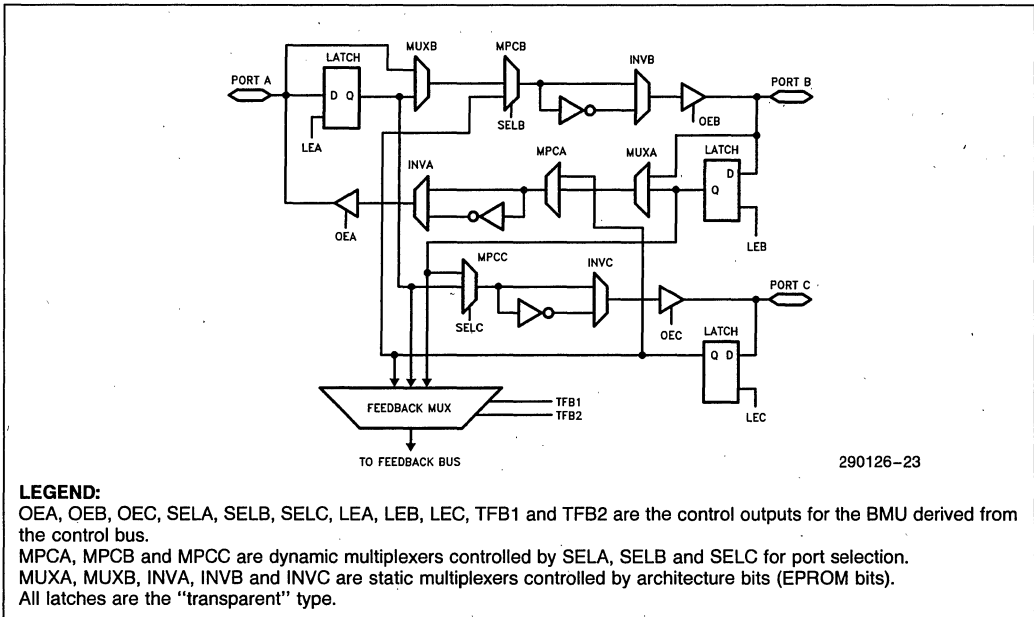


Figure 4b. BMU Logic Diagram

any incoming port can be fed internally to the array through TFB1 and TFB2. The three ports can be time-multiplexed, if needed. Port routing is controlled by signals SELA, SELB and SELC (Figure 4b).

**Programmable Logic Unit (PLU)**

An on-chip 600-gate-equivalent EPLD supplies the control signals to the bus unit and related applica-

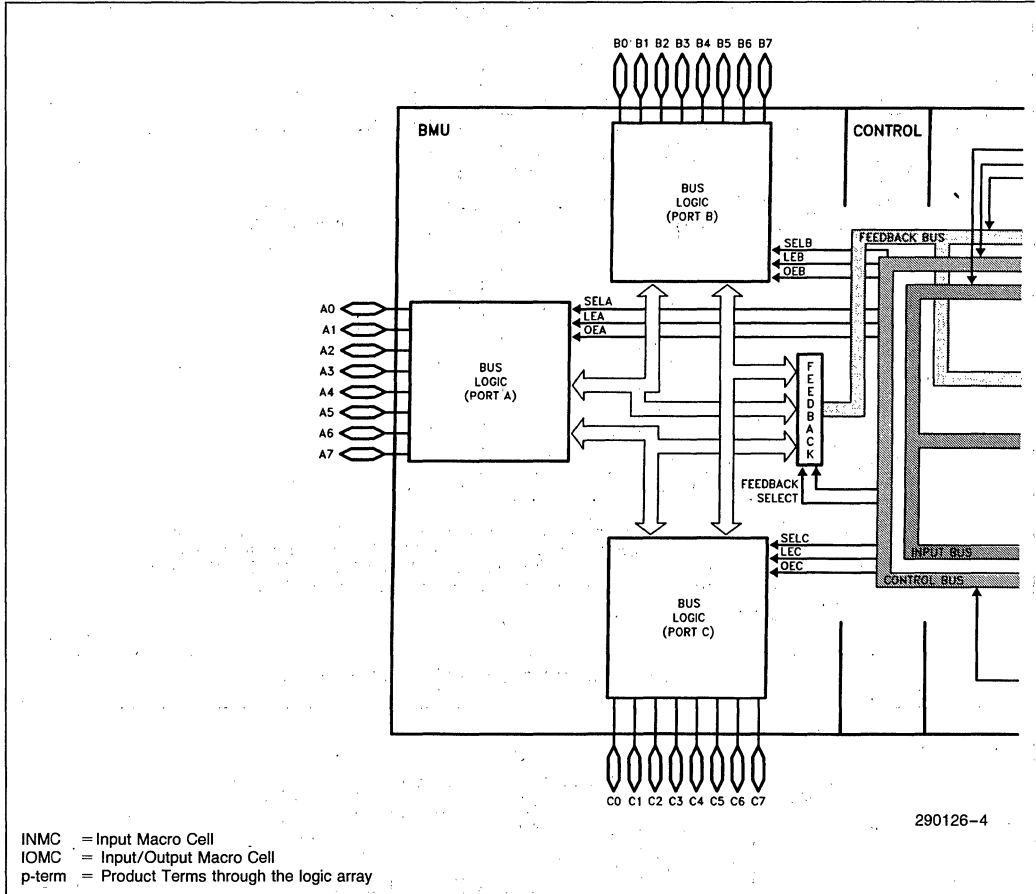
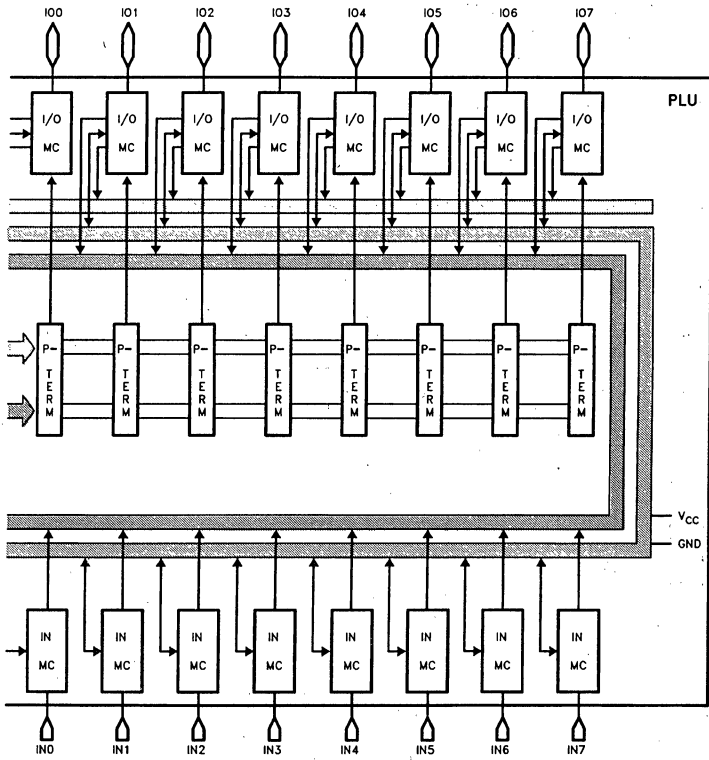


Figure 5. The 5CBIC Architecture



290126-21



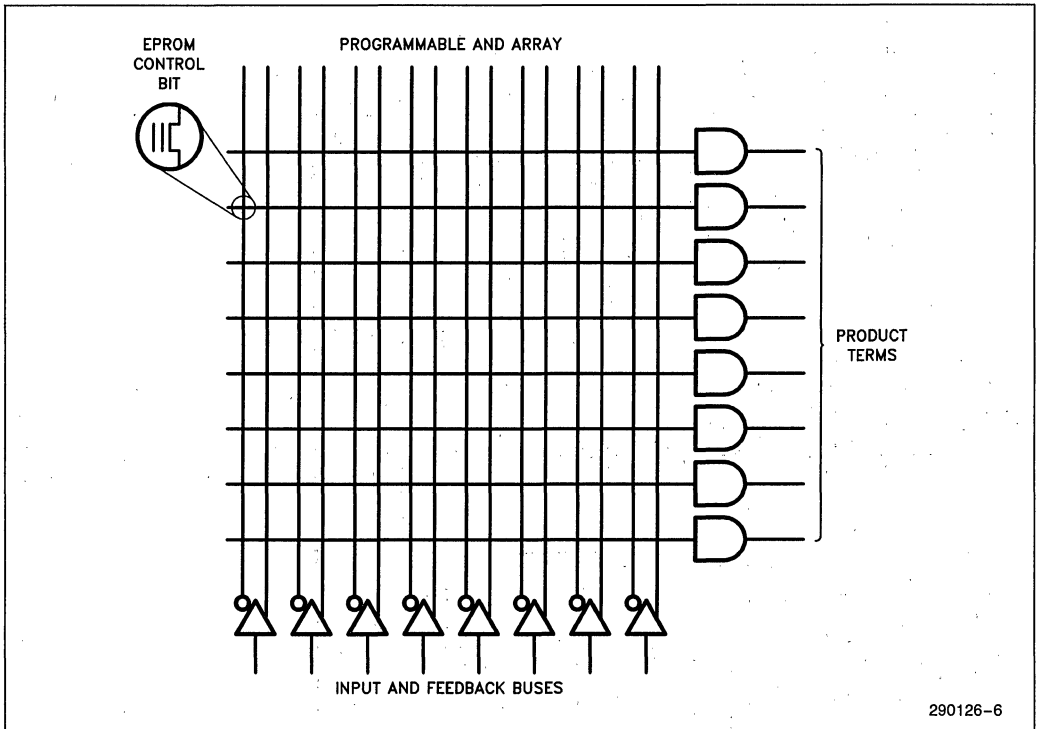


Figure 6. The Array Structure

tion functions in the system. A dedicated input port and a bidirectional I/O port, each 8 bits wide, allows control logic implementation in the 5CBIC. The macrocell based architecture enables the designer to use up to 24 inputs and 8 outputs.

The inputs, array and I/O macrocells generate a sum-of-products (AND-OR) representation of any given logic. Within the AND array, there is an EPROM connection at every intersection of an incoming signal (true and complement) and a product term to a given macrocell (Figure 6). Before programming an erased device an EPROM connection exists at every intersection. It is during the programming process that these connections are opened to generate the required connections.

The bidirectional I/O port, when configured as an input, is identical to the input port in that inputs may be latched by a signal from the control bus as shown in Figure 7. An additional flow-through option for the data inputs is available in the input macrocell.

The variable output architecture in the PLU allows the designer to select the combinatorial or registered output types on a macrocell basis. This may

be implemented by selecting the architecture bit MARB1 and the edge-triggered flip-flop (Figure 7). The Macrocells support D, T, S-R or J-K type registers for optimal design. Truth tables for these are listed in Figure 8 for easy reference. Whereas all eight of the product terms are OR-ed together at the register input for the D- and the T- registers, the J-K and the S-R configurations employ sharing of the product terms among two OR-gates.

The registers receive inputs at its data, clock, set and reset lines. Eight product terms are available for the data input and one each for the set and the clear inputs.

The clock, output enable and the latching signals can be selected by architecture bits MARB2, 6 and 3 respectively to be outputs from the control bus or one product term from the array. Designers thus have more options available for asynchronous clocking and output controls.

The macrocell output can be fed back to the array through the feedback bus or to the control bus. Figure 9 summarizes the bus structure and its relationship to the relevant units in the 5CBIC.

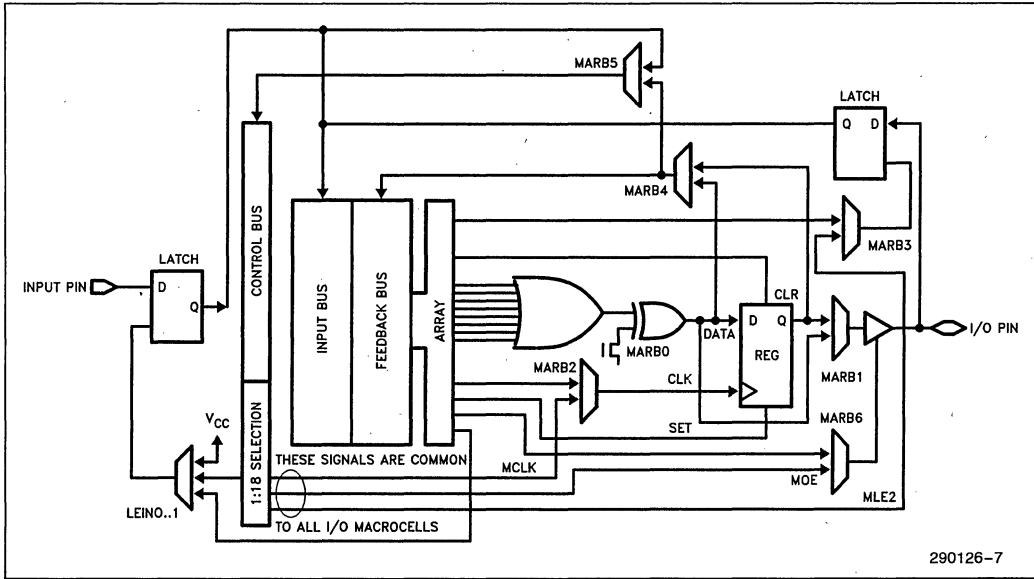


Figure 7. The Programmable Logic Unit

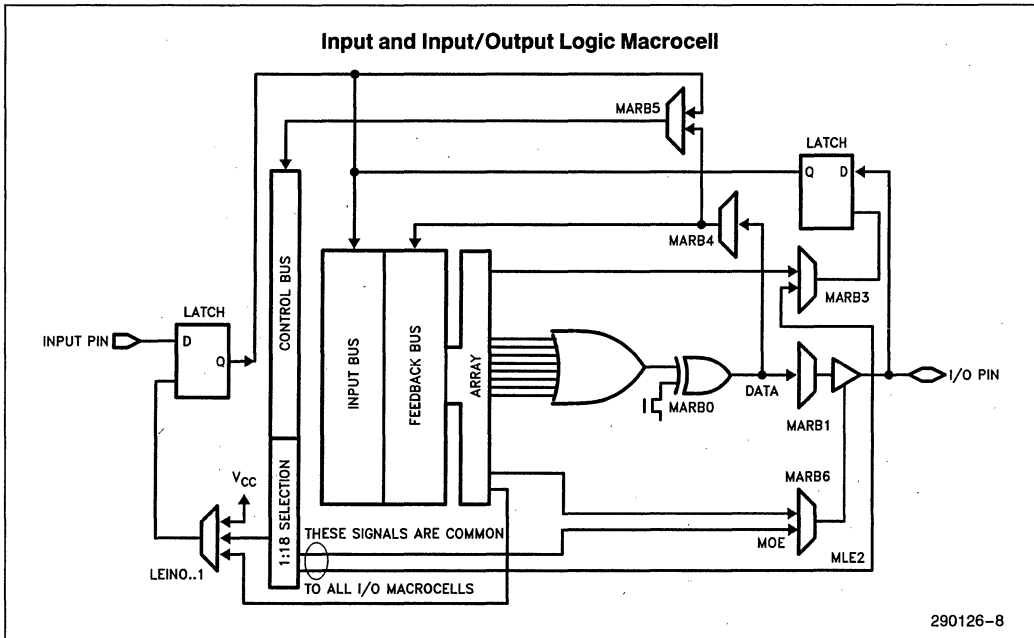


Figure 8a. Combinational

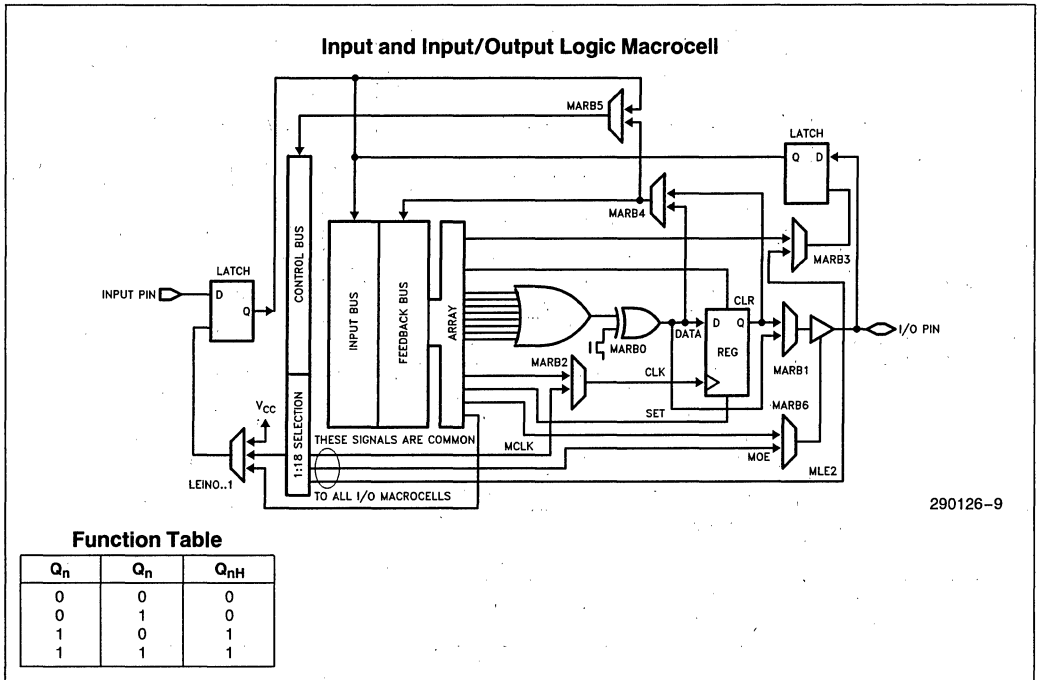


Figure 8b. D-Type Flip-Flop

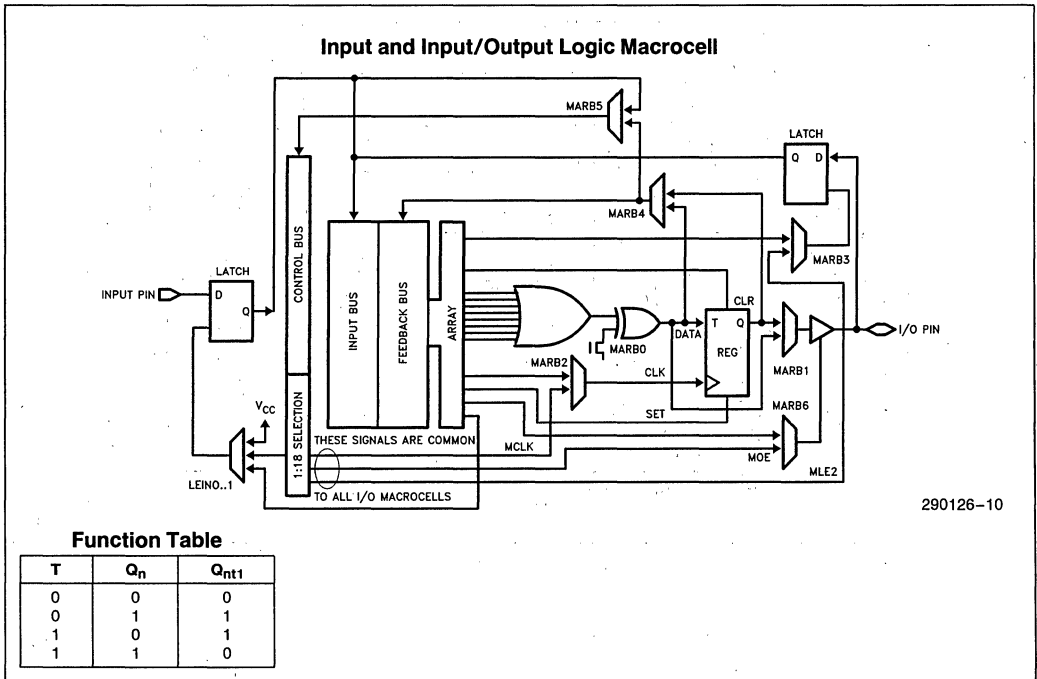


Figure 8c. Toggle Flip-Flop

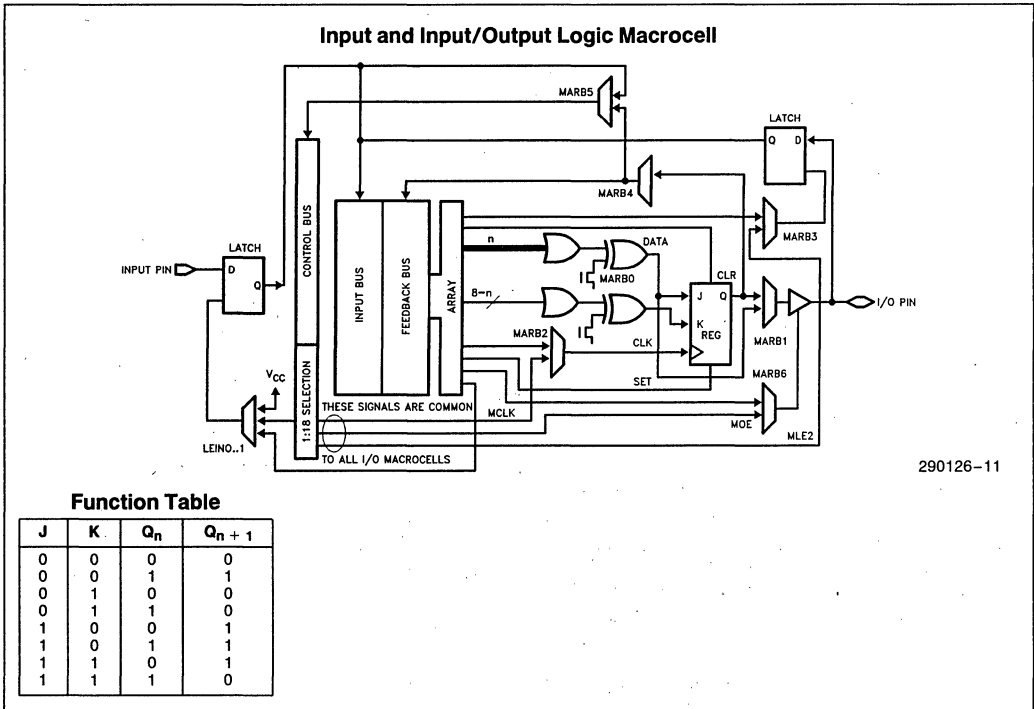


Figure 8d. J-K Flip-Flop

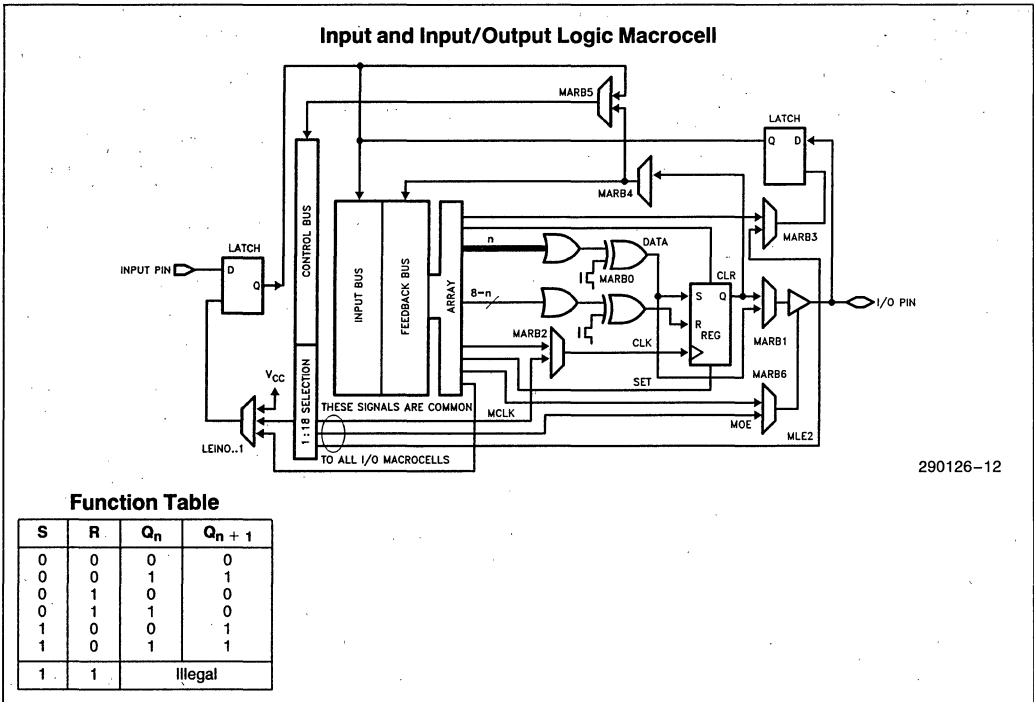
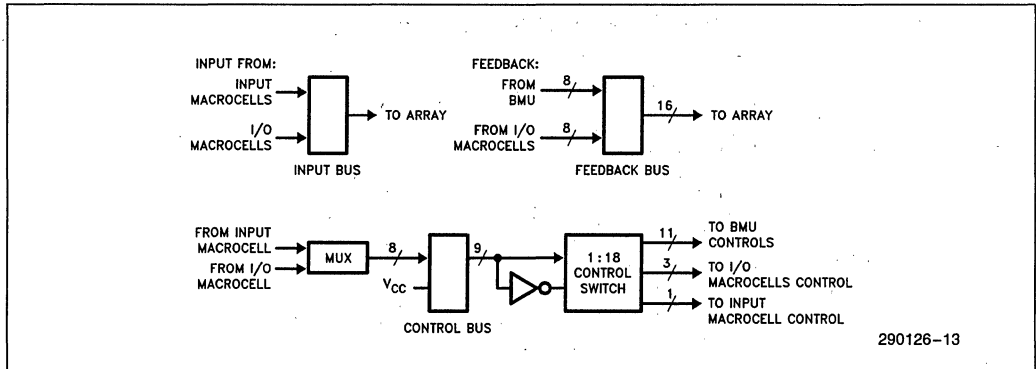


Figure 8e. S-R Flip-Flop



290126-13

Figure 9. The 5CBIC Bus Organization

### Configuring the 5CBIC

The Device Configuration Manager (DCM) in iPLS II provides a high-level graphic design entry alternative that allows bus configurations to be implemented in minutes. A more detailed explanation is given in the iPLS II manual. An ADF (Advanced Design File) is then automatically generated that defines the logic network using primitives.

The primitive necessary for configuring inter-port communication is the "BMU", while the one required for internal feedback from the BMU to the PLU is the feedback primitive "BFMUX". Tables 1 through 4 define these primitives and their fields/bits.

Table 1. BMU Architecture Bits

Architecture Bit	Selects
MUXA, MUXB	Latched or Flow-Through Port Data
INVA, INVB, INVC	True or Inverted Data Output

Table 2. BMU Primitive

OeA	8 bit	PA
SelA	I/O	PB
LeA	Ports	PC
OeB		
SelB		
LeB		
OeC		
SelC		
Lec		
BMU		

Name: BMU (Bus Management (Unit))

ADF Syntax: PortA, PortB, PortC = BMU (Type, OeA, SelA, LeA, OeB, SelB, LeB, OeC, SelC, Lec)

Description: Port A = connection to 8 parallel I/O pins labeled A0-A7

Port B = connection to 8 parallel I/O pins labeled B0-B7

Port C = connection to 8 parallel I/O pins labeled C0-C7

OeA = output enable for Port A

SelA = select B or C internal connection to Port A (0 = C, 1 = B)

LeA = input latch enable for Port A

OeB = output enable for Port B

SelB = select A or C internal connection to Port B (0 = C, 1 = A)

LeB = input latch enable for Port B

OeC = output enable for Port C

SelC = select A or B internal connection to Port C (0 = A, 1 = B)

Lec = input latch enable for Port C

Inversion Control			Input Latch			
Port:	A	B	C	A	B	C
Bit:	5	4	3	2	1	0
0	Invert Output	Invert Output	Invert Output	Latched A	Latched B	Latched C
1	No Invert	No Invert	No Invert	Direct A	Direct B	Latched C*

\*If LeC is continually high, the C latch is transparent.

**Table 3. Bus Feedback Multipler Primitive**

**BFMX**

TFB1	0	0	1	Fbk	[0:7]
TFB2	0	1	0		
	C	B	A		

Name: BFMX (Bus Feedback Multiplexer)

ADF Syntax: Fbk[0:7] = BFMX (TFB1, TFB2)

Description: Outputs.

Fbk = 8 parallel lines of feedback to logic array.

Inputs:

TFB1, TFB2 = By appling 0 or 1 as shown on the chart above, select feedback from Port A, B, or C. TFB1 and TFB2 can be set to VCC or GND, or they can be connected to any internal feedback or input node. The ports are defined in the BMU primitive section.

**Table 4. PLU Architecture Bits**

Architecture Bit	Selects
MARB0	Output Polarity
MARB1	Combinatorial or Registered Outputs
MARB2	Clock Source
MARB3	Latching Signal Source
MARB4	Combinatorial or Registered Feedback to the Logic Array
MARB5	Input Source to the Control Bus
MARB6	tri-state Control Signal

**ABSOLUTE MAXIMUM RATINGS\***

Symbol	Parameter	Min	Max	Units
V <sub>CC</sub>	Supply Voltage(1)	-2.0	7.0	V
V <sub>PP</sub>	Programming Supply Voltage(1)	-2.0	13.5	V
V <sub>I</sub>	DC Input Voltage(1)(2)	-0.5	V <sub>CC</sub> + 0.5	V
t <sub>stg</sub>	Storage Temperature	-65	+150	°C
t <sub>amb</sub>	Ambient Temperature(3)	-10	+85	°C

\*Notice: Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

**NOTES:**

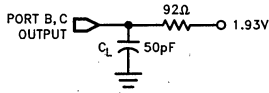
1. Voltages with respect to ground.
2. Minimum DC input is -0.5V. During transitions, the inputs may undershoot to -2.0V for periods less than 20 ns under no load conditions.
3. Under bias. Extended temperature versions are also available.

**D.C. CHARACTERISTICS** T<sub>A</sub> = 0°C to +70°C, V<sub>CC</sub> = 5.0V ±5%

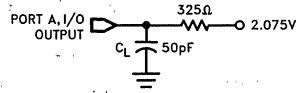
Parameter	Description	Min	Max	Unit	Test Conditions
V <sub>OH</sub>	Output High Voltage	2.4		V	TTL: I <sub>OH</sub> Port A -1 mA, Port B, C -5 mA, I, I/O -1 mA, V <sub>CC</sub> = Min
V <sub>OL</sub>	Output Low Voltage		0.45	V	I <sub>OL</sub> Port A 5 mA, Port B, C 16 mA, I, I/O 5 mA, V <sub>CC</sub> = Min
V <sub>IH</sub>	Input High Level	2.0	V <sub>CC</sub> + 0.3	V	
V <sub>IL</sub>	Input Low Level	-0.3	0.8	V	
I <sub>I</sub>	Input Leakage Current		10	μA	V <sub>SS</sub> ≤ V <sub>IN</sub> < V <sub>CC</sub> , V <sub>CC</sub> = Max
I <sub>OZ</sub>	Output Leakage Current		10	μA	V <sub>SS</sub> ≤ V <sub>OUT</sub> ≤ V <sub>CC</sub> , V <sub>CC</sub> = Max
I <sub>OS</sub> (4)	Output Short Circuit Current	BMU PLU	80 16	mA mA	V <sub>CC</sub> = Max, V <sub>OUT</sub> = 0.5
I <sub>SB</sub> (5)	Operating Current (standby, low power mode)		75	μA	V <sub>IN</sub> = V <sub>CC</sub> or Gnd, I <sub>O</sub> = 0
I <sub>CC2</sub>	Operating Current (active, low power mode)		20	mA	V <sub>IN</sub> = V <sub>CC</sub> or Gnd, f = 1 MHz, No Load
I <sub>CC3</sub>	Operating Current (active, turbo mode)		108	mA	V <sub>IN</sub> = V <sub>CC</sub> or Gnd, f = 1 MHz, No Load
C <sub>IN</sub>	Input Pin Capacitance		30	pF	
C <sub>OUT</sub>	Output Pin Capacitance		40	pF	

**NOTES:**

4. Output shorted for no more than 1 sec. and only one output shorted at a time.
5. Chip automatically goes into standby mode if logic transitions do not occur at input pins. (Approximately 100 ns after last transition).



290126-14

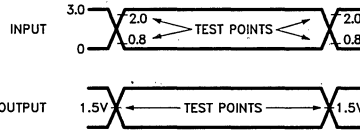


290126-15

**NOTES:**

$C_L$  includes jig capacitance  
 Device input rise and fall times < 6 ns

**Figure 10. A.C. Testing Load Circuit**



290126-16

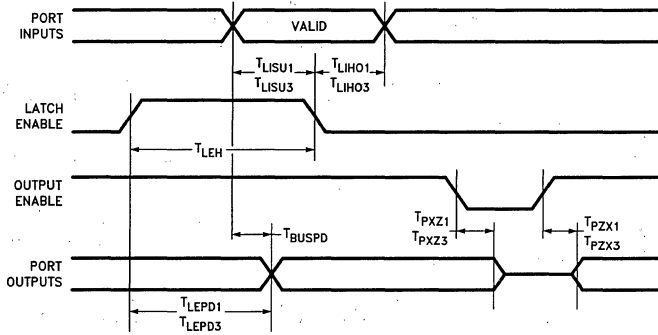
A.C. Testing: Inputs are driven at 3.0V for a Logic "1" and 0V for a Logic "0". Timing measurements are made at 2.0V for a Logic "1" and 0.8V for a Logic "0" on inputs. Outputs are measured at a 1.5V point.

**Figure 11. A.C. Testing Input, Output Waveform**

**Switching Characteristics**

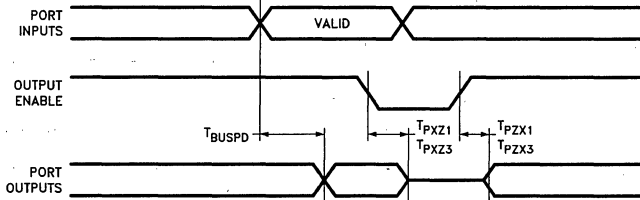
Notation:

Timing Suffix	Referenced to Control From:
1	direct input pin
2	product term
3	control bus



290126-17

**A) Latched Port Inputs**



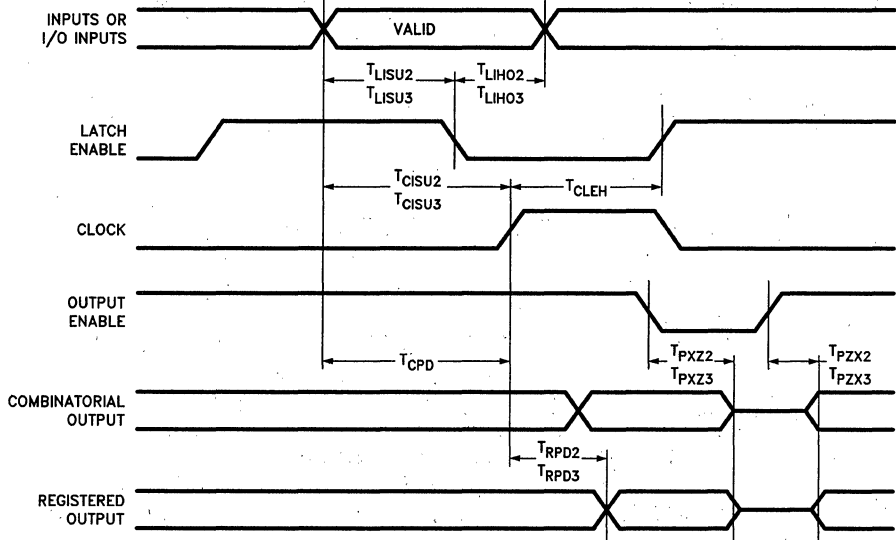
290126-22

**B) Direct Port Inputs**

**Figure 12. Bus Management Unit**

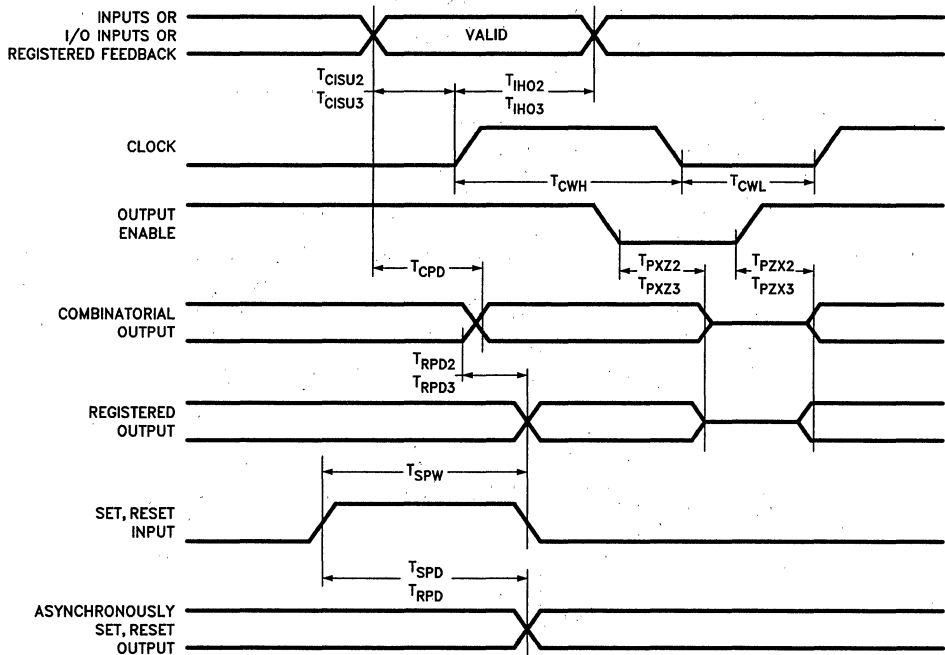


Switching Characteristics (Continued)



290126-18

A) Latched Inputs



290126-19

B) Direct Inputs

Figure 13. Programmable Logic Unit

**AC CHARACTERISTICS**
**BUS MANAGEMENT UNIT**

Symbol	Parameter	-45			Units Max
		Min	Typ	Max	
T <sub>LISU1</sub>	Port Input Setup Time to Latch Enable (Fast Option)	0			ns
T <sub>LISU3</sub>	Port Input Setup Time to Latch Enable (Control Bus)	0			ns
T <sub>LIHO1</sub>	Port Input Hold Time to Latch Enable (Fast Option)	55			ns
T <sub>LIHO3</sub>	Port Input Hold Time to Latch Enable (Control Bus)	95			ns
T <sub>LEH</sub>	Latch Enable High Time	45			ns
T <sub>BUSPD</sub>	Port to Port Propagation Delay			45	ns
T <sub>PXZ1</sub>	Valid Output to High Impedance (OE From Fast Option)			45	ns
T <sub>PXZ3</sub>	Valid Output to High Impedance (OE From Control Bus)			95	ns
T <sub>PZX1</sub>	High Impedance to Valid Output (OE From Fast Option)			45	ns
T <sub>PZX3</sub>	High Impedance to Valid Output (OE From Control Bus)			95	ns
T <sub>LEPD1</sub>	Latch Enable (From Fast Option) To Port Output Delay			65	ns
T <sub>LEPD3</sub>	Latch Enable (From Control Bus) To Port Output Delay			95	ns

**PROGRAMMABLE LOGIC UNIT**

Symbol	Parameter	-45			Units
		Min	Typ	Max	
T <sub>LISU2</sub>	Input Setup Time to Latch Enable (P-Term)	0			ns
T <sub>LISU3</sub>	Input Setup Time to Latch Enable (Control Bus)	0			ns
T <sub>LIHO2</sub>	Input Hold Time to Latch Enable (P-Term)	80			ns
T <sub>LIHO3</sub>	Input Hold Time to Latch Enable (Control Bus)	90			ns
T <sub>CISU2</sub>	Input Setup Time to Clock (P-Term)	20			ns
T <sub>CISU3</sub>	Input Setup Time to Clock (Control Bus)	60			ns
T <sub>CLEH</sub>	Clock to Latch Enable Hold Time	5			ns
T <sub>CPD</sub>	Combinatorial Output Delay			135	ns

**PROGRAMMABLE LOGIC UNIT (Continued)**

Symbol	Parameter	-45			Units
		Min	Typ	Max	
$T_{RPD2}^{(6)}$	Registered Output from Clock (P-Term)			115	ns
$T_{RPD3}^{(7)}$	Registered Output from Clock (Control Bus)			70	ns
$T_{IHO2}$	Input Hold Time to Clock (P-Term)	25			ns
$T_{IHO3}$	Input Hold Time to Clock (Control Bus)	90			ns
$T_{CWH}$	Minimum Clock Width High	43			ns
$T_{CWL}$	Minimum Clock Width Low	43			ns
$T_{SPD}$	Set Output Delay			100	ns
$T_{RPD}$	Reset Output Delay			100	ns
$T_{SPW}$	SET/RESET Pulse Width	43			ns
$T_{PXZ2}$	Valid Output to High-Impedance (OE from P-Term)			85	ns
$T_{PXZ3}$	Valid Output to High Impedance (OE from Control Bus)			95	ns
$T_{PZX2}$	High Impedance to Valid Output (OE from P-Term)			95	ns
$T_{PZX3}$	High Impedance to Valid Output (OE from Control Bus)			95	ns
$T_{CP1}$	Minimum Clock Period (Register Output to Register Input Through Feedback Path)			110	ns
F1	Maximum Internal Frequency	9.0			MHz
$T_{CP2}$	Minimum Clock Period Between Logic Transitions (Inputs to Outputs)			135	ns
F2	Maximum External Frequency	7.0			MHz

**NOTES:**

6. Data out on rising edge of clock.  
 7. Data out on falling edge of clock.

**intelligent Programming Algorithm™**

The 5CBIC supports the intelligent Programming Algorithm which rapidly programs Intel H-ELPDs (and EPROMs) using an efficient and reliable method. The intelligent Programming Algorithm is particularly suited to the production programming environment. This method greatly decreases the overall programming time while programming reliability is ensured as the incremental program margin of each bit is continually monitored to determine when the bit has been successfully programmed.

**FUNCTIONAL TESTING**

Since the logical operation of the 5CBIC is controlled by EPROM elements, the device is completely testable. Each programmable EPROM bit controlling the internal logic is tested using application-independent test program patterns. After testing, the devices are erased before shipment to customers. No post-programming tests of the EPROM array are required.

The testability and reliability of EPROM-based programmable logic devices is an important feature

over similar devices based on fuse technology. Fuse-based programmable logic devices require a user to perform post-programming tests to insure proper programming.

## DESIGN SECURITY

A single EPROM bit provides a programmable design security feature that controls the access to the data programmed into the device. If this bit is set, a proprietary design within the device cannot be copied. This EPROM security bit enables a higher degree of design security than fused-based devices since programmed data within EPROM cells is invisible even to microscopic evaluation. The EPROM security bit, along with all the other EPROM control bits, will be reset by erasing the device.

## TURBO-BIT

The device will consume quiescent current (75  $\mu$ A, typically) if no transitions are detected in the array for 100 ns or more. This mode, the power-down mode, can be enabled by selecting the Turbo Bit OFF. If this bit is enabled, however, the device consumes active current. The power-down mode will revert to its active state if a transition is detected in the array, at an extra delay of 25 ns in speed paths.

## LATCH-UP IMMUNITY

All pins of the 5CBIC have been designed to resist latch-up which is inherent in inferior CMOS structures. The 5CBIC designed with Intel's proprietary CHMOS II-E EPROM process. Thus, pins will not experience latch-up with currents up to 100 mA and voltages ranging from  $-1V$  to  $V_{CC} + 1V$ . Furthermore, the programming pin is designed to resist latch-up to the 13.5V maximum device limit.

## INTEL PROGRAMMABLE LOGIC DEVELOPMENT SYSTEM (iPLDS II)

iPLDS II provides all the tools needed to design with Intel H-Series EPLDs or compatible devices. In addition to providing development assistance, iPLDS II insulates the user from having to know all the intricate details of EPLD architecture (the machine will

optimize a design to benefit from architectural features). It contains comprehensive third generation software that supports several different design entry methods, minimizes logic, does automatic pin assignments and produces the best design fit for the selected EPLD. It is user friendly with guided menus, on-line Help messages and soft key inputs.

In addition, the iPLDS II contains programmer hardware in the form of an iUP-PC Universal Programmer-Personal Computer to enable the user to program EPLDs, read and verify programmed devices and also to graphically edit programming files. The software generates industry standard JEDEC object code output files which can be downloaded to other programmers as well.

The iPLDS II has interfaces to popular schematic capture packages to enable designs to be entered using schematics. One low-cost schematic entry method is provided by SCHEMA II-PLD, which supports EPLD primitives and user-defined macro symbols. SCHEMA II-PLD contains the EPLD Design Manager, which provides a single user interface to both SCHEMA II-PLD and iPLS II software. The other design formats supported are Boolean equation entry and State Machine design entry.

The iPLDS II operates on the IBM† PC.XT, PC/AT, or other compatible machine with the following configuration:

1. At least one floppy disk drive and hard disk drive.
2. MS-DOS†† Operation System Version 3.0 or greater.
3. 512K Memory.
4. Intel iUP-PC Universal Programmer-Personal Computer
5. A GUPI LOGIC Adaptor
6. A color monitor is suggested.

Detailed information on the Intel Programmable Logic Development System is contained in a separate Intel data sheet.

