PAL®/PLE™ DEVICE Programmable Logic Array Handbook

Fifth Edition

Monolithic Memories

PAL®/PLE™ Device Programmable Logic Array

Handbook

FIFTH EDITION

Monolithic Memories MMI

• 2175 Mission College Boulevard • Santa Clara, CA 95054-1592 • (408) 970-9700 • (910) 339-9220

Table of Contents

Table of Contents

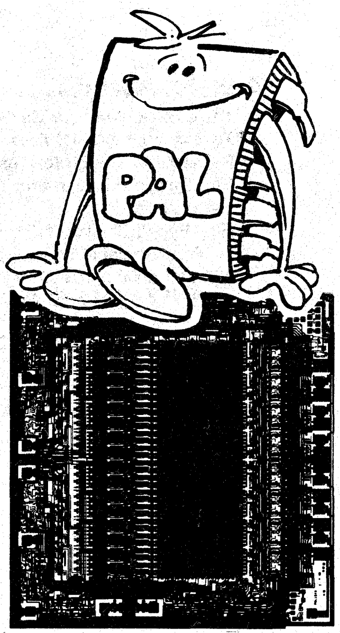
Table of Contents

Table of Contents

Logic Tutorial 8-1

Table of Contents

Table of Contents



The PAL® Device Concept

Monolithic Memories' family of PAL devices gives designers a powerful tool with unique capabilities for use in new and existing logic designs. The PAL device saves time and money by solving many of the system partitioning and interface problems brought about by increases in semiconductor device technology.

Rapid advances in large scale integration technology have led to larger and larger standard logic functions; single I.C.s now perform functions that formerly required complete circuit cards. While LSI offers many advantages, advances have been made at the expense of device flexibility. Most LSI devices still require large numbers of SSI/MSI devices for interfacing with user systems. Designers are still forced to turn to random logic for many applications.

The designer is confronted with another problem when a product is designed. Often the function is well defined and could derive significant benefits from fabrication as an integrated circuit. However, the design cycle for a custom circuit is long and the costs can be very high. This makes the risk significant enough to deter most users. The technology to support maximum flexibility combined with fast turnaround on custom logic has simply not been available. Monolithic Memories offers the programmable solution.

The PAL device family offers a fresh approach to using fuse programmable logic. PAL circuits are a conceptually unified group of devices which combine programmable flexibility with high speed and an extensive selection of interface options. PAL devices can lower inventory, cut design cycles and provide high complexity with maximum flexibility. These features, combined with lower package count and high reliability, truly make the PAL device a circuit designer's best friend.

The PAL Device — Teaching Old PROMs New Tricks

CD00910M

Monolithic Memories developed the modern PROM and introduced many of the architectures and techniques now regarded as industry standards. As the world's largest PROM manufacturer, Monolithic Memories has the proven technology and high volume production capability required to manufacture and support the PAL device.

The PAL device is an extension of the fusible link technology pioneered by Monolithic Memories for use in bipolar PROMs. The fusible link PROM first gave the digital systems designer the power to "write on silicon". In a few seconds he was able to transform a blank PROM from a general purpose device into one containing a custom algorithm, microprogram, or Boolean transfer function. This opened up new horizons for the use of PROMs in computer control stores, character generators, data storage tables and many other applications. The wide acceptance of this technology is clearly demonstrated by today's multi-million dollar PROM market.

The key to the PROM's success is that it allows the designer to customize the chip quickly and easily to fit his unique requirements. The PAL device extends this programmable flexibility by utilizing proven fusible link technology to implement logic functions. By using PAL circuits the designer can quickly and effectively implement custom logic varying in complexity from random gates to complex arithmetic functions.

ANDs and ORs

The PAL device implements the familiar sum-of-products logic by using a programmable AND array whose output terms feed a fixed OR array. Since the sum-of-products form can express any Boolean transfer function, the PAL circuit uses are only limited by the number of terms available in the AND — OR arrays. PAL devices come in different sizes to allow for effective logic optimization.

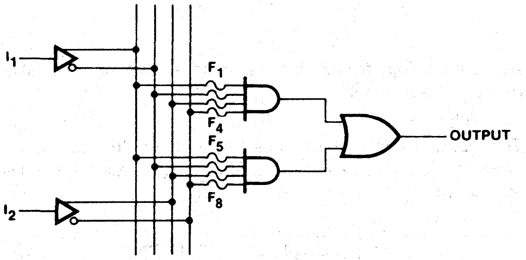
Figure 1 shows the basic PAL device structure for a two-input, one-output logic segment. The general logic equation for this segment is:

$$\text{Output} = (I_1 + \overline{f_1})(\overline{I_1} + \overline{f_2})(I_2 + \overline{f_3})(\overline{I_2} + \overline{f_4}) + (I_1 + \overline{f_5})(\overline{I_1} + \overline{f_6})(I_2 + \overline{f_7})(\overline{I_2} + \overline{f_8})$$

where the "f" terms represent the state of the fusible links in the PAL device AND array. An unblown link represents a logic 1. Thus:

fuse blown, f = 0
fuse intact, f = 1

An unprogrammed PAL device has all fuses intact.

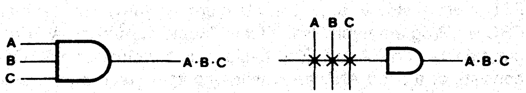


LD00860M

Figure 1

PAL Device Notation

Logic equations, while convenient for small functions, rapidly become cumbersome in large systems. To reduce possible confusion, complex logic networks are generally defined by logic diagrams and truth tables. Figure 2 shows the logic convention adopted to keep PAL device logic easy to understand and use; in the figure, an "x" represents an intact fuse used to perform the logic AND function. (Note: the input terms on the common line with the x's are not connected together.) The logic symbology shown in Figure 2 has been informally adopted by integrated circuit manufacturers because it clearly establishes a one-to-one correspondence between the chip layout and the logic diagram. It also allows the logic diagram and truth table to be combined into a compact and easy to read form, thereby serving as a convenient shorthand for PAL circuits. The two-input, one-output example from Figure 1, redrawn using the new logic convention, is shown in Figure 3.



LD00870M

Figure 2

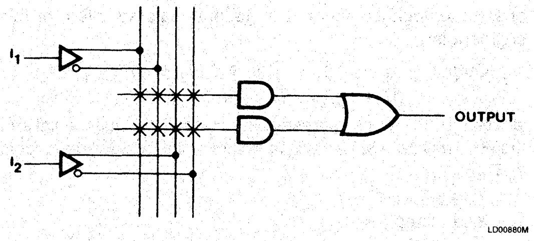


Figure 3

As a simple PAL device example, consider the implementation of the transfer function:

Output = $I_1\overline{I}_2 + \overline{I}_1 I_2$

The normal combinatorial logic diagram for this function is shown in Figure 4, with the PAL device logic equivalent shown in Figure 5.

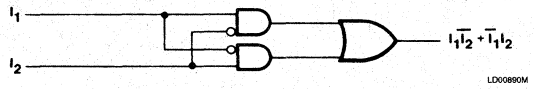


Figure 4

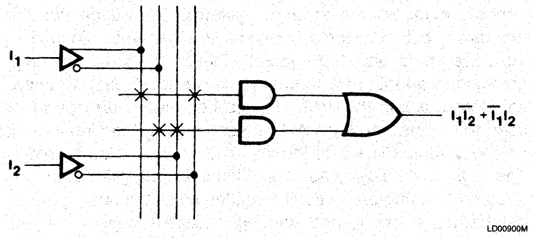


Figure 5

Using this logic convention it is now possible to compare the PAL device structure to the structure of the more familiar PROM (Programmable Read-Only Memory) and PLA (Programmable Logic Array). The basic logic structure of a PROM consists of a fixed AND array whose outputs feed a programmable OR array (Figure 6). PROMs are low-cost, easy to program, and available in a variety of sizes and organizations. They are most commonly used to store computer programs and data. In these applications the fixed input is a computer memory address; the output is the content of that memory location.

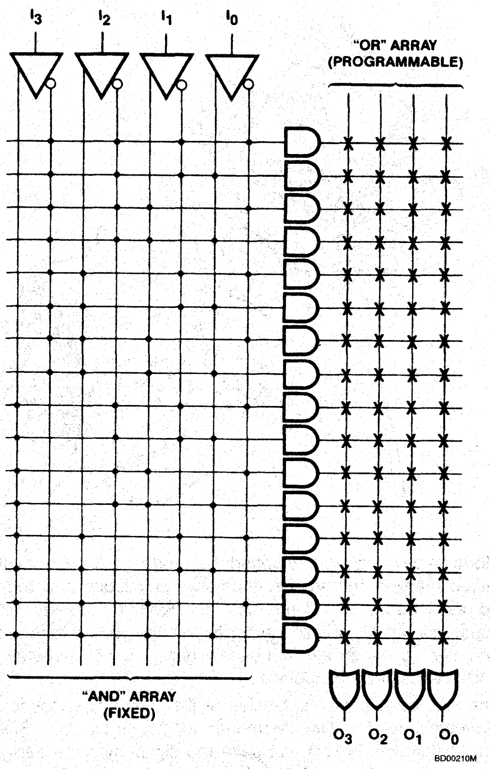


Figure 6

The basic logic structure of the PLA consists of a programmable AND array whose outputs feed a programmable OR array (Figure 7). Since the designer has complete control over all inputs and outputs, the PLA provides the ultimate flexibility for implementing logic functions. They are used in a wide variety of applications. However, this generality can make PLAs expensive, quite formidable to understand, and costly to program.

The basic logic structure of the PAL (Programmable Array Logic) device, as mentioned earlier, consists of a programmable AND array whose outputs feed a fixed OR array (Figure 8). The PAL device combines much of the flexibility of the PLA with the low cost and easy programmability of the PROM. Table 1 summarizes the characteristics of the PROM, PLA, and PAL device logic families.

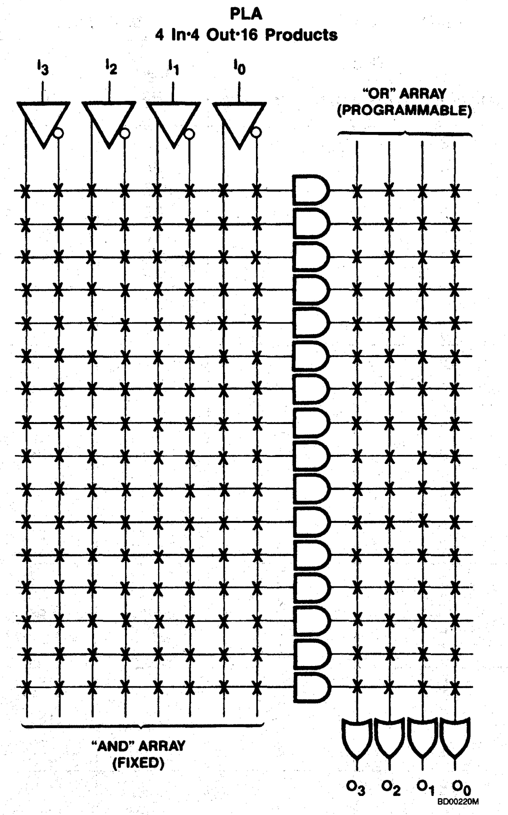


Figure 7

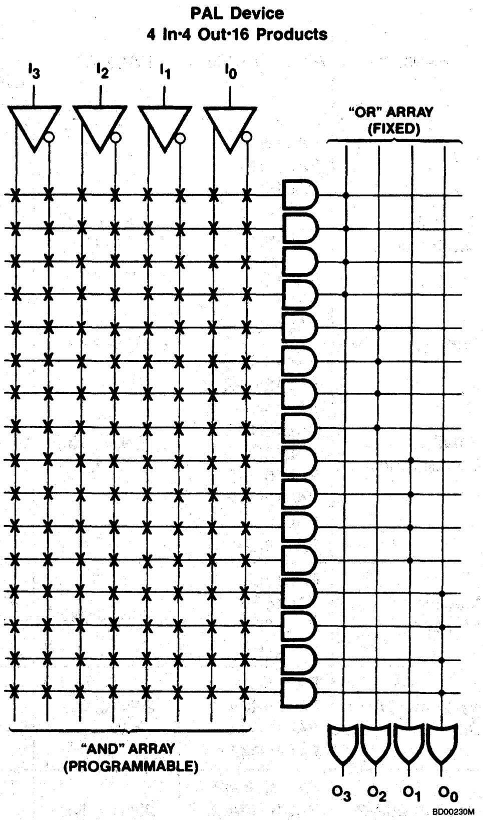


Figure 8

	AND	OR	OUTPUT OPTIONS
PROM	Fixed	Prog	TS, OC
PLA	Prog	Prog	TS, OC, Prog. Polarity
PAL Device	Prog	Fixed	TS, Registered Feedback, I/O Prog. Polarity

Table 1.

PAL Devices For Every Task

The members of the PAL device family and their characteristics are summarized in the PAL device menu. They are designed to cover the spectrum of logic functions at reduced cost and lower package count. This allows the designer to select the PAL device that best fits his application. PAL device units come in the following basic configurations:

PAL device logic arrays are available in sizes from 6 x 16 (6 input terms, 16 output terms) to 64 x 32, with both active high and active low output configurations available (ref. PAL device menu). This wide variety of input/output formats allows the PAL device to replace many different sized blocks of logic with single packages.

Monolithic Memories PAL® Device Menu

FAMILY	PART NUMBER	PACKAGE	DESCRIPTION			MAXIMUM	
			INPUTS	OUTPUTS		t_{PD}* ns	I_{CC} mA
				COMBINATORIAL	REGISTERED		
Small 20 Combinatorial	PAL10H8	20N, J, NL, F, L	10	8	–	35	90
	PAL12H6		12	6	–	35	
	PAL14H4		14	4	–	35	
	PAL16H2		16	2	–	35	
	PAL16C1		16	2	–	40	
	PAL10L8		10	8	–	35	
	PAL12L6		12	6	–	35	
	PAL14L4		14	4	–	35	
	PAL16L2		16	2	–	35	
Small 20-2 Combinatorial	PAL10H8-2	20N, J, NL, F, L	10	8	–	60	45
	PAL12H6-2		12	6	–		
	PAL14H4-2		14	4	–		
	PAL16H2-2		16	2	–		
	PAL16C1-2		16	2	–		
	PAL10L8-2		10	8	–		
	PAL12L6-2		12	6	–		
	PAL14L4-2		14	4	–		
	PAL16L2-2		16	2	–		
Medium 20 Standard	PAL16L8	20N, J, NL	16	8	–	35	180
	PAL16R8		16	–	8		
	PAL16R6		16	2	6		
	PAL16R4		16	4	4		
Medium 20A Standard	PAL16L8A	20N, J, NL, W, L	16	8	–	25	180
	PAL16R8A		16	–	8		
	PAL16R6A		16	2	6		
	PAL16R4A		16	4	4		
Medium 20A-2 Standard	PAL16L8A-2	20N, J, NL, F, L	16	8	–	35	90
	PAL16R8A-2		16	–	8		
	PAL16R6A-2		16	2	6		
	PAL16R4A-2		16	4	4		
Medium 20A-4 Standard	PAL16L8A-4	20N, J, NL, F, L	16	8	–	55	50
	PAL16R8A-4		16	–	8		
	PAL16R6A-4		16	2	6		
	PAL16R4A-4		16	4	4		
Medium 20B Standard	PAL16L8B	20N, J, NL, W, L	16	8	–	15	180
	PAL16R8B		16	–	8		
	PAL16R6B		16	2	6		
	PAL16R4B		16	4	4		
Medium 20B-2 Standard	PAL16L8B-2	20N, J, NL, W, L	16	8	–	25	90
	PAL16R8B-2		16	–	8		
	PAL16R6B-2		16	2	6		
	PAL16R4B-2		16	4	4		
Medium 20B-4 Standard	PAL16L8B-4	20N, J, NL, W, L	16	8	–	35	55
	PAL16R8B-4		16	–	8		
	PAL16R6B-4		16	2	6		
	PAL16R4B-4		16	4	4		
Medium 20D Standard	PAL16L8D	20N, J, NL	16	8	–	10	180
	PAL16R8D		16	–	8		
	PAL16R6D		16	2	6		
	PAL16R4D		16	4	4		

Monolithic Memories PAL® Device Menu

FAMILY	PART NUMBER	PACKAGE	DESCRIPTION			MAXIMUM	
			INPUTS	OUTPUTS		t_{PD}* ns	I_{CC} mA
				COMBINATORIAL	REGISTERED		
Medium 20AP Programmable Polarity	PAL16P8A	20N, J, NL	16	8	–	25/30**	180
	PAL16RP8A		16	–	8		
	PAL16RP6A		16	2	6		
	PAL16RP4A		16	4	4		
Large 20 Arithmetic	PAL16X4	20N, J, NL, F, L	16	4	4	40	225
	PAL16A4		16	4	4		240
Large 20RA Asynchronous	PAL16RA8	20N, J, NL	16	–	8	30/35**	170

* Minimum commercial t_{SU} for devices with all registered outputs.

** Polarity fuse programmed (active High).

N = Plastic DIP
NS = Plastic SKINNYDIP
J = Ceramic DIP
JS = Ceramic SKINNYDIP
NL = Plastic Leaded Chip Carrier (PLCC)
P = Pin Grid Array
L = Leadless Chip Carrier (LCC)
W = Cerpack
F = Ceramic Solder Seal Flat Pack

PAL Circuit Series 24

FAMILY	PART NUMBER	PACKAGE	DESCRIPTION			MAXIMUM	
			INPUTS	OUTPUTS		t_{PD}* ns	I_{CC} mA
				COMBINATORIAL	REGISTERED		
Small 24 Combinatorial	PAL12L10	24NS, JS, W, 28NL, 28L	10	10	–	40	100
	PAL14L8		14	8	–		
	PAL16L6		16	6	–		
	PAL18L4		18	4	–		
	PAL20L2		20	2	–		
	PAL20C1		20	2	–		
Small 24A Decoder	PAL6L16A	24NS, JS, 28NL	6	16	–	25	90
	PAL8L14A		8	14	–		
Medium 24A Standard	PAL20L8A	24NS, JS, W, 28NL, 28L	20	8	–	25	210
	PAL20R8A		20	–	8		
	PAL20R6A		20	2	6		
	PAL20R4A		20	4	4		
Medium 24A-2 Standard	PAL20L8A-2	24NS, JS, W, 28NL, 28L	20	8	–	35	105
	PAL20R8A-2		20	–	8		
	PAL20R6A-2		20	2	6		
	PAL20R4A-2		20	4	4		
Medium 24B Standard	PAL20L8B	24NS, JS, W, 28NL, 28L	20	8	–	15	210
	PAL20R8B		20	–	8		
	PAL20R6B		20	2	6		
	PAL20R4B		20	4	4		
Medium 24X Exclusive OR	PAL20L10	24NS, JS, W, 28NL, 28L	20	10	–	50	165
	PAL20X10		20	–	10		180
	PAL20X8		20	2	8		180
	PAL20X4		20	4	4		180
Medium 24XA Exclusive OR	PAL20L10A	24NS, JS, W, 28NL, 28L	20	10	–	30	165
	PAL20X10A		20	–	10		180
	PAL20X8A		20	2	8		180
	PAL20X4A		20	6	4		180
Large 24RS Shared Product Terms	PAL20S10	24NS, JS, W, 28NL, 28L	20	10	–	35/40**	240
	PAL20RS10		20	–	10	35	240
	PAL20RS8		20	2	8	35/40**	240
	PAL20RS4		20	6	4	35/40**	240
Large 24RA Asynchronous	PAL20RA10	24NS, JS, W, 28NL, 28L	20	–	10	30/35**	200
Large 24VX Varied XOR	PAL32VX10	24NS, JS, 28NL	32	–	10	30	180
	PAL32VX10A					25	
ECL	PAL10H20P8	24NS, JS, 28NL	20	8	–	6	210

PAL Circuit Series MegaPAL™

FAMILY	PART NUMBER	PACKAGE	DESCRIPTION		MAXIMUM	
			INPUTS	REGISTERED OUTPUTS	t_{PD}* ns	I_{CC} mA
1500 Gates	PAL 32R16	40N, J, 44NL, 44L	32	16	40/45**	280
5000 Gates	PAL64R32	88P, 84L	64	32	50/55**	640

Basic PAL Device Features

PAL devices offer a number of different types of outputs. The basic types include the following:

- Combinatorial
- Programmable I/O
- Registered Outputs with Feedback
- XOR Outputs

Combinatorial

The simplest PAL devices have combinatorial outputs with no feedback. The basic structure shown in Figure 9 depicts four product terms summed at an active-low output.

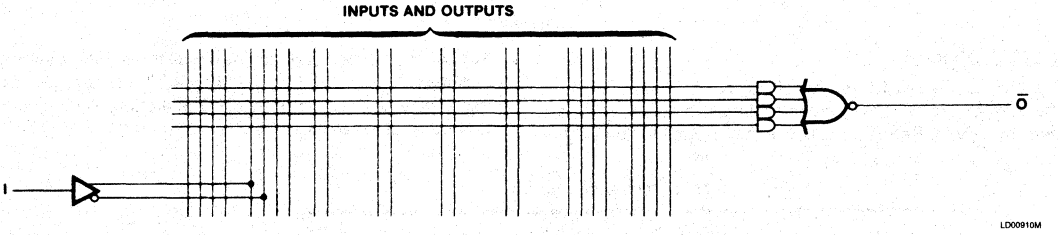


Figure 9

Programmable I/O

A feature of the high-end members of the PAL device family is programmable input/output. This allows the product terms to directly control the outputs of the PAL device (Figure 10). One product term is used to enable the three-state buffer, which in turn gates the summation term to the output pin. The output is also fed back into the PAL device array as an input. Thus the PAL device drives the I/O pin when the three-state gate is enabled; the I/O pin is an input to the PAL device array when the three-state gate is disabled. This feature can be used to allocate available pins for I/O functions or to provide bi-directional output pins for operations such as shifting and rotating serial data.

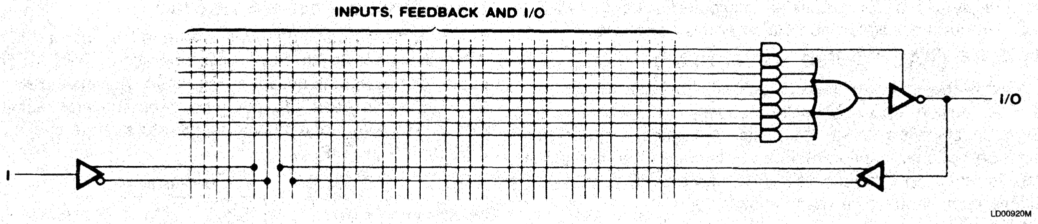


Figure 10

Registered Outputs with Feedback

Another feature of the high-end members of the PAL device family is registered data outputs with registered feedback. Each product term is summed into a D-type output flip-flop on the rising edge of the system clock (Figure 11). The Q output of the flip-flop can then be gated to the output pin by enabling the active low three-state buffer.

In addition to being available for transmission, the Q output is fed back into the PAL device array as an input term. This feedback allows the PAL device to "remember" the previous state, and it can alter its function based upon that state. This allows the designer to configure the PAL device as a state sequencer which can be programmed to execute such elementary functions as count up, count down, skip, shift, and branch. These functions can be executed by the registered PAL device at rates of up to 55.5MHz.

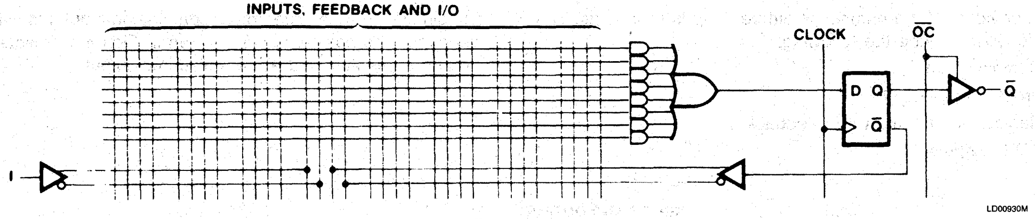


Figure 11

XOR Outputs

These PAL devices feature an exclusive-OR (XOR) function. The sum of products is segmented into two sums which are then exclusive ORed at the input of the D-type flip-flop (Figure 12). All of the features of the Registered PAL devices are included in the XOR PAL unit. The XOR function provides an easy implementation of the HOLD operation used in counters and other state sequencers.

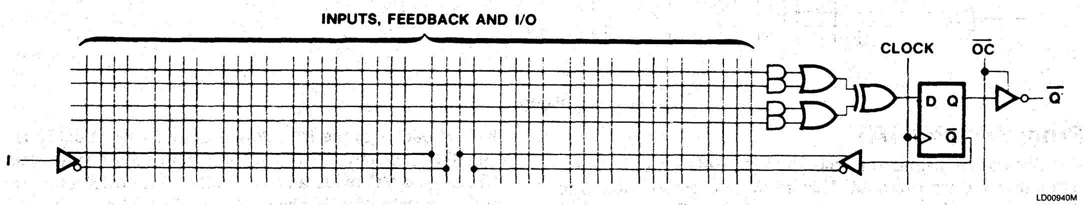


Figure 12

PAL Device Programming

PAL devices can be programmed in most standard PROM programmers with the addition of a PAL device personality card. For details on programming equipment, see the PAL Device Programmer Reference Guide in this handbook.

PALASM (PAL Device Assembler)

PALASM is the software used to define, simulate, build, and test PAL device units. PALASM accepts the PAL device Design Specification as an input file. It verifies the design against an optional function table and generates the fuse plot which is used to program the PAL devices. PALASM is available upon request for many computers.

HAL (Hard Array Logic) Device

The HAL device family is the mask programmed version of a PAL device. The HAL device is to a PAL device just as a ROM is to a PROM. A standard wafer is fabricated as far as the metal mask. Then a custom metal mask is used to fabricate aluminum links for a HAL device instead of the programmable TiW fuses used in a PAL device.

PAL Device Technology

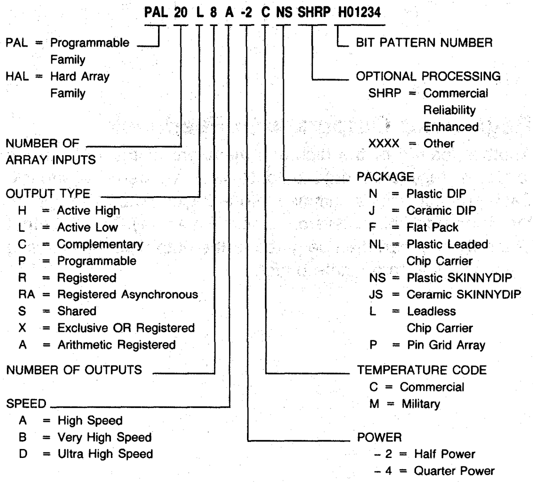
PAL circuits are manufactured using the proven TTL Schottky bipolar TiW fuse process to make fusible-link PROMs. An NPN emitter follower array forms the programmable AND array. PNP inputs provide high impedance inputs (0.25mA max) to the array. All outputs are standard TTL drivers with internal active pull-up transistors.

PAL Device Data Security

The circuitry used for programming and logic verification can be used at any time to determine the logic pattern stored in the PAL device array. For security, the PAL device has a "last fuse" which can be blown to disable the verification logic. This provides a significant deterrent to potential copiers, and it can be used to protect proprietary designs.

PAL Device Part Numbers

The PAL device part number is unique in that the part number code also defines the part's logic operation. The PAL device numbering system is shown in Figure 13. For example, a PAL 14L4CN would be a 14-input term, 4-output term, active-low PAL device with a commercial temperature range packaged in a 20-pin plastic DIP.



PAL 20 L 8 A -2 C NS SHRP H01234

PAL = Programmable Family
HAL = Hard Array Family

NUMBER OF ARRAY INPUTS

OUTPUT TYPE
H = Active High
L = Active Low
C = Complementary
P = Programmable
R = Registered
RA = Registered Asynchronous
S = Shared
X = Exclusive OR Registered
A = Arithmetic Registered

NUMBER OF OUTPUTS

SPEED
A = High Speed
B = Very High Speed
D = Ultra High Speed

BIT PATTERN NUMBER

OPTIONAL PROCESSING
SHRP = Commercial Reliability Enhanced
XXXX = Other

PACKAGE
N = Plastic DIP
J = Ceramic DIP
F = Flat Pack
NL = Plastic Leaded Chip Carrier
NS = Plastic SKINNYDIP
JS = Ceramic SKINNYDIP
L = Leadless Chip Carrier
P = Pin Grid Array

TEMPERATURE CODE
C = Commercial
M = Military

POWER
-2 = Half Power
-4 = Quarter Power

Figure 13

PAL Device Logic Symbols

The logic symbols for each of the individual PAL devices gives a concise functional description of the PAL device logic function. This symbol makes a convenient reference when selecting the PAL device that best fits a specific application. Figure 14 shows the logic symbol for a PAL10H8 array.

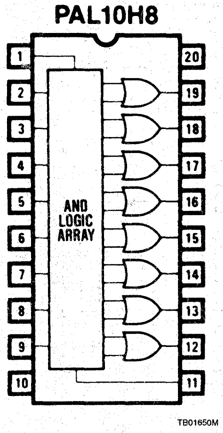


Figure 14

A PAL Device Example

As an example of how the PAL device enables the designer to reduce costs and simplify logic design, consider the design of a simple, high-volume consumer product: an electronic dice game. This type of product will be produced in extremely high volume, so it is essential that every possible production cost be minimized.

The electronic dice game is simply constructed using a free running oscillator whose output is used to drive two asynchronous modulo six counters. When the user "rolls" the dice (presses a button), the current state of the counters is decoded and latched into a display resembling the pattern seen on an ordinary pair of dice.

A conventional logic diagram for the dice game is shown in Figure 15. It is implemented using standard TTL, SSI and MSI parts, with a total I.C. count of eight: six quad gate packages and two quad D-latches. Looks like a nice clean logic design, right? Wrong!!

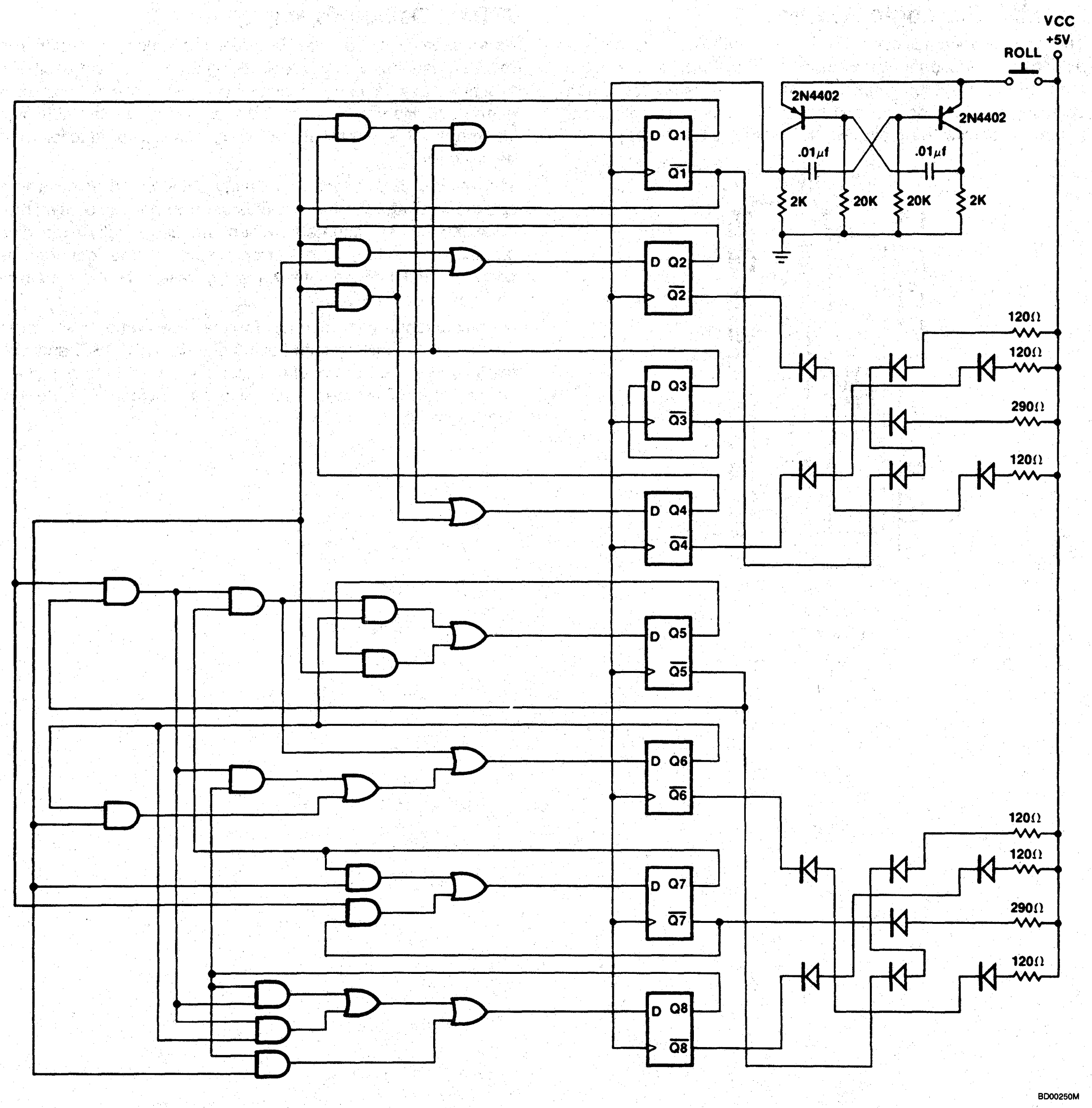

VCC
+5V
ROLL
2N4402
2N4402
.01μf
.01μf
2K
20K
20K
2K
D Q1
Q1
D Q2
Q2
D Q3
Q3
D Q4
Q4
D Q5
Q5
D Q6
Q6
D Q7
Q7
D Q8
Q8
120Ω
120Ω
290Ω
120Ω
120Ω
120Ω
290Ω
120Ω
BD00250M

Figure 15

A brief examination of Figure 15 reveals two basic facts: first, the circuit contains mostly simple, combinatorial logic, and second, it uses a clocked state transition sequence. Remembering that the PAL device family contains ample provision for these features, the PAL device catalog is consulted. The PAL16R8 has all the required functions, and the entire logic content of the circuit can be programmed into a single PAL device shown in Figure 16.

In this example, the PAL device effected an eight-to-one package count reduction and a significant cost savings. This is typical of the power and cost-effective performance that the PAL device family brings to logic design.

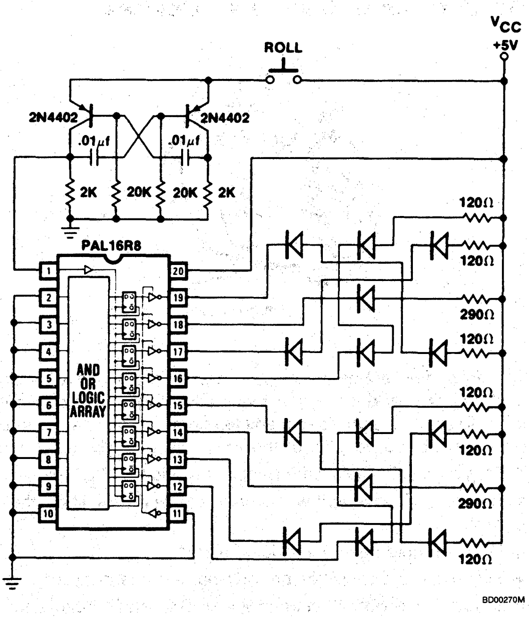
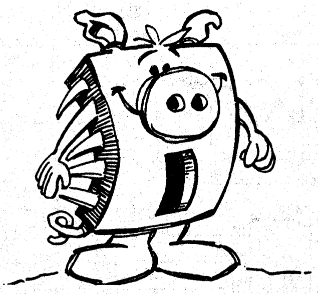


Figure 16

Advantages of Using PAL Devices



CD00920M

The PAL device has a unique place in the world of logic design. Not only does it offer many advantages over conventional logic, it also provides many features not found anywhere else. Among the benefits of the PAL device family:

- Programmable replacement for conventional TTL logic.
- Reduces IC inventories substantially and simplifies their control.
- Reduces chip count by at least 4 to 1.
- Expedites and simplifies prototyping and board layout.
- Saves space with 20-pin and 24-pin SKINNYDIP packages, and surface-mount PLCC packages.
- High speed: 10ns maximum propagation delay, on D series.
- Programmed on standard PROM programmers.
- Programmable three-state outputs.
- Special feature eliminates possibility of copying by competitors.

All of these features combine together to lower product development costs and increase product cost effectiveness. The bottom line is that PAL devices save money.

Direct Logic Replacement

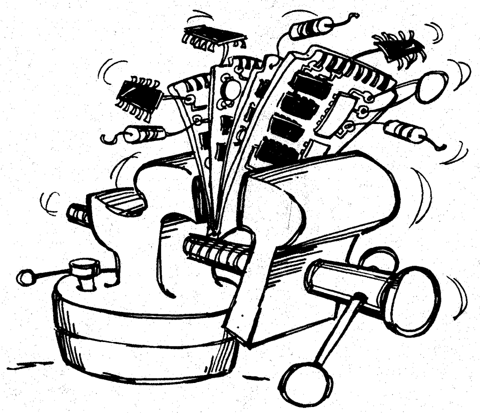
CD00940M

In both new and existing designs, the PAL device can be used to replace various logic functions. This allows the designer to optimize a circuit in many ways never before possible. The PAL device is particularly effective when used to provide interfaces required by many LSI functions. The combination of PAL device flexibility and LSI function density makes a powerful team.

Design Flexibility

The PAL device offers the systems logic designer a whole new world of options. Until now, the decision on logic system implementation was usually between SSI/MSI logic functions on one hand and microprocessors on the other. In many cases the function required is too awkward to implement the first way and too simple to justify the second. Now the PAL device offers the designer high functional density, high speed, and low cost. Even better, PAL devices come in a variety of sizes and functions, thereby further increasing the designer's options.

Space Efficiency

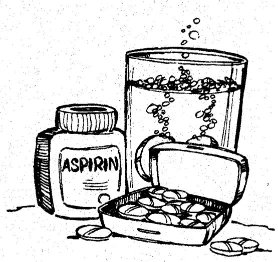


CD00930M

By allowing designers to replace many simple logic functions with single packages, the PAL device allows more compact P.C. board layouts. The PAL device space-saving 20-pin and 24-pin SKINNYDIP packages help to reduce board area further while simplifying board layout and fabrication. This means that many multi-card systems can now be reduced to one or two cards, and that can make the difference between a profitable success or an expensive disaster.

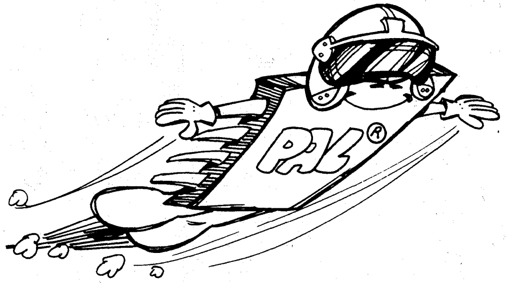
Smaller Inventory

The PAL device family can be used to replace up to 90% of the conventional TTL family. This considerably lowers both shelving and inventory cataloging requirements. In addition, small custom modifications to the standard functions are easy for PAL device users, but not so easy for standard TTL users.



CD00950M

High Speed



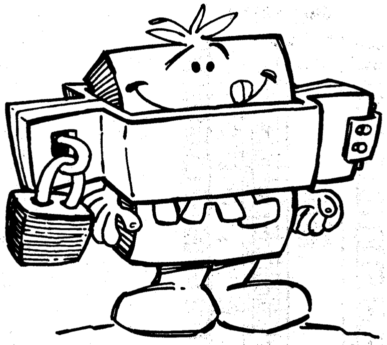
CD00960M

The PAL device family runs faster than or equal to the best of bipolar logic circuits. This makes the PAL device the ideal choice for most logical operations or control sequences which require a medium complexity and high speed. Also, in many microcomputer systems, the PAL device can be used to handle high-speed data interfaces that are not feasible for the microprocessor alone. This can be used to significantly extend the capabilities of the low-cost, low-speed NMOS microprocessors into areas formerly requiring high-cost bipolar microprocessors.

Easy Programming

The members of the PAL device family can be quickly and easily programmed using standard PROM programmers. This allows designers to use PAL devices with a minimum investment in special equipment. Many types of programmable logic, such as the PLA, require an expensive, dedicated programmer.

Secure Data



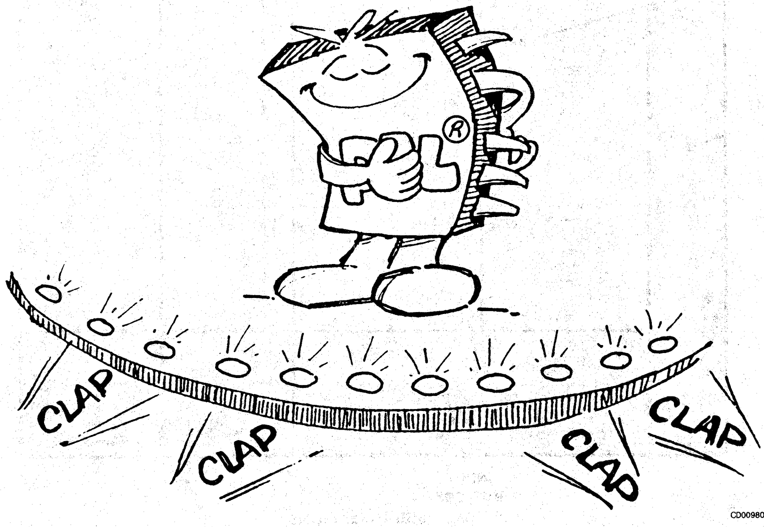
CD00970M

The PAL device verification logic can be completely disabled by blowing out a special "last link". This prevents the unauthorized copying of valuable data, and makes the PAL device perfect for use in any application where data integrity must be carefully guarded.

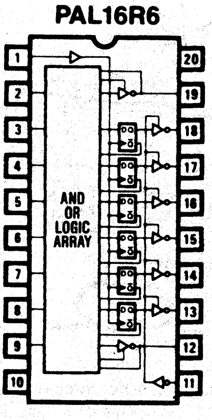
Summary

The PAL device family of logic devices offers logic designers new options in the implementation of sequential and combinatorial logic designs. The family is fast, compact, flexible, and easy to use in both new and existing designs. It promises to reduce costs in most areas of design and production with a corresponding increase in product profitability.

A Great Performer!

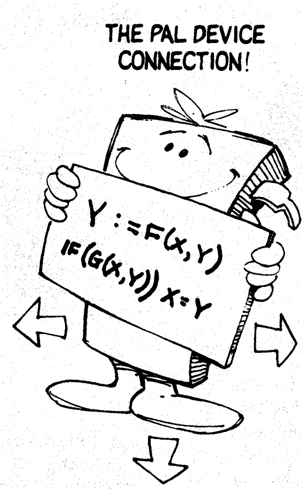


CD00980M

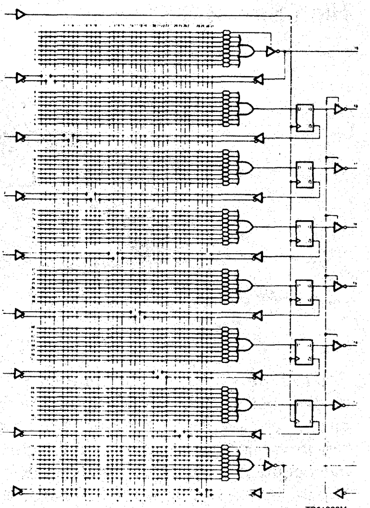


TB01670M

PAL1646 Logic Symbols

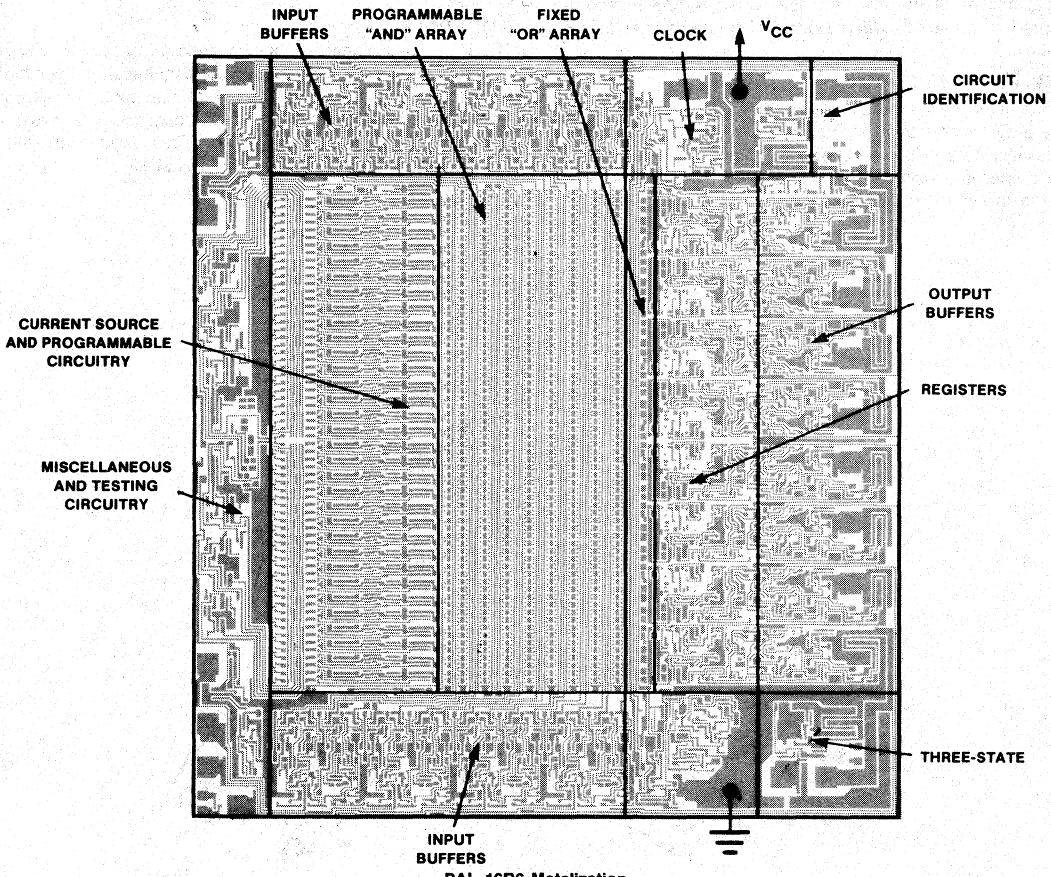


CD00990M



TB01680M

PAL16R6 Logic Diagram



PAL 16R6 Metalization

Table of Contents

PAL®(Programmable Array Logic) Devices HAL® (Hard Array Logic) Devices

Features/Benefits

- Reduces SSI/MSI chip count greater than 4 to 1
- Saves space with SKINNYDIP® and PLCC packages
- Reduces IC inventories substantially
- Expedites and simplifies prototyping and board layout
- PALASM® 2 silicon compiler provides easy design entry
- Security fuse reduces possibility of copying by competitors

Description

The PAL device family utilizes an advanced Schottky TTL process and the Bipolar PROM fusible link technology to provide user-programmable logic for replacing conventional SSI/MSI gates and flip-flops at reduced chip count.

The HAL device family utilizes standard Low-Power Schottky TTL process and automated mask pattern generation directly from logic equations to provide a semi-custom gate array for replacing conventional SSI/MSI gates and flip-flops at reduced chip count.

The PAL device lets the systems engineer "design his own chip" by blowing fusible links to configure AND and OR gates to perform his desired logic function. Complex interconnections which previously required time-consuming layout are thus "lifted" from PC board etch and placed on silicon where they can be easily modified during prototype check-out or production.

The PAL/HAL device transfer function is the familiar sum of products. Like the PROM, the PAL device has a single array of fusible links. Unlike the PROM, the PAL device is a programmable AND array driving a fixed OR array (the PROM is a fixed AND array driving a programmable OR array).

In addition the PAL/HAL device provides these options:

- Variable input/output pin ratio
- Programmable three-state outputs
- Registers with feedback
- Arithmetic capability
- Exclusive-OR gates

Unused input pins should be tied directly to V_{CC} or GND. Product terms with all fuses blown assume the logical high state, and product terms connected to both true and complement of any single input assume the logical low state. Registers consist of D-type flip-flops which are loaded on the low-to-high transition of the clock. PAL/HAL device Logic Diagrams are shown with all fuses blown, enabling the designer to use the diagrams as coding sheets. PALASM 2 software automatically generates a similar diagram, called the fuse plot.

The entire PAL device family is programmed using inexpensive conventional PROM programmers with appropriate personality and socket adapter cards. Once the PAL device is programmed and verified, two additional fuses may be blown to defeat verification. This feature gives the user a proprietary circuit which is very difficult to copy.

To design a HAL, the user first programs and debugs a PAL device using PALASM 2 software and the "PAL DEVICE DESIGN SPECIFICATION" standard format. This specification is submitted to Monolithic Memories where it is computer-processed and assigned a bit pattern number, e.g., H01234.

Monolithic Memories will provide a PAL device sample for customer qualification. The user then submits a purchase order for a HAL of the specified bit pattern number, e.g., HAL18L4 H01234. For details on ordering HAL devices, please refer to the brochure, *ProPAL, HAL, and ZHAL Devices: The Logical Solutions for Volume Programmable Logic*, available from Monolithic Memories.

Register Bypass (MegaPAL Devices)

Outputs within a bank must either be all registered or all combinatorial. Whether or not a bank of registers is bypassed depends on how the outputs are defined in the equations. A colon followed by an equal sign [: =] specifies a registered output with feedback which is updated after the low-to-high transition of the clock. An equal sign [=] defines a combinatorial output which bypasses the register. Registers are bypassed in banks of eight. Bypassing a bank of registers eliminates the feedback lines for those outputs.

Output Polarity

Output polarity is defined by comparison of the pin list and the equations. If the logic sense of a specific output in the pin list is different from the logic sense of that output as defined by its equation, the output is inverted or active low polarity. If the logic sense of a specific output in the pin list is the same as the logic sense of that output as defined by its equation, the output is active high polarity. Note that the P, RA, RS, and MegaPAL devices have programmable output polarity.

Product Term Sharing (RS, MegaPAL Devices)

The basic configuration is sixteen product terms shared between two output cells. For a typical output pair, each product term can be used by either output; but, since product term sharing is exclusive, a product term can be used by only one output, not both. If equations call for an output pair to use the same product term, two product terms are generated, one for each output. This should be taken into account when writing equations. PAL device assemblers configure product terms automatically.

Product Term Editing

A unique feature of product term sharing is the ability to edit the design after the device has been programmed. Without this feature, a new PAL device had to be programmed if the user needed to change his design. Product term editing allows the user to delete an unwanted product term and reprogram a previously unused product term to the desired fuse pattern. This feature is made possible by the product term sharing architecture. Since each product term can be routed to either output in a given pair by selecting one of two steering fuses, it is possible to blow both of the steering fuses thereby completely disabling that product term. Once disabled, that product term is powered down, saving typically 0.25mA. The desired change may now be programmed into one of the previously unused product terms corresponding to that output pair. Additional edits can be made as long as there are unused product terms for the output in question.

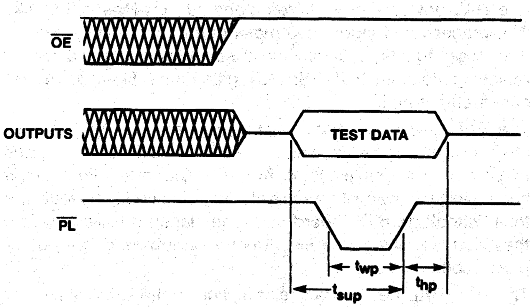
PRESET Feature (PAL64R32 only)

Register banks of eight may be PRESET to all highs on the outputs by setting the PRESET pin (PS) to a Low level. Note from the Logic Diagram that when the state of an output is High, the state of the register is Low due to the inverting tristate buffer.

TTL-Level Preload Features (RA, MegaPAL Devices)

Preload pins have been added to enable the testability of each state in state-machine design. Typically, for a modulo-n counter or a state machine there are many unreachable states for the registers. These states, and the logic which controls them are untestable without a way to "set-in" the desired starting state of the registers. In addition, long test sequences are sometimes needed to test a state machine simply to reach those starting states which are legal. Since complete logic verification is needed to ensure the proper exit from "illegal" or unused states, a way to enter these states must be provided. The ability to preload a given bank of registers is provided in this device.

To use the preload feature, several steps must be followed. First, a high level on an assertive-low output enable pin disables the outputs for that bank of registers. Next, the data to be loaded is presented at the output pins. This data is then loaded into the register by placing a low level on the PRELOAD pin, PRELOAD is asynchronous with respect to the clock.



WF00320M

Programmable Set and Reset (RA Family only)

In each SMAC, two product lines are dedicated to asynchronous set and reset. If the set product line is high, the register output becomes a logic 1. If the reset product line is high, the register output becomes a logic 0. The operation of the programmable set and reset overrides the clock. Note that set and reset are in reference to the register, independent of polarity.

Individually Programmable Register Bypass (RA Family only)

If both the set and reset product lines are high, the sum-of-products bypasses the register and appears immediately at the output, thus making the output combinatorial. This allows each output to be configured in the registered or combinatorial mode.

Programmable Clock (RA Family only)

One of the product lines in each group is connected to the clock. This provides the user with the additional flexibility of a programmable clock, so each output can be clocked independently of all the others.

Small 20 Series

	INPUTS	OUTPUTS	POLARITY	STANDARD		HALF POWER	
				t_{PD} (ns)	I_{CC} (mA)	t_{PD} (ns)	I_{CC} (mA)
PAL10H8	10	8	HIGH	35	90	65	40
PAL12H6	12	6	HIGH	35	90	65	40
PAL14H4	14	4	HIGH	35	90	65	40
PAL16H2	16	2	HIGH	35	90	65	40
PAL16C1	16	2	BOTH	40	90	65	40
PAL10L8	10	8	LOW	35	90	65	40
PAL12L6	12	6	LOW	35	90	65	40
PAL14L4	14	4	LOW	35	90	65	40
PAL16L2	16	2	LOW	35	90	65	40

Description

The Small 20 Series is made up of nine combinatorial 20-pin PAL devices. They implement simple combinatorial logic, with no feedback. Each has sixteen product terms total, divided among the outputs, with two to sixteen product terms per output.

Polarity

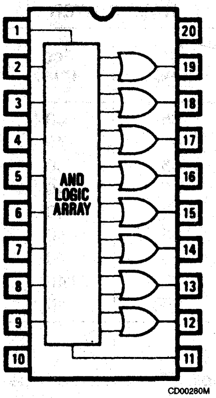
Both active high and active low versions are available for each architecture. The 16C1 offers both polarities of its single output.

Performance

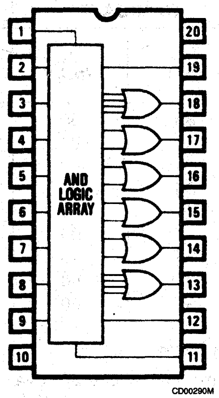
Two performance options are available. The standard series has a propagation delay (tpd) of 35 nanoseconds (ns), except for the 16C1 at 40ns. Standard supply current is 90 milliamps (mA). The half-power version consumes only 40mA, with a speed of 65ns.

DIP Pinouts

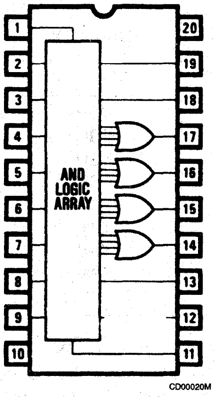
10H8/-2



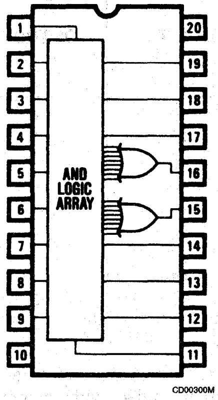
12H6/-2



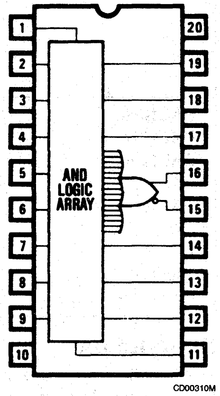
14H4/-2



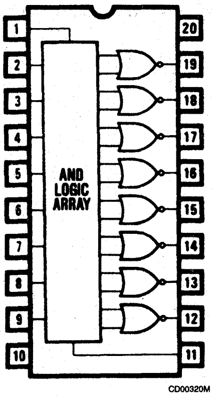
16H2/-2



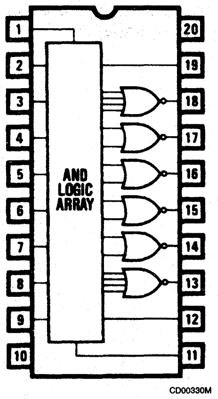
16C1/-2



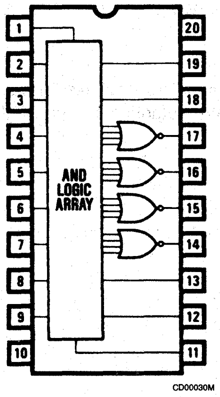
10L8/-2



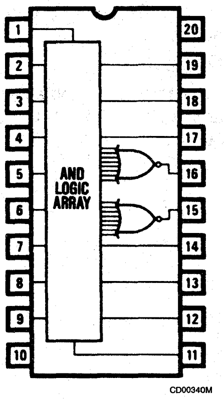
12L6/-2



14L4/-2

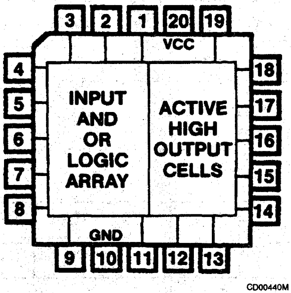


16L2/-2



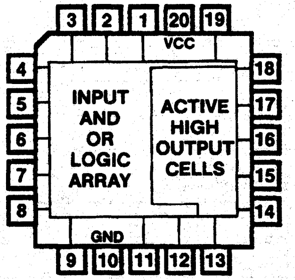
PLCC Pinouts

10H8/-2



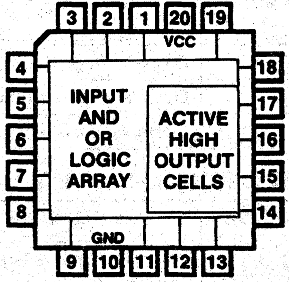
CD00440M

12H6/-2



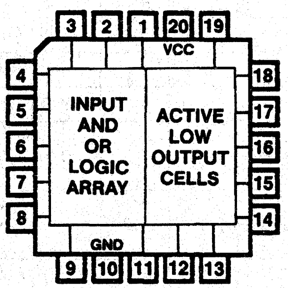
CD00060M

14H4/-2



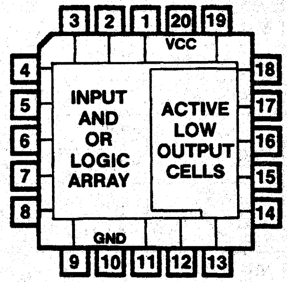
CD00450M

10L8/-2



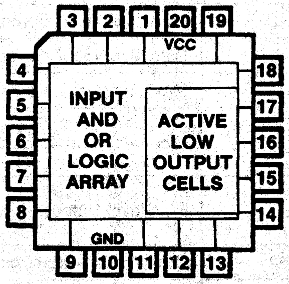
CD00460M

12L6/-2



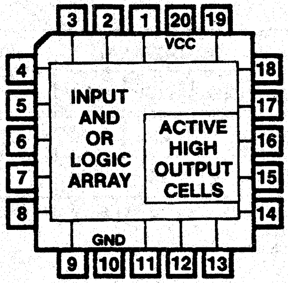
CD00070M

14L4/-2



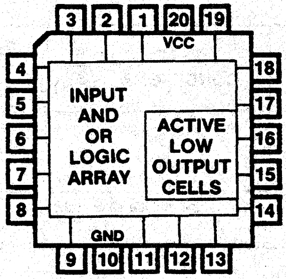
CD00470M

16H2/-2



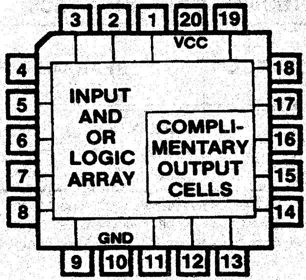
CD00480M

16L2/-2



CD00080M

16C1/-2



CD00490M

Operating Conditions

SYMBOL	PARAMETER	MILITARY			COMMERCIAL			UNIT
		MIN	TYP	MAX	MIN	TYP	MAX	
V_{CC}	Supply voltage	4.5	5	5.5	4.75	5	5.25	V
T_A	Operating free-air temperature	−55			0		75	°C
T_C	Operating case temperature			125				°C

Electrical Characteristics Over Operating Conditions

SYMBOL	PARAMETER	TEST CONDITIONS			MIN	TYP	MAX	UNIT
V_{IL}[1]	Low-level input voltage						0.8	V
V_{IH}[1]	High-level input voltage				2			V
V_{IC}	Input clamp voltage	V_{CC} = MIN		I_I = −18mA		−0.8	−1.5	V
I_{IL}	Low-level input current	V_{CC} = MAX		V_I = 0.4V		−0.02	−0.25	mA
I_{IH}	High-level input current	V_{CC} = MAX		V_I = 2.4V			25	μA
I_I	Maximum input current	V_{CC} = MAX		V_I = 5.5V			1	mA
V_{OL}	Low-level output voltage	V_{CC} = MIN	MIL	I_{OL} = 8mA		0.3	0.5	V
			COM	I_{OL} = 8mA				
V_{OH}	High-level output voltage	V_{CC} = MIN	MIL	I_{OH} = −2mA	2.4	2.8		V
			COM	I_{OH} = −3.2mA				
I_{OS}[2]	Output short-circuit current	V_{CC} = 5V		V_O = 0V	-30	−70	−130	mA
I_{CC}	Supply current	V_{CC} = MAX				55	90	mA

Switching Characteristics

SYMBOL	PARAMETER		TEST CONDITIONS	MILITARY			COMMERCIAL			UNIT
				MIN	TYP	MAX	MIN	TYP	MAX	
t_{PD}	Input or feedback to output	Except 16C1	R_1 = 560Ω R_2 = 1.1kΩ		25	45		25	35	ns
		16C1			25	45		25	40	

[1] These are absolute values with respect to the ground pin on the device and include all overshoots due to system and/or tester noise. Do not attempt to test these values without suitable equipment.

[2] No more than one output should be shorted at a time and duration of the short circuit should not exceed one second.

Operating Conditions

SYMBOL	PARAMETER	MILITARY			COMMERCIAL			UNIT
		MIN	TYP	MAX	MIN	TYP	MAX	
V_{CC}	Supply voltage	4.5	5	5.5	4.75	5	5.25	V
T_A	Operating free-air temperature	−55		125	0		75	°C

Electrical Characteristics Over Operating Conditions

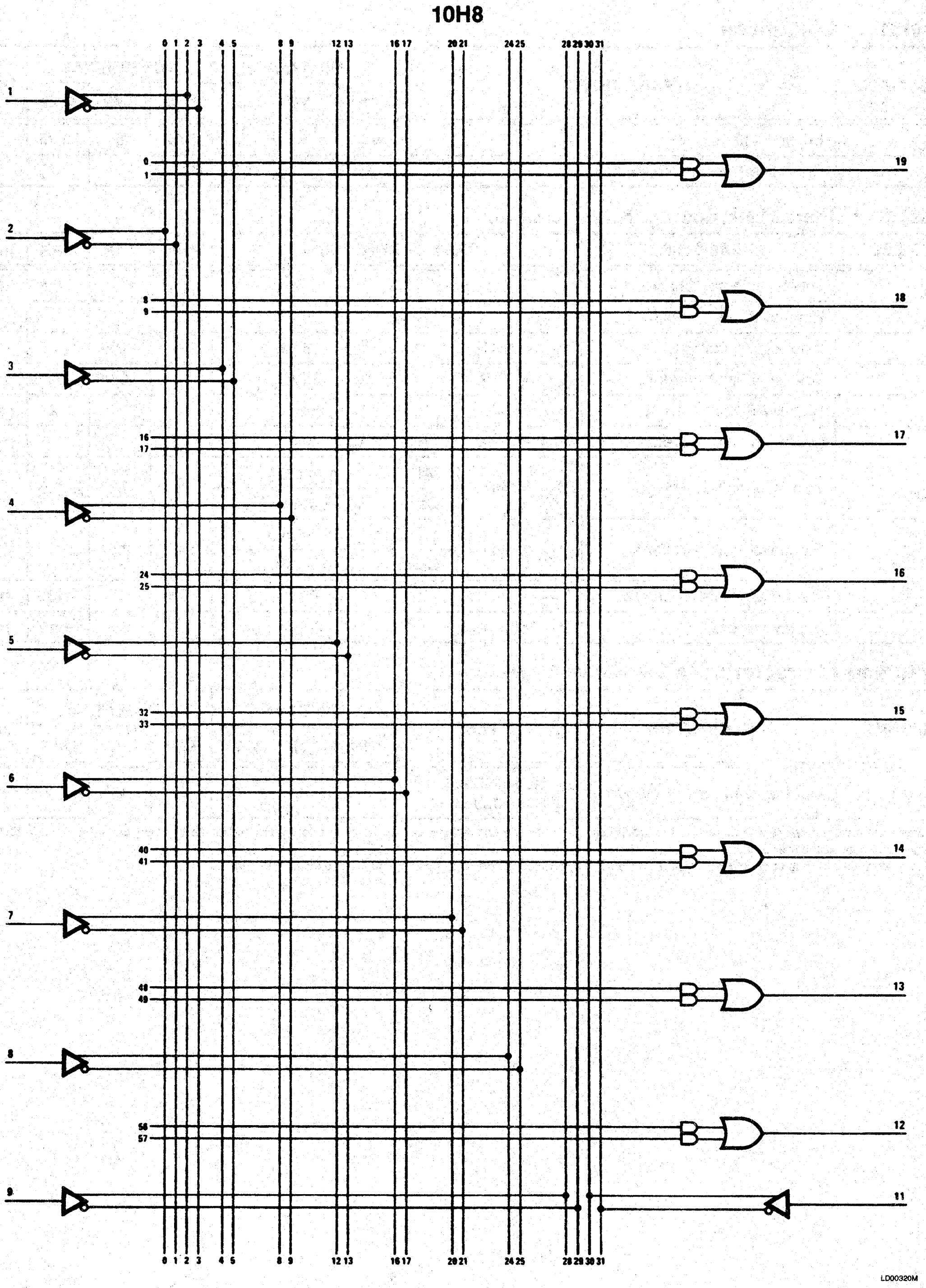
SYMBOL	PARAMETER	TEST CONDITIONS			MIN	TYP	MAX	UNIT
V_{IL} [1]	Low-level input voltage						0.8	V
V_{IH} [1]	High-level input voltage				2			V
V_{IC}	Input clamp voltage	V_{CC} = MIN		I_I = −18mA		−0.8	−1.5	V
I_{IL}	Low-level input current	V_{CC} = MAX		V_I = 0.4V		−0.02	−0.25	mA
I_{IH}	High-level input current	V_{CC} = MAX		V_I = 2.4V			25	μA
I_I	Maximum input current	V_{CC} = MAX		V_I = 5.5V			1	mA
V_{OL}	Low-level output voltage	V_{CC} = MIN	Mil	I_{OL} = 4mA		0.3	0.5	V
			Com	I_{OL} = 4mA				
V_{OH}	High-level output voltage	V_{CC} = MIN	Mil	I_{OH} = −1mA	2.4	2.8		V
			Com	I_{OH} = −1mA				
I_{OS} [2]	Output short-circuit current	V_{CC} = 5V		V_O = 0V	−30	−70	−130	mA
I_{CC}	Supply current	V_{CC} = MAX				30	45	mA

Switching Characteristics Over Operating Conditions

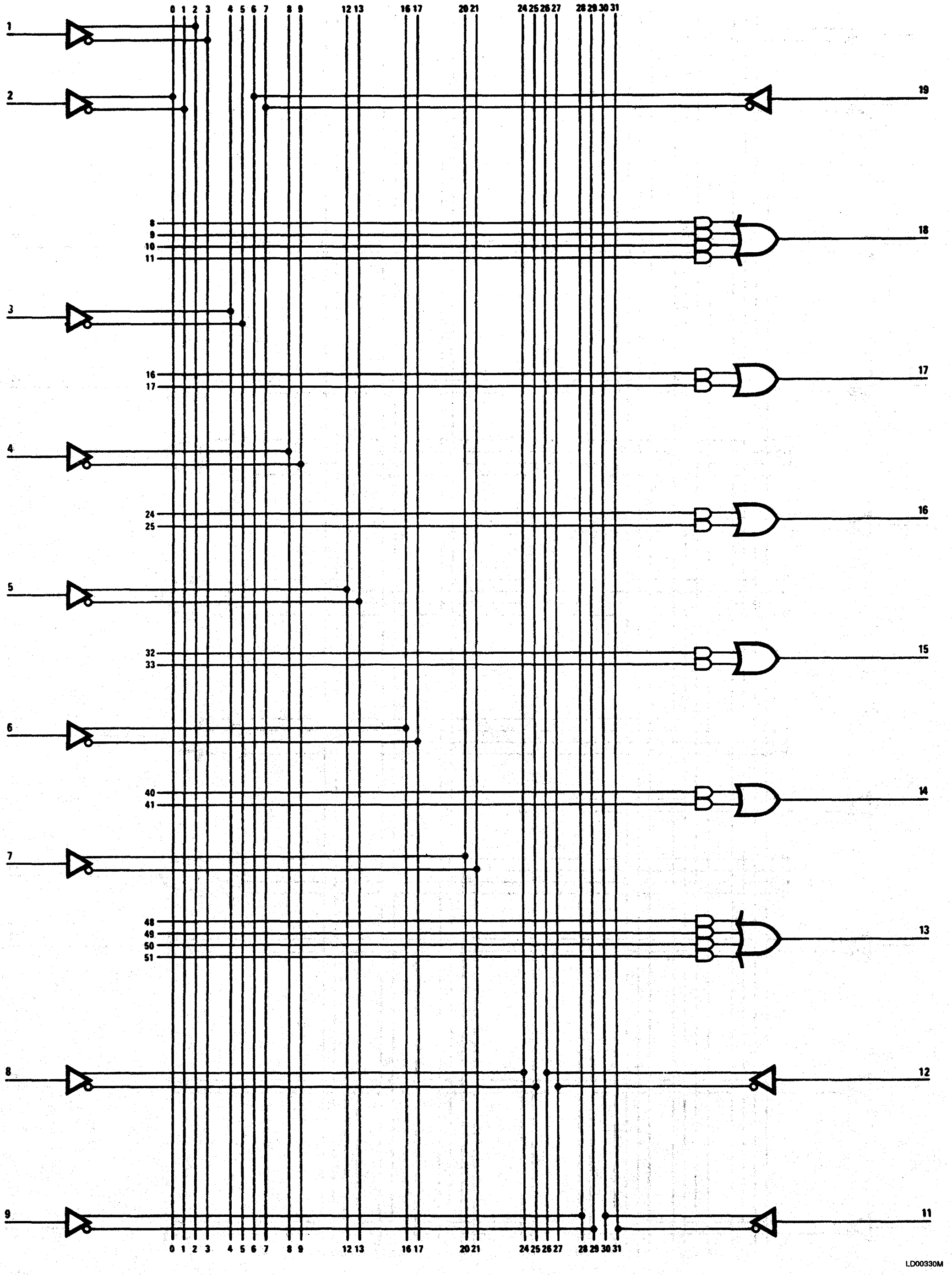
SYMBOL	PARAMETER	TEST	MILITARY			COMMERCIAL			UNIT
			MIN	TYP	MAX	MIN	TYP	MAX	
t_{PD}	Input or feedback to output	R_1 = 1.12kΩ R_2 = 2.2kΩ		45	80		45	60	ns

1. These are absolute values with respect to the ground pin on the device and include all overshoots due to system and/or tester noise. Do not attempt to test these values without suitable equipment.
2. No more than one output should be shorted at a time and duration of the short circuit should not exceed one second.

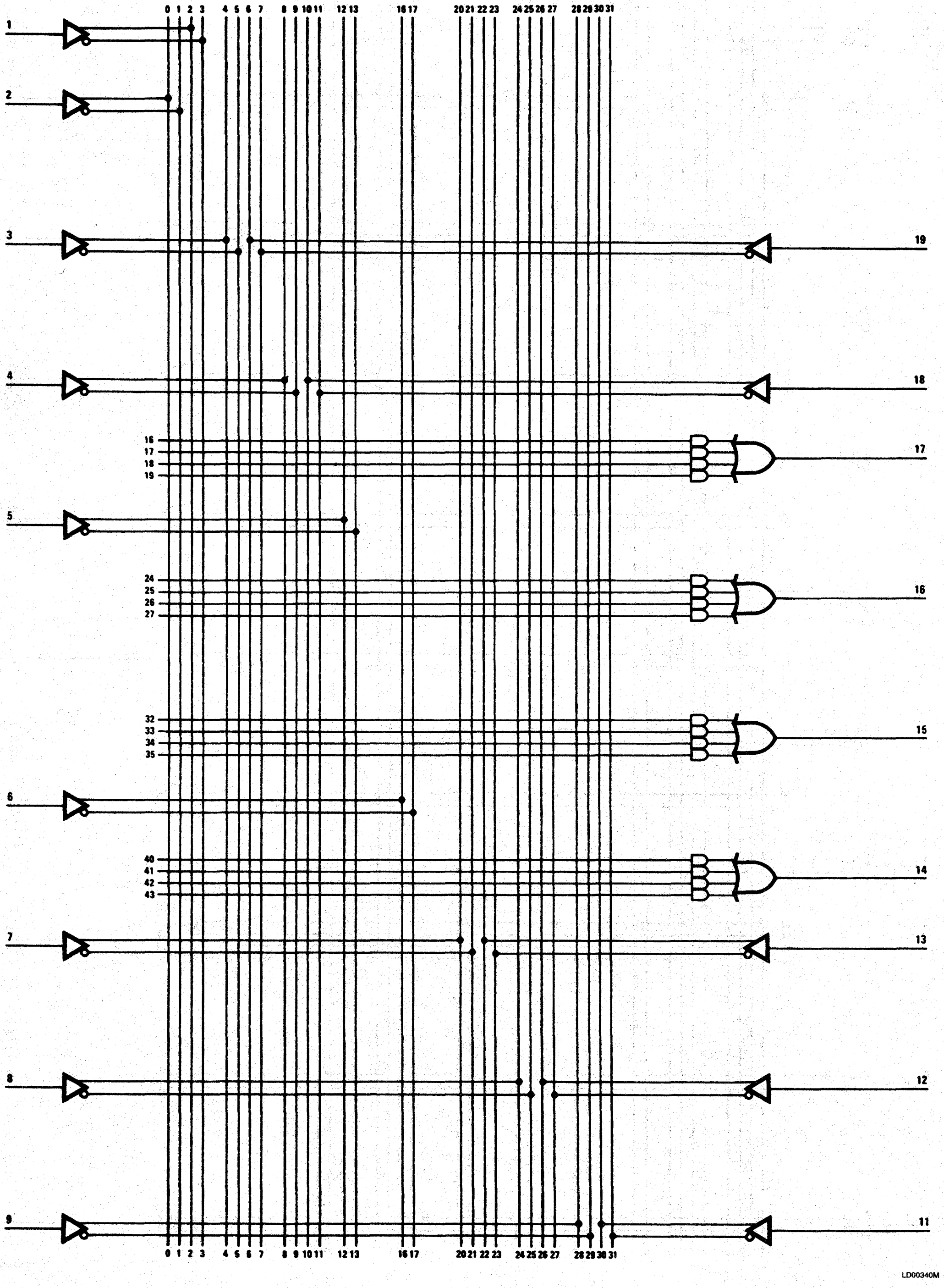
10H8 Logic Diagram



12H6

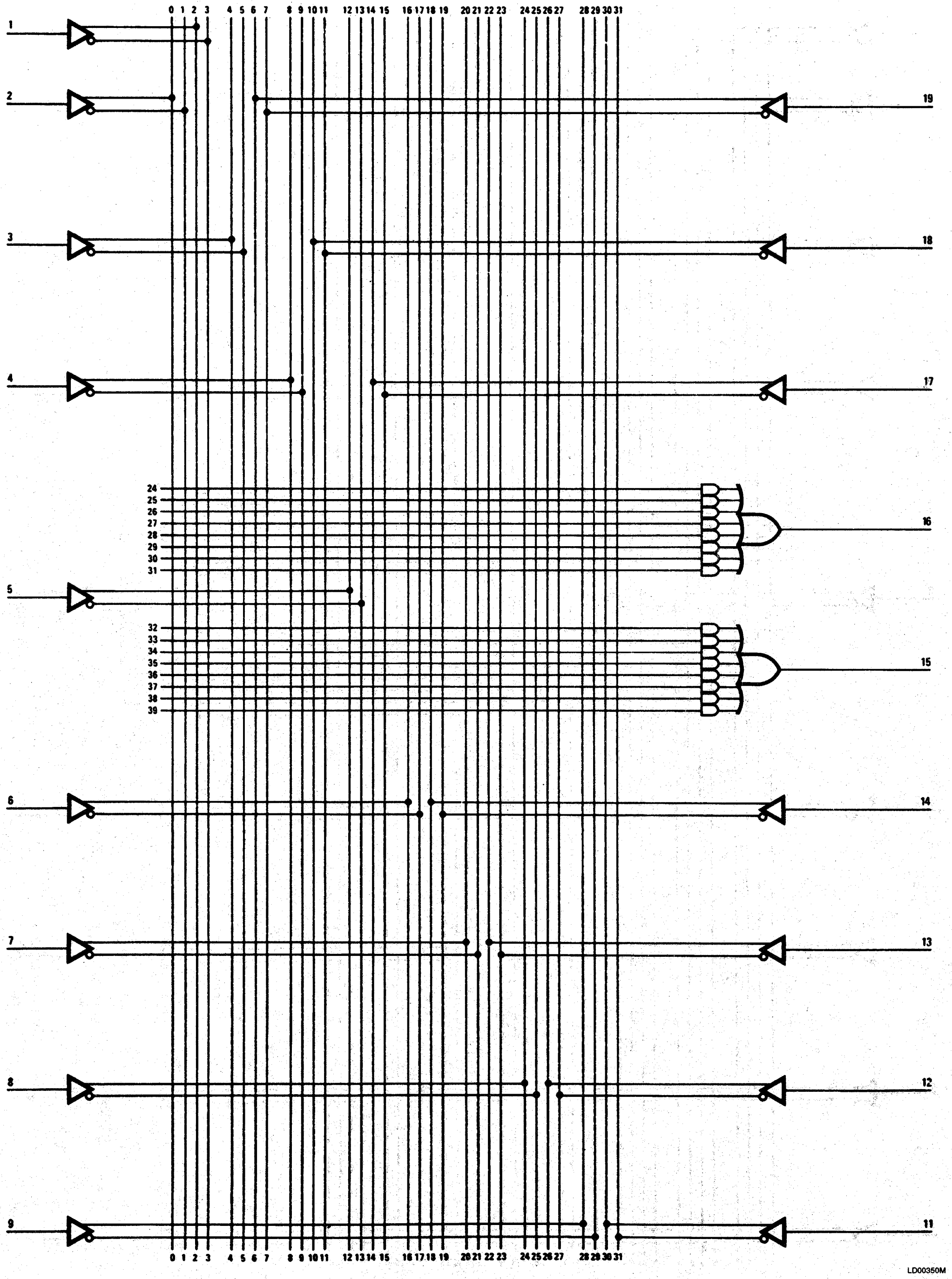

0 1 2 3 4 5 6 7 8 9 12 13 16 17 20 21 24 25 26 27 28 29 30 31
1
2
19
8
9
10
11
18
3
16
17
17
4
24
25
16
5
32
33
15
6
40
41
14
7
48
49
50
51
13
8
12
9
11
0 1 2 3 4 5 6 7 8 9 12 13 16 17 20 21 24 25 26 27 28 29 30 31
LD00330M

14H4

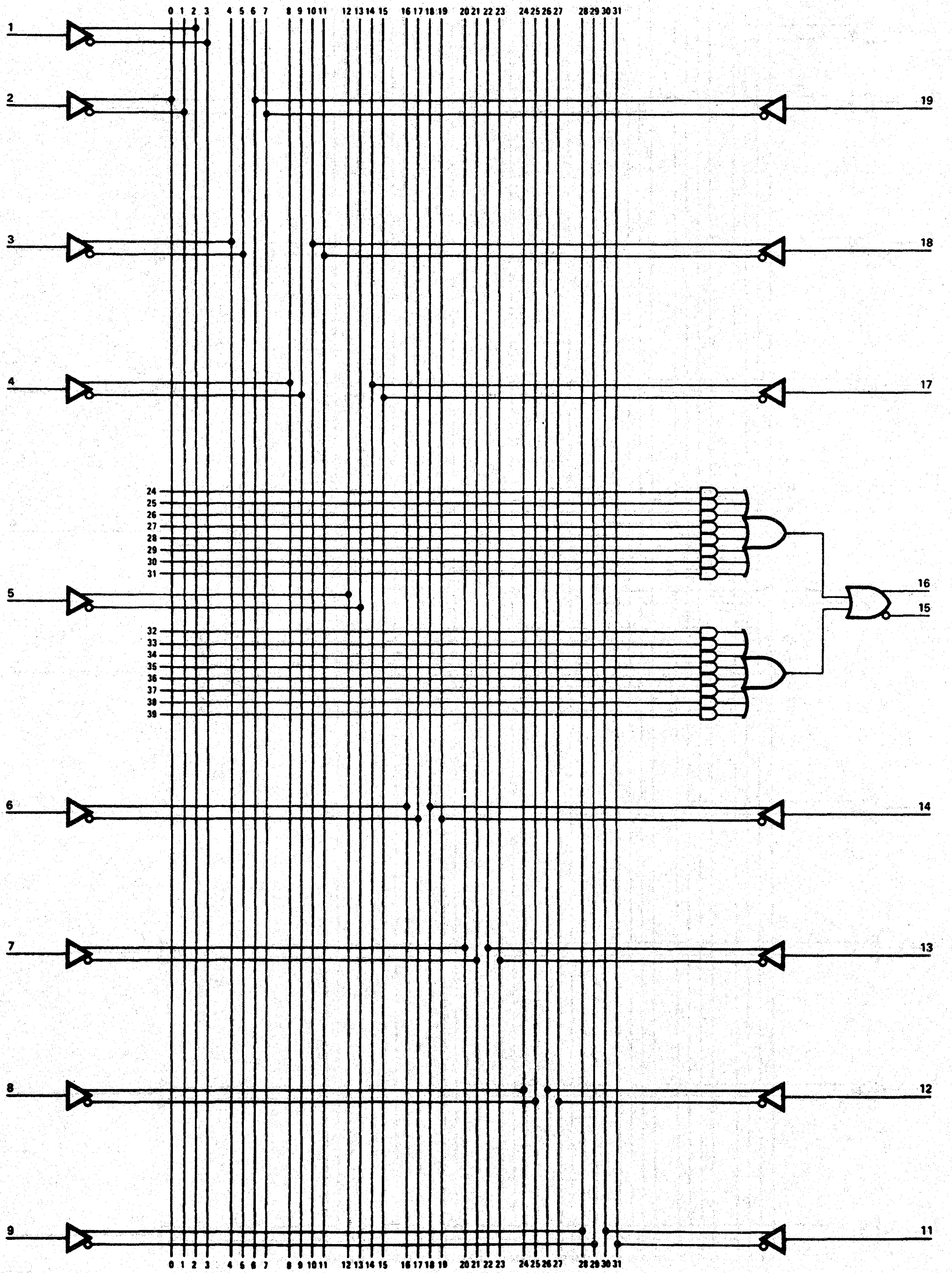

0 1 2 3 4 5 6 7 8 9 10 11 12 13 16 17 20 21 22 23 24 25 26 27 28 29 30 31
1
2
3
19
4
18
16
17
18
19
17
5
24
25
26
27
16
32
33
34
35
15
6
40
41
42
43
14
7
13
8
12
9
11
0 1 2 3 4 5 6 7 8 9 10 11 12 13 16 17 20 21 22 23 24 25 26 27 28 29 30 31
LD00340M

16H2 Logic Diagram

16H2

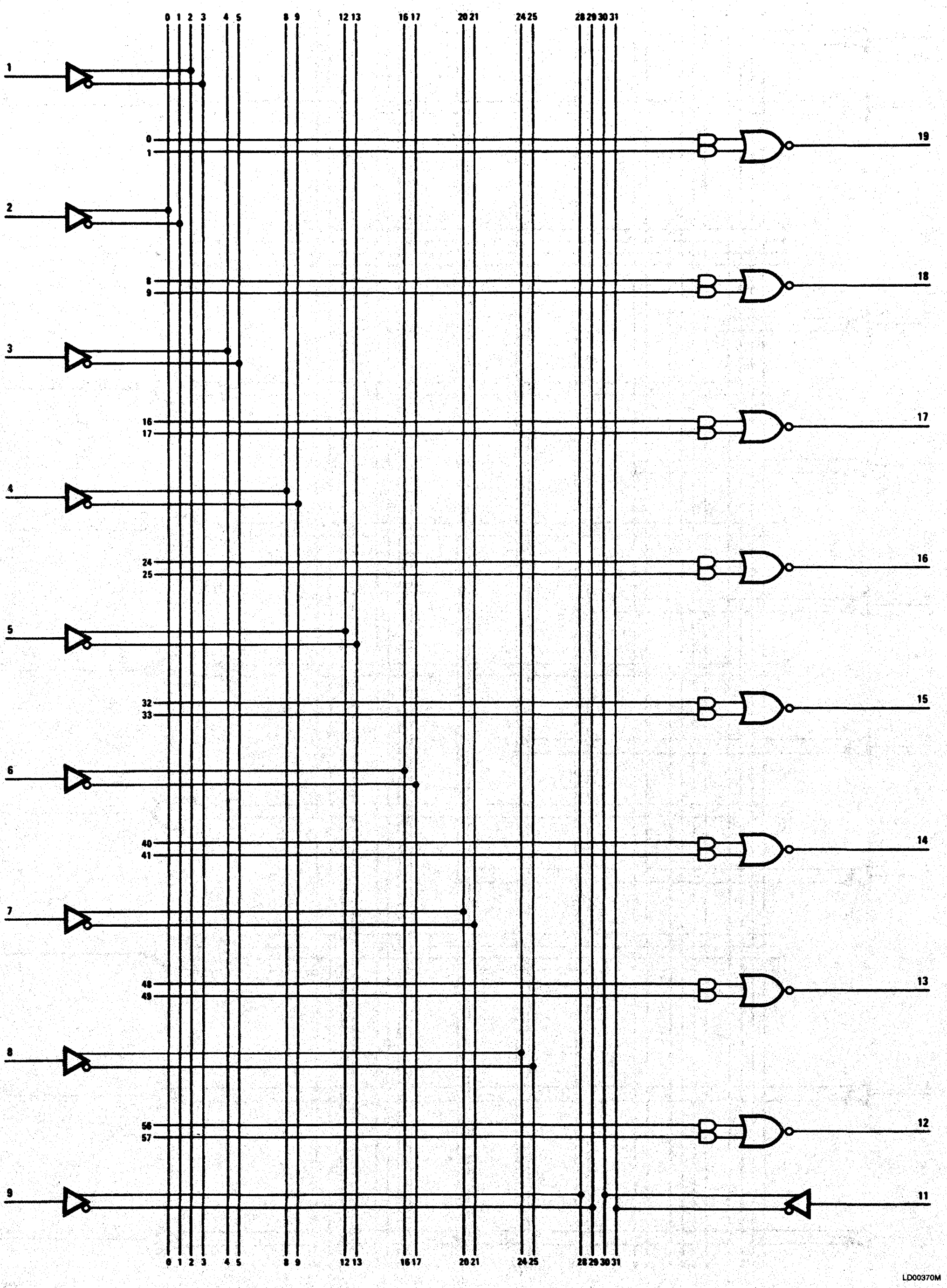


16C1



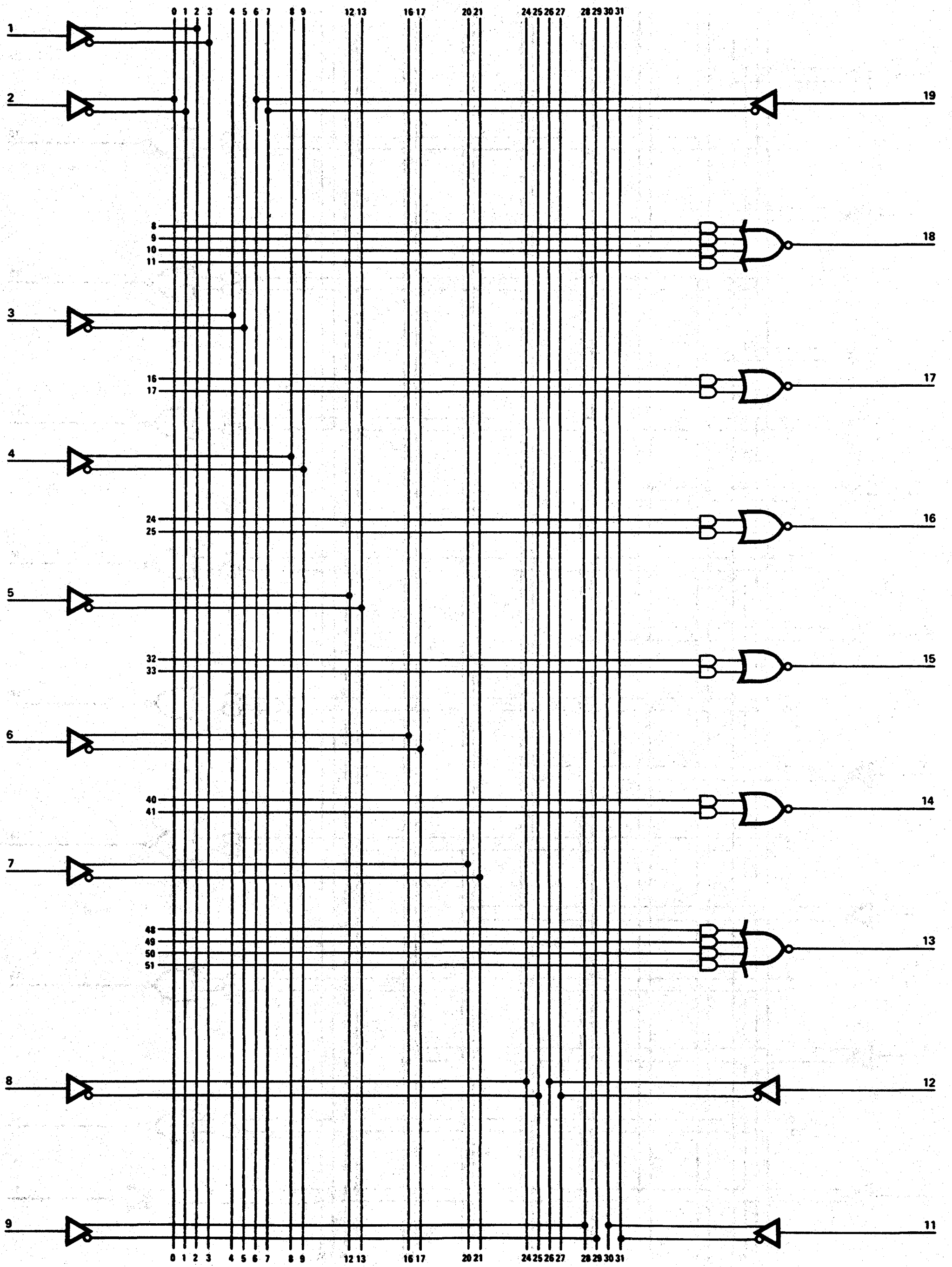
LD00360M

10L8


0 1 2 3 4 5 8 9 12 13 16 17 20 21 24 25 28 29 30 31
1
0
1
19
2
8
9
18
3
16
17
17
4
24
25
16
5
32
33
15
6
40
41
14
7
48
49
13
8
56
57
12
9
11
0 1 2 3 4 5 8 9 12 13 16 17 20 21 24 25 28 29 30 31
LD00370M

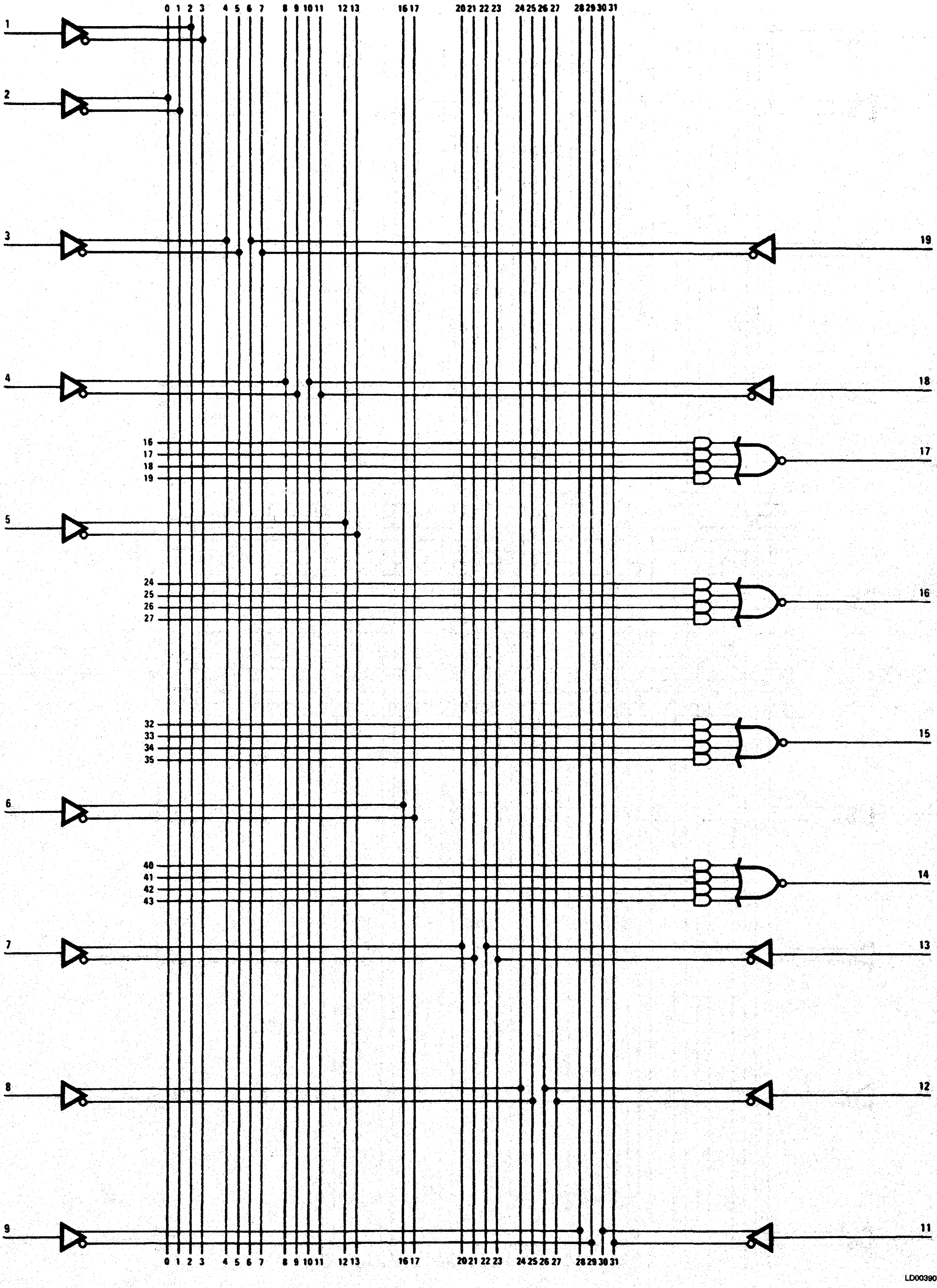
Small 20 Series
16L6 Logic Diagram

16L6

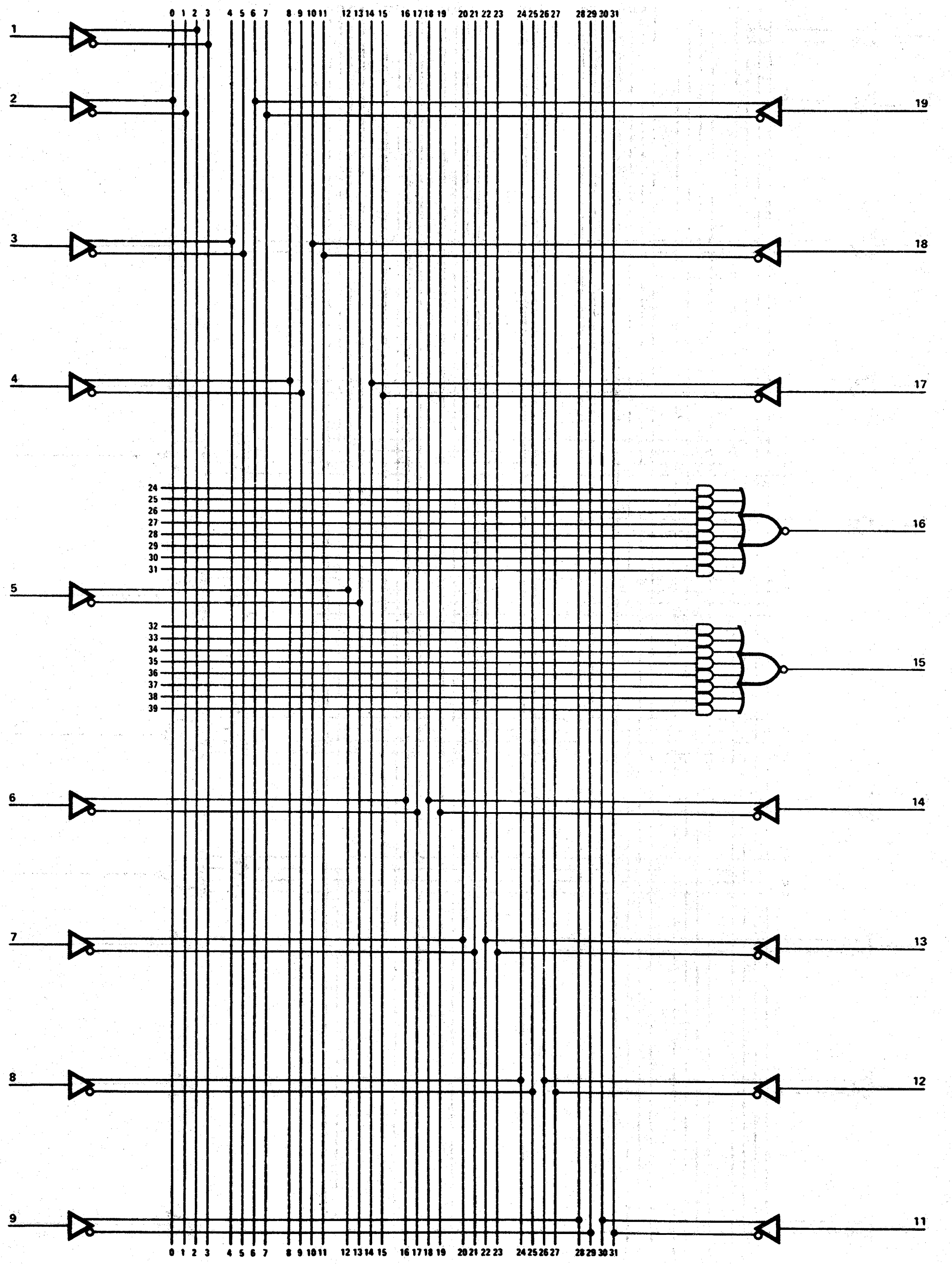


LD00380M

14L4


0 1 2 3 4 5 6 7 8 9 10 11 12 13 16 17 20 21 22 23 24 25 26 27 28 29 30 31
1
2
3
19
4
18
16
17
18
19
17
5
24
25
26
27
16
32
33
34
35
15
6
40
41
42
43
14
7
13
8
12
9
11
0 1 2 3 4 5 6 7 8 9 10 11 12 13 16 17 20 21 22 23 24 25 26 27 28 29 30 31
LD00390M

16L2



LD00400M

Medium 20 Series

	DEDICATED INPUTS	OUTPUTS	
		COMBINATORIAL	REGISTERED
PAL16L8	10	8 (6 I/O)	0
PAL16R8	8	0	8
PAL16R6	8	2 I/O	6
PAL16R4	8	4 I/O	4

Description

The Medium 20 Series offers the four most popular PAL device architectures. It also provides the fastest PAL devices in the industry.

The Medium 20 Series consists of four devices, each with sixteen array inputs and eight outputs. The devices have either 0, 4, 6, or 8 registered outputs, with the remaining being combinatorial. Each of the registered outputs feeds back into the array, for sequential designs. The combinatorial outputs also feed back into the array, except for two of the outputs on the 16L8. This feedback allows the output to also operate as an input if the output is disabled.

Enable

The combinatorial outputs are enabled by a product term. The registered outputs are enabled by a common enable pin.

Polarity

All outputs are active low.

Performance

Several speed/power versions are available:

Suffix	t_{PD} (ns)	I_{CC} (mA)
(standard)	35	180
A	25	180
A-2	35	90
A-4	55	50
B or BP*	15	180
B-2	25	90
B-4	35	55
D	10	180

* contact Monolithic Memories for datasheet

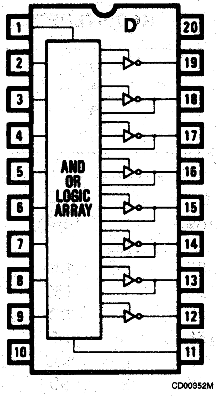
The D Series offers the fastest TTL programmable logic devices in the industry, at 10ns tpd.

Preload and Power-up Reset

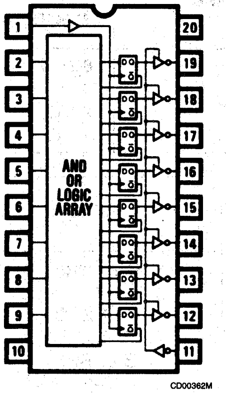
The BP Series offers register preload for device testability. The registers can be preloaded from the outputs by using supervoltages (see waveforms at end of section) in order to simplify functional testing. The PAL20BP Series also offers Power-up Reset, whereby the registers power up to a logic LOW, setting the active-low outputs to a logic HIGH.

DIP Pinouts

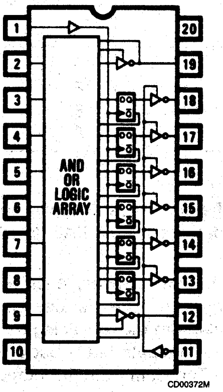
16L8/A/A-2/A-4/
B/B-2/B-4/D



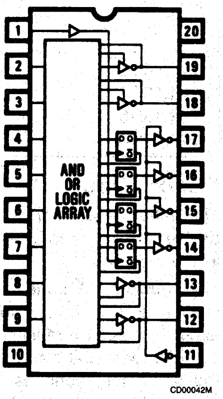
16R8/A/A-2/A-4/
B/B-2/B-4/D



16R6/A/A-2/A-4/
B/B-2/B-4/D

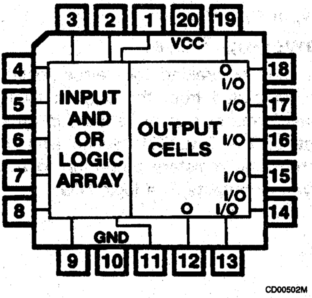


16R4/A/A-2/A-4/
B/B-2/B-4/D

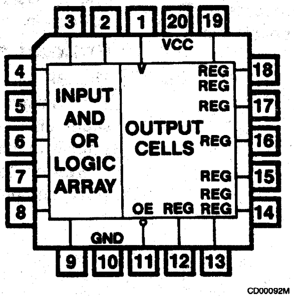


PLCC Pinouts

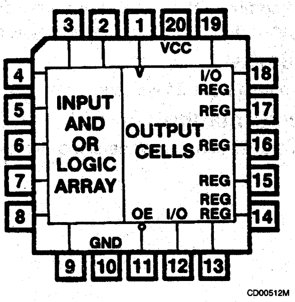
16L8/A/A-2/A-4/
B/B-2/B-4/D



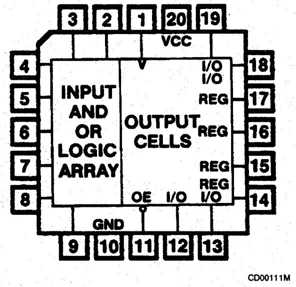
16R8/A/A-2/A-4/
B/B-2/B-4/D



16R6/A/A-2/A-4/
B/B-2/B-4/D



16R4/A/A-2/A-4/
B/B-2/B-4/D



Operating Conditions

SYMBOL	PARAMETER		COMMERCIAL[1]			UNIT
			MIN	TYP	MAX	
V_{CC}	Supply voltage		4.75	5	5.25	V
t_w	Width of clock	Low	25	10		ns
		High	25	10		
t_{su}	Set up time from input or feedback to clock	16R8, 16R6, 16R4	35	25		ns
t_h	Hold time		0	−15		ns
T_A	Operating free-air temperature		0		75	°C
T_C	Operating case temperature					°C

Electrical Characteristics

SYMBOL	PARAMETER	TEST CONDITIONS			MIN	TYP	MAX	UNIT
V_{IL}[2]	Low-level input voltage						0.8	V
V_{IH}[2]	High-level input voltage				2			V
V_{IC}	Input clamp voltage	V_{CC} = MIN		I_I = −18mA		−0.8	−1.5	V
I_{IL}[3]	Low-level input current	V_{CC} = MAX		V_I = 0.4V		−0.02	−0.25	mA
I_{IH}[3]	High-level input current	V_{CC} = MAX		V_I = 2.4V			25	μA
I_I	Maximum input current	V_{CC} = MAX		V_I = 5.5V			1	mA
V_{OL}	Low-level output voltage	V_{CC} = MIN	Com	I_{OL} = 24mA		0.3	0.5	V
V_{OH}	High-level output voltage	V_{CC} = MIN	Com	I_{OH} = −3.2mA	2.4	2.8		V
I_{OZL}[3]	Off-state output current	V_{CC} = MAX		V_O = 0.4V			−100	μA
I_{OZH}[3]				V_O = 2.4V			100	μA
I_{OS}[4]	Output short-circuit current	V_{CC} = 5V		V_O = 0V	−30	−70	−130	mA
I_{CC}	Supply current	V_{CC} = MAX	16R4, 16R6, 16R8, 16L8			120	180	mA

Switching Characteristics Over Operating Conditions

SYMBOL	PARAMETER		TEST CONDITIONS	COMMERCIAL			UNIT
				MIN	TYP	MAX	
t_{PD}	Input or feedback to output	16R6, 16R4, 16L8			25	35	ns
t_{CLK}	Clock to output or feedback				15	25	ns
t_{PZX}	Pin 11 to output enable except 16L8				15	25	ns
t_{PXZ}	Pin 11 to output disable except 16L8		R_1 = 200Ω		15	25	ns
t_{PZX}	Input to output enable	16R6, 16R4, 16L8	R_2 = 390Ω		25	35	ns
t_{PXZ}	Input to output disable	16R6, 16R4, 16L8			25	35	ns
f_{MAX}	Maximum frequency	16R8, 16R6, 16R4		16	25		MHz

[1] The PAL20 Series is designed to operate over the full military operating conditions. For availability and specifications, contact Monolithic Memories.

[2] These are absolute voltages with respect to the ground pin on the device and include all overshoots due to system and/or tester noise. Do not attempt to test these values without suitable equipment.

[3] I/O pin leakage is the worst case of I_{IL} and I_{OZL} (or I_{IH} and I_{OZH}).

[4] No more than one output should be shorted at a time and duration of the short circuit should not exceed one second.

Operating Conditions

SYMBOL	PARAMETER		MILITARY			COMMERCIAL			UNIT
			MIN	TYP	MAX	MIN	TYP	MAX	
V_{CC}	Supply voltage		4.5	5	5.5	4.75	5	5.25	V
t_w	Width of clock	Low	20	10		15	10		ns
		High	20	10		15	10		
t_{su}	Set up time from input or feedback to clock	16R8A, 16R6A, 16R4A	30	15		25	15		ns
t_h	Hold time		0	−10		0	−10		ns
T_A	Operating free-air temperature		−55			0		75	°C
T_C	Operating case temperature				125				°C

Electrical Characteristics Over Operating Conditions

SYMBOL	PARAMETER	TEST CONDITIONS			MIN	TYP	MAX	UNIT
V_{IL}[1]	Low-level input voltage						0.8	V
V_{IH}[1]	High-level input voltage				2			V
V_{IC}	Input clamp voltage	V_{CC} = MIN		I_I = −18mA		−0.8	−1.5	V
I_{IL}[2]	Low-level input current	V_{CC} = MAX		V_I = 0.4V		−0.02	−0.25	mA
I_{IH}[2]	High-level input current	V_{CC} = MAX		V_I = 2.4V			25	µA
I_I	Maximum input current	V_{CC} = MAX		V_I = 5.5V			1	mA
V_{OL}	Low-level output voltage	V_{CC} = MIN	Mil	I_{OL} = 12mA		0.3	0.5	V
			Com	I_{OL} = 24mA				
V_{OH}	High-level output voltage	V_{CC} = MIN	Mil	I_{OH} = −2mA	2.4	2.8		V
			Com	I_{OH} = −3.2mA				
I_{OZL}[2]	Off-state output current	V_{CC} = MAX		V_O = 0.4V			−100	µA
I_{OZH}[2]				V_O = 2.4V			100	µA
I_{OS}[3]	Output short-circuit current	V_{CC} = 5V		V_O = 0V	−30	−70	−130	mA
I_{CC}	Supply current	V_{CC} = MAX				120	180	mA

1. These are absolute voltages with respect to the ground pin on the device and include all overshoots due to system and/or tester noise. Do not attempt to test these values without suitable equipment.
2. I/O pin leakage is the worst case of I_{IL} and I_{OZL} (or I_{IH} and I_{OZH}).
3. No more than one output should be shorted at a time and duration of the short circuit should not exceed one second.

Switching Characteristics Over Operating Conditions

SYMBOL	PARAMETER		TEST CONDITIONS	MILITARY			COMMERCIAL			UNIT
				MIN	TYP	MAX	MIN	TYP	MAX	
t_{PD}	Input or feedback to output	16R6A, 16R4A, 16L8A	$R_1 = 200\Omega$ $R_2 = 390\Omega$		15	30		15	25	ns
t_{CLK}	Clock to output or feedback				10	20		10	15	ns
t_{PZX}	Pin 11 to output enable except 16L8A				10	25		10	20	ns
t_{PXZ}	Pin 11 to output disable except 16L8A				11	25		11	20	ns
t_{PZX}	Input to output enable	16R6A, 16R4A, 16L8A			10	30		10	25	ns
t_{PXZ}	Input to output disable	16R6A, 16R4A, 16L8A			13	30		13	25	ns
f_{MAX}	Maximum frequency	16R8A, 16R6A, 16R4A		20	40		28.5	40		MHz

Operating Conditions

SYMBOL	PARAMETER		MILITARY			COMMERCIAL			UNIT
			MIN	TYP	MAX	MIN	TYP	MAX	
V_{CC}	Supply voltage		4.5	5	5.5	4.75	5	5.25	V
t_w	Width of clock	Low	25	10		25	10		ns
		High	25	10		25	10		
t_{su}	Set up time from input or feedback to clock	16R6A-2, 16R4A-2, 16R8A-2	50	25		35	25		ns
t_h	Hold time		0	−15		0	−15		ns
T_A	Operating free-air temperature		−55		125	0		75	°C

Electrical Characteristics Over Operating Conditions

SYMBOL	PARAMETER	TEST CONDITIONS			MIN	TYP	MAX	UNIT
V_{IL}[1]	Low-level input voltage						0.8	V
V_{IH}[1]	High-level input voltage				2			V
V_{IC}	Input clamp voltage	V_{CC} = MIN		I_I = −18mA		−0.8	−1.5	V
I_{IL}[2]	Low-level input current	V_{CC} = MAX		V_I = 0.4V		−0.02	−0.25	mA
I_{IH}[2]	High-level input current	V_{CC} = MAX		V_I = 2.4V			25	μA
I_I	Maximum input current	V_{CC} = MAX		V_I = 5.5V			1	mA
V_{OL}	Low-level output voltage	V_{CC} = MIN	Mil	I_{OL} = 12mA		0.3	0.5	V
			Com	I_{OL} = 24mA				
V_{OH}	High-level output voltage	V_{CC} = MIN	Mil	I_{OH} = −2mA	2.4	2.8		V
			Com	I_{OH} = −3.2mA				
I_{OZL}[2]	Off-state output current	V_{CC} = MAX		V_O = 0.4V			−100	μA
I_{OZH}[2]				V_O = 2.4V			100	μA
I_{OS}[3]	Output short-circuit current	V_{CC} = 5V		V_O = 0V	−30	−70	−130	mA
I_{CC}	Supply current	V_{CC} = MAX				60	90	mA

Switching Characteristics Over Operating Conditions

SYMBOL	PARAMETER		TEST CONDITIONS	MILITARY			COMMERCIAL			UNIT
				MIN	TYP	MAX	MIN	TYP	MAX	
t_{PD}	Input or feedback to output	16L8A-2, 16R6A-2, 16R4A-2	R_1 = 200Ω R_2 = 390Ω		25	50		25	35	ns
t_{CLK}	Clock to output or feedback				15	25		15	25	ns
$t_{PXZ/ZX}$	Pin 11 to output disable/enable except 16L8A-2				15	25		15	25	ns
t_{PZX}	Input to output enable	16L8A-2, 16R6A-2, 16R4A-2			25	45		25	35	ns
t_{PXZ}	Input to output disable	16L8A-2, 16R6A-2, 16R4A-2			25	45		25	35	ns
f_{MAX}	Maximum frequency	16R8A-2, 16R6A-2, 16R4A-2		14	25		16	25		MHz

[1] These are absolute voltages with respect to the ground pin on the device and include all overshoots due to system and/or tester noise. Do not attempt to test these values without suitable equipment.