†IBM Personal Computer is a registered trademark of International Business Machines Corporation.

††MS-DOS is a registered trademark of Microsoft Corporation.

June 1988

# **Implementing a PS/2 POS Using the 5AC312 EPLD**

**PEDRO VARGAS**  
PROGRAMMABLE LOGIC APPLICATIONS

Order Number: 292047-001

## INTRODUCTION

The introduction of the IBM\* PS/2 (Personal System/2\*) models and the innovative Micro Channel\* has provided numerous opportunities to develop creative interface solutions. Although the interface requirements are new, the designer is faced with making a familiar choice: Use discrete chips (SSI/MSI), incorporate a PLD, or go for the custom IC solution.

In the past, using TTL on the PC/XT/AT bus was often a good choice, but the reduced size of the PS/2 adapters ("plug in boards") increases the cost of board space dramatically. The custom chip solution is probably the best for companies that have a well-defined product, large volumes, and can afford the cost of the chip development. The third choice, using a PLD, is one that has not been popular in PC bus interfacing due to the limited function and performance of most PLDs.

The Intel 5AC312 is a third-generation EPLD that gives designers the resources needed to interface to buses like the Micro Channel. In addition, it provides two benefits not completely provided by either of the other two choices; high integration, and re-programmability. The rest of this application note contains a detailed presentation of a basic POS (Programmable Option Select) implementation for the PS/2 Micro Channel that is done with the 5AC312 EPLD.

## PS/2 MICRO CHANNEL

One of the best features in the PS/2 models is the capability to do system and adapter configuration with software instead of hardware. This feature, called POS (Programmable Option Select), eliminates the need for switches on the motherboard and adapters by replacing them with programmable registers. The idea is, rather than removing boards and manually setting switches, all configuration information is located in files and can be read or written to the motherboard or to the adapters through the Micro Channel. The motherboard and each connector on the Micro Channel has a unique signal called -CD SETUP that initiates a setup mode when it is active. Only one connector at a time can be in the setup mode, which provides an organized way to perform initialization.

## POS REQUIREMENTS

Each adapter must implement POS with eight registers. Depending on the adapter function, not all of them need to be used. The first three (POS registers 0,1,2) are required because they provide the adapter ID and the adapter enable/disable function necessary during setup and error checking. In brief, the way that the system uses POS is as follows:

1. The system selects the adapter to be placed in setup mode by driving its -CD SETUP signal active.
2. The adapter is identified by reading two ID bytes from POS 0 and POS 1 (HEX 100 and 101).
3. The adapter is disabled by writing "0" to POS 2 (HEX 102).
4. If implemented, Option Select Data is written to POS 3, 4, 5.
5. The adapter is enabled by writing "1" to POS 2.
6. The adapter is out of setup mode when the system drives the -CD SETUP signal inactive.

The actual hardware implementation of POS is summarized in IBM technical documents, but the details are left up to each designer.

## ADAPTER REQUIREMENTS

The adapter used for this design is an Intel single-function card that incorporates two modems controlled by a 80C186. Since it performs only one function, there was no need to implement the POS Option Select bytes. (These POS bytes are used with multi-function adapters that do more than one task and reside in the system with similar adapters.) In this case, the only requirements were to provide the ID bytes and the enable/disable features, which are done with POS registers 0,1, and 2. Figure 1 shows the POS register layout and the typical POS hardware implementation as suggested by IBM. Table 1 defines the POS registers.

\*IBM, Personal System/2 and Micro Channel are trademarks of International Business Machines Corporation.

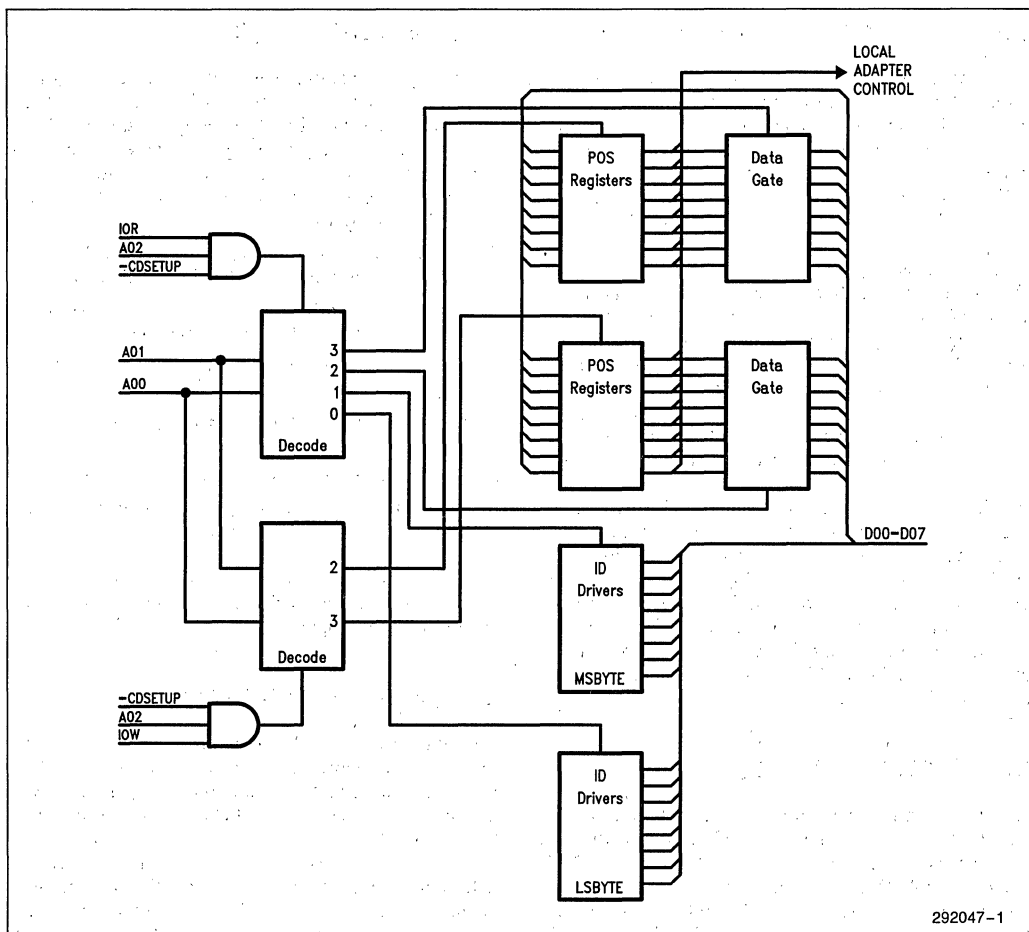


Figure 1. Typical Adaptor Implementation of POS

Table 1. POS I/O Address Decode

Address (hex)	Register	-CD SETUP	Address Bit			Function
			A2	A1	A0	
0100	POS Register 0	0	0	0	0	Adapter Identification Byte (Least Significant Byte)
0101	POS Register 1	0	0	0	1	Adapter Identification Byte (Most Significant Byte)
0102	POS Register 2	0	0	1	0	Option Select Data (Byte 1)*
0103	POS Register 3	0	0	1	1	Option Select Data (Byte 2)
0104	POS Register 4	0	1	0	0	Option Select Data (Byte 3)
0105	POS Register 5	0	1	0	1	Option Select Data (Byte 4)*
0106	POS Register 6	0	1	1	0	Subaddress Extension (Least Significant Byte)
0107	POS Register 7	0	1	1	1	Subaddress Extension (Most Significant Byte)

\*These bytes contain one or more bits with specific value assignments

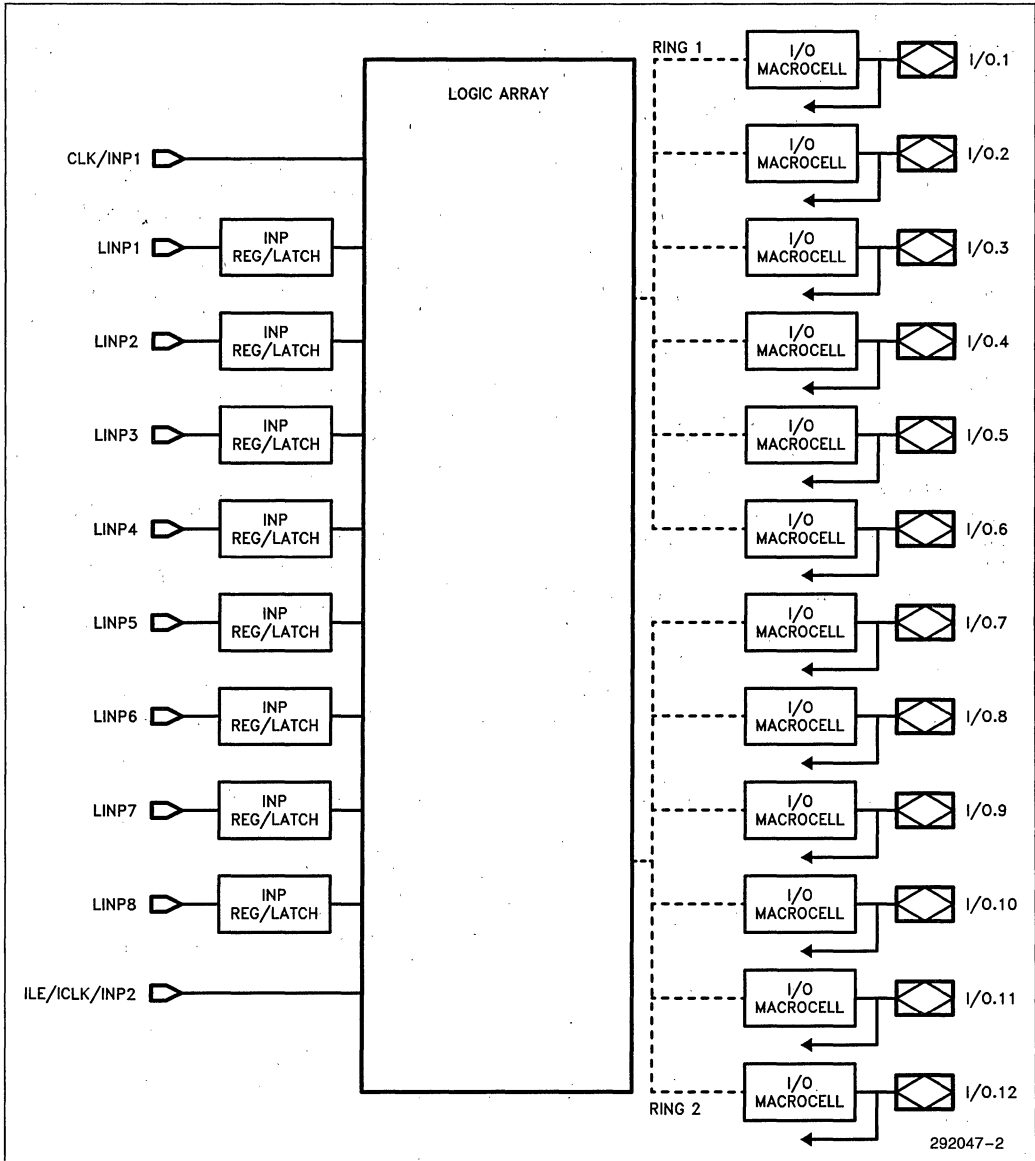


Figure 2. 5AC312 Architecture

**5AC312 EPLD**

With 12 macrocells and a host of other features, the 5AC312 is Intel's newest EPLD. The device is based on the same CHMOS process used in other Intel devices. This EPLD provides an abundant feature set, but its strength lies in being able to efficiently implement one very important function missing from most PLDs: register-logic-register functions.

**DEVICE DESCRIPTION**

The 5AC312 (Figure 2) contains 12 macrocells with programmable outputs and inputs. A macrocell is the basic block associated with each output register within the EPLD. The 5AC312 has the following features:

- 12 I/O macrocells with dual feedback for implementing buried registers.



- 8 programmable inputs that can be configured as latches, registers, or flow through inputs. These can be clocked synchronously or asynchronously.
- Product term allocation on each macrocell.
- 2 product terms on all macrocell control signals.
- 2 multi-function pins; a CLK/INPUT and a ILE/ICLK/INPUT.
- 40 MHz operation.

The 5AC312 provides three major benefits that are especially important to designers working on bus interfaces:

1. The availability of input latches (Figure 3) makes it easy to synchronize bus control signals synchronously or asynchronously. The latches can be clocked as a group of 8 or individually, as is quite common on most buses. Input latches also make state machine designs more reliable. Since buses are prone to glitches and other transients, the ability to hold the inputs stable while transitioning through states makes the difference between a clean and a jittery state machine.
2. Product Term Allocation (Patent Pending) brings a new concept to the Intel EPLD family and makes the 5AC312 unique among PLDs. This feature means that the designer can implement large designs that contain as few as zero or as many as 16 product

terms per macrocell (Figure 4). Product Term Allocation takes place in two rings of six macrocells. Within each ring (Figure 2), individual macrocells can allocate p-terms to/from adjacent macrocells. This is a real benefit in bus decoding where intermediate signals can have few or many p-terms all within the same logic function. Most designers that use PLDs have at least one horror story of a design that required 10 or more p-terms and a device that could only provide 8.

3. A flexible output structure is a must for efficient bus interfacing, which quite commonly requires lots of I/O and complex control signals. The 5AC312 meets these demands head-on with dual-feedback paths and two p-terms per control signal on all I/O macrocells. This means that certain functions, like state machines, can be buried and a pin won't be wasted because it can be used as an additional input. Also, output enables and register operations are frequently generated by a combination of memory, I/O, read, and write strobes. Many times these control signals require two p-terms or the equivalent of an external read/write multiplexer. Prior to the 5AC312, the only way to implement this in PLDs was to waste a macrocell to inefficiently provide this function. Figure 5 shows the macrocell structure and details this third benefit.

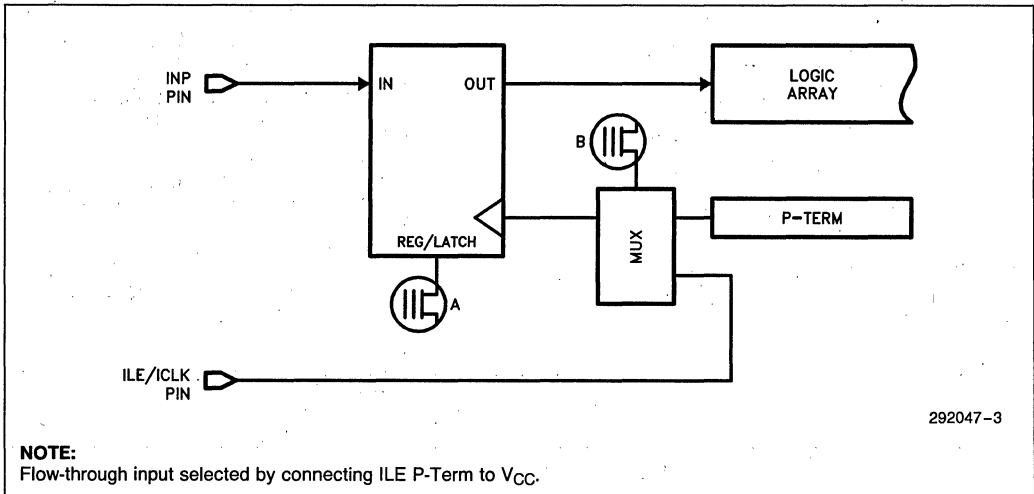
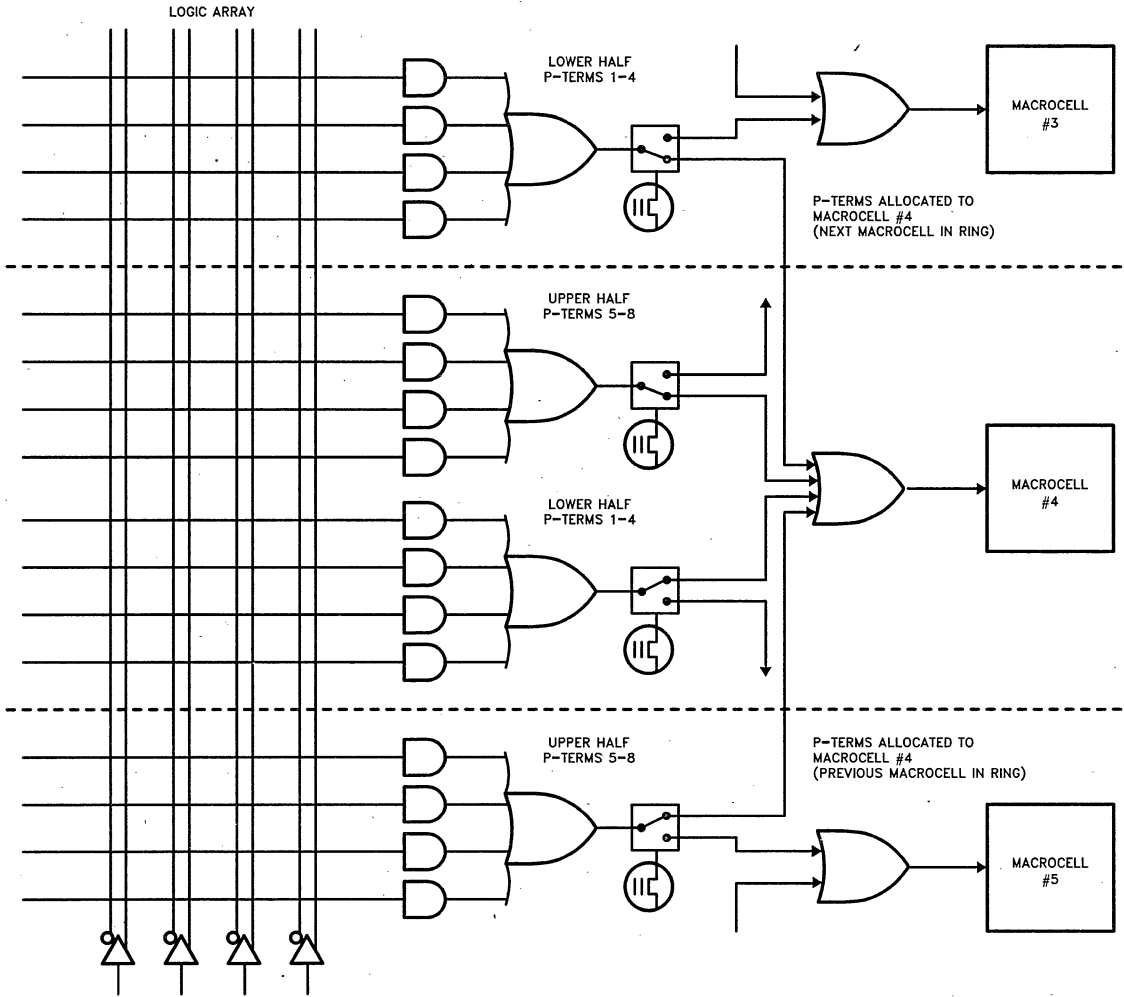
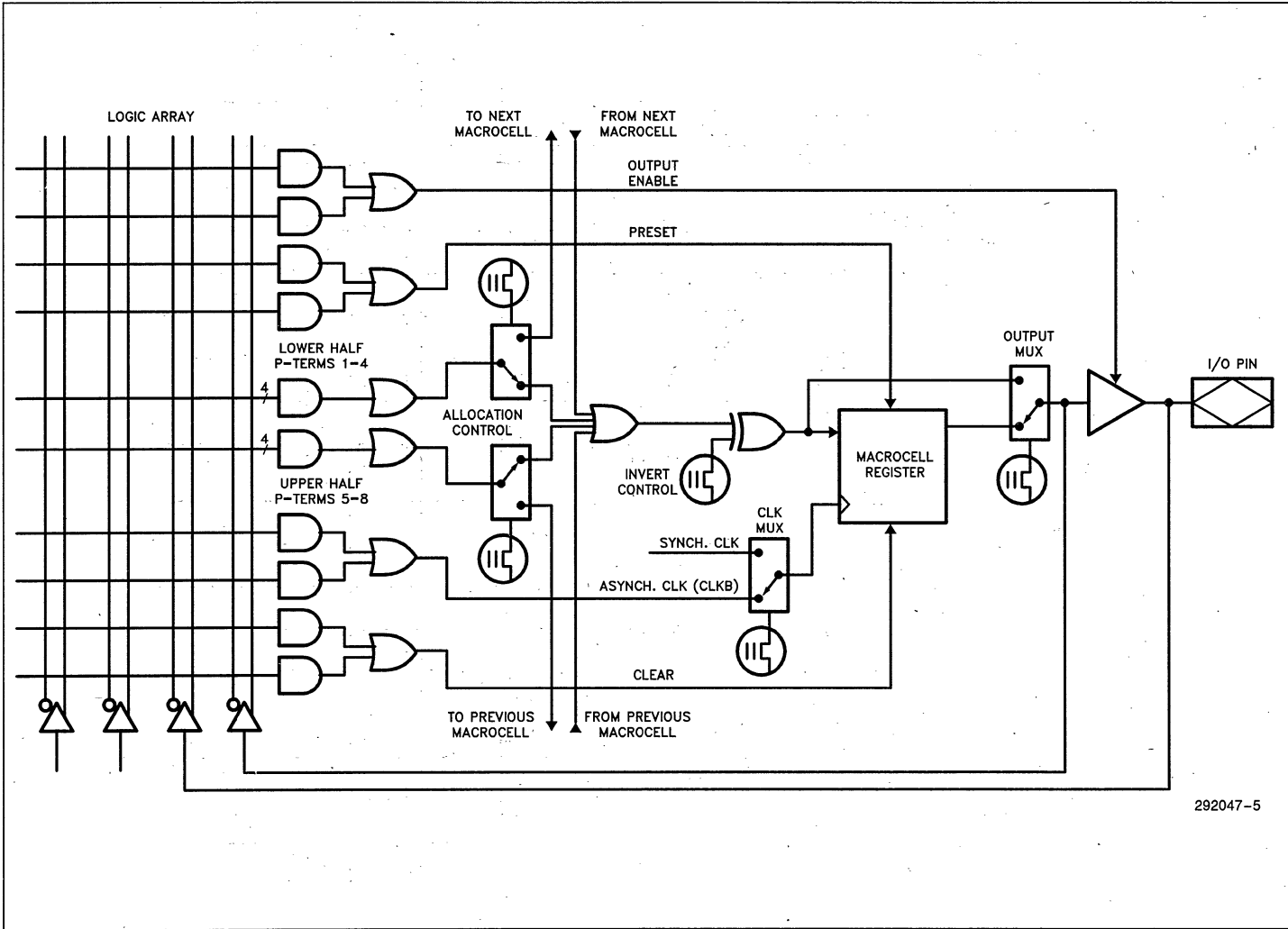


Figure 3. 5AC312 Input Structure



292047-4

Figure 4. Product Term Allocation (8 + 4 + 4)



292047-5

Figure 5. 5AC312 Basic Macrocell Structure

Since it is a CMOS EPLD, the 5AC312 has very frugal impact on the current allotted by the PS/2 power supplies. The device consumes much less power than an equivalent TTL or PAL implementation.

**5AC312 POS IMPLEMENTATION**

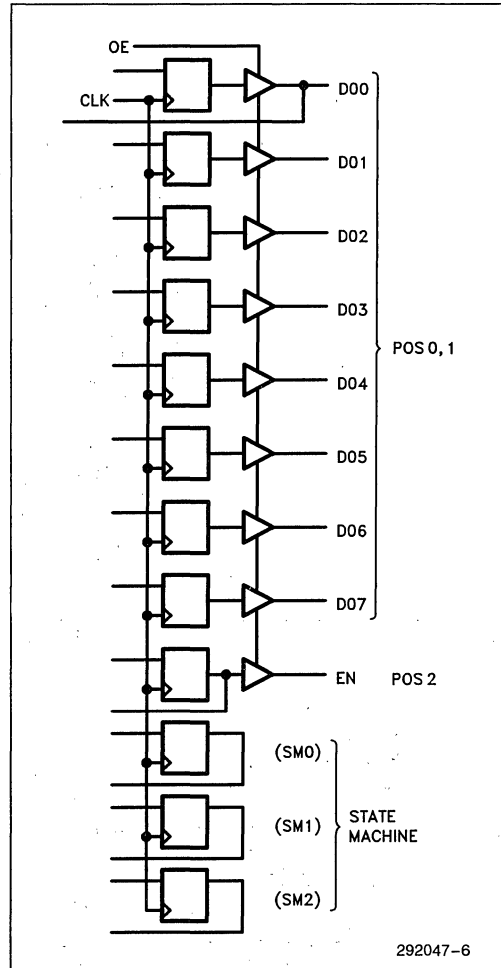
The POS requirements placed on our modem adapter are easily met by putting the 5AC312 to some creative use. In terms of performance, the 25 ns propagation delay and capability to operate to 40MHz is more than adequate for the system requirements during setup mode.

**RESOURCE ALLOCATION**

POS registers 0, 1, and 2 can be easily accommodated with 12 macrocells. Eight macrocells are used to load and output the ID bytes from POS 0 and 1. One macrocell is used as the LSB of POS register 2. The remaining three macrocells make up a state machine that internally sequences through the setup mode. The partitioning used for this design is shown in Figure 6.

The state diagram shown in Figure 7 explains the operation of the design. During S0 (state 0), the 5AC312 idles until -CD SETUP is driven active. The adapter is already disabled and POS 2 is zero because during power-up and reset the 5AC312 registers come up as logic 0. When -CD SETUP is driven active, the 5AC312 goes to S1 and loads the first ID byte, FFH. It remains in S1 while waiting for a READ POS 0 command. As soon as POS 0 is read the state machine cycles to S2 and outputs FFH which is the least significant byte for our adapter. Once READ POS 0 is inactive, the 5AC312 cycles to S3 where it loads the second ID byte (7FH) and waits for READ POS 1 active. The last ID byte is put on the bus when READ POS 1 comes and the machine goes to S4. Since we know that the next two setup operations are I/O writes, the 5AC312 remains in S5 while POS 2 is disabled and enabled per the Micro Channel bus specification. This operation, which is a bit write of a register, is easily done by using the 5AC312's dual feedback capability. While bit 0 of POS 2 uses the D00 line, internally it gets routed to a separate register. Without dual feedback this internal transceiver function would be impossible to implement.

The IBM Technical Reference Manual provides a table for suggested ID bytes arranged by adapter type. The modem card falls under the category of storage device, so 7FFFH was chosen. While IBM has assigned unique IDs for its own cards the third party choices are up



**Figure 6. Register Allocation**

for grabs. It is conceivable that more than one company may assign the same ID to their own cards. In this case, companies that implement the POS function in discrete TTL or a custom IC may have a problem. That is, they'll either have to re-do the design, which could be expensive, or, cut and jumper the board, which, goes against the Micro Channel specification and still costs.

Since the 5AC312 is re-programmable, the risk of conflicting IDs is minimized since all that is needed

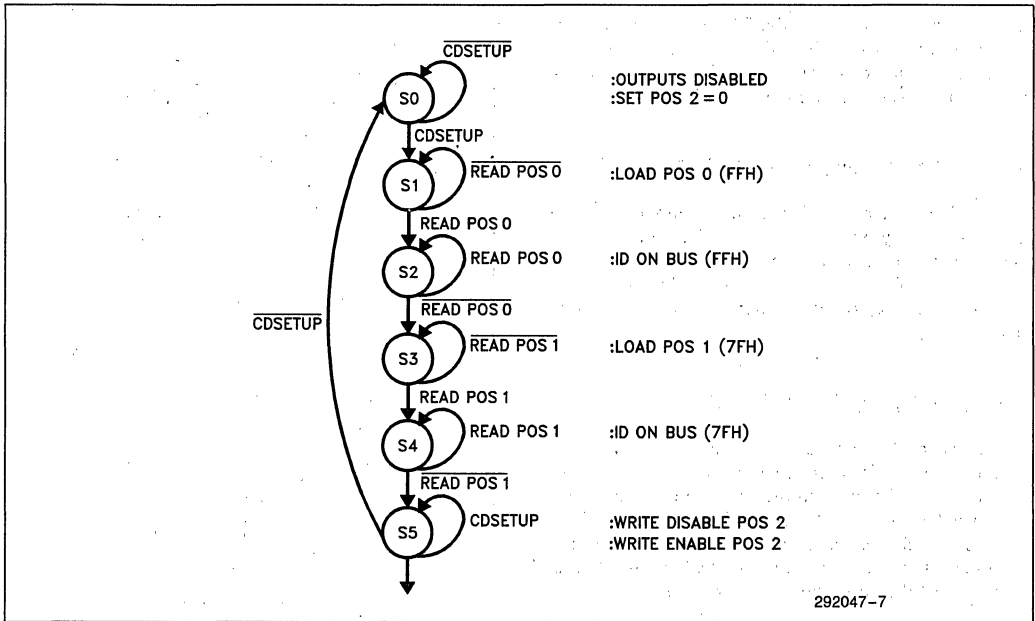


Figure 7. State Diagram

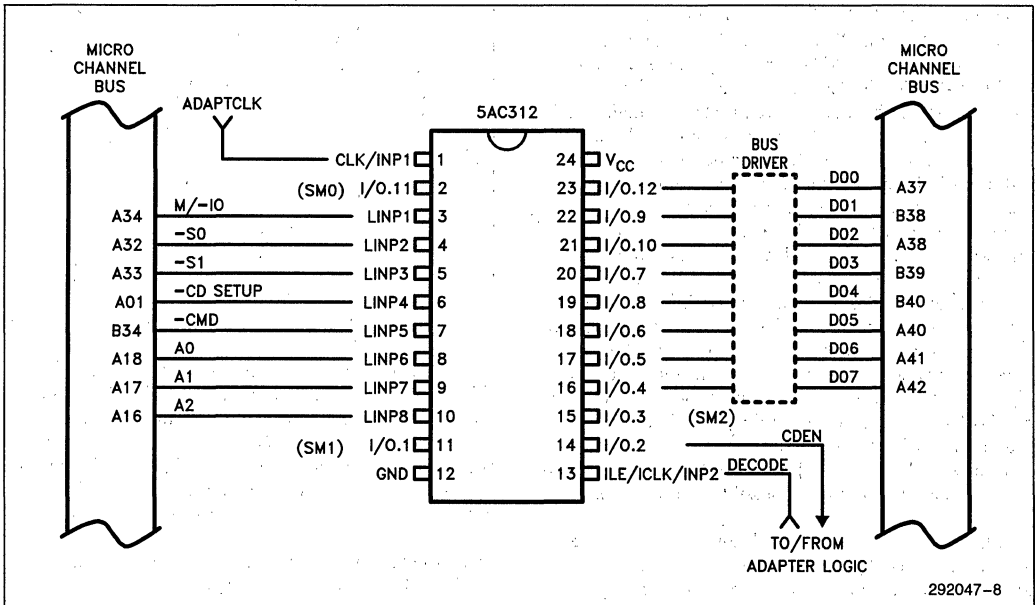


Figure 8. POS Pinout

is to burn another EPLD with the appropriate bytes. The pinout for the 5AC312 POS implementation is shown in Figure 8. Note that a bus driver is required to support the current levels on the Micro Channel. The definition for the signals can be found in the IBM literature, but briefly is as follows:

M/-I/O	Memory or Input/Output.
-S0, -S1	Status bits 0 and 1.
-CD SETUP	Card Setup.
-CMD	Command.
A0-A3	Address bits 0-3.
D00-D07	Eight bit data bus.
DECODE	Adapter decode for the higher order 16 bit address bus.
EN/-DIS	Enable/Disable adapter.

## DESIGN FILES

The 5AC312 was developed and programmed using the Intel IPLSII 1.5 EPLD software. This software provides a variety of entry methods like Boolean equation, state machine, and schematic capture. For this particular design, equation entry was the most convenient since the implementation was done in one IC.

The minimized and ordered .LEF (Logic Equation File) for the 5AC312 is shown in Figure 9. A quick glance at a few items really points out the power of the 5AC312. For example, the equation for variable ENd has 11 p-terms. Without p-term allocation this would not have fit unless it was done with two macrocells, which is wasteful. Also, the output enable control signal OE contains two p-terms for each macrocell. Again, without the 5AC312 some work-around may have been possible. But its unlikely that the design could have fit in one device, resulting in a functional but not too optimal solution. Finally, in keeping with good state machine design practices, all of the critical bus signals were received by registers with the EPLD primitive RINP (Registered Input).

A description of the Micro Channel Adapter Description File and the Configuration Utilities is beyond the scope of this article. The IBM literature is very descriptive in how adapters are setup and configured, and the reference section contains the names of the pertinent documents.

This modem adapter POS implementation was easily done in one 5AC312 device. Adding other Micro Channel interface features like arbitration or interrupt sharing would overburden it. Since this adapter did not have those requirements it was not a problem. However, a complete POS implementation with Option Select Bytes and other features could be done with the 5AC312's big brother, the 5AC324.

## SUMMARY

Interfacing to the PS/2 Micro Channel can be a difficult chore when using PLDs or other solutions that are not flexible or powerful enough. However, the 5AC312 with its powerful features like product term allocation is a giant step in the right direction to making the job easier.

## ACKNOWLEDGEMENTS

Many thanks to Thom Bowns and Dan Smith for their help with this article.

## REFERENCES

1. IBM Personal System/2, Model 80, Technical Reference.
2. IBM Personal System/2, Hardware Maintenance Reference.
3. Intel 5AC312 EPLD Data Sheet.
4. *Electronics* "Inside Technology", September 17 1987.

```

THOM BOWNS
INTEL
DECEMBER 4, 1987
1
001
5AC312
Implements POS for the PS/2 using a 5AC312.
LEF Version 1.5 Baseline 4.1i 21 Nov 1987
OPTIONS: TURBO=ON
PART:
    5AC312
INPUTS:
    MIO@3, nS0@4, nS1@5, nCDSETUP@6, nCMD@7, A0@8, A1@9, A2@10, DECODE@13,
    CLK@1, D0@23
OUTPUTS:
    D00@23, D01@22, D02@21, D03@20, D04@19, D05@18, D06@17, D07@16, EN@14,
    SM0@2, SM1@11, SM2@15
NETWORK:
    IRE = CLKB (CLK)
    CLK = INP (CLK)
    MIO = RINP (MIO, IRE, GND, GND)
    nS0 = RINP (nS0, IRE, GND, GND)
    nS1 = RINP (nS1, IRE, GND, GND)
    nCDSETUP = RINP (nCDSETUP, IRE, GND, GND)
    nCMD = RINP (nCMD, IRE, GND, GND)
    A0 = RINP (A0, IRE, GND, GND)
    A1 = RINP (A1, IRE, GND, GND)
    A2 = RINP (A2, IRE, GND, GND)
    DECODE = INP (DECODE)
    D0 = INP (D0)
    D00 = R0NF (D00d, CLK, GND, GND, OE)
    D01 = R0NF (D01d, CLK, GND, GND, OE)
    D02 = R0NF (D02d, CLK, GND, GND, OE)
    D03 = R0NF (D03d, CLK, GND, GND, OE)
    D04 = R0NF (D04d, CLK, GND, GND, OE)
    D05 = R0NF (D05d, CLK, GND, GND, OE)
    D06 = R0NF (D06d, CLK, GND, GND, OE)
    D07 = R0NF (D07d, CLK, GND, GND, OE)
    EN, EN = R0RF (ENd, CLK, GND, GND, VCC)
    SM0 = NORF (SM0d, CLK, GND, GND)
    SM1 = NORF (SM1d, CLK, GND, GND)
    SM2 = NORF (SM2d, CLK, GND, GND)

```

292047-9

Figure 9. POS Design File

## EQUATIONS:

```

SM2d = SM2' * SM1 * SM0 * nCDSETUP' * MIO' * nS0' * A2' * A1' * DECODE *
      nS1
      + SM2 * SM1' * nCDSETUP';

SM1d = SM2' * SM1' * SM0 * nCDSETUP' * MIO' * nS0' * A2' * A1' * DECODE *
      nS1
      + SM2' * SM1 * SM0' * nCDSETUP'
      + SM2' * SM1 * nCDSETUP' * DECODE'
      + SM2' * SM1 * nCDSETUP' * A1
      + SM2' * SM1 * nCDSETUP' * A2
      + SM2' * SM1 * nCDSETUP' * nS1'
      + SM2' * SM1 * nCDSETUP' * nS0
      + SM2' * SM1 * nCDSETUP' * MIO;

SM0d = (SM2' * SM0 * MIO' * nS0' * nS1 * A2' * A1' * DECODE
      + SM2 * SM0' * MIO' * nS0' * nS1 * A2' * A1' * DECODE
      + SM1 * MIO' * nS0' * nS1 * A2' * A1' * DECODE
      + SM2 * SM1
      + nCDSETUP)';

Endd = D0 * MIO' * A2' * A1 * A0' * DECODE * SM2 * SM1' * SM0 * nS1' * nS0
      + nS0' * EN
      + nS1 * EN
      + SM0' * EN
      + SM1 * EN
      + SM2' * EN
      + DECODE' * EN
      + A0 * EN
      + A1' * EN
      + A2 * EN
      + MIO * EN;

D07d = SM2' * SM0'
      + SM2' * SM1';

D06d = (SM2 * SM1)';

D05d = (SM2 * SM1)';

D04d = (SM2 * SM1)';

D03d = (SM2 * SM1)';

D02d = (SM2 * SM1)';

D01d = (SM2 * SM1)';

OE = SM2 * SM1' * SM0' * MIO' * nS0' * A2' * A1' * DECODE * nS1
      + SM2' * SM1 * SM0' * MIO' * nS0' * A2' * A1' * DECODE * nS1;

D00d = (SM2 * SM1)';

```

ENDS

292047-10

Figure 9. POS Design File (Continued)



November 1988

**Designing with the  
5AC312/5AC324 EPLDs**

**DAVID BICKEL**  
PROGRAMMABLE LOGIC APPLICATIONS  
INTEL CORPORATION

Order Number: 292049-001

## INTRODUCTION

The Intel 5AC312 EPLD (Erasable Programmable Logic Device) was developed to break down certain existing PLD architectural barriers and meet increased performance needs. The Intel 5AC312 EPLD was designed by EPLD users with direct input from system designers. In the design process, emphasis was placed first on gate utilization, and then on density.

This application note highlights the advanced architecture and features of the 5AC312 EPLD and shows the benefits of designing with this new device over more traditional PLD architectures. These features include enhanced input structure with register/latch option on all input pins (synchronous or asynchronous operation); user-controllable, software-supported p-term allocation scheme in all macrocells; and multiple p-terms on control functions (asynchronous CLK, SET, RESET, OE).

It should also be noted that the features and information described here also apply to the new 5AC324 EPLD. The 5AC324 is basically a 24 macrocell version of the 5AC312.

## PROGRAMMABLE INPUTS

The 5AC312 was designed with a highly flexible macrocell and I/O structure allowing the device to implement both combinational and sequential logic functions. The enhanced input structure not only allows the device to latch and hold incoming data, but also to implement register-combinational-register logic to easily accommodate state machine designs. Figure 1 shows a global view of the 5AC312 architecture.

The 5AC312 is equipped with 8 user-programmable input structures that can each be configured to work in one of five modes: 1) synchronous D-type register, 2) asynchronous D-type register, 3) synchronous D-type latch, 4) asynchronous D-type latch, and 5) flow-through input. Each input can be configured independently of the others. The desired configuration is implemented through the programming of EPROM architecture control bits by the logic compiler under user-control.

## MACROCELL STRUCTURE

The 5AC312 also has a unique macrocell array structure that allows for user-controllable, software-supported product term allocation in each of its 12 macrocells. Each of the 12 macrocells also has a dual feedback option with independent feedback and I/O paths. Each macrocell has 16 product terms, 8 of which control the OE, PRESET, ASYNCHRONOUS CLOCK, and

CLEAR signals (2 p-terms per signal). The other 8 feed the data input to the macrocell and are split into two groups of four (upper half and lower half). See Figure 2. Each group of four can be allocated to an adjacent macrocell if needed.

As shown in Figure 1, the 12 macrocells of the 5AC312 are further divided into two "rings" with 6 macrocells per ring. Allocation of p-terms to adjacent macrocells can occur with a given ring. See Figure 3 for p-term allocation scheme.

Each macrocell register in the 5AC312 is also equipped with an asynchronous PRESET signal. The PRESET function can be constructed in more traditional architectural devices such as the 5C060 and 5C090 using combinational logic and feedback, however two macrocells are consumed in the process. To illustrate this difference, compare Figure 2 to the implementation shown in Figure 4. The PRESET function would require additional macrocells in traditional architectures if it were expanded beyond a single p-term.

## MULTIPLE P-TERMS

Multiple p-terms on the control functions (asynchronous CLOCK, PRESET, RESET, and OE) increases the efficiency of the device. Multiplexed I/O is accomplished by controlling the output buffer associated with each macrocell using the 2 p-terms set aside for implementing an OE function. Multiple p-terms create a means to avoid using macrocells for control logic. For example, it would take two macrocells in the 5C060 and 5C090 EPLD to drive the OE line by a 2 p-term signal. To illustrate, compare Figure 2, the 5AC312 macrocell structure, to Figure 5, a diagram of how a two p-term OE signal can be implemented in a 5C060 or 5C090 EPLD.

## P-TERM ALLOCATION

P-term allocation allows for more efficient use of p-terms and thus increased device utilization by raising the number of p-terms per macrocell to 16. P-term allocation, where p-terms are dedicated to certain macrocells, should not be confused with p-term sharing, where several macrocells can actually use the same p-terms. The p-term allocation scheme in all macrocells is user-controllable and software supported, and provides the ability to satisfy designs with large p-term requirements. P-term allocation is ideal for p-term intensive applications such as complex counters or comparators.

P-term allocation in the 5AC312 is used when a design requires one of the 12 macrocells to employ more than 8 p-terms. P-term allocation is simply the transfer

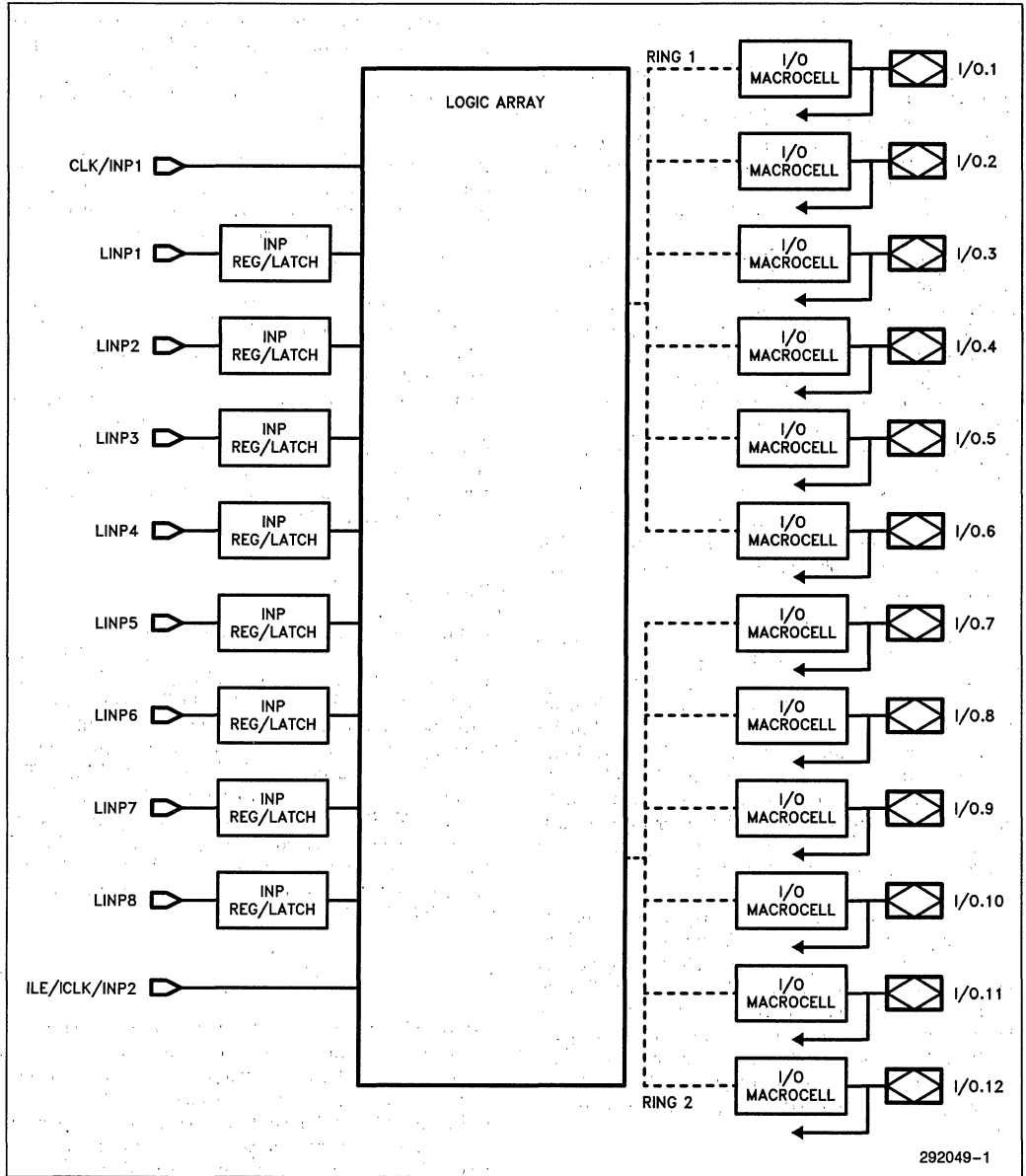
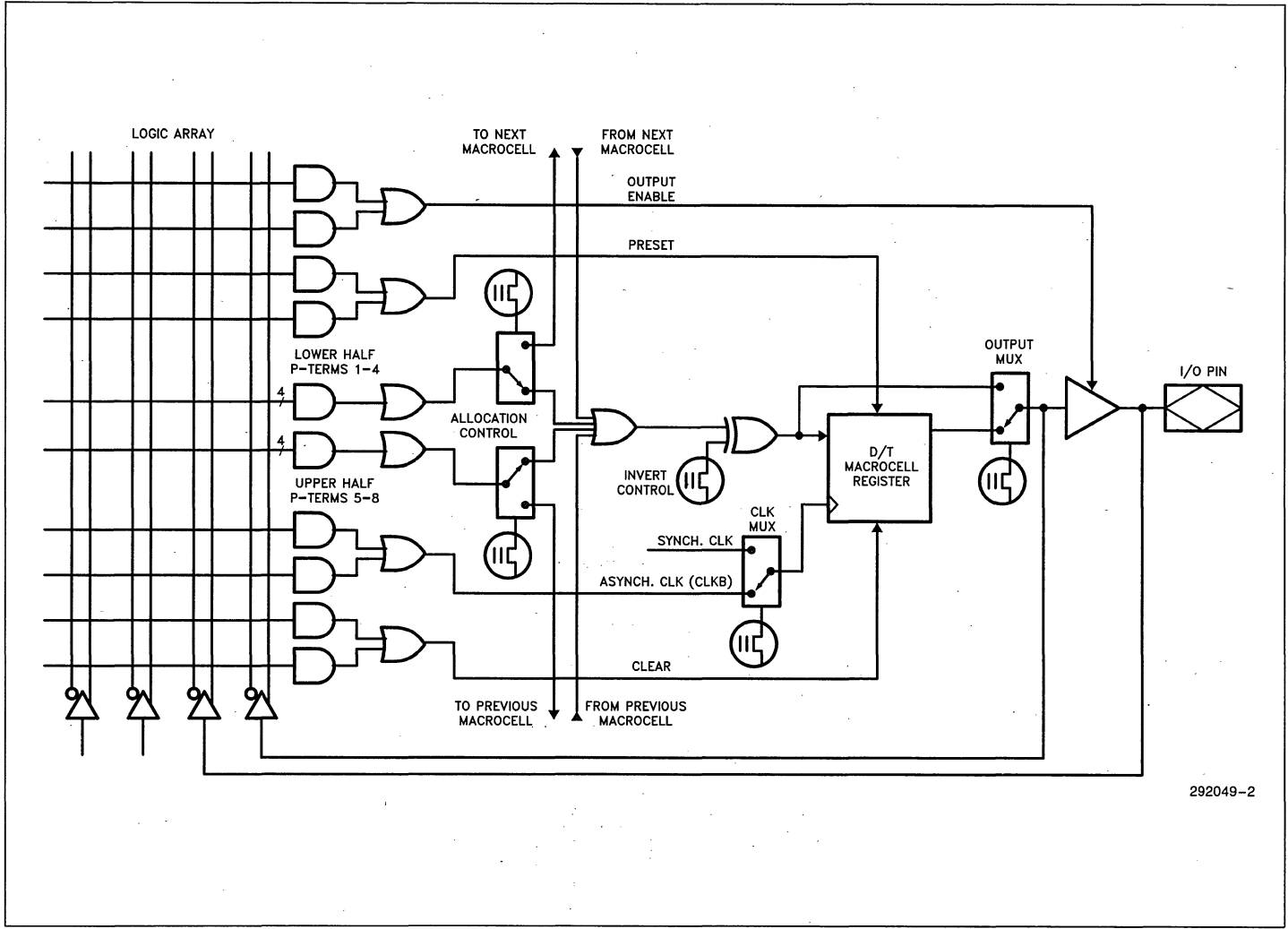


Figure 1. 5AC312 Architecture



292049-2

Figure 2. 5AC312 Basic Macrocell Structure

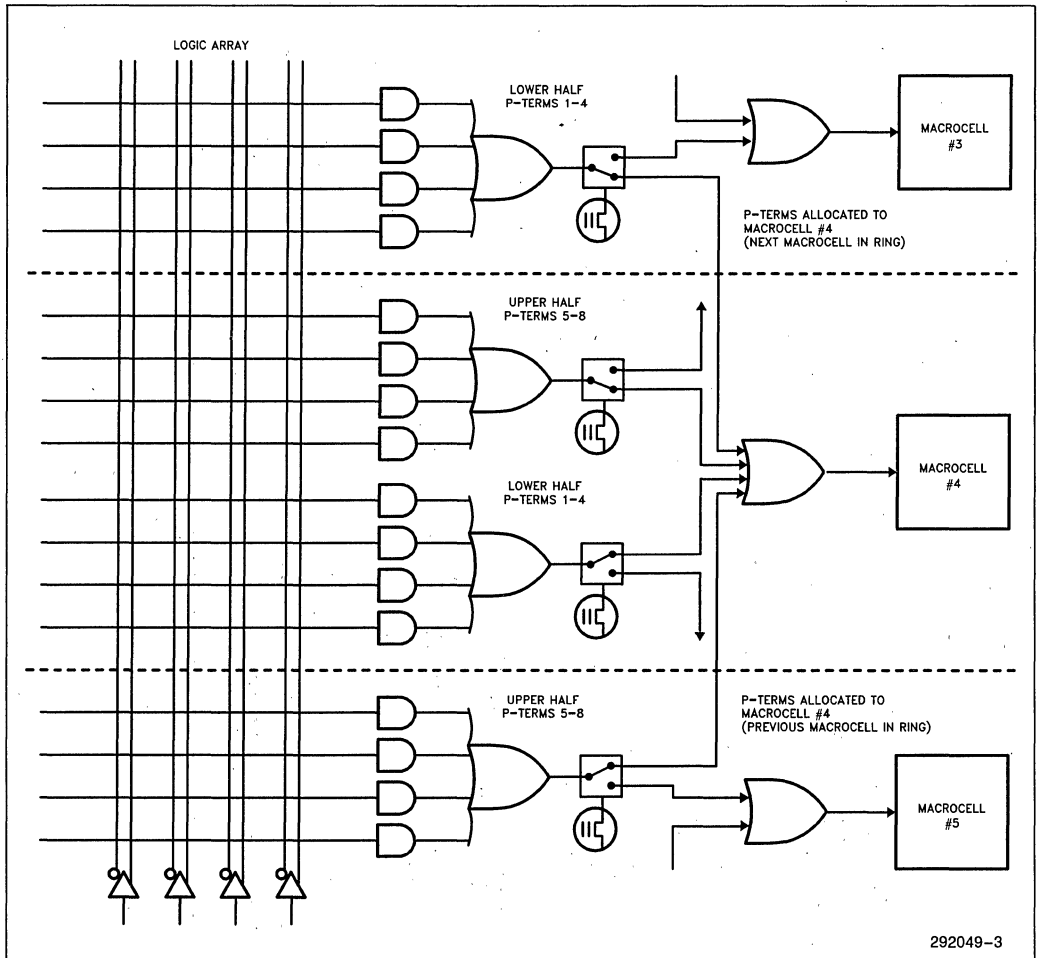
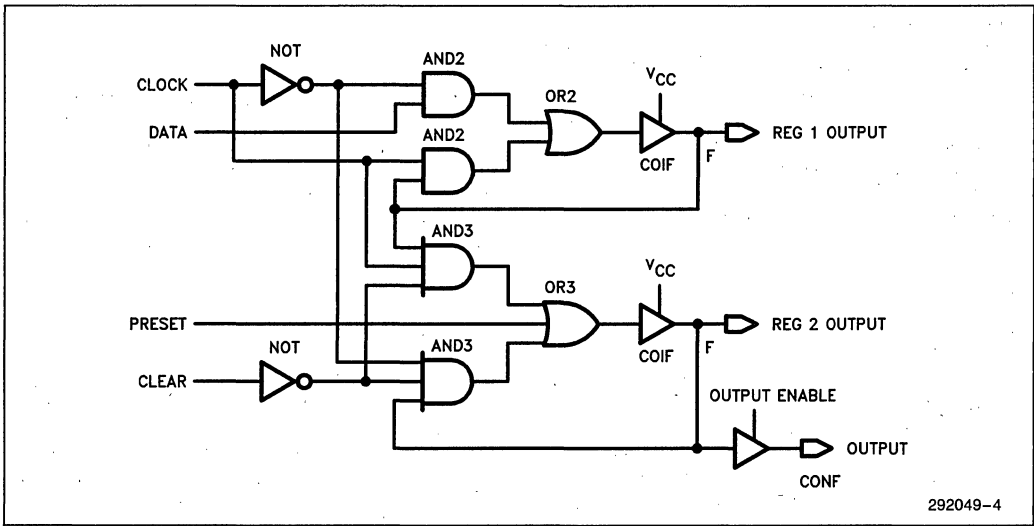


Figure 3. Product Term Allocation (8 + 4 + 4)

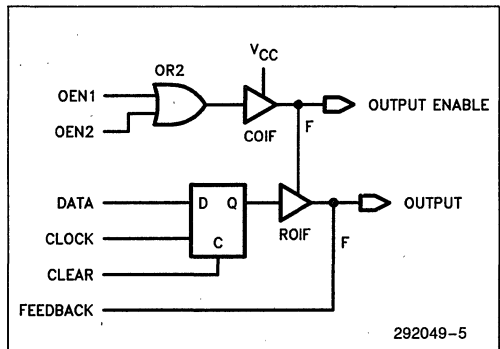


292049-4

Figure 4. Implementation of D Flip-Flop with Added Preset Function Using Combinational Logic

of logic resources (p-terms) from areas they are not being utilized to other areas within the chip where they are needed. As shown in Figure 3, each macrocell has the potential to borrow 4 more p-terms to add to the 8 it already has from each of its adjacent macrocells. This increases the maximum number of p-terms per macrocell to 16. Thus, any macrocell within the 5AC312 has the potential to satisfy logic functions requiring between 0 and 16 p-terms.

P-terms can be allocated in a "shift register" mode within each of the two rings of the macrocell; however, allocation of p-terms between rings is not possible. See Table 1 for a listing of adjacent macrocells within p-term allocation rings.



292049-5

Figure 5. Implementation of 2 P-Term OE Control Signal

Table 1. 5AC312 Product Term Allocation Rings

Ring 1			Ring 2		
Current Macrocell	Next Macrocell	Previous Macrocell	Current Macrocell	Next Macrocell	Previous Macrocell
1	2	8	7	8	12
2	3	1	8	9	7
3	4	2	9	10	8
4	5	3	10	11	9
5	6	4	11	12	10
6	1	5	12	7	11

A given macrocell's output structure is still available for use when some or all of its p-terms are allocated away. If all of the p-terms of one macrocell are allocated away to its respective adjacent macrocells, the data input to that macrocell defaults to GND. This polarity can be changed through programming of the invert select EPROM bit. The I/O register as well as all secondary controls to this I/O control block are still available and can be used as needed for design purposes.

**DUAL FEEDBACK**

The 5AC312 contains separate input and feedback paths (dual feedback) on each of the macrocell I/O control blocks. This allows designs to utilize input pins when the associated macrocells have been assigned a no output with buried feedback primitive. Multiplexed I/O is accomplished by controlling the output buffer associated with each macrocell using the 2 p-terms that implement the OE function. Registered outputs may be clocked from the synchronous CLK/INP1 pin or asynchronously clocked by the 2 p-terms available for ASYNCH\_CLK.

**POWER ON CHARACTERISTICS**

Another feature of the 5AC312 is its power-on characteristics. The I/O registers of the 5AC312 experience a reset to their inactive state upon Vcc power-up. Using the PRESET function available for each macrocell allows any particular register preset to be achieved after power-up. The inputs and outputs of the 5AC312 begin responding approximately 10 μs (6 μs typical) after Vcc power-up or after a power-loss/power-up sequence.

**POWER DOWN MODE**

A trade-off between power consumption and speed is possible when using the 5AC312 by programming the "Turbo Bit". Left unprogrammed and with no transition occurring at the device inputs for a period of approximately 100 ns, the device powers-down the internal array while holding the outputs at their previous levels. At the next input transition occurrence, the 5AC312 powers-up the array and reacts to the change in input conditions. If the "Turbo Bit" is programmed, the power-down circuitry is disabled and the device will not power-down even if there are no more transitions. The array power-up sequence requires an additional 20 ns of propagation delay. Power supply current during power-down is no more than 120 μA. See Figure 6.

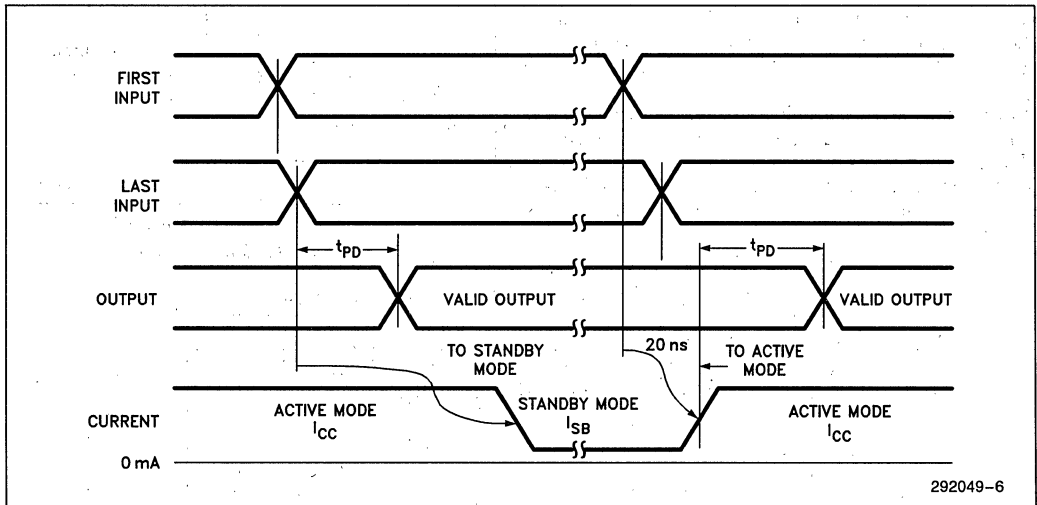


Figure 6. 5AC312 Standby and Active Mode Transitions

**EXAMPLE**

An example application for the 5AC312 can be shown by replacing a PAL\* 20R6 and a 374 D type flip-flop in a design due to a power constraint. The same implementation can be achieved consuming less power using one 5AC312 EPLD. Compare Figures 7 and 8. Straight jumpers can be substituted in the PC board where the 374 sits, and since the clock signal is already available on the PAL socket, it can be internally routed to clock the input registers of the 5AC312. The 5AC312 can then be programmed to match the existing pin assignments and therefore require no PC board re-layout. The internal circuitry of the 5AC312 allows the EPLD to act as both a D type flip-flop and a PAL.

**SUMMARY**

The 5AC312 EPLD, which uses advanced CHMOS EPROM cells as logic control elements instead of polysilicon fuses, represents an innovative device to help overcome the primary limitations of standard PLDs. With its advanced features, proprietary architecture and macrocell structure, the 5AC312 is capable of implementing high performance logic functions more effectively than was previously possible. The p-term allocation scheme is a unique feature, increasing the efficiency of the device immensely. The PRESET signal and 2 p-term control lines are also features giving the 5AC312 added efficiency in many designs.

These same architectural features have been included in the 5AC324 EPLD, making that device ideal for even higher integration applications. Refer to the 5AC324 Data Sheet for details on that device.

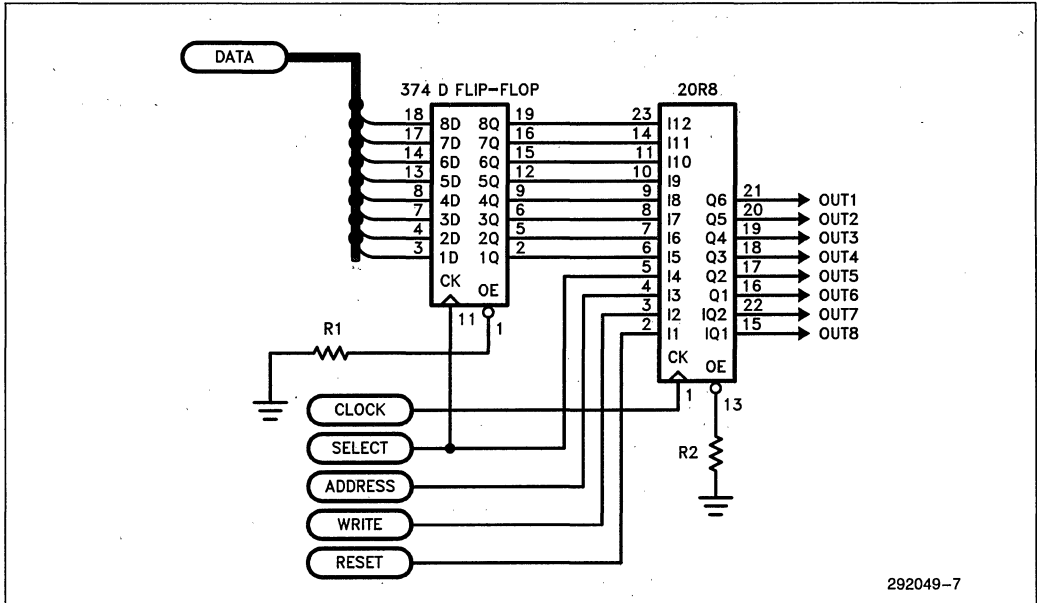
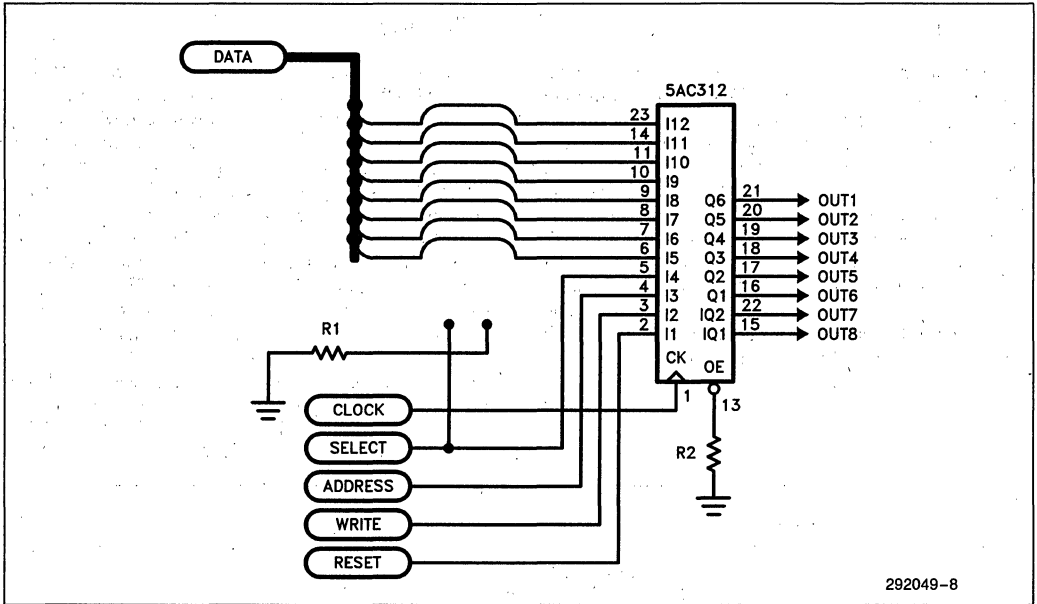


Figure 7. Original Implementation Using a 374 Flip-Flop and A PAL20R6

\*PAL is a registered trademark of Monolithic Memories, Inc.





292049-8

Figure 8. Example Implementation Using the 5AC312

PROGRAMMABLE AND/ALLOCATABLE OR BASED EPLD  
ADDRESSES THE NEEDS OF  
COMPLEX COMBINATIONAL AND SEQUENTIAL DESIGNS

Todd K. Koelling  
Applications Engineer  
Intel Corporation  
1900 Prairie City Road  
Folsom, CA 95630

INTRODUCTION

Matching programmable logic applications with programmable logic devices has become a difficult task. Increasing demands for higher integration, higher performance and lower cost continue to drive system design engineers on to new technologies. The programmable logic industry has adeptly responded by supplying a wide variety of devices. At times, however, it is hard to differentiate these devices and to determine which makes the best solution for a particular application.

In a small way, this paper will attempt to differentiate devices and to determine which devices make the best solutions for groups of applications. This task will be accomplished by taking a general look at applications, the history of PLD arrays and a new device which solves several design problems.

APPLICATIONS

College textbooks<sup>1</sup> on digital design teach that fundamentally there are only two types of applications: combinational and sequential. A combinational circuit generates outputs based on the immediate status of a group of inputs. A sequential circuit uses some mechanism to store data before generating the next set of outputs.

Inside combinational and sequential circuits are two fundamental elements: gates and registers. Gates are the prime component of combinational circuits where the output is an immediate function of the input. Registers are the static storage element added in sequential circuits to latch and hold data until the next cycle.

Figure 1 displays gates and registers graphically. The coordinates measure registers along the x-axis and gates along the y-axis. In this space, any combinational or sequential application can be displayed.

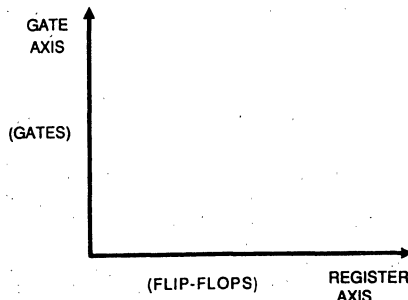


Figure 1: Gate/Register Coordinates

Common TTL functions are easily graphed. Figure 2 displays a comparator, storage register, shift register and counter. The comparator is a combinational circuit (purely gates) and hence lies along the gate axis. The storage register, on the other hand, is purely flip-flops and hence lies along the register axis. The shift register is primarily flip-flops -- placing it close to the storage register -- but it includes some gate logic, thus moving it up the gate axis. The counter is a good example of a function that lies somewhere in-between the two axes. The counter must store its current state, and thus leans heavily upon the registers, but it also uses a significant amount of gate logic to generate the next count state. The inclination toward the gate or register axis depends on the features the counter incorporates. Up and down count operation, clear and preload functions, and count enable/disable circuitry, all move the counter increasingly toward the gate axis. The magnitude of the counter (as with the other functions) depends on the number of bits and features it includes. That is, a 16-bit counter is twice as large as an 8-bit counter which is twice as large as a 4-bit counter, provided the feature set remains the same.

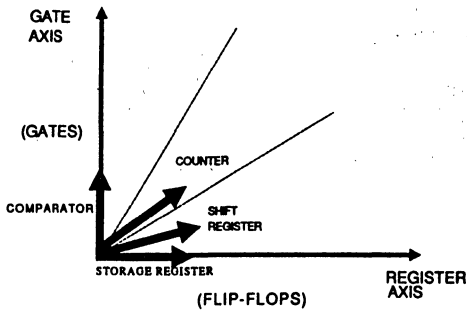


Figure 2: Common Functions on Coordinates

Having examined functions, it is easy to examine complete designs. Each design can be decomposed into a group of function blocks. This is basically the undoing of the design process in which functions are grouped together to meet some high-level purpose. If all the functions for a design are added together, a vector for the complete design is formed (Figure 3). Now, rather than just a MAGNITUDE (size), a DIRECTION is also available for each design.

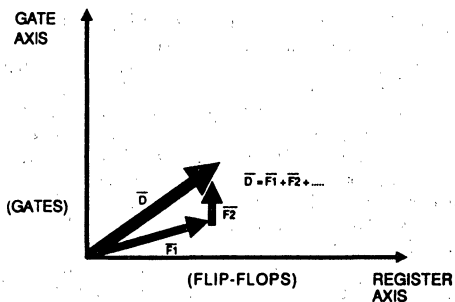


Figure 3: Design Vector

With designs defined and considered, it is useful to take a step back and once again look at the gate-register coordinate system. Any design that lands in the upper third of the coordinate system consists primarily of gates and thereby can be considered "highly combinational". Any design that lands in the lower third of the coordinate system can be considered as "highly registered" or "register intensive". Examples of this include data transfer and data storage applications. Any design that lands in the center third of the coordinate

system represents a healthy mix of both gates and registers. This means it is probably a state machine or some sort of sequential application. Hence, the middle third region will be called the "state machine" region, though some state machines may land in the other two regions. The coordinate system with the three regions segmented and labelled is shown in Figure 4.

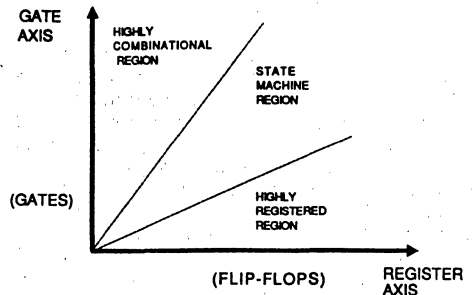


Figure 4: Application Regions

ARRAY ARCHITECTURES

Through the years, programmable logic devices have evolved by trying to meet the needs of combinational and sequential applications. This has been accomplished through higher integration, higher flexibility and higher performance and has resulted in the myriad of PLDs available today. Though the features have varied and expanded immensely, the core of the programmable logic device has remained virtually the same. It is this core -- the implementation of the combinational logic array -- which deserves a closer look.

A Brief History of PLD Array Architectures

Programmable logic first appeared in 1975 with the introduction of the Field Programmable Logic Array (FPLA). With its programmable AND/programmable OR array, the FPLA was extraordinarily powerful for integrating combinational logic. Registers were later added to the FPLA outputs, creating the Field programmable Logic Sequencer (FPLS). This device better attacked the needs of sequential applications.

In 1978, the FPLA was honed to a programmable AND/fixed OR array with the

introduction of the PAL<sup>2</sup> device. Due to its ease of design and CAD tool support, the Boolean sum-of-products part quickly became the designer's choice. In addition, the PAL included registered versions with registered outputs and registered feedback. These two attributes made the devices ideal for state machines.

In essence, the last array architecture, fixed AND/programmable OR, has been around for many years in the form of the PROM and other PROM-based logic devices. It wasn't until 1984, however, that the SRAM-based LCA introduced the fixed AND/programmable OR array to the designer in an expedient form. Due to its large number of registers per device (122 or more), the LCA has provided an excellent programmable solution for register-intensive applications.

#### Mapping of Array Architectures

Returning to the application graph developed earlier, each architecture and device is displayed (Figure 5). The combinational power and flexibility of the programmable AND/programmable OR architecture places the PLA and PLS devices in the highly combinational region.

The registered output with registered feedback PALs are optimum for state machines while some PALs, such as the 22V10, are geared to provide a large amount of combinational logic per register. Thus, most of the registered PALs fit in the state machine region, while a few, like the 22V10, move up into the highly combinational region. (Logic PALs such as the 16L8 and 20L8 fit along the y-axis as they are purely gates).

Architecturally, most EPLDs and EEPLDs have followed the track set by the PAL. They have added higher input and output flexibility, higher integration, and other worthwhile features, but the core of the architecture has remained the same -- the programmable AND/fixed OR array. Thus the EPLD and EEPLD fit the same application regions as the PAL.

The highly registered region is best covered by the LCA where its high register count and fast toggle rates are well utilized.

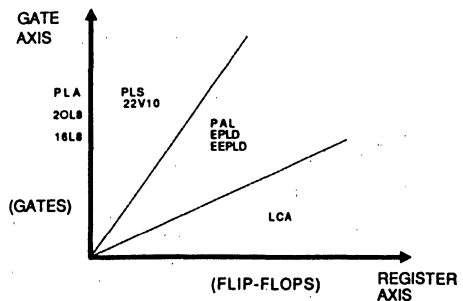


Figure 5: Device Application Areas

Where will the next innovation be? The answer may not be in an innovation, but in a very practical advancement from Intel in 1988.

#### THE PROGRAMMABLE AND/ALOCATABLE OR ARRAY

Via a novel product term allocation<sup>3</sup> scheme, the best of the PLA and the PAL have been married into a single device. In the programmable AND/allocatable OR array architecture, each macrocell output can have anywhere from 0 to 16 AND terms allocated to its OR gate in increments of four. In this manner it is very similar to the PLA with its OR input flexibility, yet it retains the Boolean sum-of-products architecture used in the PAL which offers easy design entry and optimized CAD tools for minimization and fitting. The net result is a novel approach for efficiently addressing the needs of complex combinational as well as sequential applications.

#### How P-term Allocation Works

Figure 6 displays a macrocell of the 24-pin, 12 macrocell 5AC312. Each macrocell contains eight product terms grouped into two blocks of four. Each block has its own control switch that allows it to be retained by the macrocell or lent to a neighbor. Likewise, each macrocell can ignore or borrow a block from each of its neighbors. With two neighbors for each macrocell, each macrocell can have a total of 0, 4, 8, 12 or 16 p-terms allocated to its OR gate. This allows the device to shift resources toward those equations that require a high number of p-term inputs away from those that do not.

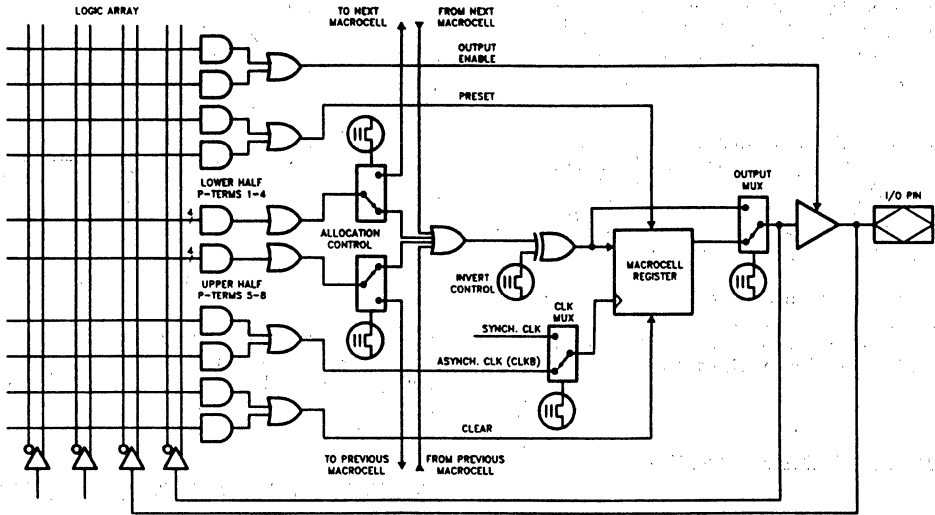


Figure 6: 5AC312 Macrocell Architecture

Combinational Example:  
Micro Channel<sup>4</sup> Decoder

Figure 7 shows the 5AC312 pinout for the micro channel decode logic on a PS/2<sup>4</sup> Ethernet<sup>5</sup> adapter. Based on the address lines, micro channel bus cycle signals, and POS/command register inputs, the 5AC312 generates the card select feedback (CDSFDBK#) and card data size 16 (CDDS16#) return signals for the micro channel. It also generates an asynchronous board select signal (ABDSEL#) that feeds an on-board arbiter. As memory is shared between the PS/2 motherboard and the adapter 82586 LAN coprocessor, the equations for all three of these outputs become quite complex (Figure 8).

Even after processing the equations through the industry-standard Espresso<sup>6</sup> minimizer employed by the Intel Logic Optimizing Compiler(LOC), the nested equations in Figure 8 expand out into the minimized equations in Figure 9. The board select, card select feedback, and data size 16 signals are 15, 14 and 14 product terms, respectively. This is not a problem for the 5AC312, however, as the LOC software

automatically allocates the device's resources to four, four p-term blocks for each signal. The three equations use a total of 12 out of the 24 four p-term blocks available in the device -- 50 % of its combinational capacity.

**Micro Channel Decode Logic**

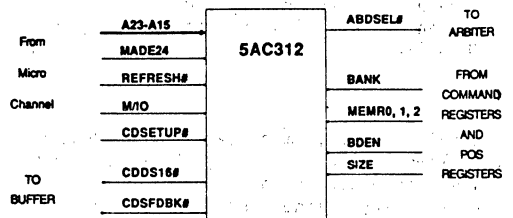


Figure 7: Pinout for Ethernet Adapter Decode Logic

Though not used in this application, since the CDSFDBK#, CDDS16#, and ABDSEL# are all driven off unlatched address decodes, another feature which makes the 5AC312 attractive for address decoding and bus interfacing is its latched input capability. On the IBM PC/AT<sup>7</sup> bus, for example, the

```

EQUATIONS:

AM1 = A23 & A22 & A21 & A20 & MEMR2 & !SIZE;           % EXTENDED RANGE 32 K %
AM2 = A23 & A22 & A21 & A20 & MEMR2 & SIZE;           % EXTENDED RANGE 64 K %
AM3 = !A23 & !A22 & !A21 & !A20 & !MEMR2;             % NOT EXTENDED RANGE %

AM4 = A19 & A18 & !A17 & !SIZE &                       % EITHER, 32 K %
      ( !A16 & !MEMR1 & !A15 & !MEMRO # % MATCH XC0000 %
        !A16 & !MEMR1 & A15 & MEMRO # % MATCH XC8000 %
        A16 & MEMR1 & !A15 & !MEMRO # % MATCH XD0000 %
        A16 & MEMR1 & A15 & MEMRO ); % MATCH XD8000 %

AM5 = A19 &                                             % EXTENDED, 64 K %
      ( ( !A18 & A17 & !MEMR1 & !A16 & !MEMRO ) # % MATCH FA0000, 64 K %
        ( !A18 & A17 & !MEMR1 & A16 & MEMRO ) # % MATCH FB0000, 64 K %
        ( A18 & !A17 & MEMR1 & !A16 & !MEMRO ) # % MATCH FC0000, 64 K %
        ( A18 & !A17 & MEMR1 & A16 & MEMRO ) ); % MATCH FD0000, 64 K %

AM6 = A19 & A18 & !A17 & SIZE &                       % NOT EXTENDED, 64 K %
      ( !MEMRO & !A16 # % MATCH OC0000, 64 K %
        MEMRO & A16 ); % MATCH OD0000, 64 K %

AM7 = MADE24 & REFRESH_ ;                               % ALL CYCLES %

AM8 = SETUP_ & M_IO;                                   % MEMORY CYCLES %

AM9 = !SETUP_ & !M_IO;                                  % SETUP CYCLES %

CDDS16= !(BDEN & !BANK & AM7 & AM8 &                 % ENALBED, SRAM, NOT SETUP AND %
          ( AM1 & AM4 # % EXTENDED,32 OR %
            AM3 & AM4 # % NOT EXTENDED, 32 OR %
            AM2 & AM5 # % EXTENDED,64 OR %
            AM3 & AM6 ) ); % NOT EXTENDED, 64 %

CDSFDBK= !( BDEN & AM7 & AM8 &                       % ENABLED, NOT SETUP AND %
            ( AM1 & AM4 # % EXTENDED,32 OR %
              AM3 & AM4 # % NOT EXTENDED, 32 OR %
              AM2 & AM5 # % EXTENDED,64 OR %
              AM3 & AM6 ) ); % NOT EXTENDED, 64 %

ABDSEL = !(AM1 & AM4 & AM7 & AM8 # % EXTENDED,32 OR %
           AM3 & AM4 & AM7 & AM8 # % NOT EXTENDED, 32 OR %
           AM2 & AM5 & AM7 & AM8 # % EXTENDED,64 OR %
           AM3 & AM6 & AM7 & AM8 # % NOT EXTENDED, 64 OR %
           AM7 & AM9 ); % SETUP CYCLE %

END$

```

Figure 8: Micro Channel Decoder  
Nested Input Equations

upper address lines (LA17 - 23), may need to be latched with the active edge of the bus address latch enable (BALE). Elsewhere in the micro channel interface, latching the address and status signals may be useful as both will go invalid about halfway through the command (CMD#) cycle. The 5AC312 is capable of latching up to eight inputs, each of which can be enabled individually or by the global latch enable.

The decode design could be implemented in a PLA or a 22V10, but the 5AC312 offers more strength over the PLA and more flexibility over the 22V10. Even with 32 and 48 p-terms feeding the programmable OR gates, most 24-pin PLAs would have trouble fitting these equations. With its fixed allocation architecture of 8, 10, 12, 14 and 16 p-terms, the 22V10 offers p-term resources comparable to those of the 5AC312 and can handle the equations. With configurable

p-term allocation, however, the 5AC312 offers a more flexible solution. In cases where latching needs to be done, the address latching must be done external to the 22V10 since it does not have the ability to directly latch inputs. This impacts both board space and performance in a negative fashion.

### Sequential Features

In addition to the allocatable OR architecture, the 5AC312 offers a host of other enhancements: separate register and pin feedback paths, register preset, and two product terms on clock, clear, preset and output enable control lines (Figure 6). These features, on top of the programmable AND/allocatable OR array, make the 5AC312 ideally suited for complex sequential designs.



Sequential Example:  
Programmable Baud Rate Generator

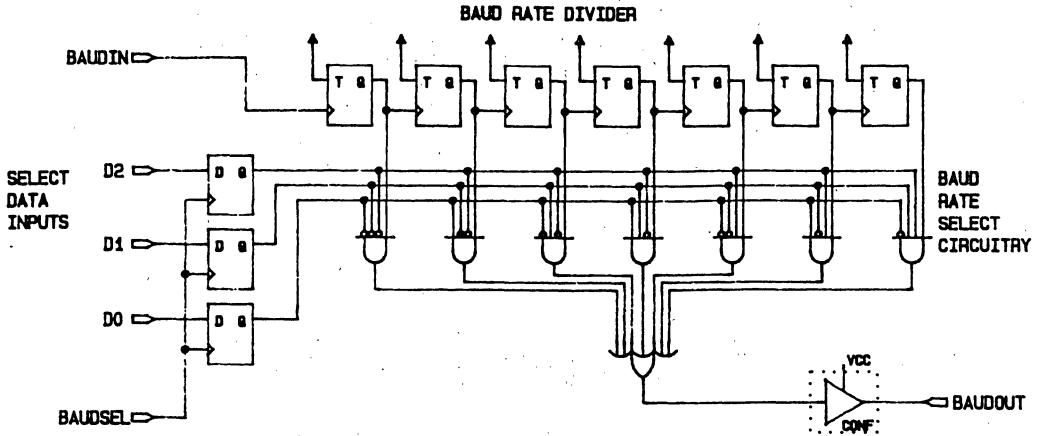


Figure 10: Programmable Baud Rate Generator Circuitry

Though not a state machine, the programmable baud rate generator circuitry for the Intel 8251 Universal Synchronous/Asynchronous Receiver/Transmitter does have a good mixture of gates and register functions, qualifying it for the state machine application region. The input baud rate (BAUDIN) is divided down to lower baud rates through a series of toggle flip-flops. Then, based on the select data stored in the D2, D1, D0 flip-flops, one of the divided-down baud rates is selected and sent out on the baud rate out pin (BAUDOUT).

Implementing the design in a standard 24-pin PLD (exemplified here by the Intel 5C060) is very costly. The data inputs must be latched inside a macrocell; using not only the macrocell but also the pin. The divide down toggle flip-flops cannot be buried, resulting in the loss of a pin for each flip-flop. The net utilization for the 5C060 implementation is 12 of 16 macrocells and 16 of 24 pins, virtually all of the device.

Implementing the same design in the Intel 5AC312 uses a much smaller amount of space.

By using the input latches available on the 5AC312, the select data inputs can be stored immediately at the input pin rather than inside a macrocell. This saves a macrocell, saves a pin, and decreases the delay time. Second, since the 5AC312 has separate register and pin feedbacks on each macrocell, the baud rate divider can be buried by using the register feedback paths while the input feedback paths remain available for use as standard inputs. Inside the 5AC312, the circuit consumes 8 of 12 macrocells, and 7 of 24 pins, a significant I/O pin savings.

In fact, the I/O pin savings is so significant that the accompanying address decode circuitry -- which would be implemented typically in a 20L8 or second PLD -- can be added to the 5AC312 (Figure 11). The 14 address inputs (A13 - A0), along with the memory or I/O status signal (M/I0) are fed into the PLD to generate the baud rate select data clock signal (BAUDSEL) and the 8251 Command/Data (C/D) and Chip Select (/CS) signals. The net result is a 10 of 12 macrocell, 24 of 24 pin, single-chip solution.



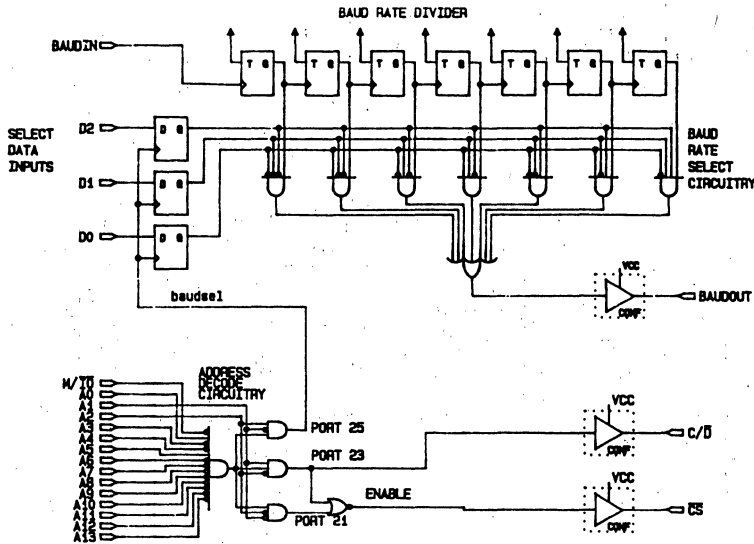


Figure 11: Programmable Baud Rate Generator - 5AC312 Implementation

CONCLUSION

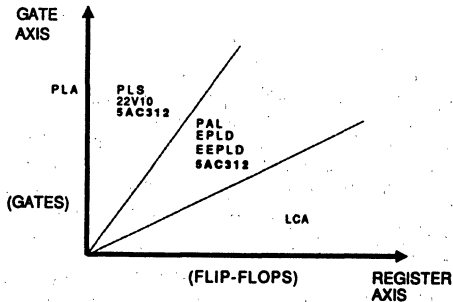


Figure 12: 5AC312 Application Areas

Based around a novel programmable AND/allocatable OR array structure, the Intel 5AC312 is uniquely suited to cover both highly combinational and complex sequential designs (Figure 12). The 5AC312 is made combinational powerful through a 0 - 16 product term allocation arrangement and sequentially powerful through separate register and pin feedback and other features. The 5AC312's latched input capability is an asset in both combinational and sequential applications. The net result is a combinational powerful, sequentially powerful 24-pin device.

Acknowledgements

Special thanks to David Poisner of the Intel Datacomm Focus Group in Folsom, CA and Prof. J. Michael Dunlap and Robert A. Miller of Willamette University in Salem, OR for the use of their designs in this paper.

References

1. An Engineering Approach to Digital Design, William I. Fletcher, Prentice-Hall Inc., 1980, pp. 276, 280, 281. (Provided as an example).
2. PAL is a registered trademark of Monolithic Memories Inc.
3. Product Term Allocation is an Intel Patent Pending.
4. Personal System/2 and Micro Channel are trademarks of International Business Machines Corp.
5. Ethernet is a trademark of Xerox Corp.
6. ESPRESSO is a copyright of the University of California at Berkeley.
7. IBM PC/AT is a registered trademark of International Business Machines Corp.

ADVANCED ARCHITECTURE PLDs SOLVE COMMON  
STATE MACHINE PROBLEMS

Liliyas S. Koumis  
Technical Marketing Engineer  
Intel  
1900 Prairie City Road  
Folsom, CA 95630

INTRODUCTION

The introduction of programmable logic devices (PLD) was a true revolution in the hardware design world. It enabled engineers to shrink circuits requiring several devices onto a single device thus simplifying their designs while saving space and power. Traditionally, PLDs have been used in combinational circuits such as address decoders as well as sequential circuits such as bus arbitration schemes. During the last few years, advances and improvements in PLD architectures enabled the devices to grow more complex while addressing the never-ending quest for higher density and faster speeds. Despite these improvements, engineers still face certain problems and limitations when implementing state machine designs with PLDs. The Intel 5AC312 and 5AC324, multipurpose generic erasable PLDs, offer a solution to these problems that gives engineers a better device to implement their designs.

A typical programmable logic device is composed of a user-programmable AND array, a fixed OR gate, followed by an output register which includes a feedback path from the output to the programmable AND array. Combination of these elements is commonly referred to as a 'macrocell.' The existence of a feedback path from the output registers to the AND array makes PLDs ideal candidates for state machine implementations.

TYPES OF STATE MACHINES POSSIBLE IN PLDs

The three basic categories of state machines are Class A, Class B and Class C, better known as MEALY, MOORE TYPE A and MOORE TYPE B respectively. It is possible to implement any of the classes of state machines in a PLD, however the efficiencies vary with state machine class. The main characteristic of the Class A machine is that its outputs to the external world are a

function of both the input and the present state of the machine as shown in Figure 1. Mathematically, this can be expressed as:

$$z^{n+1} = f(x^n, y^n)$$

Class A Machine  
(Mealy Machine)

where 'z' is the decoded output. 'x' is the input. 'y' is output of the next state decoder and 'n' is the present state.