[2] I/O pin leakage is the worst case of I_{IL} and I_{OZL} (or I_{IH} and I_{OZH}).

[3] No more than one output should be shorted at a time and duration of the short circuit should not exceed one second.

Operating Conditions

SYMBOL	PARAMETER		MILITARY			COMMERCIAL			UNIT
			MIN	TYP	MAX	MIN	TYP	MAX	
V_{CC}	Supply voltage		4.5	5	5.5	4.75	5	5.25	V
t_w	Width of clock	Low	40	20		30	20		ns
		High	40	20		30	20		
t_{su}	Set up time from input or feedback to clock	16R8A-4, 16R6A-4, 16R4A-4	90	45		60	45		ns
t_h	Hold time		0	−15		0	−15		ns
T_A	Operating free-air temperature		−55		125	0		75	°C

Electrical Characteristics Over Operating Conditions

SYMBOL	PARAMETER	TEST CONDITIONS			MIN	TYP	MAX	UNIT
V_{IL}[1]	Low-level input voltage						0.8	V
V_{IH}[1]	High-level input voltage				2			V
V_{IC}	Input clamp voltage	V_{CC} = MIN		I_I = −18mA		−0.8	−1.5	V
I_{IL}[2]	Low-level input current	V_{CC} = MAX		V_I = 0.4V		−0.02	−0.25	mA
I_{IH}[2]	High-level input current	V_{CC} = MAX		V_I = 2.4V			25	μA
I_I	Maximum input current	V_{CC} = MAX		V_I = 5.5V			1	mA
V_{OL}	Low-level output voltage	V_{CC} = MIN	Mil	I_{OL} = 4mA		0.3	0.5	V
			Com	I_{OL} = 8mA				
V_{OH}	High-level output voltage	V_{CC} = MIN	Mil	I_{OH} = −1mA	2.4	2.8		V
			Com	I_{OH} = −1mA				
I_{OZL}[2]	Off-state output current	V_{CC} = MAX		V_O = 0.4V			−100	μA
I_{OZH}[2]				V_O = 2.4V			100	μA
I_{OS}[3]	Output short-circuit current	V_{CC} = 5V		V_O = 0V	−30	−70	−130	mA
I_{CC}	Supply current	V_{CC} = MAX				30	50	mA

Switching Characteristics Over Operating Conditions

SYMBOL	PARAMETER		TEST CONDITIONS	MILITARY			COMMERCIAL			UNIT
				MIN	TYP	MAX	MIN	TYP	MAX	
t_{PD}	Input or feedback to output	16R6A-4, 16R4A-4, 16L8A-4	R_1 = 800Ω R_2 = 1.56kΩ		35	75		35	55	ns
t_{CLK}	Clock to output or feedback				20	45		20	35	ns
$t_{PXZ/ZX}$	Pin 11 to output disable/enable – except 16L8A-4				15	40		15	30	ns
t_{PZX}	Input to output enable	16R6A-4, 16R4A-4, 16L8A-4			30	65		30	50	ns
t_{PXZ}	Input to output disable	16R6A-4, 16R4A-4, 16L8A-4			30	65		30	50	ns
f_{MAX}	Maximum frequency	16R8A-4, 16R6A-4, 16R4A-4		8	18		11	18		MHz

[1] These are absolute voltages with respect to the ground pin on the device and include all overshoots due to system and/or tester noise. Do not attempt to test these values without suitable equipment.

[2] I/O pin leakage is the worst case of I_{IL} and I_{OZL} (or I_{IH} and I_{OZH}).

[3] No more than one output should be shorted at a time and duration of the short circuit should not exceed one second.

Operating Conditions

SYMBOL	PARAMETER		COMMERCIAL[1]			UNIT
			MIN	TYP	MAX	
V_{CC}	Supply voltage		4.75	5	5.25	V
t_w	Width of clock	Low	10	6		ns
		High	10	5		
t_{su}	Setup time from input or feedback to clock	16R8B 16R6B 16R4B	15	10		ns
t_h	Hold time		0	−10		ns
T_A	Operating free-air temperature		0	25	75	°C

Electrical Characteristics Over Operating Conditions

SYMBOL	PARAMETER	TEST CONDITION		COMMERCIAL			UNIT
				MIN	TYP	MAX	
V_{IL}[2]	Low-level input voltage					0.8	V
V_{IH}[2]	High-level input voltage			2			V
V_{IC}	Input clamp voltage	V_{CC} = MIN	I_I = −18mA		−0.8	−1.5	V
I_{IL}[3]	Low-level input current	V_{CC} = MAX	V_I = 0.4V		−0.02	−0.25	mA
I_{IH}[3]	High-level input current	V_{CC} = MAX	V_I = 2.4V			25	μA
I_I	Maximum input current	V_{CC} = MAX	V_I = 5.5V			1	mA
V_{OL}	Low-level output voltage	V_{CC} = MIN	I_{OL} = 24mA		0.3	0.5	V
V_{OH}	High-level output voltage	V_{CC} = MIN	I_{OH} = −3.2mA	2.4	2.8		V
I_{OZL}[3]	Off-state output current	V_{CC} = MAX	V_O = 0.4V			−100	μA
I_{OZH}[3]			V_O = 2.4V			100	μA
I_{OS}[4]	Output short-circuit current	V_{CC} = 5V	V_O = 0V	−30	−70	−130	mA
I_{CC}	Supply current	V_{CC} = MAX			120	180	mA

Switching Characteristics Over Operating Conditions

SYMBOL	PARAMETER		TEST CONDITIONS	COMMERCIAL			UNIT
				MIN	TYP	MAX	
t_{PD}	16L8B, 16R4B, 16R6B input or feedback to output		Commercial R_1 = 200Ω R_2 = 390Ω		12	15	ns
t_{CLK}	Clock to output or feedback except 16L8B				8	12	ns
t_{PZX}	Pin 11 to output enable except 16L8B				10	15	ns
t_{PXZ}	Pin 11 to output disable except 16L8B				10	15	ns
t_{PZX}	Input to output enable	16R6B, 16R4B, and 16L8B			12	22	ns
t_{PXZ}	Input to output disable	16R6B, 16R4B, and 16L8B			12	15	ns
f_{MAX}	16R8B, 16R6B, 16R4B	Feedback		37	45		MHz
	Maximum frequency	No feedback		50	55		

1. The PAL20B Series is designed to operate over the full military operating conditions. For availability and specifications, contact Monolithic Memories.
2. These are absolute voltages with respect to the ground pin on the device and include all overshoots due to system and/or tester noise. Do not attempt to test these values without suitable equipment.
3. I/O pin leakage is the worst case of I_{IL} and I_{OZL} (or I_{IH} and I_{OZH}).
4. No more than one output should be shorted at a time and duration of the short circuit should not exceed one second.

Operating Conditions

SYMBOL	PARAMETER		COMMERCIAL[1]			UNIT
			MIN	TYP	MAX	
V_{CC}	Supply voltage		4.75	5	5.25	V
t_w	Width of clock	Low	15	10		ns
		High	15	10		
t_{su}	Setup time from input or feedback to clock	16R8B-2 16R6B-2 16R4B-2	25	15		ns
t_h	Hold time		0	−10		ns
T_A	Operating free-air temperature		0	25	75	°C

Electrical Characteristics Over Operating Conditions

SYMBOL	PARAMETER	TEST CONDITION		COMMERCIAL			UNIT
				MIN	TYP	MAX	
V_{IL}[2]	Low-level input voltage					0.8	V
V_{IH}[2]	High-level input voltage			2			V
V_{IC}	Input clamp voltage	V_{CC} = MIN	I_I = −18mA		−0.8	−1.5	V
I_{IL}[3]	Low-level input current	V_{CC} = MAX	V_I = 0.4V		−0.02	−0.25	mA
I_{IH}[3]	High-level input current	V_{CC} = MAX	V_I = 2.4V			25	µA
I_I	Maximum input current	V_{CC} = MAX	V_I = 5.5V			1	mA
V_{OL}	Low-level output voltage	V_{CC} = MIN	I_{OL} = 24mA		0.3	0.5	V
V_{OH}	High-level output voltage	V_{CC} = MIN	I_{OH} = −3.2mA	2.4	2.8		V
I_{OZL}[3]	Off-state output current	V_{CC} = MAX	V_O = 0.4V			−100	µA
I_{OZH}[3]			V_O = 2.4V			100	µA
I_{OS}[4]	Output short-circuit current	V_{CC} = 5V	V_O = 0V	−30	−100	−250	mA
I_{CC}	Supply current	V_{CC} = MAX			60	90	mA

Switching Characteristics Over Operating Conditions

SYMBOL	PARAMETER	TEST CONDITIONS	COMMERCIAL			UNIT
			MIN	TYP	MAX	
t_{PD}	Input or feedback to output 16L8B-2, 16R4B-2, and 16R6B-2	Commercial R_1 = 200Ω R_2 = 390Ω		17	25	ns
t_{CLK}	Clock to output or feedback except 16L8B-2			10	15	ns
t_{PZX}	Pin 11 to output enable except 16L8B-2			10	20	ns
t_{PXZ}	Pin 11 to output disable except 16L8B-2			11	20	ns
t_{PZX}	Input to output enable 16R6B-2, 16R4B-2, and 16L8B-2			10	25	ns
t_{PXZ}	Input to output disable 16R6B-2, 16R4B-2, and 16L8B-2			13	25	ns
f_{MAX}	Maximum frequency 16R8B-2, 16R6B-2, and 16R4B-2		28.5	40		MHz

[1] The PAL20B-2 Series is designed to operate over the full military operating conditions. For availability and specifications, contact Monolithic Memories.

[2] These are absolute voltages with respect to the ground pin on the device and include all overshoots due to system and/or tester noise. Do not attempt to test these values without suitable equipment.

[3] I/O pin leakage is the worst case of I_{IL} and I_{OZL} (or I_{IH} and I_{OZH}).

[4] No more than one output should be shorted at a time and duration of the short circuit should not exceed one second.

Operating Conditions

SYMBOL	PARAMETER		COMMERCIAL[1]			UNIT
			MIN	TYP	MAX	
V_{CC}	Supply voltage		4.75	5	5.25	V
t_w	Width of clock	Low	25	10		ns
		High	25	10		
t_{su}	Setup time from input or feedback to clock	16R8B-4 16R6B-4 16R4B-4	35	25		ns
t_h	Hold time		0	−10		ns
T_A	Operating free-air temperature		0	25	75	°C

Electrical Characteristics Over Operating Conditions

SYMBOL	PARAMETER	TEST CONDITION		COMMERCIAL			UNIT
				MIN	TYP	MAX	
V_{IL}[2]	Low-level input voltage					0.8	V
V_{IH}[2]	High-level input voltage			2			V
V_{IC}	Input clamp voltage	V_{CC} = MIN	I_I = −18mA		−0.8	−1.5	V
I_{IL}[3]	Low-level input current	V_{CC} = MAX	V_I = 0.4V		−0.02	−0.25	mA
I_{IH}[3]	High-level input current	V_{CC} = MAX	V_I = 2.4V			25	μA
I_I	Maximum input current	V_{CC} = MAX	V_I = 5.5V			1	mA
V_{OL}	Low-level output voltage	V_{CC} = MIN	I_{OL} = 8mA		0.3	0.5	V
V_{OH}	High-level output voltage	V_{CC} = MIN	I_{OH} = −1mA	2.4	2.8		V
I_{OZL}[3]	Off-state output current	V_{CC} = MAX	V_O = 0.4V			−100	μA
I_{OZH}[3]			V_O = 2.4V			100	μA
I_{OS}[4]	Output short-circuit current	V_{CC} = 5V	V_O = 0V	−30	−100	−250	mA
I_{CC}	Supply current	V_{CC} = MAX			30	55	mA

Switching Characteristics Over Operating Conditions

SYMBOL	PARAMETER	TEST CONDITIONS	COMMERCIAL			UNIT
			MIN	TYP	MAX	
t_{PD}	Input or feedback to output 16L8B-4, 16R4B-4, and 16R6B-4	R_1 = 800Ω R_2 = 1.56KΩ		25	35	ns
t_{CLK}	Clock to output or feedback except 16L8B-4			15	25	ns
t_{PZX}	Pin 11 to output enable except 16L8B-4			15	25	ns
t_{PXZ}	Pin 11 to output disable except 16L8B-4			15	25	ns
t_{PZX}	Input to output enable 16R6B-4, 16R4B-4, and 16L8B-4			25	35	ns
t_{PXZ}	Input to output disable 16R6B-4, 16R4B-4, and 16L8B-4			25	35	ns
f_{MAX}	Maximum frequency 16R8B-4, 16R6B-4, and 16R4B-4		16	25		MHz

[1] The PAL20B-4 Series is designed to operate over the full military operating conditions. For availability and specifications, contact Monolithic Memories.

[2] These are absolute voltages with respect to the ground pin on the device and include all overshoots due to system and/or tester noise. Do not attempt to test these values without suitable equipment.

[3] I/O pin leakage is the worst case of I_{IL} and I_{OZL} (or I_{IH} and I_{OZH}).

[4] No more than one output should be shorted at a time and duration of the short circuit should not exceed one second.

Operating Conditions

SYMBOL	PARAMETER			COMMERCIAL[1]			UNIT
				MIN	TYP	MAX	
V_{CC}	Supply voltage			4.75	5	5.25	V
t_w	Width of clock	Low	16R8D, 16R6D, 16R4D	8	6		ns
		High		8	5		
t_{su}	Setup time from input or feedback to clock			10	8		ns
t_h	Hold time			0	-6		ns
T_A	Operating free-air temperature			0	25	75	°C

Electrical Characteristics

SYMBOL	PARAMETER	TEST CONDITION		MIN	TYP	MAX	UNIT
V_{IL}[2]	Low-level input voltage					0.8	V
V_{IH}[2]	High-level input voltage			2			V
V_{IC}	Input clamp voltage	V_{CC} = MIN	I_I = −18mA		−0.8	−1.5	V
I_{IL}[3]	Low-level input current	V_{CC} = MAX	V_I = 0.4V		−0.02	−0.25	mA
I_{IH}[3]	High-level input current	V_{CC} = MAX	V_I = 2.4V			25	μA
I_I	Maximum input current	V_{CC} = MAX	V_I = 5.5V			200	μA
V_{OL}	Low-level output voltage	V_{CC} = MIN	I_{OL} = 24mA		0.3	0.5	V
V_{OH}	High-level output voltage	V_{CC} = MIN	I_{OH} = −3.2mA	2.4	3.4		V
I_{OZL}[3]	Off-state output current	V_{CC} = MAX	V_O = 0.4V			−100	μA
I_{OZH}[3]			V_O = 2.4V			100	μA
I_{OS}[4]	Output short-circuit current	V_{CC} = 5V	V_O = 0V	−30	−70	−130	mA
I_{CC}	Supply current	V_{CC} = MAX			120	180	mA
C_{IN}	Input Capacitance	V_{IN} = 2.0V@f = 1MHz			2		pF
C_{OUT}	Output Capacitance	V_{OUT} = 2.0V@f = 1MHz			4		
$C_{CLK,\ EN}$	Clock/Enable Capacitance	$V_{CLK,\ EN}$ = 2.0V@f = 1MHz			9		

Switching Characteristics Over Operating Conditions

SYMBOL	PARAMETER		TEST CONDITIONS	COMMERCIAL			UNIT
				MIN	TYP	MAX	
t_{PD}	Input or feedback to output 16L8D, 16R6D, 16R4D		R_1 = 200Ω R_2 = 390Ω	3	8	10	ns
t_{CLK}	Clock to output or feedback except 16L8D			2	6	8	ns
t_{PZX}	Pin 11 to output enable except 16L8D			3	8	10	ns
t_{PXZ}	Pin 11 to output disable except 16L8D			3	8	10	ns
t_{PZX}	Input to output enable 16L8D, 16R6D, 16R4D			1	8	10	ns
t_{PXZ}	Input to output disable 16L8D, 16R6D, 16R4D			1	8	10	ns
f_{MAX}	Maximum frequency 16R8D, 16R6D, 16R4D	Feedback		55.5	70		MHz
		No feedback		62.5	75		

1. The PAL20D Series is designed to operate over the full military operating conditions. For availability and specifications, contact Monolithic Memories.
2. These are absolute voltages with respect to the ground pin on the device and include all overshoots due to system and/or tester noise. Do not attempt to test these values without suitable equipment.
3. I/O pin leakage is the worst case of I_{IL} and I_{OZL} (or I_{IH} and I_{OZH}).
4. No more than one output should be shorted at a time and duration of the short circuit should not exceed one second.

Metastability

Metastability is a condition which can occur in any latch or flip-flop if the minimum setup or hold times are violated. In most cases, the flip-flop will either react to the input or remain in its current state, both of which are stable results. The flip-flop can also reach an "in-between" condition called the metastable state, which is stable only if there is no noise in the system and the flip-flop is perfectly balanced. This metastable condition lasts until the flip-flop falls into one of its two stable states, which can take longer than the normal response time.

The PAL20D Series exhibits better metastability characteristics than most other registered devices. It is less likely to enter the metastable state and recovers faster to a stable state. As a result, the PAL20D Series can make an excellent synchronizer circuit, and the metastability characteristics have been specified for designs in which the setup and hold times may not always be met.

Metastability Characteristics Over Operating Conditions

SYMBOL	PARAMETER	TEST CONDITIONS	COMMERCIAL			UNIT
			MIN	TYP	MAX	
p	Poisson process rate		0.85	1.05		ns^{-1}
k	MTBF constant			0.8	1.0	μs^{-1}
t_{MET}	Minimum recovery time in asynchronous mode	MTBF = 10 years f_d = (1/3)f d = 3		20	30	ns
f_{MET}	Maximum frequency in asynchronous mode	MTBF = 10 years f_d = (1/3)f d = 3	21	26		MHz

Definition of Variables

MTBF (Mean Time Between Failures): the average time between metastable occurrences that cause a violation of the device specifications. Metastability characteristics are calculated at an arbitrary MTBF of 10 years for the convenience of the user.

p (Poisson process rate): experimentally calculated factor which determines the slope of the curve of probability of failure.

k (MTBF constant): experimentally calculated factor which determines the magnitude of the curve of probability of failure.

tsu (setup time): the specified minimum time interval allowed between the application of a data signal at a specified device input pin (pin 9 on the device under test) and a subsequent clock transition. For the PAL20D Series, tsu is 10 nanoseconds.

tCLK (clock to output time): the specified maximum time interval between a clock transition and the availability of valid signals at an output pin. For the PAL20D Series, tCLK is 8 nanoseconds.

fMAX (maximum frequency): specified maximum frequency for the device under test. Calculated as 1/(tsu + tCLK). For the PAL20D Series, this calculates to 55.5 Megahertz.

f(clock frequency): actual clock frequency for the device under test.

f_d (data frequency): actual data frequency for a specified input to the device under test.

d (data ratio): the ratio of the clock frequency to the data frequency (f/f_d).

t(time delay): the additional time allowed per period beyond that required by the specifications. t is the actual time between clock transitions beyond the required period of (tsu + tCLK).

tMET (metastability recovery time): minimum t required to guarantee recovery from metastability, with specified test conditions.

fMET (metastability frequency): maximum f clock frequency to limit metastability failures, with specified test conditions.

Metastability Equations

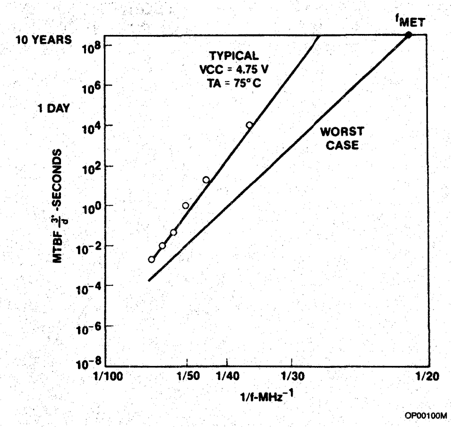
$MTBF = k\ (d/3)\ (1/f)^2\ e^{(p/f)}$

$fMAX = 1/(tsu + tCLK)$

$f = 1/(tsu + tCLK + t)$

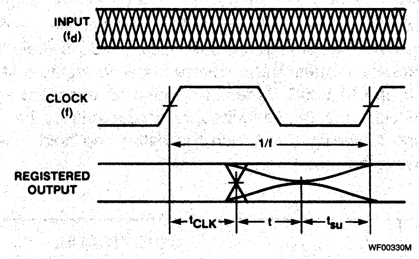
$f = d\ (f_d)$

Metastability vs. Clock Frequency



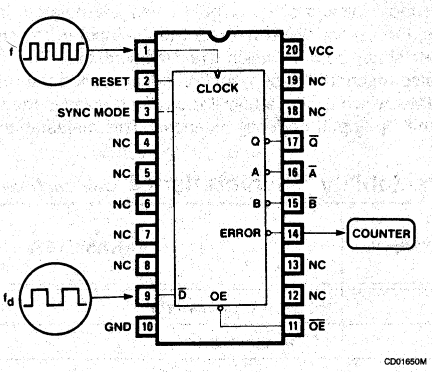
*Normalized to d = 3; multiply by 3/d for other data frequencies.

Metastability Waveforms



Metastability Test Circuit

PAL16R4D

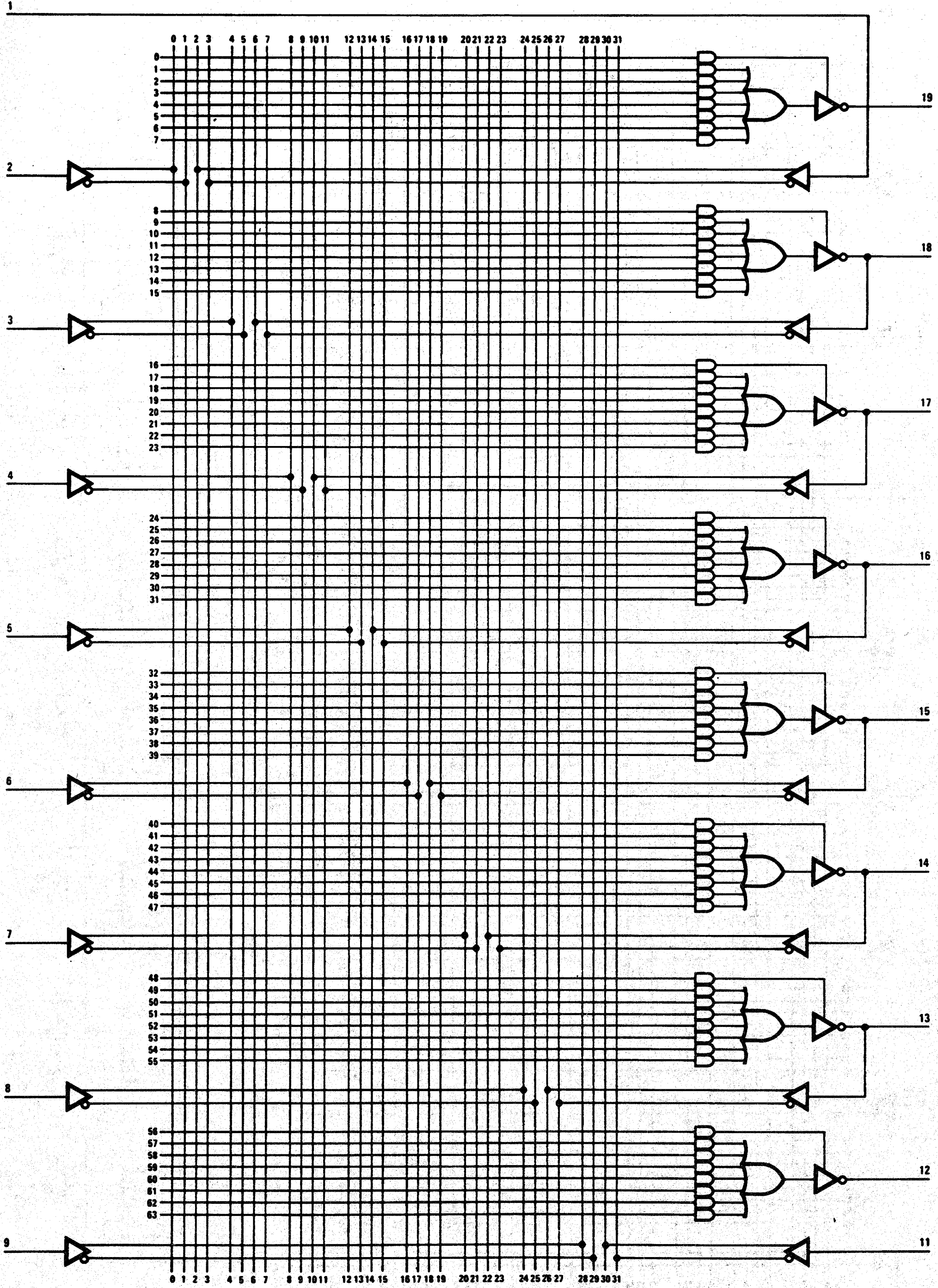


Metastability Test Pattern File

```
CHIP Metastability_Test PAL16R4

CLOCK RESET SYNC_MODE NC NC NC NC NC /D GND
/OE NC NC /ERROR /B /A /Q NC NC VCC

EQUATIONS

Q      := /Q* SYNC_MODE    ;TOGGLE SYNCHRONOUS INPUT (TESTS f MAX)
        +  D*/SYNC_MODE    ;TOGGLE ASYNCHRONOUS INPUT (TESTS META.)

A      :=  A*/Q            ;HOLD A (IF NOT ERROR)
        + /A* Q            ;TOGGLE A (IF NOT ERROR)
        +  ERROR           ;SET A IF ERROR

B      :=  B*/Q*/ERROR     ;HOLD B IF NOT ERROR, OR RESET
        + /B* Q*/ERROR     ;TOGGLE B IF NOT ERROR, OR RESET

ERROR  := /A*/B            ;COMPARE A AND B,
        +  A* B            ; ERROR GOES HIGH IF A EQUALS B
        +  RESET           ;INITIALIZE A AND B TO OPPOSITE PHASES
```

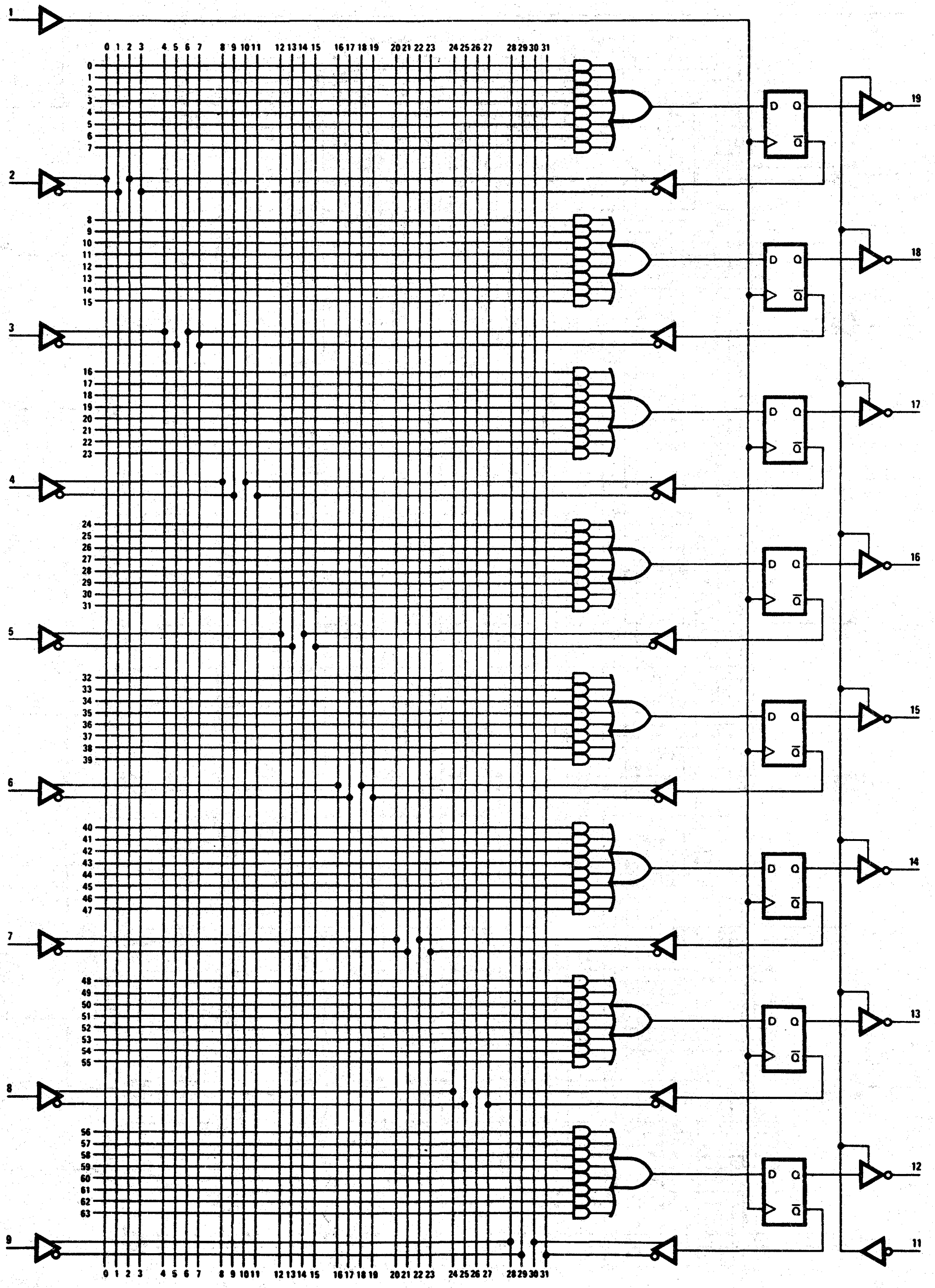
TB02030M

16L8



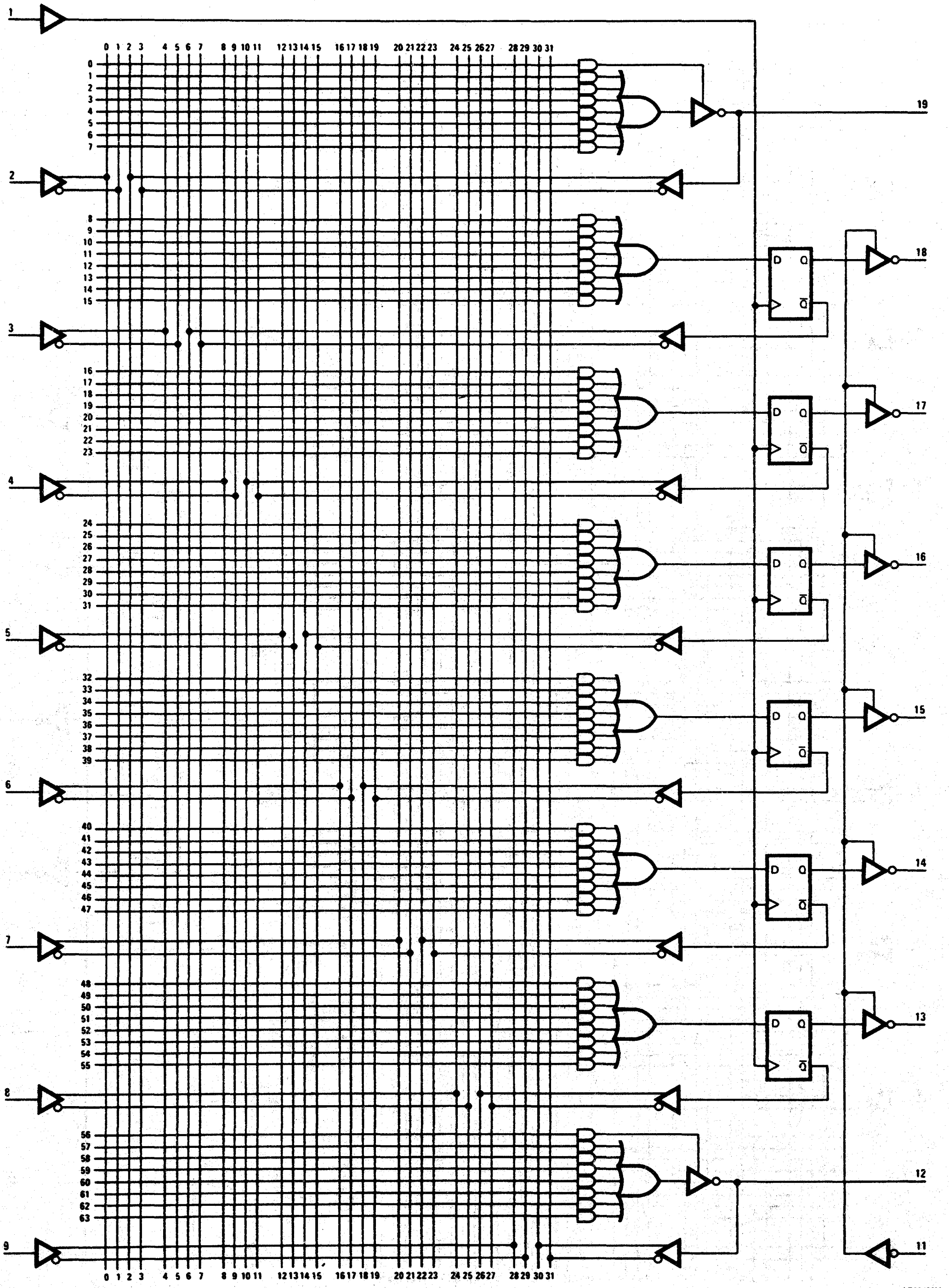
LD00410M

16R8



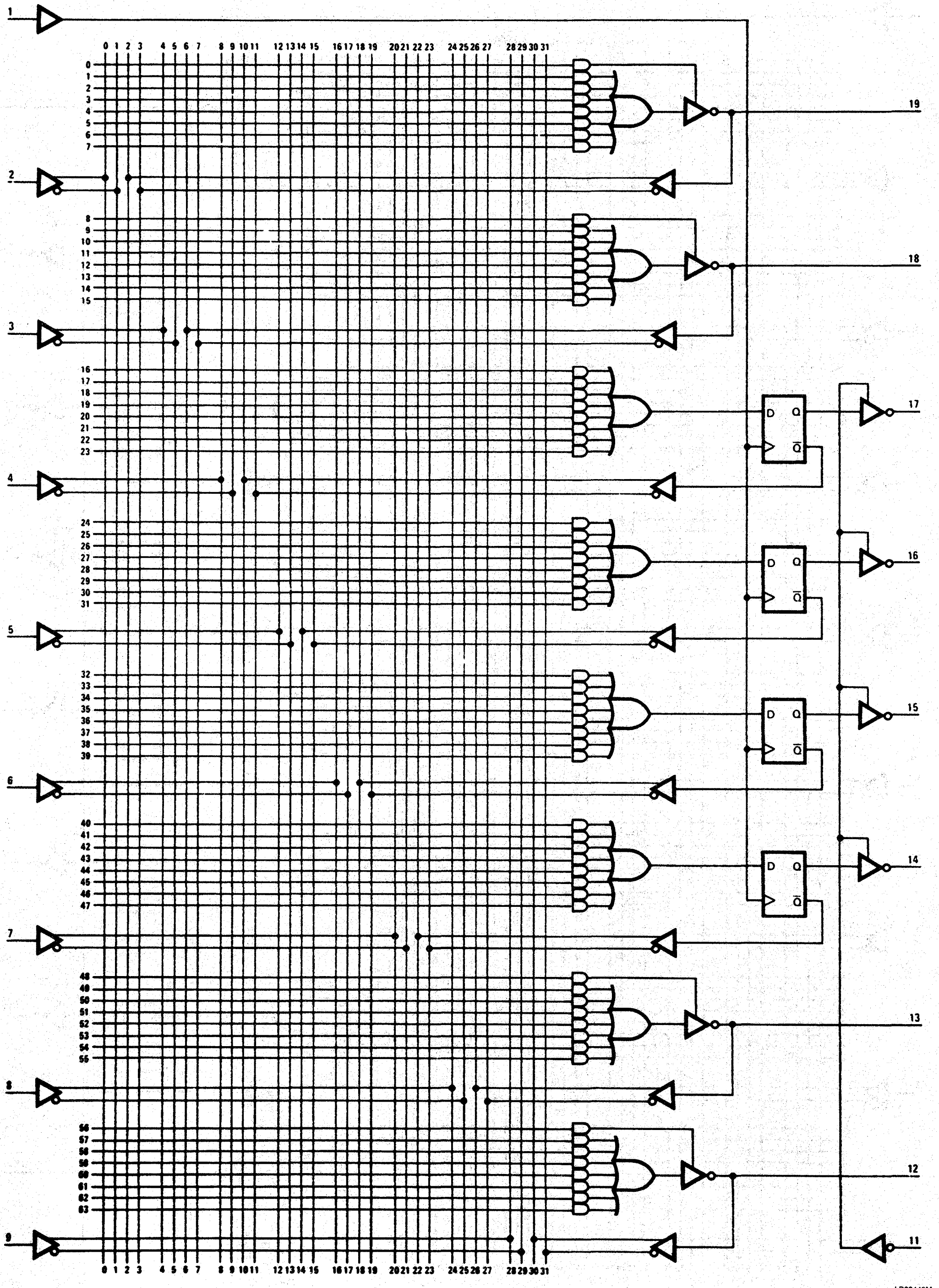
LD00420M

16R6



LD00430M

16R4


1
2
3
4
5
6
7
8
9
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63
D
Q
Q̄
19
18
17
16
15
14
13
12
11
LD00440M

Medium 20PA Series

	ARRAY INPUTS	OUTPUTS		t_{PD}* (ns)	I_{CC} (mA)
		COMBINATORIAL	REGISTERED		
PAL16P8A	16	8	0	25/30	180
PAL16RP8A	16	0	8	25/30	180
PAL16RP6A	16	6	2	25/30	180
PAL16RP4A	16	4	4	25/30	180

* 25ns active low, 30ns active high

Description

The Medium 20PA Series is equivalent to the Medium 20 Series, with the addition of programmable polarity. With programmable polarity unused, these devices are equivalent to the Medium 20A Series.

Polarity

Each of these devices offers programmable polarity on each output. If the polarity fuse is unused, the output is active low. If the polarity fuse is programmed, the output is inverted to active high.

Preload and Power-up Reset

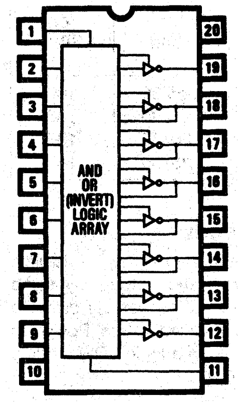
Each device also offers register preload for device testability. The registers can be preloaded from the outputs by using supervoltages (see waveforms at end of section) in order to simplify functional testing. This series also offers Power-up Reset, whereby the registers power up to a logic LOW, setting the active-low outputs to a logic HIGH.

Performance

Performance varies according to the use of the programmable polarity. Active low outputs have a tpd of 25ns, while active high outputs have a tpd of 30ns due to the extra inversion. All devices consume 180mA maximum ICC.

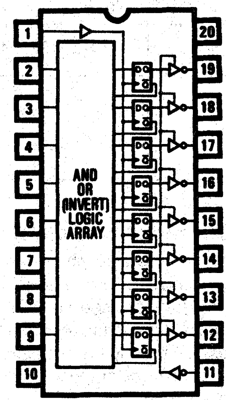
DIP Pinouts

16P8A



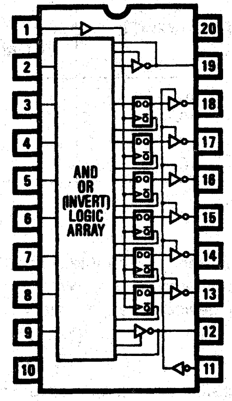
CD00400M

16RP8A



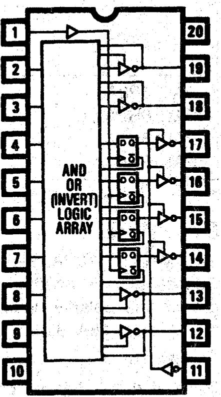
CD00410M

16RP6A



CD00420M

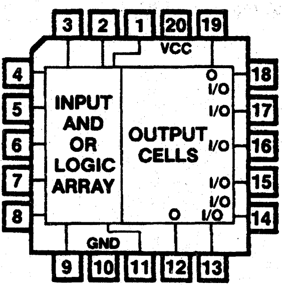
16RP4A



CD00430M

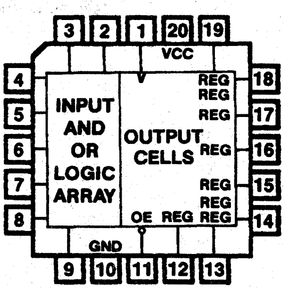
PLCC Pinouts

16P8A



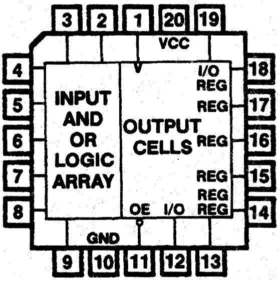
CD00100M

16RP8A



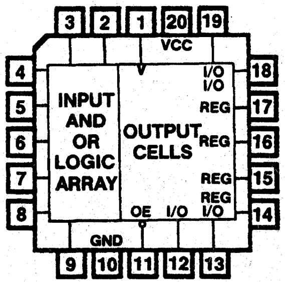
CD00530M

16RP6A



CD00540M

16RP4A



CD00110M

Operating Conditions

SYMBOL	PARAMETER			COMMERCIAL[1]			UNIT
				MIN	TYP	MAX	
V_{CC}	Supply voltage			4.75	5	5.25	V
t_w	Width of clock	Low		20	14		ns
		High		10	6		
t_{su}	Setup time from input or feedback to clock	16RP8A 16RP6A 16RP4A	Polarity fuse intact	25	15		ns
			Polarity fuse blown	30	20		
t_h	Hold time			0	-10		ns
T_A	Operating free-air temperature			0		75	°C

Electrical Characteristics Over Operating Conditions

SYMBOL	PARAMETER	TEST CONDITION		COMMERCIAL			UNIT
				MIN	TYP	MAX	
V_{IL}[2]	Low-level input voltage					0.8	V
V_{IH}[2]	High-level input voltage			2			V
V_{IC}	Input clamp voltage	V_{CC} = MIN	I_I = −18mA		-0.8	-1.5	V
I_{IL}[3]	Low-level input current	V_{CC} = MAX	V_I = 0.4V		-0.02	-0.25	mA
I_{IH}[3]	High-level input current	V_{CC} = MAX	V_I = 2.4V			25	μA
I_I	Maximum input current	V_{CC} = MAX	V_I = 5.5V			1	mA
V_{OL}	Low-level output voltage	V_{CC} = MIN	I_{OL} = 24mA		0.3	0.5	V
V_{OH}	High-level output voltage	V_{CC} = MIN	I_{OH} = −3.2mA	2.4	2.8		V
I_{OZL}[3]	Off-state output current	V_{CC} = MAX	V_O = 0.4V			-100	μA
I_{OZH}[3]			V_O = 2.4V			100	μA
I_{OS}[4]	Output short-circuit current	V_{CC} = 5V	V_O = 0V	-30	-70	-130	mA
I_{CC}	Supply current	V_{CC} = MAX			120	180	mA

[1] The PAL20PA Series is designed to operate over the full military operating conditions. For availability and specifications, contact Monolithic Memories.

[2] These are absolute voltages with respect to the ground pin on the device and include all overshoots due to system and/or tester noise. Do not attempt to test these values without suitable equipment.

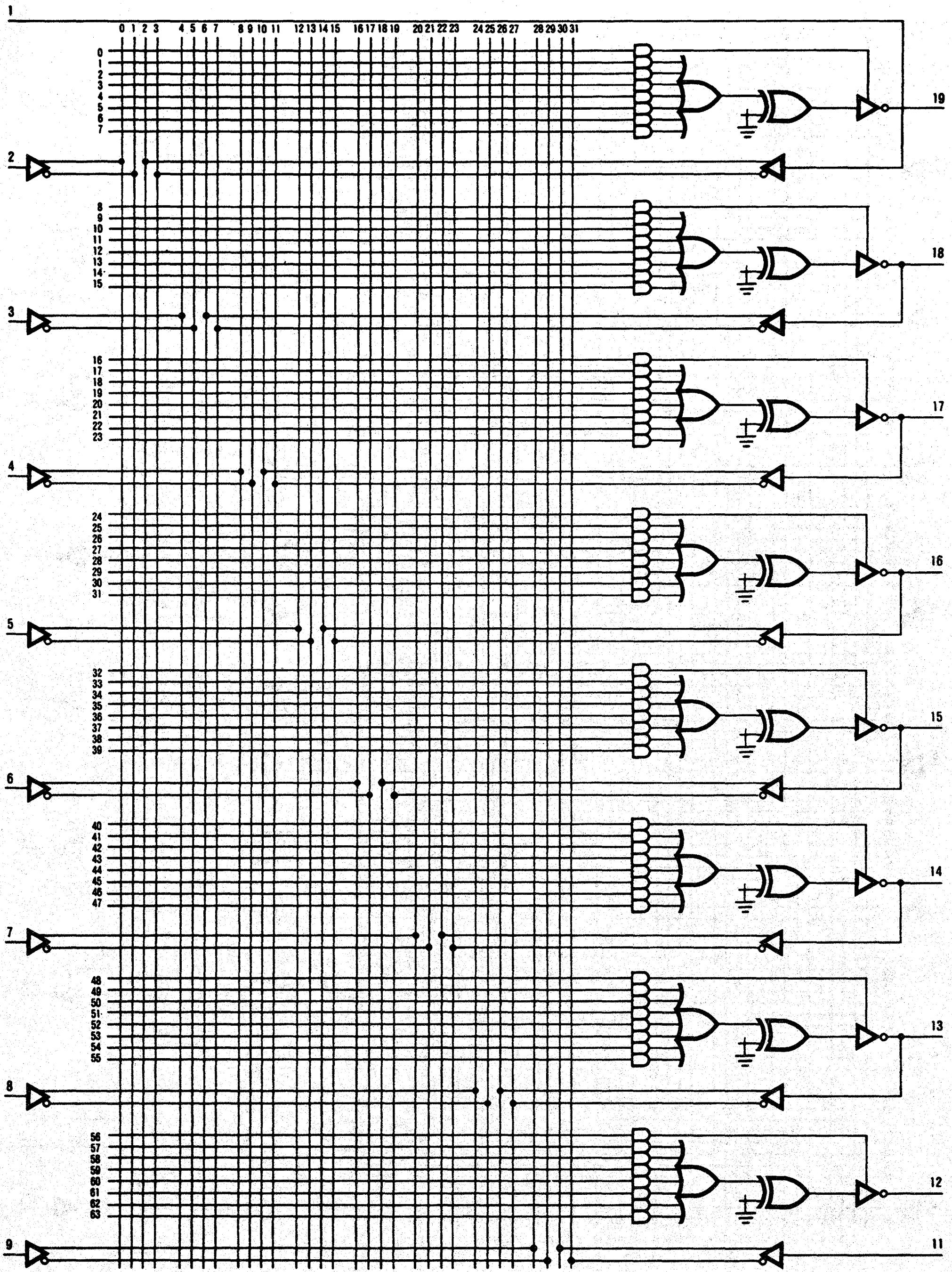
[3] I/O pin leakage is the worst case of I_{IL} and I_{OZL} (or I_{IH} and I_{OZH}).

[4] No more than one output should be shorted at a time and duration of the short circuit should not exceed one second.

Switching Characteristics Over Operating Conditions

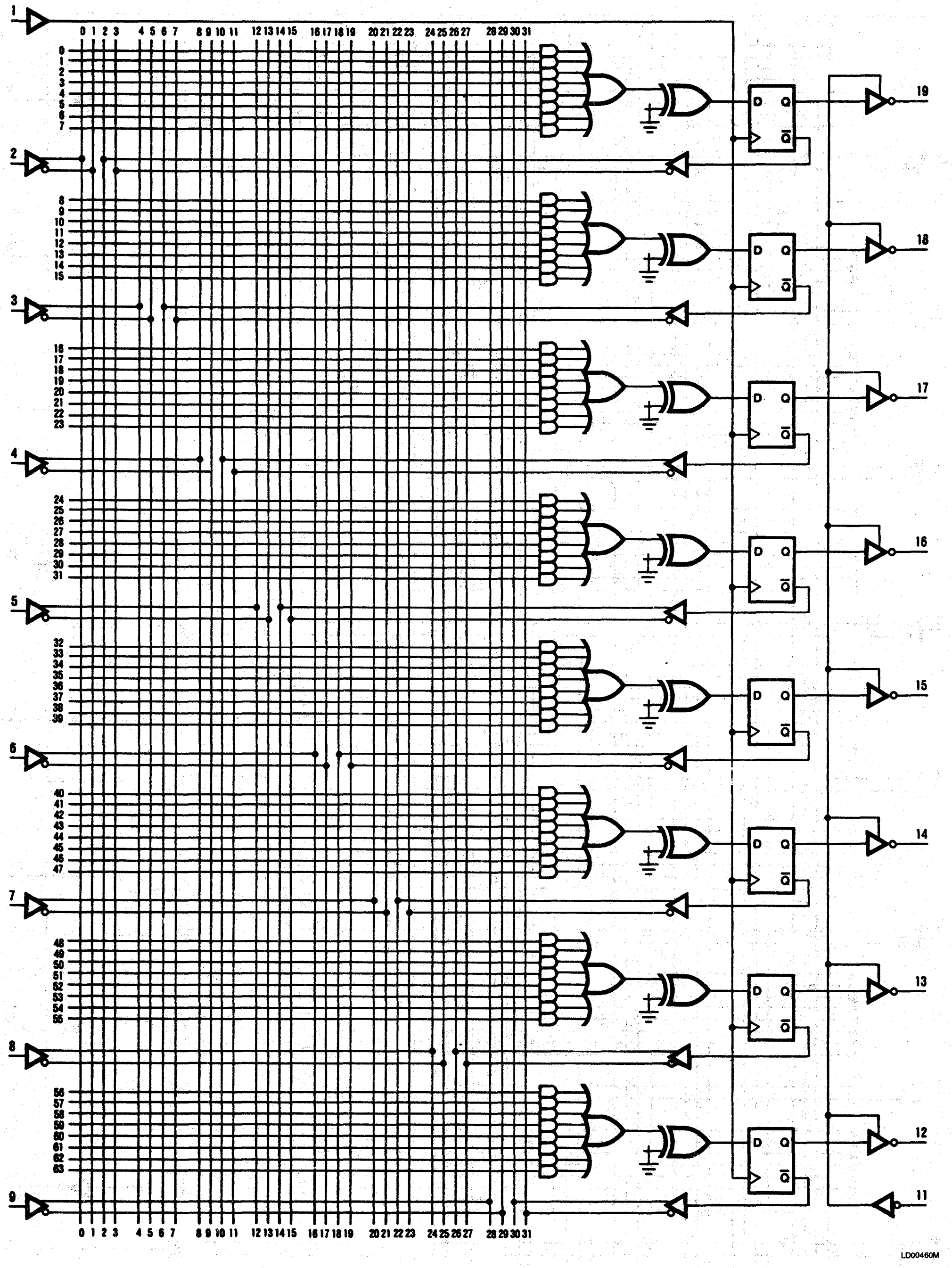
SYMBOL	PARAMETER		TEST CONDITIONS	COMMERCIAL			UNIT
				MIN	TYP	MAX	
t_{PD}	Input or feedback to output 16P8A, 16RP6A, 16RP4A	Polarity fuse intact	R_1 = 200Ω R_2 = 390KΩ		15	25	ns
		Polarity fuse blown			20	30	
t_{CLK}	Clock to output or feedback				10	15	ns
t_{PZX}	Pin 11 to output enable except 16P8A				10	20	ns
t_{PXZ}	Pin 11 to output disable except 16P8A				11	20	ns
t_{PZX}	Input to output enable	16RP6A, 16RP4A, and 16P8A			10	25	ns
t_{PXZ}	Input to output disable	16RP6A, 16RP4A, and 16P8A			13	25	ns
f_{MAX}	Maximum frequency 16RP8A, 16RP6A, 16RP4A	Polarity fuse intact		28.5	40		MHz
		Polarity fuse blown		25	33		

16P8A

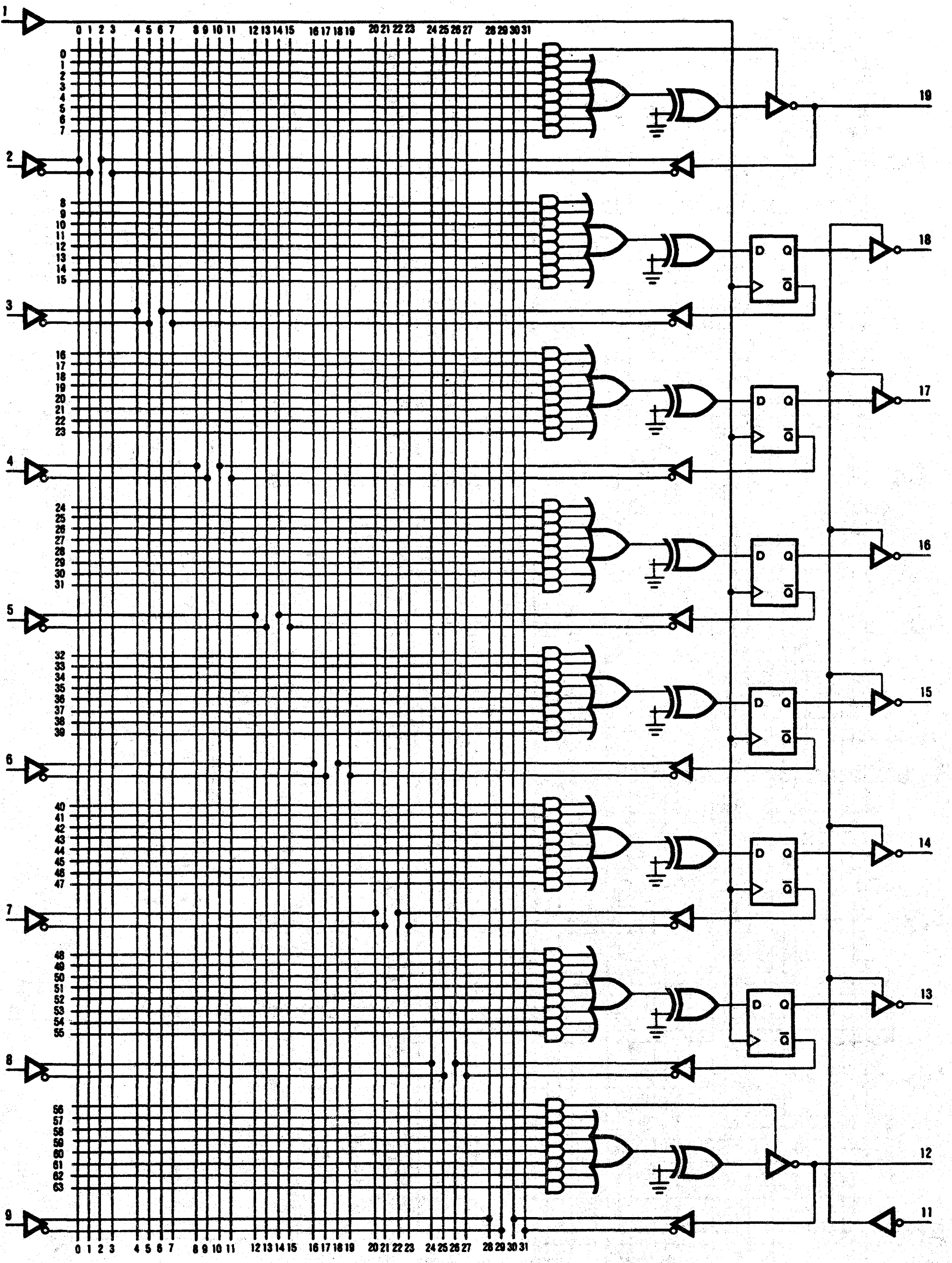


LD00450M

16RP8A

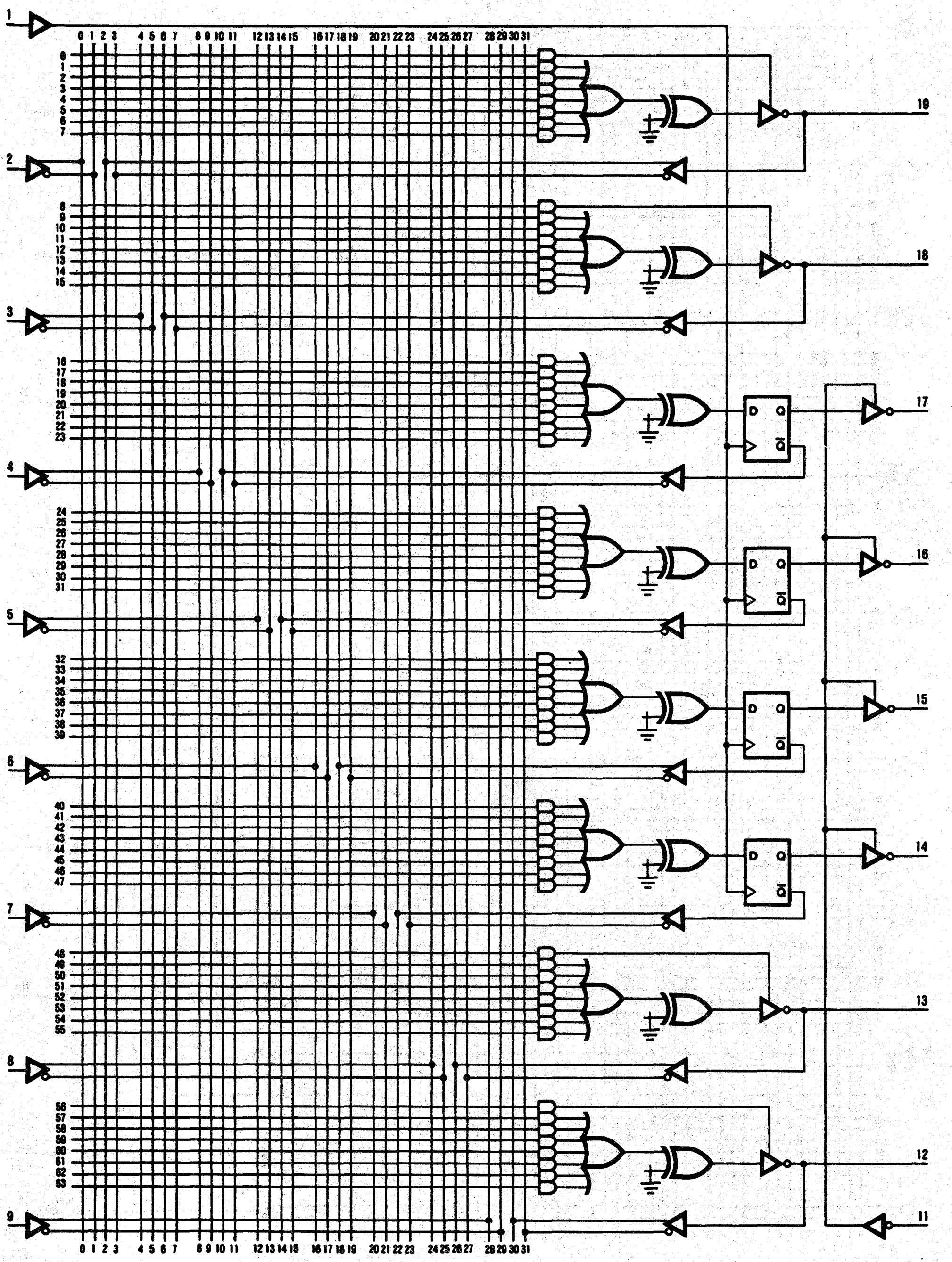

1
2
3
4
5
6
7
8
9
11
12
13
14
15
16
17
18
19
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31
D
Q
Q̄
LD00460M

16RP6A



LD00470M

16RP4A



LD00480M

Large 20 Arithmetic Series

	ARRAY INPUTS	OUTPUTS		PRODUCT TERMS
		COMBINATORIAL	REGISTERED	
PAL16X4	16	4	4	64
PAL16A4	16	4	4	74

Description

The PAL16X4 and PAL16A4 have arithmetic gated feedback. These are specialized devices for arithmetic applications.

Arithmetic Gated Feedback

The arithmetic functions (add, subtract, greater than, and less than) are implemented by addition of gated feedback to the features of the XOR PAL device. The XOR at the input of the D-type flip-flop allows carrys from previous operations to be XORed with two variable sums generated by the PAL device array. The flip-flop Q output is fed back to be gated with input terms A (Figure 13). This gated feedback provides any one of the sixteen possible Boolean combinations which are mapped in the Karnaugh map (Figure 14). Figure 15 shows how the PAL device array can be programmed to perform these sixteen operations. These features provide for versatile operations on two variables and facilitate the parallel generation of carrys necessary for fast arithmetic operations.

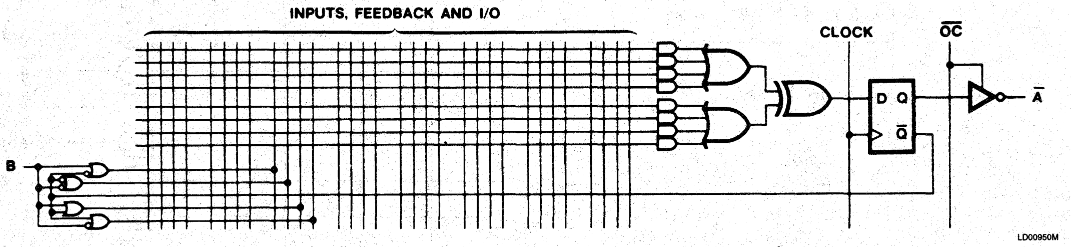


Figure 13

$(\overline{A}+B), (\overline{A}+\overline{B})$ / $(A+\overline{B}), (A+B)$	--	-x	xx	x-
--	1	$\overline{A}+\overline{B}$	$\overline{A}$	$\overline{A}+B$
-x	A+B	A:+:B	$\overline{A}\cdot B$	B
xx	A	$A\cdot\overline{B}$	0	$A\cdot B$
x-	$A+\overline{B}$	$\overline{B}$	$\overline{A}\cdot\overline{B}$	A:•:B

TB01630M

Figure 14

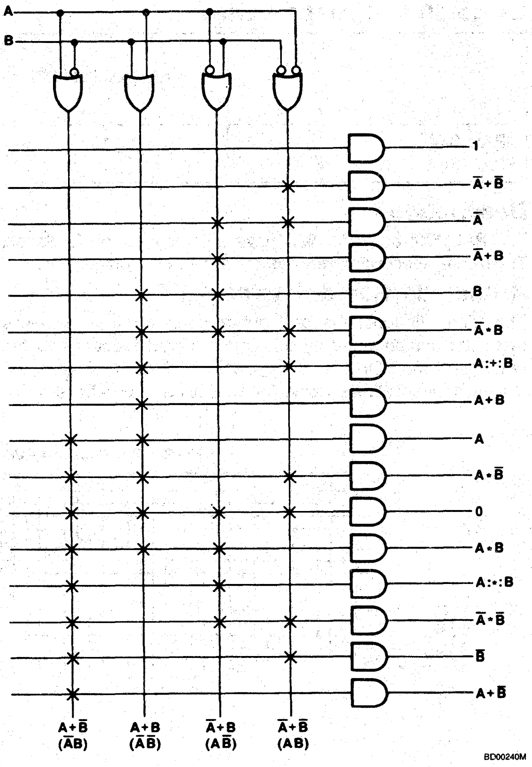
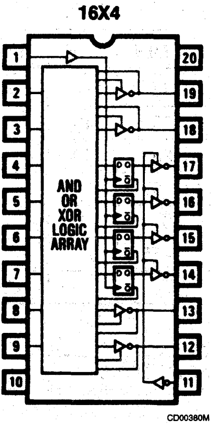
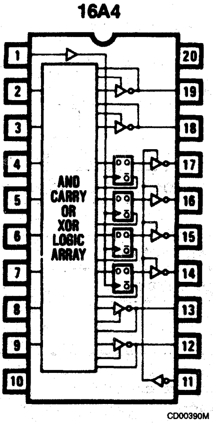

A
B
1
$\overline{A}+\overline{B}$
$\overline{A}$
$\overline{A}+B$
B
$\overline{A}\cdot B$
A:+:B
A+B
A
$A\cdot\overline{B}$
0
$A\cdot B$
A:•:B
$\overline{A}\cdot\overline{B}$
$\overline{B}$
$A+\overline{B}$
$A+\overline{B}$ $(\overline{A}B)$
$A+B$ $(\overline{A}\overline{B})$
$\overline{A}+B$ $(A\overline{B})$
$\overline{A}+\overline{B}$ (AB)
BD00240M

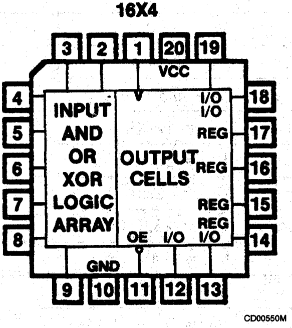
Figure 15

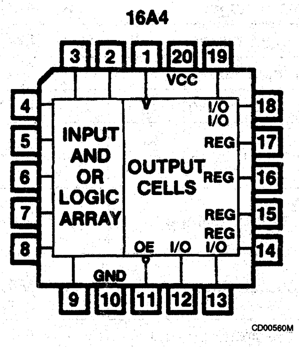
DIP Pinouts





PLCC Pinouts





Operating Conditions

SYMBOL	PARAMETER		MILITARY			COMMERCIAL			UNIT
			MIN	TYP	MAX	MIN	TYP	MAX	
V_{CC}	Supply voltage		4.5	5	5.5	4.75	5	5.25	V
t_w	Width of clock	Low	25	10		25	10		ns
		High	25	10		25	10		
t_{su}	Set up time from input or feedback to clock		55	30		45	30		ns
t_h	Hold time		0	−15		0	−15		ns
T_A	Operating free-air temperature		−55			0		75	°C
T_C	Operating case temperature				125				°C

Electrical Characteristics

SYMBOL	PARAMETER	TEST CONDITIONS			MIN	TYP	MAX	UNIT
V_{IL}[1]	Low-level input voltage						0.8	V
V_{IH}[1]	High-level input voltage				2			V
V_{IC}	Input clamp voltage	V_{CC} = MIN		I_I = −18mA		−0.8	−1.5	V
I_{IL}[2]	Low-level input current	V_{CC} = MAX		V_I = 0.4V		−0.02	−0.25	mA
I_{IH}[2]	High-level input current	V_{CC} = MAX		V_I = 2.4V			25	μA
I_I	Maximum input current	V_{CC} = MAX		V_I = 5.5V			1	mA
V_{OL}	Low-level output voltage	V_{CC} = MIN	Mil	I_{OL} = 12mA		0.3	0.5	V
			Com	I_{OL} = 24mA				
V_{OH}	High-level output voltage	V_{CC} = MIN	Mil	I_{OL} = −2mA	2.4	2.8		V
			Com	I_{OL} = −3.2mA				
I_{OZL}[2]	Off-state output current	V_{CC} = MAX		V_O = 0.4V			−100	μA
I_{OZH}[2]				V_O = 2.4V			100	μA
I_{OS}[3]	Output short-circuit current	V_{CC} = 5V		V_O = 0V	−30	−70	−130	mA
I_{CC}	Supply current	V_{CC} = MAX	16X4			160	225	mA
			16A4			170	240	

Switching Characteristics Over Operating Conditions

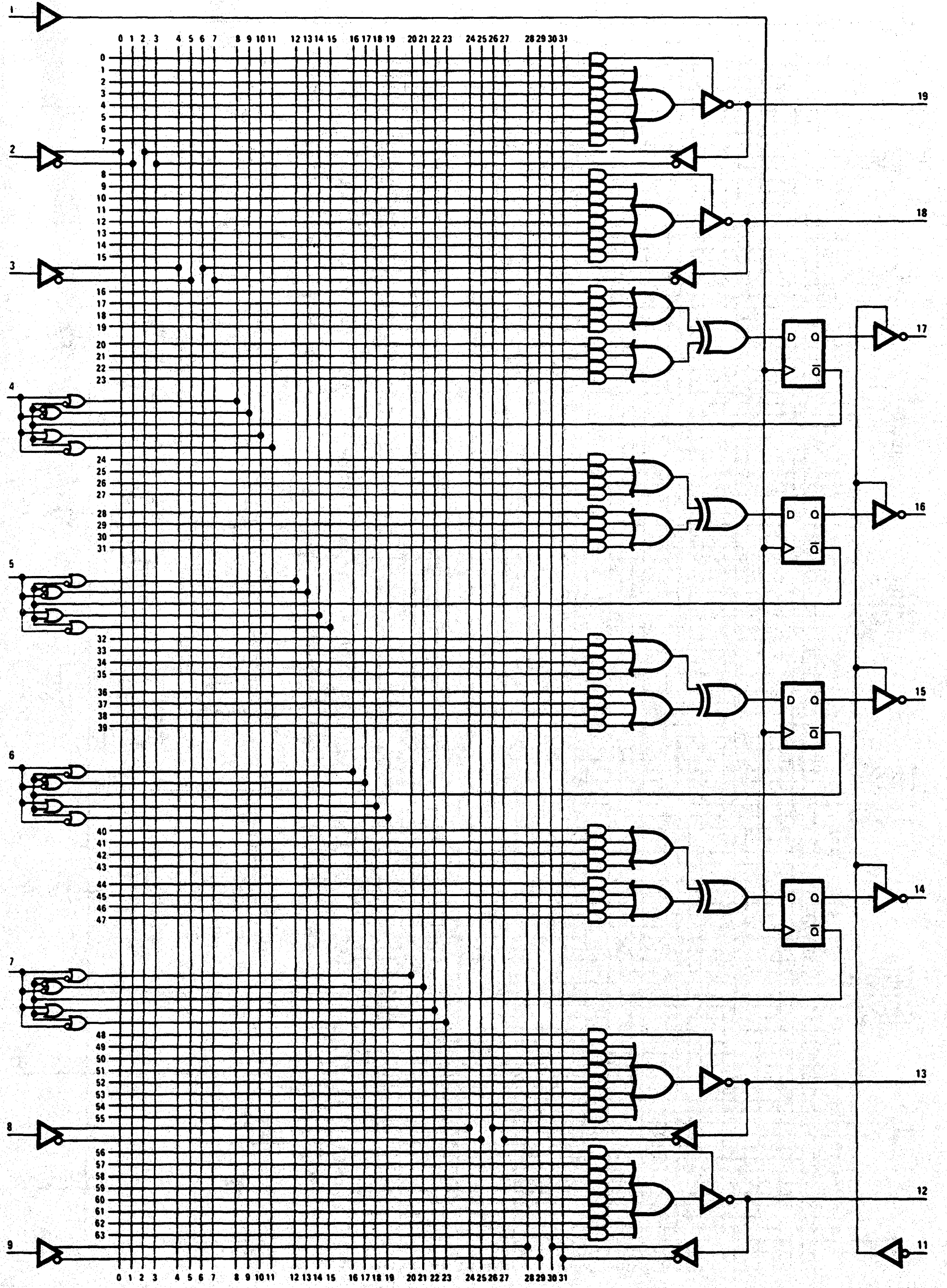
SYMBOL	PARAMETER	TEST CONDITIONS	MILITARY			COMMERCIAL			UNIT
			MIN	TYP	MAX	MIN	TYP	MAX	
t_{PD}	Input or feedback to output	R_1 = 200Ω R_2 = 390Ω		30	45		30	40	ns
t_{CLK}	Clock to output or feedback			15	25		15	25	ns
t_{PZX}	Pin 11 to output enable			15	25		15	25	ns
t_{PXZ}	Pin 11 to output disable			15	25		15	25	ns
t_{PZX}	Input to output enable			30	45		30	40	ns
t_{PXZ}	Input to output disable			30	45		30	40	ns
f_{MAX}	Maximum frequency		12	22		14	22		MHz

[1] These are absolute voltages with respect to the ground pin on the device and include all overshoots due to system and/or tester noise. Do not attempt to test these values without suitable equipment.

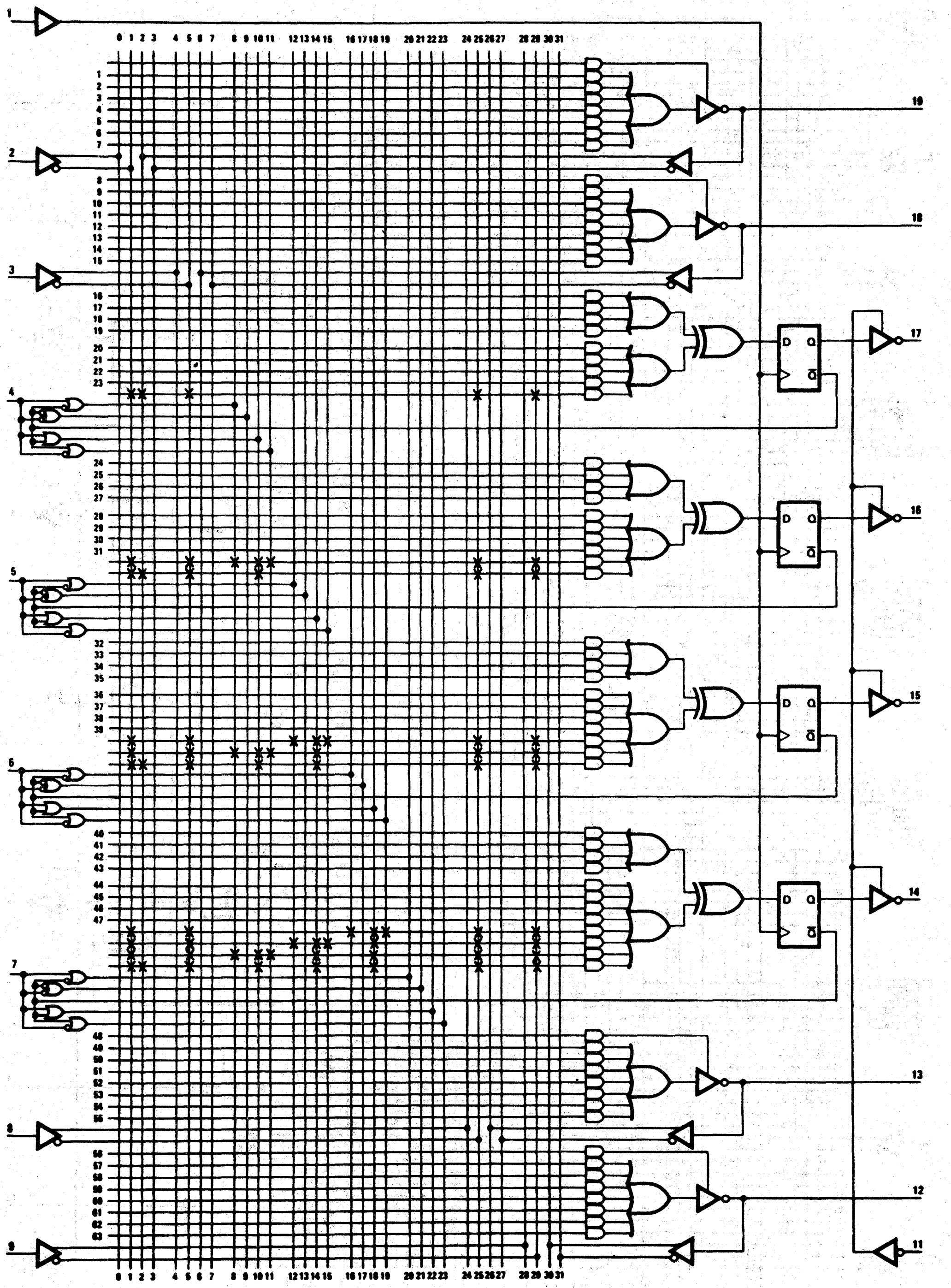
[2] I/O pin leakage is the worst case of I_{IL} and I_{OZL} (or I_{IH} and I_{OZH}).

[3] No more than one output should be shorted at a time and duration of the short circuit should not exceed one second.

16X4


1
2
3
4
5
6
7
8
9
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
D
Q
Q̄
19
18
17
16
15
14
13
12
11
LD00490M

16A4


1
2
3
4
5
6
7
8
9
11
12
13
14
15
16
17
18
19
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31
D Q
Q̄
LD00500M

Large 20RA (PAL16RA8) Description

The PAL16RA8 is a 20-pin registered asynchronous PAL device. It is a 20-pin version of the original asynchronous PAL device, the PAL20RA10. This versatile device features programmable clock, enable, set, and reset, all of which can operate asynchronously to other flip-flops in the same device. It also has individual flop-flop bypass, allowing this one device to provide any combination of registered and combinatorial outputs.

Programmable Clock

The clock input to each flip-flop comes from the programmable array, allowing the flip-flops to be clocked independently if desired.

Programmable Set and Reset

Each flip-flop has a product line for asynchronous set and one product for asynchronous reset. If the chosen product line is high, the flip-flop will set (become a logic HIGH), or reset (become a logic LOW). The sense of the output pin is inverted if the output is active low.

Programmable Polarity

Each flip-flop has individually programmable polarity. The unprogrammed state is active low.

Programmable Flip-flop Bypass

If both the set and reset product lines are high, the flip-flop is bypassed and the output becomes combinatorial. Thus each output can be configured to be registered or combinatorial.

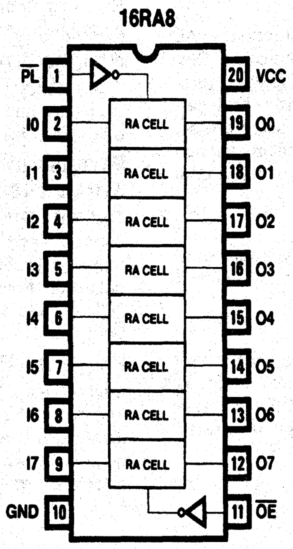
Programmable and Hard-Wired Three-State Outputs

The PAL16RA8 provides a product term dedicated to output control. There is also an output control pin (pin 11). The output is enabled if both the output control pin is low and the output control product term is high. If the output control pin is high all outputs will be disabled. If an output control product term is low, then that output will be disabled.

Register Preload and Power-up Reset

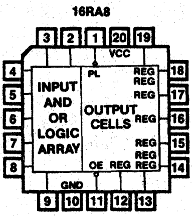
Each device also offers register preload for device testability. The registers can be preloaded from the outputs by using TTL level signals in order to simplify functional testing. This series also offers Power-up Reset, whereby the registers power up to a logic LOW, setting the active-low outputs to a logic HIGH.

DIP Pinout



CD00050M

PLCC Pinout



CD00121M

Operating Conditions

SYMBOL	PARAMETER		COMMERCIAL[1]			UNIT
			MIN	TYP	MAX	
V_{CC}	Supply voltage		4.75	5	5.25	V
t_w	Width of clock		20	13		ns
t_{wp}	Preload pulse width		35	15		ns
t_{su}	Setup time for input or feedback to clock		20	10		ns
t_{sup}	Preload setup time		25	5		ns
t_h	Hold time	Polarity fuse intact	10	-2		ns
		Polarity fuse blown	0	-6		
t_{hp}	Preload hold time		25	5		ns
T_A	Operating free-air temperature		0		75	°C
T_C	Operating case temperature					°C

Electrical Characteristics Over Operating Conditions

SYMBOL	PARAMETER	TEST CONDITION		MIN	TYP	MAX	UNIT
V_{IL}[2]	Low-level input voltage					0.8	V
V_{IH}[2]	High-level input voltage			2			V
V_{IC}	Input clamp voltage	V_{CC} = MIN	I_I = −18mA		−0.8	−1.5	V
I_{IL}[3]	Low-level input current	V_{CC} = MAX	V_I = 0.4V		−0.02	−0.25	mA
I_{IH}[3]	High-level input current	V_{CC} = MAX	V_I = 2.4V			25	μA
I_I	Maximum input current	V_{CC} = MAX	V_I = 5.5V			1	mA
V_{OL}	Low-level output voltage	V_{CC} = MIN	I_{OL} = 8mA		0.3	0.5	V
V_{OH}	High-level output voltage	V_{CC} = MIN	I_{OH} = −3.2mA	2.4	2.8		V
I_{OZ}[3]	Off-state output current	V_{CC} = MAX	V_O = 2.4V/V_O = 0.4V	−100		100	μA
I_{OS}[4]	Output short-circuit current	V_{CC} = 5V	V_O = 0V	−30	−70	−130	mA
I_{CC}	Supply current	V_{CC} = MAX			135	170	mA

Switching Characteristics Over Operating Conditions

SYMBOL	PARAMETER		TEST CONDITIONS	COMMERCIAL			UNIT
				MIN	TYP	MAX	
t_{PD}	Input or feedback to output	Polarity fuse intact	R_1 = 560Ω R_2 = 1.1KΩ		20	30	ns
		Polarity fuse blown			25	35	
t_{CLK}	Clock to output or feedback			10	17	30	ns
t_S	Input to asynchronous set				22	35	ns
t_R	Input to asynchronous reset				27	40	ns
t_{PZX}	Pin 11 to output enable				10	20	ns
t_{PXZ}	Pin 11 to output disable				10	20	ns
t_{PZX}	Input to output enable				18	30	ns
t_{PXZ}	Input to output disable				15	30	ns
f_{MAX}	Maximum frequency			20	35		MHz

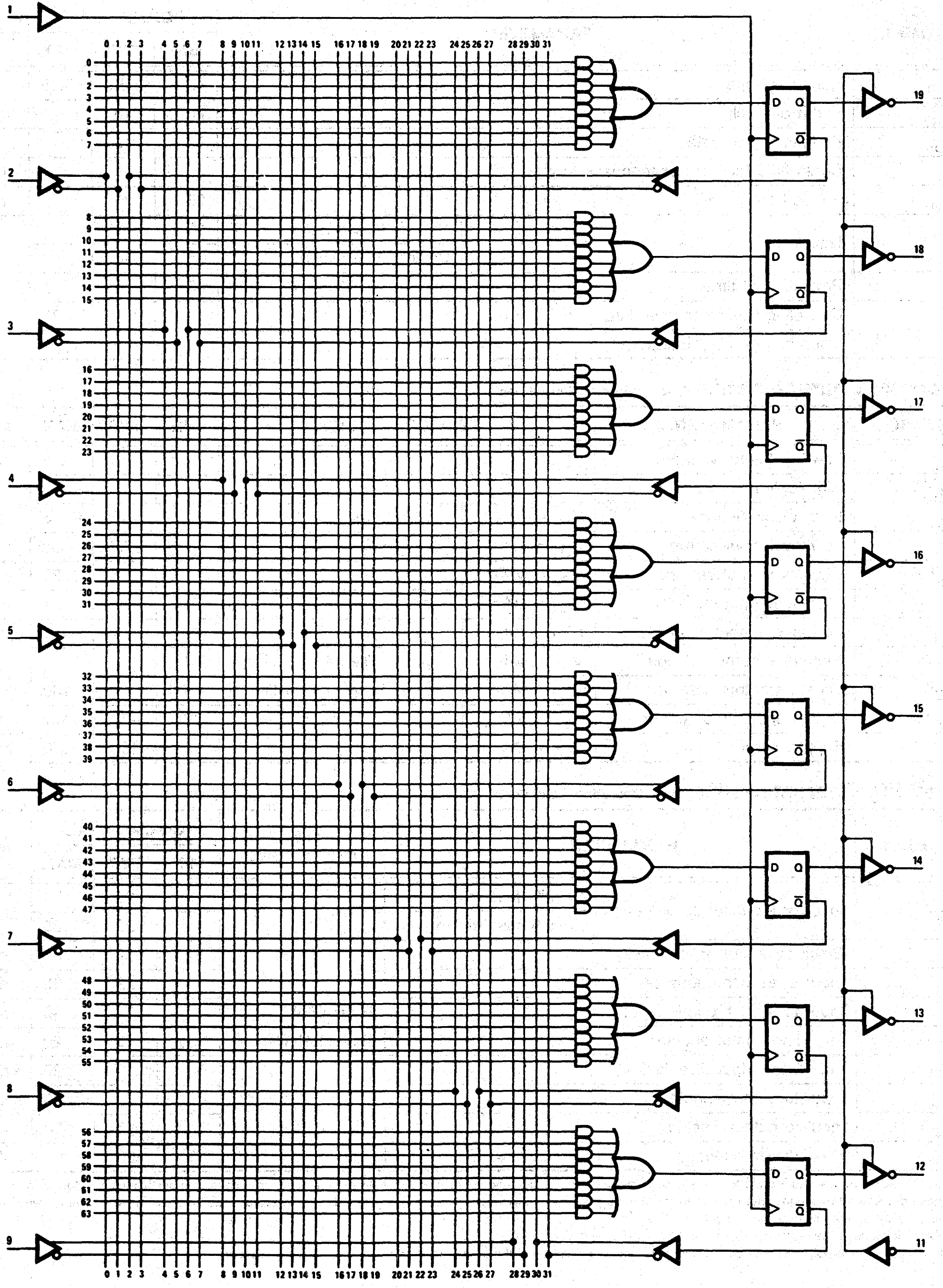
[1] The PAL20RA Series is designed to operate over the full military operating conditions. For availability and specifications, contact Monolithic Memories.

[2] These are absolute voltages with respect to the ground pin on the device and include all overshoots due to system and/or tester noise. Do not attempt to test these values without suitable equipment.

[3] I/O pin leakage is the worst case of I_{IL} and I_{OZL} (or I_{IH} and I_{OZH}).

[4] No more than one output should be shorted at a time and duration of the short circuit should not exceed one second.

16RA8



LD00420M

Small 24 Series

	INPUTS	OUTPUTS	POLARITY	T_{PD} (ns)	I_{CC} (mA)
PAL12L10	12	10	LOW	40	100
PAL14L8	14	8	LOW	40	100
PAL16L6	16	6	LOW	40	100
PAL18L4	18	4	LOW	40	100
PAL20L2	20	2	LOW	40	100
PAL20C1	20	2	LOW	40	100

Description

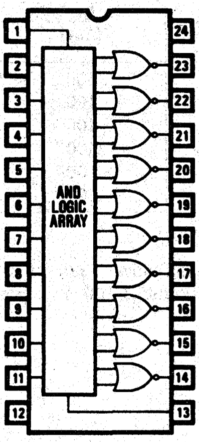
The Small 24 Series is made up of six combinatorial 24-pin PAL devices. They implement simple combinatorial logic, with no feedback.

Performance

The standard series has a propagation delay (tpd) of 40 nanoseconds (ns). Standard supply current is 100 milliamps (mA).

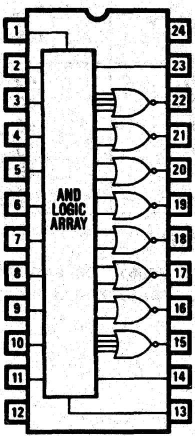
DIP Pinouts

12L10



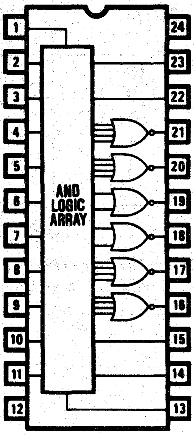
CD00580M

14L8



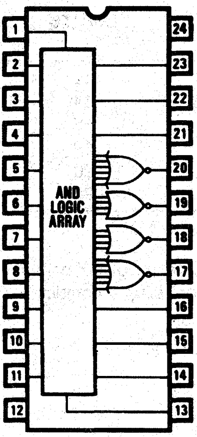
CD00590M

16L6



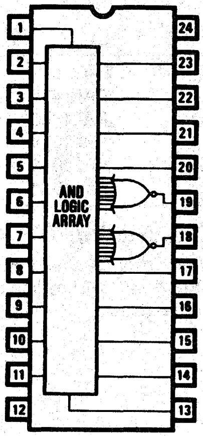
CD00130M

18L4



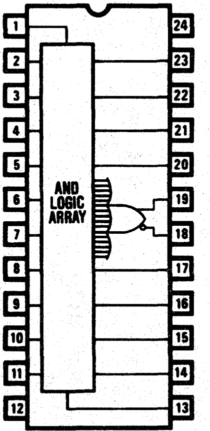
CD00570M

20L2



CD00600M

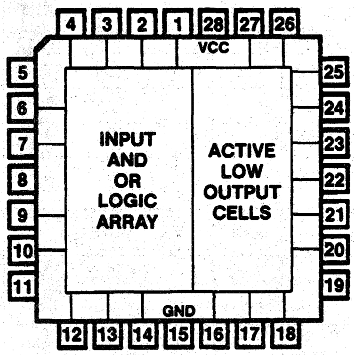
20C1



CD00610M

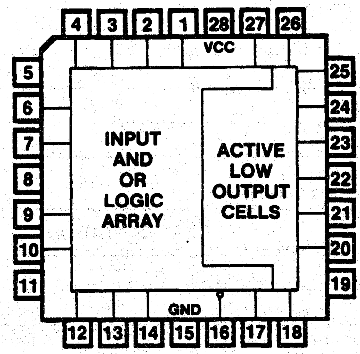
PLCC Pinouts

12L10



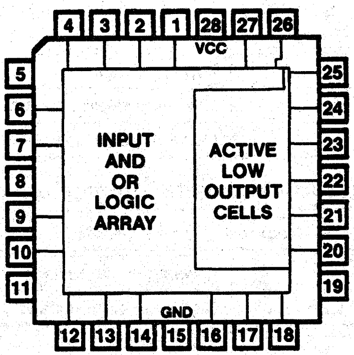
CD00730M

14L8



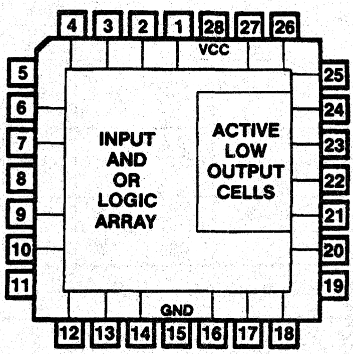
CD00180M

16L6



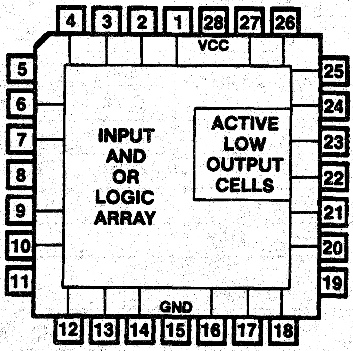
CD00740M

18L4



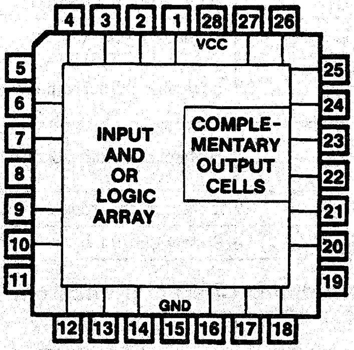
CD00750M

20L2



CD00190M

20C1



CD00760M

Operating Conditions

SYMBOL	PARAMETER	MILITARY			COMMERCIAL			UNIT
		MIN	TYP	MAX	MIN	TYP	MAX	
V_{CC}	Supply voltage	4.5	5	5.5	4.75	5	5.25	V
T_A	Operating free-air temperature	−55			0		75	°C
T_C	Operating case temperature			125				°C

Electrical Characteristics Over Operating Conditions

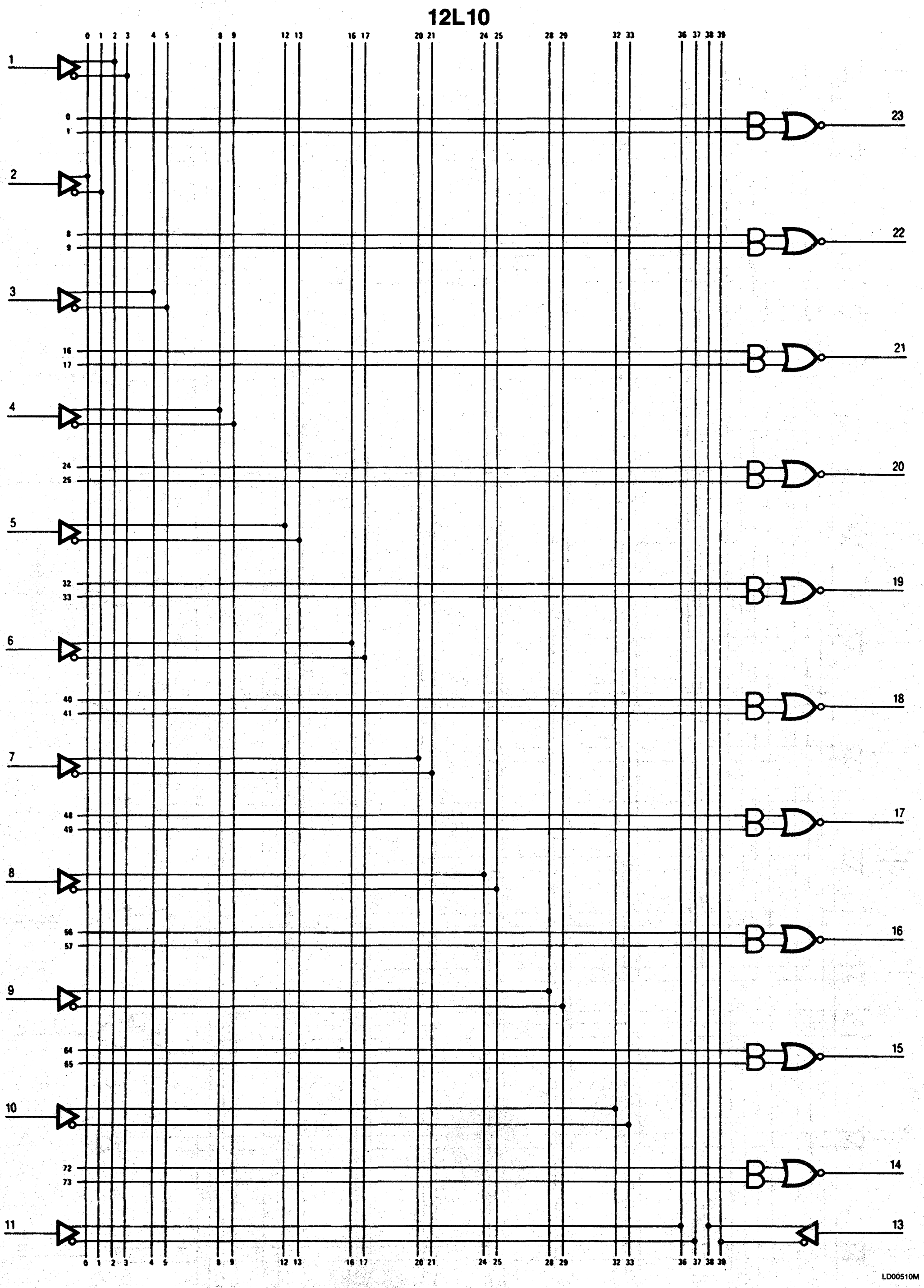
SYMBOL	PARAMETER	TEST CONDITIONS		MIN	TYP	MAX	UNIT
V_{IL}[1]	Low-level input voltage					0.8	V
V_{IH}[1]	High-level input voltage			2			V
V_{IC}	Input clamp voltage	V_{CC} = MIN	I_I = −18mA		−0.8	−1.5	V
I_{IL}	Low-level input current	V_{CC} = MAX	V_I = 0.4V		−0.02	−0.25	mA
I_{IH}	High-level input current	V_{CC} = MAX	V_I = 2.4V			25	μA
I_I	Maximum input current	V_{CC} = MAX	V_1 = 5.5V			1	mA
V_{OL}	Low-level output voltage	V_{CC} = MIN	MIL I_{OL} = 8mA		0.3	0.5	V
			COM I_{OL} = 8mA				
V_{OH}	High-level output voltage	V_{CC} = MIN	MIL I_{OH} = −2mA	2.4	2.8		V
			COM I_{OH} = −3.2mA				
I_{OS}[2]	Output short-circuit current	V_{CC} = 5V	V_O = 0V	−30	−70	−130	mA
I_{CC}	Supply current	V_{CC} = MAX			60	100	mA

Switching Characteristics Over Operating Conditions

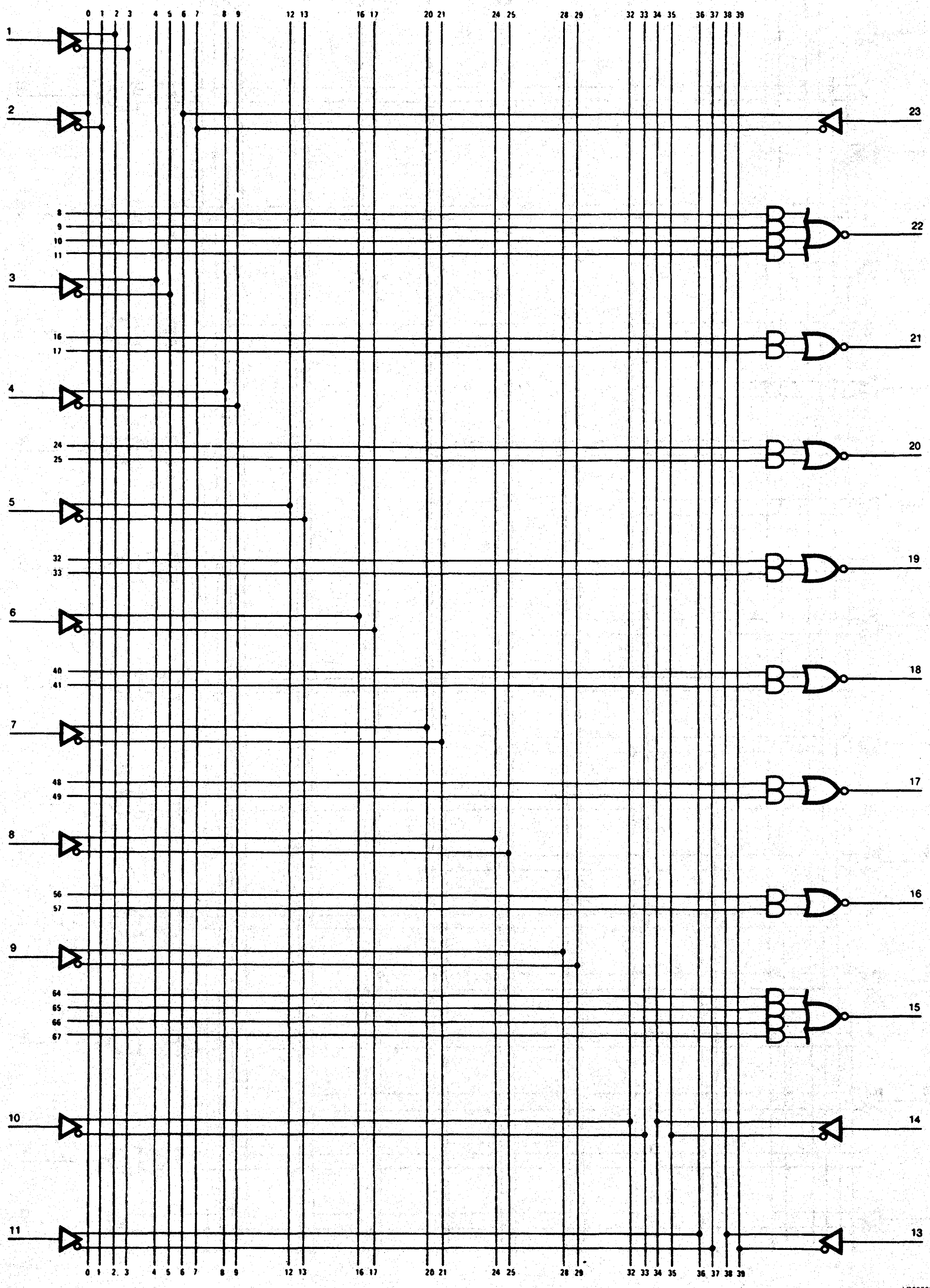
SYMBOL	PARAMETER	TEST CONDITIONS	MILITARY			COMMERCIAL			UNIT
			MIN	TYP	MAX	MIN	TYP	MAX	
t_{PD}	Input or feedback to output	R1 = 560Ω R2 = 1.1kΩ		25	45		25	40	ns

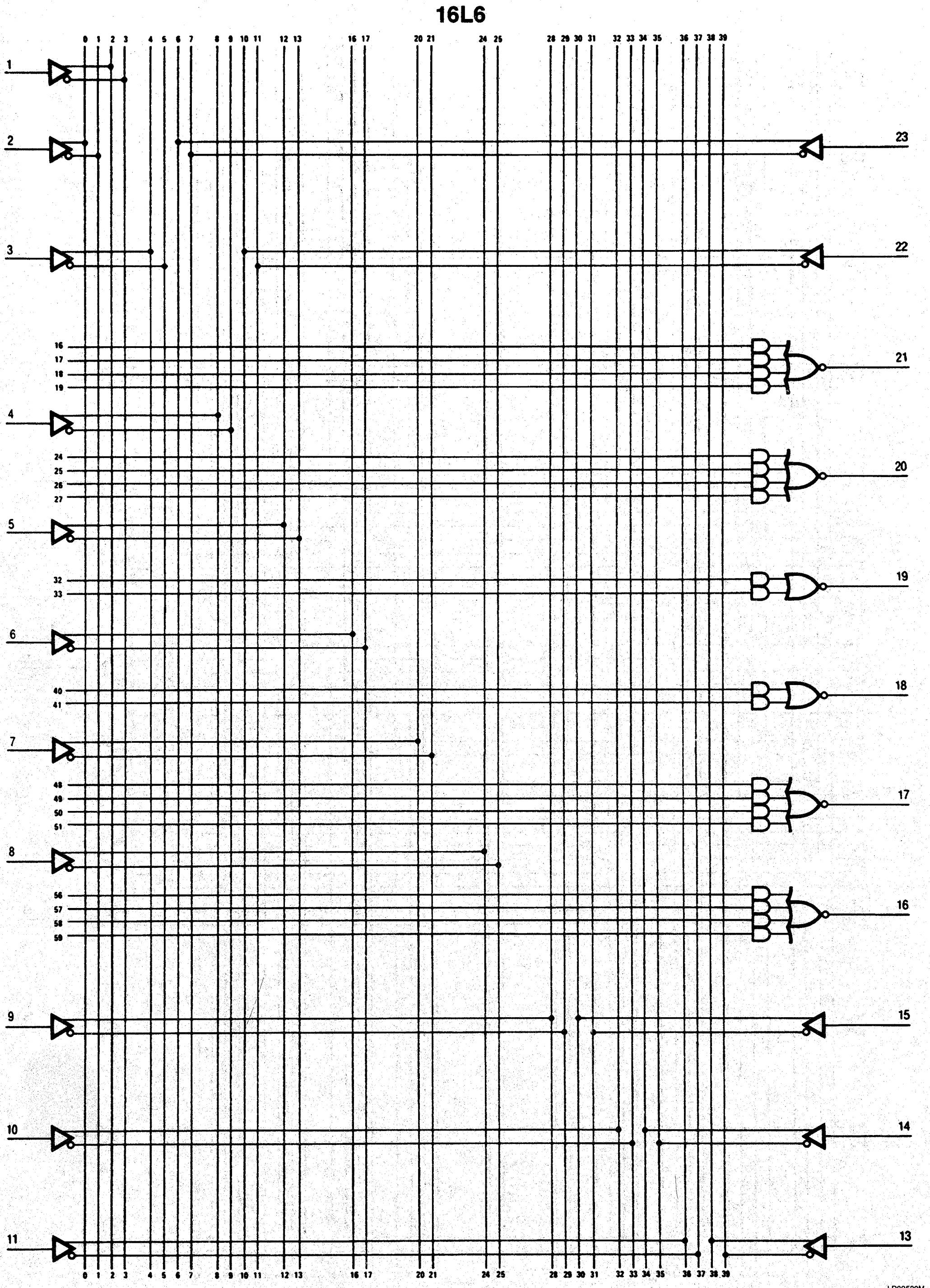
[1] These are absolute values with respect to the ground pin on the device and include all overshoots due to system and/or tester noise. Do not attempt to test these values without suitable equipment.

[2] No more than one output should be shorted at a time and duration of the short circuit should not exceed one second.


12L10
0 1 2 3 4 5 8 9 12 13 16 17 20 21 24 25 28 29 32 33 36 37 38 39
1
2
3
4
5
6
7
8
9
10
11
0
1
8
9
16
17
24
25
32
33
40
41
48
49
56
57
64
65
72
73
23
22
21
20
19
18
17
16
15
14
13
LD00510M

14L8

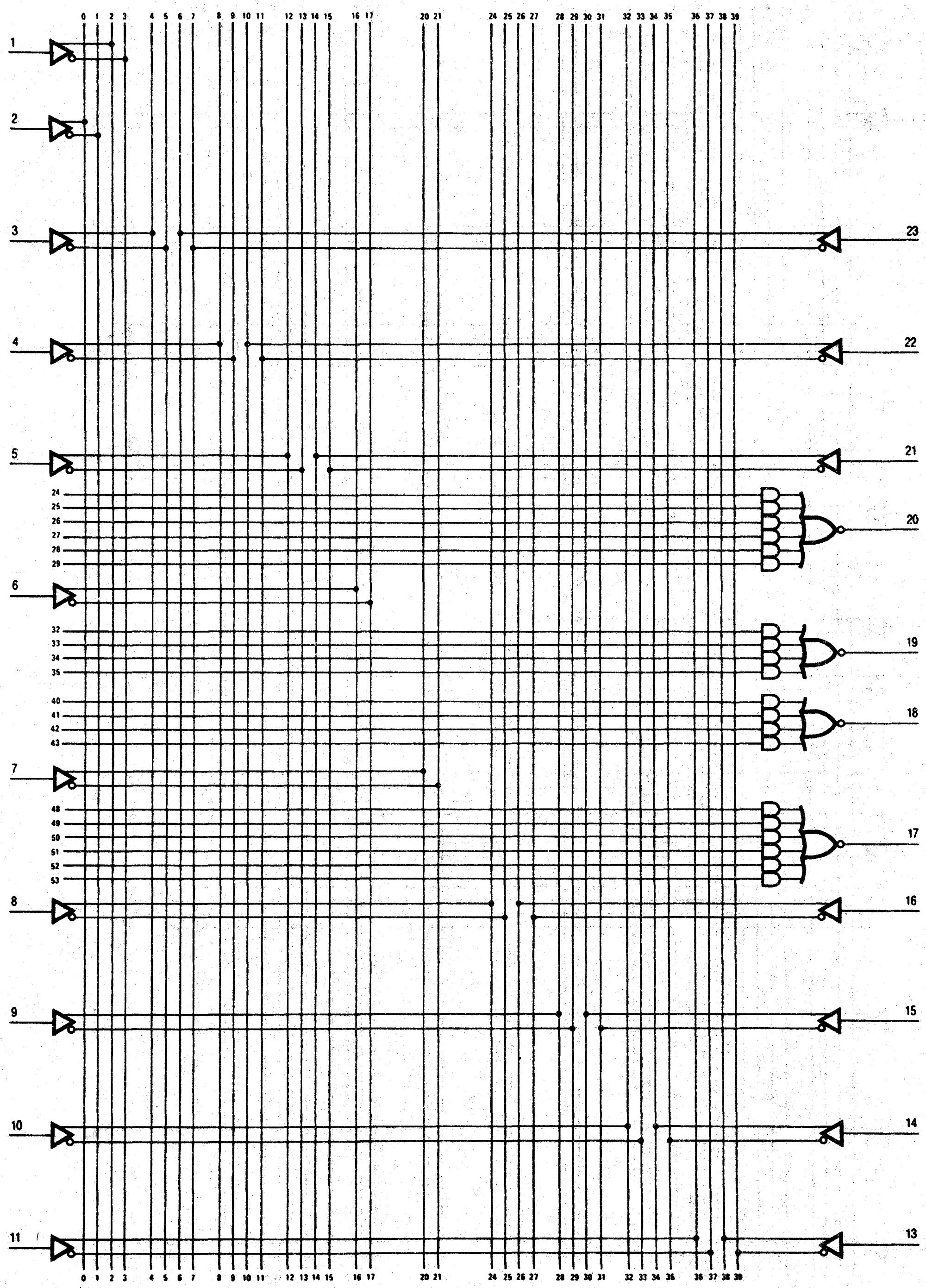

0 1 2 3 4 5 6 7 8 9 12 13 16 17 20 21 24 25 28 29 32 33 34 35 36 37 38 39
1
2
23
8
9
10
11
22
3
16
17
21
4
24
25
20
5
32
33
19
6
40
41
18
7
48
49
17
8
56
57
16
9
64
65
66
67
15
10
14
11
13
0 1 2 3 4 5 6 7 8 9 12 13 16 17 20 21 24 25 28 29 32 33 34 35 36 37 38 39
LD00520M


16L6
0 1 2 3 4 5 6 7 8 9 10 11 12 13 16 17 20 21 24 25 28 29 30 31 32 33 34 35 36 37 38 39
1
2
23
3
22
16
17
18
19
21
4
24
25
26
27
20
5
32
33
19
6
40
41
18
7
48
49
50
51
17
8
56
57
58
59
16
9
15
10
14
11
13
0 1 2 3 4 5 6 7 8 9 10 11 12 13 16 17 20 21 24 25 28 29 30 31 32 33 34 35 36 37 38 39

LD00530M

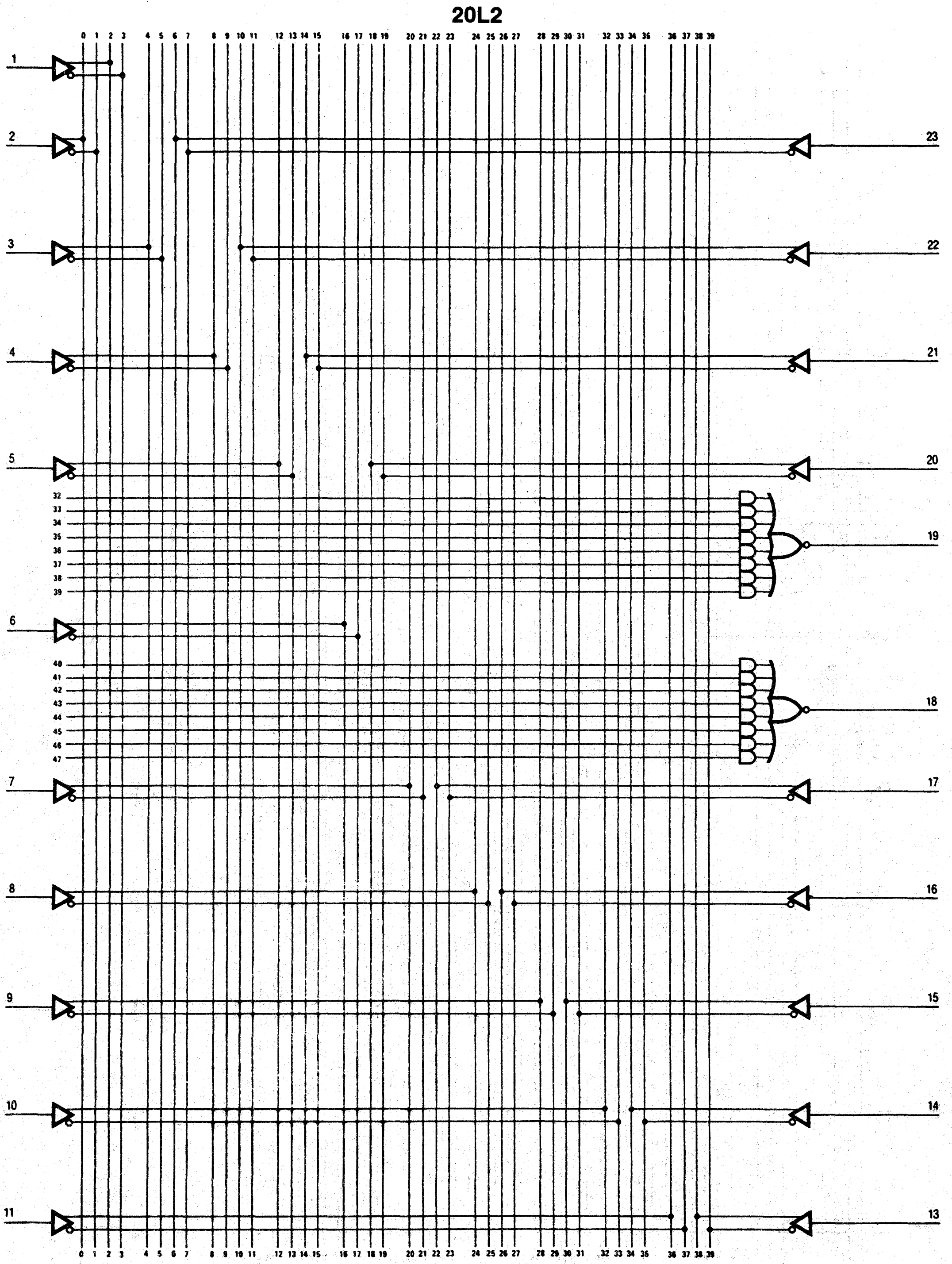
Small 24 Series
18L4 Logic Diagram

18L4



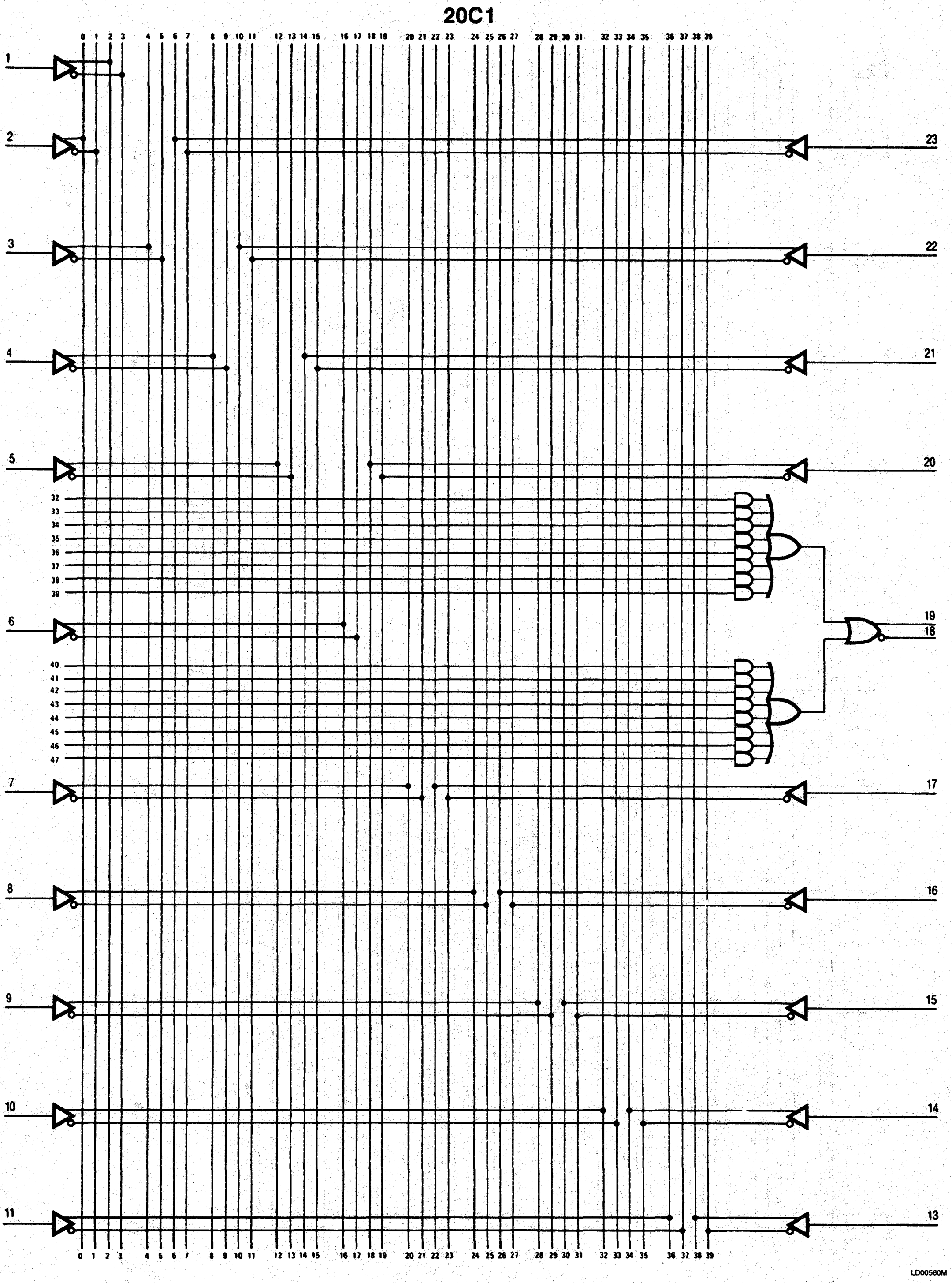
LD00540M

20L2 Logic Diagram



LD00550M

20C1 Logic Diagram



Small 24A Decoder Series

	INPUTS	OUTPUTS	t_{PD} (ns)	I_{CC} (mA)
PAL6L16A	6	16	25	90
PAL8L14A	8	14	25	90

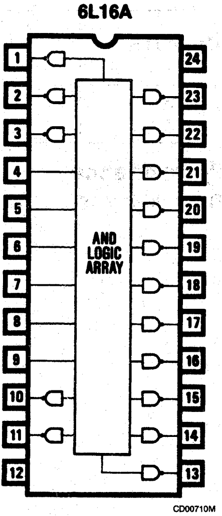
Description

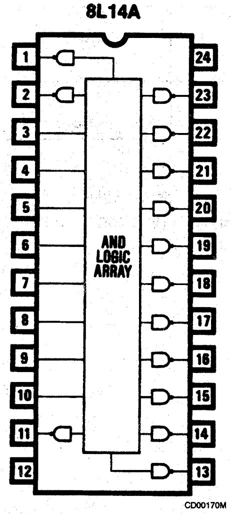
The Small 24A Decoder Series provides a wide number of outputs, especially useful in decoding applications. These two parts implement simple combinatorial logic.