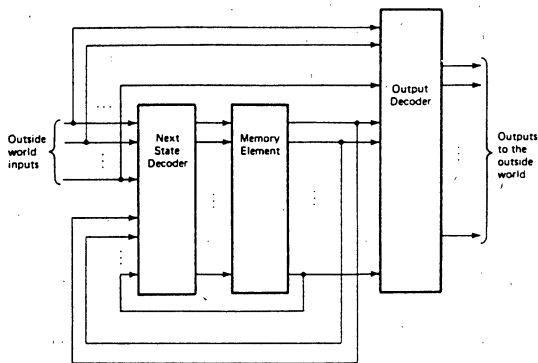


Figure 1: Class A State Machine

The Class B machine differs from Class A in that its output is only dependent on the present state of the machine through output decoding, or better expressed as:

$$z^{n+1} = f(y^n)$$

Class B Machine  
(Moore Type A Machine)

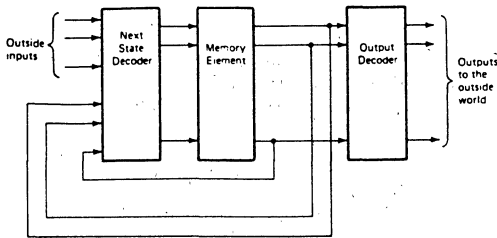


Figure 2: Class B State Machine

Finally, the Class C machine is essentially the same as the Class B but requires no output decoding; the outputs are the next state of the machine:

$$z^{n+1} = y^{n+1}$$

Class C Machine  
(Moore type B Machine)

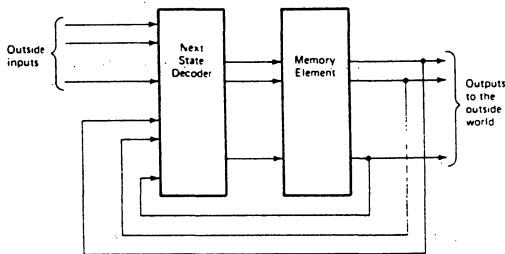


Figure 3: Class C Machine

If a Class A or a Class B machine is implemented in a standard PLD, two macrocells would be required per state; one macrocell has to be used for the input decoding and the other for the output decoding. The use of an extra macrocell for output decoding is not an efficient use of the device resources. For example, the largest state machine that can be implemented in an eight-register PLD is one with only four states variables. On the other hand, since a Class C machine does not perform output decoding, only one macrocell per state is required. As a result, that a machine with eight state variables could easily fit in the same eight-register PLD.

#### PROBLEMS WITH STATE MACHINE DESIGNS IN PLDs

Despite the architectural advantages of implementing a Class C machine, traditional PLDs still inherit serious timing problems and structural shortcomings.

The first problem is violation of setup and hold times of the output registers. This is encountered commonly in environments where the inputs are asynchronously changing with the device clock. Sources for these problems may be different data paths for each input or origination of these input signals from circuits that use a different clock than that used for the output registers. This kind of problem causes the output to glitch and may cause the machine to enter an invalid or incorrect state. The problem traditionally has been solved by adding external metastable-hardened registers to synchronize the inputs with the outputs. This solution, however, has obvious drawbacks such as an increase in chip count, additional time delays, and an increase in power consumption. In some cases, an eight-register IC is added even when only a few inputs must be synchronized.

The second most common problem is missed input pulses of short duration. In modern and complex circuits, short pulses may arrive at the inputs and disappear at a faster rate than the PLD clock. This causes these inputs to be missed by the PLD, which in turn causes erroneous operation of the state machine. Traditional PLDs offer no solution to this problem. Engineers have had to resort to adding circuitry to ensure that the inputs are present long enough to be seen by the PLD clock. This additional circuitry further complicates designs, adds

delays, increases power consumption and adds an unnecessary burden to the engineers.

Finally, the most frustrating problem for engineers is that the number of product terms available per macrocell is fixed. Engineers usually assign the states to the output pins randomly, write the equations and allow the software to determine whether the equations fit their chosen device. If the number of product terms required to implement the given equations is greater than the fixed number of available product terms, the designer devotes additional time and resorts to more 'innovative' approaches, such as breaking down the state machine into smaller parts to fit the device. This not only introduces additional time delays and reduces the effective use of the device, but it also increases development time and board cost.

### SOLUTIONS TO THESE COMMON PROBLEMS

The above discussion illustrates the need for a more sophisticated generation of Programmable Logic Devices that address these problems. Two new Intel PLDs, the 5AC312 and 5AC324, are specifically targetted at fulfilling the requirements of better set-up and hold timing, faster input clock rate and flexible product terms. The AC in the part name stands for "Advanced CMOS", the 3 stands for third generation and 12/24 is the number of macrocells in each device. This family of Intel Erasable Programmable Logic Devices uses advanced CHMOS\* EPROM cells instead of polysilicon fuses as a logic control elements. This process enhances the testability and reliability of the devices while significantly reducing power consumption. For the remaining portions of this paper, the 5AC312 will be referred for ease of reference, but for all practical purposes, the 5AC324 is functionally identical to the 5AC312 but contains twice the number of macrocells and ten register/latched inputs instead of eight.

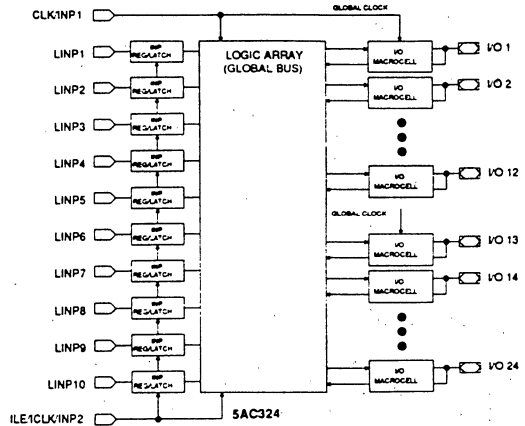
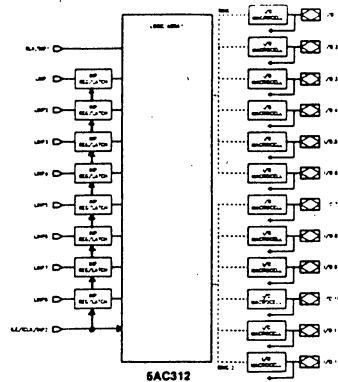


Figure 4: Architecturally Advanced 5AC312 and 5AC324 Global Block Diagrams

As it can be seen in Figure 4, the 5AC312 has the architectural features of a Class C machine but also offers additional features address the issues discussed earlier. The input structure of the 5AC312 offers several programmable options, each addressing a particular need or problem.

To address the first problem-violation of setup and hold times of the output registers-the 5AC312 offers an additional register/latch input with a programmable clock. The clock can be the same as the output register clock shifted by 180°, a separate high frequency clock, or be generated by a product term from the logic array.

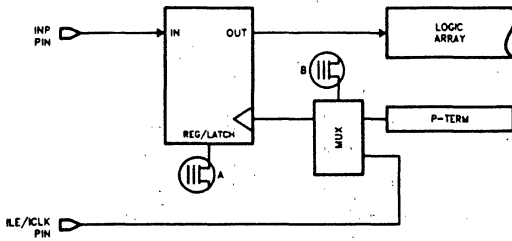


Figure 5: 5AC312/5AC324 Input Structure

By using the first clock option, synchronization is achieved, and thus the risk of output glitches is minimized. By cascading the input and output registers and shifting the input clock 180° from the output register clock, an additional advantage is gained by allowing enough time to satisfy the setup and hold time requirements of the output registers. Metastability characteristics of the device is of particular concern and are discussed later in this paper.

The second option, a separate high frequency clock, enables the device to sample inputs of very short time duration. This clock operates up to 50MHz, with an input register setup of only 5ns. If the latch feature is selected, the setup time is reduced to 0ns. Of course, a mode can be selected where the input data flows through, bypassing the register/latch combination. The third clock option, clocking the input registers with the output of a product term from the logic array, is ideal for applications where registers are to be clocked only when a certain input condition is met.

To address the fixed product term problem, the 5AC312 implements an innovative solution called 'product term allocation.' In each

individual macrocell, the eight product terms are sub-divided into two groups of four. The product term allocation is achieved by allowing each of these four product term groups to be borrowed from or lent to adjacent macrocells. By allocating product terms between adjacent macrocells, any register can be driven by as many as 16 product terms by borrowing unused product terms from its neighbors. Conversely, as little as zero product terms can be used if a macrocell lends to both of its neighbors. The 12 macrocells in the device have been divided into two groups of six each (or two groups of 12 for the 5AC324) called the "rings" that help define adjacent macrocells for borrowing and lending purposes.

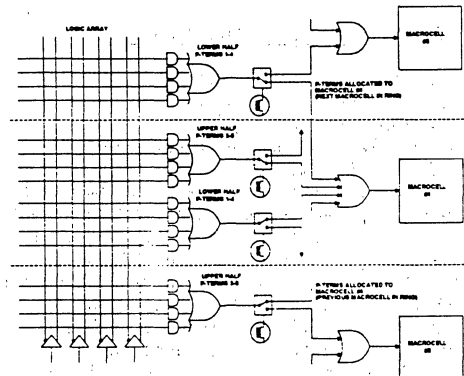


Figure 6: Product Term Allocation

The efficiency of this configuration can be demonstrated with a simple example. Assume an excitation function needs four product terms and another function needs 10 product terms. Implementing these functions with a fixed eight-product-term PLD requires one macrocell to implement the four-product-term function, and two macrocells to implement the 10 product terms function. To fit the 10 product term function, the equation needs to be broken into two parts, thereby increasing the delay. Therefore, out of 24 available product terms (three groups of

eight), 14 are used (14/24 = 58% efficiency). Using the 5AC312 to implement the same functions yields the following: the four-product-term function is implemented with half of a macrocell, allowing the other half to be allocated to the adjacent macrocell for implementing the 10-product-term function. No design-splitting is required. Therefore, out of 16 product terms, 14 are used. This translates to 88% product term utilization. The product term allocation is completely transparent to the user since it is achieved through software. When the compiler determines that an additional number of product terms is required, it automatically allocates resources to fit the required excitation function.

### METASTABILITY CHARACTERISTICS

Although metastability is a relatively rare event, ignoring it can cause serious timing problems. The input registers found in the 5AC312 offer excellent recovery time where metastability is of concern. Metastability can be simply described as the inability of a register to decide the state of its output within a fixed amount of time. This event usually occurs when synchronizing an external event with a periodic clock. If a flip flop is clocked nearly at the same time as changing data, there is a small window of time where the output of the register is unknown. This window of time is the recovery time ( $t_{re}$ ) of the flip-flop and is typically in the order of nano-seconds. Designers at Intel have performed tests to obtain the recovery time for the 5AC3xx family of devices and have concluded that the Intel devices have better recovery times than familiar TTL devices such as 74F74. Table 1 shows sample data taken at a clock frequency of 2MHz and data frequency of 1MHz.

DEVICE	Recovery Time (ns)
7474	1.6
74LS74	1.5
74S374	0.91
74F373	0.70
74F74	0.40
5AC312/5AC324	0.35

Table 1

Additional information on the procedure used to obtain this data and its use can be found in references one and three.

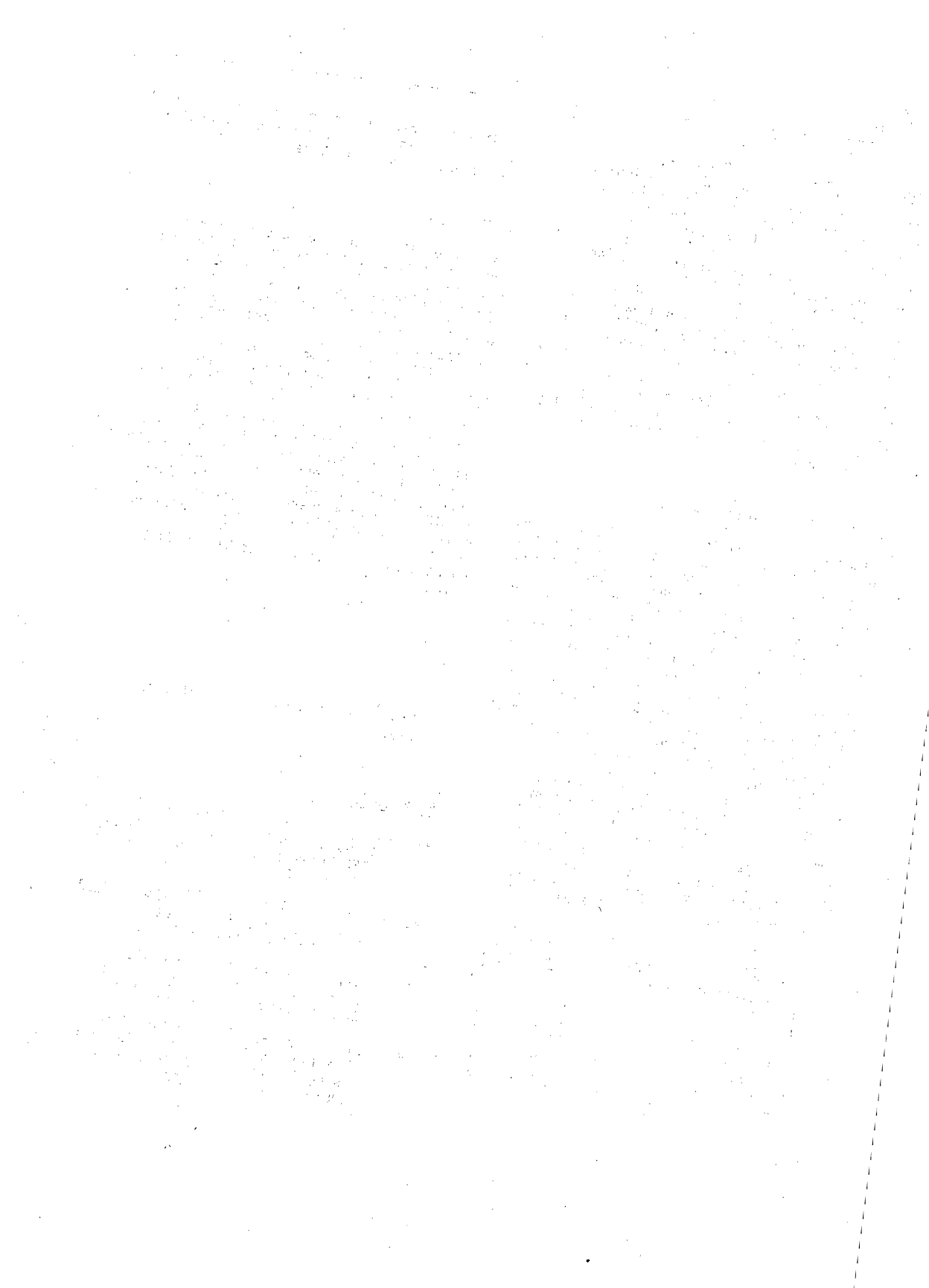
### CONCLUSION

Because of their internal architectural characteristics Programmable Logic Devices have become the ideal method to implement state machine designs. This is evident by the wide variety of applications where programmable logic devices are found today. The latest generation of PLDs, with the advantages of programmable I/O pins and expanded number affixed product terms, are certain to replace traditional off-the-shelf logic as designers discover their usefulness in modern applications. However, to overcome the problems associated with setup and hold time violations, missed inputs and fixed number of product terms, a new generation of PLDs was needed. The Intel 5AC312 and 5AC324 overcome these problems by providing selectable input register/latch option with excellent metastability characteristics and allocatable product terms.

\* CMOS is a patented process of Intel Corporation.

### REFERENCES:

1. Chaney, Thomas J., "Measured Flip-Flop Responses to Marginal Triggering," IEEE Transaction on Computers, vol. C-32 No. 12, December 1983.
2. Fletcher, William I., "An Engineering Approach to Digital Design", Prentice-Hall Inc., 1980.
3. Stoll, Peter A., "How to Avoid Synchronization Problems," VLSI Design, November/December 1982.
4. Weigl, Karl H., "Eliminating Common Problems in State Machine Designs Using Innovative PLD Architectures," SOUTHCON, December 1987.





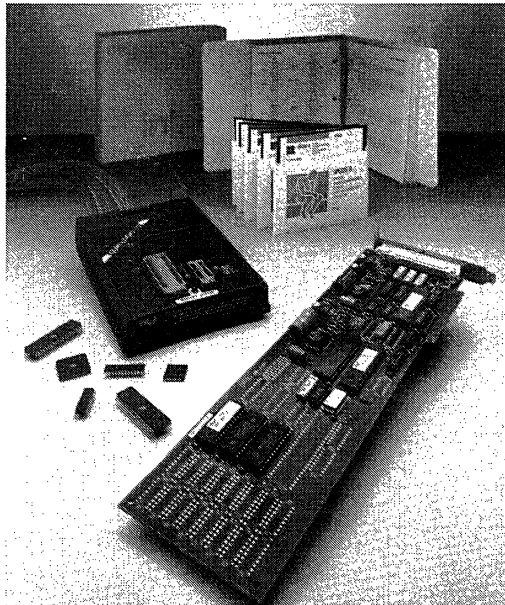




## iPLDS II THE INTEL PROGRAMMABLE LOGIC DEVELOPMENT SYSTEM VERSION II

- **Hardware and Software Necessary to Turn Design Concepts into Functional Erasable Programmable Logic Devices (EPLDs)**
- **Menu-driven Software with On-line Help Messages for All Stages of the Design Process**
- **iUP-PC Hardware Programs Intel EPLD's, EPROM's, E<sup>2</sup>PROM's, Peripherals, and Microcontrollers with one PC-based System**
- **All Equipment Interfaces with the IBM PC/XT\*, PC/AT\*, and True Compatibles**
- **JEDEC Standard Design File, Part Utilization Report, Minimized Equation File, and Compiler Error File All Available as Outputs**
- **Supports a Variety of Input Methods:**
  - Schematic Entry
  - TTL Library
  - EPLD Primitives Library
- **Text Editor Entry**
  - State Machine
  - Boolean Equations
- **Macro Expander Accepts TTL, and User-Defined Macros and Expands Them into Equivalent EPLD Primitives**
- **Espresso\*\* Minimizer Reduces Logic Equations to Least Number of Product Terms**
- **Supports All Intel EPLD's Including the 5AC312, 5AC324, and 85C508**

Release 2.0 of Intel's Programmable Logic Development System II (iPLDS II) is a powerful set of tools for transforming a logic design into customized silicon. The system provides design entry, logic compilation, and device programming capability on a desktop using an IBM PC/XT, PC/AT, or compatible.



**iPLDS II Components Picture**

290134-1

\*IBM PC/XT, PC/AT are registered trademarks of International Business Machines Corporation.

\*\*ESPRESSO is a copyrighted by the University of California at Berkeley and is used with permission.

## INTRODUCTION TO PROGRAMMABLE LOGIC DESIGN

When performing a programmable logic design on a CAD system, the design must first be entered using one of a variety of entry methods. These methods typically include schematic capture or Boolean equation entry using a standard text editor. Less typical entry methods include netlist entry, whereby a hand drawn schematic can be entered in a node-by-node fashion, or state machine entry in a text or graphical mode.

Once the design has been entered into the CAD package, several processing steps may occur. The design is usually translated into a format usable by the software, logic reduction may be performed, and, finally, some form of programming file can be produced. Most CAD packages also produce documentation of the minimization and device fitting results, including the final pin assignments.

Once the programming file has been generated, the design can be transferred into silicon in a programming manner similar to that used for EPROMs.

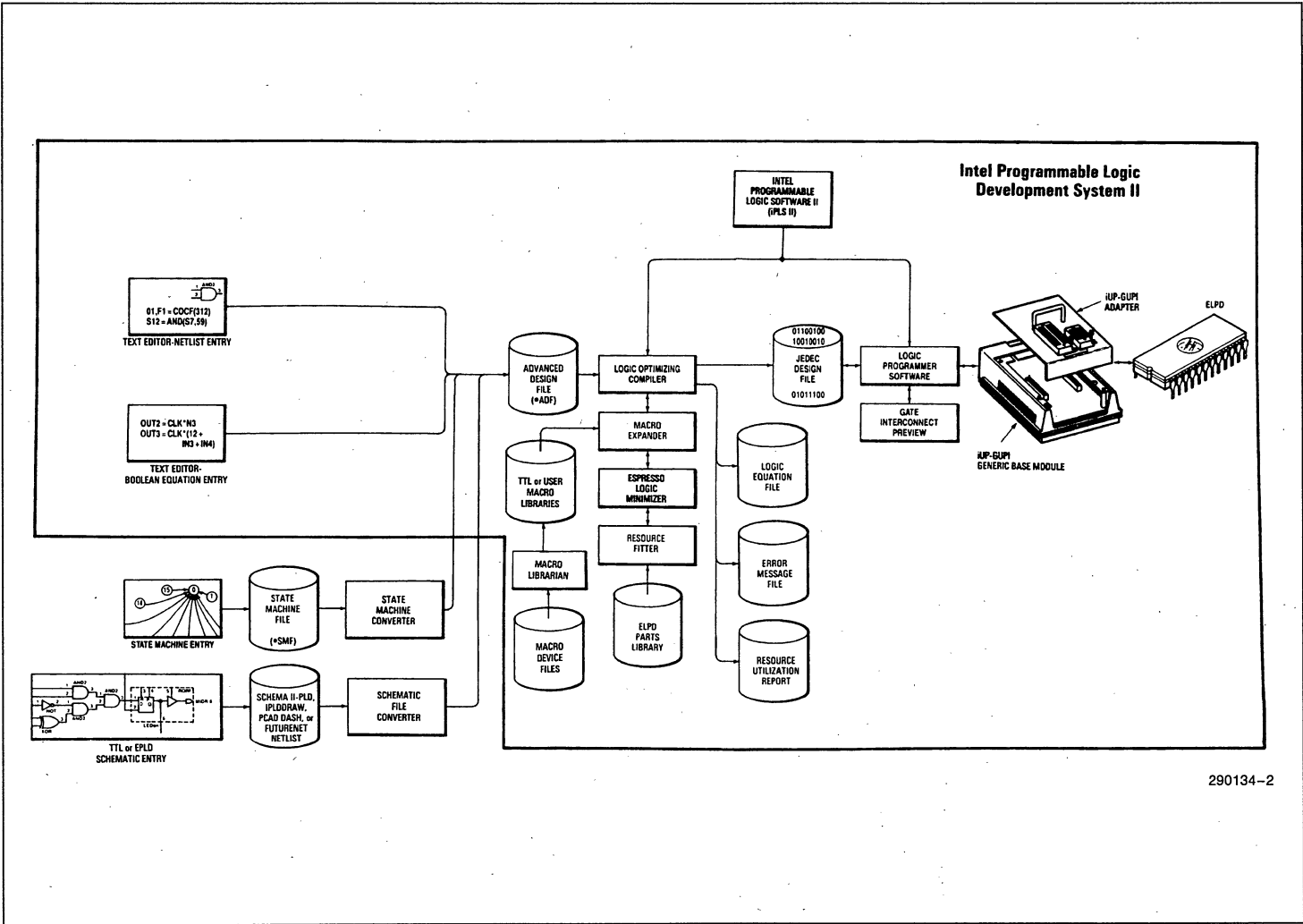
## FUNCTIONAL DESCRIPTION OF IPLDS II

All of the design entry methods with the exception of graphic state machine entry are supported by the iPLDS II software. iPLDS II supports netlist and Boolean equation entry using any standard text editor. State machine software and schematic capture libraries are also available from Intel as optional entry methods. Depending on the entry format used, the design may require translation into Advanced Design File (ADF) format. Once the design is in ADF form, the Logic Optimizing Compiler expands any macros, minimizes all equations, and fits the design into a device-specific JEDEC Design File. The JEDEC Design File is programmed into the EPLD by the Logic Programmer Software using the iUP-PC hardware. Thus, the circuit design is transformed into an operating EPLD on one workstation.

The Intel Programmable Logic Software II (iPLS II) is composed of four functional modules: design entry, netlist conversion, file compilation and device programming.

### Design Entry

Design entry is typically accomplished by creating an ADF using an ASCII text editor, or by using a schematic capture package.



290134-2

## Netlist Conversion

If schematic capture of state machine entry is used, the design must be converted into an ADF format. The optional SCHEMA II-PLD schematic capture package is a low-cost way to enter schematic designs. SCHEMA II-PLD supports EPLD primitives and TTL or user-defined macro symbols. It also outputs directly in ADF format. SCHEMA II-PLD contains the EPLD Manager, which provides a single user interface to both SCHEMA II-PLD and iPLS II software.

The P-CAD†† and Futurenet† systems may be used to capture EPLD symbols provided the EPLD libraries and ADF converters are used. State machine entry may be performed via the iSTATE software and a standard text editor.

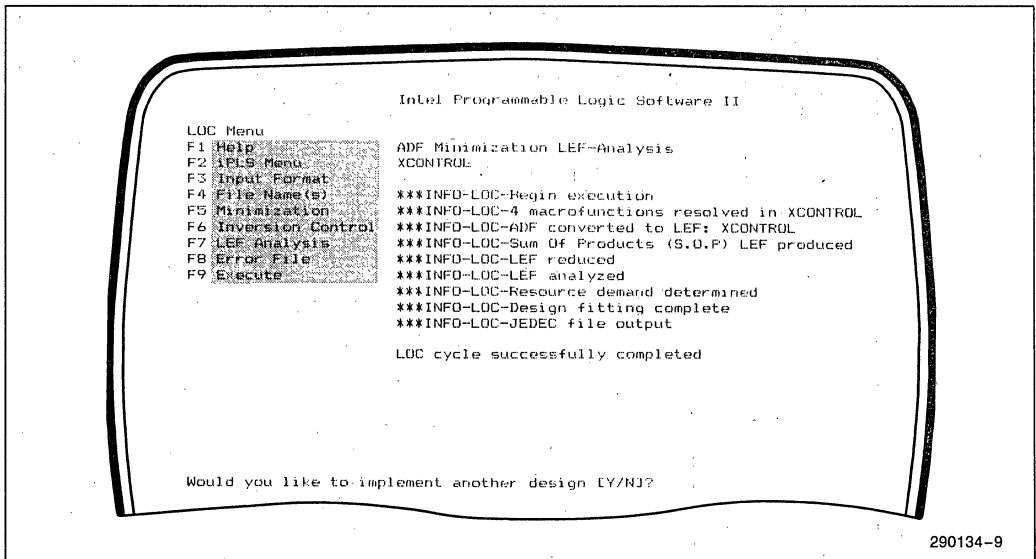
†Futurenet is a registered trademark of FutureNet Corporation.

††P-CAD is a registered trademark of P-CAD Corporation.

## File Compilation

File compilation is performed by the LOGIC OPTIMIZING COMPILER. The LOC accepts an ADF and converts it into an industry standard JEDEC file which is used to program the device. As a part of this process, the LOC expands TTL macros into equivalent EPLD logic, minimizes the logic equations using the Espresso algorithm, and maps the network and logic equations into a cell map for the selected device. The final output of the LOC is a JEDEC Design File. The JEDEC Design File describes the input design for the designated EPLD in JEDEC standard format.

For designs using the 5AC312 or 5AC324 iPLS II R2.0 utilizes proprietary algorithms to efficiently use the device resources. The improved Fitter in R2.0 optimizes fitting for all devices.



290134-9

### Logic Optimizing Compiler Main Menu

## Device Programming

The programming hardware is controlled by the LOGIC PROGRAMMER SOFTWARE. LPS takes the JEDEC file produced by the LOC and programs it into the device. LPS can also read a programmed device or verify that a device has been programmed correctly.

The Intel Universal Programmer for the Personal Computer (iUP-PC) is a versatile programming solution in a PC-based system. Installed in an IBM PC/XT, PC/AT or compatible host, the iUP-PC emulates the performance of the standalone INTEL iUP-200A Universal Programmers. As such, it supports the iUP Generic Universal Programmer Interface (iUP-GUPI). With the appropriate socket adapters for the iUP-GUPI, the iUP-PC supports all Intel EPLDs. Future EPLDs will be supported by new GUPI adapters or adapter upgrades. Many other Intel devices—EPROMs, EEPROMs, and microcontrollers—are also supported by the GUPI. The iUP-PC is controlled by the LPS or the iPPS (Intel PROM Programmer Software). iPLDS II includes the iUP-PC, which contains the iPPS, PCPP programming card, interconnect cable, and the GUPI base. GUPI adapters are available separately.

## iPLS II SOFTWARE

The Intel Programmable Logic Software II (iPLS II) has many options and enhancements for implementing a logic design. iPLS II accommodates a wide variety of design input methods. Schematics, state machines or Boolean equations may all be used provided the proper formats and convertors are implemented as needed. No matter what method is chosen, the Logic Optimizing Compiler will minimize and fit the design during compilation. Finally, iPLS II contains the Logic Programmer Software which controls the iUP-PC programming hardware for all Intel EPLDs.

### I. Design Input

The entire spectrum of design input methods is available to the logic designer in iPLS II. Everything from TTL schematics to Boolean equations are accepted and processed by the LOC.

### A. TTL SCHEMATIC ENTRY

SCHEMA II-PLD is an optional software package that allows EPLD design to be implemented with standard TTL functions. SCHEMA II-PLD contains a symbol library that includes common SSI/MSI TTL symbols. SCHEMA II-PLD also outputs directly in ADF format. The TTL symbols appear in the ADF in the form of macro calls. During compilation, iPLS II automatically expands these calls from its TTL macro library. Thus, with SCHEMA II-PLD, conversion to EPLD logic primitives is performed automatically in a manner completely transparent to the user.

Only parts supported by the SCHEMA II-PLD TTL symbol library and the iPLS II TTL macro definition library may be used for TTL schematic entry. In most cases, this won't be a limitation as the most common parts are included in both libraries. Parts not in the macro libraries may be created by the user and stored in proprietary user libraries. SCHEMA II-PLD also supports creating of user-defined macro symbols. The iPLS II Macro Librarian supports creation of iPLS II macro libraries.

### B. SCHEMATIC ENTRY WITH INTEL SYMBOL LIBRARY

If the user prefers designing with EPLD logic primitives but still wants to use schematic entry, SCHEMA II-PLD, in addition to supporting TTL schematic capture, also supports design using EPLD primitive symbols. Users can enter their design and have both a schematic drawing and an ADF version of the design. The logic symbols are loaded from the Intel library and connected in the usual manner. For quicker use of EPLD primitives, a second library, EPLDMAC.LIB is available for use. Optional symbol libraries are also available for PC-CAPS\* by P-CAD Corporation and DASH-2, -3, -4\*\* by FutureNet (iSLIBPCAD, iSLIBFNET). The iSIMLIB optional library is available for simulating logic designs with P-CAD's PC-LOGS logic simulator.

\*PC-CAPS and PC-LOGS are registered trademarks of P-CAD Corporation.

\*\*DASH-2, -3, -4 are registered trademarks of FutureNet Corporation.

## C. TEXT EDITOR ENTRY

Designers who are familiar with the logic primitives and the Advanced Design File format can directly enter ADFs with a standard text editor. The bulk of the design entry can be accomplished using Boolean Equations obtained from a Karnaugh map or truth table. Hence, the need for conversion to gates is eliminated. This method of entry is useful for sub-circuits that will be incorporated into larger designs.

## D. STATE MACHINE ENTRY

In the past, state diagrams or flowcharts (ASM charts) were merely abstractions used to obtain the logic equations necessary to implement TTL designs. With the advent of the iPLS II state machine convertor (iSTATE), this is no longer the case. Using an IF THEN / ELSE format, the designer may enter the state machine description without having to extract the logic and convert the equations into TTL components. The state machine to Boolean logic conversion is handled by the state machine convertor, provided the input file adheres to the specified State Machine File (SMF) format.

### Summary of Optional Entry Requirements:

#### TTL Schematic Capture

1. TTL Macro Library
2. EPLD Custom Macro Library
3. SCHEMA II-PLD

#### PC-CAPS

1. Intel Library used to design logic circuit
2. Component List Output
3. PCAD convertor used in LOC  
(Library and convertor contained in iSLIBPCAD)

#### DASH-2, -3, -4

1. Intel Library used to design logic circuit
2. Pin List Output
3. FutureNet convertor used in LOC  
(Library and convertor contained in iSLIBFNET)

#### State Machines

1. State Machine File (SMF) format used
2. Optional state machine convertor used in LOC  
(Convertor contained in iSTATE)

## II. Logic File Compilation

Before programming the part, the designer must compile the input design file into a JEDEC standard file. This function is performed by the Logic Optimizing Compiler.

### LOGIC OPTIMIZING COMPILER (LOC)

Once the input file is in Advanced Design File (ADF) format, the LOC will compile it into a device-specific JEDEC Design File. The first phase of this compilation is performed by the MACRO EXPANDER. The Macro Expander expands Intel or TTL macros into equivalent EPLD equations. The second phase is performed by the ESPRESSO MINIMIZER. The minimizer reduces all the logic equations to their simplest form using the ESPRESSO II-MV algorithm. The final phase of compilation is performed by the FITTER. The Fitter creates a cell map of the minimized equations according to the resources available within the specified device.

### MACRO EXPANDER

The input design file is initially passed through the MACRO EXPANDER. The Macro Expander searches the file for any non-EPLD network elements. If found, the Expander then searches the User Libraries and TTL Library for the unidentified element. Once the element is located, the design file element is replaced by the equivalent EPLD primitive implementation found in the library. Having the Expander search the User Libraries allows the user to create his own macros. User macro files are created with a standard ASCII text editor and are stored in libraries by the iPLS II Macro Librarian.

### ESPRESSO MINIMIZER

The minimization in the LOC is performed by the ESPRESSO II-MV MINIMIZER. Developed by the University of California at Berkeley, the ESPRESSO II-MV algorithm is regarded by many as being the best minimization method available. ESPRESSO II-MV uses DeMorgan's and other logic theorems to reduce the equations to the least number of product terms possible. Since product terms are the key variable in the EPLD architecture, the ESPRESSO II-MV Minimizer provides the simplest equations possible. As a result, the success rate for fitting large designs is dramatically increased.

### FITTER

The FITTER examines the architecture of the specified device, then tries to map the minimized equations into the resources available. The Fitter automatically assigns pins unless pin assignments are

already specified in the design input file. The fitting sequence continues until a successful fit is accomplished or all possible implementations are exhausted. Release 2.0 of iPLS II includes a new, faster Fitter that supports PGA packages and the 5AC312, 5AC324, and 85C508. Also included in this new Fitter is the capability to allocate p-terms to adjacent macrocells for devices such as the 5AC312 and 5AC324 that support p-term allocation.

## OUTPUT FILES

### — JEDEC Design File

A properly designed circuit results in the desired file from the LOC—the JEDEC Design File. The JEDEC Design File is a device-tailored EPROM cell programming map expressed in JEDEC standard format.

### — Resource Utilization Report

The Resource Utilization Report gives an in-depth view of what was used inside the EPLD. Items such as device pinout, macrocell usage, and feedback arrangements are all listed. Unused resources are also listed to aid the user in adding logic or merging EPLD designs.

### — Logic Equation File

The LEF file lists the logic equations after they have passed through the minimizer. It is these equations that are actually implemented in the final design.

### — Compiler Error File

If a logic circuit is incorrectly designed, messages are produced by the LOC denoting the errors. To assist the redesign, these errors are placed into the Compiler Error File for later reference.

## FILE MERGING

Once a design is successfully implemented, the LOC can merge it with other designs by simultaneously running the two ADF's. In this manner, LSI circuits can be broken into manageable chunks that can be implemented and tested individually. After each portion is completed, the subcircuits can be merged into one ADF to implement the total design.

## III. Device Programming

After the design has been successfully entered, minimized and fitted, the designer programs his part using the JEDEC file produced by the LOC. Programming is accomplished by running the Logic Programmer Software.

## LOGIC PROGRAMMER SOFTWARE

To program a device with the LPS, the user enters the file name and device to be programmed. The LPS checks if the device is blank, programs the device, then verifies that the device was programmed correctly. As a part of the Intel EPLD Programming Algorithm, each programmed cell is checked. Adding the complete device check after programming gives double verification that the part has been successfully programmed.

It is also possible to read a pre-programmed device and program other devices with the program read. The JEDEC Editor in LPS provides a hierarchical view of the device from the pin level, to the macrocell level, to the product term level. At the product term level, individual EPROM cells may be set or reset to connect or disconnect the logic equation inputs.

If the user does not want an EPLD to be read, the Security bit may be set when running the LPS. The Security Bit prevents a device from being examined after it has been programmed. This function is useful for protecting confidential designs.

## IUP-PC HARDWARE

The Intel Universal Programmer for the Personal Computer consists of the PCPP programming card, 50-lead interconnect cable, GUPI base and GUPI adapter. Together they form a system for programming most PROM-type Intel devices directly from the PC host.

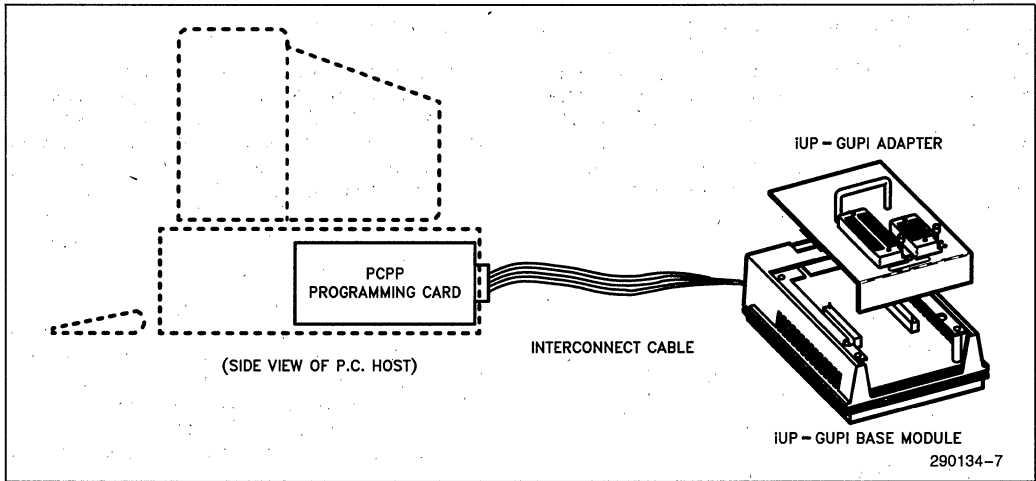
## PCPP

The Personal Computer Personal Programmer (PCPP) is the programmer interface card that fits into the IBM AT/XT or true compatible. It is capable of driving both the iUP-GUPI base and the iUP-FAST27K personality module. The PCPP emulates the performance of the Intel iUP-200A. The LPS or iPPS (Intel PROM Programmer Software) controls the PCPP, causing the programming card to generate the control signals for the GUPI base.

## GUPI BASE

The Generic Universal Programmer Interface (GUPI) is used for all programmable logic support. As all





**The Intel Universal Programmer for the Personal Computer (iUP-PC)**

signal generation to devices is done by the GUIPI, the programming waveforms are extremely reliable. Using the GUIPI also allows upgrading for future devices with the simple addition of a plug-in adaptor. Future Intel EPLDs will be supported by the GUIPI system.

vice description data for a family of similar devices. New devices will be supported by new adaptors or by upgrades to existing adaptors.

**GUIPI ADAPTERS**

Table 1 details the GUIPI adapters required for the logic devices. The adapters available for programming EPROM's, E<sup>2</sup>PROM's and microcontrollers can be found in the data sheet for the iUP-PC (Intel order number 290130). The adapters contain the de-

**SPECIFICATIONS**

**Host System**

The iPLDS II software requires an IBM PC/XT, PC/AT or other true compatible computer capable of running MS-DOS\* version 3.0 or later. The computer must have a 360KB double-sided, double-density diskdrive, a hard disk, and 512KB of RAM. Addi-

**Table 1. Intel Programmable Logic Development System II Programming Support**

Device	Number of Macrocells	iUP-GUPI Adapter	Package Type Supported
5C031, EP310	8	GUIPI LOGIC-12	20 Pin DIP
5C032, EP320	8	GUIPI LOGIC-12	20 Pin DIP
5C060, EP600	16	GUIPI LOGIC-IID	24 Pin DIP
5C090, EP900	24	GUIPI LOGIC-IID	40 Pin DIP
5C121, EP1200	28	GUIPI LOGIC-12	40 Pin DIP
5C180, EP1800	48	GUIPI LOGIC-18	68 Pin PLCC and JLCC
5C180PGA	48	GUIPI LOGIC-18PGA	68 Pin PGA
5CBIC	(inPLU):8 (# of Ports):5	GUIPI LOGIC-BIC	44 Pin PLCC
5AC312	12	GUIPI LOGIC-IID	24 Pin DIP
5AC324	24	GUIPI40D44J	40-Pin DIP
85C508	8	GUIPI85EPLD28	28-Pin DIP and PLCC

(EPXXX Devices from Altera Corp.)

tional memory is recommended (640K) and is required for the optional schematic capture programs. A color monitor is recommended, as the color graphics available provide a better representation of the data than a monochrome display. The PCPP programming card requires one full-size card slot in the host computer. iPLSII (Intel Programmable Logic Software) can run on the IBM PS/2.

\*MS-DOS is a trademark of Microsoft Corporation

## Operating Environment

## Electrical Characteristics

### PCPP: Worst Case Power Consumption at IBM PC I/O Channel

Supply Voltage	Voltage Variance	Personality Module	Max. Current Drain
+5V	+5%, -4%	FAST27K	1.898 A
-12V	+10%, -9%	FAST27K	102.9 mA
+12V	+5%, -4%	GUPI	530 mA

## Physical Characteristics

### PCPP:

Length: 13.3 inches (33.9 cm)

Height: 3.9 inches (10.0 cm)

### INTERCONNECT CABLE:

50 lead ribbon cable

Length: 3.0 feet (91.4 cm)

Width: 2.43 inches (5.5 cm)

## GUPI:

Length: 7.0 inches (17.8 cm)

Width: 5.5 inches (1.4 cm)

Height: 1.6 inches (4.1 cm)

## Environmental Characteristics

Operating Temperature: 10°C to 40°C

Operating Relative Humidity: 85% Maximum

## Equipment Supplied

### HARDWARE

- PCPP programming card
- Interconnect cable
- GUPI base
- (GUPI-LOGIC adaptors purchased separately)

### SOFTWARE

- iPLS II
- iPPS
- PLDUTIL

### DOCUMENTATION

- iPLS II User's Guide-V2.0 (order number 450196)
- PCPP User's Guide (order number 168161)

**ORDERING INFORMATION**

Order Code	Product Description
iPLDS II	Intel Programmable Logic Development System II: iPLS software, iUP-PC, iPLS II User's Guide
iPLS II	Intel Programmable Logic Software II: Logic Builder design entry, Logic Optimizing Compiler, Logic Programmer Software, iPLS II User's Guide
iUP-PC	Intel Universal Programmer for the Personal Computer: PCPP programming card, interconnect cable, iUP-GUPI base, Intel PROM Programming Software PCPP User's Guide
MLIB	iPLS II Macro Librarian: Macro Librarian Software and User's Guide Supplement for creating user-defined macro libraries.
iSTATE	Intel State Machine Software: Entry format documentation, state machine convertor for LOC
iSLIBFNET	Intel Symbol Library—FutureNet: EPLD symbol library for FutureNet DASH-2 schematic capture package, Futurenet Pinlist convertor for LOC
iSLIBPCAD	Intel Symbol Library—PCAD: EPLD symbol library for PCAD PC-CAPS schematic capture package, PCAD Component List convertor for LOC
iSIMLIB	Intel Simulation Library (PC-LOGS): EPLD simulation library for PC-LOGS simulator by PCAD

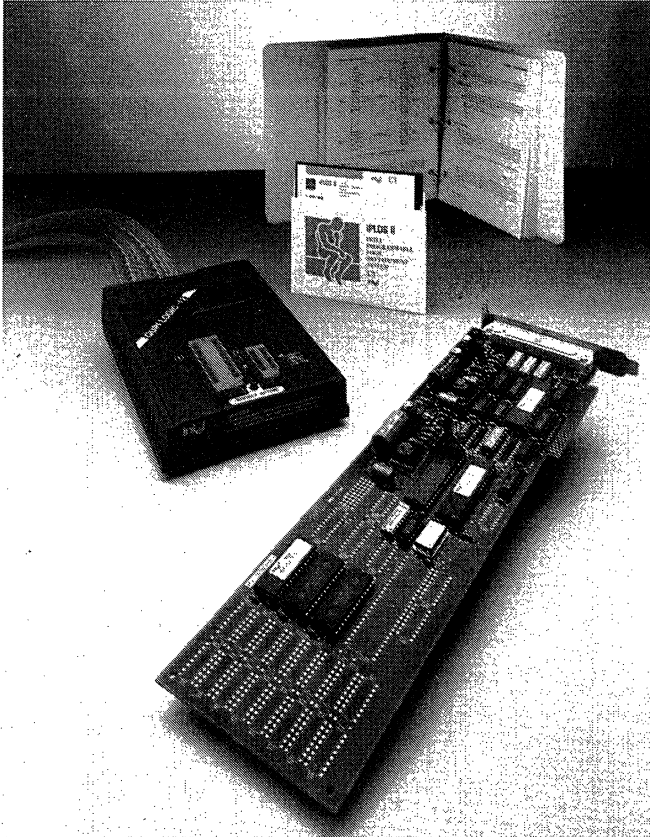
PLDUTIL	PLD Utilities: Functional Simulatory TTL Macro Library EPLD Custom Macro Library
iUP-GUPI	Intel Universal Programmer—Generic Universal Programmer Interface: Generic programmer base which holds GUPI adaptors
GUPI LOGIC-IID	GUPI Adaptor for the 5AC312, 5C060 and 5C090.
GUPI LOGIC-12	GUPI Adaptor for the 5C031, 5C032, 5C121 and future 20 DIP EPLDs
GUPI-LOGIC-18	GUPI Adaptor for the 5C180 and future 68 pin PLCC and JLCC EPLDs
GUPI LOGIC-18PGA	GUPI Adaptor for the 5C180 device in a 68 pin PGA package.
GUPI-LOGIC-BIC	GUPI Adaptor for the 5CBIC and follow-on products
GUPI40D44J	GUPI Adaptor for the 5AC324; includes 40-pin DIP and 44-pin JLCC sockets.
GUPI85EPLD28	GUPI Adaptor for the 85C508; includes 28-pin DIP and JLCC sockets.
ADAPT24TO28	Adapts 24 pin DIP socket to 28 pin PLCC socket; for use with GUPI LOGIC-09 and GUPI LOGIC-IID.
ADAPT40TO44	Adapts 40 pin DIP socket to 44 pin PLCC socket; for use with GUPI LOGIC-09 and GUPI LOGIC-IID.



## iUP-PC INTEL UNIVERSAL PROGRAMMER FOR THE PERSONAL COMPUTER

- Personal Computer Version of the iUP-200A/201A Universal Programmers
- Runs on an IBM PC/AT\*, PC/XT\* or True Compatible
- GUI and FAST27K Personality Modules Provide Support for Numerous Device Families
- Utilizes the intelligent™ and Quick-Pulse Programming™ Algorithms
- Easily Upgradable for new Devices Through Low-Cost Plug-In Adapters
- Extremely Versatile—Programs Intel or Intel-Compatible EPROM, E<sup>2</sup>PROMs, EPLDs, Peripherals and Micro-Controllers, Including the Latest Intel EPLDs

The Intel Universal Programmer for the Personal Computer, iUP-PC, provides a high performance programming solution from a PC host. Through plug-in adapters for the Generic Universal Programmer Interface (iUP-GUPI), the iUP-PC supports all Intel EPLDs and most other Intel programmable devices. Upgrades for new devices are made by the simple addition of a GUIP adapter or the upgrade of an existing adapter.



290130-1

**NOTE:**  
GUIP Adapter NOT included.

\*IBM PC/AT and PC/XT are registered trademarks of International Business Machines Corporation.

## FUNCTIONAL DESCRIPTION

The iUP-PC provides a fast, versatile and reliable programming solution from a Personal Computer host. Downloading to a stand-alone programmer or moving from one workstation to another is no longer required. With the iUP-PC, the designer may do his development and programming on one workstation. Through the Generic Universal Programmer Interface (iUP-GUPI), the iUP-PC is made extremely versatile. With the iUP-GUPI the designer may program across EPROM, E<sup>2</sup>PROM, Microcontroller, Peripheral and EPLD device categories with the mere change of a plug in adapter. No other hardware or software addition is needed. As all of the programming signals are generated at the GUPI base, extremely reliable waveforms reach the device.

## COMPONENTS

The iUP-PC programming system consists of five components:

**PCPP**—The Personal Computer Personal Programmer (PCPP) is an IBM PC/XT form factor expansion card which fits into an IBM PC/XT, PC/AT or true compatible.

**Interconnect Cable**—A 50 lead ribbon cable connects the PCPP to the iUP-GUPI.

**iUP-GUPI**—The Intel Universal Programmer—Generic Universal Programmer Interface (iUP-GUPI) is

the programming base which holds the device adapters.

**GUPI Adapters\***—The GUPI Adapters plug-in to the iUP-GUPI base. They carry the sockets and hardware for a particular device family.

**iPPS**—The Intel PROM Programmer Software (iPPS) runs on a personal computer under DOS and controls the PCPP/host communication.

**\*NOTE:**

Though the iUP-GUPI base is included in the iUP-PC package, the GUPI Adapters are NOT included. The desired adapters must be ordered separately.

## PCPP CARD

The PCPP is an 8085-based co-processor board. Communication between the host and the PCPP may be controlled by the iPPS or LPS (Logic Programmer Software). Version 2.3 or greater of iPPS is required for running the iUP-PC on a personal computer. LPS is the programming software included in Intel's Programmable Logic Software II (iPLS II).

The PCPP is capable of driving the iUP-GUPI and FAST27/K modules. Future Intel EPLDs will be supported by an iUP-GUPI adapter or adapter upgrade.

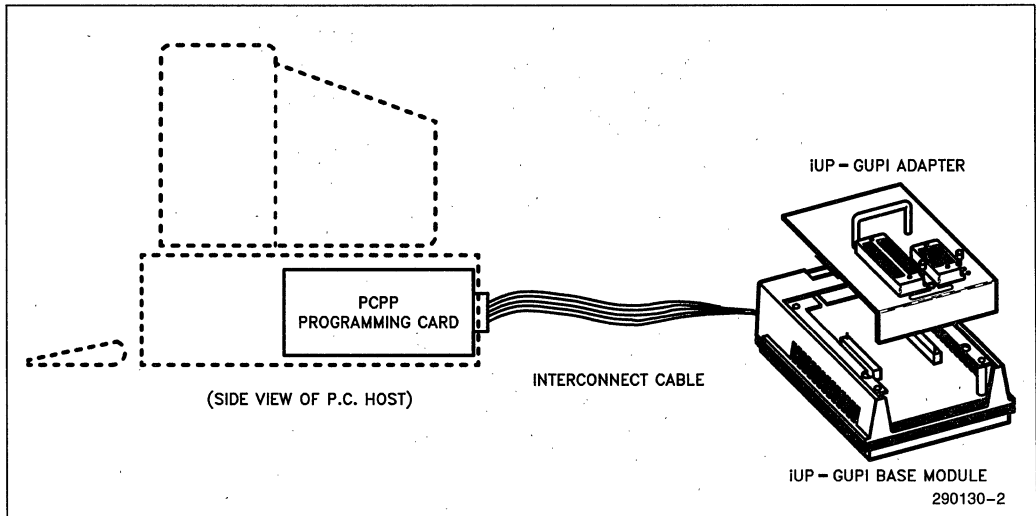


Figure 1. The Intel Universal Programmer for the Personal Computer (iUP-PC)

## iUP-GUPI MODULE

The iUP-GUPI is a generic base module that enables the iUP-PC system to accept low-cost plug-in adapters. These adapters configure the system to support a wide variety of programmable devices—EPROMs, microcontrollers, and EPLDs—as well as device package types.

The iUP-GUPI module connects to the PCPP card via a ribbon cable. An opening in the top of the iUP-GUPI provides easy plug-in installation of the GUIPI adapters (refer to Figure 2).

The iUP-GUPI offers the programming performance of earlier Intel personality modules, with the fastest Intel programming algorithms for each device type. For example, the iUP-GUPI uses the new Quick-Pulse Programming algorithm to program the 1-Meg EPROM in seconds.

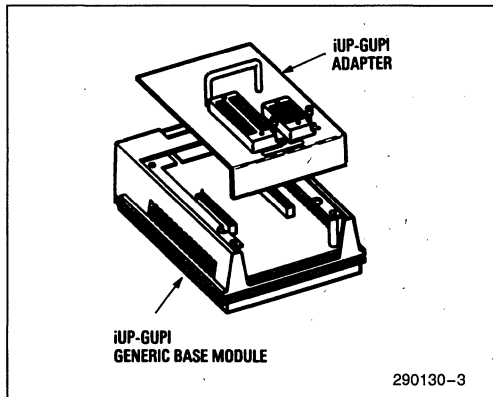


Figure 2. GUIPI Adapter Installation

## GUIPI ADAPTERS

The iUP-GUPI adapters provide the final link of the iUP-PC programming system. The adapters provide the proper sockets and characteristic information for families of Intel devices.

The iUP-GUPI LOGIC adapters complete the programming solution of the Intel Programmable Logic Development System II (iPLDS II). The GUIPI LOGIC adapters provide support for the entire range of Erasable Programmable Logic Devices (EPLDs). The adapters support families EPLDs with similar architecture, such as the 5C060 and 5C090. All future EPLDs will be supported by the GUIPI LOGIC adapter system.

Intel's one megabit EPROMs are also supported with GUIPI adapters. Adapters are available for the 27010, 27011, and 27210. The page mode of the 27011 is supported by the GUIPI 27011 adapter. Other Intel EPROM support is provided with the FAST27/K personality module.

Intel's first flash memory products are supported by the GUIPI FLASH Adapter.

The MCS-51 and MCS-96 microcontroller families are supported by the GUIPI MSC-51 and GUIPI 8796 adapters. Supplemental adapters provide support for the variety of microcontroller package types. The 8741 and 8742 peripheral components are supported by the GUIPI 8742 adapter.

Table 1 displays a cross-reference of the EPLD GUIPI adapters and the devices they support. Table 2 displays a cross-reference of the EPROM/Microcontroller adapters and the devices they support. Note that these tables are current at the time of printing. Contact your Intel sales representative for information on current support.

Table 1. EPLD GUIPI Module Adapters

Device Type	GUIPI Logic-IIID	GUIPI Logic-12	GUIPI 40D44J	GUIPI Logic-18	GUIPI Logic-18PGA	GUIPI 85EPLD28	GUIPI Logic-BIC
EPLD	5C060 5C090  5AC312	5C031 5C032  5C121	   5AC324	   5C180	   5C180G	   85C508	   5CBIC
Package Types	DIP*	DIP	DIP PLCC	PLCC CJ	PGA	DIP PLCC	PLCC

\*ADAPT units available to adapt DIP socket for PLCC package.

Table 2. EPROM/Microcontroller GUP1 Module Adapters

Device Type	GUPI 27010	GUPI 27011	GUPI 27210	GUPI FLASH	GUPI 8742	GUPI MCS-51	GUPI 8796	GUPI 8796LCC	GUPI 87C51GB	GUPI MCS-96LCC
EPROM	27010	27011	27210							
Flash				27F64 27F256 28F256						
Peripheral Microcontroller					8741AH 8742AH	8751H 87C51 8752BH 87C51FA 87C51FB	8794BH 8795BH 8796BH 8797BH	8796BH 8797BH	87C51GB	8797BH 87C196KB
<b>Package Types</b>	DIP	DIP	DIP	DIP	DIP	PLCC DIP	PGA DIP	LCC	PLCC	PLCC

### The iUP-Fast 27/K Personality Module

With the iUP-Fast 27/K personality module the user can program, read, and verify the contents of Intel's high density EPROMs, from the page-programmable (512K) 27513, to the CMOS 27C64, 27C256, and 87C64 EPROMs. This personality module supports the intelligent Programming algorithms and the intelligent Identifier™. The intelligent Identifier lets the personality module interrogate the PROM device in the program/master socket. It determines whether the type selected matches the type of PROM device installed and then selects the proper intelligent Programming algorithm. The intelligent Programming algorithms reduce programming time up to a factor of 10.

Low cost, plug-in upgrade kits allow addition of support for Intel's latest EPROMs. The first upgrade kit added support for the 27512 and innovative page-programmable 27513 plus the 27128A and 2817A. It has now been replaced by a second upgrade kit, iUP-Fast 27/K-U2 adding support for Intel's new CMOS EPROMs. (refer to Table 3).

As shown in Figure 3 the iUP-Fast 27/K personality module contains two 28-pin sockets, a hexadecimal display (0 through F), and a red LED that indicates when power is being applied to a socket. The program socket holds the device being programmed. The master socket will be used in future upgrades.

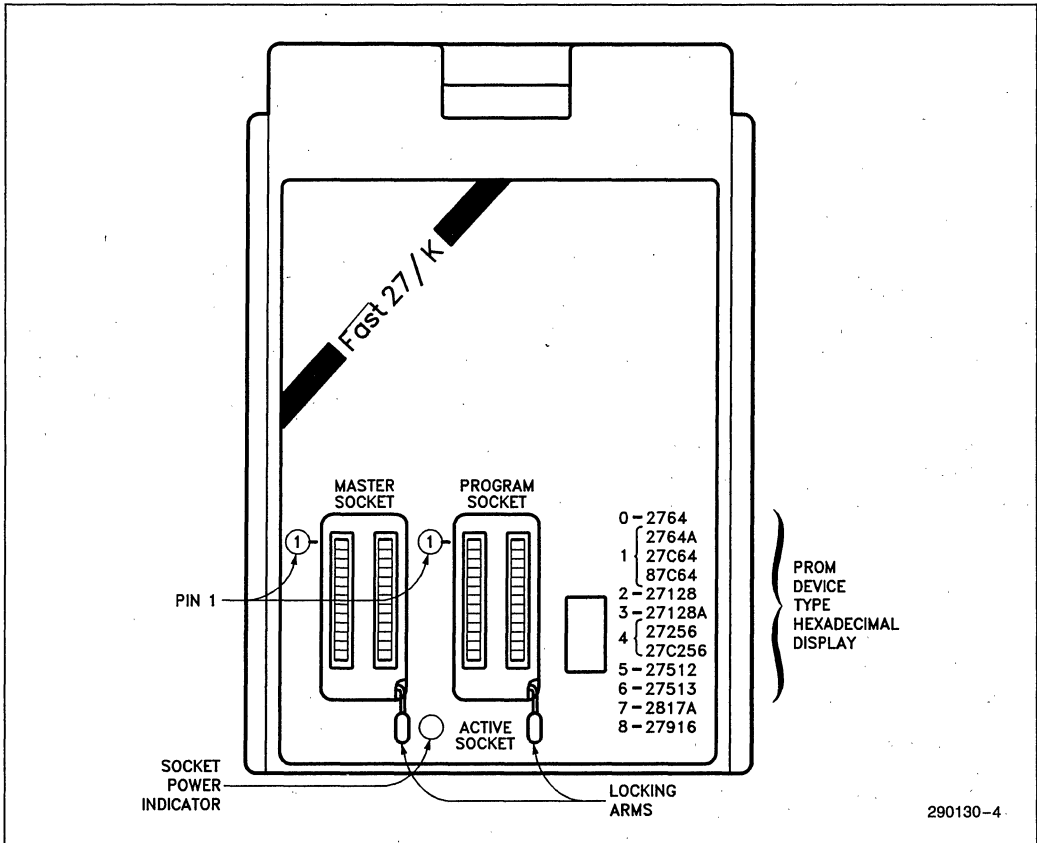


Figure 3. iUP-Fast 27/K Personality Module with U2 Upgrade



The hexadecimal display shows the PROM device type selected.

**Table 3. FAST27/K Module Device Support**

Prom Type	Fast 27/K Module	Fast 27/K U2 Kit	Fast 27/K-CON* Kit
EPROM	2764	2764	2764
	2764A	2764A	2764A
		27C64	27C64
		87C64	87C64
	27128	27128	27128
		27128A	27128A
	27256	27256	27256
		27C256	27C256
		27512	27512
		27513	27513
E <sup>2</sup> PROM		2817A	2817A

\*Uses Quick-Pulse Programming Algorithm.

**iPPS SOFTWARE**

The iPPS software, included with the iUP-PC brings increased flexibility to PROM programming. The iPPS software provides user control through an easy-to-use interactive interface and performs the following functions to make programming quick and easy:

- Reads PROMs, ROMs and EPLDs.
- Programs PROMs directly or from a file.
- Verifies PROM data with buffer data.
- Prints PROM buffer, or device file contents on the system printer.
- Performs interactive formatting operations such as interleaving, nibble swapping, bit reversal, and block moves.
- Programs multiple PROMs from the source file, prompting the user to insert new PROMs.
- Uses a buffer to change PROM contents.

With the iPPS software the user can load programs from system memory or directly from a disk file. Access to the disk lets the user create and manipulate data in a virtual buffer. This block of data can be formatted into different PROM word sizes for program storage into several different PROM types. In addition, a program stored in the target PROM, the system memory, or a system disk file can be interleaved with a second program and entered into a specific target PROM or PROMs.

The iPPS software supports data manipulation in the following Intel formats: 8080 hexadecimal ASCII, 8080 absolute object, 8086 hexadecimal ASCII, 8086 absolute object, 80286 absolute object, and 80386 bootloadable object. Addresses and data can be displayed in binary, octal, decimal, or hexadecimal. The user can easily change default data formats as well as number bases.

**iUP-PC SPECIFICATIONS**

**HOST SYSTEM**

The iPPS will run on an IBM PC/XT, PC/AT or other true compatible with a DOS operating system. The PCPP requires one full-sized card slot inside the PC.

**OPERATING ENVIRONMENT**

**Electrical Characteristics**

**PCPP:**  
Worst Case Power Consumption at IBM PC I/O Channel

Supply Voltage	Voltage Variance	Personality Module	Max. Current Drain
+ 5V	+ 5%, - 4%	FAST27K	1.898 A
- 12V	+ 10%, - 9%	FAST27K	102.9 mA
+ 12V	+ 5%, - 4%	GUPI	530 mA

**Physical Characteristics**

**PCPP:**

Length: 13.3 inches (33.9 cm)  
Height: 3.9 inches (10.0 cm)

**Interconnect Cable:**

50 lead ribbon cable  
Length: 3.0 feet (91.4 cm)  
Width: 2.43 inches (5.5 cm)

**iUP-GUPI:**

Length: 7.0 inches (17.8 cm)  
Width: 5.5 inches (1.4 cm)  
Height: 1.6 inches (4.1 cm)

**Environmental Characteristics**
**Environmental Class: B**
**Temperature:**

Operating	10 to 40 degrees C
Non-Operating	-40 to 70 degrees C

**Relative Humidity:**

Operating	85% Maximum
Non-Operating	95% Maximum

**DOCUMENTATION**

168161—PCPP User's Guide

166428—iUP-GUPI Module User's Guide

User's Guides for Adaptors, FAST 27/K Modules, and upgrades included with respective units.

**ORDERING INFORMATION**
**Order Code**
**Product Description**

iUPPC

Universal Programmer for the Personal Computer: PCPP Programming Card, 50-Lead Interconnect Cable, iUP-GUPI, iPPS, PCPP User's Guide

ADAPT24TO28	28-Pin PLCC Socket Adapter for GUPI LOGIC-IID
ADAPT40TO44	44-Pin PLCC Socket Adapter for GUPI LOGIC-IID
piUPGUPI	Generic Universal Programmer Interface (Base)
GUPI LOGICIID	GUPI Logic Adapter
GUPI40D44J	GUPI Logic Adapter
GUPI85EPLD28	GUPI Logic Adapter
GUPI LOGIC12	GUPI Logic Adapter
GUPI LOGIC18	GUPI Logic Adapter
GUPI LOGIC18PGA	GUPI Logic Adapter for 5C180 PGA
GUPI LOGICBIC	GUPI Logic Adapter
GUPI27010	iUP-GUPI EPROM Adapter
GUPI27011	iUP-GUPI EPROM Adapter
GUPI27210	iUP-GUPI EPROM Adapter
GUPI8742	iUP-GUPI Peripheral Adapter
GUPIMCS51	iUP-GUPI Microcontroller Adapter
GUPI87C51GB	iUP-GUPI Microcontroller Adapter
GUPI8796	iUP-GUPI Microcontroller Adapter
GUPI8796LCC	iUP-GUPI Microcontroller Adapter
piUPFAST 27K	EPROM Personality Module
iUPFAST 27KU2	FAST 27/K Upgrade Kit
iUPFAST 27KCON	Adds Quick-Pulse algorithm and device support
iUPFAST 27KIT	Combines piUPFAST 27K and iUPFAST 27KU2



## iUP-200A/iUP-201A UNIVERSAL PROM PROGRAMMERS

### MAJOR iUP-200A/iUP-201A FEATURES:

- Personality Module Plug-Ins Provide Programming Support for Intel and Intel-Compatible EPROMs, EPLDs, Microcontrollers, Flash Memories, and other Programmable Devices
- PROM Programming Software (iPPS) Makes Programming Easy with IBM PC, XT, AT, and PC Compatibles
- Supports Personality Modules and GUPI Base W/Adaptors
- iUP-200A Provides On-Line Operation with a Built-In Serial RS232 Interface and Software for a PC Environment
- iUP-201A Provides Same On-Line Performance and Adds Keyboard and Display for Stand-Alone Use
- iUP-201A Stand-Alone Capability Includes Device Previewing, Editing, Duplication, and Download from any Source Over RS232C Port
- Updates and Add-Ons Have Maintained Even the Earliest iUP-200 and iUP-201 Users at the State-of-Art

The iUP-200A and iUP-201A universal programmers program and verify data in Intel and Intel compatible, programmable devices. The iUP-200A and iUP-201A universal programmers provide on-line programming and verification in a growing variety of development environments using the Intel PROM programming software (iPPS). In addition, the iUP-201A universal programmer supports off-line, stand-alone program editing, duplication, and memory locking. The iUP-200A universal programmer is expandable to an iUP-201A model.



210319-1

## FUNCTIONAL DESCRIPTION

The iUP-200A universal programmer operates in on-line mode. The iUP-201A universal programmer operates in both on-line and off-line mode.

### On-Line System Hardware

The iUP-200A and iUP-201A universal programmers are free-standing units that, when connected to a host computer with at least 64K bytes of memory, provide on-line programming and verification of Intel programmable devices. In addition, the universal programmer can read the contents of the ROM versions of supported devices.

The universal programmer communicates with the host through a standard RS232C serial data link. Different versions of the iUP-200A and iUP-201A are equipped with different cables, including the cable most commonly used for interfacing to that host. Care should be taken that the version with the correct cable for your particular system is selected, as cable requirements can vary with your host configuration. A serial converter is needed when using the MDS 800 as a host system. (Serial converters are available from other manufacturers.)

Each universal programmer contains the CPU, selectable power supply, static RAM, programmable timer, interface for personality modules, RS232C interface for the host system, and control firmware in EPROM. The iUP-201A also has a keyboard and display.

A personality module or GUPI Adaptor adapts the universal programmer to a family of devices; it contains all the hardware and software necessary to program either a family of Intel devices or a single Intel device. The user inserts the personality module into the universal programmer front panel.

### On-Line System Software

The iUP-200A and iUP201A includes your choice of one copy of Intel's PROM Programming software iPPS, selected from a list of versions for different operating systems and hosts. Each version includes the software implementation designed for that host and O.S. and the RS232C cable most commonly used. Additional versions may be purchased separately if you decide to change hosts at a later date. The iPPS software provides user control through an easy-to-use interactive interface. The iPPS software performs the following functions to make EPROM programming quick and easy:

- Reads devices
- Programs devices directly or from a file

- Verifies device data with buffer data
- Locks device memory from unauthorized access (on devices which support this feature)
- Prints device contents on the network or development system printer
- Performs interactive formatting operations such as interleaving, nibble swapping, bit reversal, and block moves
- Programs multiple devices from the source file, prompting the user to insert new devices
- Uses a buffer to change device contents

All iPPS commands, as well as program address and data information, are entered through the host system ASCII keyboard and displayed on the system CRT.

The iPPS software supports data manipulation in the following Intel formats: 8080 hexadecimal ASCII, 8080 absolute object, 8086 hexadecimal ASCII, 8086 absolute object, and 80286 absolute object. Addresses and data can be displayed in binary, octal, decimal, or hexadecimal. The user can easily change default data formats as well as number bases. iPPS can also access disk files.

For programming Intel EPLDs, the iUP-200A/201A can be controlled by Intel's Logic Programming Software (LPS). LPS programs EPLDs from JEDEC files produced by Intel's logic compiler. (iPPS can also program EPLDs, but only from pre-programmed device masters.)

## System Expansion

The iUP-200A universal programmer can be easily upgraded (by the user) to an iUP-201A universal programmer for off-line operation. The upgrade kit (iUP-PAK-A) is available from Intel or your local Intel distributor.

### Off-Line System

The iUP-201A universal programmer has all the on-line features of the iUP-200A universal programmer plus off-line editing, device duplication, program verification, and locking of device memory independent of the host system. The iUP-201A universal programmer also accepts Intel hexadecimal programs developed on non-Intel development systems. Just a few keystrokes download the program into the iUP RAM for editing and loading into a device.

Off-line commands are entered via a 16-character keypad. A 24-character display shows programmer status.

## SYSTEM DIAGNOSTICS

Both the iUP-200A and iUP-201A universal programmers include self-contained system diagnostics that verify system operation and aid the user in fault isolation.

## PERSONALITY MODULES

For some devices, a personality module is the interface between the iUP-200A/iUP-201A universal programmer (or an iPDS system) and a selected device. Personality modules contain all the hardware and firmware for reading and programming a family of Intel devices. Table 1 lists the devices supported by the different modules.

For most devices, the GUPI module and interchangeable GUPI Adaptors provide the interface between the programmer and the device being programmed (see Figure 1). The GUPI (Generic Universal Programmer Interface) module is a base module that interfaces to the iUP-200A/201A and GUPI

Adaptors. GUPI Adaptors tailor the GUPI module base signals to a family of devices or an individual device. The GUPI module and GUPI Adaptors provide a lower-cost method of device support than if unique Personality Modules were offered for each device/family. Tables 2 and 3 show which Adaptors support which devices.

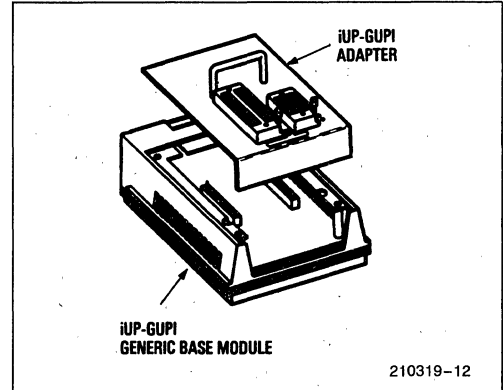


Figure 1. GUPI Adaptor

Table 1. iUP Personality Programming Modules

Device Type	Fast 27/K Module	Fast 27/K U2 Kit	Fast 27/K-CON* Kit	F27/128 Module	F87/44A Module	F87/51A Module
EPROM	2764 2764A  27128  27256	2764 2764A 27C64 87C64 27128 27128A 27256 27C256 27512 27513	2764 2764A 27C64 87C64 27128 27128A 27256 27C256 27512 27513	2716 2732 2732A 2764		
				27128		
				27256		
E <sup>2</sup> PROM		2817A	2817A	2815 2816		
Microcontroller					8041A 8042 8044AH 8741H 8742 8744H	8748 8748H  8749H 8751 8751H 8048 8048H 8049 8049H 8050H 8051
					8755A	

\*Quick-Pulse Programming™ algorithm

**Table 2. iUP-GUPI Adaptors for Programming Memories**

Device Type	GUPI 27010	GUPI 27011	GUPI 27210	GUPI Flash	GUPI 8742	GUPI MCS-51	GUPI 8796	GUPI 8796LCC	GUPI 87C51GB	GUPI MCS-96LCC
EPROM	27010	27011	27210							
Flash				27F64 27F256 28F256						
Peripheral Microcontroller					8741AH 8742AH	8751H 87C51 8752BH 87C51FA 87C51FB	8794BH 8795BH 8796BH 8797BH	8796BH 8797BH	87C51GB	8797BH 87C196KB
Package Types	DIP	DIP	DIP	DIP	DIP	PLCC DIP	PGA DIP	LCC	PLCC	PLCC

**Table 3. Programming Adaptors for EPLDs**

Device Type	GUPI Logic-IID	GUPI Logic-12	GUPI 40D44J	GUPI Logic-18	GUPI Logic-18PGA	GUPI 85EPLD28	GUPI Logic-BIC
EPLD	5C060 5C090  5AC312	5C031 5C032  5C121	   5AC324	5C180	5C180G	85C508	5CBIC
Package Types	DIP*	DIP	DIP PLCC	PLCC CJ	PGA	DIP PLCC	PLCC

\*ADAPT units available to adapt DIP socket for PLCC package.

## iUP-200A/iUP201A SPECIFICATIONS

### Control Processor

Intel 8085A microprocessor  
6.144 MHz clock rate

### Memory

RAM—4.3 bytes static  
ROM—12K bytes EPROM

### Interfaces

Keyboard: 16-character hexadecimal and 12-function keypad (iUP-201A model only)

Display: 24-character alphanumeric (iUP-201A model only)

### Software

Monitor—system controller in pre-programmed EPROM

iPPS — Intel PROM programming software on supplied diskette

### Physical Characteristics

Depth: 15 inches (38.1 cm)

Width: 15 inches (38.1 cm)

Height: 6 inches (15.2 cm)

Weight: 15 pounds (6.9 kg)

### Electrical Characteristics

Selectable 100, 120, 200, or 240 Vac  $\pm$  10%; 50-60 Hz

Maximum power consumption—80 watts

### Environmental Characteristics

Reading Temperature: 10°C to 40°C

Programming Temperature: 25°C  $\pm$  5°

Operating Humidity: 10% to 85% relative humidity

### Reference Material

166041-001— *iUP-200A/201A Universal Programmer User's Guide.*

166042-001— *Getting Started with the iUP-200A/201A (For ISIS/iNDX Users).*

166043-001— *Getting Started with the iUP-200A/201A (For DOS Users).*

164853 — *iUP-200A/201A Universal Programmer Pocket Reference.*

## ORDERING INFORMATION

### Product

Order Code	Description
iUP-200A 211A	On-Line PROM programmer with iPPS rel 1.4 on Single density ISIS II floppy
iUP-200A 212B	On-Line PROM programmer with iPPS rel 1.4 on Double density ISIS II floppy
iUP-200A 213C	On-Line PROM programmer with iPPS rel 2.0 for Series IV, on mini-floppy
iUP-200A 216D	On-Line PROM programmer with iPPS rel 2.0 for PC/DOS, and cable for PC or XT
iUP-200A 217D	On-Line PROM programmer with iPPS rel 2.0 for PC/DOS, and cable for AT
iUP-201A 211A	Off-Line and on-line PROM programmer with iPPS rel 1.4 on Single density ISIS II floppy
iUP-201A 212B	Off-Line and on-line PROM programmer with iPPS rel 1.4 on Double density ISIS II floppy
iUP-201A 213C	Off-Line and on-line PROM programmer with iPPS rel 2.0 for Series IV on mini-floppy
iUP-201A 216D	Off-Line and on-line PROM programmer with iPPS rel 2.0 for PC/DOS, and cable for PC or XT
iUP-201A 217D	Off-Line and on-line PROM programmer with iPPS rel 2.0 for PC/DOS, and cable for AT
iUP-200/201 U1* Upgrade Kit	Upgrades an iUP-200/201 universal programmer to an iUP-200A/201A universal programmer
iUP-DL	Download Support Kit for iUP-200A/201A upgrades programmer to support adaptors that use software programming (.DSS) files.
iUP-PAK-A Upgrade Kit	Upgrades an iUP-200/A universal programmer to an iUP-201A universal programmer

\*Most personality modules can be used only with an iUP-200A/201A universal programmer or an iUP-200/iUP201 universal programmer upgraded to an A with the iUP-200/iUP-201 U1 upgrade kit.



<b>Product Order Code</b>	<b>Description</b>
piUP-GUPI	Generic Universal Programmer Interface (Base)

**Software Sold Separately**

<b>Product Order Code</b>	<b>Description</b>
211A	PROM programming software rel 1.4 on Single density ISIS II floppy
212B	PROM programming software rel 1.4 on Double density ISIS II floppy

<b>Product Order Code</b>	<b>Description</b>
213C	PROM programming software rel 2.0 for Series IV on mini-floppy
216D	PROM programming software rel 2.0 for PC/DOS with cable for PC or PC XT
217D	PROM programming software rel 2.0 for PC/DOS with cable for PC AT

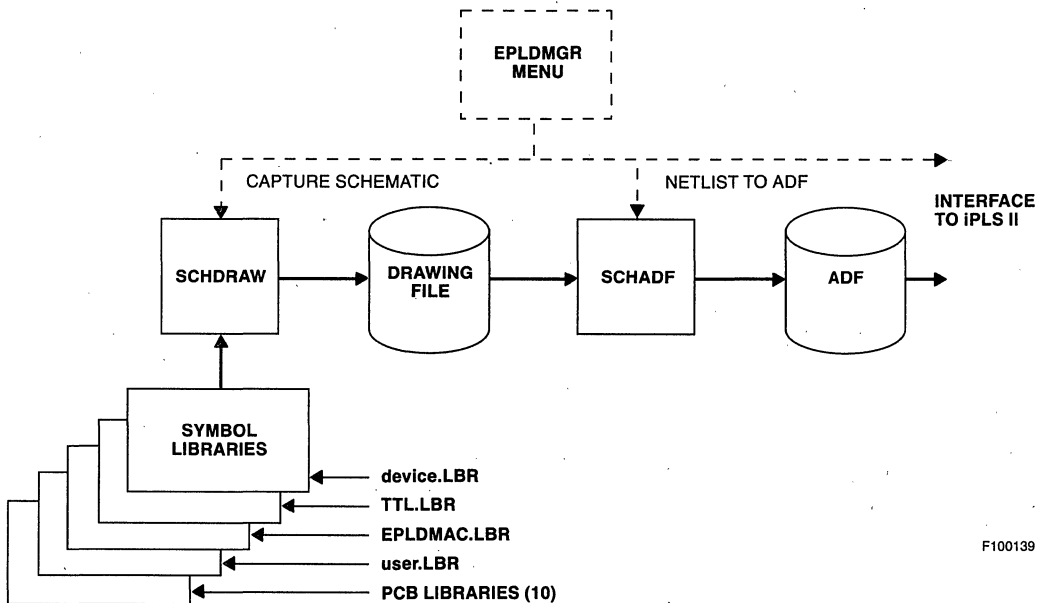
# SCHEMA II-PLD

SCHEMA II-PLD is a low-cost schematic capture software for designing with Intel EPLDs and with standard MSI, SSI, and discrete components. For EPLD designs, SCHEMA II-PLD outputs Advanced Design Files (ADFs) that can subsequently be compiled by iPLS II software. Figure 1 shows the flow to generate a drawing file and convert it to an ADF for processing by iPLS II. SCHEMA II-PLD supports EPLD design primitive symbols as well as MSI and SSI macro symbols, allowing designers to combine TTL and EPLD symbols as needed. An EPLD Custom library supports groups of EPLD symbols and "generic" function symbols such as counter, multiplexers, etc. The ability to create user-defined symbols that can be translated into ADF macro calls adds to SCHEMA II-PLD's power and versatility.

SCHEMA II-PLD provides fast, smooth panning, combined mouse/keyboard support, instant command execution, and automatic "step and repeat" to make schematic capture as quick and easy as possible. In addition to the symbol libraries targeted for EPLD design, SCHEMA II-PLD provides over 10 symbol libraries for standard PCB design. Its sophisticated library management routines, reentrant object editor, and true "hierarchical" design capability makes SCHEMA II-PLD a powerful tool for professional designers.

The EPLD Manager software included with SCHEMA II-PLD provides a single user interface to both SCHEMA II-PLD and iPLS II software modules. EPLD Manager software is also available separately to users who already own SCHEMA II.

**Order Codes:**      **SCHEMA II-PLD**      (SCHEMA II and EPLD Manager)  
                          **EPLDMGR**              (EPLD Manager)



F100139

**Figure 1. SCHEMA II-PLD Schematic Capture Flow for EPLD Designs**

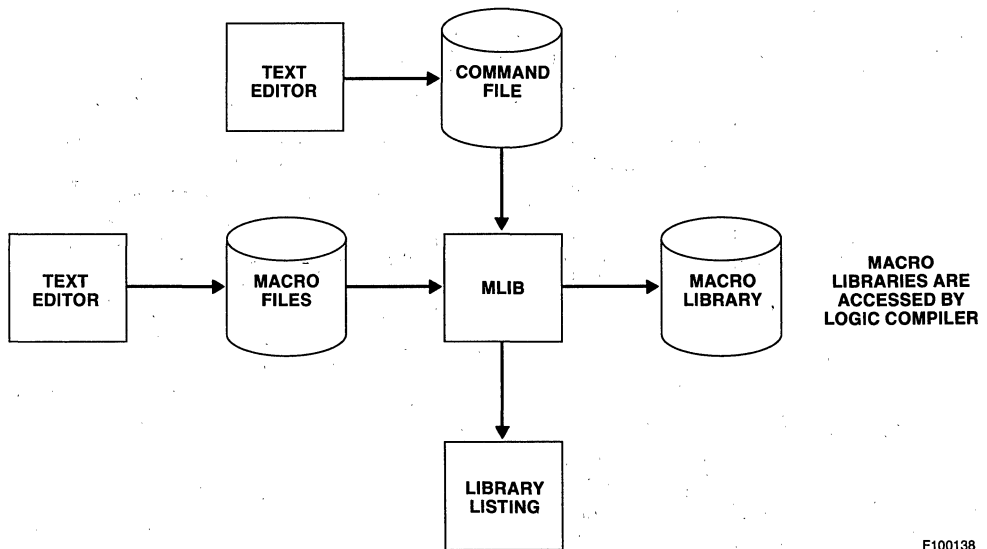
## iPLS II MACRO LIBRARIAN

The iPLS II Macro Librarian (MLIB) is a software package that allows designers to build user-defined macro definition libraries for EPLD designs. Macro libraries can include TTL macros available from Intel, or proprietary macros developed by a user. User-defined macros are developed as individual macro files using a text editor, and then are combined into macro libraries by MLIB, where they can be accessed by the LOC. Figure 1 shows the flow to build a macro library. Use of macros in ADFs (Advanced Design Files) allows EPLD design to proceed at a higher level than with EPLD primitives alone.

Macro files are standard ASCII files that describe the function of the macro. The Network and Equations sections of macro files follow ADF format. The header section, which differs from ADF format, defines the macro name, calling sequence, and defaults. MLIB combines these files into a library that can be accessed by the macro expander in the LOC (Logic Optimizing Compiler). MLIB can be invoked from the command line, from command files, or from a combination of both. The macro expander identifies and expands each macro call in an ADF with the corresponding macro definition from macro libraries. The first occurrence of a macro is used.

Two preconfigured libraries are available from Intel: (1) TTL macro library, and (2) an EPLD Custom macro Library. These libraries are described in the "PLDUTIL" Product Brief.

**Order Code: MLIB**



**Figure 1. Flow to Build a Macro Library**

# PLDUTIL

PLDUTIL V1.0 contains the following utilities for designing with Intel EPLDs:

- SIM, version 2.1 of a basic Functional Simulator for EPLD designs
- TTL.LIB, version 3.6 of Intel's TTL macro definition library
- EPLDMAC.LIB, version 1.0 of Intel's EPLD custom macro definition library.

## Functional Simulator

The Functional Simulator allows designers to perform basic function simulation of EPLD designs. By verifying proper operation of a design with the Simulator, designers can catch logic errors before devices are programmed and installed in products.

Design information is provided by the minimized LEF (Logic Equation File) generated by the iPLS II logic compiler. Input stimulus for the Simulator is in the form of a user-generated ASCII vector file containing strings of 1s and 0s (see Figure 1). Vector files can also contain expected output values to serve as a reference for the simulated outputs. The simulator produces state machine or waveform output and supports bidirectional signals. Output registers can be preloaded to speed the process of simulating counter and state machine transitions.

The Functional Simulator operates on any IBM PC/XT, PC/AT, or compatible computer. A.C. timing simulation is not supported.

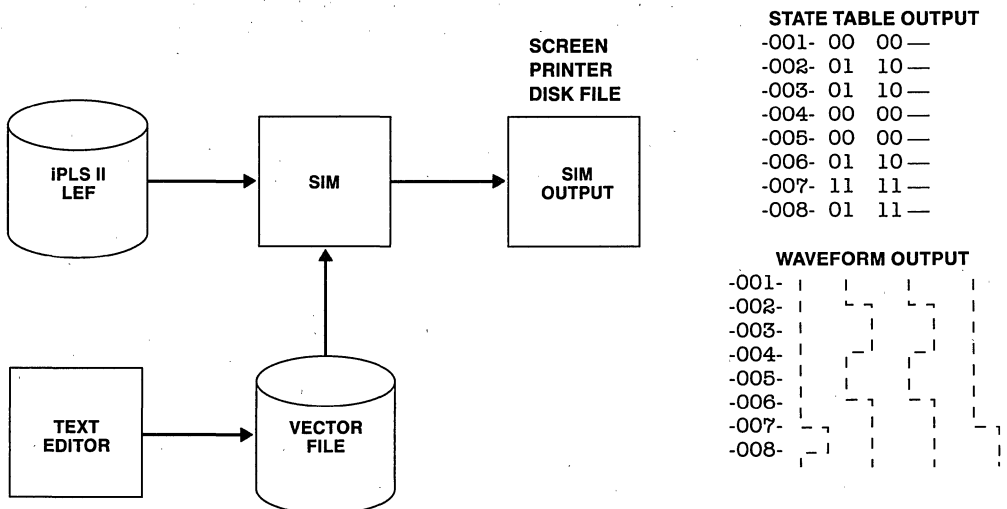


Figure 1. Functional Simulator Flow

## **TTL Macro Library**

The TTL Macro Library contains macro definitions for most common 74-series TTL devices. The library is accessed by the iPLS II macro expander module when compiling an ADF (Advanced Design File). When the macro expander identifies a macro call in an ADF, it searches available libraries for the respective ADF macro definition, and replaces the macro call by the ADF implementation.

Macro definitions implement the TTL functions via EPLD design primitives and Boolean equations. In some cases, precise TTL emulation is not possible. In addition to the built TTL macro library, the TTL.LIB disk contains the individual device files (.DEV) used for each macro, and document files (.DOC) describing the implementation details for each macro. The device files can be used to build user-defined macro libraries using the iPLS II Macro Librarian.

TTL.LIB provides the ADF macro definitions for compilation. Macro symbols for use with supported schematic capture packages are provided with the schematic capture software. A complete listing of the contents of the TTL macros in the library is provided in Applications Brief AB-18, *TTL Macro Library Listing for EPLD Designs*.

## **EPLD Custom Macro Library**

The EPLD Custom Macro Library contains macro definitions for a set of common EPLD primitive groups and "generic" logic functions. Included in the library are groups of INPs, CONFs, RORFs, etc. Also included are frequently used counters, multiplexers, decoders, etc. The library is accessed by the iPLS II macro expander module when compiling an ADF (Advanced Design File). When the macro expander identifies a macro call in an ADF, it searches available libraries for the respective ADF macro definition, and replaces the macro call by the ADF implementation.

In addition to the built library, the EPLDMAC.LIB disk contains the individual device files (.DEV) used for each macro, and document files (.DOC) describing the implementation details for each macro. The device files can be used to build user-defined macro libraries using the iPLS II Macro Librarian.

EPLDMAC.LIB provides the ADF macro definitions for compilation. Macro symbols for use with supported schematic capture packages are provided with the schematic capture software. A complete listing of the contents of EPLD Custom Macro Library is provided in Applications Brief AB-21, *EPLD Custom Macro Library Listing for EPLD Designs*.

**Order Code:**        **PLDUTIL**        (Functional Simulator, TTL.LIB, EPLDMAC.LIB and User Documentation)

# UTILITIES

## PAL2ADF UTILITY

### Description

This document is a brief note on the use of the PAL2ADF program in translating PALASM 1 files into Intel's Advanced Design File (ADF) format. Descriptions for actual use can be found on the accompanying Manual page in the file PAL2ADF.MAN.

The PALASM file serves as a template for mapping the PALASM equations into ADF. The translation is performed as follows:

- 1) Read PAL description, and set the PAL pins to their appropriate EPLD primitive counterparts
- 2) Parse file and produce network description
- 3) Translate equations to ADF

### PAL Configuration Database

When it is translating a PALASM file, PAL2ADF reads a database (default:PAL2ADF.DAT) that tells it:

- How many pins the PAL has
- Which default EPLD to translate to
- What pins are special inputs (Clock and Output Enable defaults)
- What EPLD I/O primitives to use for each PAL pin

The EPLD I/O primitives specify the network architecture that the EPLD must take on in order to mimic the functionality of the PAL. See the PAL2ADF.DAT file for more information.

### Reconfiguring Outputs

In step (2) above, several checks are done in order to make sure that the network is configured appropriately. These primarily involve output pins, although input pins can be specified as well.

The first reconfiguration is for active low outputs in their equations. i.e.,

PALASM:  $\text{/SIGNAL} = A * \text{/B} + C$

becomes

ADF:  $\text{SIGNAL} = \text{/(A * /B + C)}$ ;

The other reconfigurations are slightly more complex. Consider a PAL pin X which is an output with a D-latch. The output value is fed back into the P-term array after the Output Enable. This is described as a Registered Output Registered Feedback (RORF) in the Intel EPLDs. The default network description for this pin then is:

NETWORK:

...  
 $X, X = \text{RORF} (X_p, \text{CLK}, \text{GND}, \text{GND}, \text{OE})$   
...

where CLK and OE are the default Clock and Output Enable signals.

Normally, there would be an equation that would describe  $X_p$ . (The 'p' is used to name the P-term value.) If, however, the X feedback is never used in an equation, then the I/O macrocell is reconfigured to a Registered Output No Feedback (RONF).

NETWORK:

$$\overset{\dots}{X} = \text{RONF}(\overset{\dots}{X_p}, \text{CLK}, \text{GND}, \text{GND}, \text{OE})$$

For those I/O pins on the PAL which are used strictly as inputs, these use the Combinatorial Output I/O Feedback (COIF) primitive, with the Output Enable shut off (GND). The P-term is tied to the feedback, in order to satisfy the semantics of ADF.