Performance

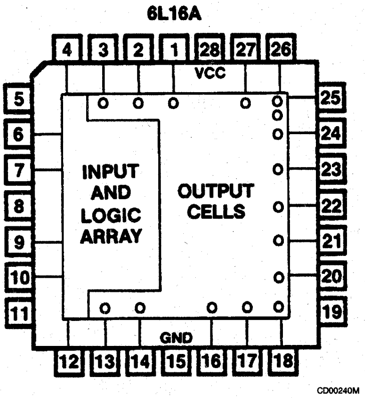
These devices offer 25ns speed at only 90mA supply current.

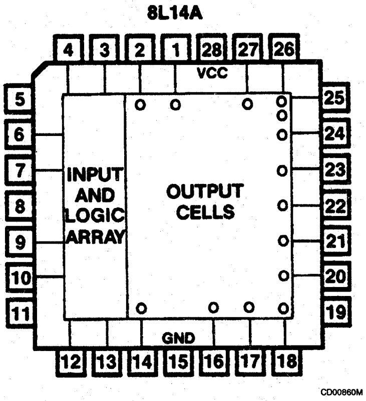
DIP Pinouts





PLCC Pinouts





Operating Conditions

SYMBOL	PARAMETER	COMMERCIAL[1]			UNIT
		MIN	TYP	MAX	
V_{CC}	Supply voltage	4.75	5	5.25	V
T_A	Operating free-air temperature	0	25	75	°C

Electrical Characteristics Over Operating Conditions

SYMBOL	PARAMETER	TEST CONDITION		MIN	TYP	MAX	UNIT
V_{IL}[2]	Low-level input voltage					0.8	V
V_{IH}[2]	High-level input voltage			2			V
V_{IC}	Input clamp voltage	V_{CC} = MIN	I_I = −18mA		−0.8	−1.5	V
I_{IL}	Low-level input current	V_{CC} = MAX	V_I = 0.4V		−0.02	−0.25	mA
I_{IH}	High-level input current	V_{CC} = MAX	V_I = 2.4V			25	μA
I_I	Maximum input current	V_{CC} = MAX	V_I = 5.5V			1	mA
V_{OL}	Low-level output voltage	V_{CC} = MIN	I_{OL} = 8mA		0.3	0.5	V
V_{OH}	High-level output voltage	V_{CC} = MIN	I_{OH} = −3.2mA	2.4	2.8		V
I_{OS}[3]	Output short-circuit current	V_{CC} = 5V	V_O = 0V	−30	−70	−130	mA
I_{CC}	Supply current	V_{CC} = MAX			60	90	mA

Switching Characteristics Over Operating Conditions

SYMBOL	PARAMETER	TEST CONDITIONS	COMMERCIAL			UNIT
			MIN	TYP	MAX	
t_{PD}	Input to output propagation delay	R_1 = 560Ω R_2 = 1.1KΩ		15	25	ns

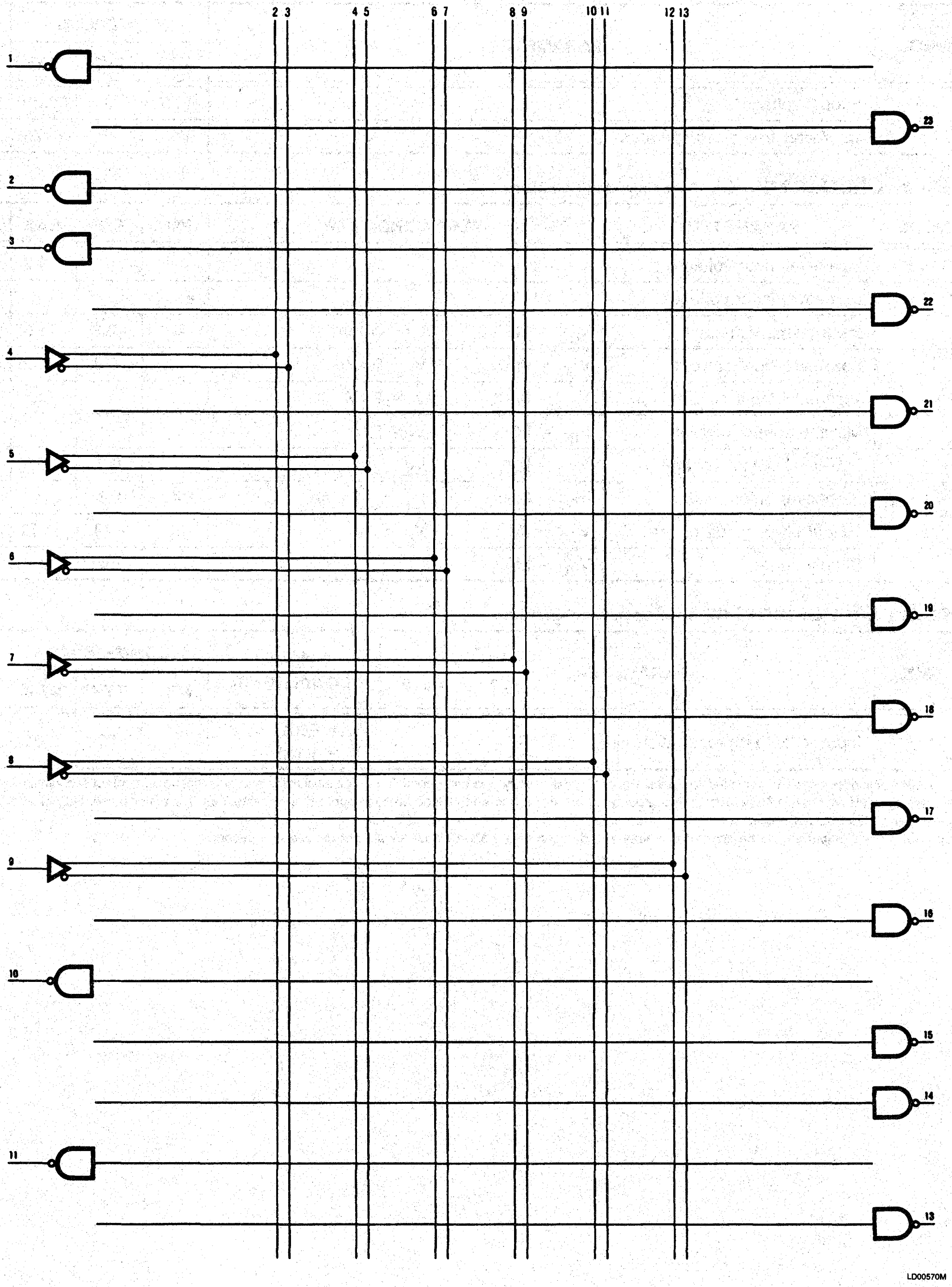
[1] The PAL24A Decoder Series is designed to operate over the full military operating conditions. For availability and specifications, contact Monolithic Memories.

[2] These are absolute voltages with respect to the ground pin on the device and include all overshoots due to system and/or tester noise. Do not attempt to test these values without suitable equipment.

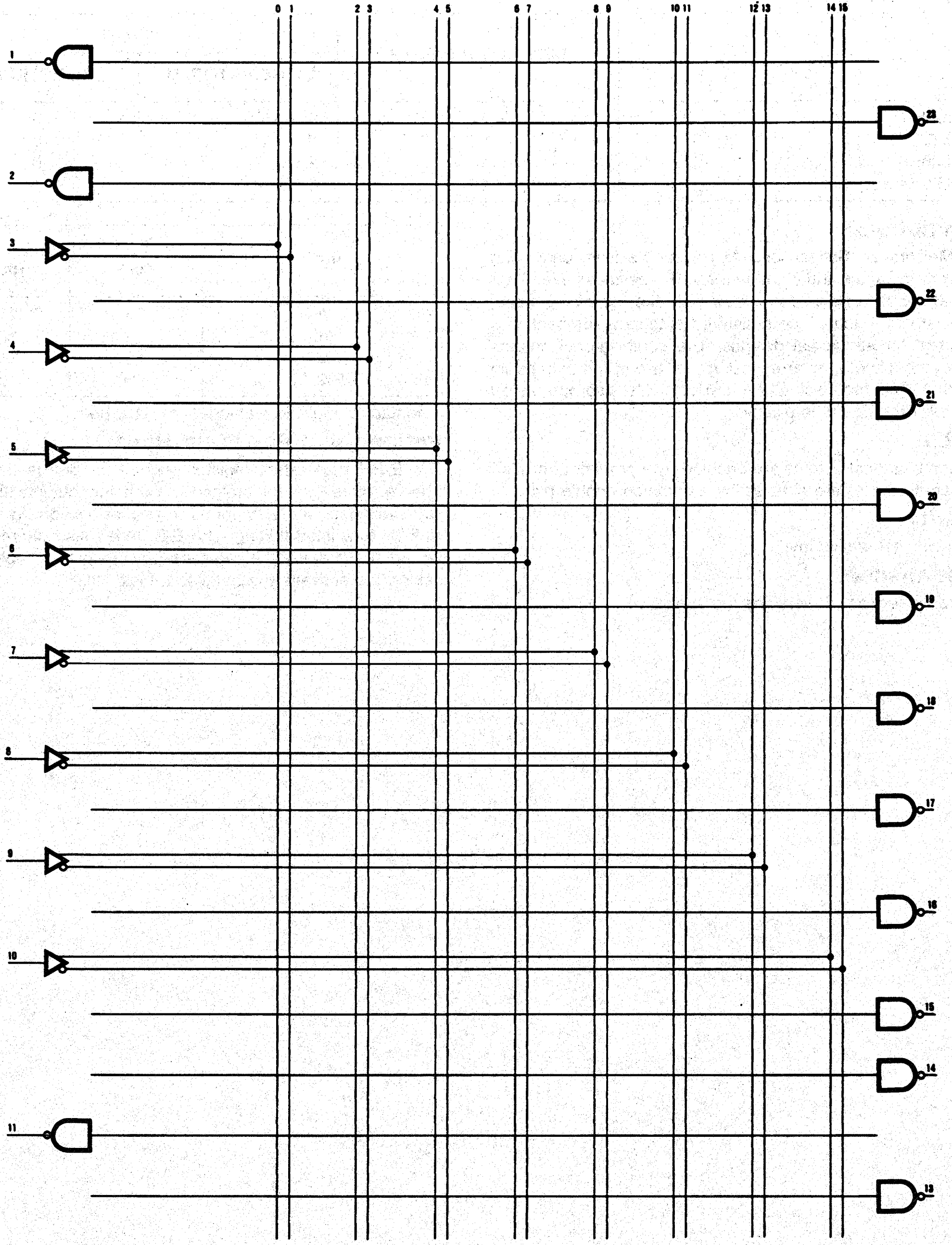
[3] No more than one output should be shorted at a time and duration of the short circuit should not exceed one second.

6L16A Logic Diagram

6L16A



8L14A



LD00580M

Medium 24 Series

	DEDICATED INPUTS	OUTPUTS	
		COMBINATORIAL	REGISTERED
PAL20L8	12	8 (6 I/O)	0
PAL20R8	10	0	8
PAL20R6	10	2 I/O	6
PAL20R4	10	4 I/O	4

Description

The Medium 24 Series consists of four devices, each with twenty array inputs and eight outputs. The devices have either 0, 4, 6, or 8 registered outputs, with the remaining being combinatorial. Each of the registered outputs feeds back into the array, for sequential designs. The combinatorial outputs also feed back into the array, except for two of the outputs on the 20L8. This feedback allows the output to also operate as an input if the output is disabled.

Enable

The combinatorial outputs are enabled by a product term. The registered outputs are enabled by a common enable pin.

Polarity

All outputs are active low.

Performance

Several speed/power versions are available:

Suffix	t_{PD} (ns)	I_{CC} (mA)
A	25	210
A-2	35	105
B	15	210
B-2 *	25	105

* contact Monolithic Memories for datasheet

Preload and Power-up Reset

The B-2 Series offers register preload for device testability. The registers can be preloaded from the outputs by using supervoltages (see waveforms at end of section) in order to simplify functional testing. The B-2 Series also offers Power-up Reset, whereby the registers power up to a logic LOW, setting the active-low outputs to a logic HIGH.

CMOS ZPAL™ 24 Series
Features/Benefits

- CMOS technology provides zero standby power
- Lowest power 24-pin PAL® device family; consumes only 3mA/MHz
- 35ns maximum propagation delay
- Programmable replacement for CMOS/TTL logic
- Reduces chip count by greater than six to one
- Instant prototyping and easier board layout
- HC/HCT compatible for use in CMOS or TTL systems
- Offered over both the Commercial and Industrial temperature ranges
- Low-cost, one-time programmable SKINNYDIP® and PLCC packages save board space

Description

The CMOS ZPAL24 Series offers the first family of PAL devices with true CMOS power consumption. Under standby conditions (inputs and clock not changing), the devices consume a maximum current of 100μA, less than 1% that of the quarter-power PAL devices. This low power consumption allows the devices to be powered by a battery almost indefinitely.

While operating, the devices consume additional power only when the inputs or clock change. Power consumption is directly proportional to the frequency of changes to the inputs. I_{CC} is therefore specified as 3mA per 1MHz of operating frequency, starting from 5mA at 1MHz. Thus, the maximum current at 8MHz would be 5mA + 7x3mA, or 26mA.

The devices have HC and HCT compatible inputs and outputs for use in CMOS and TTL systems. This feature allows the ZPAL circuits to be used for direct replacement of discrete CMOS as well as TTL logic.

Areas of Application

- Portable computers
- Battery-operated instrumentation
- Low-power industrial equipment
- Standard CMOS/TTL logic replacement

Features

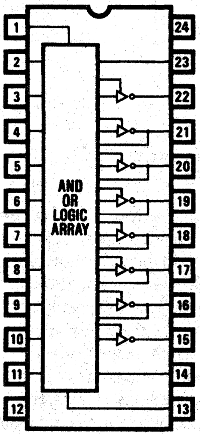
The CMOS ZPAL24 Series includes the four standard 24-pin PAL device architectures. All four devices have twenty array inputs and eight outputs, with varying numbers of registers: zero (20L8), four (20R4), six (20R6), and eight (20R8). The combinatorial outputs on the registered devices, and six of the outputs on the 20L8, are I/O pins that can be individually programmed as inputs or outputs. Each output register, a D-type flip-flop, also feeds back into the array, for implementation of synchronous state machine designs. Registered outputs are enabled by an external input, while the combinatorial outputs use a product term to control the enable function.

The basic PAL device architecture is a programmable AND array feeding a fixed OR array. The programmable AND array consists of a set of cells similar to those used in EPROMs. Erasable by UV light, the cells can be programmed and erased in the factory to ensure 100% programming and functional yields.

Windowed packages will be made available in the future, allowing erasure in the field. Windowed packages allow easy prototype testing and reconfiguration.

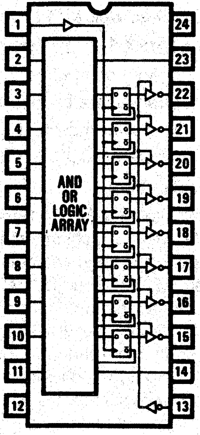
DIP Pinouts

20L8A/A-2/B



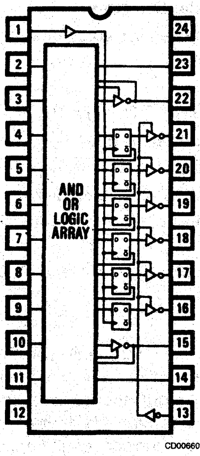
CD00650M

20R8A/A-2/B



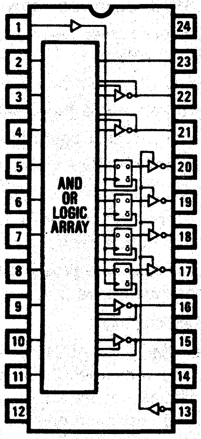
CD00150M

20R6A/A-2/B



CD00660M

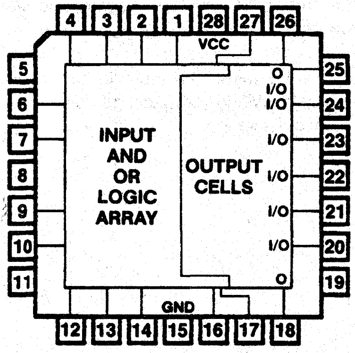
20R4A/A-2/B



CD00670M

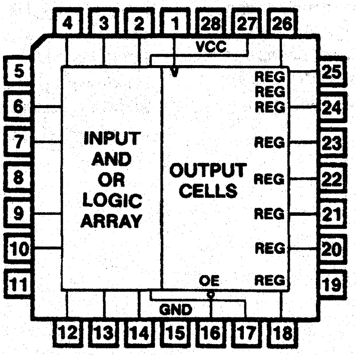
PLCC Pinouts

20L8A/A-2/B



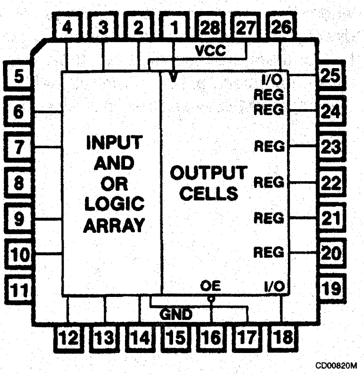
CD00210M

20R8A/A-2/B



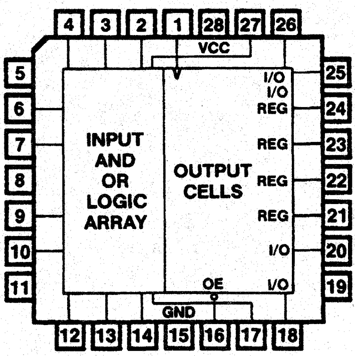
CD00800M

20R6A/A-2/B



CD00820M

20R4A/A-2/B



CD00220M

Operating Conditions

SYMBOL	PARAMETER		MILITARY			COMMERCIAL			UNIT
			MIN	TYP	MAX	MIN	TYP	MAX	
V_{CC}	Supply voltage		4.5	5	5.5	4.75	5	5.25	V
t_w	Width of clock	Low	20	7		15	7		ns
		High	20	7		15	7		
t_{su}	Set up time from input or feedback to clock	20R8A 20R6A 20R4A	30	15		25	15		ns
t_h	Hold time		0	−10		0	−10		ns
T_A	Operating free-air temperature		−55			0		75	°C
T_C	Operating case temperature				125				°C

Electrical Characteristics Over Operating Conditions

SYMBOL	PARAMETER	TEST CONDITIONS			MIN	TYP	MAX	UNIT
V_{IL}[1]	Low-level input voltage						0.8	V
V_{IH}[1]	High-level input voltage				2			V
V_{IC}	Input clamp voltage	V_{CC} = MIN		I_I = −18mA		−0.8	−1.5	V
I_{IL}[2]	Low-level input current	V_{CC} = MAX		V_I = 0.4V		−0.02	−0.25	mA
I_{IH}[2]	High-level input current	V_{CC} = MAX		V_I = 2.4V			25	µA
I_I	Maximum input current	V_{CC} = MAX		V_I = 5.5V			1	mA
V_{OL}	Low-level output voltage	V_{CC} = MIN	Mil	I_{OL} = 12mA		0.3	0.5	V
			Com	I_{OL} = 24mA				
V_{OH}	High-level output voltage	V_{CC} = MIN	Mil	I_{OH} = −2mA	2.4	2.8		V
			Com	I_{OH} = −3.2mA				
I_{OZL}[2]	Off-state output current	V_{CC} = MAX		V_O = 0.4V			−100	µA
I_{OZH}[2]				V_O = 2.4V			100	µA
I_{OS}[3]	Output short-circuit current	V_{CC} = 5V		V_O = 0V	−30	−90	−130	mA
I_{CC}	Supply current	V_{CC} = MAX				160	210	mA

[1] These are absolute voltages with respect to the ground pin on the device and include all overshoots due to system and/or tester noise. Do not attempt to test these values without suitable equipment.

[2] I/O pin leakage is the worst case of I_{IL} and I_{OZL} (or I_{IH} and I_{OZH}).

[3] No more than one output should be shorted at a time and duration of the short circuit should not exceed one second.

Switching Characteristics Over Operating Conditions

SYMBOL	PARAMETER		TEST CONDITIONS	MILITARY			COMMERCIAL			UNIT
				MIN	TYP	MAX	MIN	TYP	MAX	
t_{PD}	Input or feedback to output	20R6A 20R4A 20L8A	R_1 = 200Ω R_2 = 390Ω		15	30		15	25	ns
t_{CLK}	Clock to output or feedback				10	20		10	15	ns
t_{PZX}	Pin 13 to output enable except 20L8A				10	25		10	20	ns
t_{PXZ}	Pin 13 to output disable except 20L8A				11	25		11	20	ns
t_{PZX}	Input to output enable	20R6A 20R4A 20L8A			10	30		10	25	ns
t_{PXZ}	Input to output disable	20R6A 20R4A 20L8A			13	30		13	25	ns
f_{MAX}	Maximum frequency	20R8A 20R6A 20R4A		20	40		28.5	40		MHz

Operating Conditions

SYMBOL	PARAMETER		COMMERCIAL[1]			UNIT
			MIN	TYP	MAX	
V_{CC}	Supply voltage		4.75	5	5.25	V
t_w	Width of clock	Low	25	10		ns
		High	25	10		
t_{su}	Setup time from input or feedback to clock	20R8A-2, 20R6A-2, 20R4A-2	35	25		ns
t_h	Hold time		0	−15		ns
T_A	Operating free-air temperature		0	25	75	°C

Electrical Characteristics Over Operating Conditions

SYMBOL	PARAMETER	TEST CONDITION		MIN	TYP	MAX	UNIT
V_{IL}[2]	Low-level input voltage					0.8	V
V_{IH}[2]	High-level input voltage			2			V
V_{IC}	Input clamp voltage	V_{CC} = MIN	I_I = −18mA		−0.8	−1.5	V
I_{IL}[3]	Low-level input current	V_{CC} = MAX	V_I = 0.4V		−0.02	−0.25	mA
I_{IH}[3]	High-level input current	V_{CC} = MAX	V_I = 2.4V			25	μA
I_I	Maximum input current	V_{CC} = MAX	V_I = 5.5V			1	mA
V_{OL}	Low-level output voltage	V_{CC} = MIN	I_{OL} = 24mA		0.3	0.5	V
V_{OH}	High-level output voltage	V_{CC} = MIN	I_{OH} = −3.2mA	2.4	2.8		V
I_{OZL}[3]	Off-state output current	V_{CC} = MAX	V_O = 0.4V			−100	μA
I_{OZH}[3]			V_O = 2.4V			100	μA
I_{OS}[4]	Output short-circuit current	V_{CC} = 5V	V_O = 0V	−30	−70	−130	mA
I_{CC}	Supply current	V_{CC} = MAX			80	105	mA

Switching Characteristics Over Operating Conditions

SYMBOL	PARAMETER	TEST CONDITIONS	MILITARY			COMMERCIAL			UNIT
			MIN	TYP	MAX	MIN	TYP	MAX	
t_{PD}	Input or feedback to output 20L8A-2 20R6A-2 20R4A-2	Commercial R_1 = 200Ω R_2 = 390Ω		25	50		25	35	ns
t_{CLK}	Clock to output or feedback except 20L8A-2			15	25		15	25	ns
$t_{PXZ/ZX}$	Pin 13 to output disable/enable except 20L8A-2			15	25		15	25	ns
t_{PZX}	Input to output enable 20L8A-2 20R6A-2 20R4A-2			25	45		25	35	ns
t_{PXZ}	Input to output disable 20L8A-2 20R6A-2 20R4A-2			25	45		25	35	ns
f_{MAX}	Maximum frequency 20R8A-2 20R6A-2 20R4A-2		14	19		16	19		MHz

[1] The PAL24A-2 Series is designed to operate over the full military operating conditions. For availability and specifications, contact Monolithic Memories.

[2] These are absolute voltages with respect to the ground pin on the device and include all overshoots due to system and/or tester noise. Do not attempt to test these values without suitable equipment.

[3] I/O pin leakage is the worst case of I_{IL} and I_{OZL} (or I_{IH} and I_{OZH}).

[4] No more than one output should be shorted at a time and duration of the short circuit should not exceed one second.

Operating Conditions

SYMBOL	PARAMETER			COMMERCIAL[1]			UNIT
				MIN	TYP	MAX	
V_{CC}	Supply voltage			4.75	5	5.25	V
t_w	Width of clock	Low	20R8B, 20R6B, 20R4B	10	6		ns
		High		12	8		
t_{su}	Setup time from input or feedback to clock			15	10		ns
t_h	Hold time			0	−10		ns
T_A	Operating free-air temperature			0	25	75	°C

Electrical Characteristics Over Operating Conditions

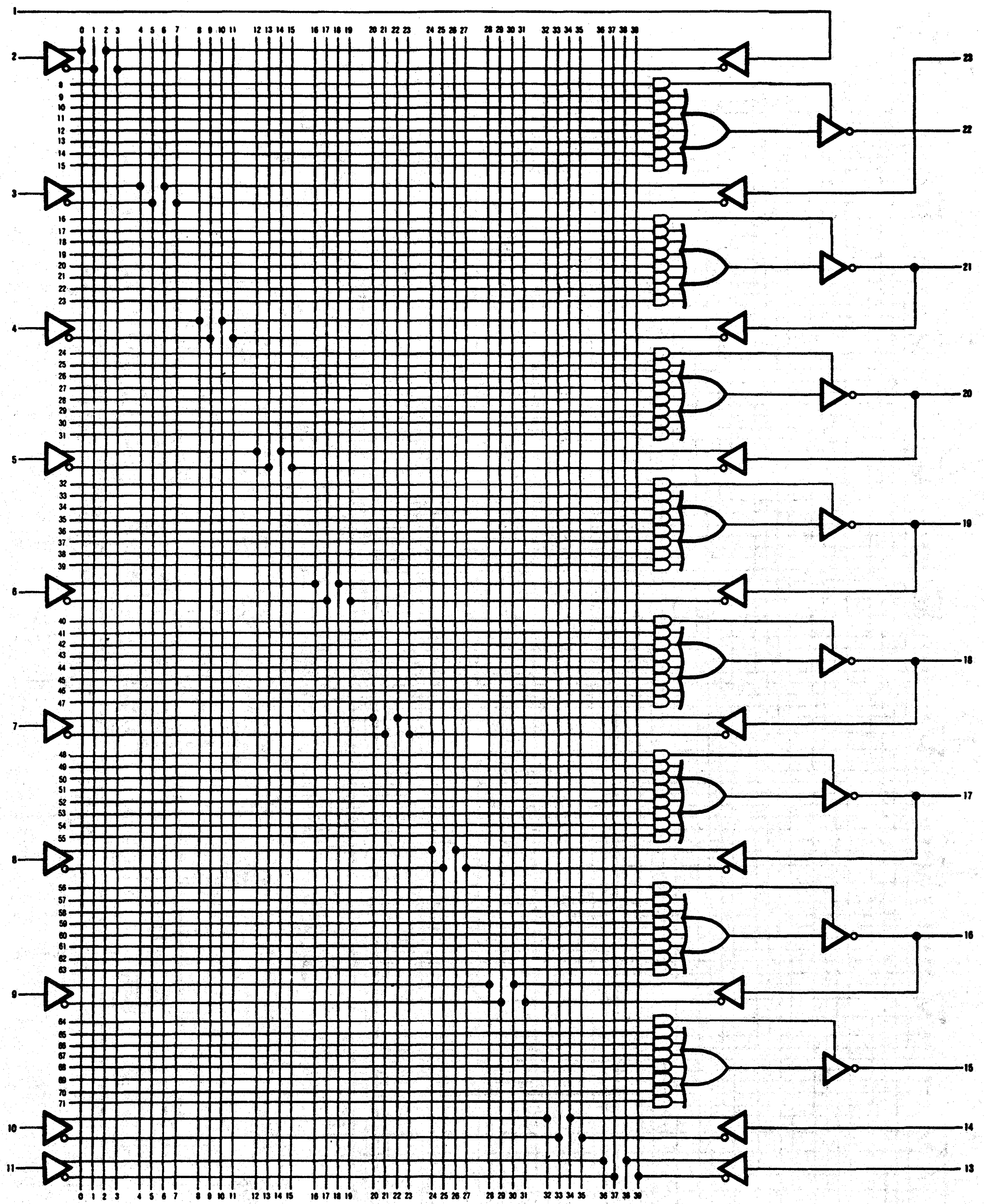
SYMBOL	PARAMETER	TEST CONDITION		COMMERCIAL			UNIT
				MIN	TYP	MAX	
V_{IL}[2]	Low-level input voltage					0.8	V
V_{IH}[2]	High-level input voltage			2			V
V_{IC}	Input clamp voltage	V_{CC} = MIN	I_I = −18mA		−0.8	−1.5	V
I_{IL}[3]	Low-level input current	V_{CC} = MAX	V_I = 0.4V		−0.02	−0.25	mA
I_{IH}[3]	High-level input current	V_{CC} = MAX	V_I = 2.4V			25	μA
I_I	Maximum input current	V_{CC} = MAX	V_I = 5.5V			1	mA
V_{OL}	Low-level output voltage	V_{CC} = MIN	I_{OL} = 24mA		0.3	0.5	V
V_{OH}	High-level output voltage	V_{CC} = MIN	I_{OH} = −3.2mA	2.4	2.8		V
I_{OZL}[3]	Off-state output current	V_{CC} = MAX	V_O = 0.4V			−100	μA
I_{OZH}[3]			V_O = 2.4V			100	μA
I_{OS}[4]	Output short-circuit current	V_{CC} = 5V	V_O = 0V	−30	−70	−130	mA
I_{CC}	Supply current	V_{CC} = MAX			140	210	mA

Switching Characteristics Over Operating Conditions

SYMBOL	PARAMETER		TEST CONDITIONS	COMMERCIAL			UNIT
				MIN	TYP	MAX	
t_{PD}	Input or feedback to output 20L8B, 20R6B, 20R4B		Commercial R_1 = 200Ω R_2 = 390Ω		12	15	ns
t_{CLK}	Clock to output or feedback except 20L8B				8	12	ns
t_{PZX}	Pin 13 to output enable except 20L8B				10	15	ns
t_{PXZ}	Pin 13 to output disable except 20L8B				8	12	ns
t_{PZX}	Input to output enable 20R6B, 20R4B, 20L8B				12	18	ns
t_{PXZ}	Input to output disable 20R6B, 20R4B, 20L8B				12	15	ns
f_{MAX}	Maximum frequency 20R8B, 20R6B, 20R4B	Feedback		37	40		MHz
		No feedback		45	50		

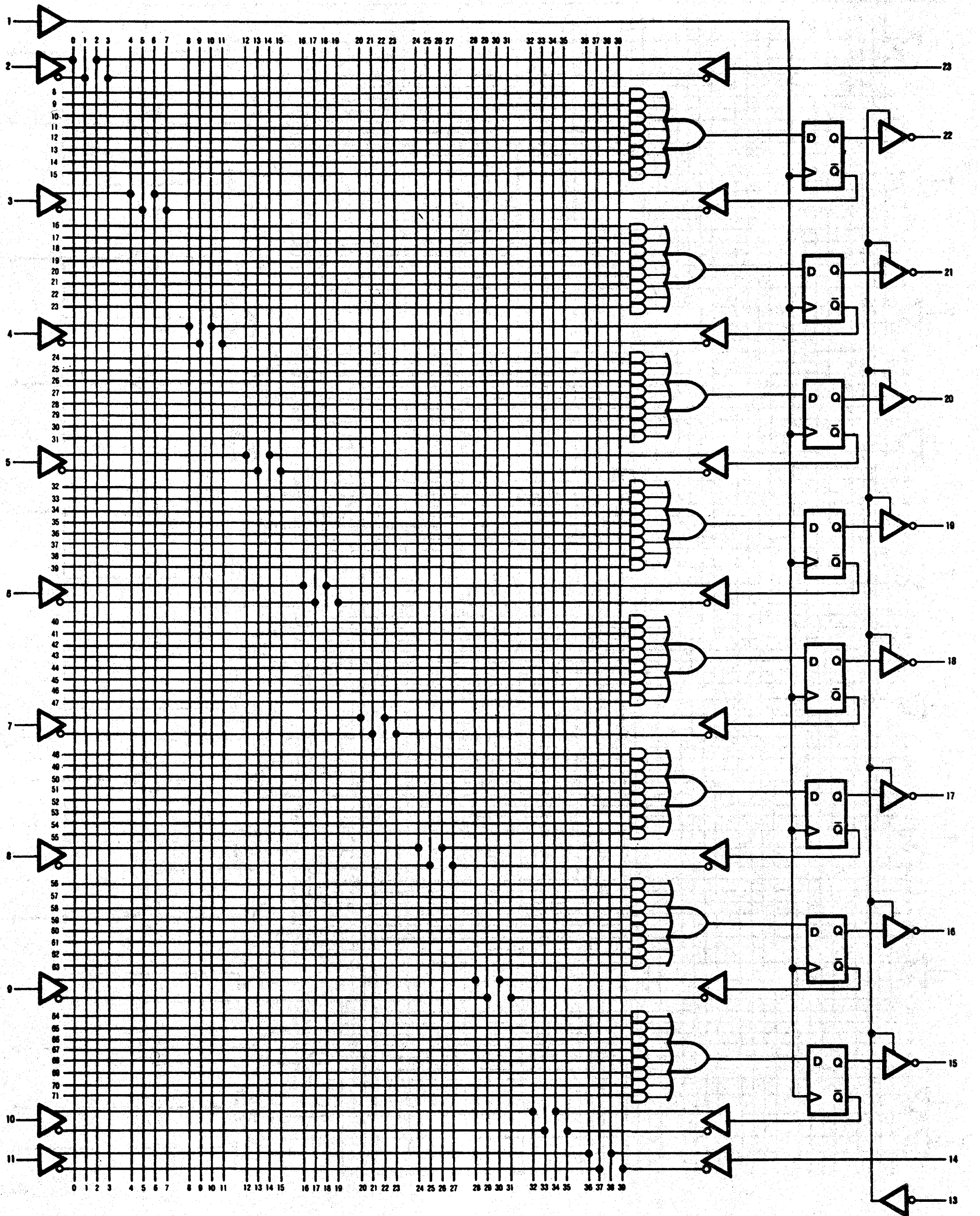
1. The PAL24B Series is designed to operate over the full military operating conditions. For availability and specifications, contact Monolithic Memories.
2. These are absolute voltages with respect to the ground pin on the device and include all overshoots due to system and/or tester noise. Do not attempt to test these values without suitable equipment.
3. I/O pin leakage is the worst case of I_{IL} and I_{OZL} (or I_{IH} and I_{OZH}).
4. No more than one output should be shorted at a time and duration of the short circuit should not exceed one second.

20L8



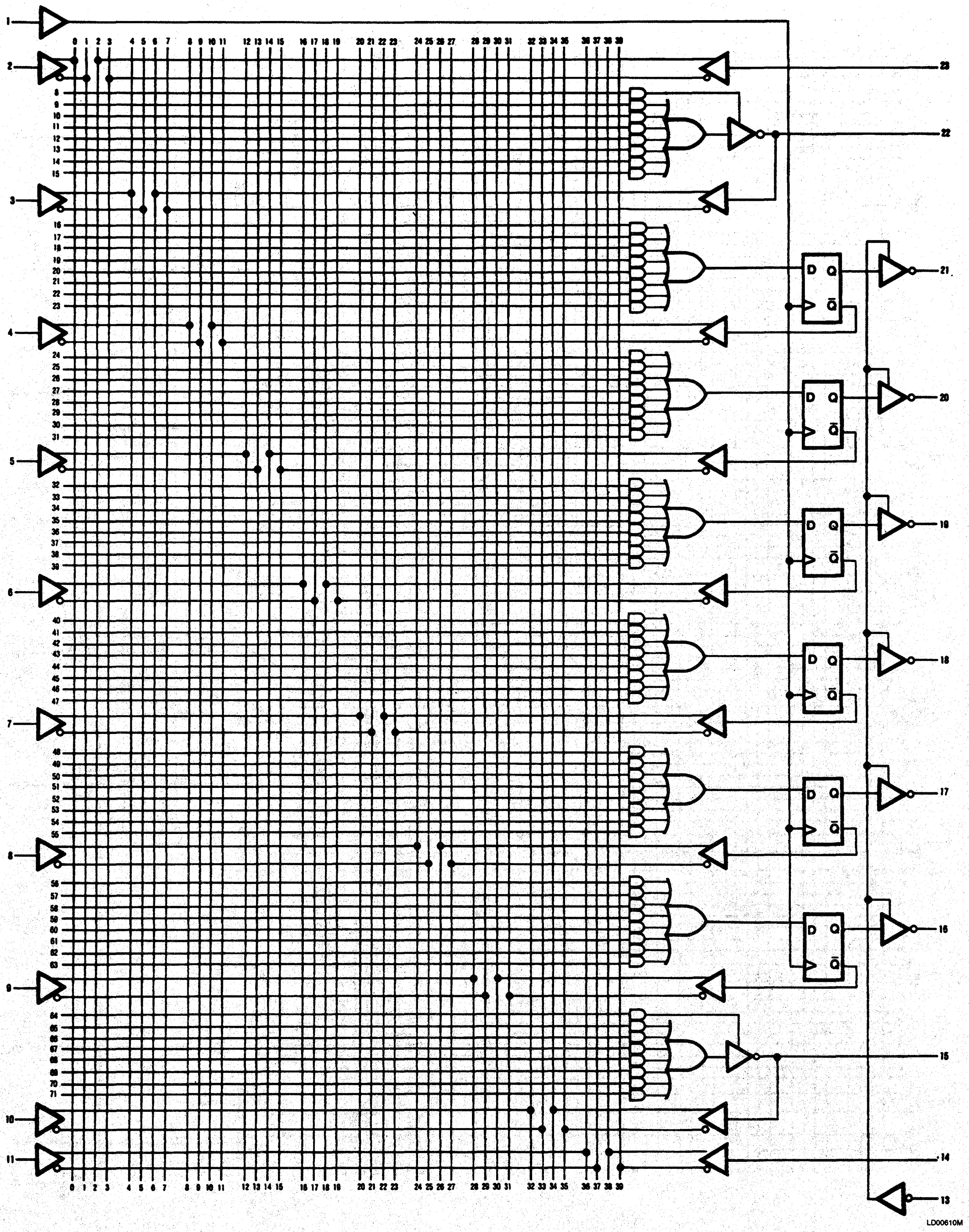
LD00590M

20R8

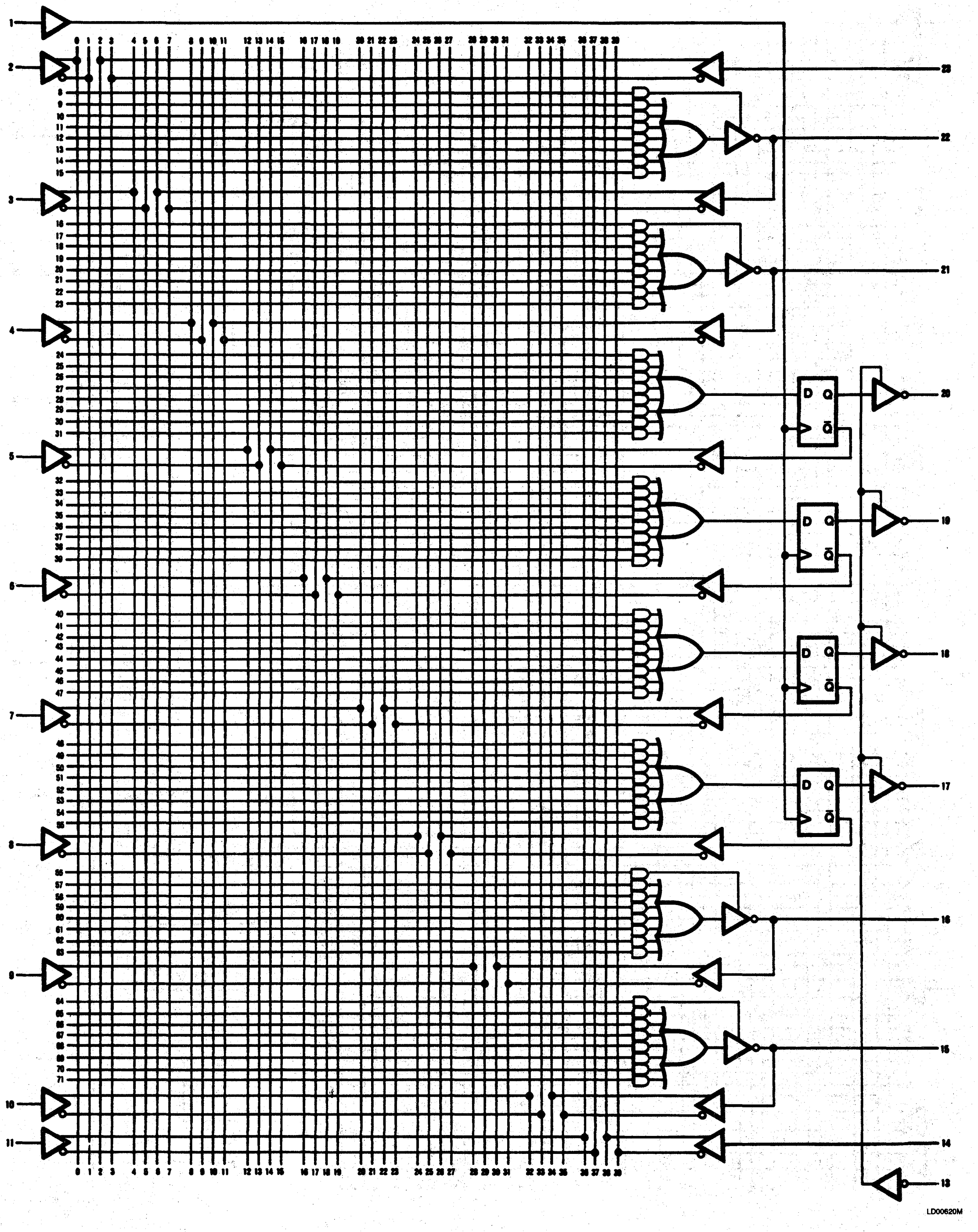


LD00600M

20R6


1
2
3
4
5
6
7
8
9
10
11
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39
8 9 10 11 12 13 14 15
16 17 18 19 20 21 22 23
24 25 26 27 28 29 30 31
32 33 34 35 36 37 38 39
40 41 42 43 44 45 46 47
48 49 50 51 52 53 54 55
56 57 58 59 60 61 62 63
64 65 66 67 68 69 70 71
D
Q
Q̄
23
22
21
20
19
18
17
16
15
14
13
LD00610M

20R4


1
2
3
4
5
6
7
8
9
10
11
13
14
15
16
17
18
19
20
21
22
23
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71
D
Q
Q̄
LD00620M

Medium 24X Series

	ARRAY INPUTS	OUTPUTS		STANDARD		HIGH SPEED	
		COMBINATORIAL	REGISTERED	t_{PD} (ns)	I_{CC} (mA)	t_{PD} (ns)	I_{CC} (mA)
PAL20L10	20	10	0	50	165	30	165
PAL20X10	20	0	10	50	180	30	180
PAL20X8	20	2	8	50	180	30	180
PAL20X4	20	6	4	50	180	30	180

Description

The PAL24X Series offers Exclusive-OR (XOR) gates preceding each register. The XOR gate has as its inputs two sums, each of two product terms. The XOR gate is very efficient for counting applications.

Enable

The combinatorial outputs are enabled by a product term. The registered outputs are enabled by a common enable pin.

Polarity

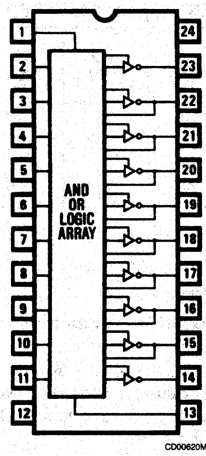
All outputs are active low.

Preload and Power-up Reset

The 24XA Series offers register preload for device testability. The registers can be preloaded from the outputs by using supervoltages (see waveforms at end of section) in order to simplify functional testing. The 24XA Series also offers Power-up Reset, whereby the registers power up to a logic LOW, setting the active-low outputs to a logic HIGH.

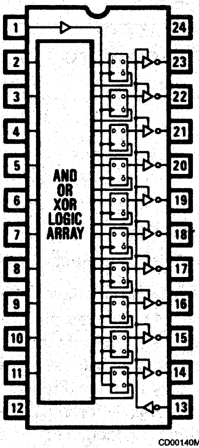
DIP Pinouts

20L10/A



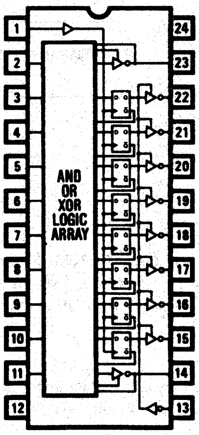
CD00620M

20X10/A



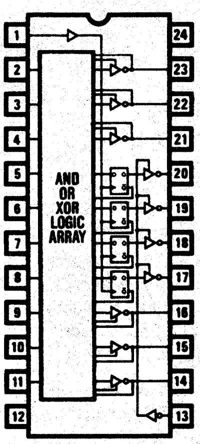
CD00140M

20X8/A



CD00630M

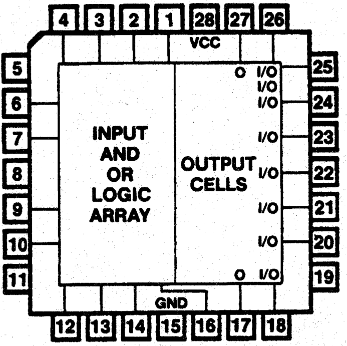
20X4/A



CD00640M

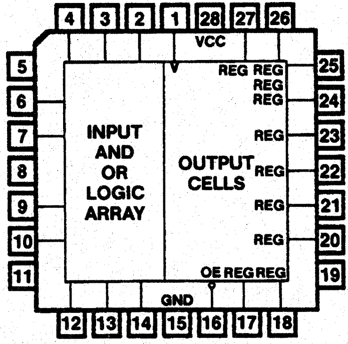
PLCC Pinouts

20L10/A



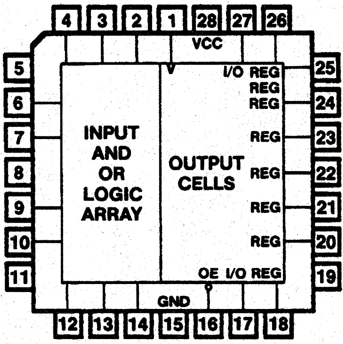
CD00770M

20X10/A



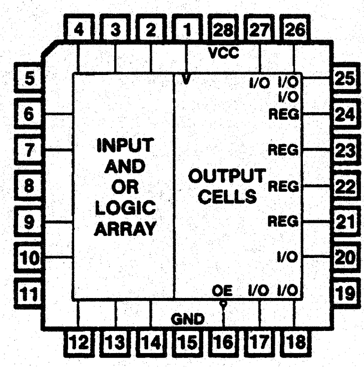
CD00200M

20X8/A



CD00780M

20X4/A



CD00790M

Operating Conditions

SYMBOL	PARAMETER		MILITARY			COMMERCIAL			UNIT
			MIN	TYP	MAX	MIN	TYP	MAX	
V_{CC}	Supply voltage		4.5	5	5.5	4.75	5	5.25	V
t_w	Width of clock	Low	40	20		35	20		ns
		High	30	10		25	10		
t_{su}	Setup time from input or feedback to clock	20X10, 20X8, 20X4	60	38		50	38		ns
t_h	Hold time		0	−15		0	−15		ns
T_A	Operating free-air temperature		−55			0		75	°C
T_C	Operating case temperature				125				°C

Electrical Characteristics (Over Operating Conditions)

SYMBOL	PARAMETER	TEST CONDITIONS			MIN	TYP	MAX	UNIT
V_{IL}[1]	Low-level input voltage						0.8	V
V_{IH}[1]	High-level input voltage				2			V
V_{IC}	Input clamp voltage	V_{CC} = MIN		$I_I = -18mA$		−0.8	−1.5	V
I_{IL}[2]	Low-level input current	V_{CC} = MAX		$V_I = 0.4V$		−0.02	−0.25	mA
I_{IH}[2]	High-level input current	V_{CC} = MAX		$V_I = 2.4V$			25	μA
I_I	Maximum input current	V_{CC} = MAX		$V_I = 5.5V$			1	mA
V_{OL}	Low-level output voltage	V_{CC} = MIN	Mil	$I_{OL} = 12mA$		0.3	0.5	V
			Com	$I_{OL} = 24mA$				
V_{OH}	High-level output voltage	V_{CC} = MIN	Mil	$I_{OH} = -2mA$	2.4	2.8		V
			Com	$I_{OH} = -3.2mA$				
I_{OZL}[2]	Off-state output current	V_{CC} = MAX		$V_O = 0.4V$			−100	μA
I_{OZH}[2]				$V_O = 2.4V$			100	μA
I_{OS}[3]	Output short-circuit current	V_{CC} = 5V		$V_O = 0V$	−30	−70	−130	mA
I_{CC}	Supply current	V_{CC} = MAX	20X10	20X8 20X4		120	180	mA
I_{CC}	Supply current	V_{CC} = MAX	20L10			90	165	mA

[1] These are absolute voltages with respect to the ground pin on the device and include all overshoots due to system and/or tester noise. Do not attempt to test these values without suitable equipment.

[2] I/O pin leakage is the worst case of I_{IL} and I_{OZL} (or I_{IH} and I_{OZH}).

[3] No more than one output should be shorted at a time and duration of the short circuit should not exceed one second.