NETWORK:

$$\overset{\dots}{YY}, \overset{\dots}{YY} = \text{COIF}(\overset{\dots}{YY_p}, \text{GND})$$

EQUATIONS:

$$\overset{\dots}{YY_p} = \overset{\dots}{YY};$$

If the PAL pin is being used strictly as an output and is never used in an equation, then the primitive is reconfigured to a Combinatorial Output No Feedback (CONF).

NETWORK:

$$\overset{\dots}{YY}, \overset{\dots}{YY} = \text{CONF}(\overset{\dots}{YY_p}, \text{GND})$$

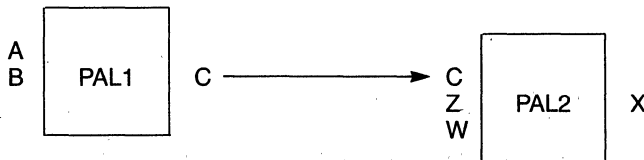
This is the same as above where a RORF is reconfigured to a RONF.

### Multiple PAL Designs into 1 EPLD

It is possible to incorporate multiple PALASM descriptions into one EPLD. If each PALASM description is disjoint, (i.e., they have different pin names for each pin) then you can simply translate each file (with the pinlist information OFF) and compile them together with the iPLS Logic Optimizing Compiler (LOC).

The compiler allows you to specify multiple ADF files, allowing different subnetworks within one EPLD. You will probably want to use a larger EPLD to fit all the designs in.

If the PAL designs are not disjoint, then there are some steps that can be done by hand to integrate the designs. A simple example would be where one PAL feeds another a signal, and the second uses that to generate another signal.



In this case, C is an output of PAL1, and an input to PAL2. In PAL2, C,Z, and W generate the signal X. Suppose we have the equations:

PAL1

$$C = A / B$$

PAL2

$$X = C * Z * W + C / Z * W$$

In the resulting ADFs, the following NETWORKS are produced:

ADF for PAL1:

NETWORK:

$$A = \text{INP}(A)$$

$$B = \text{INP}(B)$$

$$C = \text{CONF}(C_p, VCC)$$

EQUATIONS:

$$C = A / B;$$

ADF for PAL2:

NETWORK:

$$Z = \text{INP}(Z)$$

$$W = \text{INP}(W)$$

$$C = \text{INP}(C)$$

$$X, X = \text{RORF}(X_p, \text{CLK}, \text{GND}, \text{GND}, \text{OE})$$

EQUATIONS:

$$X_p = C * Z * W + C / Z * W;$$

These can be joined together into a single ADF:

NETWORK:

$$A = \text{INP}(A)$$

$$B = \text{INP}(B)$$

$$Z = \text{INP}(Z)$$

$$W = \text{INP}(W)$$

$$X, X = \text{RORF}(X_p, \text{CLK}, \text{GND}, \text{GND}, \text{OE})$$

EQUATIONS:

$$C = A / B;$$

$$X_p = C * Z * W + C / Z * W;$$

Notice how C is now an intermediate variable rather than an actual signal. This is obviously a simple example, yet similar techniques can be applied to more complex cases. As much more logic can be placed into larger EPLDs, the job of splitting functions across multiple devices is reduced.

## Availability

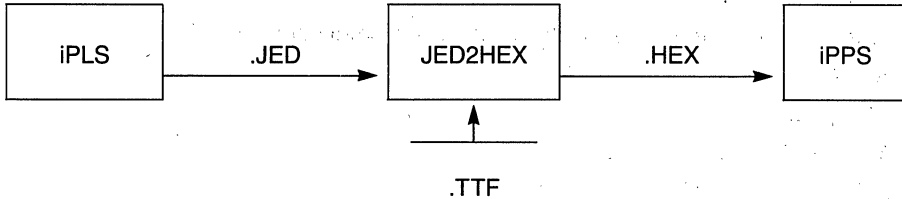
The PAL2ADF utility is available at no cost to Intel EPLD customers. Contact your local Intel sales office.



# JED2HEX CONVERSION UTILITY

## Description

JED2HEX is a utility to convert JEDEC files created by iPLS (.JED) into Intellec HEX files which can then be read by Intel's iPPS software. This allows programming of EPLDs via Intel's iUP200A/201A using a GUP I base and the appropriate adaptor (e.g. LOGIC-12). The following diagram represents a typical development cycle.



**INSTALLATION:** To install the utility and its device specific files, place the master disk in drive A: and invoke the JINSTALL.BAT batch file with the destination path for the utility and device files. Example:

```
A: JINSTALL C: MYPATH
```

When using JED2HEX, attach the package description letter when entering the device type. That is, enter 5C121D for a 5C121 ceramic DIP when prompted for the device type. Entering 5C121 will result in:

```
***ERROR: Device File Missing
```

To determine the packages supported in your JED2HEX software, examine all the .tff extension files; it is the .tff files which the device type command attempts to match.

When using iPPS, a file format of 8080 or 8086 must be specified when copying the JED2HEX generated HEX file to the buffer or directly into a device. If 8080 or 8086 is not specified, the default file format type of 80386 will be chosen and a "GENERAL ERROR — ILLEGAL FILE TYPE SPECIFIED" will result. An example of the proper COPY format:

```
PPS> COPY a: filename.HEX TO PROM 86
```

## Availability

The JED2HEX Conversion Utility is available at no cost to Intel EPLD Customers. Contact your local Intel sales office.



October 1988

# **TTL Macro Library Listing for EPLD Designs**

**PROGRAMMABLE LOGIC APPLICATIONS  
INTEL CORPORATION**

Order Number: 292037-003

## TTL Macros

The following is a list of TTL macros that are in TTL.LIB version 3.6. This library is available through the Intel EPLD customer hot line.

These macros are called from an Advanced Design File (ADF). Schematic capture packages such as Schema II-PLD create ADFs with the correct macro invocation for each TTL device listed here.

Macros listed here are grouped by general function.

### SSI GATES

7400	2 Input NAND
7402	2 Input NOR
7404	1 Input INVERTER
7408	2 Input AND
7410	3 Input NAND
7411	3 Input AND
7420	4 Input NAND
7421	4 Input AND
7427	3 Input NOR
7430	8 Input NAND
7432	2 Input OR
7486	2 Input XOR

## MSI FUNCTIONS

### Decoders/Demultiplexers

7442	(10) BCD to Decimal
7444	(10) Excess-3-Gray to Decimal
7447X	(7) BCD to 7-Segment—Active Low Output
7449	(7) BCD to 7-Segment—Active High Output
74138	(8) 1-of-8 Decoder
74139	(4) Single 1-of-4 Decoder
74145	(10) BCD to Decimal
74154	(16) 1-of-16 Decoder
74155	(8) Dual 1-of-4
74156	(8) Dual 1-of-4

### Multiplexers

74151	(2) 8-to-1
74153	(2) Dual 4-to-1—Active High Output
74157	(4) Quad 2-to-1—Active High Output
74158	(4) Quad 2-to-1—Active Low Output
74253	(2) Dual 4-to-1—Three-State Output
74257X	(4) Quad 2-to-1—Active High, Three-State Output
74258X	(4) Quad 2-to-1—Active Low, Three-State Output
74298XA	(4) Quad 2-to-1—Active High with Storage
74298XB	(4) Quad 2-to-1—Active High with Storage
74352	(2) Dual 4-to-1—Active Low Output

**Counters**

	Type	Clear	Load	Clk	Extras
7490XD	(4) BCD Decade	S	9	R	
7490XQ	(4) Bi-Quinary	S	9	R	
74160	(5) BCD Decade	A	S	R	RCO
74161	(5) 4-Bit Binary	A	S	R	RCO
74162	(5) BCD Decade	S	S	R	RCO
74163	(5) 4-Bit Binary	S	S	R	RCO
74168	(5) BCD Decade	—	S	R	U/D, RCO
74169	(5) 4-Bit Binary	—	S	R	U/D, RCO
74176XD	(4) BCD Decade	A	S	R	
74176XQ	(4) Bi-Quinary	A	S	R	
74177X	(4) 4-Bit Binary	A	S	R	
74190XA	(6) BCD Decade	—	S	R	U/D, RCO, MM
74190XB	(6) BCD Decade	—	S	R	U/D, RCO, MM
74191XA	(7) 4-Bit Binary	—	S	R	U/D, RCO, MM
74290XD	(4) BCD Decade	S	9	R	
74290XQ	(4) Bi-Quinary	S	9	R	
74390X	(4) Bi-Quinary/BCD	A	—	F	
74393XA	(4) 4-Bit Binary	A	—	F	
74393XB	(4) 4-Bit Binary	A	—	F	

S = Synchronous  
A = Asynchronous  
9 = Synchronous Set-to-9

R = Rising-Edge Triggered  
F = Falling-Edge Triggered

U/D = Up/Down  
RCO = Ripple Carry Output  
MM = Max/Min Output

**Single Flip-Flops**

7472XA	(2) AND-Gated JK Master/Slave
7472XB	(2) AND-Gated JK Master/Slave
7473X	(2) JK with Clear
7474X	(2) D with Preset and Clear
74112XA	(3) JK with Preset and Clear
74112XB	(2) JK with Clear

**Latches**

7475X	(8) 4-Bit Bistable
7477X	(4) Quad D-Type
74259XA	(8) Octal Addressable D-Type
74259XB	(8) Octal Addressable D-Type
74373X	(8) Octal D-Type

**Multiple Flip-Flops (Registers)**

74174X	(6) Hex D
74175X	(8) Quad D with Q and /Q
74273X	(8) Octal D
74377	(8) Octal D with Common Enable
74378	(6) Hex D

## Shift Registers

7491	(8)	8-Bit—Serial-In, Serial-Out
7495XA	(4)	4-Bit—Serial-In/Parallel-In, Parallel-Out
7495XB	(4)	4-Bit—Serial-In/Parallel-In, Parallel-Out
7495XC	(4)	4-Bit—Serial-In/Parallel-In, Parallel-Out
7496X	(5)	5-Bit—Serial-In/Parallel-In, Parallel-Out
74164	(8)	8-Bit—Serial-In, Parallel-Out
74165X	(9)	8-Bit—Serial-In/Parallel-In, Serial-Out
74194	(4)	4-Bit Bi-Directional—Serial-In/Parallel-In, Parallel-Out
74395XA	(5)	4-Bit Cascadable—Serial-In/Parallel-In, Parallel-Out
74395XB	(5)	4-Bit Cascadable—Serial-In/Parallel-In, Parallel-Out

## Miscellaneous

7482X	(4)	2-Bit Adder
7483X	(8)	4-Bit Adder
7485X	(7)	4-Bit Magnitude Comparator
7487	(4)	4-Bit True/Complement Element
74143X	(17)	4-Bit Counter; 4-Bit Latch; 7 Segment Decoder
74180X	(4)	8-Bit Parity Generator/Checker
74180XA	(4)	8-Bit Parity Generator/Checker
74182	(5)	Look-Ahead Carry Generator
74183	(2)	Single-Bit Full Adder with Carry/Save
74280X	(5)	9-Bit Odd/Even Parity Generator/Checker

## DEMORGAN EQUIVALENTS (BUBBLE GATES)

	Bubble AND (NOR)	Bubble NAND (OR)	Bubble NOR (AND)	Bubble OR (NAND)
2 Input	BAND2	BNAND2	BNOR 2	BOR2
3 Input	BAND3	BNAND3	BNOR 3	BOR3
4 Input	BAND4	BNAND4	BNOR 4	BOR4
6 Input	BAND6	BNAND6	BNOR 6	BOR6
8 Input	BAND8	BNAND8	BNOR 8	BOR8
12 Input	BAND12	BNAND12	BNOR 12	BOR12

## INPUT/OUTPUT MACROS

INPUT	N/A	Generates Input Pin and Node in ADF
OUTPUT	(1)	Generates Enabled Output Buffer in ADF
OUTP	(1)	Output Pin (Used in SCHEMA II-PLD)
74125	(1)	Single Three-State Output, Active Low Enable
74126	(1)	Single Three-State Output, Active High Enable

### NOTES:

1. All TTL macros duplicate TTL function only. They **DO NOT DUPLICATE** performance characteristics such as open-collector, totem-pole, or high-drive output.
2. Any TTL macros which deviate in some way from standard TTL function are denoted with an appended "X" (see device .DOC file for details). Appended "D"s and "Q"s indicate counters configured to Decimal or bi-Quinary mode; appended "A"s and "B"s indicate a macro configured for a family of EPLD devices (e.g. 5C060, 5C090, 5C180).
3. The (#) indicates the maximum number of EPLD macrocells consumed if all outputs are used. If an output is not used, the macro compression phase of the Macro Expander will remove the signal unless it is used as feedback inside the macro definition.
4. /Q's should be avoided as pin outputs if possible. The EPLD is structured such that the Q is readily available as a pin output and both the Q and /Q are readily available as feedbacks. Using /Q as a pin output, however, requires an extra macrocell and adds to the propagation delay.



# APPLICATION BRIEF

AB-21

October 1988

## **EPLD Custom Macro Library Listing for EPLD Designs**

PROGRAMMABLE LOGIC APPLICATIONS  
INTEL CORPORATION

Order Number: 292050-001

## EPLD CUSTOM MACROS

The following is a list of the macros contained in version 1.0 of Intel's EPLD Custom Macro Library (EPLDMAC.LIB). This library is available through the Intel EPLD customer hot line.

These macros are called from an Advanced Design File (ADF). Schematic capture packages such as SCHEMA II-PLD create an ADF with the correct macro invocation syntax for each macro listed here.

The macros are grouped by function. The macro name is followed by the least number of macrocells used and a description of the macro's function.

### INPUTS

2INP	(0)	2 Input Pins
4INP	(0)	4 Input Pins
6INP	(0)	6 Input Pins
8INP	(0)	8 Input Pins

### BURIED FEEDBACK

4NOCF	(4)	4 "No Output Combinational Feedback" I/O Primitives
6NOCF	(6)	6 "No Output Combinational Feedback" I/O Primitives
8NOCF	(8)	8 "No Output Combinational Feedback" I/O Primitives

### COMBINATIONAL I/O

4CONF	(4)	4 "Combinational Output No Feedback" I/O Primitives
6CONF	(6)	6 "Combinational Output No Feedback" I/O Primitives
8CONF	(8)	8 "Combinational Output No Feedback" I/O Primitives
4COIF	(4)	4 "Combinational Output Input Feedback" I/O Primitives
6COIF	(6)	6 "Combinational Output Input Feedback" I/O Primitives
8COIF	(8)	8 "Combinational Output Input Feedback" I/O Primitives

### REGISTERED I/O

4RONF	(4)	4 "Registered Output No Feedback" I/O Primitives
6RONF	(6)	6 "Registered Output No Feedback" I/O Primitives
8RONF	(8)	8 "Registered Output No Feedback" I/O Primitives
4ROIF	(4)	4 "Registered Output Input Feedback" I/O Primitives
6ROIF	(6)	6 "Registered Output Input Feedback" I/O Primitives
8ROIF	(8)	8 "Registered Output Input Feedback" I/O Primitives
4RORF	(4)	4 "Registered Output Registered Feedback" I/O Primitives
6RORF	(6)	6 "Registered Output Registered Feedback" I/O Primitives
8RORF	(8)	8 "Registered Output Registered Feedback" I/O Primitives

**LATCHES/REGISTERS**

4REG	(4)	4 Registers with Common Clock and Clear
6REG	(6)	6 Registers with Common Clock and Clear
8REG	(8)	8 Registers with Common Clock and Clear
4LATCH	(4)	4 Transparent Data Latches with Common Enable
6LATCH	(6)	6 Transparent Data Latches with Common Enable
8LATCH	(8)	8 Transparent Data Latches with Common Enable
8TRANS	(8)	8-Bit Bi-Directional Data Transceiver
RSLATCH	(1)	Set-Reset Latch
DLATCH	(1)	Standard D-Type, Transparent Latch
DFFPRE	(2)	D Flip-Flop with Preset and Clear

**MULTIPLEXERS/ENCODERS**

2MUX	(0)	2-to-1 Multiplexer
D2MUX	(0)	Two 2-to-1 Multiplexers with Common Select
Q2MUX	(0)	Four 2-to-1 Multiplexers with Common Select
4MUX	(0)	4-to-1 Multiplexer
8MUX	(0)	8-to-1 Multiplexer
16MUX	(0)	16-to-1 Multiplexer
10MUXBCD	(0)	10-to-4 BCD Encoder

**CONVERTERS/DECODERS**

BINGRY	(0)	4-Bit Binary to Gray Code Converter
GRYBIN	(0)	4-Bit Gray Code to Binary Converter
1DEC	(0)	1-to-2 Decoder
2DEC	(0)	2-to-4 Decoder
4DEC	(0)	4-to-16 Decoder
3DEC	(0)	3-to-8 Decoder
7SEG	(0)	4-Bits to Seven Segment Display Decoder

**COUNTERS/DIVIDERS**

2CNT	(2)	2-Bit Counter with Preload and Clear
4CNT	(4)	4-Bit Counter with Preload and Clear
8CNT	(8)	8-Bit Counter with Preload and Clear
16CNT	(16)	16-Bit Counter with Preload and Clear
BCDCNT	(4)	4-Bit BCD Counter with Preload and Clear
FDIV2	(4)	Divides Input Frequency By 2, 4, 8, and 16
FDIV5	(4)	Divides Input Frequency By 5, 10, 15, and 20



**SHIFT REGISTERS**

- 2SHIFT (2) 2-Bit Serial or Parallel In Shift Register with Enable
- 4SHIFT (4) 4-Bit Serial or Parallel In Shift Register with Enable
- 8SHIFT (8) 8-Bit Serial or Parallel In Shift Register with Enable
- 16SHIFT (16) 16-Bit Serial or Parallel In Shift Register with Enable

**ARITHMETIC OPERATIONS**

- 1ADD (0) 1-Bit Full Adder
- 2MULT (0) 2-Bit Multiplier
- 4COMP (0) 4-Bit Magnitude Comparator...Equality Only
- 8COMP (2) 8-Bit Magnitude Comparator...Equality Only
- 8PAREVN (2) 8-Bit Even Parity Generator
- 8PARODD (2) 8-Bit Odd Parity Generator



October 1988

# **Using Macros in EPLD Designs**

**DANIEL E. SMITH**  
PROGRAMMABLE LOGIC APPLICATIONS  
INTEL CORPORATION

Order Number: 292039-002

## INTRODUCTION

The iPLS II (Intel Programmable Logic Software) Logic Optimizing Compiler includes a Macro Expander that supports the use of macros in EPLD designs. This application note shows how to use the TTL and EPLD Custom macros available from Intel with ADFs created by a text editor. Included are descriptions of macro file support, guidelines for using macros, and two design examples.

## OVERVIEW

iPLS II allows designers to include macro calls in design files to implement common circuit functions. Macro calls are subsequently expanded by the LOC (Logic Optimizing Compiler) into ADF network and/or equation entries required to perform the desired functions. Use of macros allows designs to proceed at a high level, which simplifies and shortens the design process. Macros can be connected together or used in conjunction with standard iPLS II EPLD primitives. Designing with macros is analogous in many ways to using sub-routines in software.

Macros can be used in ADFs (Advanced Design Files) created by a text editor, or by several schematic capture software products. This application note covers use of macros in ADFs created by a text editor. Macro support at this level includes the following:

- A TTL macro library (TTL.LIB) for designing with common TTL circuit equivalents
- An EPLD custom macro library (EPLDMAC.LIB) for designing with "generic" macros.

- A Macro Expander in the LOC that expands macro calls in ADFs with the contents of the corresponding macros from libraries.

Figure 1 shows text editor/ADF macro support for iPLS II. Note that the ADF can be created by any standard ASCII text editor (text editor supplied by user). Creation of user-defined macros is covered in application note, AP-312 "Creating Macros for EPLD Designs", order number 292040. Use of macros with schematic capture software is covered in the documentation for the respective software package.

This note discusses use of macros under the following headings:

- Macro Libraries, briefly describes the two libraries available from Intel.
- Using Macros, describes macro files, how to call macros, the process of macro expansion, calling multiple macro calls, and some basic guidelines to follow and pitfalls to avoid.
- Two examples showing use of TTL macros, and mixing macros and EPLD primitives.

## MACRO LIBRARIES

Intel offers two macro libraries: a TTL Library and an EPLD Custom Library.

## TTL Macro Library

A TTL macro library (TTL.LIB) is available from Intel to support design entry using familiar 74-series logic

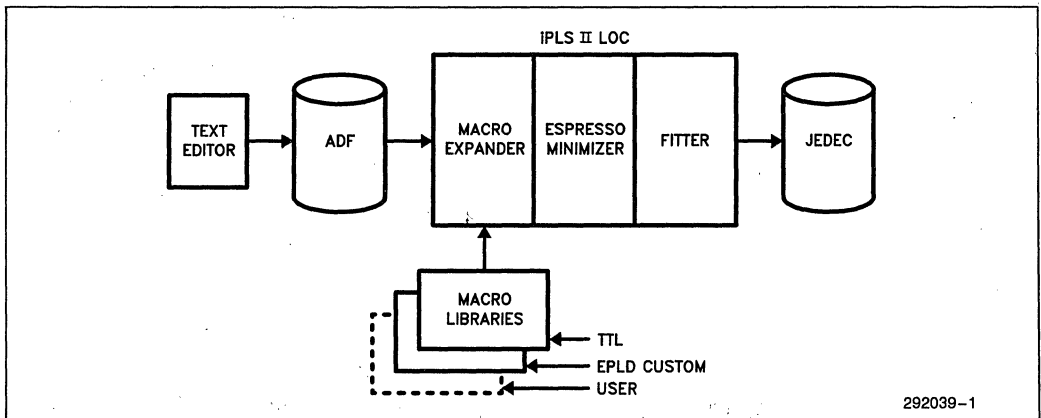


Figure 1. Text Editor/ADF Macro Support for iPLS II

devices. The library contains macros that implement the most widely used 74-series device functions as well as macros for some members of other logic families. Each device in the library is supported by a .DOC file. The .DOC file describes the macro syntax and lists any notable differences between the macro implementation and the TTL part.

### EPLD Custom Macro Library

An EPLD custom macro library (EPLDMAC.LIB) is available from Intel to support design entry using groups of EPLD primitives or "generic" functions such as latches, registers, counters, decoders, etc.

### USING MACROS

The iPLS II Macro Expander is automatically invoked by the LOC when an ADF is submitted to the compiler. When invoked, the Macro Expander identifies macro calls in ADFs, searches macro libraries for a corresponding macro, and expands the call with ADF network and equation entries from the macro file. The expanded file is then compiled normally.

### Macro Files

Figure 2 shows the macro file for a 74138 TTL device, a commonly used one-of-eight decoder. Note that the first line contains the name and I/O signals for the device. Signals are listed in the order in which they appear on the actual TTL device, including VCC and GND (i.e., A = pin 1, B = pin 2, ..., VCC = pin 16). The sequence of signals in this line determines how the macro is "called" from an ADF.

Some of the macros in the TTL library have an "X" suffix appended to the filename, for example 74138X. This suffix indicates that the macro is device-specific (not supported on all EPLDs) or that there is some difference from the TTL device. This information is described in the .DOC file for each macro.

The second line of the macro file contains defaults for each input and place holders (blanks) for each output. The default for an input sets the input to an intelligent level (i.e., enables are enabled, clears, preset, loads are disabled, etc.).

Macro files can contain a Network section, an Equation section, or both. A Network section is not needed when the macro functions can all be implemented in Boolean equations. When used, the Network section contains EPLD design primitives. An Equations section is not needed when the macro functions can all be implemented in the Network section. Macro files end with the keyword "ENDEF".

### Macro Calls

All macro calls appear in the Network section of an ADF. Macro calls use the same part/function name and signal sequence used on the first line of the macro file. The signal names in the macro and the macro call do not need to match, but the order of signals in the call is **crucial** to proper implementation of the macro function. For example, the macro call for the 74138 device could be any one of the following examples:

```
74138 (A, B, C, G2A, G2B, G1, Y7, GND, Y6, Y5, Y4, Y3, Y2, Y1, Y0, VCC)
```

```
74138 (D1, D2, D3, EN1, EN2, EN3, 07, GND, 06, 05, 04, 03, 02, 01, 00, VCC)
```

```

74138(A, B, C, nG2A, nG2B, G1, nY7, GND, nY6, nY5, nY4, nY3, nY2, nY1, nY0, VCC)
DEFAULT: (GND, GND, GND, GND, GND, VCC, , GND, . . . . ., VCC)

NETWORK:

EQUATIONS:

nY0 = !(A * IB * IC * !nG2A * !nG2B * G1);
nY1 = !(A * !B * IC * !nG2A * !nG2B * G1);
nY2 = !(A * B * IC * !nG2A * !nG2B * G1);
nY3 = !(A * !B * IC * !nG2A * !nG2B * G1);
nY4 = !(A * IB * C * !nG2A * !nG2B * G1);
nY5 = !(A * !B * C * !nG2A * !nG2B * G1);
nY6 = !(A * B * C * !nG2A * !nG2B * G1);
nY7 = !(A * B * C * !nG2A * !nG2B * G1);

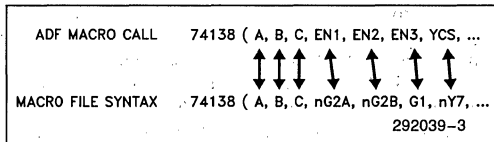
ENDEF
$
    
```

292039-2

Figure 2. Sample TTL Macro File (74138.DEV)

74138 (A, B, C, ENA, ENB, ENC, Y7, GND, Y6, Y5, Y4, Y3, Y2, Y1, Y0, VCC)

In each case, the part name corresponds to the macro part name. The names of the signals differ, but the order of signals match the macro. During processing, the Macro expander assigns node connections between the macro call and the macro file based on the positions of signals, not the names of the signals. For example, note the following macro call to macro file signal assignments:



TTL macro signals originating outside the target EPLD require a prior INPUT macro call in the Network section. All signals used as outputs require a prior OUTPUT macro call in the Network section. Figure 3 shows a sample ADF that uses the 74138 macro. Each input is listed in the INPUTS: declaration and has an INPUT macro call. Outputs are listed in the OUTPUTS: declaration and have OUTPUT macro calls. (EPLD INP and CONF primitive statements may also be used in place of INPUT and OUTPUT macro calls, if desired.)

Gate arrays support a much richer selection of input and output types than EPLDs. Gate array signals originating outside the target gate array device require the

appropriate gate array input or output macro calls. When using gate array macros with EPLDs, the I/O macros are implemented in terms of EPLD primitives. Note that when designs targeted for gate arrays are partitioned for multiple EPLDs, many internal gate array signals are transformed into EPLD input and output signals. These signals must be supported by INPUT and OUTPUT macro calls.

## Macro Expansion

The Macro Expander identifies and expands each macro call in an ADF with the corresponding macro definition from macro libraries (the TTL library in the case of the 74138). The Macro Expander searches libraries in the following order and in the directories listed:

- **MACRO.LIB**—first in the current directory, then in other directories specified by the DOS "PATH" variable.
- **user libraries (filename.LIB)**—names for user libraries are specified in the "IPLS" environment variable. If a pathname and filename are both specified (SET IPLS=C:\MACLIB\USR1.LIB;), the path is treated as an absolute path. If a filename alone is specified (set IPLS=USR1.LIB;), the Macro Expander searches for that library in the directories specified by the "PATH" variable. (IPLS can be set in an AUTOEXEC.BAT file.)
- **TTL macro library (TTL.LIB)**—first in the current directory, then in other directories specified by the DOS "PATH" variable.

```

YOUR NAME
YOUR COMPANY
DATE
1
A
5C060
One-of-Eight Decoder

OPTIONS: TURBO=OFF
PART: 5C060
INPUTS: A,B,C,G2A,G2B,G1
OUTPUTS: Y7,Y6,Y5,Y4,Y3,Y2,Y1,Y0

NETWORK:

INPUT(A,A)
INPUT(B,B)
INPUT(C,C)
INPUT(G2A,G2A)
INPUT(G2B,G2B)
INPUT(G1,G1)
OUTPUT(Y7,Y7)
OUTPUT(Y6,Y6)
OUTPUT(Y5,Y5)
OUTPUT(Y4,Y4)
OUTPUT(Y3,Y3)
OUTPUT(Y2,Y2)
OUTPUT(Y1,Y1)
OUTPUT(Y0,Y0)
74138(A,B,C,G2A,G2B,G1,Y7,GND,Y6,Y5,Y4,Y3,Y2,Y1,Y0,VCC)

END$

```

292039-4

Figure 3. ADF File Calling the 74138 Macro

- EPLD Custom macro library (EPLDMAC.LIB)—first in the current directory, then in other directories specified by the DOS "PATH" variable.
- reserved library (INTEL.LIB)—first in the current directory, then in other directories specified by the DOS "PATH" variable.

Only the first occurrence of a macro is used. The names TTL.LIB, EPLDMAC.LIB, and INTEL.LIB are reserved by Intel. They may not be used for user libraries and may not be specified in the "IPLS" variable. The "IPLS" variable can contain more than one library name. Each library can have an absolute path or can rely on the "PATH" variable to determine the search path.

The Macro Expander uses the ADF Network and Equation entries from the macro libraries and assigns the appropriate primitives for INPUT and OUTPUT calls. INP primitives are assigned to replace the INPUT macro calls. The OUTPUT calls are assigned primitives with output pins and output enables are supplied where needed.

Combination of primitives is automatically performed when needed. For example, when a feedback primitive such as a NORF feeds an output primitive such as a RONF, the Macro Expander combines the two primitives into a RORF. Combination of primitives conserves resources and results in the shortest possible delay path through the device.

During macro expansion, unused nodes are eliminated. For example, the VCC and GND nodes that correspond to TTL power and ground pins are eliminated. If an input node is not connected to a node in the ADF, the default value for that node is assigned from the

```

NETWORK :
% INPUT (A, A) %
%^% A=INP (A)
% INPUT (B, B) %
%^% B=INP (B)
% INPUT (C, C) %
%^% C=INP (C)
% INPUT (G2A, G2A) %
%^% G2A=INP (G2A)
% INPUT (G2B, G2B) %
%^% G2B=INP (G2B)
% INPUT (G1, G1) %
%^% G1=INP (G1)
% OUTPUT (Y7, Y7) %
%^% Y7=CONF (Y7, VCC)
% OUTPUT (Y6, Y6) %
%^% Y6=CONF (Y6, VCC)
% OUTPUT (Y5, Y5) %
%^% Y5=CONF (Y5, VCC)
% OUTPUT (Y4, Y4) %
%^% Y4=CONF (Y4, VCC)
% OUTPUT (Y3, Y3) %
%^% Y3=CONF (Y3, VCC)
% OUTPUT (Y2, Y2) %
%^% Y2=CONF (Y2, VCC)
% OUTPUT (Y1, Y1) %
%^% Y1=CONF (Y1, VCC)
% OUTPUT (Y0, Y0) %
%^% Y0=CONF (Y0, VCC)
% 74138 (A, B, C, G2A, G2B, G1, Y7, GND, Y6, Y5, Y4, Y3, Y2, Y1, Y0, VCC) %

EQUATIONS :
%^% Y0=! ( !A*!B*!C*!G2A*!G2B*G1 ) ;
%^% Y1=! ( A*!B*!C*!G2A*!G2B*G1 ) ;
%^% Y2=! ( !A*B*!C*!G2A*!G2B*G1 ) ;
%^% Y3=! ( A*B*!C*!G2A*!G2B*G1 ) ;
%^% Y4=! ( !A*!B*C*!G2A*!G2B*G1 ) ;
%^% Y5=! ( A*!B*C*!G2A*!G2B*G1 ) ;
%^% Y6=! ( !A*B*C*!G2A*!G2B*G1 ) ;
%^% Y7=! ( A*B*C*!G2A*!G2B*G1 ) ;
    
```

292039-5

Figure 4. Network and Equations for 74138.SDF

DEFAULT: section of the macro file. Note, however, that the default value for each input in the macro file may be the value that disables the input or, for data inputs, is usually a logic 0. To be certain of the level used, specify a "VCC" or "GND" in the macro call for unused inputs.

The INPUT and OUTPUT calls and the original macro call are "commented out" by surrounding them with percent (%) signs. The %% string is placed at the start of lines where primitives are created by the Macro Expander. The fully expanded file is written to the disk using the original filename and a .SDF extension. Figure 4 shows the Network and Equation sections for the 74138 SDF.

One final note with regard to compiling ADFs that use macros. Warning messages are typically encountered while compiling files that use macros. The most common message is "\*\*\*\*WARN-XLT-Node Missing Destination". This message is displayed as unused nodes from a macro are deleted. For example, if a macro using a NOCF primitive is combined with a CONF and the original feedback is not needed, the warning is displayed as the feedback is deleted.

## Multiple Macro Calls

The Macro Expander allows use of more than one macro in ADFs. Each macro must have its own call, even when the same macro is used more than once.

For example, to implement two 74138s, each case or "instance" must have its own call:

```
74138 (A, B, C, G2A, G2B, G1, Y7, GND, Y6, Y5,
Y4, Y3, Y2, Y1, YO, VCC)
```

```
74138 (A, B, C, G3A, G3B, G1, YF, GND, YE, YD,
YC, YB, YA, Y9, Y8, VCC)
```

In this example, many of the inputs are routed to both devices. The Macro Expander automatically generates internal nodes for each instance of the macro. Each node is assigned a unique number based on the position of the macro in the Network section (i.e., . . 0140, . . 0141, etc. for nodes connecting to the 14th primitive in the Network section).

For traceability, you can define your own instance names for nodes of different macros by including the instance name in a comment immediately following the macro call. For example, to call two 74161 macros, one as Shift Register A and the other as Shift Register B, enter the calls as follows:

```
74161 (CLR, CK, A, B, C, D, ENP, , LD, ENT, QD,
QC, QB, QA, RDL, ) % SFA %
```

```
74161 (CLR, CK, E, F, G, H, ENP, , LD, ENT, QH,
QG, QF, QE, RC2, ) % SFB %
```

The Macro Expander uses the first three ASCII characters after the first percent sign (%), except for white space, to create instance numbers. For example, internal nodes for the first three signals of each macro call will be:

```
..SFAN1, ..SFAN2, ..SFAN3,
```

```
..SFBN1, ..SFBN2, ..SFBN3
```

where SFA/SFB are the user-defined instance names and N1, N2, N3 are the node numbers associated with each instance. For cases where no internal nodes numbers are generated, the Macro Expander simply ignores the instance name.

Outputs from one macro call can be used as inputs for other calls, as follows:

```
74138 (A, B, C, G2A, G2B, G1, Y7, GND, Y6, Y5,
Y4, Y3, Y2, Y1, YO, VCC)
```

```
74138 (A, B, C, Y7, G3B, G1, YF, GND, YE, YD, YC,
YB, YA, Y9, Y8, VCC)
```

Here the Y7 output from the first decoder feeds an enable input of the second decoder.

Different macros are connected in the same manner. For example, the following macro calls connect the outputs from a 74138 decoder to the inputs of 74175 latches:

```
74138 (A, B, C, G2A, G2B, G1, Y7, GND, Y6, Y5,
Y4, Y3, Y2, Y1, YO, VCC)
```

```
74175 (CLR, 0Q, n0Q, YO, Y1, n1Q, 1Q, GND, CLK,
2Q, n2Q, Y2, Y3, n3Q, 3Q, VCC)
```

```
74175 (CLR, 4Q, n4Q, Y4, Y5, n5Q, 5Q, GND, CLK,
6Q, n6Q, Y6, Y7, n7Q, 7Q, VCC)
```

Each decoder output is routed to a 74175 input. The 74175 macro produces both true and complement latched outputs.

## Guidelines/Pitfalls

The following paragraphs discuss some general guidelines for using macros:

- Because the Macro Expander supports only one level of hierarchy, there is a tendency for p-terms to multiply quickly when several macros are connected together. In many cases, the total number of p-terms exceeds the capacity of the target EPLD. One method of avoiding problems with excessive p-terms is to route the outputs from a macro function through EPLD macrocells and use the feedbacks from the macrocells as inputs to the subsequent macro functions. This partitioning of functions trades off device resources for a lower p-term count.

- Implementation of some TTL macros requires primitives that are not supported on all devices. The .DOC file for a device notes any device dependency. In many cases, a modification to the basic TTL functions results in device independence. For example, a NOCF, which is not supported on all EPLDs, can be changed to a COIF, which is supported on all devices.
- Some macros use primitives that specify an output pin (COIF, CONF, RORF, etc.). These primitives must be supported with a signal name in the OUTPUTS: declaration and by an OUTPUT call in the Network Section of the ADF. Failure to provide this support causes the following error message during compilation:

\*\*\*ERR-XLT-undeclared output name

If you encounter this error, check the macro file for output primitives that require ADF support.

### Macro Usage Summary

ADF macro calls must observe the following guidelines:

- Macros are called from the Network Section of an ADF.
- The name in the call must match the name in the macro file (e.g., 74138 = 74138).
- All input and output pins on the target device must have both: (1) a corresponding signal name in the INPUTS: or OUTPUTS: declaration, and (2) a corresponding INPUT or OUTPUT macro call in the Network section. It is recommended that the same node name be used on both sides of each INPUT and OUTPUT macro call. This is required when macros containing CONFs are used. (EPLD INP and CONF primitives may also be used).
- All INPUT and OUTPUT calls in the Network section must precede any other macro call.
- Node connections within an ADF are made based on the names of the nodes.
- Connections between the macro call and macro files are based on the position of signal names in the call. Therefore, the sequence of inputs and outputs in a macro call must match the sequence of inputs and outputs in the corresponding macro file.

### EXAMPLE 1: TTL MACROS

This section provides an example design using TTL macros.

### Circuit

The design is a two-stage decoder using a 74138 macro and two 74139 macros. Figure 5 shows the schematic for the circuit. Each 74139 macro represents one half of a TTL 74139 device. Note that two of the outputs from the 74138 are routed back to enable the two 74139 decoders.

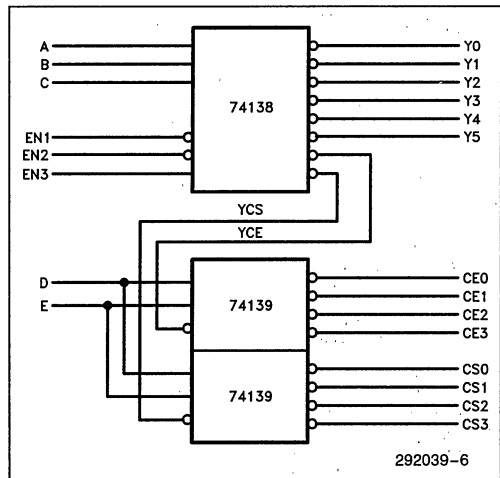


Figure 5. Schematic Diagram for Two-Stage Decoder

Figure 6 shows the ADF file containing the macro calls that implement the circuit. The two internal feedback signals (YCS and YCE) do not show up in the INPUTS: or OUTPUTS: declarations and are not represented by INPUT or OUTPUT calls in the Network section. The sequence of signals in the INPUTS: and OUTPUTS: declarations of the ADF is not important.

In the NETWORK: section, however, order is important. INPUT and OUTPUT calls must be listed before any other macro calls. This is a requirement of the Macro Expander. The sequence of signals within the ADF macro call is **critical**, as the Macro Expander automatically assigns macro call signals to macro file signals based on position.

Internal connections between macros are established by assigning the same name to the respective signals. For example, YCS in the 74138 macro call in Figure 7 represents the nY6 output from the 74138, while YCS in the 74139 macro call represents the 1G input to one 74139 decoder. Use of the same name establishes the connection. In the same manner, use of the signal name YCE connects the nY7 output from the 74138 to the 1G input of the second 74139.



```

DANIEL E. SMITH
INTEL CORPORATION
2/27/87
1
A
5CO90
TWO-STAGE DECODER

OPTIONS: TURBO=OFF
PART: 5CO90
INPUTS: A,B,C,D,E,EN1,EN2,EN3
OUTPUTS: Y0,Y1,Y2,Y3,Y4,Y5,CS0,CS1,CS2,CS3,CE0,CE1,CE2,CE3

NETWORK:

INPUT (A,A)
INPUT (B,B)
INPUT (C,C)
INPUT (D,D)
INPUT (E,E)
INPUT (EN1,EN1)
INPUT (EN2,EN2)
INPUT (EN3,EN3)
OUTPUT (Y0,Y0)
OUTPUT (Y1,Y1)
OUTPUT (Y2,Y2)
OUTPUT (Y3,Y3)
OUTPUT (Y4,Y4)
OUTPUT (Y5,Y5)
OUTPUT (CS0,CS0)
OUTPUT (CS1,CS1)
OUTPUT (CS2,CS2)
OUTPUT (CS3,CS3)
OUTPUT (CE0,CE0)
OUTPUT (CE1,CE1)
OUTPUT (CE2,CE2)
OUTPUT (CE3,CE3)
74138(A,B,C,EN1,EN2,EN3,YCS,GND,YCE,Y5,Y4,Y3,Y2,Y1,Y0,VCC)
74139(YCS,D,E,CS0,CS1,CS2,CS3,GND,VCC)
74139(YCE,D,E,CE0,CE1,CE2,CE3,GND,VCC)

ENDS

```

292039-7

Figure 6. ADF File for Two-Stage Decoder Using TTL Macros

## Sample Session

This session assumes familiarity with the iPLS II Logic Optimizing Compiler (LOC). For detailed information on the LOC, refer to Chapter 4 of the *iPLS II User's Guide*, order number: 450196. Proceed as follows to implement the TTL macro design shown here:

1. Use a standard ASCII text editor to create the ADF shown in Figure 7 under the name DECODE.ADF.
2. Invoke the iPLS II Menu by entering:

```
IPLS <Enter>
```

3. Invoke the LOC from the Main Menu by pressing <F4>.

4. Answer the LOC prompts as follows:

```

Input Format?           <Enter>
File Name?             DECODE <Enter>
Minimization?         Y
Inversion Control?    N
LEF Analysis?         Y
Error Message File    <Enter>

```

The LOC then asks:

Do you wish to run under the above conditions [Y/N]?

Enter: Y

The LOC expands the macros and compiles the expanded file to produce a JEDEC programming file (DECODE.JED), a utilization report file (DECODE.RPT), a minimized equation file (DECODE.LEF), and an error message file (DECODE.ERR). For tracability, a file called DECODE.SDF is created to show the expanded form of the ADF output by the Macro Expander.

5. The LOC terminates execution with the following message:

LOC cycle successfully completed

You can examine the LEF file to see the minimized form of the design. The LEF shows the EPLD primitives used to implement the design. Macro calls are not shown. If you wish, you can also use LPS (Logic Programmer Software) to program a part.

## EXAMPLE 2: MIXING MACROS AND EPLD PRIMITIVES

This final example uses TTL macros together with standard EPLD primitives.

### Circuit

The example circuit here is the 74138 macro used in example 1 with two of the outputs routed through additional combinatorial logic and RDNF (Registered Output — No Feedback) primitives. Figure 7 shows the

circuit. CS2 and CS3 are qualified by two additional inputs (RD\* and WR\*) to set or clear two latches. This is a configuration commonly used in microcomputer systems, where control signals are set and reset based on the address and command signals but not on a data value. A read to the port decoded by CS2 sets output LCS2 (Latched CS2) high. A write to that same port clears LCS2 low.

Figure 8 shows the ADF that implements the example circuit. This is the same ADF used in Figure 6, with the addition of several primitives and equations. The data inputs to both latches are tied to VCC. When RD\* and the chip enable are both low, the respective clock signal goes low. As RD\* or chip enable go high, the rising edge of the clock signal triggers the register, driving the output high.

Note that many Intel EPLDs do not support multiple product terms for register clocks. Therefore, the clock buffer primitive is driven by a macrocell configured as a COIF (Combinatorial Output—Input Feedback). Control signals (Clear and Preset) for many EPLDs also support only one product term. In this case, however, the NOR gate driving the clear input to the RDNFs can be minimized to a single p-term. Thus a low on WR\* and chip enable clears the respective latch to logic 0. (The intermediate macrocell for the Read function can be omitted for EPLDs that support two p-terms on register clocks.)

The connections between the TTL macros and the EPLD primitive are made by assigning the appropriate names to the input and output nodes. The CS2 and CS3 signals from the first example are no longer outputs, but are simply inputs to equations that feed the LCS2 and LCS3 RDNF primitives.