Switching Characteristics (Over Operating Conditions)

SYMBOL	PARAMETER	TEST CONDITIONS	MILITARY			COMMERCIAL			UNIT
			MIN	TYP	MAX	MIN	TYP	MAX	
t_{PD}	Input or feedback to output 20X8, 20X4, 20L10	$R_1 = 200\Omega$ $R_2 = 390\Omega$		35	60		35	50	ns
t_{CLK}	Clock to output or feedback except 20L10			20	35		20	30	ns
$t_{PXZ/ZX}$	Pin 13 to output disable/enable except 20L10			20	45		20	35	ns
t_{PZX}	Input to output enable except 20X10			35	55		35	45	ns
t_{PXZ}	Input to output disable except 20X10			35	55		35	45	ns
f_{MAX}	Maximum frequency 20X10, 20X8, 20X4		10.5	16		12.5	16		MHz

Operating Conditions

SYMBOL	PARAMETER		COMMERCIAL[1] MIN	TYP	MAX	UNIT
V_{CC}	Supply voltage		4.75	5	5.25	V
t_w	Width of clock	Low	25	15		ns
		High	15	7		
t_{su}	Setup time from input or feedback to clock	20X10A, 20X8A, 20X4A	30	20		ns
t_h	Hold time		0	-15		ns
T_A	Operating free-air temperature		0	25	75	°C

Electrical Characteristics (Over Operating Conditions)

SYMBOL	PARAMETER	TEST CONDITION		MIN	TYP	MAX	UNIT
V_{IL}[2]	Low-level input voltage					0.8	V
V_{IH}[2]	High-level input voltage			2			V
V_{IC}	Input clamp voltage	V_{CC} = MIN	I_I = −18mA		-0.8	−1.5	V
I_{IL}[3]	Low-level input current	V_{CC} = MAX	V_I = 0.4V		-0.02	-0.25	mA
I_{IH}[3]	High-level input current	V_{CC} = MAX	V_I = 2.4V			25	μA
I_I	Maximum input current	V_{CC} = MAX	V_I = 5.5V			1	mA
V_{OL}	Low-level output voltage	V_{CC} = MIN	I_{OL} = 24mA		0.3	0.5	V
V_{OH}	High-level output voltage	V_{CC} = MIN	I_{OH} = −3.2mA	2.4	2.8		V
I_{OZL}[3]	Off-state output current	V_{CC} = MAX	V_O = 0.4V			−100	μA
I_{OZH}[3]			V_O = 2.4V			100	μA
I_{OS}[4]	Output short-circuit current	V_{CC} = 5V	V_O = 0V	-30	-70	−130	mA
I_{CC}	Supply Current	V_{CC} = MAX	20X10A,20X8A,20X4A		140	180	mA
			20L10A		115	165	

Switching Characteristics (Over Operating Conditions)

SYMBOL	PARAMETER		TEST CONDITIONS	COMMERCIAL MIN	TYP	MAX	UNIT
t_{PD}	Input or feedback to output 20L10A, 20X8A, and 20X4A		Commercial R_1 = 200Ω R_2 = 360Ω		23	30	ns
t_{CLK}	Clock to output or feedback				10	15	ns
t_{PZX}	Pin 13 to output enable except 20L10A				11	20	ns
t_{PXZ}	Pin 13 to output disable except 20L10A				10	20	ns
t_{PZX}	Input to output enable	20X8A, 20X4A, and 20L10A			19	30	ns
t_{PXZ}	Input to output disable	20X8A, 20X4A, and 20L10A			15	30	ns
f_{MAX}	Maximum frequency	20X10A, 20X8A, and 20X4A		22.2	32		MHz

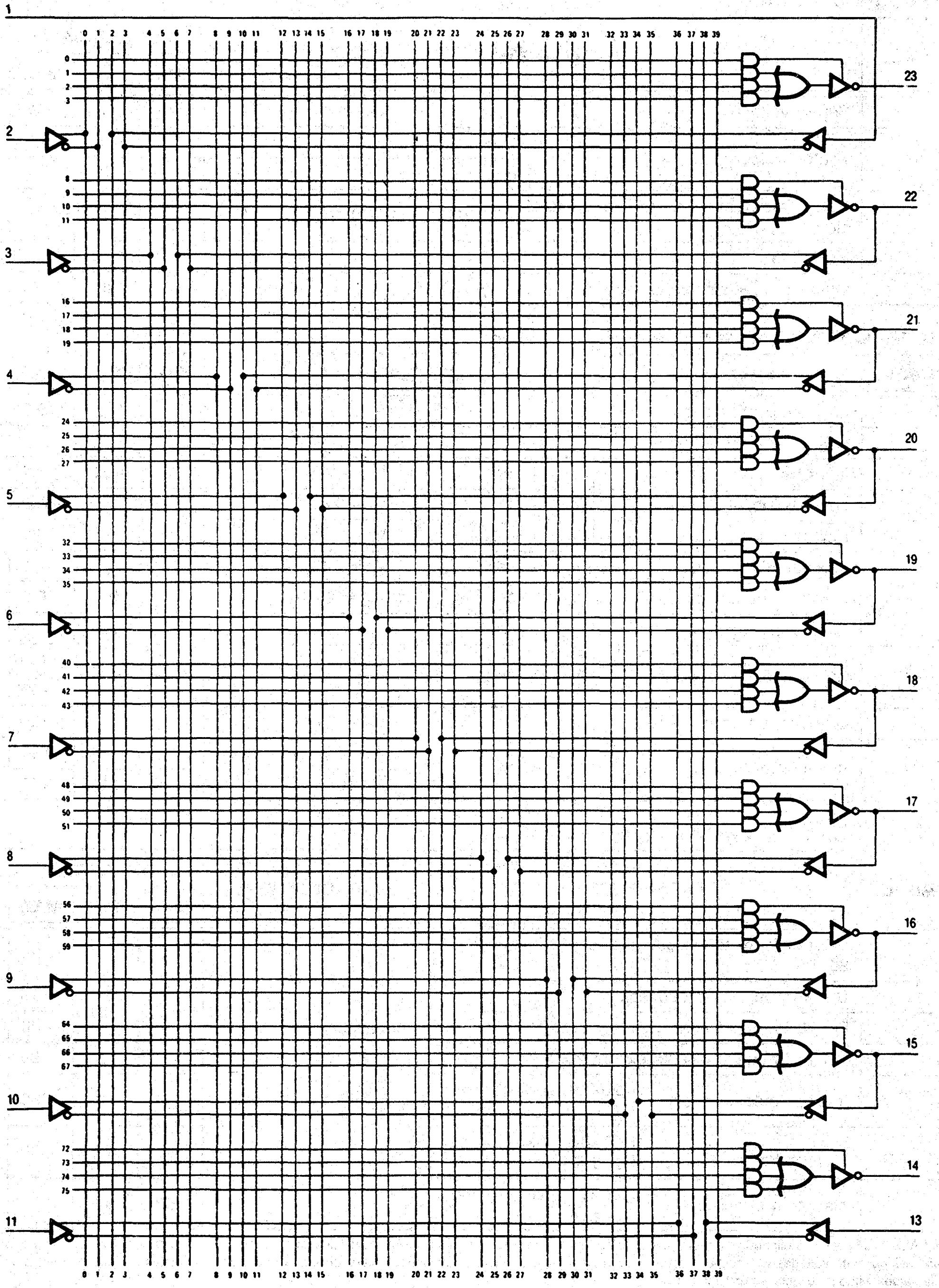
[1] The PAL24XA Series is designed to operate over the full military operating conditions. For availability and specifications, contact Monolithic Memories.

[2] These are absolute voltages with respect to the ground pin on the device and include all overshoots due to system and/or tester noise. Do not attempt to test these values without suitable equipment.

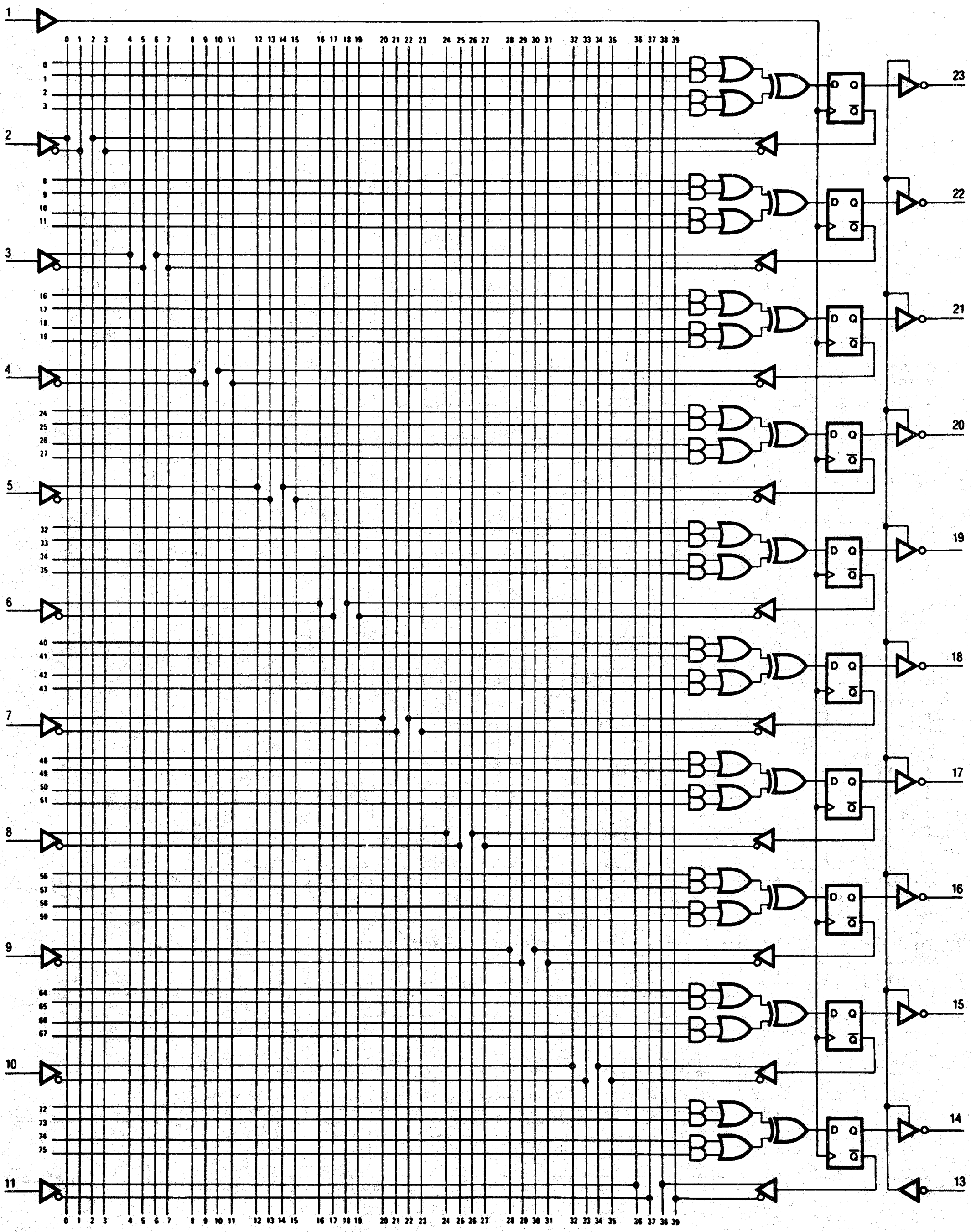
[3] I/O pin leakage is the worst case of I_{IL} and I_{OZL} (or I_{IH} and I_{OZH}).

[4] No more than one output should be shorted at a time and duration of the short circuit should not exceed one second.

20L10


1
2
3
4
5
6
7
8
9
10
11
23
22
21
20
19
18
17
16
15
14
13
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39
0 1 2 3
8 9 10 11
16 17 18 19
24 25 26 27
32 33 34 35
40 41 42 43
48 49 50 51
56 57 58 59
64 65 66 67
72 73 74 75
LD00630M

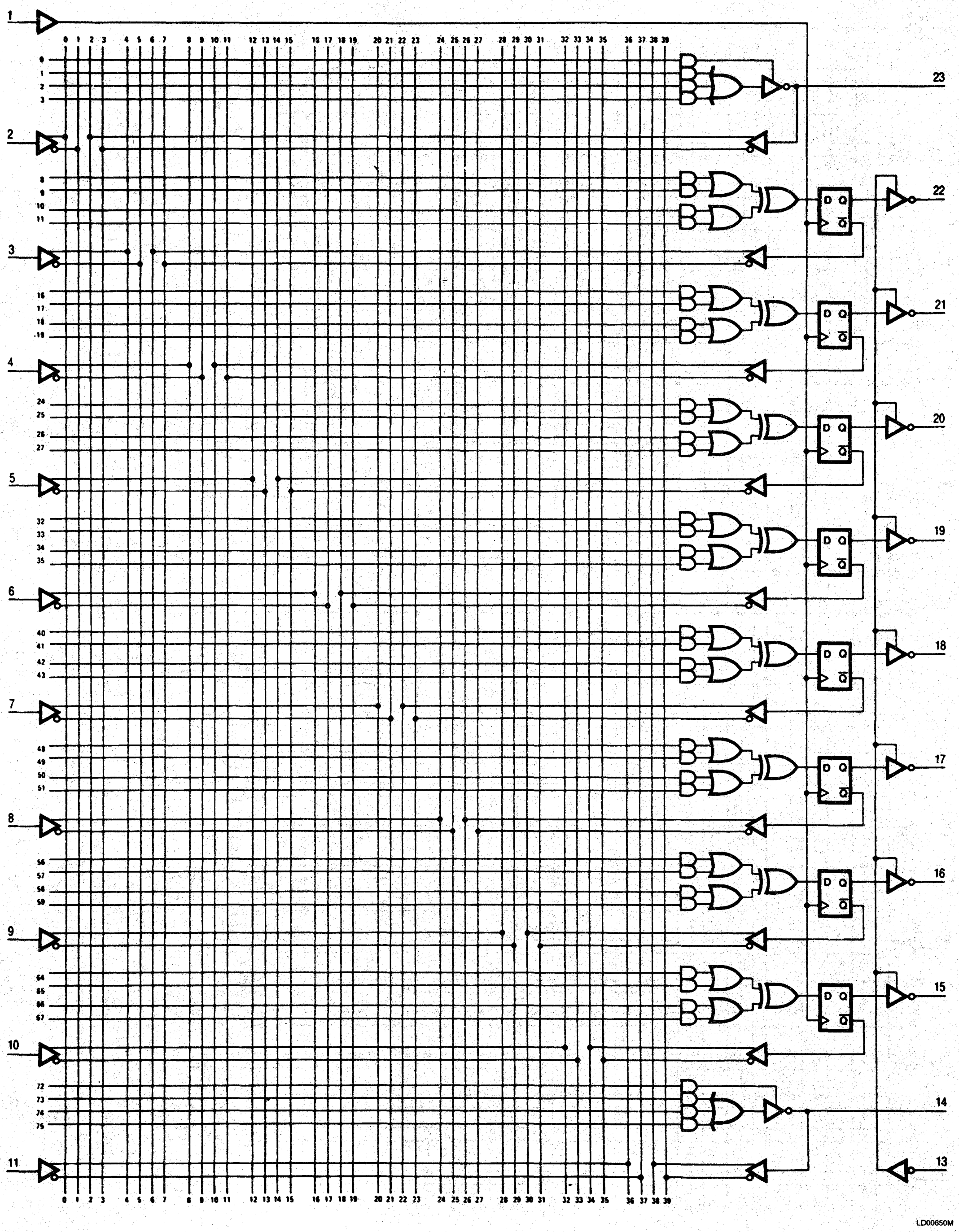
20X10



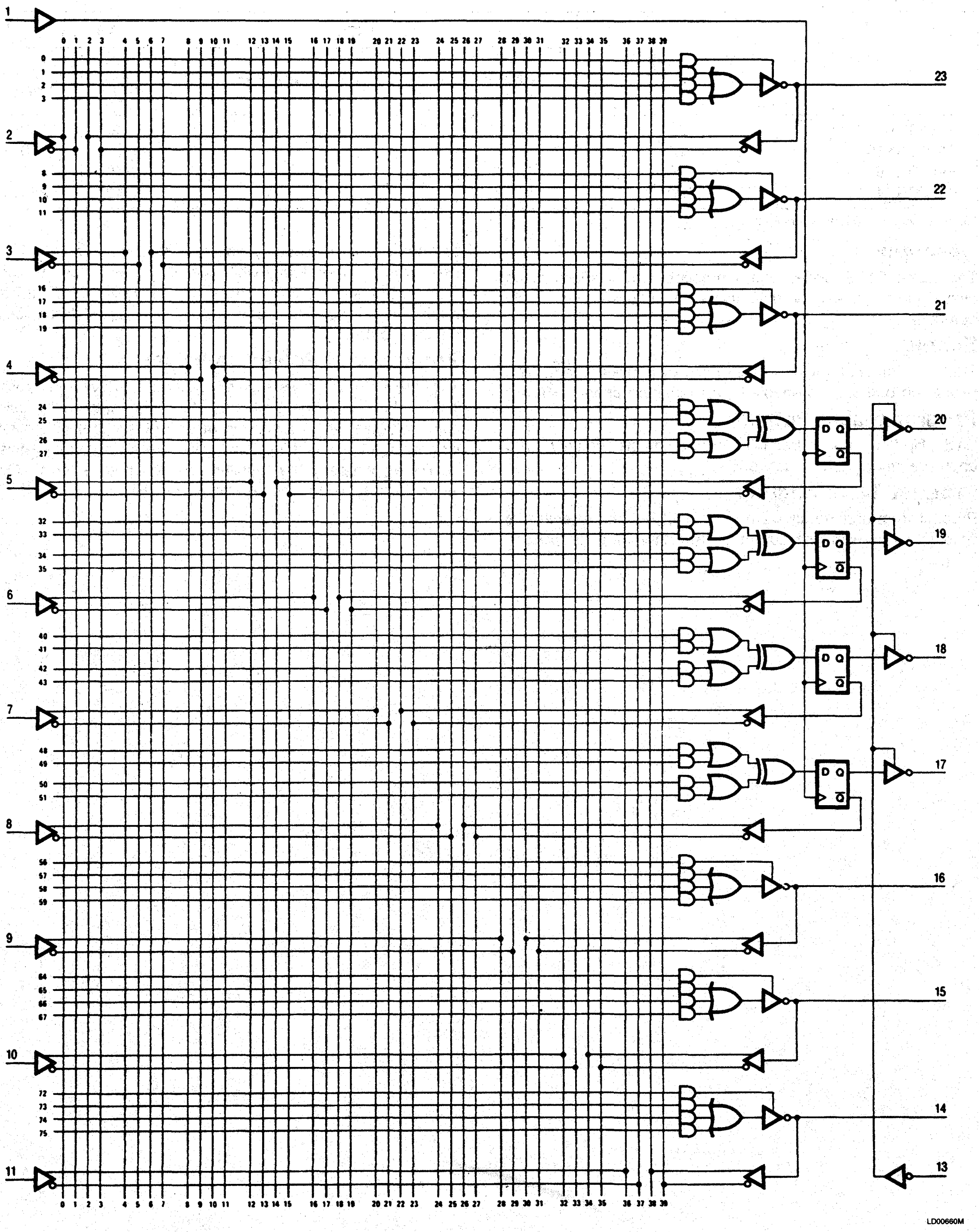
LD00640M

20X8 Logic Diagram

20X8



20X4


1
2
3
4
5
6
7
8
9
10
11
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39
0
1
2
3
8
9
10
11
16
17
18
19
24
25
26
27
32
33
34
35
40
41
42
43
48
49
50
51
56
57
58
59
64
65
66
67
72
73
74
75
D Q
Q̄
23
22
21
20
19
18
17
16
15
14
13
LD00660M

Large 24RS Series

	ARRAY INPUTS	OUTPUTS		STANDARD	
		COMBINATORIAL	REGISTERED	t_{PD}* (ns)	I_{CC} (mA)
PAL20S10	20	10	0	35/40	240
PAL20RS10	20	0	10	35	240
PAL20RS8	20	2	8	35/40	240
PAL20RS4	20	6	4	35/40	240

*35ns active low, 40ns active high

Description

The Large 24RS Series offers product term sharing, which allows up to sixteen product terms to be used at a single output.

Enable

The combinatorial outputs are enabled by a product term. The registered outputs are enabled by a common enable pin.

Programmable Polarity

Each flip-flop has individually programmable polarity. The unprogrammed state is active low.

Product Term Sharing

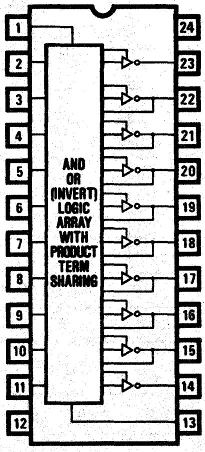
Product term sharing allows each pair of outputs to share its product terms with one output or the other (not both). Each pair has a total of sixteen product terms; thus, one output can use zero to sixteen terms while the other has sixteen to zero. Product terms can only be shared mutually exclusively. If both outputs need the same term, it must be created twice, once for each output.

Preload and Power-up Reset

The 24RS Series offers register preload for device testability. The registers can be preloaded from the outputs by using supervoltages (see waveforms at end of section) in order to simplify functional testing. The 24RS Series also offers Power-up Reset, whereby the registers power up to a logic LOW, setting the active-low outputs to a logic HIGH.

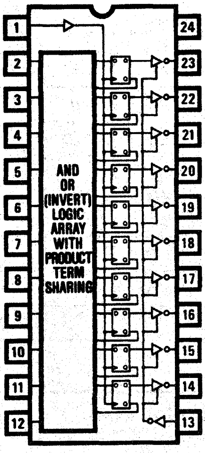
DIP Pinouts

20S10



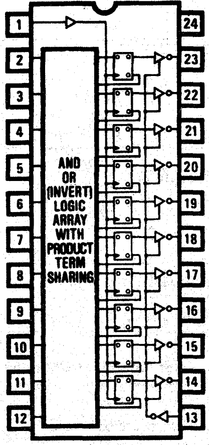
CD00680M

20RS10



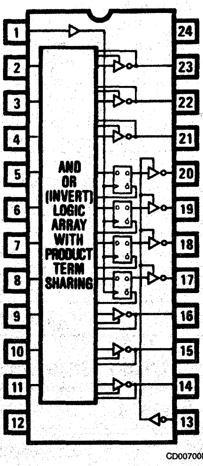
CD00690M

20RS8



CD00160M

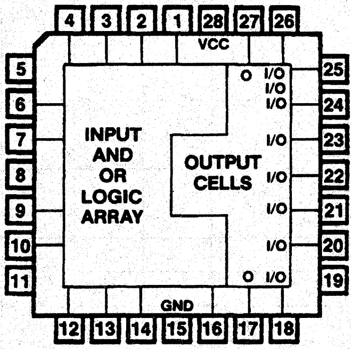
20RS4



CD00700M

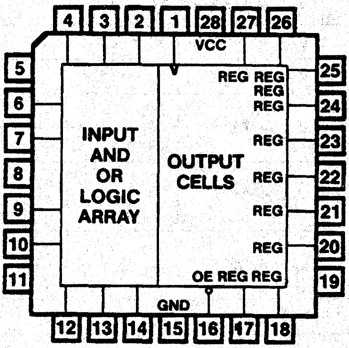
PLCC Pinouts

20S10



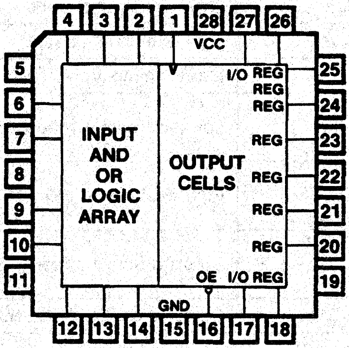
CD00830M

20RS10



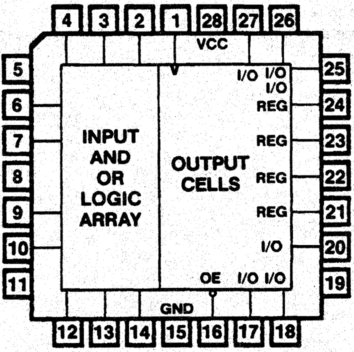
CD00230M

20RS8



CD00840M

20RS4



CD00850M

Operating Conditions

SYMBOL	PARAMETER			MILITARY			COMMERCIAL			UNIT
				MIN	TYP	MAX	MIN	TYP	MAX	
V_{CC}	Supply voltage			4.5	5	5.5	4.75	5	5.25	V
t_w	Width of clock	Low		20	10		15	10		ns
		High		20	10		15	10		
t_{su}	Setup time from input or feedback to clock	20RS10 20RS8 20RS4		40	25		35	25		ns
t_h	Hold time			0	−10		0	−10		ns
T_A	Operating free-air temperature			−55			0		75	°C
T_C	Operating case temperature					125				°C

Electrical Characteristics (Over Operating Conditions)

SYMBOL	PARAMETER	TEST CONDITION			MIN	TYP	MAX	UNIT
V_{IL}[1]	Low-level input voltage						0.8	V
V_{IH}[1]	High-level input voltage				2			V
V_{IC}	Input clamp voltage	V_{CC} = MIN	I_I = −18mA			−0.8	−1.5	V
I_{IL}[2]	Low-level input current	V_{CC} = MAX	V_I = 0.4V			−0.02	−0.25	mA
I_{IH}[2]	High-level input current	V_{CC} = MAX	V_I = 2.4V				25	μA
I_I	Maximum input current	V_{CC} = MAX	V_I = 5.5V				1	mA
V_{OL}	Low-level output voltage	V_{CC} = MIN	Mil	I_{OL} = 12mA		0.3	0.5	V
			Com	I_{OL} = 24mA				
V_{OH}	High-level output voltage	V_{CC} = MIN	Mil	I_{OH} = −2mA	2.4	2.8		V
			Com	I_{OH} = −3.2mA				
I_{OZL}[2]	Off-state output current	V_{CC} = MAX		V_O = 0.4V			−100	μA
I_{OZH}[2]				V_O = 2.4mA			100	
I_{OS}[3]	Output short-circuit current	V_{CC} = 5V		V_O = 0V	−30	−70	−130	mA
I_{CC}	Supply current	V_{CC} = MAX				175	240	mA

[1] These are absolute voltages with respect to the ground pin on the device and include all overshoots due to system and/or tester noise. Do not attempt to test these values without suitable equipment.

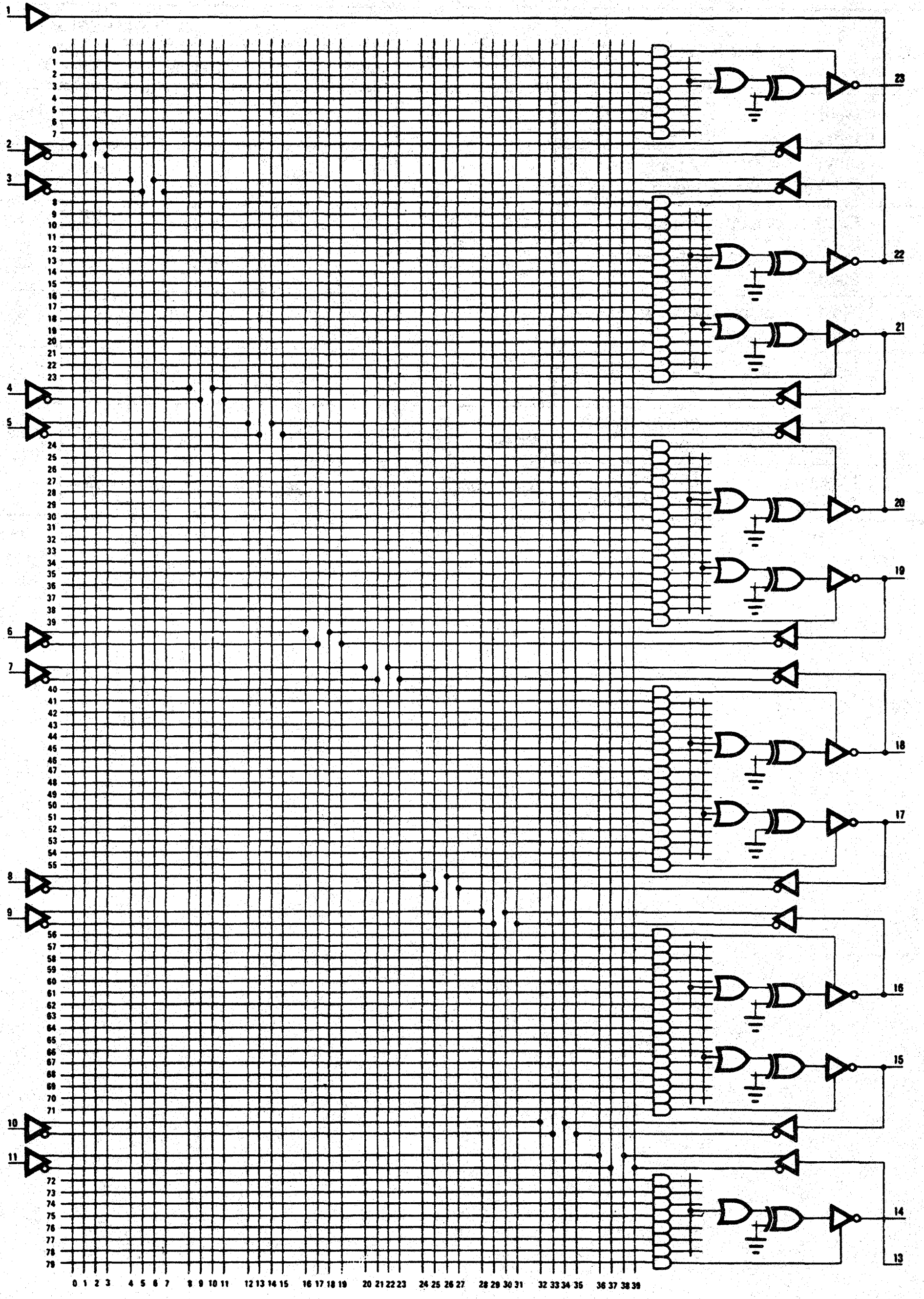
[2] I/O pin leakage is the worst case of I_{IL} and I_{OZL} (or I_{IH} and I_{OZH}).

[3] No more than one output should be shorted at a time and duration of the short circuit should not exceed one second.

Switching Characteristics (Over Operating Conditions)

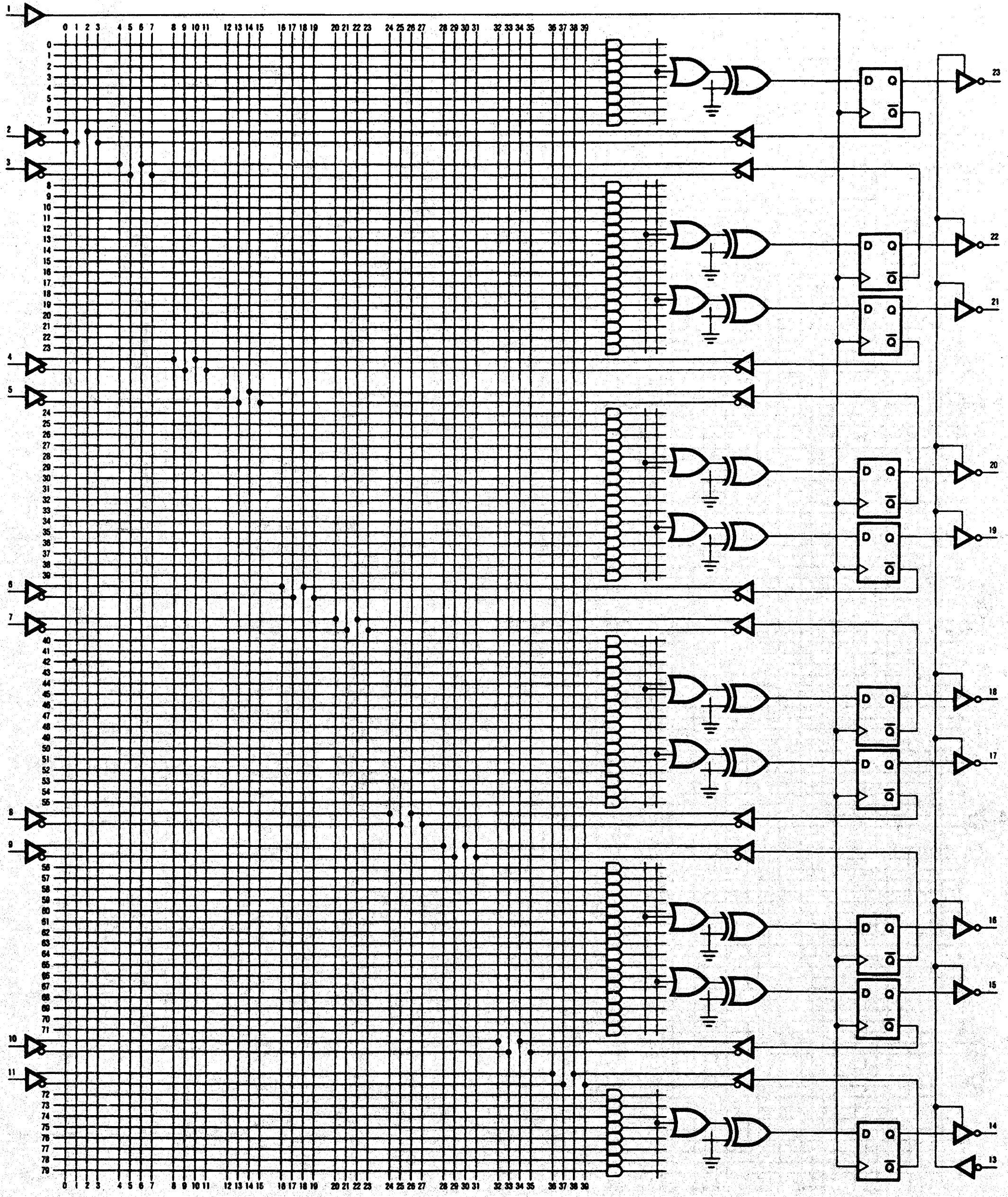
SYMBOL	PARAMETER		TEST CONDITIONS	MILITARY			COMMERCIAL			UNIT
				MIN	TYP	MAX	MIN	TYP	MAX	
t_{PD}	Input or feedback to output 20S10, 20RS8, 20RS4	Polarity fuse intact	Commercial R_1 = 200Ω R_2 = 390KΩ		25	40		25	35	ns
		Polarity fuse blown			30	45		30	40	
t_{CLK}	Clock to output or feedback				12	20		12	17	ns
t_{PZX}	Pin 13 to output enable except 20S10				10	25		10	20	ns
t_{PXZ}	Pin 13 to output disable except 20S10				11	25		11	20	ns
t_{PZX}	Input to output enable	20S10, 20RS8, 20RS4			25	35		25	35	ns
t_{PZX}	Input to output disable	20S10, 20RS8, 20RP4	Military R_1 = 390Ω R_2 = 750Ω		13	30		13	25	ns
f_{MAX}	Maximum frequency	20RS10, 20RS8, 20RS4		18	28		20	28		MHz

20S10



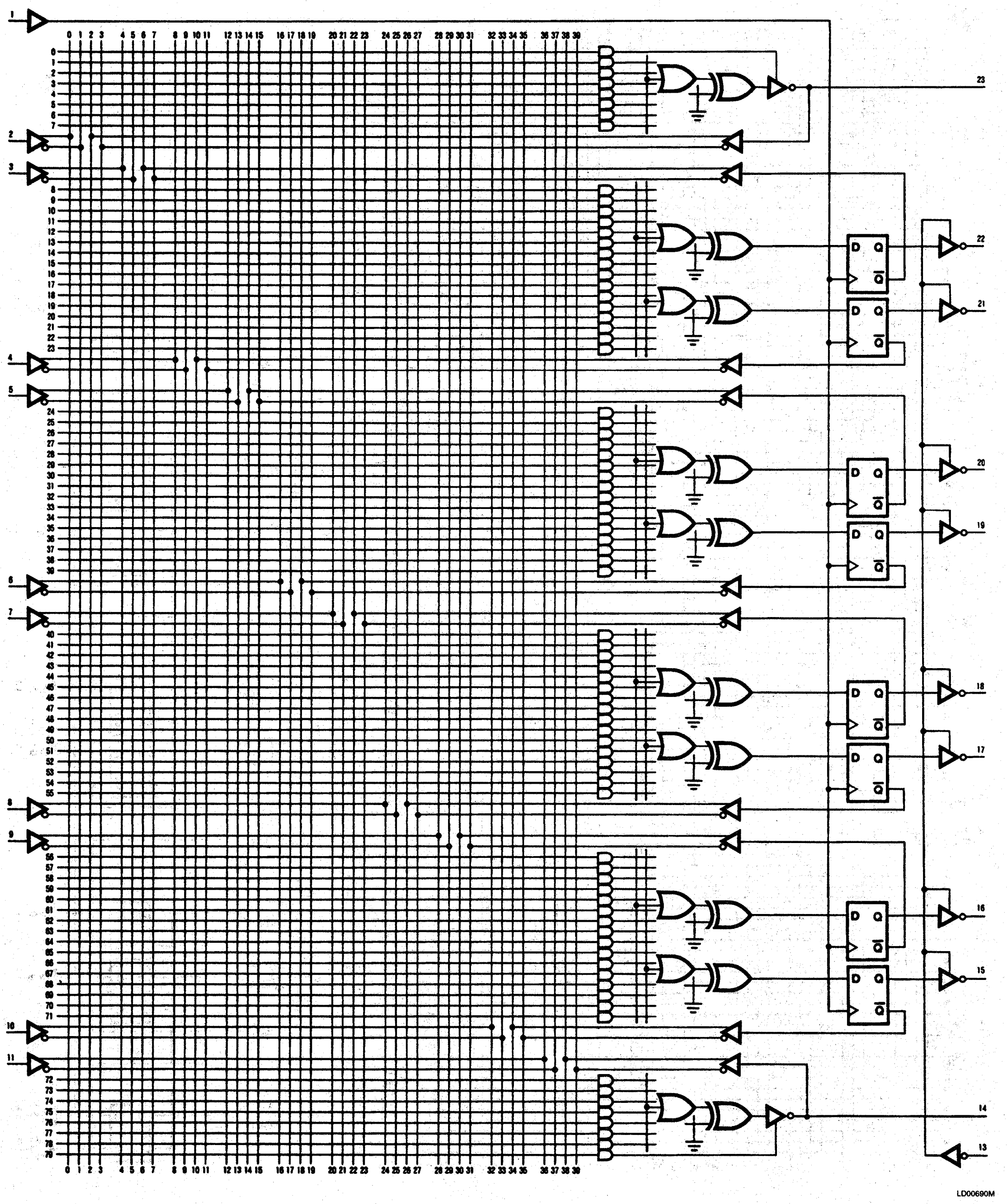
LD00670M

20RS10

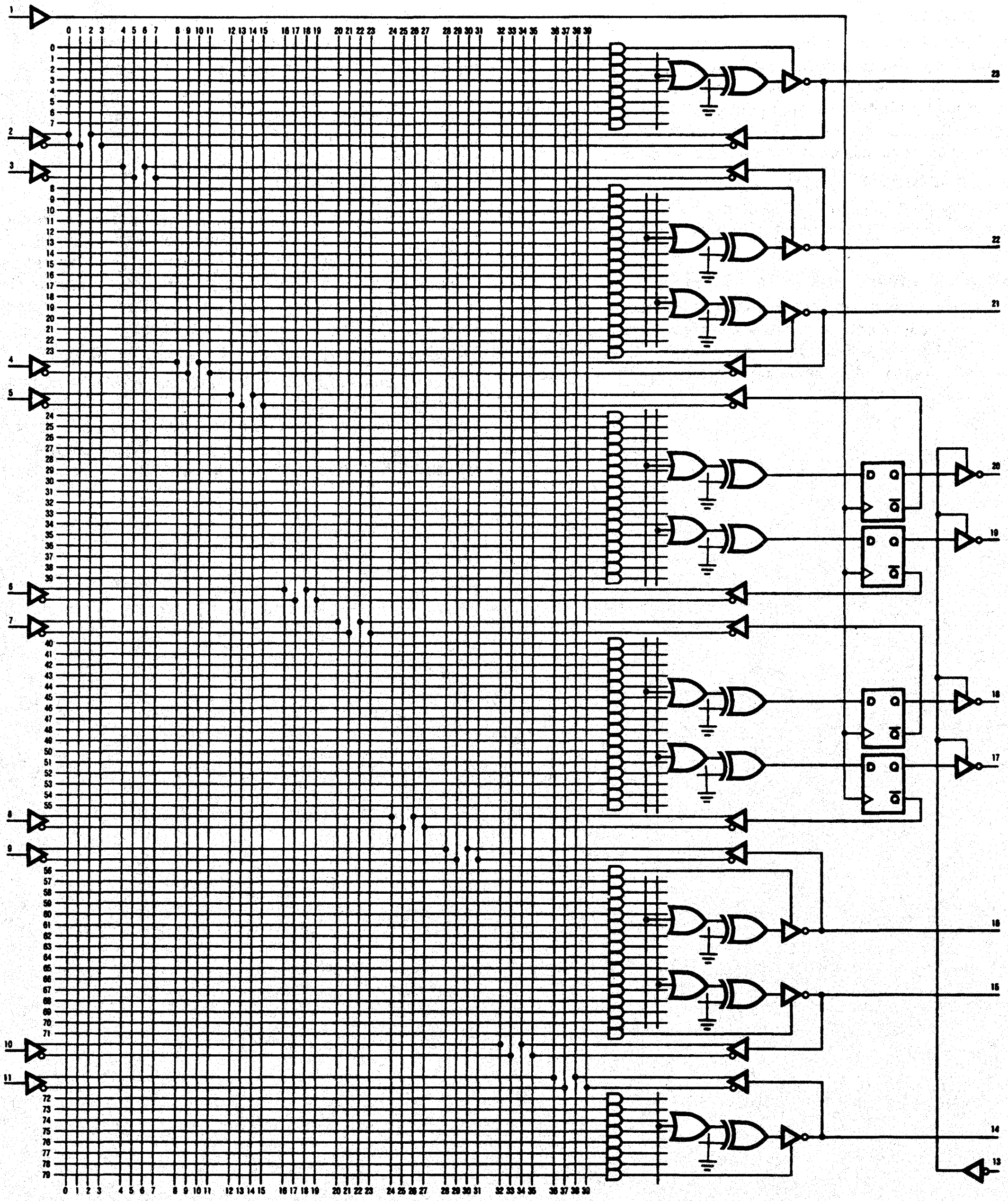


LD00680M

20RS8


1
2
3
4
5
6
7
8
9
10
11
13
14
15
16
17
18
19
20
21
22
23
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39
0 1 2 3 4 5 6 7
8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39
40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55
56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71
72 73 74 75 76 77 78 79
D
Q
Q̄
LD00690M

20RS4



LD00700M

Large 24RA (PAL20RA10)

Description

The PAL20RA10 is a 24-pin registered asynchronous PAL device. This versatile device features programmable clock, enable, set, and reset, all of which can operate asynchronously to other flip-flops in the same device. It also has individual flop-flop bypass, allowing this one device to provide any combination of registered and combinatorial outputs.

Programmable Clock

The clock input to each flip-flop comes from the programmable array, allowing the flip-flops to be clocked independently if desired.

Programmable Set and Reset

Each flip-flop has a product line for asynchronous set and one product for asynchronous reset. If the chosen product line is high, the flip-flop will set (become a logic HIGH), or reset (become a logic LOW). The sense of the output pin is inverted if the output is active low.

Programmable Polarity

Each flip-flop has individually programmable polarity. The unprogrammed state is active low.

Programmable Flip-flop Bypass

If both the set and reset product lines are high, the flip-flop is bypassed and the output becomes combinatorial. Thus each output can be configured to be registered or combinatorial.

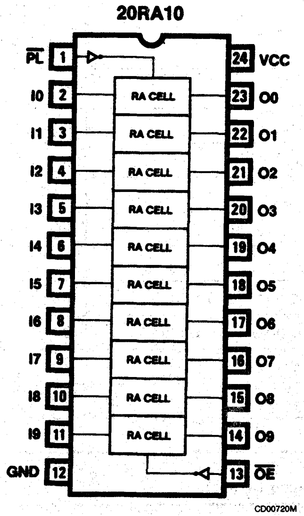
Programmable and Hard-Wired Three-State Outputs

The PAL20RA10 provides a product term dedicated to output control. There is also an output control pin (pin 13). The output is enabled if both the output control pin is low and the output control product term is high. If the output control pin is high all outputs will be disabled. If an output control product term is low, then that output will be disabled.

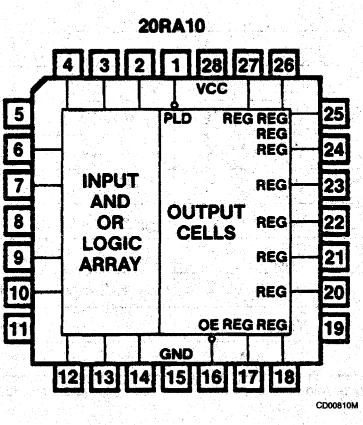
Register Preload and Power-up Reset

Each device also offers register preload for device testability. The registers can be preloaded from the outputs by using TTL level signals in order to simplify functional testing. This series also offers Power-up Reset, whereby the registers power up to a logic LOW, setting the active-low outputs to a logic HIGH.

DIP Pinout



PLCC Pinout



Operating Conditions

SYMBOL	PARAMETER		MILITARY			COMMERCIAL			UNIT
			MIN	TYP	MAX	MIN	TYP	MAX	
V_{CC}	Supply voltage		4.5	5	5.5	4.75	5	5.25	V
t_w	Width of clock	Low	25	13		20	13		ns
		High	25	13		20	13		
t_{wp}	Preload pulse width		45	15		35	15		ns
t_{su}	Setup time for input or feedback to clock		25	10		20	10		ns
t_{sup}	Preload setup time		30	5		25	5		ns
t_h	Hold time	Polarity fuse intact	10	-2		10	-2		ns
		Polarity fuse blown	0	-6		0	-6		
t_{hp}	Preload hold time		30	5		25	5		ns
T_A	Operating free-air temperature		-55			0		75	°C
T_C	Operating case temperature				125				°C

Electrical Characteristics (Over Operating Conditions)

SYMBOL	PARAMETER	TEST CONDITION		MIN	TYP	MAX	UNIT
V_{IL}[1]	Low-level input voltage					0.8	V
V_{IH}[1]	High-level input voltage			2			V
V_{IC}	Input clamp voltage	V_{CC} = MIN	I_I = -18mA		-0.8	-1.5	V
I_{IL}[2]	Low-level input current	V_{CC} = MAX	V_I = 0.4V		-0.02	-0.25	mA
I_{IH}[2]	High-level input current	V_{CC} = MAX	V_I = 2.4V			25	μA
I_I	Maximum input current	V_{CC} = MAX	V_I = 5.5V			1	mA
V_{OL}	Low-level output voltage	V_{CC} = MIN	I_{OL} = 8mA		0.3	0.5	V
V_{OH}	High-level output voltage	V_{CC} = MIN	I_{OH}: Mil-2mA Com-3.2mA	2.4	2.8		V
I_{OZ}[2]	Off-state output current	V_{CC} = MAX	V_O = 2.4 V/V_O = 0.4V	-100		100	μA
I_{OS}[3]	Output short-circuit current	V_{CC} = 5V	V_O = 0V	-30	-70	-130	mA
I_{CC}	Supply current	V_{CC} = MAX			155	200	mA

[1] These are absolute voltages with respect to the ground pin on the device and include all overshoots due to system and/or tester noise. Do not attempt to test these values without suitable equipment.

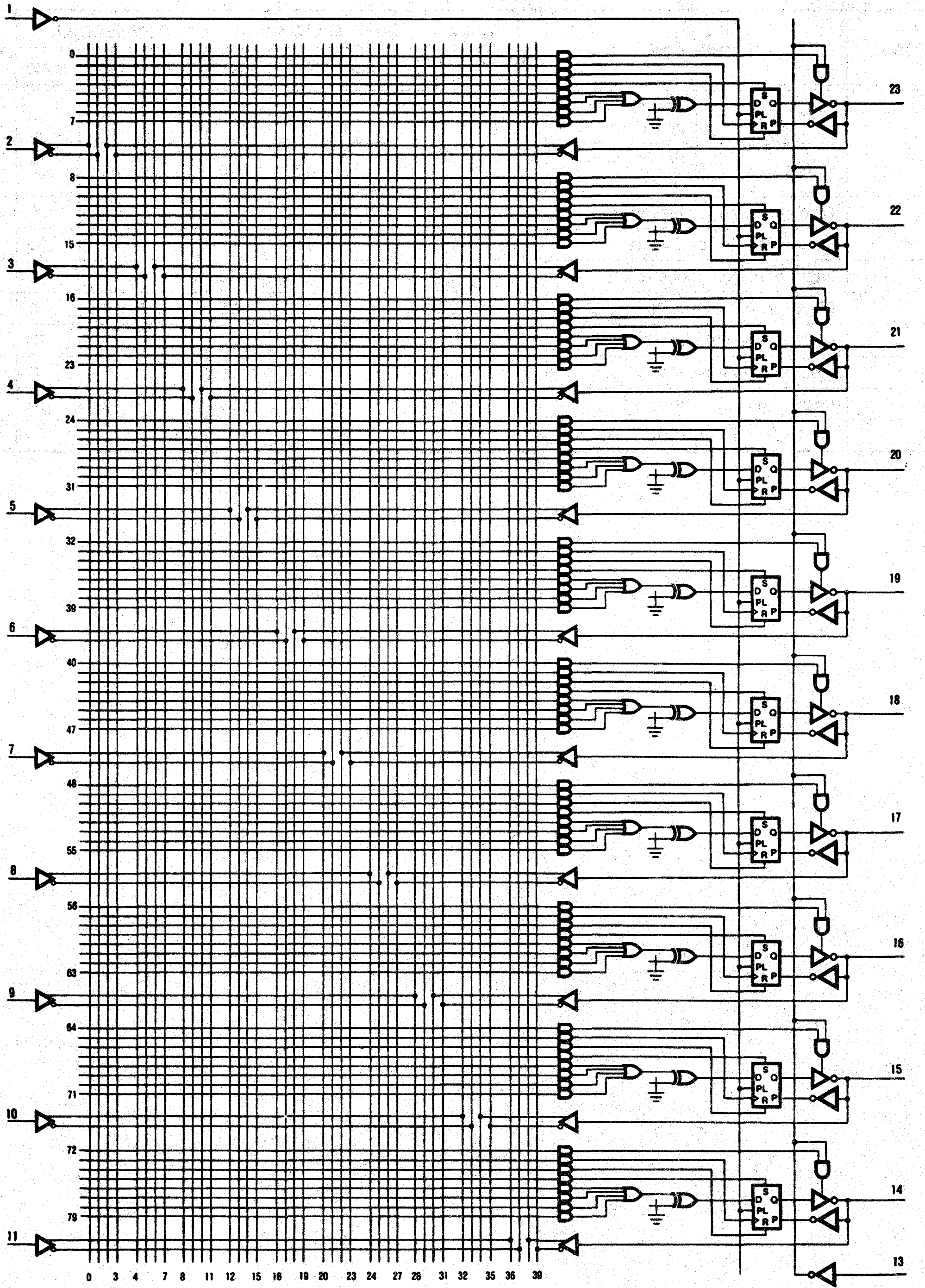
[2] I/O pin leakage is the worst case of I_{IL} and I_{OZL} (or I_{IH} and I_{OZH}).

[3] No more than one output should be shorted at a time and duration of the short circuit should not exceed one second.

Switching Characteristics (Over Operating Conditions)

SYMBOL	PARAMETER		TEST CONDITIONS	MILITARY			COMMERCIAL			UNIT
				MIN	TYP	MAX	MIN	TYP	MAX	
t_{PD}	Input or feedback to output	Polarity fuse intact	$R_1 = 560\Omega$ $R_2 = 1.1K\Omega$		20	35		20	30	ns
		Polarity fuse blown			25	40		25	35	
t_{CLK}	Clock to output or feedback			10	17	35	10	17	30	ns
t_S	Input to asynchronous set				22	40		22	35	ns
t_R	Input to asynchronous reset				27	45		27	40	ns
t_{PZX}	Pin 13 to output enable				10	25		10	20	ns
t_{PXZ}	Pin 13 to output disable				10	25		10	20	ns
t_{PZX}	Input to output enable				18	35		18	30	ns
t_{PXZ}	Input to output disable				15	35		15	30	ns
f_{MAX}	Maximum frequency			16	35		20	35		MHz

20RA10

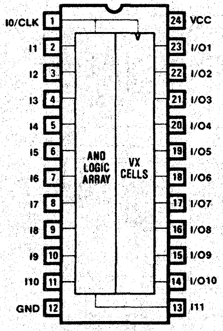


LD00710M

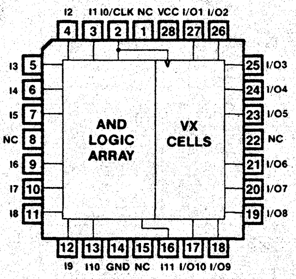
Features/Benefits

- Dual independent feedback paths allow buried state registers or input registers
- Programmable flip-flops allow J-K, S-R, T or D types for the most efficient use of product terms
- 10 input/output macrocells for flexibility
- Programmable registered or combinatorial outputs
- Programmable output polarity
- Global register asynchronous preset/synchronous reset or synchronous preset/asynchronous reset
- Automatic register preset on power up
- Preloadable output registers for testability
- Varied product term distribution
  –Up to 16 product terms per output
- High speed
  –25ns "A" version
  –30ns standard version
- Space-saving 24-pin 300-mil SKINNYDIP® package or 28-pin chip carrier
- Pin-compatible functional superset of 22V10

Pin Configurations



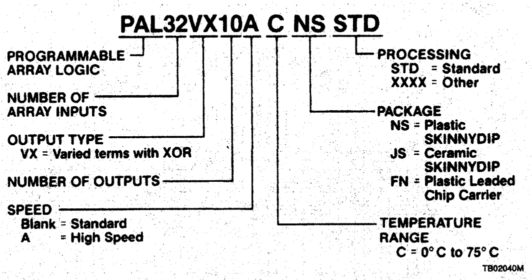
SKINNYDIP Package



Plastic Leaded Chip Carrier

CD01660M

Ordering Information



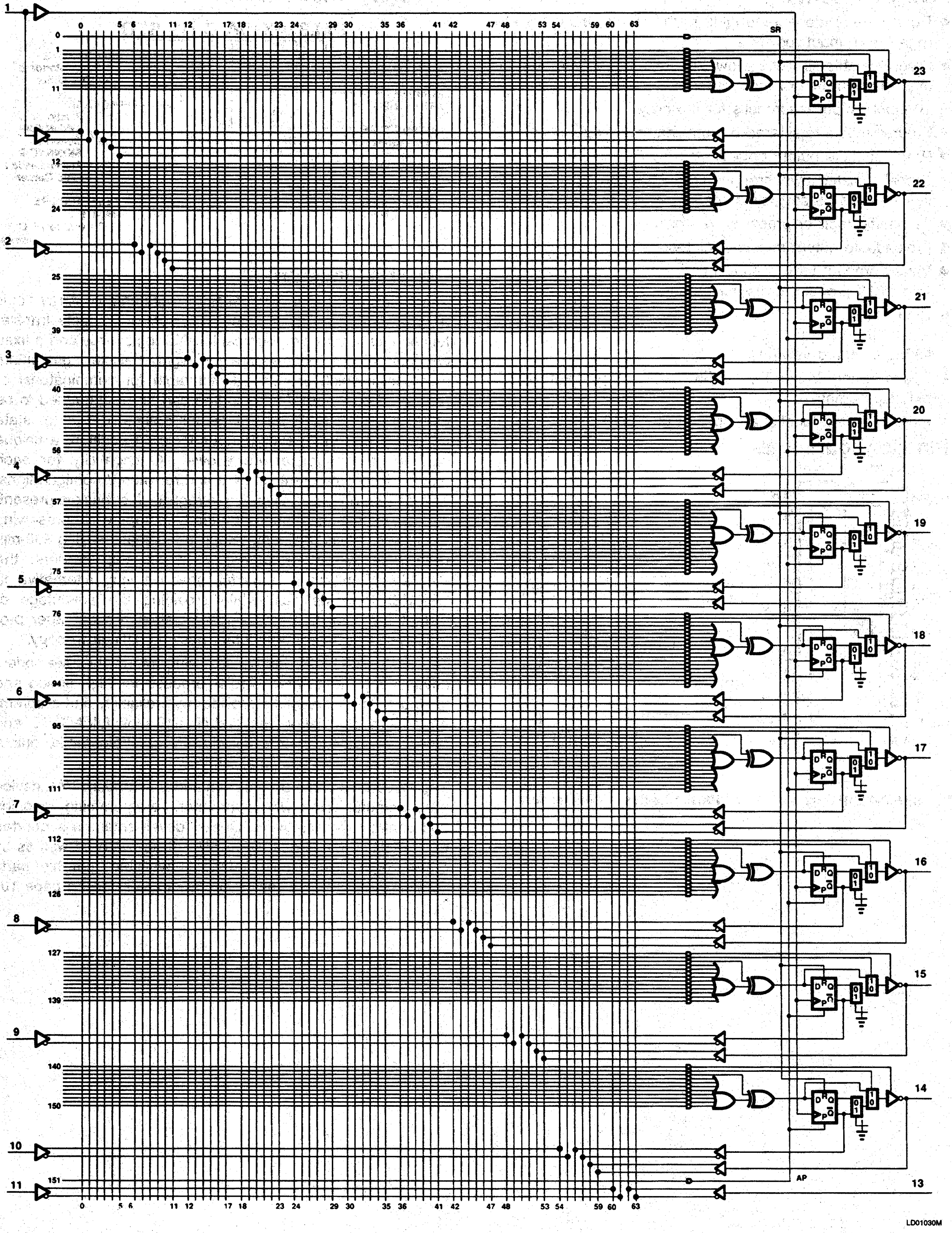
General Description

The PAL32VX10 is a high-density Programmable Array Logic (PAL®) device which implements a sum-of-products transfer function via a user-programmable AND logic array and a fixed OR logic array. Featured are ten highly flexible input/output macrocells which are user-configurable for combinatorial or registered operation. Each flip-flop can be programmed to be either a J-K, S-R, T, or D-type for optimal design of state machines and other synchronous logic. In addition, a unique dual feedback architecture allows I/O capability for each macrocell in both combinatorial and registered configurations. This can be achieved even when register feedback is present, and allows implementation of buried flip-flops while preserving the external macrocell input. Supplied in space-saving 300-mil-wide dual in-line packages or 28-pin chip carriers, the PAL32VX10 offers a powerful, space saving alternative to SSI/MSI logic devices, while providing the advantage of instant prototyping. Security fuses defeat readout after programming and make proprietary designs difficult to copy.

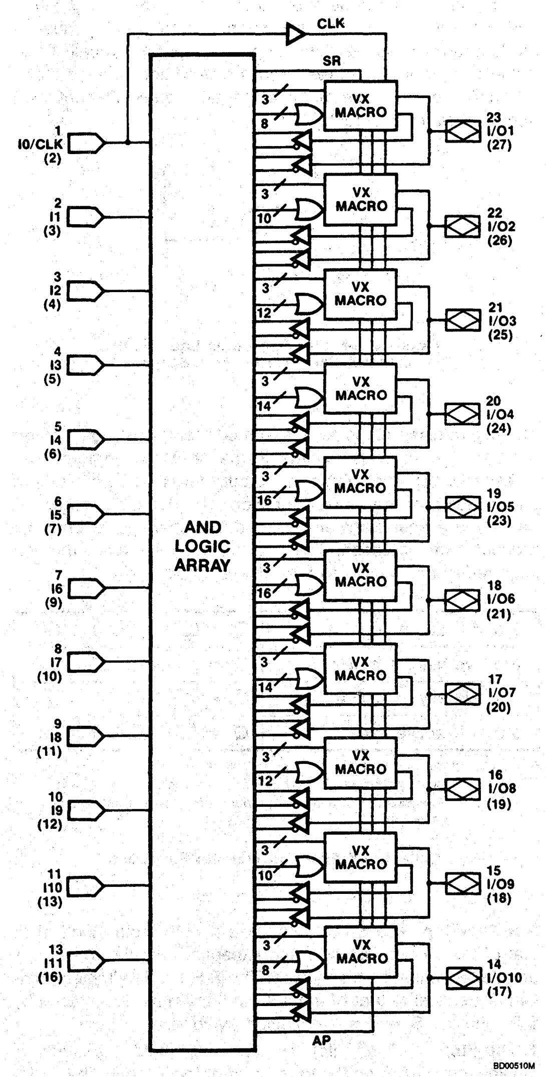
The PAL32VX10 is fabricated using Monolithic Memories' advanced oxide-isolated bipolar process for high speed and low power. TiW fuse links provide high reliability and programming yields. Special on-chip test circuits allow full AC, DC, and functional testing before programming. Preloadable output registers facilitate functional testing.

The PAL32VX10 can be programmed on standard PAL device programmers, fitted with appropriate programming modules and configuration software. Design development is supported by Monolithic Memories' PALASM® 2 software as well as by other programmable logic CAD tools available from third party vendors. Approved development tools are listed on page 10.

Logic Diagram



Block Diagram



Note: PLCC pin numbers are indicated in parentheses.
PLCC pins 1, 8, 15, and 22 are not connected.

Description of Architecture

The PAL32VX10 has twelve dedicated input lines and ten programmable I/O macrocells. Pin 1 serves either as an array input or as a clock for all flip-flops. Buffers for device inputs have complementary outputs to provide user-programmable input signal polarity. The fuse matrix implements a programmable AND logic array, which drives a fixed OR logic array.

The high level of flexibility built into each macrocell, shown in Figure 1, allows the PAL32VX10 to implement over thirty different architecture options. Each macrocell can be individually programmed to implement a variety of combinatorial or registered logic functions.

Dual Output Feedback

Dual feedback paths associated with each macrocell provide independent feedback paths directly into the array from both the flip-flop output and the output pin. Unlike other devices which have a single feedback path, the PAL32VX10 allows each output to have full I/O capability when configured as either a combinatorial output or a registered output, even if register feedback to the array is used. Thus registers can be loaded from their outputs.

If a macrocell is configured as a dedicated input, by disabling the three-state output buffer, the dual feedback architecture allows use of the associated register as an input register or as a "buried" state register, avoiding waste of the flip-flop, as shown in Figure 2.

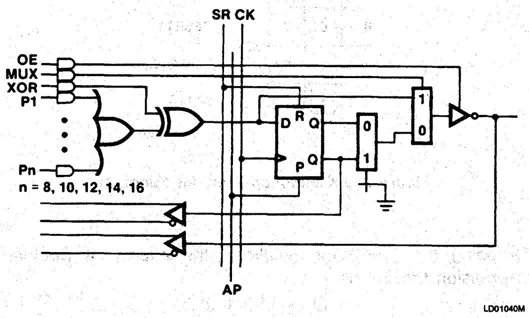


Figure 1. PAL32VX10 Macrocell

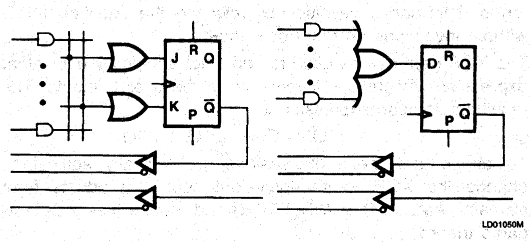


Figure 2. Buried Flip-Flops with Dedicated Inputs

Programmable Flip-Flops

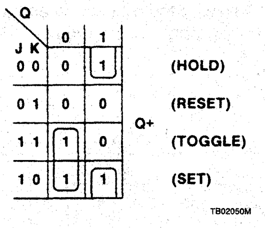
Each output macrocell contains a unique programmable flip-flop consisting of a basic D-type flip-flop driven by an XOR gate. This allows the user to choose the optimal flip-flop for the design, since either J-K, S-R, or T-type flip-flops can be synthesized from such a structure without wasting product terms.

As indicated in the macrocell logic diagram, one input of the XOR gate is connected to a single product term, while the second input is connected to the output of the OR logic array. The XOR gate output feeds the input of the D flip-flop. The way in which the XOR gate is used to synthesize the different flip-flop types is described in detail below.

D Flip-Flop. The D flip-flop option is implemented directly. In this configuration, the XOR gate on the input of the flip-flop can be used to program the logic polarity of the transfer function.

J-K Flip-Flop. The J-K flip-flop option can be easily synthesized with a more sophisticated manipulation of the XOR gate inputs and the D flip-flop output.

The transfer function of a J-K flip-flop can be mapped in the Karnaugh Map of Figure 3, where Q + represents the next state of the flip-flop:



J K \ Q	0	1	
0 0	0	1	(HOLD)
0 1	0	0	(RESET)
1 1	1	0	(TOGGLE)
1 0	1	1	(SET)

Figure 3. J-K Flip-Flop Transfer Function

Dropping the (+) for simplicity, the equivalent Boolean expression for Q + is:

$$Q:=\overline{K}*Q+J*\overline{Q}$$

In general, J and K can be sum-of-product expressions which are provided in the PAL architecture only in active-high form. Thus, a direct implementation of $\overline{K}$ expressions must invoke a DeMorgan transformation, which can use excessive product terms. This can be avoided by rewriting the equation for Q without inversions on the J or K inputs.

The XOR gate can be used to construct a logically equivalent expression without any inversions on the J or K inputs. The rewritten Boolean expression is:

$$Q:=Q:+:(J*\overline{Q}+K*Q)$$

To check that these expressions are logically equivalent, change the XOR to its equivalent sum of products form (remember A: +: B = $A*\overline{B}+\overline{A}*B$) and reduce (using DeMorgan's theorem):

$$Q:=Q*\overline{(J*\overline{Q}+K*Q)}\qquad+\overline{Q}*(J*\overline{Q}+K*Q)$$
$$Q:=Q*((\overline{J}+Q)*(\overline{K}+\overline{Q}))\qquad+\overline{Q}*J*\overline{Q}+\overline{Q}*K*Q$$
$$Q:=Q*(\overline{J}*\overline{K}+\overline{J}*\overline{Q}+Q*\overline{K}+Q*\overline{Q})+J*\overline{Q}$$
$$Q:=\overline{J}*\overline{K}*Q+\overline{K}*Q+J*\overline{Q}$$

which simplifies to $Q:=\overline{K}*Q+J*\overline{Q}$.

Since J and K are, in general, sums of products, J and K in either expression can be substituted with (J1 + J2 + ... + Jm) and (K1 + K2 + ... + Kn − m), where n is the total number of product terms associated with a given output macrocell. Thus, the total n-product term resource is shared between the J and K control inputs (Figure 4). Note that all J terms will contain $\overline{Q}$ and all K terms will contain Q.

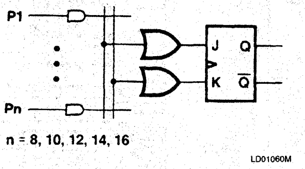


Figure 4. J-K Flip-Flop Logic Equivalent; J and K Can Also be Active-Low

The above discussions have assumed that it was most convenient to "group ones" in the Karnaugh Map. Sometimes it takes fewer product terms to "group zeros", i.e., implement the inversion of the desired function. The equations shown in Table 1 are equivalent and can be interchanged to optimize product term utilization. This can be readily proved through logic reductions similar to that above.

J and K active high	$Q:=Q:+:(J*\overline{Q}+J*Q)$
J active high, K active low	$Q:=J*\overline{Q}+\overline{K}*Q$
J active low, K active high	$\overline{Q}:=\overline{J}*\overline{Q}+K*Q$
J and K active low	$Q:=\overline{Q}:+:(\overline{J}*\overline{Q}+\overline{K}*Q)$

Note: J = sum of products J1 + J2 + ... + Jm
K = sum of products K1 + K2 + ... + Kn − m
n = total number of available product terms for a given macrocell (8 to 16)

Table 1. J-K Flip-Flop Transfer Functions

S-R Flip-Flop. The S-R flip-flop has a truth table identical to that of the J-K flip-flop, with the exception that the J = K = 1 (toggle) condition is not allowed. The S-R flip-flop implementation is identical to that of the J-K flip-flop, with J-K replaced by S-R, and the S = R = 1 condition avoided.

T Flip-Flop. A T (toggle) flip-flop either holds its state or toggles, depending on the logic state of the T input. The T flip-flop is a subset of the J-K flip-flop and can be considered equivalent to a J-K type with J = K. The general transfer function and its active-low T equivalent are both given in Table 2.

$Q:=Q:+:T$
$Q:=\overline{Q}:+:\overline{T}$

Note: T = sum of products T1 + T2 + T3 + ... + Tn

Table 2. T Flip-Flop Transfer Functions

Summary

The PAL32VX10 can synthesize J-K, S-R, T, and D flip-flops, whichever is most convenient for the application, without sacrificing product terms. Additionally, the synthesized equations can use the active-high or active-low forms of the inputs, allowing the designer to minimize product term requirements.

Flip-Flop Bypass

Any output in the PAL32VX10 can be configured to be combinatorial by bypassing the output flip-flop. This is done by setting the output multiplexer to the appropriate state. The multiplexer is controlled by a product term which can be set unconditionally for a permanent combinatorial (all fuses opened, product term high) or registered (all fuses intact, product term low) output configuration, or can be programmed to bypass the output flip-flop "on the fly," allowing signals to be routed directly to output pins under user-specified conditions.

Varied Product Term Distribution

An increased number of product terms has been provided in the PAL32VX10 over previous generation PAL devices. These terms are distributed among the ten macrocells in a varied manner, ranging from eight to sixteen terms per output. The five output pairs have 8, 10, 12, 14, or 16 product terms available for the OR gate within each macrocell. In addition, each macrocell has one XOR product term and two architecture control product terms.

Programmable I/O

Each macrocell has a three-state output buffer with programmable three-state control. Control is implemented by a single product term, allowing specification of enable/disable functions controlled by any device input or output. Each macrocell can be configured as a dedicated input by disabling the buffer drive capability. When this is done, the associated register can still be used as an input register or buried state register, due to the independent register feedback path.

Programmable Preset and Reset

The ten macrocell flip-flops share common programmable preset and reset control for easy system initialization. The Q outputs of the register will go to the logic low state following a low-to-high transition on pin 1 (I0/CLK) when the synchronous reset (SR) product term is asserted. The register will be forced to the logic high state independent of the clock when the asynchronous preset (AP) product term is asserted.

Programmable Polarity

The polarity of each macrocell output can be set active high or active low.

Combinatorial Outputs. The XOR gate provides polarity control for combinatorial outputs, with the single product term to the XOR gate controlling the invert/not invert function. With all fuses intact, there is no inversion through the XOR gate, creating an active low output. Opening all fuses forces the product term high, inverting data and creating an active high output.

Registered Outputs. Output polarity for registered outputs can be determined in two ways. For D-type registered outputs, polarity can be set by the XOR gate, as is the case with combinatorial outputs. Using this method to set polarity, preset and reset will not be affected.

Polarity, as observed from the output pin, can also be determined by the flip-flop output multiplexer. Note that this does not affect the polarity of the register feedback signal, but does affect preset and reset. By changing the flip-flop output multiplexer, the preset and reset functions are exchanged, relative to the controlling product terms.

With the multiplexer fuse intact, the Q output is routed to the output pin, configuring an active low output. With the multiplexer fuse opened, $\overline{Q}$ is routed to the output pin, and synchronous reset becomes synchronous preset. Similarly, asynchronous reset becomes asynchronous preset.

Polarity options for J-K, S-R, and T flip-flops have been discussed in the section on programmable flip-flops.

Power-Up Preset

All flip-flops power up to a logic high for predictable system initialization. Outputs of the PAL32VX10 will be high or low depending on the state of the register output multiplexers.

Register Preload

The register on the PAL32VX10 can be preloaded to facilitate functional testing of complex state machine designs. This feature allows direct loading of arbitrary states, thereby making it unnecessary to cycle through long test vector sequences to reach a desired state. In addition, transitions from illegal states can be verified by loading in illegal states and observing proper recovery.

Security Fuse

After programming and verification, a PAL32VX10 design can be secured by programming the security fuses. Once programmed, these fuses defeat readback of the internal fuse pattern by a device programmer, making proprietary designs very difficult to copy.

Quality and Testability

The PAL32VX10 offers a very high level of built-in quality. Special on-chip test circuitry provides a means of verifying performance of all AC and DC parameters prior to programming. In addition, these built-in test paths verify complete functionality of each device to provide the highest post-programming functional yields in the industry.

Absolute Maximum Ratings

	Operating	Programming
Supply voltage V_{CC}	−0.5V to 7V	−0.5V to 12V
Input voltage	−1.5V to 5.5V	−1.0V to 22V
Off-state output voltage	5.5V	12V
Storage temperature		−65°C to +150°C

Operating Conditions

SYMBOL	PARAMETER		COMMERCIAL[1]						UNIT
			STD			A			
			MIN	TYP	MAX	MIN	TYP	MAX	
V_{CC}	Supply voltage		4.75	5	5.25	4.75	5	5.25	V
t_w	Width of clock	Low	20	10		18	10		ns
		High	20	10		18	10		
t_{su}	Setup time from input or feedback to clock	Product terms P_1-P_n, SR	30	20		25	20		ns
		Product term XOR	35	25		30	25		
t_h	Hold time		0	−10		0	−10		ns
t_{aw}	Asynchronous preset width		−30	20		25	20		ns
t_{ar}	Asynchronous preset recovery time		−30	20		25	20		ns
t_{sr}	Asynchronous reset recovery time		−30	20		25	20		ns
T_A	Operating case temperature		0	25	75	0	25	75	°C

Electrical Characteristics Over Operating Conditions

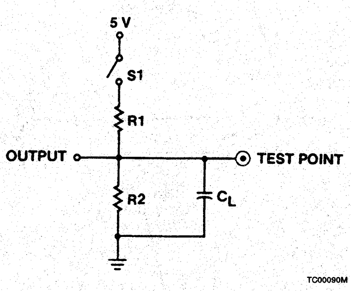
SYMBOL	PARAMETER	TEST CONDITION		MIN	TYP	MAX	UNIT
V_{IL}[2]	Low-level input voltage					0.8	V
V_{IH}[2]	High-level input voltage			2			V
V_{IC}	Input clamp voltage	V_{CC} = MIN	I_I = −18mA		−0.8	−1.5	V
I_{IL}[3]	Low-level input current	V_{CC} = MAX	V_I = 0.4V		−0.02	−0.25	mA
I_{IH}[3]	High-level input current	V_{CC} = MAX	V_I = 2.4V			25	μA
I_I	Maximum input current	V_{CC} = MAX	V_I = 5.5V			200	μA
V_{OL}	Low-level output voltage	V_{CC} = MIN	I_{OL} = 16mA		0.35	0.5	V
V_{OH}	High-level output voltage	V_{CC} = MIN	I_{OH} = −3.2mA	2.4	3.4		V
I_{OZL}[3]	Off-state output current	V_{CC} = MAX	V_O = 0.4V			−100	μA
I_{OZH}[3]			V_O = 2.4V			100	μA
I_{OS}[4]	Output short-circuit current	V_{CC} = 5V	V_O = 0V	−30	−70	−130	mA
I_{CC}	Supply current	V_{CC} = MAX			140	180	mA
C_{IN}	Input capacitance	V_{IN} = 2.0V at f = 1MHz			6		pF
C_{OUT}	Output capacitance	V_{OUT} = 2.0V at f = 1MHz			11		

1. The PAL32VX10/A is designed to operate over the full military operating conditions. For availability and specifications contact Monolithic Memories.
2. These are absolute voltages with respect to the ground pin on the device and include all overshoots due to system and or tester noise. Do not attempt to test these values without suitable equipment.
3. I/O pin leakage is the worst case of I_{IL} and I_{OZL} (or I_{IH} and I_{OZH}).
4. No more than one output should be shorted at a time and duration of the short circuit should not exceed one second.

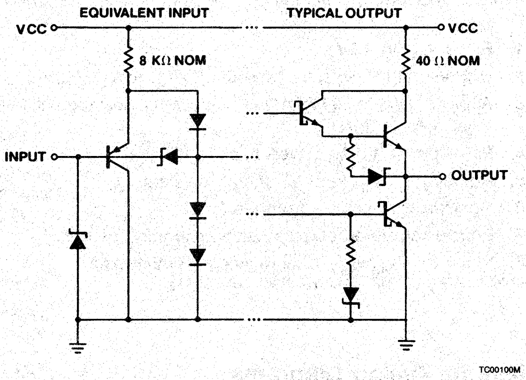
Switching Characteristics Over Operating Conditions

SYMBOL	PARAMETER			TEST CONDITION	STD			A			UNIT
					MIN	TYP	MAX	MIN	TYP	MAX	
t_{PD}	Input or feedback to output		Product terms P_1-P_n	$R_1 = 200\Omega$ $R_2 = 390\Omega$		15	30		15	25	ns
			Product term XOR			25	35		20	30	
t_{CLK}	Clock to output or feedback					10	15		10	15	ns
t_{PZX}	Input to output enable					20	30		20	25	ns
t_{PXZ}	Input to output disable					20	30		20	25	ns
t_{AP}	Asynchronous preset to output					20	30		20	25	ns
t_{CR}	Input or feedback to registered output from combinatorial configuration					75	90		75	90	ns
t_{RC}	Input or feedback to combinatorial output from registered configuration					75	90		75	90	ns
f_{MAX}	Maximum frequency	Feedback ($1/t_{P1}$)	Product terms P_1-P_0		22.5	35		25	35		MHz
			Product term XOR		20	30		22.2	30		
		No feedback ($1/t_{P2}$)			25	40		27.7	40		

Switching Test Load

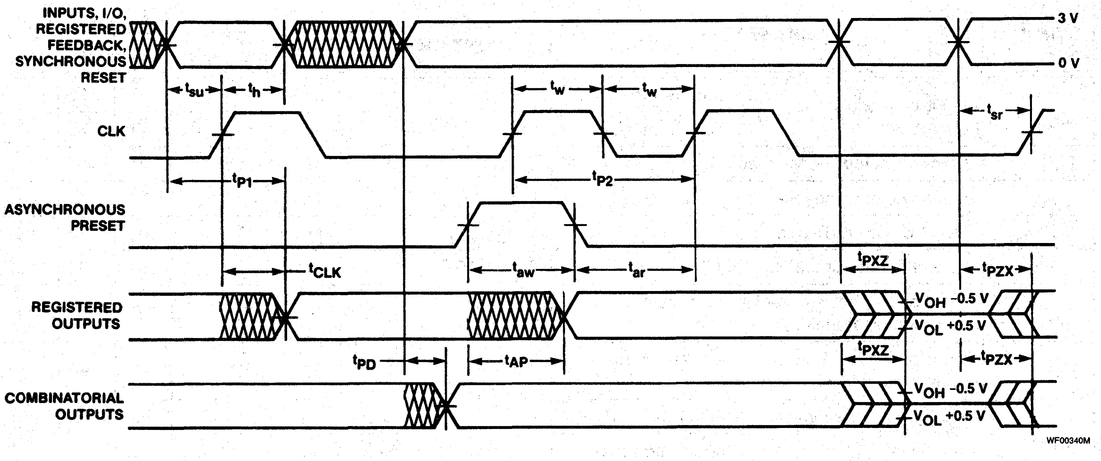


Schematic of Inputs and Outputs



Notes:
1. t_{PD} is tested with switch S_1 closed. C_L = 50pF and measured at 1.5V output level.
2. t_{PZX} is measured at the 1.5V output level with C_L = 50pF. S_1 is open for high impedance to "1" test, and closed for high impedance to "0" test.
3. t_{PXZ} is tested with C_L = 5pF. S_1 is open for "1" to high impedance test, measured at V_{OH} −0.5V output level; S_1 is closed for "0" to high impedance test measured at V_{OL} +0.5V output level.

Switching Waveforms

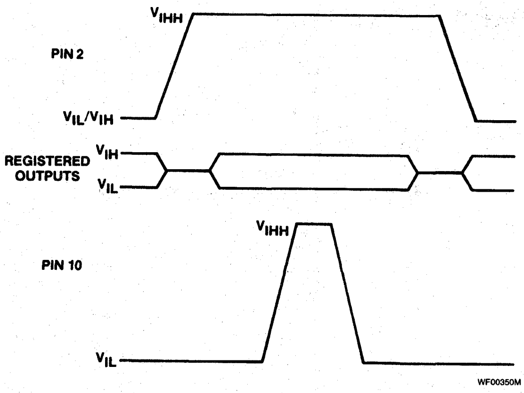


Output Register Preload

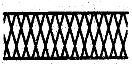
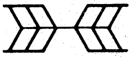
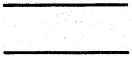
The preload function allows the register to be loaded from the output pins. This feature aids functional testing of sequential designs by allowing direct setting of output states. The procedure is:

1. Raise V_{CC} to 4.5V.
2. Disable output registers by setting pin 2 to V_{IHH}(12V).
3. Apply V_{IL}/V_{IH} to all registered output pins. Leave combinatorial outputs floating.
4. Pulse pin 10 to V_{IHH}, then back to 0V.
5. Remove V_{IL}/V_{IH} from all output registers.
6. Remove high voltage from pin 2.
7. Enable registered outputs per programmed pattern.
8. Verify for V_{OL}/V_{OH} at all registered output pins.

Note: V_{IHH} = 11.0 (MIN), 11.5 (TYP) and 12.0 (MAX).



Key to Timing Diagrams

WAVEFORM	INPUTS	OUTPUTS
	DON'T CARE; CHANGE PERMITTED	CHANGING; STATE UNKNOWN
	NOT APPLICABLE	CENTER LINE IS HIGH IMPEDANCE STATE
	MUST BE STEADY	WILL BE STEADY

WF00360M

f_{MAX} Parameters

The parameter fMAX is the maximum speed at which the PAL device is guaranteed to operate. Because flexibility inherent to PAL devices allows a choice of clocked flip-flop designs, for the convenience of the user, fMAX is specified to address two major classes of synchronous designs.

The simplest type of synchronous design can be described as a data path application. In this case, data is presented to the data terminal of the flip-flop and clocked through; no feedback is employed (Figure 1). Under these conditions, the frequency of operation is limited by the greater of the data setup time (tsu) or the minimum clock period (tw high + tw low, or tP2). This parameter is designated fMAX (no feedback).

For synchronous sequential designs, i.e., state machines, where logical feedback is required, inputs to flip-flop data terminals originate from the device input pins or flip-flop outputs via the internal feedback paths (Figure 2). Under these conditions, fMAX is defined as the reciprocal of (tsu + tCLK), or tP1, and is designated fMAX (feedback).

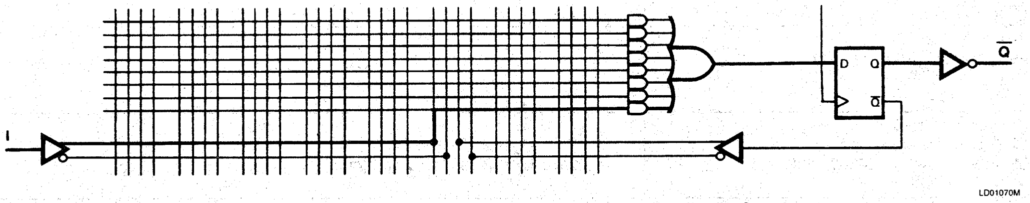


Figure 1. Data Path Register Configuration Without Feedback, Q: = I

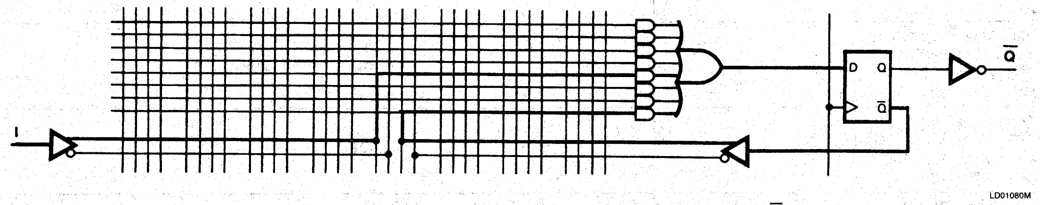


Figure 2. State Machine Configuration With Feedback, Q: = I + Q̄

Use of XOR Product Term

The speed of the PAL32VX10 is specified according to the use of the Exclusive-OR (XOR) product term in the macrocell. Note that the macrocell data input is a function of the two-input XOR gate, whose inputs are the OR of the product terms P1-Pn and the single additional XOR product term (Figure 3).

The specification for the path through the single XOR product term is 5ns slower than through the P1-Pn product terms and the OR gate. As a result, if the single XOR product term is changing, the macrocell data input will not be available until 5ns later than if only the P1-Pn product terms were changing.

This difference between paths affects tPD, tsu, and fMAX (feedback). As a result, these three parameters are specified both for only the P1-Pn product terms changing ("Product terms P1-Pn") and with the single XOR product term changing ("Product term XOR") (Figure 4).

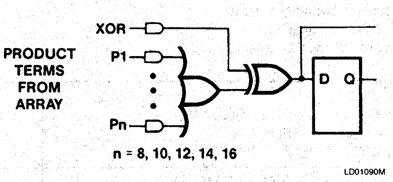


Figure 3.

SPECIFICATION		EXPLANATION
t_{PD}, t_{su}, f_{MAX} (feedback)	Product terms P1-Pn	If only the P1-Pn product terms are changing (XOR term is not changing)
	Product term XOR	If XOR term is changing

Figure 4.

MANUFACTURER	PROGRAMMER CONFIGURATION
Data I/O Corp. 10525 Willows Rd. NE, PO Box 97046 Redmond, WA 98073-9746 (800) 247-5700	System 29A, 29B LogicPak™ 303A Adapter 303A-011A/B-V01 Family/Pinout Code: 22-77
Digelec Inc. 1602 Lawrence Ave., Suite 113 Ocean, NJ 07712 (201) 493-2420	Contact manufacturer
Digital Media 11770 East Warner Ave., Suite 225 Fountain Valley, CA 92708 (714) 751-1373	Contact manufacturer
Japan Macnics Corp. 2999 Monterey/Salinas Hwy. Monterey, CA 93940 (408) 373-3607	Contact manufacturer
Kontron Electronics Inc. 1230 Charleston Rd. Mountain View, CA 94039-7230 (415) 965-7020	System EPP-80 Module UPM-B rev. 1.48 or later
Micropross Parc d'activite des Pres 5, rue Denis-Papin 59650 Villeneuve-d'Ascq (20) 47.90.40	Contact manufacturer
Stag Microsystems Inc. 528-5 Weddell Dr. Sunnyvale, CA 94089 (408) 745-1991	Contact manufacturer
Storey Systems 3201 N. Hwy 67, Suite H Mesquite, TX 75150 (214) 270-4135	Contact manufacturer
Structured Design 988 Bryant Way Sunnyvale, CA 94087 (408) 988-0725	Contact manufacturer
Valley Data Sciences Charleston Business Park 2426 Charleston Rd. Mountain View, CA 94043 (415) 968-2900	Contact manufacturer
Varix Corp. 1210 E. Campbell Rd., Suite 100 Richardson, TX 75081 (214) 437-0777	Contact manufacturer
MANUFACTURER	**SOFTWARE DEVELOPMENT SYSTEM**
Monolithic Memories, Inc. IdeaLogic® Exchange M/S 09-25 2175 Mission College Blvd. Santa Clara, CA 95054-1592 (800) 247-6527 ext. 6105	PALASM® 2 Software rev. 2.21 and later
Data I/O Corp. 10525 Willows Rd. NE, PO Box 97046 Redmond, WA 90873-9746 (800) 247-5700	ABEL™ Software — Contact manufacturer
Personal CAD Systems Assisted Technology Division 1290 Parkmoor Ave. San Jose, CA 95126 (408) 971-1300	CUPL™ Software — Contact manufacturer

IdeaLogic is a trademark of Monolithic Memories.
LogicPak and ABEL are trademarks of Data I/O Corporation.
CUPL is a trademark of Personal CAD Systems.

Features/Benefits

- User-programmable synchronous state machine
- 25MHz maximum frequency for compatibility with 12.5MHz processors
- 14 inputs (8 external), 8 outputs, 128 states
- PAL® array optimizes product terms and states
- Internal feedback adds versatility and control
- Optimized for four-way branching
- User-selectable asynchronous preset or asynchronous enable function
- Power-up preset for start-up in known state
- Diagnostics-On-Chip™ shadow register eases chip and board-level testing
- PROSE™ device software makes it easy to "write your sequencer in PROSE"
- Programmed on standard logic programmers
- Security fuse prevents pattern duplication
- Space-saving 24-pin 300-mil SKINNYDIP® and 28-pin PLCC and LCC packages

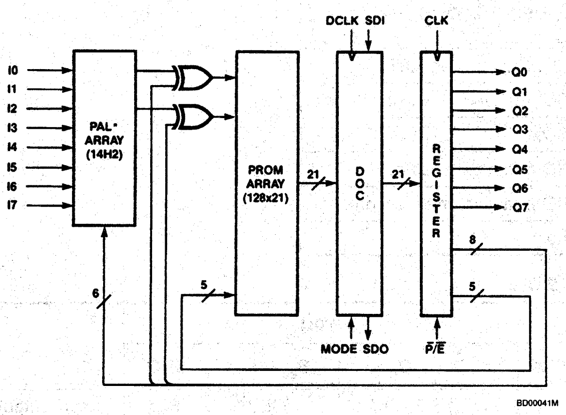
Description

The PMS14R21 programmable sequencer is the first member of the PROSE (PROgrammable SEquencer) family. The PMS14R21 is a high-speed, 14-input, 8-output state machine. It consists of a 128x21 PROM array preceded by a 14H2 PAL array. The PAL array is efficient for a large number of input conditions, while the PROM array is optimal for a large number of product terms and states. The combination allows a very efficient state machine with a large number of inputs and state bits. The PAL array, with eight product terms per output, operates on the eight conditional and six state inputs to select two control bits to the PROM. Two Exclusive-OR gates between the two arrays help to minimize product terms and redundant states. Five lines feed back from the PROM to form the primary address for the next state. The PROM stores up to 128 states of eight outputs and thirteen feedback control signals.

Applications

- High speed sequential logic
- Peripheral controller
- Cache control sequencer
- Signal processing sequencer
- Industrial control

Block Diagram

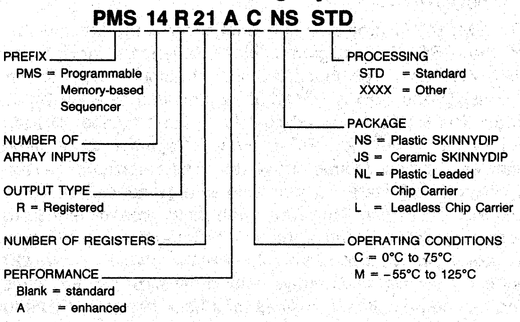


Definition of Signals

I0-I7	Primary inputs to the PAL array
Q0-Q7	Outputs from the register
$\overline{P}/\overline{E}$	Programmable asynchronous preset ($\overline{P}$) or asynchronous enable ($\overline{E}$)
CLK	Clock for output register
DCLK	Clock for diagnostic register
MODE	Selects diagnostic functions
SDI	Serial data input to shadow register
SDO	Serial data output from shadow register

PROSE™ and Diagnostics-On-Chip™ are trademarks of Monolithic Memories, Inc.

PROSE Part Numbering System



Diagnostics-On-Chip Feature

The PMS14R21 is the newest member of the Diagnostics-On-Chip family. These devices incorporate a serial shadow register on-chip which facilitates board-level testing. The shadow register has a Serial Data Input (SDI), Serial Data Output (SDO) and its own clock (DCLK). The MODE control configures the shadow register either in parallel with the output register or in serial shift mode (see function table). Other devices with this feature are listed below.

Diagnostics Family Members

PART NUMBER	DESCRIPTION
53/63DA441	1K x 4 PROM (async. enables)
53/63DA442	1K x 4 PROM (async/sync. enables)
53/63DA841	2K x 4 PROM
53/63D1641	4K x 4 PROM (async. enable)
53/63DA1643	4K x 4 PROM (async. initialization)
54/74S818	8-bit register

Software Support

PROSE device software from Monolithic Memories provides full support for the PMS14R21. Based on PALASM®2 syntax, the software automatically converts a state machine description directly into the PAL and PROM array fuse maps, for downloading to a programmer. The syntax supports both Mealy and Moore state machine models, and makes optimal use of the features of the PROSE device. Simulation support is also provided, both for design checking and for generation of test vectors for device testing. Additional support is available from third-party software vendors, including the ABEL™ package from Data I/O.

Programming

Both the PAL and PROM arrays are programmed on standard logic programmers using the JEDEC programming format. The TiW fuses program from the low to the high state. Programming also sets the architectural fuse which selects between asynchronous preset or asynchronous output enable; the unprogrammed state is preset. If asynchronous preset is selected, asserting the pin low will set all outputs and feedback bits high.

Power-up Preset

Power-up preset is provided for system start-up in a known state. It has the same effect as preset; all output register bits go high.

Diagnostic Function Table

INPUTS				OUTPUTS			OPERATION
MODE	SDI	CLK	DCLK	$Q_{20} - Q_0$	$S_{20} - S_0$	SDO	
L	X	↑	*	$Q_n \leftarrow$ PROM	HOLD	S_{20}	Load output register from PROM array
L	X	*	↑	HOLD	$S_n \leftarrow S_{n-1}$ $S_0 \leftarrow$ SDI	S_{20}	Shift shadow register data
L	X	↑	↑	$Q_n \leftarrow$ PROM	$S_n \leftarrow S_{n-1}$ $S_0 \leftarrow$ SDI	S_{20}	Load output register from PROM array while shifting shadow register data
H	X	↑	*	$Q_n \leftarrow S_n$	HOLD	SDI	Load output register from shadow register
H	L	*	↑	HOLD	$S_n \leftarrow Q_n$	SDI	Load shadow register from output bus and feedback
H	H	*	↑	HOLD	HOLD	SDI	† No operation

* Clock must be steady or falling.

† Reserved operation for 54/74S818 8-Bit Diagnostic Register.

Features/Benefits

- 20 logic inputs: 12 external, 8 feedback
- 8 outputs with programmable polarity
- ECL technology for ultra-high speed — max t_{PD} = 6ns
- 32 product terms with term sharing
- 10KH ECL compatible
- Fully AC tested
- Input pull-down resistors
- Voltage compensated
- Space-saving 24-pin SKINNYDIP® and 28-pin PLCC packages
- Programmable using standard TTL programmers with adapter
- Greater than 99% programming yield
- Security fuse prevents unauthorized copying

Description

The PAL10H20P8 is a 10KH family compatible ECL PAL device having twelve dedicated inputs and eight outputs with feedback. A programmable AND array and a fixed OR array make possible the implementation of a wide variety of logic functions with far fewer packages than with SSI devices. The logic is implemented by opening metal fuse connections within the AND array. Designs can be specified by using any of a variety of software packages which accept the design and assemble a file that can be downloaded into a device programmer. Fuses are programmed using any of the qualified PAL device programmers.

The outputs are equipped with programmable polarity. They can drive a 50Ω termination (to V_{CC} - 2.0V). Product term sharing is provided to allow greater flexibility in assigning product terms to outputs.

The input pins have 50kΩ internal pull-down resistors, which allow unused inputs to be left open. Open inputs will assume a logic low state.

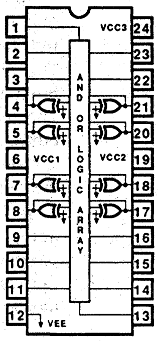
Features

Each output has a programmable polarity fuse, allowing for more efficient representation of many logic functions. Each output is active high with polarity fuse intact, and active low with the polarity fuse blown.

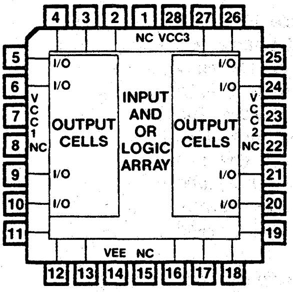
The programmable AND array contains a total of thirty-two product terms. Product terms are arranged in groups of eight. The terms in each group can be shared mutually exclusively between two adjacent output cells. If a particular product term is needed for two outputs, then two identical product terms are generated: one for each output.

A security fuse is provided to help protect the fuse pattern from unauthorized copying. Once the security fuse has been programmed, it is no longer possible to verify the contents of the fuse array electrically. The security fuse has no effect on functionality.

DIP Pinout



PLCC Pinout



CD00880M

Absolute Maximum Ratings

These ratings specify the conditions above which the device may be permanently damaged. AC and DC specifications are not necessarily guaranteed over this range.

Supply voltage V_{EE} (V_{CC1} = V_{CC2} = V_{CC3} = 0V)	–8.0V to 0V
Input voltage V_I (V_{CC1} = V_{CC2} = V_{CC3} =0V)	0V to V_{EE}
Output current, I_{OUT}	
Continuous	35mA
Surge	100mA
Storage temperature range, T_{stg}	–65°C to 150°C

Operating Conditions

SYMBOL	PARAMETER	COMMERCIAL			UNIT
		MIN	TYP	MAX	
V_{EE}	Supply voltage (V_{CC} = 0V)	–5.46	–5.2	–4.94	V
T_A	Operating free-air temperature	0		75	°C

Electrical Characteristics V_{EE} = –5.2V ± 5% (See note 1)

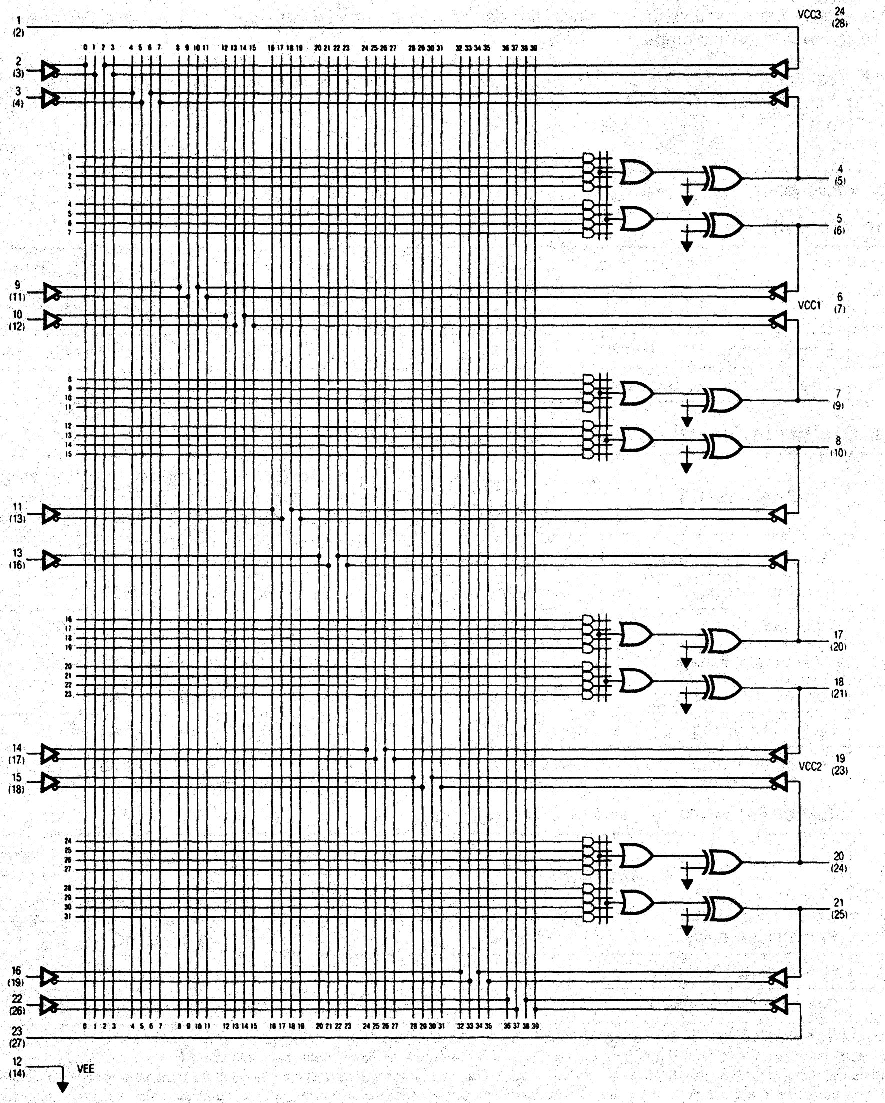
SYMBOL	PARAMETER	TEST CONDITIONS	0°C		25°C		75°C		UNIT
			MIN	MAX	MIN	MAX	MIN	MAX	
I_{EE}	Power supply current	Inputs V_{IN} = V_{IH} Max	–	210	–	210	–	210	mA
I_{inH}	Input current high	V_{IH} Min < V_{in} < V_{IH} Max	–	425	–	265	–	265	μA
I_{inL}	Input current low	V_{IL} Min < V_{in} < V_{IL} Max	0.5	–	0.5	–	0.3	–	μA
V_{OH}	High output voltage	(See Note 2)	–1.02	–0.84	–0.98	–0.81	–0.92	–0.735	V_{dc}
V_{OL}	Low output voltage	(See Note 2)	–1.95	–1.63	–1.95	–1.63	–1.95	–1.60	V_{dc}
V_{IH}	High input voltage	(See Note 2)	–1.17	–0.84	–1.13	–0.81	–1.07	–0.735	V_{dc}
V_{IL}	Low input voltage	(See Note 2)	–1.95	–1.48	–1.95	–1.48	–1.95	–1.45	V_{dc}

Switching Characteristics V_{EE} = –5.2V ± 5% (See note 2)

SYMBOL	PARAMETER	0°C		25°C		75°C		UNIT
		MIN	MAX	MIN	MAX	MIN	MAX	
t_{PD}	Propagation delay	2.0	6.0	2.0	6.0	2.0	6.0	ns
t_R	Rise time (20%-80%)	0.7	2.2	0.7	2.0	0.7	2.2	ns
t_F	Fall time (80%-20%)	0.7	2.2	0.7	2.0	0.7	2.2	ns

Note:
1. Each ECL 10KH series circuit has been designed to meet the specifications shown in test table after thermal equilibrium has been established. The circuit is in test socket or mounted on a printed board and transverse air flow greater than 500 linear fpm is maintained.
2. Outputs are terminated through a 50Ω resistor to V_{CC} – 2.0V. Discrete carbon resistors should be used for terminations. Multiple-resistor packs and metal film discrete resistors are inductive and should be avoided. The single-ended nature of the outputs demands strict adherence to ground and termination plane design techniques.
3. If pin 13 (PLCC pin 16) is not used, it should be left open or terminated to V_{TT} (= V_{CC} – 2.0V). It should not be terminated to V_{EE}.

Logic Diagram



LD00840M

MegaPAL Devices

	ARRAY INPUTS	REGISTERED OUTPUTS	t_{PD} (ns)	I_{CC} (mA)
PAL32R16	32	16	40	280
PAL64R32	64	32	50	640

Description

The MegaPAL Devices offer very high density programmable logic.

Programmable Polarity

Each flip-flop has individually programmable polarity. The unprogrammed state is active low.

Product Term Sharing

Product term sharing allows each pair of outputs to share its product terms with one output or the other (not both). Each pair has a total of sixteen product terms; thus, one output can use zero to sixteen terms while the other has sixteen to zero. Product terms can only be shared mutually exclusively. If both outputs need the same term, it must be created twice, once for each output.

Register Bypass

Registers in either device can be bypassed in banks of eight, creating a set of combinatorial outputs.

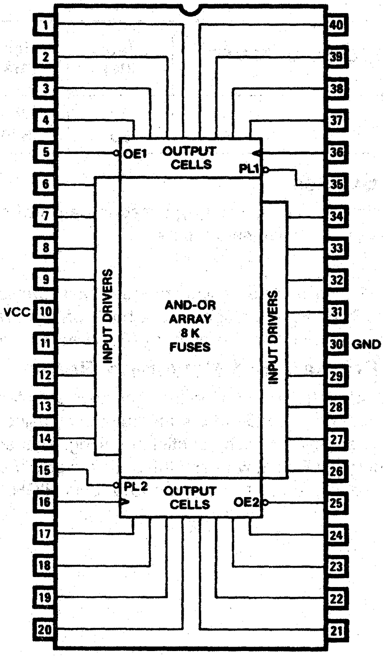
Reset

The PAL64R32 also features asynchronous reset (previously called preset). The reset function sets a bank of eight registers to a logic LOW, setting the output HIGH if active low.

Register Preload and Power-up Reset

Both devices also offer register preload for device testability. The registers can be preloaded from the outputs by using TTL level signals in order to simplify functional testing. This series also offers Power-up Reset, whereby the registers power up to a logic LOW, setting active-low outputs to a logic HIGH.

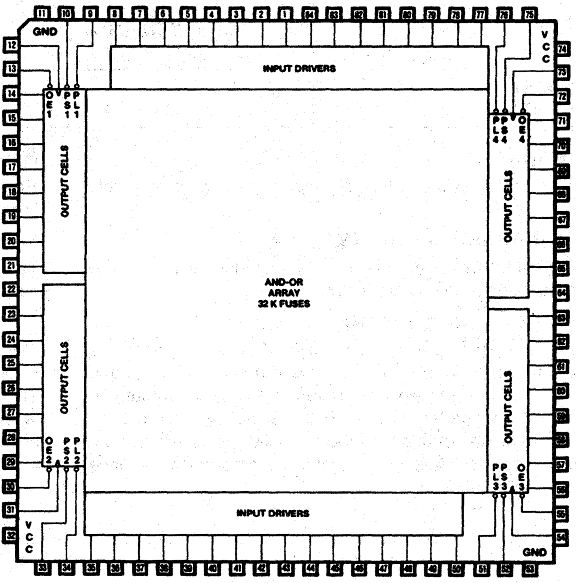
32R16



DIP

CD00250M

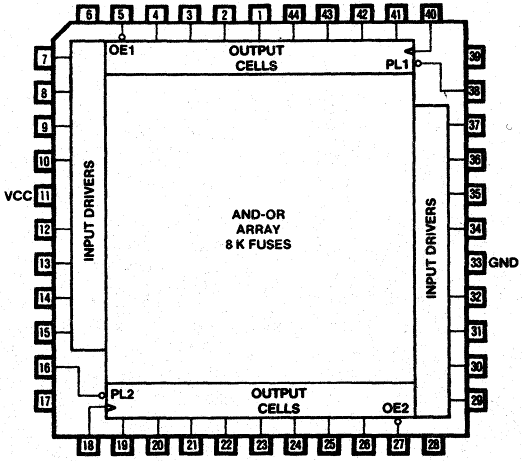
64R32



Plastic Chip Carrier

CD00260M

32R16



Plastic Chip Carrier

CD00270M

Testing Conditions

SYMBOL	PARAMETER	COMMERCIAL			UNIT
		MIN	TYP	MAX	
t_{wp}	Preload pulse width	35			ns
t_{sup}	Preload setup time	50			ns
t_{hp}	Preload hold time	5			ns
t_{PRW}	Preset pulse width	25			ns
t_{PRR}	Preset recovery time	35			ns

Operating Conditions

SYMBOL	PARAMETER		MILITARY			COMMERCIAL			UNIT
			MIN	TYP	MAX	MIN	TYP	MAX	
V_{CC}	Supply voltage		4.5	5	5.5	4.75	5	5.25	V
t_w	Width of clock	Low	25			20			ns
		High	25			20			
t_{wp}	Preload pulse width		45			35			ns
t_{su}	Setup time for input to clock	Polarity fuse intact	50			40			ns
		Polarity fuse blown	50			40			
t_{sup}	Preload setup time		30			25			ns
t_h	Hold time		0	−10		0	−10		ns
t_{hp}	Preload hold time		10			5			ns
T_A	Operating free-air temperature		−55			0		75	°C
T_C	Operating case temperature				125				°C

Electrical Characteristics (Over Operating Conditions)

SYMBOL	PARAMETER	TEST CONDITION			MIN	TYP	MAX	UNIT
V_{IL} [1]	Low-level input voltage						0.8	V
V_{IH} [1]	High-level input voltage				2			V
V_{IC}	Input clamp voltage	V_{CC} = MIN		I_I = −18mA		−0.8	−1.5	V
I_{IL} [2]	Low-level input current	V_{CC} = MAX		V_I = 0.4V		−0.02	−0.25	mA
I_{IH} [2]	High-level input current	V_{CC} = MAX		V_I = 2.4V			25	μA
I_I	Maximum input current	V_{CC} = MAX		V_I = 5.5V			1	mA
V_{OL}	Low-level output voltage	V_{CC} = MIN	Mil	I_{OL} = 8mA		0.3	0.5	V
			Com	I_{OL} = 8mA				
V_{OH}	High-level output voltage	V_{CC} = MIN	Mil	I_{OH} = −2mA	2.4	2.8		V
			Com	I_{OH} = −3.2mA				
I_{OZL} [2]	Off-state output current	V_{CC} = MAX		V_O = 0.4V			−100	μA
I_{OZH} [2]				V_O = 2.4V			100	μA
I_{OS} [3]	Output short-circuit current	V_{CC} = MAX		V_O = 0V	−30	−70	−130	mA
I_{CC}	Supply current	V_{CC} = MAX				200	280	mA

Switching Characteristics (Over Operating Conditions)

SYMBOL	PARAMETER		TEST CONDITIONS	MILITARY			COMMERCIAL			UNIT
				MIN	TYP	MAX	MIN	TYP	MAX	
t_{PD}	Input to output	Polarity fuse intact	R_1 = 560Ω R_2 = 1.1KΩ			50			40	ns
		Polarity fuse blown				55			45	
t_{CLK}	Clock to output or feedback					30			25	ns
t_{PZX}	Output enable					25			20	ns
t_{PXZ}	Output disable					25			20	ns
f_{MAX}	Maximum frequency			14			16			MHz

1. These are absolute voltages with respect to the ground pin on the device and include all overshoots due to system and/or tester noise. Do not attempt to test these values without suitable equipment.
2. I/O pin leakage is the worst case of I_{IL} and I_{OZL} (or I_{IH} and I_{OZH}).
3. No more than one output should be shorted at a time and duration of the short circuit should not exceed one second.

Operating Conditions

SYMBOL	PARAMETER		COMMERCIAL[1]			UNIT
			MIN	TYP	MAX	
V_{CC}	Supply voltage		4.75	5	5.25	V
t_w	Width of clock	Low	20			ns
		High	20			
t_{su}	Setup time from input to clock	Polarity fuse intact	40			ns
		Polarity fuse blown	40			
t_h	Hold time		0	-10		ns
T_A	Operating free-air temperature		0		75	°C
T_C	Operating case temperature					°C

Electrical Characteristics (Over Operating Conditions)

SYMBOL	PARAMETER	TEST CONDITION		COMMERCIAL			UNIT
				MIN	TYP	MAX	
V_{IL}[2]	Low-level input voltage					0.8	V
V_{IH}[2]	High-level input voltage			2			V
V_{IC}	Input clamp voltage	V_{CC} = MIN	I_I = −18mA		-0.8	-1.5	V
I_{IL}[3]	Low-level input current	V_{CC} = MAX	V_I = 0.4V		-0.02	-0.25	mA
I_{IH}[3]	High-level input current	V_{CC} = MAX	V_I = 2.4V			25	μA
I_I	Maximum input current	V_{CC} = MAX	V_I = 5.5V			1	mA
V_{OL}	Low-level output voltage	V_{CC} = MIN	I_{OL} = 8mA		0.3	0.5	V
V_{OH}	High-level output voltage	V_{CC} = MIN	I_{OH} = −0.4mA	2.4	2.8		V
I_{OZL}[3]	Off-state output current	V_{CC} = MAX	V_O = 0.4V			-100	μA
I_{OZH}[3]			V_O = 2.4V			100	μA
I_{OS}[4]	Output short-circuit current	V_{CC} = 5V	V_O = 0V	-10	-40	-60	mA
I_{CC}	Supply current	V_{CC} = MAX			400	640	mA

Switching Characteristics (Over Operating Conditions)

SYMBOL	PARAMETER		TEST CONDITIONS	COMMERCIAL			UNIT
				MIN	TYP	MAX	
t_{PD}	Input to output	Polarity fuse intact	R_1 = 560Ω R_2 = 1.1KΩ			50	ns
		Polarity fuse blown				55	
t_{CLK}	Clock to output or feedback					22	ns
t_{PZX}	Output enable					30	ns
t_{PXZ}	Output disable					30	ns
t_{PRH}	Preset to output					35	ns
f_{MAX}	Maximum frequency			16	20		MHz

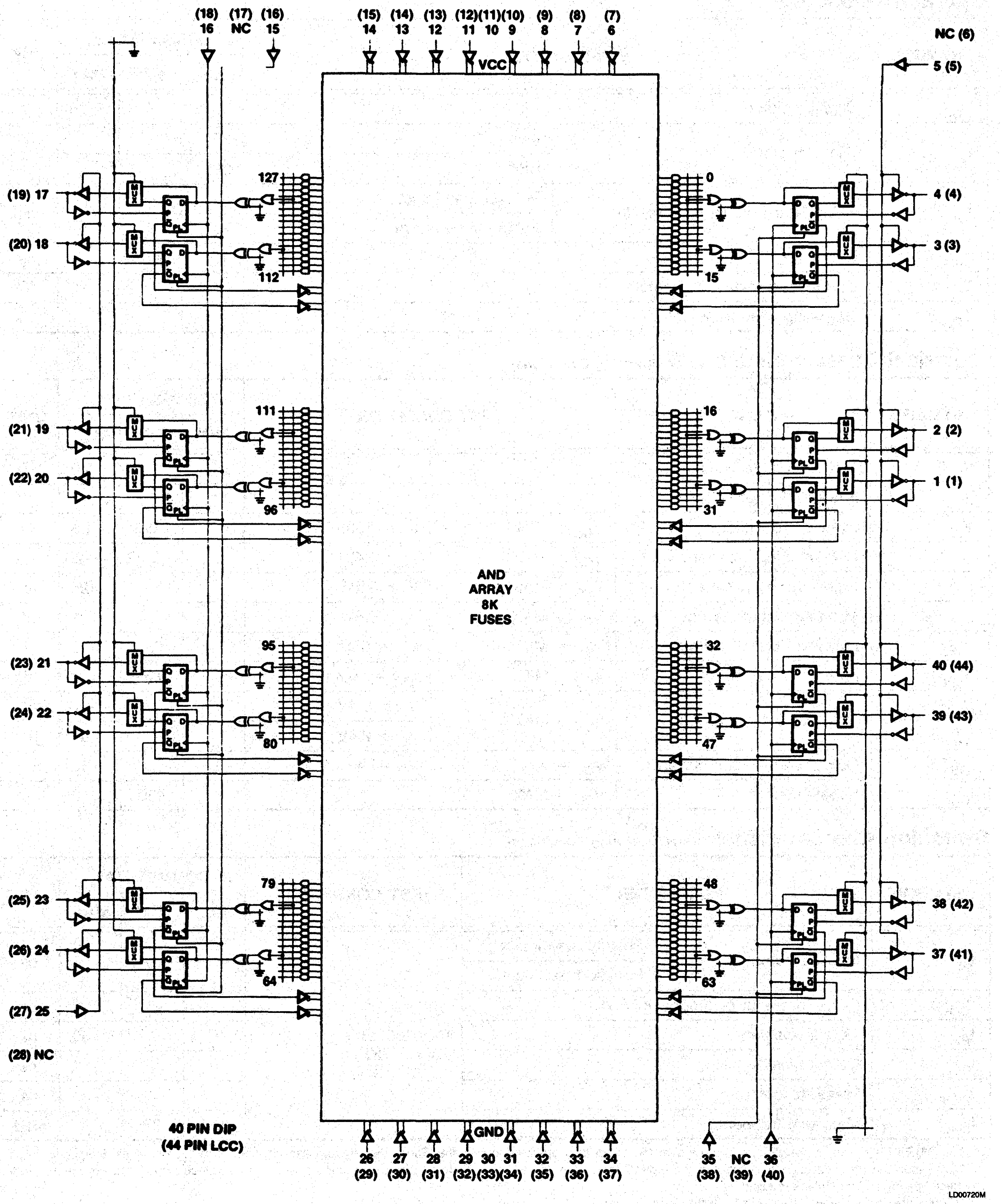
[1] The PAL64R32 Series is designed to operate over the full military operating conditions. For availability and specifications, contact Monolithic Memories.