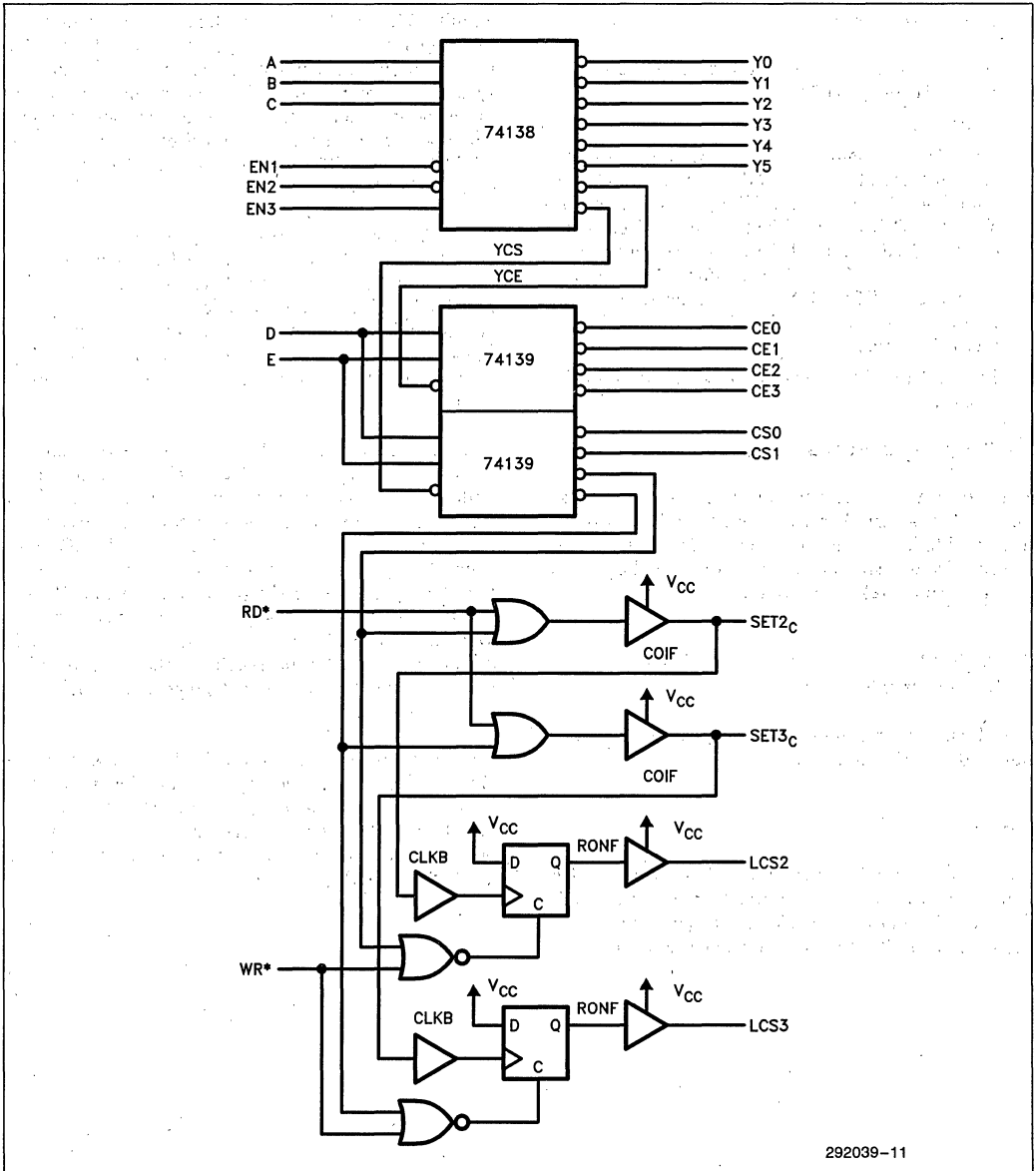


Figure 7. Schematic of Decoder Circuit with Latched Outputs

```

DANIEL E. SMITH
INTEL CORPORATION
2/27/87
1
A
5C090
DECODER WITH TWO LATCHED OUTPUTS

OPTIONS:  TURBO=OFF
PART:      5C090
INPUTS:   A, B, C, D, E, EN1, EN2, EN3, RD*, WR*
OUTPUTS:  SET2c, SET3c, Y0, Y1, Y2, Y3, Y4, Y5, CS0, CS1, LCS2, LCS3, CE0, CE1, CE2, CE3

NETWORK:

INPUT (A, A)
INPUT (B, B)
INPUT (C, C)
INPUT (D, D)
INPUT (E, E)
INPUT (EN1, EN1)
INPUT (EN2, EN2)
INPUT (EN3, EN3)
OUTPUT (Y0, Y0)
OUTPUT (Y1, Y1)
OUTPUT (Y2, Y2)
OUTPUT (Y3, Y3)
OUTPUT (Y4, Y4)
OUTPUT (Y5, Y5)
OUTPUT (CS0, CS0)
OUTPUT (CS1, CS1)
OUTPUT (CE0, CE0)
OUTPUT (CE1, CE1)
OUTPUT (CE2, CE2)
OUTPUT (CE3, CE3)
74138(A, B, C, EN1, EN2, EN3, YCS, GND, YCE, Y5, Y4, Y3, Y2, Y1, Y0, VCC)
74139(YCS, D, E, CS0, CS1, CS2, CS3, GND, VCC)
74139(YCE, D, E, CE0, CE1, CE2, CE3, GND, VCC)
RD = INP(RD*)
WR = INP(WR*)
LCS2 = RONF(VCC, SET2, CLR2, GND, VCC)
LCS3 = RONF(VCC, SET3, CLR3, GND, VCC)
SET2 = CLKB(SET2c)
SET3 = CLKB(SET3c)
SET2c, SET2c = CO1F(ST2, VCC)
SET3c, SET3c = CO1F(ST3, VCC)

EQUATIONS:

ST2 = RD + CS2;
CLR2 = /(WR + CS2);
ST3 = RD + CS3;
CLR3 = /(WR + CS3);

END$

```

292039-12

Figure 8. ADF File for Decoder with Latched Outputs

## Sample Session

To implement this ADF in an actual session, follow the steps described for Example 1, substituting the name LDECODE for DECODE. iPLS II produces a JEDEC programming file (LDECODE.JED), a utilization re-

port file (LDECODE.RPT), a minimized equation file (LDECODE.LEF), and an error message file (LDECODE.ERR). For traceability, a file called LDECODE.SDF is created to show the expanded form of the ADF output by the Macro Expander.



**APPLICATION  
NOTE**

**AP-312**

October 1988

**Creating Macros  
for EPLD Designs**

**DANIEL E. SMITH**  
PROGRAMMABLE LOGIC APPLICATIONS  
INTEL CORPORATION

Order Number: 292040-002

## INTRODUCTION

The iPLS II (Intel Programmable Logic Software II) Logic Optimizing Compiler includes a Macro Expander that supports the use of macros in EPLD designs. These macros can include TTL and EPLD custom macros available from Intel, or proprietary macros developed by a user. This application note shows how to create user-defined macros and how to build macro libraries with Intel's Macro Librarian, an optional software package for use with iPLS II. A design example also shows creation of a user-defined macro and its use in an ADF (Advanced Design File). Detailed information on using the TTL Macros in iPLS II ADFs are described in a companion application note, AP-311 "Using Macros in EPLD Designs", Order Number: 292039. This application note concentrates on creating macros; it assumes that you have read and understood the discussion on using macros in AP-311.

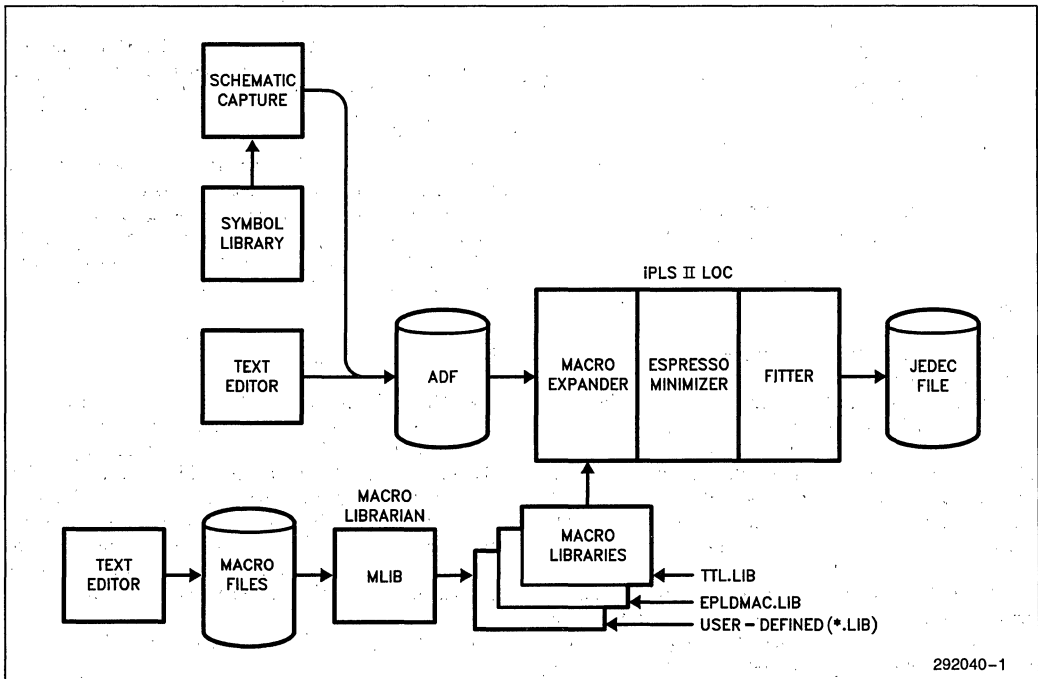
## OVERVIEW

iPLS II allows designers to include macro calls in design files to implement common circuit functions. Macros calls are subsequently expanded by the LOC (Logic Optimizing Compiler) into the ADF network and/or equation entries required to perform the desired functions. Macros can be connected together or used in conjunction with standard iPLS II EPLD primitives.

By following the macro file format described in this note, users can also create their own proprietary macros with an ASCII text editor. These macro files can then be stored in user-defined libraries by using Intel's Macro Librarian software. User-defined macros can be called from ADFs created by a text editor or by schematic capture software that supports user-defined symbols and that outputs in ADF format. User-defined macros can optimize development of EPLD designs by modularizing the design process and by allowing the design process to proceed at a higher level than with EPLD primitives alone. iPLS II support for user-defined macros (see in Figure 1) includes the following:

- MLIB, the optional iPLS II Macro Librarian for creating macro libraries from individual user-defined macro files.
- a Macro Expander in the LOC that expands macro calls in ADFs with the contents of the corresponding macros from libraries.

This application note describes how to create macro files, store them in libraries with MLIB, and shows how to call them from ADFs created by a text editor. *For information on creating user-defined macro symbols with schematic capture packages, refer to the appropriate manual for the schematic capture package you are using.* SCHEMA II-PLD available from Intel supports user-defined symbols and outputs in ADF format.



292040-1

Figure 1. Macro Support for iPLS II

(SCHEMA II-PLD is based on SCHEMA II from Omaton, Inc. The Intel EPLD Design Manager, also available from Intel, allows existing SCHEMA II users to design with EPLDs and macros.)

## MACRO FILES

This section describes iPLS II macro files. **User-defined macro files must follow the guidelines presented here to be successfully processed by the Macro Librarian (MLIB) and expanded by the iPLS II LOC Macro Expander.**

Macro filenames follow DOS conventions. It is recommended that macro filenames end with the extension .DEV, which is the default for MLIB. Only one macro can be contained in a macro file. Macro files are comprised of three sections:

- Header
- Network Section
- Equation Section

All macro files must end with the literal "ENDEF". Figure 2 shows a sample macro file for a proprietary part (16207), a "black box" containing random logic.

```

16207 (A,B,C,D,E,F,U,V,W,X,Y,Z)
DEFAULT: (GND,GND,GND,VCC,VCC,VCC, , , , , )

EQUATIONS:
U = /(A * B);
V = /( /E * A * B);
W = /(D * C * A * /E);
X = /( /D * E);
Y = /(F * D * A);
Z = F * /E;

ENDEF
292040-2

```

Figure 2. Sample Macro File for "Black Box" (16207.DEV)

### Header

Headers for macro files contain two lines. The first line includes the name of the macro function and a list of inputs and outputs for the macro. The second line contains defaults for the device.

The name of the macro can be a device number (16207, 83546, etc.), function name (ADDRCNT, CMDLO, etc.), or any name up to eight characters long. No spaces or comments precede the name.

Inputs and Outputs follow immediately after the macro name and are enclosed in parentheses. I/O signal names may be up to eight characters long, but may not contain pin numbers. For user-defined macros, signals may be listed in any order desired. For example, any of the following entries are legal:

16207 (A,B,C,D,E,F,U,V,W,X,Y,Z)

16207 (B,D,A,R,Z,U,W,C,F,X,E,Y)

16207 (Z,Y,X,W,V,U,F,E,D,C,B,A)

Note that this first line of the header forms the template used to call the Macro from the ADF. The Macro Expander connects ADF nodes in the macro call to I/O signals in the macro file on the basis of position, not on the basis of node name.

The second line in the header specifies defaults for inputs (VCC or GND) in cases where those signals are left unconnected. The DEFAULT: line must be included in the macro definition file, even when no defaults are used in the ADF. The keyword DEFAULT: is the first entry in this line. The default values for all signals follow immediately and are enclosed in parentheses. Input defaults may be VCC or GND. The position of the default value corresponds to the signal listed in the previous line.

Defaults for outputs are blank, but a comma (,) must be present (place holder) for each output signal except the last. For example, the 16207 black box contains six inputs (A through F) and six outputs (U through Z). The first two lines for this macro might be:

```

16207 (A,B,C,D,E,F,U,V,W,X,Y,Z)
DEFAULT: (GND,GND,GND,VCC,VCC,VCC,,,,,)

```

Defaults for inputs A through C are GND; defaults for inputs D through F are VCC. Defaults for the outputs are not specified, but the comma denotes the positions for those signals.

Defaults should be chosen with care. Clears, Presets, Loads, etc. should be disabled in most cases. Enables should be enabled. Input defaults can also be left blank as long as those inputs are connected to nodes in the ADF that calls the macro, but it is recommended that they be specified in the macro file.

### Network Section

The NETWORK: section lists the EPLD primitives used to implement the desired functions. The Network Section follows ADF syntax rules. As far as possible, the macros should be implemented in equations to eliminate concern about feedbacks and output enables. In the case of a circuit that requires macrocell registers, the feedback-only form of the primitive should be used so that the Macro Expander can make the correct pin connections. The following example shows this:

```
OUT1 = NORF (INd,CLK,GND,GND)
```

During processing, the Macro Expander connects the feedback to an output (if necessary) and supplies the required output enable node name. The Macro Expander also eliminates unneeded Network and Equations entries if they are not used by an ADF.

If no network entries are required (i.e., a macro implemented entirely in equations), the entire Network section may be omitted, including the keyword NETWORK:. In many cases, equations alone can implement the desired functions.

### Equations Section

The EQUATIONS: section lists the Boolean equations for the desired functions and follows ADF syntax rules, with one exception; intermediate equations are not permitted in macro files. If no equation entries are required (i.e., a macro implemented entirely in the Network Section), the entire Equation section may be omitted, including the keyword EQUATIONS:.

### Comments and White Space

Comments can be placed anywhere in a macro file except before the name and signals on the first line. Comments must be enclosed in percent signs, as follows:

`% THIS IS A SAMPLE COMMENT %`

White space can appear on any line except the first two lines.

## MACRO LIBRARIAN

The Macro Librarian (MLIB) is an optional software package that combines individual macro files into macro libraries. These libraries are in turn used by the LOC Macro Expander. MLIB can be invoked from the command line, from command files, or from a combination of both. Figure 3 shows a block diagram of the Macro Librarian.

Syntax for MLIB command lines is as follows:

`MLIB [-options] [@cmdfile] [file1 file2 ...]`  
 <Enter>

- d directory. Displays directory information for the library being created.
- v verbose. Print status during processing. When not specified, status messages are suppressed.
- l lib list. Lists the contents of existing macro library to console. This option may not be used while building a library.
- o lib name of the target macro library. MACRO.LIB is the default when no name is specified. TTL.LIB, EPLDMAC.LIB, and INTEL.LIB are reserved for Intel libraries and may not be used.
- s string include version stamp in macro library. The version string can be up to 7 characters long. "V1.00" is the default stamp.

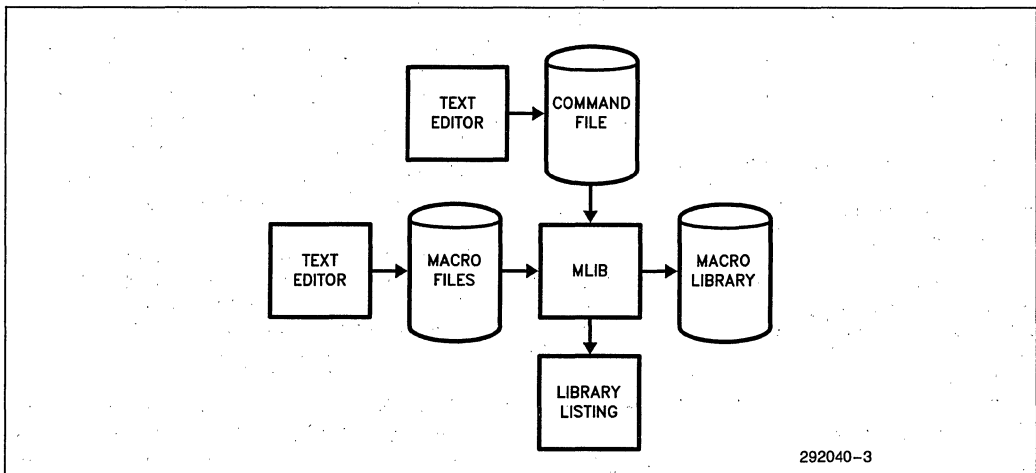


Figure 3. Macro Librarian Block Diagram



-c string include copyright string in macro library. The copyright string can be up to 61 characters long and, if blanks are used, must be contained in quotation marks, for example, "texta textb".

@cmdfile name of command file. The command file can include options and macro filenames. The @ symbol must precede the filename.

file1 ... name of device files to be included in the macro library. Separate files by spaces.

For example, the following command line:

```
MLIB -v -s 2.00 -o USER.LIB @USERLIST <Enter>
```

creates a library called USER.LIB that includes all the individual macro files contained in the command file USERLIST. MLIB displays status messages as it processes the macro files in USERLIST (-v). The library is created as version 2.00 (-s).

Macro library filenames follows DOS conventions and should end with the extension .LIB to be recognized by the Macro Expander. TTL.LIB, EPLDMAC.LIB, and INTEL.LIB are reserved and may not be used.

USERLIST is the name of the command file and must be preceded by the @ symbol. The command file is simply an ASCII text file that can be modified to contain any number of macros desired. MLIB processes the entire list of macros on each invocation. To add a new macro to an existing library, add the name of the macro to USERLIST, and create the new library by entering the command line shown above. Command file names follow DOS conventions. MLIB supplies a .DEV extension if no extension is specified. MLIB searches first in the current directory, then along the DEV environment variable, and finally along the PATH environment variable for the files.

In order to connect inut and output primitives, the files INPUT.DEV and OUTPUT.DEV must be included in at least one of the libraries. These files are contained in the TTL macro library.

Figure 4 shows a sample MLIB command file that includes options, the library name, and the names of seven macro files to be included in the library in addition to the INPUT and OUTPUT macros. The format of the command file is free form. Note that comments can be included in the command file and must be contained within percent (%) signs.

Note that the -l option cannot be included in an MLIB command file; it can only appear on the command line. The -l option lists the contents of existing libraries; it does not list library contents while building a library.

```
-o PROJA.LIB % macro library name %
-v
-s V1.50 % version number %
-c "Copyright (C) Date, Your Company, Your Name"
  % copyright information %
-d % display directory %

% Include the following macros %

INPUT.DEV   OUTPUT.DEV   7408.DEV
7487.DEV   74138.DEV   74139.DEV
74151.DEV  74157.DEV   74251.DEV

292040-4
```

Figure 4. Sample Command File for MLIB

The command line to process the file shown in Figure 4 is as follows:

```
MLIB @SAMPLE <Enter>
```

where SAMPLE is the name of the command file.

To list the contents of PROJA.LIB after creation, invoke MLIB as follows:

```
MLIB -l PROJA.LIB
```

This command line lists the macros in PROJA.LIB to the screen. The DOS file redirection capability can also be used to create a disk file listing the contents of macro libraries. For example:

```
MLIB -l PROJA.LIB > PROJA.DOC
```

## SAMPLE SESSION: COMMAND DECODER USING MACROS

Decoding logic is one common function implemented by programmable logic devices. The target circuit for this example is a device that decodes microprocessor command signals in selected address ranges. The target application and decoder requirements are as follows:

- The target application is a 16-bit microcomputer system with 1-Megabyte of memory and about two dozen I/O ports.
- The memory is divided into shared memory (lower 512K bytes) and local memory (upper 512K bytes). Shared memory resides off the processor board and requires active low memory command signals. Local memory resides on-board and requires active high memory command signals.
- I/O ports are also split between on-board devices requiring active high signals and off-board devices requiring active low signals. I/O devices between the address range F000-FFFFH are on-board; devices below that range (0000-EFFFH) are off-board.

- All interrupt requests are resolved by an on-board interrupt controller. Therefore, only an active high on-board interrupt acknowledge signal is needed.
- On-board control signals are always high or low, never three-stated. Off-board control signals are three-stated when not being used to execute a bus cycle. An external bus arbiter accepts a request signal from the command decoder and, after gaining

control of the bus, sends address enable and command enable signals back to the command decoder.

Figure 5 shows a block diagram of the application, including the target EPLD design. The three functional blocks to be included in the EPLD are highlighted (not shaded).

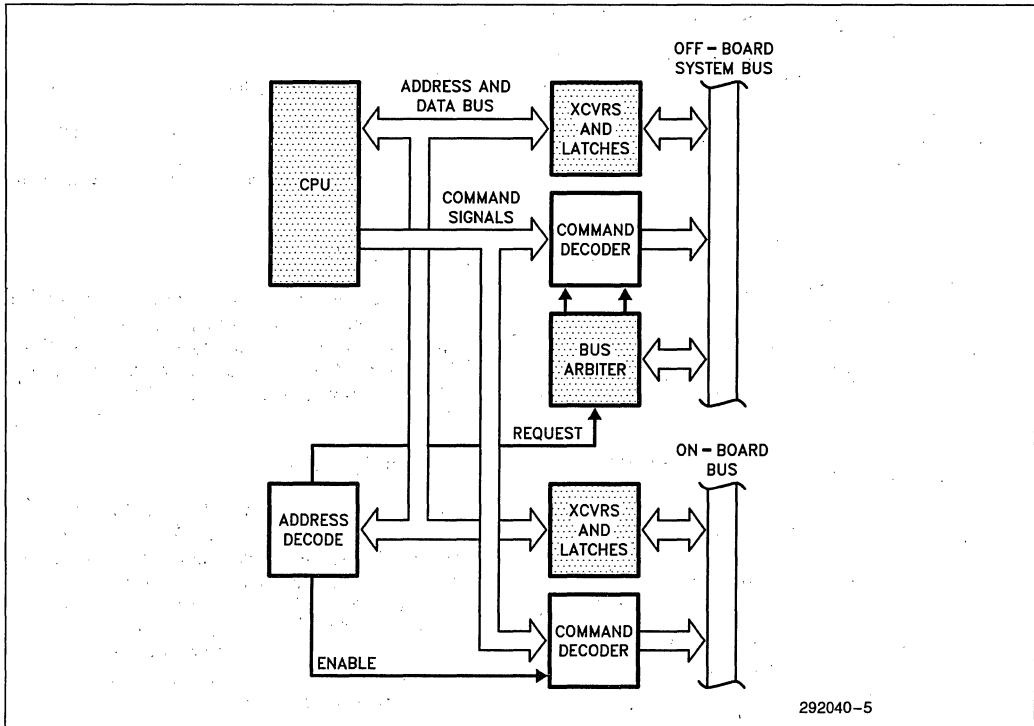


Figure 5. Block Diagram of Target Circuit and Application

### Creating the Macro

Figure 6 shows a schematic diagram for the active low command decoder implemented with OR gates (low inputs enable the outputs; high inputs disable the outputs). Figure 7 shows the macro file that implements the circuit (CMDLO.DEV). This file was created with an ASCII text editor. Used as is, it provides the active low outputs for the design. With inputs RD, WR, and INTAIN inverted, it also provides the active high outputs for the design. This design uses CONF primitives to implement the three-state outputs in the macro. As an alternative, equations alone could have been used with the CONFs included in the ADF.

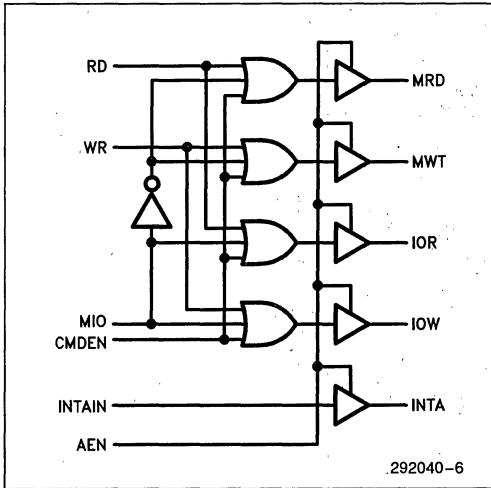


Figure 6. Schematic Diagram of Command Decoder

### Building the Library

Use your text editor to create an MLIB command file that includes CMDLO.DEV, INPUT.DEV, and OUTPUT.DEV. The following example shows a sample command file named MACLIST.

```
-v % show status %
-c "1987, AP-312 Sample Macro Library"
-o AP312.LIB
-d % show the list %

% include the following macros %

CMDLO.DEV INPUT.DEV OUTPUT.DEV
```

Invoke the Macro Librarian with the following command line:

```
MLIB @MACLIST
```

The Macro Librarian processes the three macro files and stores them in a user library named AP312.LIB. The library contains the copyright statement "1987, AP-312 Sample Macro Library". When processing is complete, MLIB returns control to DOS.

### Creating the ADF

Figure 8 shows a schematic diagram for the target circuit. Figure 9 shows the ADF for the circuit (COMCODE.ADF), which invokes both instances of the CMDLO macro and contains equations used to enable the decoders under the proper conditions. The ADF signal named ONBEN (On-Board Enable) enables the active high decoder). The AEN (Address Enable) input to the on-board decoder is left unconnected. The default (always enabled) will be used.

```
CMDLO (MIO, RD, WR, INTAIN, CMDEN, AEN, MRD, MWT, IOR, IOW, INTA)
DEFAULT: (GND, VCC, VCC, VCC, GND, GND, . . . .)
```

NETWORK:

```
MRD = CONF (MRDc, AEN)
MWT = CONF (MWTc, AEN)
IOR = CONF (IORc, AEN)
IOW = CONF (IOWc, AEN)
INTA = CONF (INTAIN, AEN)
```

EQUATIONS:

```
MRDc = /MIO + RD + CMDEN;
MWTc = /MIO + WR + CMDEN;
IORc = MIO + RD + CMDEN;
IOWc = MIO + WR + CMDEN;
```

ENDEF

292040-7

Figure 7. Macro File for Command Decoder (CMDLO.DEV)

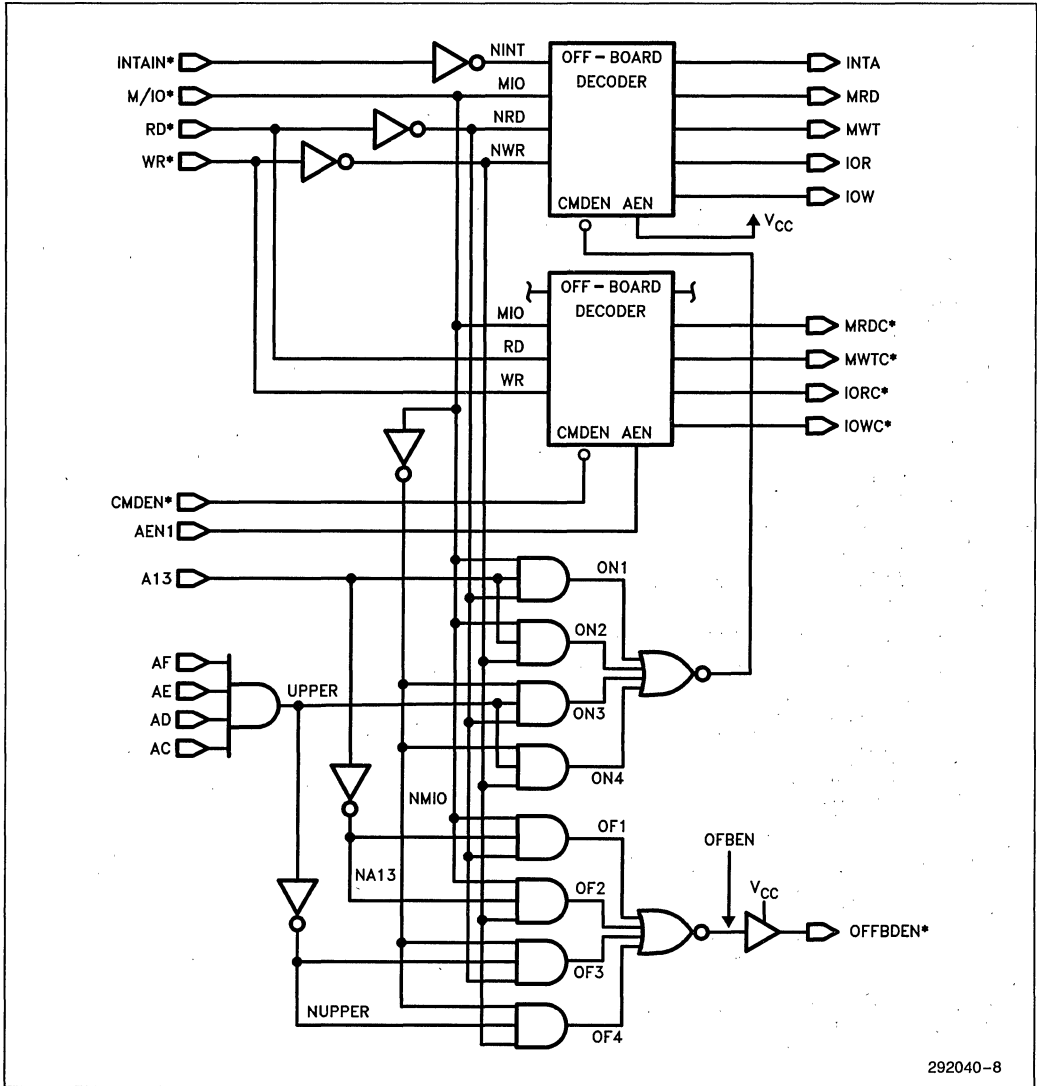


Figure 8. Schematic Diagram for COMCODE.ADF

```

DANIEL E. SMITH
INTEL CORPORATION
4/7/87
1
A
16209-001
COMMAND DECODER

OPTIONS:  TURBO=ON
PART: 5C090
INPUTS: MIO, RD, WR, INTAIN, CMDEN, AEN1, A13, AF, AE, AD, AC
OUTPUTS: MRD, MWT, IOR, IOW, INTA, MRDC, MWTC, IORC, IOWC, OFFBDEN

NETWORK:

INPUT(MIO,MIO)
INPUT(RD,RD)
INPUT(WR,WR)
INPUT(INTAIN,INTAIN)
INPUT(CMDEN,CMDEN)
INPUT(AEN1,AEN1)
INPUT(A13,A13)
INPUT(AF,AF)
INPUT(AE,AE)
INPUT(AD,AD)
INPUT(AC,AC)

OUTPUT(MRD,MRD)
OUTPUT(MWT,MWT)
OUTPUT(IOR,IOR)
OUTPUT(IOW,IOW)
OUTPUT(INTA,INTA)
OUTPUT(MRDC,MRDC)
OUTPUT(MWTC,MWTC)
OUTPUT(IORC,IORC)
OUTPUT(IOWC,IOWC)

CMDLO(MIO,RD,WR,,CMDEN,AEN1,MRDC,MWTC,IORC,IOWC,)% OFB %

CMDLO(MIO,NRD,NWR,NINT,ONBEN,VCC,MRD,MWT,IOR,IOW,INTA)% ONB %

OFFBDEN = CONF(OFBEN,VCC)
OFBEN = NOR(OF1,OF2,OF3,OF4)
ONBEN = NOR(ON1,ON2,ON3,ON4)
NRD = NOT(RD)
NWR = NOT(WR)
NINT = NOT(INTAIN)
NMIO = NOT(MIO)
NUPPER = NOT(UPPER)
NA13 = NOT(A13)

EQUATIONS:

UPPER = (AF * AE * AD * AC);
ON1 = (MIO * A13 * NRD);
ON2 = (MIO * A13 * NWR);
ON3 = (NMIO * UPPER * NRD);
ON4 = (NMIO * UPPER * NWR);
OF1 = (MIO * NA13 * NRD);
OF2 = (MIO * NA13 * NWR);
OF3 = (NMIO * NUPPER * NRD);
OF4 = (NMIO * NUPPER * NWR);

END$

```

292040-9

Figure 9. ADF for COMCODE.ADF

OFFBEN (Off-Board Enable) requests permission to access the off-board bus from the external bus arbiter. The bus arbiter enables the off-board decoder via AEN1 (Address Enable 1) and CMDEN (Command Enable). CMDEN allows the appropriate signal to go high or low, and AEN1 causes the outputs to independently enter or exit a high impedance state (three-state).

Note the same name is used for both nodes of each INPUT and OUTPUT macro call. Use of the same name ensures proper connection when the Macro Expander eliminates redundant primitives (for example, a CONF feeding another CONF).

## Compiling the Design

Proceed as follows to compile the ADF.

1. Include AP312.LIB in the IPLS environment variable. From the DOS command prompt, type:  
SET IPLS=C:\IPLSII\AP312.LIB; ... <Enter>  
For user-defined macro libraries that are regularly accessed, the IPLS variable can be set in an AUTOEXEC.BAT file.
2. Invoke the iPLS II Menu by entering:  
IPLS <Enter>
3. Invoke the LOC from the Main Menu by pressing <F4>.
4. Answer the LOC prompts as follows:  
Input Format? <Enter>  
File Name? COMCODE <Enter>  
Minimization? Y  
Inversion Control? N  
LEF Analysis? Y  
Error Message File COMCODE.ERR <Enter>

The LOC then asks:

Do you wish to run under the above conditions  
[Y/N]?

Enter: Y

The LOC expands the macros and compiles the expanded file to produce a JEDEC programming file (COMCODE.JED), a utilization report file (COMCODE.RPT), a minimized logic equation file (COMCODE.LEF) and an error message file (COMCODE.ERR). For traceability, a file called COMCODE.SDF is created to show the expanded form of the ADF output by the Macro Expander.

5. The LOC terminates execution with the following message:

LOC cycle successfully completed

You can examine the LEF file to see the minimized form of the design. The LEF shows the EPLD primitives used to implement the design. Macro calls are not shown in the LEF. If you wish, you can also use LPS (Logic Programmer Software) to program a part.



## Tools for Optimizing PLD Designs

Alan J. Coppola  
Tool Architect

*Intel Corporation*  
*M/S EY2-11*  
*5200 NE Elam Young Pkwy.*  
*Hillsboro, OR 97123*  
*(503)681-2177*

### Introduction:

The purpose of this paper is to describe a design methodology for Programmable Logic Devices (PLD's) and to survey current PLD optimization techniques.

#### 1. Perspective: Where do PLD's fit in?

The use of Programmable Logic Devices (PLD's) represents a middle ground in logic design. The two common approaches to logic implementation in today's market are Board Design methods (building a solution from a selection of pre-fabricated standard parts - TTL/SSI/MSI) and Custom/Semi-Custom design methods (fabricating a custom logic chip to solve the problem at hand, and then building a much simpler board).

With the Board Design approach, PCB's carry the fruit of a designer's labor to the customer. Many little black boxes and other electrical circuit components make up the brunt of a PCB's load. Many times there are large islands of functionality to be connected together via encoding/decoding and timing circuits. The islands of functionality (i.e. microprocessor, microcontroller, RAM, EPROM, transceiver, etc.) all have different protocols, and all speak different languages at different speeds.

Integrating the major devices of a board together involves much "glue" logic. The typical designer spends time and effort looking through a TTL parts catalog to find the best fit for a design based on functionality, performance and price. After a preliminary function-based board is laid out, modification passes are made based on the parts needed and their availability. Large designs increase the length and risk of this process. Even though most major CAD vendors are addressing the problem of board design, simulation, test and interface with the Custom/Semi-Custom arena, board design and manufacturing tools are rapidly becoming the primary practical obstacle to effective production of end-user systems. PLD's reduce the complexity of the end-user board, hence reducing the length and risk of the implementation and manufacturing process.

With the Custom/Semi-Custom design approach, the designer is free to address functionality directly. The designer has much flexibility in the functionality, speed and integration facets of logic design. Certainly, the number of parts on an end-user PCB can be greatly reduced by integrating most of a board's function into a few custom devices. The problems associated with all the flexibility lie in the physical design, modeling, and tools areas. Specifically, in the physical design area, problems include process specific bottlenecks, NRE charges and long lead times.

In the modeling area, extensive simulations must occur before and after the device is built, as each device is custom crafted. Finally, in the CAD tools area, a highly functional, but hard to use set of tools guide and control the whole process. The tools are the best in terms of functionality, but the worst in terms of cost and ease of use.

The job of ASIC vendors in the next ten years is to make the Custom/Semi-Custom problems disappear or become acceptable to the logic designer of the next generation. The facts are clear. Without advanced tools which automate much of the logic designer's work, the Custom/Semi-Custom approach only works for large scale, large volume or special purpose devices. Silicon compilers and other Custom/Semi-Custom design methodologies are working hard to overcome the inherent problems of this type of design.

The use of PLD's in a design is a compromise between the flexibility of a Custom/Semi-Custom design, and the standard Board design methodology.

The definition of PLD which I am using for the purposes of this paper is very general. A Programmable Logic Device is any device, which can be programmed by the user, to realize a chunk of combinatorial or sequential logic. A subset of the most popular, or newest types of PLD's are: PAL's, PLE's (Monolithic Memories), EPLD's (Intel, Altera), EEPLD's (Lattice), FPLA's, FPLS's (Signetics), LCA's (Xilinx), and ERASIC's (Exel). All except the last two are based on some form of two-level (AND/OR) registered array logic. I will mainly be concerned with two-level array logic devices.

The good points of designing with PLD's are:

1. The integration of many small chunks of TTL and SSI/MSI logic into a few PLDs. Essential for efficient use of resources.
2. An easier overall development cycle than the Custom/Semi-Custom route. Also easier than standard board TTL development cycle once the learning curve is passed.
3. Much cheaper than the Custom/Semi-Custom design method for all but large volume designs. Usually cheaper than standard board TTL design method.
4. The breadboard character and modifiability of PLDs makes them an excellent R&D and learning vehicle in the design environment.



5. PLD CAD development tools are available to convert standard TTL logic representations into the complicated fixed architectures of an individual device. The CAD tools automate this process, to a large degree, so that the user can use their own design techniques.

The bad points of designing with PLDs' are:

1. For large designs, using PLD's is not as functional or robust as using Custom/Semi-Custom logic.
2. The CAD tools available for PLD design are not as useful in automating the whole design process as those in the Custom/Semi-Custom arena. In fact, most of the tools are derived from those used in Custom/Semi-Custom design.
3. The speed and function constraints of the fixed device architectures can be inhibiting. For example, not all types of designs fit well into two-level array logic.

## 2. The PLD Development Environment

There are three pieces in a PLD development environment. First is the input part. The design specification needs to be entered in some form, such as a schematic, a finite state machine, or a high level language description. Second is the processing part. This must include some kind of compilation of the input into object code (JEDEC code), and usually also includes optimization of the design. Third is the output part. This includes the object code (JEDEC), test results/vectors, and statistics.

A PLD development environment has an underlying language for representing design specifications, which is usually more general than the intended devices. Often, the system provides means for accepting input in other forms, which then translates into the underlying language. We shall use the standard term "Hardware Description Language" (HDL) when referring to this language, as this is where these languages are heading in terms of complexity and future directions. Examples of PLD HDLs' are ABEL by Data/IO-Futurenet, CUPL by Assisted Technology, ADF by Intel(Altera), PALASM by Monolithic Memories, LOG/IC by Kontron, and AMAZE by Signetics.

There are four questions one can ask about a PLD development environment and it's HDL. The answers to these questions determine, for the user, which systems to use. The four questions are:

**Question 1:** Is it easy to learn and easy to use (User-Friendly) ?

**Question 2:** Is it generic enough to support the current applications and new devices yet to come ? (HDL expressiveness and functionality)

**Question 3:** Does the compiler use optimization techniques to produce JEDEC code ? (i.e. logic minimization)

**Question 4:** Are there alternate logic entry tools, like schematic capture and Finite State Machine entry, are there hooks for simulation and other design methodologies, like gate arrays ?

As an example, we answer the four questions for the HDL of the Intel Programmable Logic Development System (iPLDS)(Altera). The HDL of iPLDS is called Advanced Design File(ADF).

**Question 1:** Is it easy to learn and easy to use ?(User-Friendly)

**Answer:** The ADF language is made easier to learn for the novice user by two items:

### a. Graphical Interface Tools

**Logic Builder(LB):** A graphical netlist entry and syntax checking aid to the user entering designs which have already been written down on paper in a schematic fashion.

**Logic Programming(LP):** A graphical JEDEC file editor, which allows the user to modify the JEDEC file. More importantly, the tool allows the user to investigate and learn the device architecture via a user-oriented graphical interface, and then to program the part directly from this interface.

### b. Primitives:

A large("80) set of logic and I/O Macrocell primitives which capture all of the current standard ways to represent small chunks of memory and combinational logic. This is useful for the novice and occasional user, who doesn't have the time, or want to learn the abstractions involved in more generic HDLs'.

**Question 2:** Is it generic enough to support the current applications and new devices yet to come ? (HDL expressiveness and functionality)

**Answer:** Yes, it is generic enough to support those current and future architectures based on two-level registered logic, which are produced by Intel and Altera. The large number of primitives make the language unwieldy for support of new devices not falling into this realm. Finally, it supports only Intel and Altera devices.

**Question 3:** Does the compiler use optimization techniques to produce JEDEC code ? (i.e. logic minimization)

**Answer:** Yes, logic minimization, DeMorgan's inversion of outputs, and automatic fitting of resources and pins to the given device are supported in an integrated fashion.

**Question 4:** Are there alternate logic entry tools, like schematic capture and Finite State Machine entry, are there hooks for simulation and other design methodologies, like gate arrays ?

**Answer:** Alternate entry methods include schematic capture, FSM entry, and Graphical Netlist entry(LB). There are currently no tools for functional simulation, nor is there any way of interfacing to other design tools. Both topics are being considered for the future.

The third party HDLs', like ABEL, CUPL, and LOG/IC are, in general, harder to learn and use. They afford greater expressiveness and generality in addressing design problems. Because of the need to work with most devices on the market, these HDL's resemble more closely their high-power cousins in the Custom/Semi-Custom design arena. They have no automatic resource fitting and pin-assignment, but do have a robust set of integrated tools involving functional simulation, schematic capture and FSM entry available.

## 3. PLD Optimization Tools

Here we introduce and describe the current mix of optimization tools in the PLD development workshop. The goal of CAD optimization tools for PLD design is to speed up the design cycle by reducing designer time, and to cram larger designs into a fixed device. The optimization tools of a typical PLD development





system usually reside in the HDL compiler. The tools can be viewed as compiler optimization tools.

The four optimization tools of concern to us are:

1. **Logic Minimization** - A tool to reduce the complexity of the logic equations implementing a design.
2. **Finite State Machine(FSM) Compiler** - A tool which takes an FSM description and translates or compiles it into an HDL.
3. **DeMorgan's Inversion** - A tool which can reduce the amount of logic needed by inverting the sense of some of the output signals of the device.
4. **Fitting and Pin Assignment** - A tool which automatically fits the resources and pins which the user chooses not to.

We discuss each topic separately, and illustrate, by the use of the Dice Example, (Figures 1-3) most of these features.

#### Logic Minimization:

Logic minimization is currently the best optimization tool available for PLDs. Logic minimization for current PLD development systems is strictly a two-level logic tool, and totally replaces, for combinational logic purposes, the Karnaugh Map and Quine-McCluskey Algorithm methodology for finding the minimum size set of equations. For two-level logic(AND/OR), most compilers use a single or multiple output heuristic minimizer. University research and industrial experience show that the NP-complete problem of two-level logic minimization has been effectively solved for the size of problems currently being considered in practice. Currently, the most effective minimizers are Espresso, McBoole, and Presto-II. Espresso has been shown, in aggregate, to be within one-percent of the minimum answer on 104 test cases. On current PLD size problems, the test cases indicate that Espresso will find the minimum solution almost always [1, 2, 3, 4].

The user of minimization tools usually has a concern about the process. The minimization tools have the side-effect of producing a reduced equation set that doesn't always reflect the designer's thought process relative to the un-minimized equation set. This results in confusion in trying to understand the design from the reduced equation set. For the same reason, editing a JEDEC file to change a few bits of a design is error-prone and to be avoided. To independently verify the meaning of the design and the resulting minimized boolean equations, functional test vectors should be created and run through a functional simulator. This will catch design errors and give a truth-table verification of the design. For some HDLs, there are tools available that will automatically generate test vectors. Actually, due to the fixed architectures of PLDs, automatic test generation is much more feasible in the PLD arena than in the Custom/Semi-Custom area. Of course, a designer can choose to not have the equations minimized if they already fit into the target device before minimization, but in complicated designs, this is rarely the case, as the Dice Example at the end of this paper shows.

Future PLD tools will depend more and more on logic minimization. Just as a high-level language programmer looks

less and less at the object code produced by a compiler, the logic designer will not be concerned with the minimized code output of the PLD logic compiler.

#### FSM Compiler:

An FSM compiler is included in the list of PLD optimization tools because it allows for a compact representation of a sequential circuit. This leads to ways to introduce systematic optimization techniques like logic minimization and automatic state assignment at a transparent level for the user. Designing at the level of states and transitions results in a more compact HDL description for the same logic function, with fewer mistakes being made. FSM description also allows the user to change memory element types and state assignments in an error-free manner. FSM's are also well understood, theoretically, and are a ripe area for future tool development. Most logic texts present many hand tabular methods for optimizing FSM designs. These hand methods can be automated and extended to produce new tools for FSM-based design.

#### DeMorgan's Inversion:

DeMorgan's inversion, in AND/OR type PLD architectures, with inversion control in the I/O macrocells, refers to logically inverting an output signal phase in such a way that the number of p-terms realizing the complement function is less than the original function. This can save the user from an un-solvable p-term fitting problem due to too many p-terms when using one sense of an equation. For devices with single output macrocells, like PALs, and EPLDs, the complement of the single output equation is computed and then minimized. The sense of the equation with the least number of p-terms is then the one that is implemented in the device under programming.

#### Fitting and Pin Assignment:

The fitting and pin assignment problem refers to compiling a design file, and having the compiler automatically choose those device resources and pins that the user did not assign in the design file. In the past, device architectures have been simple enough and small enough so that fitting and pin assignment were not a problem for the user. Now, with increasing size, complexity, and non-homogeneity of the device architecture, a heuristic CAD tool, which is like an automatic place and route tool, is a necessity. If a device architecture is homogeneous with respect to structure and resources, fitting is not a problem, as there is no contention for resources or placement of those resources. Fitting is a problem when there are multiple clocks and types of clocks, multiple device sections (like quadrants), varying numbers of p-terms per quadrant, product term sharing and steering, input pins, I/O macrocells of varying types, or buried registers. The greater the number and size of the features, the greater the fitting problem. Without a tool to help, the designer must do the fitting by hand, leading to errors and not finding an allowable fit.

Some of the large scale devices that exhibit these problems are Intel's (Altera's) 5C121 (EP1210) and 5C180 (EP1800). The fitting and automatic pin assignment tools of iPLDs relieve the user from having to deal with this problem.

### 3. Future PLD Optimization Tools

This section describes new directions for PLD development systems optimization tools. Optimization tools must be near transparent to the user to get universal acceptance. Of the four optimization tools mentioned above, all but the FSM compiler tool satisfy that criterion.

There are basically two types of optimization tools which will appear in the PLD arena. The first type are tools which are ported from, or interfaced to the Custom/Semi-Custom environment. The current logic description and synthesis tools of silicon compilers and Custom/Semi-Custom CAD tools fit into this classification. The second type are new tools which will address the architecture-specific optimization issues. The tools in this group will use methods based on logic optimization and expert-system techniques. These two methodologies will be applied to taking abstract specifications and realizing them automatically into multiple devices, or in taking multiple abstract specifications and realizing them in one device.

#### Portation of Custom/Semi-Custom Tools:

Available ideas ready for porting to the PLD environment down the HDL path include implementing a subset of VHDL (VHSIC Hardware Description Language) [5], and having the compiler produce an EDIF (Electronic Design Interchange Format) [6] intermediate format. In this way, interfacing with other toolboxes of any type will be easier. New device support will also be easier, given the generic nature of VHDL. Using VHDL would also standardize an HDL, and allow designers to learn one HDL which will last for a long time. Also, PLD tools which interface with the Custom/Semi-Custom toolset involving board design, testing and manufacturing is needed now, and is being addressed by the major CAD vendors. Standardization, like VHDL and EDIF will, eventually, lower the cost of these interfaces.

The new logic minimization algorithms, like Espresso, and new state assignment tools, like KISS [7] and STASH [8] can be used in the PLD environment. The algorithms and methods of tools involving placement and routing can be applied to the fitting/pin assignment problem. On the logic synthesis side, new tools which combine expert-systems with multi-level logic optimization can be applied to PLD devices which allow multi-level logic to be easily implemented.

The key point, irregardless of the actual tools from the Custom/Semi-Custom arena which are productized is that the user have an essentially transparent view of any new optimization tools.

Once a PLD is manufactured, the functionality cannot be changed. This fact leads to the belief that tools can be created which map logic, which is too big or too slow, into multiple devices by doing automatic logic partitioning. The converse problem of fitting multiple chunks of communicating logic into one device may also be addressed. Tools to fit multiple state machines into one device, or to partition a schematic or FSM into two or more devices is a first step. For example, the Dice Example (Figures 1-3) has three small state machines, which are integrated into one device and design file. Expert-systems can capture the rules for partitioning, and the database of all allowable devices, while optimization techniques can make the expert-systems work as well as, or better than a logic designer.

### Conclusion:

We have surveyed the reasons for, and components of PLD development systems, with emphasis on the Hardware Description Languages, and optimization methods in such systems. The conclusions of the survey are that the HDL's are the essential cornerstone of any PLD system, and will control the future directions of any new PLD development tools. The second conclusion is that two-level logic minimization, FSM compiler, and automatic fitting tools are the most important in the PLD optimization area. Also, recent breakthroughs and public availability of heuristic minimizers point to increased use of such tools. Finally, future directions, and an expanding market indicate a wide range of new tools will appear. The key emphasis will be on making them transparent to the user, who, when all is said and done, knows how to design logic best!

### References

- [1] R. Rudell and A. Sangiovanni-Vincentelli, "ESPRESSO-MV: Algorithms for Multiple Valued Logic Minimization", in Proc. Cust. Int. Circ. Conf., IEEE, Portland, OR, May, 1985.
- [2] M.R. Dagenais, V.K. Agarwal and N.C. Rumin, "McBoole: A New Procedure for Exact Logic Minimization", IEEE Trans. on CAD, Jan. 1986, 229-238.
- [3] M. Bartholomeus and H.D. Man, "Presto-II: Yet Another Logic Minimizer for Programmed Logic Arrays", Proc. Int. Symp. Circ. Syst., June 1985, 58.
- [4] R. Rudell, "Multiple-Valued Logic Minimization for PLA Synthesis", M.S. Thesis, University of California, Berkeley, 1986.
- [5] V. D. Agrawal, ed., "VHDL: The VHSIC Hardware Description Language", IEEE Design and Test of Computers, April, 1986.
- [6] J.P. Eurich, "A Tutorial Introduction to the Electronic Design Interchange Format", In Proc. of 23rd Design Automation Conference, July, 1986, 327-333.
- [7] G. DeMicheli, R.K. Brayton, and A. Sangiovanni-Vincentelli, "Optimal State Assignment for Finite-State Machines", IEEE Trans. on CAD, July, 1985, 269-285.
- [8] A. J. Coppola, "An Implementation of a State Assignment Heuristic", In Proc. of 23rd Design Automation Conference, July, 1986, 643-649.

## Dice Example Description

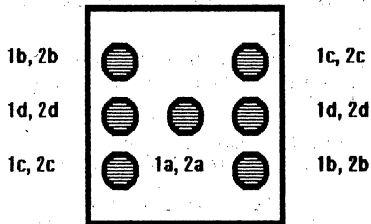
**Problem:** Design a circuit that will roll two dice. Push a switch to start the dice rolling. When the switch is released, a (pseudo) random set of numbers will be displayed.

The example is written using the FSM compiler module of iPLDS. This example is a modification of an existing Application Note[AP-279] design, which is written in ADF language.

The Dice Example pseudo-randomly rolls two dice. The Dice Example is composed of three FSMs. The first two are essentially up-counters, which count from one to six, using the notation groups of one or two LED's, for each of four outputs, to represent the six faces of a die. A picture which indicates the LED groupings, by listing the output signal name next to the LED controlled by it, is given in Diagram 1. The groupings for both die are identical, and hence, listed next to each other.

The third machine generates a short pseudo-random bit sequence by implementing a Linear Feedback Shift Register(LFSR), with three registers. The pseudo-random bit sequences from the LFSR are used to add probabilistic transitions to the up-counter model of each die. The implementation of the LFSR is by straight memorization of the sequence, by means of the state variables of an up-counter.

### Dice LED Encoding



Die 1 Signals: 1a, 1b, 1c, 1d  
Die 2 Signals: 2a, 2b, 2c, 2d

The Dice Example shows the usefulness of logic minimization. Figure 2 shows the FSM Language(State Machine File) representation of the Dice Example under the iPLDS system. Figure 3 shows the ADF code which resulted, as an intermediate step, in the compilation process.

The target device, the 5C060, has 16 I/O macrocells, but each macrocell has only enough room for 8 p-terms. There are 11 equations that result from the Dice Example design. Four for each die, to control the LED's, and three from the state variables of the Linear Feedback Shift Register.

We present a before and after minimization table, showing the effect of the minimization and the automatic DeMorgan's Inversion step.

Equation	Inputs	p-terms before min	p-terms after min
Sv3.d	3	3	2
Sv2.d	3	3	2
Sv1.d	3	3	2
2d.d	6	3	3
2o.d	6	9	4
2b.d	6	6	4
2a.d	6	10	6
1d.d	6	3	3
1c.d	6	9	4
1b.d	6	5	4
1a.d	6	10	7

A necessary condition to fit into the 5C060 is that all of the numbers in the last column be no more than 8, as there are no more than 8 p-terms connected to any macrocell. This particular problem took 2 minutes of CPU time on an 8Mhz PC/AT. Reducing these equations by hand, even for this simple example, would be difficult. The need for the automatic minimizer is clear in this example. Without it, a designer would either have to reduce the equations resulting from the FSM Language by hand, or not use an FSM Language at all, and do the whole design using hand-crafted methods.

Figure 1.



## Dice Example FSM Language(SMF) Description

Alan Coppola  
Intel  
July 21, 1986  
Part No.: LasVegas  
Ver. 3.0  
5C060  
Roll a pair of die  
LB Version 4.01, Baseline 27.1 4/9/86  
PART: 5C060

% No pins assigned:  
Automatic Pin Assignment and Fitting %

INPUTS: clk1, clk2, Go  
OUTPUTS: 1a, 1b, 1c, 1d, 2a, 2b, 2c, 2d

NETWORK:  
clk1 = INP(clk1)  
clk2 = INP(clk2)  
Go = INP(Go)

% Three term LFSR, implemented by storing sequence  
in state variables, which act as flipping coins.  
%

MACHINE: LFSR  
CLOCK: clk2

STATES: [Coin1 Coin2 Coin3]  
S0 [0 0 0]  
S1 [1 0 0]  
S2 [1 1 0]  
S3 [0 1 1]  
S4 [1 0 1]  
S5 [0 1 0]  
S6 [0 0 1]

%  
State equations are:  
Coin2 := Coin1  
Coin3 := Coin2  
Coin1 := ~(Coin2 xor Coin3)

%  
S0:  
S1:  
S2:  
S3:  
S4:  
S5:  
S6:  
S0

MACHINE: Die\_Roll\_1  
CLOCK: clk1

%  
State variables are used as outputs to die.  
Each state encodes the set of LED's to light  
to realize that die value.  
%

STATES: [1a 1b 1c 1d]  
Reset [0 0 0 0]  
One [1 0 0 0]  
Two [0 1 0 0]  
Three [1 1 0 0]  
Four [0 1 1 0]  
Five [1 1 1 0]  
Six [0 1 1 1]

%  
Coin2 is a pseudo-random coin, which controls the  
up-counter transitions, so that the dice roll is  
pseudo-random.  
%

Reset:  
If Go Then One  
One:  
If Go\*Coin2 Then Two  
Two:  
If Go\*Coin2 Then Three  
Three:  
If Go\*Coin2 Then Four  
Four:  
If Go\*Coin2 Then Five  
Five:  
If Go\*Coin2 Then Six  
Six:  
If Go\*Coin2 Then One

MACHINE: Die\_Roll\_2

%  
Duplicate of Die\_Roll\_1 machine, except for  
a different die, using a different pseudo-random coin.  
%

CLOCK: clk2  
STATES: [2a 2b 2c 2d]  
ResetDie2 [0 0 0 0]  
OneDie2 [1 0 0 0]  
TwoDie2 [0 1 0 0]  
ThreeDie2 [1 1 0 0]  
FourDie2 [0 1 1 0]  
FiveDie2 [1 1 1 0]  
SixDie2 [0 1 1 1]

ResetDie2:  
If Go\*Coin3 Then OneDie2  
OneDie2:  
If Go\*Coin3 Then TwoDie2  
TwoDie2:  
If Go\*Coin3 Then ThreeDie2  
ThreeDie2:  
If Go\*Coin3 Then FourDie2  
FourDie2:  
If Go\*Coin3 Then FiveDie2  
FiveDie2:  
If Go\*Coin3 Then SixDie2  
SixDie2:  
If Go\*Coin3 Then OneDie2

END\$

Figure 2.



## Dice Example Hardware Description Language(ADF)

```

Alan Coppola
Intel
July 21, 1986
Part No.: LasVegas
Ver. 3.0
5C060
Roll a pair of die
LB Version 4.01, Baseline 27.1 4/9/86
SMV Version 1.01 BETA2 Baseline 26.1 4/3/86
PART: 5C060

INPUTS:
clk1, clk2, Go

OUTPUTS:
1a, 1b, 1c, 1d, 2a, 2b, 2c, 2d

NETWORK:

    clk1 = INP(clk1)
    clk2 = INP(clk2)
    Go = INP(Go)
%
Three term LFSR, implemented by storing sequence
in state variables, which act as flipping coins.
%
%
I/O's for State Machine "LFSR"
%
Coin1 = NORF(Coin1.d, clk2, GND, GND)
Coin2 = NORF(Coin2.d, clk2, GND, GND)
Coin3 = NORF(Coin3.d, clk2, GND, GND)
%
I/O's for State Machine "Die_Roll_1"
%
1a, 1a = RDRF(1a.d, clk1, GND, GND, VCC)
1b, 1b = RDRF(1b.d, clk1, GND, GND, VCC)
1c, 1c = RDRF(1c.d, clk1, GND, GND, VCC)
1d, 1d = RDRF(1d.d, clk1, GND, GND, VCC)
%
I/O's for State Machine "Die_Roll_2"
%
2a, 2a = RDRF(2a.d, clk2, GND, GND, VCC)
2b, 2b = RDRF(2b.d, clk2, GND, GND, VCC)
2c, 2c = RDRF(2c.d, clk2, GND, GND, VCC)
2d, 2d = RDRF(2d.d, clk2, GND, GND, VCC)

EQUATIONS:
%
Boolean Equations for State Machine "LFSR"
%
%
Current State Equations for "LFSR"
%
S0 = Coin1*Coin2*Coin3;
S1 = Coin1*Coin2*Coin3;
S2 = Coin1*Coin2*Coin3;
S3 = Coin1*Coin2*Coin3;
S4 = Coin1*Coin2*Coin3;
S5 = Coin1*Coin2*Coin3;
S6 = Coin1*Coin2*Coin3;
%
SV Defining Equations for State Machine "LFSR"
%
Coin1.d = S1.n + S2.n + S4.n;
Coin2.d = S2.n + S3.n + S5.n;
Coin3.d = S3.n + S4.n + S6.n;
%
Next State Equations for State Machine "LFSR"
%
S1.n = S0;
S2.n = S1;
S3.n = S2;
S4.n = S3;

```

```

S5.n = S4;
S6.n = S5;
%
Boolean Equations for State Machine "Die_Roll_1"
%
%
Current State Equations for "Die_Roll_1"
%
Reset = 1a*1b*1c*1d;
One = 1a*1b*1c*1d;
Two = 1a*1b*1c*1d;
Three = 1a*1b*1c*1d;
Four = 1a*1b*1c*1d;
Five = 1a*1b*1c*1d;
Six = 1a*1b*1c*1d;
%
SV Defining Equations for State Machine "Die_Roll_1"
%
1a.d = One.n + Three.n + Five.n;
1b.d = One.n + Reset.n;
1c.d = Four.n + Five.n + Six.n;
1d.d = Six.n;
%
Next State Equations for State Machine "Die_Roll_1"
%
One.n = Six * Go * Coin2 + One * (Go * Coin2)
+ Reset * Go;
Reset.n = Reset * (Go);
Three.n = Three * (Go * Coin2) + Two * Go * Coin2;
Four.n = Four * (Go * Coin2) + Three * Go * Coin2;
Five.n = Five * (Go * Coin2) + Four * Go * Coin2;
Six.n = Six * (Go * Coin2) + Five * Go * Coin2;
%
Boolean Equations for State Machine "Die_Roll_2"
%
%
Current State Equations for "Die_Roll_2"
%
ResetDie2 = 2a*2b*2c*2d;
OneDie2 = 2a*2b*2c*2d;
TwoDie2 = 2a*2b*2c*2d;
ThreeDie2 = 2a*2b*2c*2d;
FourDie2 = 2a*2b*2c*2d;
FiveDie2 = 2a*2b*2c*2d;
SixDie2 = 2a*2b*2c*2d;
%
SV Defining Equations for State Machine "Die_Roll_2"
%
2a.d = OneDie2.n + ThreeDie2.n + FiveDie2.n;
2b.d = OneDie2.n + ResetDie2.n;
2c.d = FourDie2.n + FiveDie2.n + SixDie2.n;
2d.d = SixDie2.n;
%
Next State Equations for State Machine "Die_Roll_2"
%
OneDie2.n = SixDie2 * Go * Coin3
+ OneDie2 * (Go * Coin3)
+ ResetDie2 * Go * Coin3;
ResetDie2.n = ResetDie2 * (Go * Coin3);
ThreeDie2.n = ThreeDie2 * (Go * Coin3)
+ TwoDie2 * Go * Coin3;
FourDie2.n = FourDie2 * (Go * Coin3)
+ ThreeDie2 * Go * Coin3;
FiveDie2.n = FiveDie2 * (Go * Coin3)
+ FourDie2 * Go * Coin3;
SixDie2.n = SixDie2 * (Go * Coin3)
+ FiveDie2 * Go * Coin3;
END$

```

Figure 3.





## EPLD THIRD PARTY PROGRAMMING SUPPORT\*

Company	Model	Type	Module	Adaptor	Devices Supported
Adams/Macdonald Enterprises (Promac)	P11	Universal	—	PA-1	5C031, 032, 060
	Sprint Plus	Universal	—	—	5C031, 032, 060
Data I/O	29B	Universal	LogicPack V.4	303A-010 V.2 303A-011A V.7 303A-011B (PLCC)	5C090,121 5C031, 032, 060, 5AC312 5C060
	40	Universal	—	— Chipsite (PLCC)	5C031, 032, 060, 090, 121 5C060, 090, 180
	60A/H	Logic	—	—	5C031, 032, 060
	Unisite	Universal	—	Site 40 Chipsite (PLCC)	5C031, 5C032, 5C060, 5C090, 5C121, 5AC312 5C090, 180
Digelec	803-LDC	PLD	—	—	5C031, 060, 121
	803-DP5	Logic	—	—	5C031, 060, 121
Kontron	EPP-80	Universal	UPM/B UPM/C	— —	5C031, 032, 060 5C031, 032, 060, 090, 121
Oliver Advanced Engineering	Omni 64	Universal	— —	OM-S-20 LCC OM-S-24 LCC	5C031 5C060, 090, 121
Stag	ZL30	Logic	—	—	5C031, 032, 060
	ZL30A	Logic	—	— 30A640	5C031, 032, 060 5C031, 032, 060, 090, 121, 180
	ZL33	Gang	—	—	5C031, 060
	PPZ	Universal	ZM2200	—	5C031, 032, 060, 090, 121

\*Claimed by the manufacturer to support the listed devices. Not qualified by Intel.



## PLA TO EPLD REPLACEMENT

Already in wide use throughout the electronics industry are numerous different Programmable Logic Devices. Many of these are PALs from MMI. Currently, two of our EPLD products, the 5C060 and 5C031 can functionally replace most 24-pin and 20-pin PALs, respectively. A third product, the 5AC312, with its architecturally advanced features, can replace most designs using more complex PALs such as the 20RA10, 22V10, and 32V10.

### The 5C031

The 5C031 is a direct, drop-in replacement for most 20-pin PALs, although some PALs have an incompatible architecture.

### The 5C060

The 5C060 is NOT a drop-in replacement for any 24-pin PAL, though it can functionally replace most. The reason for this is that pin 1 is used as the main clock on registered PALs and as an input on non-registered. Also, pin 13 is used as an OE line on some PALs, and as an input on others. The 5C060, however, uses pin 1 as the left-half synchronous clock input and pin 13 as the right-half synchronous clock input.

While that may not be a problem in some PAL designs, those designs that require clocking or inputs on pins 1 or 13 will necessitate hardware modifications. In the case of the registered PALs, the connection to pin 1 must be rerouted to pin 13 and the OE connected to one of the available inputs (if used). In this manner, the 5C060 can functionally replace the PAL.

### The 5AC312

The 5AC312 is a direct, drop-in replacement for the 20RA10 as well as many of the other simple 24-pin logic devices. The 5AC312 can also serve as a drop-in replacement for most designs using the 22V10 or 32V10 devices.

#### 5C031/5C032 As a 20-Pin PAL Replacement

100% Compatible	Functionally Compatible
10H8, -2	
12H6, -2	
14H4, -2	
16H2, -2	
10L8, -2	
12L6, -2	16R6A
16L8, A-2, A-4	16R4A
16R4, A-2, A-4	16L8A
14L4, -2	16RP6A
16L2, -2	16RP4A
16R8, A-2, A-4	16P8A
16R6, A-2, A-4	16R8A
16P8, -2	16RP8A
16RP8, -2	
16RP6, -2	
16RP4, -2	
16V8	
These are 25 ns-45 ns PALs.	These are 15 ns PALs.

#### 5C060 As a 24-Pin PAL Replacement

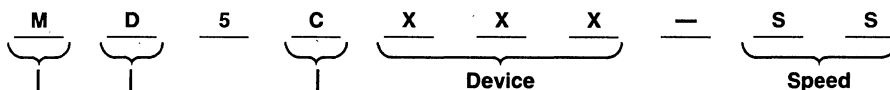
Modified Replacement	Functionally Compatible
12L10	20L8A
14L8	20R8A
16L6	20R6A
18L4	20R4A
20L2	
20L10	
20L8	
20R8	
20R6	
20R4	
20RA10	
With hardware modifications	These are 15 ns PALs.

#### 5AC312 As a 24-Pin PAL Replacement

100% Compatible	100% Compatible (Qualified)
20L8	22V10
20R8	32V10
20R6	Dependent on the number of product terms used.
20R4	
20RA10	

## ORDERING INFORMATION

Intel EPLDs are identified as follows:



**Technology**

- C — CHMOS
- AC — Advanced CHMOS

**Package Type**

- A — Hermetic, Pin Grid Array
- D — Hermetic, Type D (Cerdip) Dip
- N — Plastic, Leaded Chip Carrier
- CJ — Ceramic, J Leaded Chip Carrier
- P — Plastic Dip and Plastic Flatpack
- R — Hermetic, Leadless Chip Carrier
- X — Unpackaged Device

- A — Indicates automotive operating temperature range ( $-40^{\circ}\text{C}$  to  $+125^{\circ}\text{C}$ )
- J — Indicates a JAN qualified device, but is for internal identification purposes only. All JAN devices must be ordered by M38510 part number. (Example: M38510/42001 BQB), and will be marked in accordance with MIL-M-38510 specifications.
- L — Indicates extended operating temperature range ( $-40^{\circ}\text{C}$  to  $+85^{\circ}\text{C}$ ) express product with  $160 \pm 8$  hrs. dynamic burn-in.
- M — Indicates military operating temperature range ( $-55^{\circ}\text{C}$  to  $+125^{\circ}\text{C}$ )
- Q — Indicates commercial temperature range ( $0^{\circ}\text{C}$  to  $70^{\circ}\text{C}$ ) express product with  $160 \pm 8$  hrs. dynamic burn-in.
- T — Indicates extended temperature range ( $-40^{\circ}\text{C}$  to  $+85^{\circ}\text{C}$ ) express product without burn-in.
- No letter indicates commercial temperature range ( $0^{\circ}\text{C}$  to  $70^{\circ}\text{C}$ ) without burn-in.

**Examples:**

QD5C060-45 Commercial with burn-in, ceramic Dip, 060 (600 gate) device, 45 nanosecond.

\*On military temperature devices, B suffix indicates MIL-STD-883C level B processing.

## Device Feature Comparison

	5C031	5C032	5C060	5C090	5C121	5C180	5CBIC	5AC312	5AC324
<b>INPUTS</b>									
Dedicated	10	10	4	12	12	12	8	10	12
Maximum	18	18	20	36	36	60	16	22	36
Input Latches					Y		Y	Y	Y
<b>I/O</b>									
Number	8	8	16	24	24	48	32	12	24
Tri-State	Y	Y	Y	Y	Y	Y	Y	Y	Y
Programmable Polarity	Y	Y	Y	Y	Y	Y	Y	Y	Y
<b>MACROCELLS</b>									
Number	8	8	16	24	28	48	8	12	24
<b>REGISTERS</b>									
Number	8	8	16	24	28	48	8	12	24
Types	D	D	D/T/ RS/JK	D/T/ RS/JK	D	D/T/ RS/JK	D/T/ RS/JK	D/T/ RS/JK	D/T/ RS/JK
Buried Reg. S					4				
Preload	Y	Y	Y	Y	Y	Y	Y	Y	Y
By-Pass	Y	Y	Y	Y	Y	Y	Y	Y	Y
Reset	Y	Y	Y	Y	Y	Y	Y	Y	Y
Preset	Y						Y	Y	Y
<b>PRODUCT TERMS</b>									
Number	74	72	160	240	236	480	112	200	394
Sharing					Y				
Variable Prod. Term Distribution					Y				
<b>LOCAL/GLOBAL BUSES</b>									
					Y	Y			
<b>CLOCKS</b>									
Asynchronous Clocking	1	1	2	2	2	4		2	2
Programmable Clock Edges			Y	Y		Y	Y	Y	Y
					Y				
<b>SECURITY BIT</b>									
	Y	Y	Y	Y	Y	Y	Y	Y	Y
<b>TURBO BIT (LOW POWER)</b>									
		Y	Y	Y	Y	Y	Y	Y	Y

# ELPD CUSTOMER SUPPORT

## Hotline

The Intel EPLD Technical Hotline is manned by application personnel from 8:00 a.m. to 5:00 p.m. (PST) every business day. The number for the United States and Canada is 1-800-323-EPLD (1-800-323-3753). Outside of the U.S. and Canada, contact your local Intel Sales Office. The Hotline is provided to assist with technical questions concerning Intel EPLDs.

## BBS

Intel has a Bulletin Board System for registered iPLS and iPLS II customers to electronically transfer information. Any registered person with a modem can log onto the system. The current number is (916) 985-2308. If your communication software supports file transfers, you can receive utilities, software updates, and the latest information on EPLDs via the Bulletin Board.

Data format for the BBS is as follows:

Start Bits: 1

Stop Bits: 1

Data Bits: 8

Speed: 300 or 1200 BAUD

Transmit/receive protocols supported are:

ASCII

XMODEM

KERMIT

TELINK

Cyclic Redundancy on XMODEM

## EPLD Customer Design Support Center

Intel has a Customer Design Support Center to help customers who are implementing EPLD designs. Service includes answering questions, device selection assistance, and design partitioning as well as limited prototyping, and product/design evaluation and implementation. For more information on the Design Support Center, contact your local Intel field sales office.

## COMPATIBLE COMPUTERS FOR iPLDS II

A partial list of computers that have been verified to be software compatible with the Intel Programmable Logic Development System (iPLDS II) is given below:

AT&T 6300 and 6300+

Compaq family of PCs (88, 86, 286, 386)

IBM AT

IBM XT

IBM XT-286

IBM Personal System II Model 30

HP Vectra

Sperry IT

Tandy 3000 HD

The IBM Personal System II Models 50, 60, 70, and 80 can run iPLS II (Intel Programmable Logic Software)



# DOMESTIC SALES OFFICES

## ALABAMA

Intel Corp.  
5015 Bradford Dr., #2  
Huntsville 35893  
Tel: (205) 830-4010

## ARIZONA

Intel Corp.  
11225 N. 28th Dr.  
Suite D-214  
Phoenix 85029  
Tel: (602) 869-4980

Intel Corp.  
1161 N. El Dorado Place  
Suite 301  
Tucson 85715  
Tel: (602) 299-6815

## CALIFORNIA

Intel Corp.  
21515 Vanowen Street  
Suite 118  
Canoga Park 91303  
Tel: (818) 704-8500

Intel Corp.  
2250 E. Imperial Highway  
Suite 218  
El Segundo 90245  
Tel: (213) 640-6040

Intel Corp.  
1510 Arden Way, Suite 101  
Sacramento 95815  
Tel: (916) 920-8096

Intel Corp.  
4350 Executive Drive  
Suite 105  
San Diego 92121  
Tel: (619) 452-5880

Intel Corp.\*  
400 N. Tustin Avenue  
Suite 460  
Santa Ana 92705  
Tel: (714) 835-9642  
TWX: 910-595-1114

Intel Corp.\*  
San Tomas #4  
2700 San Tomas Expressway  
2nd Floor  
Santa Clara 95051  
Tel: (408) 986-9086  
TWX: 910-338-0255  
FAX: 408-727-2620

## COLORADO

Intel Corp.  
4445 Northpark Drive  
Suite 100  
Colorado Springs 80907  
Tel: (719) 594-6822

Intel Corp.\*  
650 S. Cherry St., Suite 915  
Denver 80222  
Tel: (303) 321-8086  
TWX: 910-931-2289

## CONNECTICUT

Intel Corp.  
26 Mill Plain Road  
2nd Floor  
Danbury 06811  
Tel: (203) 748-3130  
TWX: 710-456-1199

## FLORIDA

Intel Corp.  
6363 N.W. 6th Way, Suite 100  
Ft. Lauderdale 33309  
Tel: (305) 771-0500  
TWX: 510-959-9407  
FAX: 305-772-8193

Intel Corp.  
5850 T.G. Lee Blvd.  
Suite 340  
Orlando 32822  
Tel: (407) 240-8000  
FAX: 407-240-8097

Intel Corp.  
11300 4th Street North  
Suite 170  
St. Petersburg 33716  
Tel: (813) 577-2413  
FAX: 813-578-1607

## GEORGIA

Intel Corp.  
3280 Pointe Parkway  
Suite 200  
Norcross 30092  
Tel: (404) 449-0541

## ILLINOIS

Intel Corp.\*  
300 N. Martingale Road, Suite 400  
Schaumburg 60173  
Tel: (312) 605-8031  
FAX: 312-605-9762

## INDIANA

Intel Corp.  
8777 Purdue Road  
Suite 125  
Indianapolis 46268  
Tel: (317) 875-0623

## IOWA

Intel Corp.  
1930 St. Andrews Drive N.E.  
2nd Floor  
Cedar Rapids 52402  
Tel: (319) 393-5510

## KANSAS

Intel Corp.  
10985 Coody St.  
Suite 140, Bldg. D  
Overland Park 66210  
Tel: (913) 345-2727

## MARYLAND

Intel Corp.\*  
7321 Parkway Drive South  
Suite C  
Hanover 21076  
Tel: (301) 796-7500  
TWX: 710-862-1944

Intel Corp.  
7833 Walker Drive  
Suite 550  
Greenbelt 20770  
Tel: (301) 441-1020

## MASSACHUSETTS

Intel Corp.\*  
Westford Corp. Center  
3 Carlisle Road  
2nd Floor  
Westford 01886  
Tel: (508) 692-3222  
TWX: 710-343-5333

## MICHIGAN

Intel Corp.  
7071 Orchard Lake Road  
Suite 100  
West Bloomfield 48322  
Tel: (313) 851-8096

## MINNESOTA

Intel Corp.  
3500 W. 80th St., Suite 360  
Bloomington 55431  
Tel: (612) 835-8722  
TWX: 910-576-2667

## MISSOURI

Intel Corp.  
4203 Earth City Expressway  
Suite 131  
Earth City 63045  
Tel: (314) 291-1990

## NEW JERSEY

Intel Corp.\*  
Parkway 109 Office Center  
328 Newman Springs Road  
Red Bank 07701  
Tel: (201) 747-2233

Intel Corp.  
280 Corporate Center  
75 Livingston Avenue  
First Floor  
Roseland 07068  
Tel: (201) 740-0111  
FAX: 201-740-0626

## NEW MEXICO

Intel Corp.  
8500 Menaul Boulevard N.E.  
Suite B 295  
Albuquerque 87112  
Tel: (505) 292-8086

## NEW YORK

Intel Corp.  
127 Main Street  
Binghamton 13905  
Tel: (607) 773-0337  
FAX: 607-723-2677

Intel Corp.\*  
850 Cross Keys Office Park.  
Fairport 14450  
Tel: (716) 425-2750  
TWX: 510-253-7391

Intel Corp.\*  
2950 Expressway Dr., South  
Suite 130  
Istislandia 11722  
Tel: (516) 231-3300  
TWX: 510-227-6236

Intel Corp.  
Westage Business Center  
Bldg. 300, Route 9  
Fishkill 12524  
Tel: (914) 897-3860  
FAX: 914-897-3125

## NORTH CAROLINA

Intel Corp.  
5800 Executive Center Dr.  
Suite 105  
Charlotte 28212  
Tel: (704) 588-8966  
FAX: 704-588-2236

Intel Corp.  
2700 Wycliff Road  
Suite 102  
Raleigh 27607  
Tel: (919) 781-8022

## OHIO

Intel Corp.\*  
3401 Park Center Drive  
Suite 220  
Dayton 45414  
Tel: (513) 890-5350  
TWX: 810-450-2528

Intel Corp.\*  
25700 Science Park Dr., Suite 100  
Beachwood 44122  
Tel: (216) 464-2736  
TWX: 810-427-9298

## OKLAHOMA

Intel Corp.  
6801 N. Broadway  
Suite 115  
Oklahoma City 73162  
Tel: (405) 848-8086

## OREGON

Intel Corp.  
15254 N.W. Greenbrier Parkway  
Building B  
Beaverton 97006  
Tel: (503) 645-8051  
TWX: 910-457-8741

## PENNSYLVANIA

Intel Corp.\*  
455 Pennsylvania Avenue  
Suite 230  
Fort Washington 19034  
Tel: (215) 641-1000  
TWX: 510-851-2077

Intel Corp.\*  
400 Penn Center Blvd., Suite 610  
Pittsburgh 15235  
Tel: (412) 823-4970

## PUERTO RICO

Intel Microprocessor Corp.  
South Industrial Park  
P.O. Box 910  
Las Piedras 00671  
Tel: (809) 733-8616

## TEXAS

Intel Corp.  
313 E. Anderson Lane  
Suite 314  
Austin 78752  
Tel: (512) 454-8628

Intel Corp.\*  
12000 Ford Road  
Suite 400  
Dallas 75234  
Tel: (214) 241-8087  
Tel: (214) 464-1180

Intel Corp.  
7322 S.W. Freeway  
Suite 1490  
Houston 77074  
Tel: (713) 988-9086  
TWX: 910-881-2490

## UTAH

Intel Corp.  
428 East 8400 South  
Suite 104  
Murray 84107  
Tel: (801) 253-9051

## VIRGINIA

Intel Corp.  
1504 Santa Rosa Road  
Suite 108  
Richmond 23288  
Tel: (804) 252-5668

## WASHINGTON

Intel Corp.  
155 108th Avenue N.E.  
Suite 386  
Bellevue 98004  
Tel: (206) 453-8086  
TWX: 910-443-3002

Intel Corp.  
408 N. Millian Road  
Suite 102  
Spokane 99206  
Tel: (509) 928-8086

## WISCONSIN

Intel Corp.  
330 S. Executive Dr.  
Suite 102  
Brookfield 53005  
Tel: (414) 784-8087  
FAX: (414) 796-2115

## CANADA

### BRITISH COLUMBIA

Intel Semiconductor of Canada, Ltd.  
4885 Canada Way, Suite 202  
Burnaby V5G 4L6  
Tel: (604) 298-0387  
FAX: (604) 298-8234

### ONTARIO

Intel Semiconductor of Canada, Ltd.  
2650 Queensview Drive  
Suite 250  
Ottawa K2B 8H6  
Tel: (613) 829-9714  
TLX: 053-4115

Intel Semiconductor of Canada, Ltd.  
190 Attwell Drive  
Suite 500  
Rexdale M9W 6H8  
Tel: (416) 675-2105  
TLX: 06963574  
FAX: (416) 675-2438

### QUEBEC

Intel Semiconductor of Canada, Ltd.  
620 St. John Boulevard  
Pointe Claire H9R 3K2  
Tel: (514) 694-9130  
TWX: 514-694-9134



# EUROPEAN SALES OFFICES

## DENMARK

Intel  
Gjølstevej 61, 3rd Floor  
2400 Copenhagen NV  
Tel: (45) (01) 19 80 33  
TLX: 19567

## FINLAND

Intel  
Ruostilantie 2  
00390 Helsinki  
Tel: (358) 0 544 644  
TLX: 123332

## FRANCE

Intel  
1, Rue Edison-BP 303  
78054 St. Quentin-en-Yvelines Cedex  
Tel: (33) (1) 30 57 70 00  
TLX: 699016

Intel  
4, Quai des Etoiles  
69321 Lyon Cedex 05  
Tel: (33) (16) 78 42 40 89  
TLX: 305153

## WEST GERMANY

Intel\*  
Dornacher Strasse 1  
8016 Feldkirchen bei Muenchen  
Tel: (49) 089/90992-0  
TLX: 5-29177  
FAX: 904-3648

Intel  
Hohenzollern Strasse 5  
3000 Hannover 1  
Tel: (49) 0511/344081  
TLX: 9-23625

Intel  
Abraham Lincoln Strasse 16-18  
6200 Wiesbaden  
Tel: (49) 06121/7605-0  
TLX: 4-186163

Intel  
Zettachring 10A  
7000 Stuttgart 80  
Tel: (49) 0711/728728-0  
TLX: 7-254826

## ISRAEL

Intel\*  
Ailudin Industrial Park-Neve Sharef  
P.O. Box 43202  
Tel-Aviv 61430  
Tel: (972) 03-498080  
TLX: 971615

## ITALY

Intel\*  
Milanofori Palazzo E  
20090 Assago  
Milano  
Tel: (39) (02) 824 40 71  
TLX: 341286

## NETHERLANDS

Intel\*  
Marron Meesweg 93  
3068 AV Rotterdam  
Tel: (31) 10.407.11.11  
TLX: 22263

## NORWAY

Intel  
Hvamveien 4-PO Box 92  
2013 Skjetten  
Tel: (47) (6) 842 420  
TLX: 78018

## SPAIN

Intel  
Zurbaran, 28  
28010 Madrid  
Tel: (34) 410 40 04  
TLX: 46680

## SWEDEN

Intel\*  
Dalvingar 24  
171 38 Solna  
Tel: (46) 8 734 01 00  
TLX: 12261

## SWITZERLAND

Intel  
Zuerichstrasse  
8185 Winkel-Ruetli bei Zuerich  
Tel: (41) 01/860 82 82  
TLX: 825977

## UNITED KINGDOM

Intel\*  
Pipers Way  
Swindon, Wiltshire SN3 1RJ  
Tel: (44) (0793) 696000  
TLX: 444447/8

# EUROPEAN DISTRIBUTORS/REPRESENTATIVES

## AUSTRIA

Bacher Electronics G.m.b.H.  
Rotenmuhlgasse 28  
1120 Wien  
Tel: (43) (0222) 83 56 46-0  
TLX: 131532

## BELGIUM

Inelco Belgium S.A.  
Av. des Croix de Guerre 94  
1120 Bruxelles  
Oorlogskrusenlaan, 94  
1120 Brussel  
Tel: (32) (02) 216 01 60  
TLX: 64475

## DENMARK

ITT-Multikomponent  
Naverland 29  
2600 Glostrup  
Tel: (45) (0) 2 45 66 45  
TLX: 34 355

## FINLAND

OY Fintronic AB  
Merkurikatu 24A  
00210 Helsinki  
Tel: (358) (0) 6926022  
TLX: 124224

## FRANCE

Generim  
Z.A. de Courtaboeuf  
Av. de la Battique-BP 88  
91943 Les Ulis Cedex  
Tel: (33) (1) 69 07 78 78  
TLX: 691700

Jermyn  
73-79, rue des Solets  
Silic 585  
94663 Rungis Cedex  
Tel: (33) (1) 45 50 04 00  
TLX: 260567

Metrologie  
Tour d'Asnieres  
4, av. Laurent-Cely  
92096 Asnieres Cedex  
Tel: (33) (1) 47 90 62 40  
TLX: 611448

Tekelec-Airtronic  
Rue Carle Verneil - BP 2  
92015 Sevres Cedex  
Tel: (33) (1) 45 34 75 35  
TLX: 204552

## WEST GERMANY

Electronic 2000 AG  
Stahnruberring 12  
8000 Muenchen 82  
Tel: (49) 089/42001-0  
TLX: 522561

ITT Multikomponent GmbH  
Postfach 1265  
Bahnhofstrasse 44  
7141 Moeglingen  
Tel: (49) 07141/4879  
TLX: 726472

Jermyn GmbH  
Im Dachsstueck 9  
6250 Limburg  
Tel: (49) 06431/508-0  
TLX: 415257-0

Metrologie GmbH  
Meglingerstrasse 49  
8000 Muenchen 71  
Tel: (49) 089/78042-0  
TLX: 5213189

Proelectron Vertriebs GmbH  
Max Planck Strasse 1-3  
6072 Dreieich  
Tel: (49) 06103/3040  
TLX: 417903

Micro Marketing Ltd.  
Glenageary Office Park  
Glenageary  
Co. Dublin  
Tel: (21) (853) (01) 85 63 25  
TLX: 31584

## IRELAND

Micro Marketing Ltd.  
Glenageary Office Park  
Glenageary  
Co. Dublin  
Tel: (21) (853) (01) 85 63 25  
TLX: 31584

## ISRAEL

Electronics Ltd.  
11 Rozansin Street  
P.O.B. 39300  
Tel-Aviv 61392  
Tel: (972) 03-475151  
TLX: 33638

## ITALY

Intesi  
Divisione ITT Industries GmbH  
Viale Milanofori  
Palazzo E/5  
20090 Assago  
Milano  
Tel: (39) 02/824701  
TLX: 311351

Lasi Elettronica S.p.A.  
V. le Fulvio Testi, 126  
20092 Cinisello Balsamo  
Milano  
Tel: (39) 02/2440012  
TLX: 352040

## NETHERLANDS

Koning en Hartman  
1 Energieweg  
2627 AP Delft  
Tel: (31) 15609906  
TLX: 36250

## NORWAY

Nordisk Elektrikk (Norge) A/S  
Postboks 123  
Snedrevingen 4  
1364 Hvalstad  
Tel: (47) (02) 84 62 10  
TLX: 77546

## PORTUGAL

Diram  
Avenida Marques de Tomar, 46-A  
1000 Lisboa  
Tel: (351) (1) 73 48 34  
TLX: 14182

## SPAIN

ATD Electronica, S.A.  
Plaza Ciudad de Viena, 6  
28040 Madrid  
Tel: (34) 234 40 00  
TLX: 42754

ITT-SESA  
Calle Miguel Angel, 21-3  
28010 Madrid  
Tel: (34) 419 09 57  
TLX: 27461

## SWEDEN

Nordisk Elektronik AB  
Huvudstagan  
Box 1409  
171 27 Solna  
Tel: (46) 08-734 97 70  
TLX: 105 47

## SWITZERLAND

Industrie A.G.  
Hertstrasse 31  
8304 Wallisellen  
Tel: (41) (801) 83 05 04 0  
TLX: 56788

## TURKEY

EMPA Electronic  
Lindwurmstrasse 95A  
8000 Muenchen 2  
Tel: (49) 089/53 80 570  
TLX: 526573

## UNITED KINGDOM

Accent Electronic Components Ltd.  
Jubilee House, Jubilee Road  
Letchworth, Herts SG8 1TL  
Tel: (44) (0462) 866666  
TLX: 62693

Bytech-Cornway Systems  
3 The Western Centre  
Western Road  
Bracknell RG12 1RW  
Tel: (44) (0344) 53333  
TLX: 847201

Jermyn  
Vestry Estate  
Oxford Road  
Sevenoaks  
Kent TN14 5EU  
Tel: (44) (0732) 450144  
TLX: 35142

MMD  
Unit 8 Southview Park  
Caversham  
Reading  
Berkshire RG4 0AF  
Tel: (44) (0734) 48 16 66  
TLX: 846669

Rapid Silicon  
Rapid House  
Denmark Street  
High Wycombe  
Buckinghamshire HP11 2ER  
Tel: (44) (0494) 442266  
TLX: 837931

Rapid Systems  
Rapid House  
Denmark Street  
High Wycombe  
Buckinghamshire HP11 2ER  
Tel: (44) (0494) 450244  
TLX: 837931

## YUGOSLAVIA

Rapido Electronic Components S.p.a.  
Via C. Beccaria, 8  
34133 Trieste  
Italia  
Tel: (39) 040/360555  
TLX: 460461



**UNITED STATES**  
Intel Corporation  
3065 Bowers Avenue  
Santa Clara, CA 95051

**JAPAN**  
Intel Japan K.K.  
5-6 Tokodai, Tsukuba-shi  
Ibaraki, 300-26

**FRANCE**  
Intel Corporation S.A.R.L.  
1, Rue Edison, BP 303  
78054 Saint-Quentin-en-Yvelines Cedex

**UNITED KINGDOM**  
Intel Corporation (U.K.) Ltd.  
Pipers Way  
Swindon  
Wiltshire, England SN3 1RJ

**WEST GERMANY**  
Intel Semiconductor GmbH  
Dornacher Strasse 1  
8016 Feldkirchen bei Muenchen

**HONG KONG**  
Intel Semiconductor Ltd.  
10/F East Tower  
Bond Center  
Queensway, Central

**CANADA**  
Intel Semiconductor of Canada, Ltd.  
190 Attwell Drive, Suite 500  
Rexdale, Ontario M9W 6H8