[2] These are absolute voltages with respect to the ground pin on the device and include all overshoots due to system and/or tester noise. Do not attempt to test these values without suitable equipment.

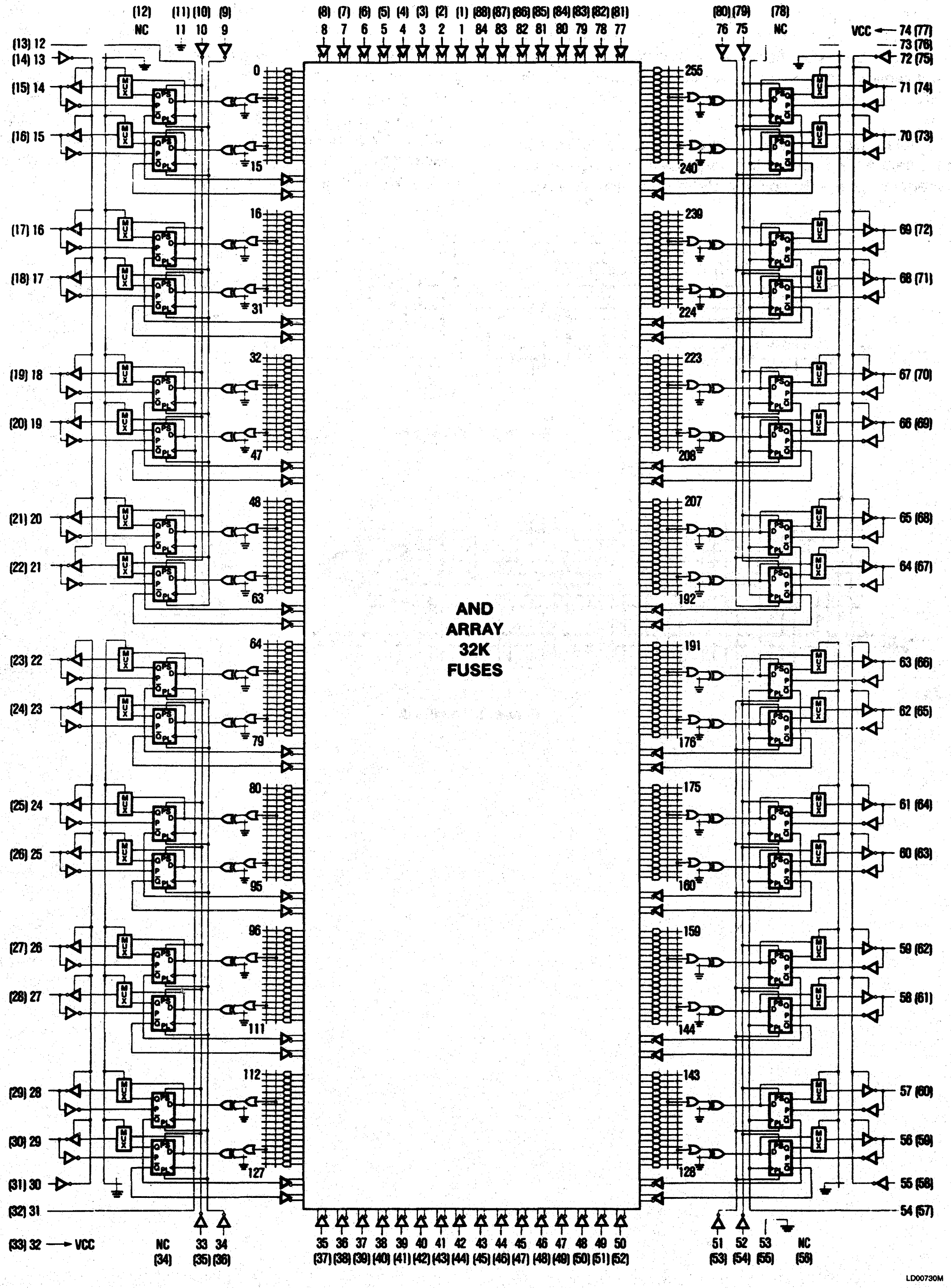
[3] I/O pin leakage is the worst case of I_{IL} and I_{OZL} (or I_{IH} and I_{OZH}).

[4] No more than one output should be shorted at a time and duration of the short circuit should not exceed one second.

32R16


(18) 16
(17) NC
(16) 15
(15) 14
(14) 13
(13) 12
(12) 11
(11) 10
(10) 9
(9) 8
(8) 7
(7) 6
VCC
NC (6)
5 (5)
(19) 17
(20) 18
(21) 19
(22) 20
(23) 21
(24) 22
(25) 23
(26) 24
(27) 25
(28) NC
4 (4)
3 (3)
2 (2)
1 (1)
40 (44)
39 (43)
38 (42)
37 (41)
127
112
111
96
95
80
79
64
0
15
16
31
32
47
48
63
MUX
D Q
P
Q̄
PL
AND
ARRAY
8K
FUSES
40 PIN DIP
(44 PIN LCC)
GND
26 (29)
27 (30)
28 (31)
29 (32)
30 (33)
31 (34)
32 (35)
33 (36)
34 (37)
35 (38)
NC (39)
36 (40)
LD00720M

64R32


(12) NC
(11) 11
(10) 10
(9) 9
(8) 8
(7) 7
(6) 6
(5) 5
(4) 4
(3) 3
(2) 2
(1) 1
(88) 84
(87) 83
(86) 82
(85) 81
(84) 80
(83) 79
(82) 78
(81) 77
(80) 76
(79) 75
(78) NC
VCC ← 74 (77)
73 (76)
72 (75)
71 (74)
70 (73)
69 (72)
68 (71)
67 (70)
66 (69)
65 (68)
64 (67)
63 (66)
62 (65)
61 (64)
60 (63)
59 (62)
58 (61)
57 (60)
56 (59)
55 (58)
54 (57)
(13) 12
(14) 13
(15) 14
(16) 15
(17) 16
(18) 17
(19) 18
(20) 19
(21) 20
(22) 21
(23) 22
(24) 23
(25) 24
(26) 25
(27) 26
(28) 27
(29) 28
(30) 29
(31) 30
(32) 31
(33) 32 → VCC
NC (34)
33 (35)
34 (36)
35 (37)
36 (38)
37 (39)
38 (40)
39 (41)
40 (42)
41 (43)
42 (44)
43 (45)
44 (46)
45 (47)
46 (48)
47 (49)
48 (50)
49 (51)
50 (52)
51 (53)
52 (54)
53 (55)
NC (56)
MUX
PS
PL
D
Q
P
0
15
16
31
32
47
48
63
64
79
80
95
96
111
112
127
255
240
239
224
223
208
207
192
191
176
175
160
159
144
143
128
AND ARRAY 32K FUSES
LD00730M

f_{MAX} Parameters

The parameter f_{MAX} is the maximum speed at which the PAL device is guaranteed to operate. Because flexibility inherent to PAL devices allows a choice of clocked flip-flop designs, for the convenience of the user, f_{MAX} for the B-speed devices is specified to address two major classes of synchronous designs.

The simplest type of synchronous design can be described as a data path application. In this case, data is presented to the data terminal of the flip-flop and clocked through; no feedback is employed (Figure 1). Under these conditions, the frequency of operation is limited by the greater of the data setup time (t_{su}) or the minimum clock period (t_w high + t_w low). This parameter is designated f_{MAX} (no feedback).

For synchronous sequential designs, i.e., state machines, where logical feedback is required, inputs to flip-flop data terminals originate from the device input pins of flip-flop outputs via the internal feedback paths (Figure 2). Under these conditions, f_{MAX} is defined as the reciprocal of (t_{su} + t_{CLK}) and is designated f_{MAX} (feedback).

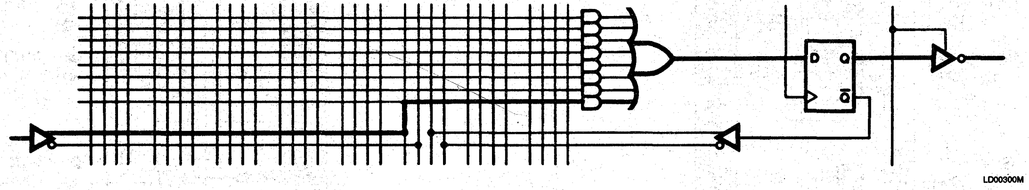


Figure 1. No Feedback

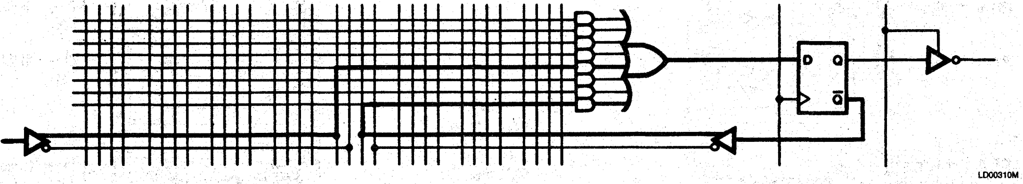


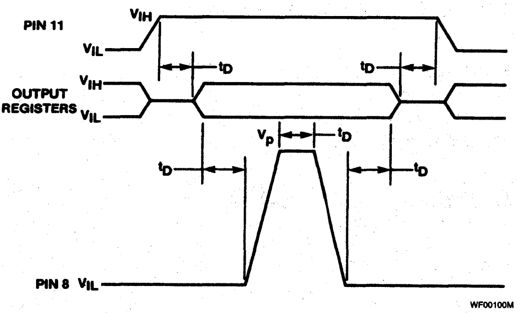
Figure 2. Feedback

Output Register PRELOAD

The PRELOAD function allows the register to be loaded from data placed on the output pins. This feature aids functional testing of sequential designs by allowing direct setting of output states. The procedure for PRELOAD is as follows:

Series 20PA

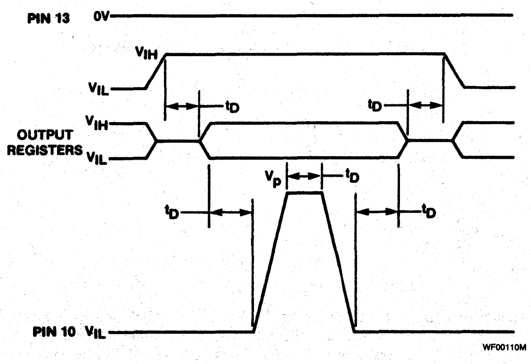
1. Raise V_{CC} to 4.5V.
2. Disable output registers by setting pin 11 to V_{IH}.
3. Apply V_{IL}/V_{IH} to all registered output pins.
4. Pulse pin 8 to V_p, then back to 0V.
5. Remove V_{IL}/V_{IH} from all output registers.
6. Lower pin 11 to V_{IL} to enable the output registers.
7. Verify for V_{OL}/V_{OH} at all registered output pins.



WF00100M

Series 24RS/24XA

1. Raise V_{CC} to 4.5V.
2. Disable output registers by setting pin 13 to V_{IH}.
3. Apply V_{IL}/V_{IH} to all registered output pins.
4. Pulse pin 10 to V_p, then back to 0V.
5. Remove V_{IL}/V_{IH} from all output registers.
6. Lower pin 13 to V_{IL} to enable the output registers.
7. Verify for V_{OL}/V_{OH} at all registered output pins.

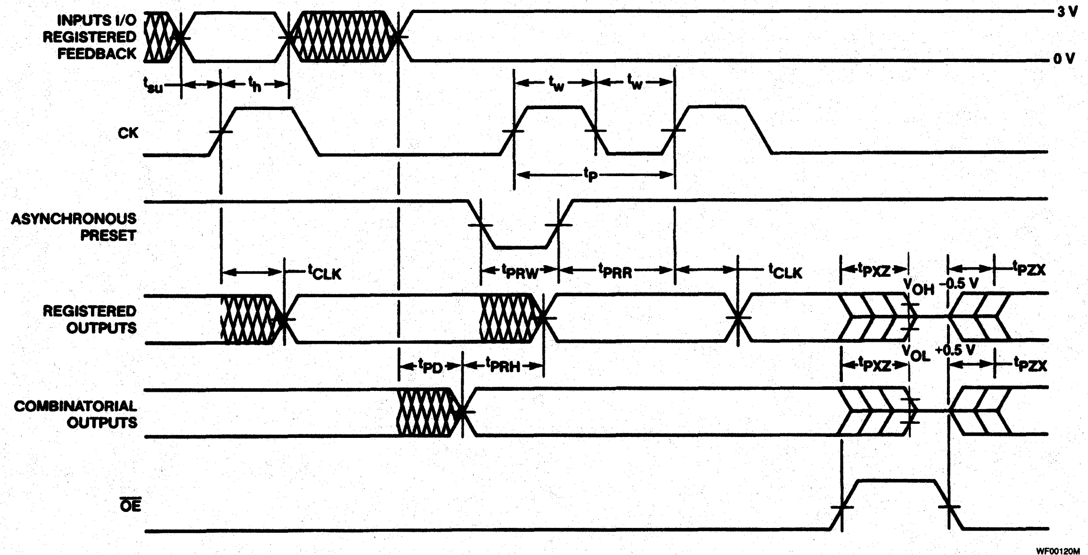


WF00110M

Power-Up RESET

All devices with this PRELOAD feature also have power-up RESET. All registers power up to a logic high for predictable system initialization.

Switching Waveforms



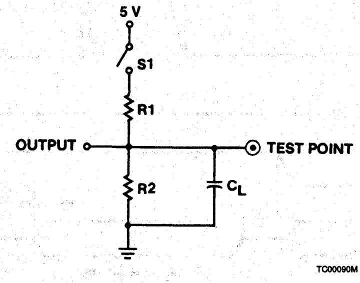
WF00120M

Notes: 1. Input pulse amplitude 0V to 3.0V.
2. Input access measured at the 1.5V level.

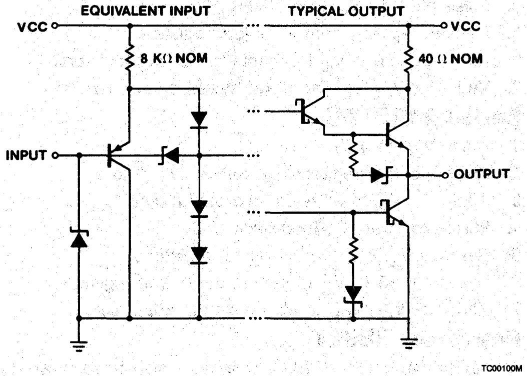
Absolute Maximum Ratings

	Operating	Programming
Supply voltage V_{CC}	−0.5V to 7V	−0.5V to 12V
Input voltage	−1.5V to 5.5V	−1.0V to 22V
Off-state output voltage	5.5V	12V
Storage temperature		−65°C to +150°C

Switching Test Load



Schematic of Inputs and Outputs



Notes:
1. t_{PD} is tested with switch S_1 closed, C_L = 50pF and measured at 1.5V output level.
2. t_{PZX} is measured at the 1.5V output level with C_L = 50pF. S_1 is open for high impedance to "1" test, and closed for high impedance to "0" test.
3. t_{PXZ} is tested with C_L = 5pF. S_1 is open for "1" to high impedance test, measured at V_{OH} −0.5V output level; S_1 is closed for "0" to high impedance test measured at V_{OL} +0.5V output level.

Table of Contents

Data I/O Corporation

Digelec

Kontron

Micropross

Promac

Stag Microsystems

Storey Systems

Structured Design

Valley Data Sciences

Varix

Data I/O Corporation

20 Pin Device Families

10525 Willows Road N.E.
PO Box 97046
Redmond, WA 98073-9746
(800) 247-5700

Models 19, 29A, 29B
Model 60

3

Series	Part Number	System	LogicPak™	Adapter	Family Code	Pinout Code
Small 20/-2 Combinatorial	PAL10H8/-2	Model 60 Rev. V05	303A	303A-002-V08 303A-011A/B-V01	22	18
	PAL10L8/-2		↓	↓	↓	13
	PAL12H6/-2		↓	↓	↓	19
	PAL12L6/-2	Models 19 29A 29B	↓	↓	↓	14
	PAL14H4/-2	↓	↓	↓	↓	20
	PAL14L4/-2	↓	↓	↓	↓	15
	PAL16H2/-2	↓	↓	↓	↓	22
	PAL16L2/-2	↓	↓	↓	↓	16
	PAL16C1/-2	↓	↓	↓	↓	21
Medium 20/ 20A/20A-2/4/ 20B-2/-4 Standard	PAL16L8/A/-2/-4/B-2/-4	↓	↓	↓	↓	17
	PAL16R8/A/-2/-4/B-2/-4	↓	↓	↓	↓	24
	PAL16R6/A/-2/-4/B-2/-4	↓	↓	↓	↓	↓
	PAL16R4/A/-2/-4/B-2/-4	↓	↓	↓		↓
Medium 20B/D	PAL16L8B/D	↓	↓	↓	30	17
	PAL16R8B/D	↓	↓	↓	↓	24
	PAL16R6B/D	↓	↓	↓	↓	↓
	PAL16R4B/D	↓	↓	↓	↓	↓
Medium 20BP Standard	PAL16L8BP	↓	303A-V04	↓	22	17
	PAL16R8BP	↓	↓	↓	↓	67
	PAL16R6BP	↓	↓	↓	↓	↓
	PAL16R4BP	↓	↓	↓	↓	↓
Medium 20PA Programmable Polarity	PAL16P8A	↓	↓	↓	↓	30
	PAL16RP8A	↓	↓	↓	↓	31
	PAL16RP6A	↓	↓	↓	↓	↓
	PAL16RP4A	↓	↓	↓	↓	↓
Large 20 Arithmetic	PAL16X4	↓	303A	↓	↓	24
	PAL16A4	↓	↓	↓	↓	
Large 20RA Asynchronous	PAL16RA8	↓	↓	↓	↓	30

Note: The software and hardware revisions listed are the earliest revisions that support these products.
Later software and hardware revisions can also be assumed to support these products.

Data I/O

24 Pin and MegaPAL™ Device Families

Series	Part Number	System	LogicPak™	Adapter	Family Code	Pinout Code
Small 24 Combinatorial	PAL12L10	Model 60	303A	303A-002-V08	22	01
	PAL14L8	Rev. V05		303A-011A/B-V01		02
	PAL16L6	Models				03
	PAL18L4	19				04
	PAL20L2	29A				05
	PAL20C1	29B				12
Small 24A Decoder	PAL6L16A					48
	PAL8L14A					49
Medium 24A/ 24A-2/B Standard	PAL20L8A/-2/B					26
	PAL20R8A/-2/B					27
	PAL20R6A/-2/B					
	PAL20R4A/-2/B					↓
Medium 24X Exclusive-OR	PAL20L10					06
	PAL20X10					23
	PAL20X8					
	PAL20X4		↓			↓
Medium 24XA Exclusive-OR	PAL20L10A		303A-V04			06
	PAL20X10A					36
	PAL20X8A					
	PAL20X4A					↓
Large 24RS Shared Product Terms	PAL20S10					43
	PAL20RS10					44
	PAL20RS8					
	PAL20RS4			↓		46
Large 24A Registered XOR	PAL22RX8A			303A-011A/B-V01		78
Large 24/A Varied XOR	PAL32VX10/A		↓			77
Large 24RA Asynchronous	PAL20RA10		303A	303A-002-V08 303A-011A/B-V01		45
ECL Combinatorial	PAL10H20P8[1]	↓	303A-V04	303A-ECL		42
MegaPAL™ Devices	PAL32R16	Model 29B	303A	303A-088-A/B		47
	PAL64R32		350A/-23/A/B		↓	84

NOTE: The software and hardware revisions listed are the earliest revisions that support these products. Later software and hardware revisions can also be assumed to support these products.

[1] Not supported on Model 19

Digelec

20 Pin Device Families

1602 Lawrence Ave.
Suite 113
Ocean, NJ 07712
(201) 493-2420

System UP803

Series	Part Number	FAM 52 Rev.	Adapter	Adapter Rev.
Small 20/-2 Combinatorial	PAL10H8/-2	5.4	DA53	A-3
	PAL10L8/-2			
	PAL12H6/-2			
	PAL12L6/-2			
	PAL14H4/-2			
	PAL14L4/-2			
	PAL16H2/-2			
	PAL16L2/-2			
	PAL16C1/-2			
Medium 20/ 20A/20A-2/4/ 20B-2/-4 Standard	PAL16L8/A/-2/-4/B-2/-4			
	PAL16R8/A/-2/-4/B-2/-4			
	PAL16R6/A/-2/-4/B-2/-4			
	PAL16R4/A/-2/-4/B-2/-4			
Medium 20B/D	PAL16L8B/D			
	PAL16R8B/D			
	PAL16R6B/D			
	PAL16R4B/D			
Medium 20BP Standard	PAL16L8BP			
	PAL16R8BP			
	PAL16R6BP			
	PAL16R4BP			
Medium 20PA Programmable Polarity	PAL16P8A	5.4	DA53	A-3
	PAL16RP8A			
	PAL16RP6A			
	PAL16RP4A			
Large 20 Arithmetic	PAL16X4			
	PAL16A4			
Large 20RA Asynchronous	PAL16RA8	Under Development		

NOTE: The software and hardware revisions listed are the earliest revisions that support these products.
Later software and hardware revisions can also be assumed to support these products.

Digelec

24 Pin and MegaPAL™ Device Families

Series	Part Numbers	FAM 52	Adapter	Adapter Rev.
Small 24 Combinatorial	PAL12L10	5.4	DA55	C-1
	PAL14L8	↓	↓	↓
	PAL16L6			
	PAL18L4			
	PAL20L2			
	PAL20C1			
Small 24A Decoder	PAL6L16A	Under Development		
	PAL8L14A			
Medium 24A/ 24A-2/B Standard	PAL20L8A/-2/B	5.4	DA55	C-1
	PAL20R8A/-2/B	↓	↓	↓
	PAL20R6A/-2/B			
	PAL20R4A/-2/B			
Medium 24X Exclusive-OR	PAL20L10			
	PAL20X10			
	PAL20X8			
	PAL20X4			
Medium 24XA Exclusive-OR	PAL20L10A			
	PAL20X10A			
	PAL20X8A			
	PAL20X4A			
Large 24RS Shared Product Terms	PAL20S10			
	PAL20RS10			
	PAL20RS8			
	PAL20RS4			
Large 24A Registered XOR	PAL22RX8A			
Large 24/A Varied XOR	PAL32VX10/A			
Large 24RA Asynchronous	PAL20RA10			
ECL Combinatorial	PAL10H20P8			
MegaPAL™ Devices	PAL32R16	Under Development		
	PAL64R32			

NOTE: The software and hardware revisions listed are the earliest revisions that support these products. Later software and hardware revisions can also be assumed to support these products.

Kontron

20 Pin Device Families

1230 Charleston Road
Mountain View, CA 94039-7230
(415) 965-7020

System MPP-80S

System EPP-80

Series	Part Number	Socket Adapter	UPM-B Rev.
Small 20/-2 Combinatorial	PAL10H8/-2	SA-27	1.44
	PAL10L8/-2	↓	↓
	PAL12H6/-2	↓	↓
	PAL12L6/-2	↓	↓
	PAL14H4/-2	↓	↓
	PAL14L4/-2	↓	↓
	PAL16H2/-2	↓	↓
	PAL16L2/-2	↓	↓
	PAL16C1/-2	↓	↓
Medium 20/ 20A/20A-2/4/ 20B-2/-4 Standard	PAL16L8/A/-2/-4/B-2/-4	↓	↓
	PAL16R8/A/-2/-4/B-2/-4	↓	↓
	PAL16R6/A/-2/-4/B-2/-4	↓	↓
	PAL16R4/A/-2/-4/B-2/-4	↓	↓
Medium 20B/D	PAL16L8B/D	Under Development	↓
	PAL16R8B/D		↓
	PAL16R6B/D		↓
	PAL16R4B/D		↓
Medium 20BP Standard	PAL16L8BP	SA-27	↓
	PAL16R8BP	Under Development	↓
	PAL16R6BP	↓	↓
	PAL16R4BP	↓	↓
Medium 20PA Programmable Polarity	PAL16P8A	↓	↓
	PAL16RP8A	↓	↓
	PAL16RP6A	↓	↓
	PAL16RP4A	↓	↓
Large 20 Arithmetic	PAL16X4	↓	↓
	PAL16A4	↓	↓
Large 20RA Asynchronous	PAL16RA8	↓	1.47

NOTE: The software and hardware revisions listed are the earliest revisions that support these products.
Later software and hardware revisions can also be assumed to support these products.

Kontron

24 Pin and MegaPAL™ Device Families

Series	Part Number	Socket Adapter	UPM-B Rev.
Small 24 Combinatorial	PAL12L10	SA-27-1	1.44 ↓
	PAL14L8		
	PAL16L6		
	PAL18L4		
	PAL20L2		
	PAL20C1		
Small 24A Decoder	PAL6L16A	Under Development	1.48
	PAL8L14A		
Medium 24A/ 24A-2/B Standard	PAL20L8A/-2/B	SA-27-1 ↓	1.44 ↓
	PAL20R8A/-2/B		
	PAL20R6A/-2/B		
	PAL20R4A/-2/B		
Medium 24X Exclusive-OR	PAL20L10		
	PAL20X10		
	PAL20X8		
	PAL20X4		
Medium 24XA Exclusive-OR	PAL20L10A		
	PAL20X10A		
	PAL20X8A		
	PAL20X4A		
Large 24RS Shared Product Terms	PAL20S10	Under Development ↓	
	PAL20RS10		
	PAL20RS8		
	PAL20RS4		
Large 24A Registered XOR	PAL22RX8A		Under Development
Large 24/A Varied XOR	PAL32VX10/A		1.48
Large 24RA Asynchronous	PAL20RA10		1.44
ECL Combinatorial	PAL10H20P8		1.47
MegaPAL™ Devices	PAL32R16		Under Development
	PAL64R32		

NOTE: The software and hardware revisions listed are the earliest revisions that support these products. Later software and hardware revisions can also be assumed to support these products.

Micropross

20 Pin Device Families

Parc d'activite des Pres
5, rue Denis-Papin
59650 Villeneuve-d'Ascq
Tel 20479040

ROM 5000

Programmer

Series	Part Number	Rev.
Small 20/-2 Combinatorial	PAL10H8/-2	3.5
	PAL10L8/-2	
	PAL12H6/-2	
	PAL12L6/-2	
	PAL14H4/-2	
	PAL14L4/-2	
	PAL16H2/-2	
	PAL16L2/-2	
	PAL16C1/-2	
Medium 20/ 20A/20A-2/4/ 20B-2/-4 Standard	PAL16L8/A/-2/-4/B-2/-4	
	PAL16R8/A/-2/-4/B-2/-4	
	PAL16R6/A/-2/-4/B-2/-4	
	PAL16R4/A/-2/-4/B-2/-4	
Medium 20B/D	PAL16L8B/D	
	PAL16R8B/D	
	PAL16R6B/D	
	PAL16R4B/D	
Medium 20BP Standard	PAL16L8BP	
	PAL16R8BP	
	PAL16R6BP	
	PAL16R4BP	
Medium 20PA Programmable Polarity	PAL16P8A	
	PAL16RP8A	
	PAL16RP6A	
	PAL16RP4A	
Large 20 Arithmetic	PAL16X4	
	PAL16A4	
Large 20RA Asynchronous	PAL16RA8	Under Development

NOTE: The software and hardware revisions listed are the earliest revisions that support these products.
Later software and hardware revisions can also be assumed to support these products.

Micropross

24 Pin and MegaPAL™ Device Families

Series	Part Number	Rev.
Small 24 Combinatorial	PAL12L10	3.5
	PAL14L8	
	PAL16L6	
	PAL18L4	
	PAL20L2	
	PAL20C1	
Small 24A Decoder	PAL6L16A	Under Development
	PAL8L14A	
Medium 24A/ 24A-2/B Standard	PAL20L8A/-2/B	3.5
	PAL20R8A/-2/B	
	PAL20R6A/-2/B	
	PAL20R4A/-2/B	
Medium 24X Exclusive-OR	PAL20L10	
	PAL20X10	
	PAL20X8	
	PAL20X4	
Medium 24XA Exclusive-OR	PAL20L10A	
	PAL20X10A	
	PAL20X8A	
	PAL20X4A	
Large 24RS Shared Product Terms	PAL20S10	
	PAL20RS10	
	PAL20RS8	
	PAL20RS4	
Large 24A Registered XOR	PAL22RX8A	
Large 24/A Varied XOR	PAL32VX10/A	
Large 24RA Asynchronous	PAL20RA10	
ECL Combinatorial	PAL10H20P8	
MegaPAL™ Devices	PAL32R16	Under Development
	PAL64R32	

NOTE: The software and hardware revisions listed are the earliest revisions that support these products. Later software and hardware revisions can also be assumed to support these products.

Promac

20 Pin Device Families

Adams MacDonald Enterprises, Inc.

2999 Monterey/Salinas Highway
Monterey, CA 93940
(408) 373-3607

Promac P3

Programmer

Series	Part Number	Software Rev.	S1/S2
Small 20/-2 Combinatorial	PAL10H8/-2	3.0	0/1
	PAL10L8/-2		0/6
	PAL12H6/-2		0/2
	PAL12L6/-2		0/7
	PAL14H4/-2		0/3
	PAL14L4/-2		0/8
	PAL16H2/-2		0/4
	PAL16L2/-2		0/9
	PAL16C1/-2		0/5
Medium 20/ 20A/20A-2/4/ 20B-2/-4 Standard	PAL16L8/A/-2/-4/B-2/-4		0/10
	PAL16R8/A/-2/-4/B-2/-4		0/11
	PAL16R6/A/-2/-4/B-2/-4		0/12
	PAL16R4/A/-2/-4/B-2/-4		0/13
Medium 20B/D	PAL16L8B/D		5/0
	PAL16R8B/D		5/1
	PAL16R6B/D		5/2
	PAL16R4B/D		5/3
Medium 20BP Standard	PAL16L8BP		0/10
	PAL16R8BP		0/11
	PAL16R6BP		0/12
	PAL16R4BP		0/13
Medium 20PA Programmable Polarity	PAL16P8A		1/0
	PAL16RP8A		1/3
	PAL16RP6A		1/2
	PAL16RP4A		1/1
Large 20 Arithmetic	PAL16X4		0/14
	PAL16A4		0/15
Large 20RA Asynchronous	PAL16RA8	↓	1/12

NOTE: The software and hardware revisions listed are the earliest revisions that support these products. Later software and hardware revisions can also be assumed to support these products.

Promac

24 Pin and MegaPAL™ Device Families

Series	Part Number	Software Rev.	S1/S2
Small 24 Combinatorial	PAL12L10	3.0	2/2
	PAL14L8		2/3
	PAL16L6		2/4
	PAL18L4		2/5
	PAL20L2		2/6
	PAL20C1		2/1
Small 24A Decoder	PAL6L16A		3/11
	PAL8L14A		3/10
Medium 24A/ 24A-2/B Standard	PAL20L8A/-2/B		2/8
	PAL20R8A/-2/B		2/9
	PAL20R6A/-2/B		2/10
	PAL20R4A/-2/B		2/11
Medium 24X Exclusive-OR	PAL20L10		2/7
	PAL20X10		2/12
	PAL20X8		2/13
	PAL20X4		2/14
Medium 24XA Exclusive-OR	PAL20L10A		2/7
	PAL20X10A		2/15
	PAL20X8A		3/0
	PAL20X4A		3/1
Large 24RS Shared Product Terms	PAL20S10		3/5
	PAL20RS10		3/6
	PAL20RS8		3/7
	PAL20RS4		3/8
Large 24A Registered XOR	PAL22RX8A		Under Development
Large 24/A Varied XOR	PAL32VX10/A		↓
Large 24RA Asynchronous	PAL20RA10		3/4
ECL Combinatorial	PAL10H20P8	↓	Under Development
MegaPAL™ Devices	PAL32R16	Under Development	
	PAL64R32		

NOTE: The software and hardware revisions listed are the earliest revisions that support these products. Later software and hardware revisions can also be assumed to support these products.

Stag Microsystems

20 Pin Device Families

1600 Wyatt Dr. Suite 3
Santa Clara, CA 95054
(408) 988-1118

ZL30
PPZ

Series	Part Number	Code	ZL30 Rev.	ZM2200
Small 20/-2 Combinatorial	PAL10H8/-2	20-20	30-35	14
	PAL10L8/-2	20-25	↓	↓
	PAL12H6/-2	20-21	↓	↓
	PAL12L6/-2	20-26	↓	↓
	PAL14H4/-2	20-22	↓	↓
	PAL14L4/-2	20-27	↓	↓
	PAL16H2/-2	20-23	↓	↓
	PAL16L2/-2	20-28	↓	↓
	PAL16C1/-2	20-24	↓	↓
Medium 20/ 20A/20A-2/4/ 20B-2/-4 Standard	PAL16L8/A/-2/-4/B-2/-4	20-29	↓	↓
	PAL16R8/A/-2/-4/B-2/-4	20-30	↓	↓
	PAL16R6/A/-2/-4/B-2/-4	20-31	↓	↓
	PAL16R4/A-2/-4/B-2/-4	20-32	↓	↓
Medium 20B/D	PAL16L8B/D	22-29	30-39	12
	PAL16R8B/D	22-30	↓	↓
	PAL16R68/D	22-31	↓	↓
	PAL16R4B/D	22-32	↓	↓
Medium 20BP Standard	PAL16L8BP	20-29	30-35	14
	PAL16R8BP	20-30	↓	19
	PAL16R6BP	20-31	↓	↓
	PAL16R4BP	20-32	↓	↓
Medium 20PA Programmable Polarity	PAL16P8A	20-38	↓	12
	PAL16RP8A	20-11	↓	↓
	PAL16RP6A	20-12	↓	↓
	PAL16RP4A	20-13	↓	↓
Large 20 Arithmetic	PAL16X4	20-33	↓	14
	PAL16A4	20-34	↓	
Large 20RA Asynchronous	PAL16RA8	20-19	30-37	12

NOTE: The software and hardware revisions listed are the earliest revisions that support these products.
Later software and hardware revisions can also be assumed to support these products.

Stag Microsystems

24 Pin and MegaPAL™ Device Families

Series	Part Number	Code	ZL30 Rev.	ZM2200
Small 24 Combinatorial	PAL12L10	21-50	30-35	14
	PAL14L8	21-51	↓	
	PAL16L6	21-52	↓	12
	PAL18L4	21-53	↓	↓
	PAL20L2	21-54	↓	↓
	PAL20C1	21-55	↓	↓
Small 24A Decoder	PAL6L16A	Under Development		
	PAL8L14A			
Medium 24A/ 24A-2/B Standard	PAL20L8A/-2/B	21-56	30-35	12
	PAL20R8A/-2/B	21-57	↓	↓
	PAL20R6A/-2/B	21-58	↓	↓
	PAL20R4A/-2/B	21-59	↓	↓
Medium 24X Exclusive-OR	PAL20L10	21-60	↓	↓
	PAL20X10	21-61	↓	↓
	PAL20X8	21-62	↓	↓
	PAL20X4	21-63	↓	↓
Medium 24XA Exclusive-OR	PAL20L10A	21-60	↓	↓
	PAL20X10A	21-61	↓	↓
	PAL20X8A	21-62	↓	↓
	PAL20X4A	21-63	↓	↓
Large 24RS Shared Product Terms	PAL20S10	21-81	30-39	↓
	PAL20RS10	21-80	↓	↓
	PAL20RS8	21-79	↓	↓
	PAL20RS4	21-78	↓	↓
Large 24A Registered XOR	PAL22RX8A	Under Development		↓
Large 24/A Varied XOR	PAL32VX10/A	↓		↓
Large 24RA Asynchronous	PAL20RA10	21-77	30-37	↓
ECL Combinatorial	PAL10H20P8	Under Development		↓
MegaPAL™ Devices	PAL32R16	Under Development		
	PAL64R32			

NOTE: The software and hardware revisions listed are the earliest revisions that support these products.
Later software and hardware revisions can also be assumed to support these products.

Storey Systems

20 Pin Device Families

3201 North Hwy. 67
Suite H
Mesquite, TX 75150
(214) 270-4135

P240
Programmer

Series	Part Number	Rev.
Small 20/-2 Combinatorial	PAL10H8/-2	2.0
	PAL10L8/-2	
	PAL12H6/-2	
	PAL12L6/-2	
	PAL14H4/-2	
	PAL14L4/-2	
	PAL16H2/-2	
	PAL16L2/-2	
	PAL16C1/-2	
Medium 20/ 20A/20A-2/4/ 20B-2/-4 Standard	PAL16L8/A/-2/-4/B-2/-4	
	PAL16R8/A/-2/-4/B-2/-4	
	PAL16R6/A/-2/-4/B-2/-4	
	PAL16R4/A/-2/-4/B-2/-4	
Medium 20B/D	PAL16L8B/D	4.0
	PAL16R8B/D	
	PAL16R6B/D	
	PAL16R4B/D	
Medium 20BP Standard	PAL16L8BP	2.0
	PAL16R8BP	Under Development
	PAL16R6BP	
	PAL16R4BP	
Medium 20PA Programmable Polarity	PAL16P8A	4.0
	PAL16RP8A	
	PAL16RP6A	
	PAL16RP4A	
Large 20 Arithmetic	PAL16X4	2.0
	PAL16A4	
Large 20RA Asynchronous	PAL16RA8	4.04

NOTE: The software and hardware revisions listed are the earliest revisions that support these products.
Later software and hardware revisions can also be assumed to support these products.

Storey Systems

24 Pin and MegaPAL™ Device Families

Series	Part Number	Rev.
Small 24 Combinatorial	PAL12L10	2.0 ↓
	PAL14L8	
	PAL16L6	
	PAL18L4	
	PAL20L2	
	PAL20C1	
Small 24A Decoder	PAL6L16A	Under Development
	PAL8L14A	
Medium 24A/ 24A-2/B Standard	PAL20L8A/-2/B	2.0 ↓
	PAL20R8A/-2/B	
	PAL20R6A/-2/B	
	PAL20R4A/-2/B	
Medium 24X Exclusive-OR	PAL20L10	
	PAL20X10	
	PAL20X8	
	PAL20X4	
Medium 24XA Exclusive-OR	PAL20L10A	
	PAL20X10A	Under Development ↓
	PAL20X8A	
	PAL20X4A	
Large 24RS Shared Product Terms	PAL20S10	
	PAL20RS10	
	PAL20RS8	
	PAL20RS4	
Large 24A Registered XOR	PAL22RX8Aa	
Large 24/A Varied XOR	PAL32VX10/A	
Large 24RA Asynchronous	PAL20RA10	4.04
ECL Combinatorial	PAL10H20P8	Under Development
MegaPAL™ Devices	PAL32R16	
	PAL64R32	

NOTE: The software and hardware revisions listed are the earliest revisions that support these products.
Later software and hardware revisions can also be assumed to support these products.

Structured Design

20 Pin Device Families

1700 Wyatt Drive
Suite 7
Santa Clara, CA 95054
(408) 988-0725

SD 20/24 System

SD 1000 System

Series	Part Number	SD 20/24	SD 1000
Small 20/-2 Combinatorial	PAL10H8/-2	1.6	1.05
	PAL10L8/-2		
	PAL12H6/-2		
	PAL12L6/-2		
	PAL14H4/-2		
	PAL14L4/-2		
	PAL16H2/-2		
	PAL16L2/-2		
	PAL16C1/-2		
Medium 20/ 20A/20A-2/4/ 20B-2/-4 Standard	PAL16L8/A/-2/-4/B-2/-4		
	PAL16R8/A/-2/-4/B-2/-4		
	PAL16R6/A/-2/-4/B-2/-4		
	PAL16R4/A/-2/-4/B-2/-4	↓	
Medium 20B/D	PAL16L8B/D	Under Development	
	PAL16R8B/D		
	PAL16R6B/D		
	PAL16R4B/D		↓
Medium 20BP Standard	PAL16L8BP	1.6	1.05
	PAL16R8BP	Under Development	
	PAL16R6BP		
	PAL16R4BP		
Medium 20PA Programmable Polarity	PAL16P8A		
	PAL16RP8A		
	PAL16RP6A		
	PAL16RP4A		
Large 20 Arithmetic	PAL16X4	1.6	1.05
	PAL16A4		
Large 20RA Asynchronous	PAL16RA8	Under Development	

NOTE: The software and hardware revisions listed are the earliest revisions that support these products.
Later software and hardware revisions can also be assumed to support these products.

Structured Design

24 Pin and MegaPAL™ Device Families

Series	Part Number	SD 20/24	SD 1000
Small 24 Combinatorial	PAL12L10	1.6	1.05
	PAL14L8		
	PAL16L6		
	PAL18L4		
	PAL20L2		
	PAL20C1		
Small 24A Decoder	PAL6L16A	Under Development	
	PAL8L14A		
Medium 24A/ 24A-2/B Standard	PAL20L8A/-2/B	1.6	1.05
	PAL20R8A/-2/B		
	PAL20R6A/-2/B		
	PAL20R4A/-2/B		
Medium 24X Exclusive-OR	PAL20L10		
	PAL20X10		
	PAL20X8		
	PAL20X4		
Medium 24XA Exclusive-OR	PAL20L10A		
	PAL20X10A		
	PAL20X8A		
	PAL20X4A		
Large 24RS Shared Product Terms	PAL20S10	Under Development	
	PAL20RS10		
	PAL20RS8		
	PAL20RS4		
Large 24A Registered XOR	PAL22RX8A		
Large 24/A Varied XOR	PAL32VX10/A		
Large 24RA Asynchronous	PAL20RA10		
ECL Combinatorial	PAL10H20P8		
MegaPAL™ Devices	PAL32R16		
	PAL64R32		

NOTE: The software and hardware revisions listed are the earliest revisions that support these products. Later software and hardware revisions can also be assumed to support these products.

Valley Data Sciences

20 Pin Device Families

Charleston Business Park
2426 Charleston Road
Mountain View, CA 94043
(415) 968-2900

160 Series

Programmer

Series	Part Number	Rev.
Small 20/-2 Combinatorial	PAL10H8/-2	1.03
	PAL10L8/-2	
	PAL12H6/-2	
	PAL12L6/-2	
	PAL14H4/-2	
	PAL14L4/-2	
	PAL16H2/-2	
	PAL16L2/-2	
	PAL16C1/-2	
Medium 20/ 20A/20A-2/4/ 20B-2/-4 Standard	PAL16L8/A/-2/-4/B-2/-4	
	PAL16R8/A/-2/-4/B-2/-4	
	PAL16R6/A/-2/-4/B-2/-4	
	PAL16R4/A/-2/-4/B-2/-4	
Medium 20B/D	PAL16L8B/D	
	PAL16R8B/D	
	PAL16R6B/D	
	PAL16R4B/D	
Medium 20BP Standard	PAL16L8BP	↓
	PAL16R8BP	Under Development
	PAL16R6BP	
	PAL16R4BP	
Medium 20PA Programmable Polarity	PAL16P8A	1.03
	PAL16RP8A	
	PAL16RP6A	
	PAL16RP4A	
Large 20 Arithmetic	PAL16X4	
	PAL16A4	↓
Large 20RA Asynchronous	PAL16RA8	Under Development

NOTE: The software and hardware revisions listed are the earliest revisions that support these products.
Later software and hardware revisions can also be assumed to support these products.

Valley Data Sciences

24 Pin and MegaPAL^{TM} Device Families

Series	Part Number	Rev.
Small 24 Combinatorial	PAL12L10	1.03
	PAL14L8	
	PAL16L6	
	PAL18L4	
	PAL20L2	
	PAL20C1	
Small 24A Decoder	PAL6L16A	Under Development
	PAL8L14A	
Medium 24A/ 24A-2/B Standard	PAL20L8A/-2/B	1.03
	PAL20R8A/-2/B	
	PAL20R6A/-2/B	
	PAL20R4A/-2/B	
Medium 24X Exclusive-OR	PAL20X10	
	PAL20L10	
	PAL20X8	
	PAL20X4	
Medium 24XA Exclusive-OR	PAL20X10A	
	PAL20L10A	
	PAL20X8A	
	PAL20X4A	
Large 24RS Shared Product Terms	PAL20RS10	
	PAL20S10	
	PAL20RS8	
	PAL20RS4	
Large 24A Registered XOR	PAL22RX8A	
Large 24/A Varied XOR	PAL32VX10/A	
Large 24RA Asynchronous	PAL20RA10	
ECL Combinatorial	PAL10H20P8	Under Development
MegaPAL^{TM} Devices	PAL32R16	1.04 + 1.1 Adapter
	PAL64R32	Under Development

NOTE: The software and hardware revisions listed are the earliest revisions that support these products.
Later software and hardware revisions can also be assumed to support these products.

Varix

20 Pin Device Families

1210 E. Campbell Road
Richardson, TX 75081
(214) 437-0777

OMNI

Programmer

Series	Part Number	Rev.
Small 20/-2 Combinatorial	PAL10H8/-2	3.18
	PAL10L8/-2	↓
	PAL12H6/-2	↓
	PAL12L6/-2	↓
	PAL14H4/-2	↓
	PAL14L4/-2	↓
	PAL16H2/-2	↓
	PAL16L2/-2	↓
	PAL16C1/-2	↓
Medium 20/ 20A/20A-2/4/ 20B-2/-4 Standard	PAL16L8/A/-2/-4/B-2/-4	↓
	PAL16R8/A/-2/-4/B-2/-4	↓
	PAL16R6/A/-2/-4/B-2/-4	↓
	PAL16R4/A/-2/-4/B-2/-4	↓
Medium 20B/D	PAL16L8B/D	↓
	PAL16R8B/D	↓
	PAL16R6B/D	↓
	PAL16R4B/D	↓
Medium 20BP Standard	PAL16L8BP	3.18
	PAL16R8BP	5.00
	PAL16R6BP	↓
	PAL16R4BP	↓
Medium 20PA Programmable Polarity	PAL16P8A	3.18
	PAL16RP8A	↓
	PAL16RP6A	↓
	PAL16RP4A	↓
Large 20 Arithmetic	PAL16X4	↓
	PAL16A4	↓
Large 20RA Asynchronous	PAL16RA8	Under Development

NOTE: The software and hardware revisions listed are the earliest revisions that support these products.
Later software and hardware revisions can also be assumed to support these products.

Varix

24 Pin and MegaPAL™ Device Families

Series	Part Number	Rev.
Small 24 Combinatorial	PAL12L10	3.18
	PAL14L8	
	PAL16L6	
	PAL18L4	
	PAL20L2	
	PAL20C1	
Small 24A Decoder	PAL6L16A	
	PAL8L14A	
Medium 24A/ 24A-2/B Standard	PAL20L8A/-2/B	
	PAL20R8A/-2/B	
	PAL20R6A/-2/B	
	PAL20R4A/-2/B	
Medium 24X Exclusive-OR	PAL20L10	
	PAL20X10	
	PAL20X8	
	PAL20X4	
Medium 24XA Exclusive-OR	PAL20L10A	
	PAL20X10A	
	PAL20X8A	
	PAL20X4A	
Large 24RS Shared Product Terms	PAL20S10	Under Development
	PAL20RS10	
	PAL20RS8	
	PAL20RS4	
Large 24A Registered XOR	PAL22RX8A	
Large 24/A Varied XOR	PAL32VX10/A	
Large 24RA Asynchronous	PAL20RA10	3.18
ECL Combinatorial	PAL10H20P8	Under Development
MegaPAL™ Devices	PAL32R16	
	PAL64R32	

NOTE: The software and hardware revisions listed are the earliest revisions that support these products. Later software and hardware revisions can also be assumed to support these products.

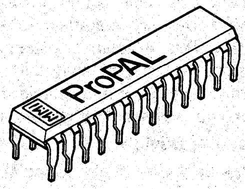
Table of Contents

HAL/ZHAL Devices

So you have discovered the convenience and flexibility of designing with PAL® devices from Monolithic Memories. You have implemented a design using PAL devices, and taken that design into production. Now may be the time to consider ways of reducing the efforts you put into programming, testing, and marking large volumes of PAL devices. Wouldn't it be more convenient if you could be relieved of the duties and costs of volume programming and testing and still reap the benefits afforded by programmable logic?

Or perhaps you are considering a semicustom product, but you're a little nervous about going to a gate array. Wouldn't it be preferable if you could find a semicustom product which allows easy, inexpensive prototyping, provides fast prototype turnaround, comes fully tested, can have a custom marking, has low NRE charges, provides design flexibility, and has an assured second source?

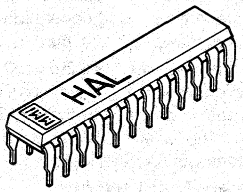
Well, Monolithic Memories, the inventor of the PAL device, offers the logical solutions. ProPAL, HAL, and ZHAL devices make the transition from user-programmed devices to customized production-ready devices easy and risk free.



CD01010M

ProPAL Devices

ProPAL (Programmed PAL) devices are simply PAL devices that Monolithic Memories programs and tests for you. You receive a fully functional unit without having to do the programming and testing. But you still have the flexibility to handle design changes easily.

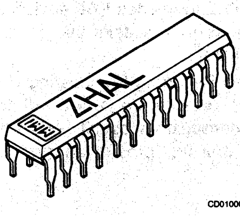


CD01020M

HAL Devices

HAL (Hard Array Logic) devices are to PAL devices as ROMs are to PROMs. Instead of fuses in the logic array, your pattern is implemented using metal links that are masked in during wafer fabrication. So your need to program devices is eliminated. And because the devices have their functionality masked in, Monolithic Memories can provide full functional testing before shipping the product. You can place the devices in your boards with a minimum of handling and the highest level of confidence.

Monolithic Memories offers a HAL device for every PAL device. Any PAL device design you program can be implemented in a HAL version, allowing you to move smoothly into volume production.



CD01000M

ZHAL Devices

Monolithic Memories now provides a third alternative for the programmable logic user: new Zero-Standby-Power CMOS HAL devices.

For the first time there are HAL devices which can implement any pattern from the Series 20 and Series 24 PAL devices with the greatly reduced power consumption only CMOS can offer.

For high complexity designs reaching into the thousands of gates, Zero Power MegaHAL™ devices provide the natural semicustom VLSI alternative to gate arrays. The MegaPAL™ devices provide the flexibility and fast design turn-around you need for prototyping. Once you are ready for production, the CMOS MegaHAL devices provide the same functionality with Zero Power.

All of the ZHAL devices are fully HC/HCT compatible, making them easy to use in TTL and CMOS environments.

Should You Use a PAL, ProPAL, or HAL Device?

PAL devices offer the flexibility and convenience needed for prototyping your innovative designs. They provide a means for designing an efficient system by integrating functions and saving you board space. For flexible production, it makes sense to program and test your own PAL devices. This is especially true if you need low volumes per pattern. You always have the option of making last minute design tweaks as you fine tune your system design.

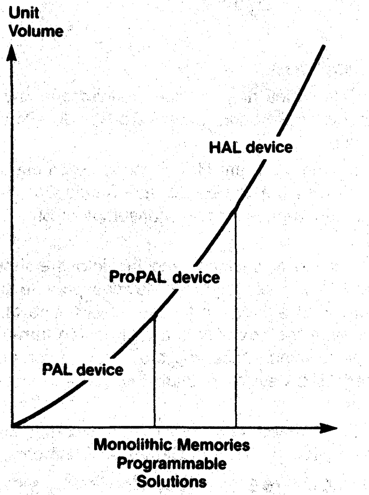
Once your production volumes reach a moderate volume of a few thousand devices per year for each pattern, you may wish to dedicate your production resources to newer designs, instead of programming and testing production volumes. Yet in order to be able to make quick design updates, you might not want to commit to a HAL mask. ProPAL devices provide the ideal solution by eliminating programming and testing needs while retaining enough flexibility to accommodate design changes.

When you feel that your design has stabilized and your volume has ramped up to several thousand devices per year, a HAL device becomes the most cost effective way to purchase your programmable logic. By shifting the burdens to Monolithic Memories, who can handle large volumes easily, you can concentrate your energies on more productive projects.

How Does MMI Do This?

Monolithic Memories takes your proven PAL device design and either arranges to program ProPAL devices in volume, or generates a custom mask for a HAL or ZHAL device. And all without the normal risks inherent in purchasing a semicustom product. Why? Because:

- You can prototype your system and initiate production using standard Monolithic Memories PAL devices. You don't have to worry about making a mistake that could put your design schedule in jeopardy.
- The nominal Non-Recurring Engineering (NRE) charges for ProPAL and HAL devices are far lower than those normally required for a semicustom circuit. And they can even be amortized over your first production quantity.



OP00010M

- You save on the costs of programming devices. This will also shorten your production cycle, since you can plug the devices into the socket with no additional processing.
- All of the devices are tested for full functionality before they leave Monolithic Memories. You save on the costs of testing and generating test programs.
- Monolithic Memories is geared towards providing volumes of high quality devices. No one knows how to test programmable logic as well as Monolithic Memories. Between the thorough, efficient testing and marking capabilities and the option to provide burn-in for extra reliability, you can obtain a higher quality device that if you did the programming and testing yourself.
- MMI can provide custom marking. This saves you the added expense of stripping the mark from standard devices and then remarking them with your own mark.
- HAL devices are secure by design. If you prefer ProPAL devices, they can also be secured for you at the factory.
- ProPAL device lead time is only 1 to 2 weeks longer than that of unprogrammed PAL devices.
- HAL device turn-around time is a mere 6 to 8 weeks or less from acceptance of your design package to receipt of first units.
- If you find yourself with an unexpected demand, you need not turn away business for lack of HAL device stock. You can always use ProPAL devices to make up for any temporary shortfall.

How Can You Take Advantage of This?

The following are some guidelines which you can use to help convert your designs to ProPAL, HAL or ZHAL devices.

1. Send in Your Design

You will need to provide your logic equations from either PALASM®, PALASM 2 or ABEL™ on magnetic media*.

When Monolithic Memories generates vectors for use in functionally testing your pattern, "seed" vectors are helpful in providing the foundation upon which the final test vectors will be based.

A master PAL device containing your design is needed for Monolithic Memories to verify that the pattern you submitted has been correctly processed. If you cannot provide a Monolithic Memories master PAL device, Monolithic Memories will accept your design inputs and provide ProPAL samples for your approval.

For your convenience, a checklist is included to help you prepare all of the necessary materials to be submitted to Monolithic Memories. This will also help Monolithic Memories process your design, resulting in smoother and faster turn-around. Copies of this form are available from your Sales Representative, or you can simply copy the attached form.

2. MMI Will Verify the Design

Upon receiving your design package, Monolithic Memories will enter your design into their computer and verify that there are no format or syntax problems. A fuse map will be generated, and sample ProPAL devices programmed.

If any questions are encountered at this stage, they will be resolved with you before any further processing takes place.

3. MMI Will Check the Samples

If you have approved immediate production of your devices, Monolithic Memories will make a fuse for fuse comparison between the samples and the master device you provide. If there are no discrepancies, test generation will be started immediately (or upon receipt of your purchase order).

If you prefer to see programmed sample ProPAL devices prior to initiating production, Monolithic Memories can provide them for your approval before proceeding further. Sample approval is also needed when no master devices are provided or when a discrepancy is found during verification.

4. MMI Will Generate Test Vectors

A functional test sequence is generated using TGEN™, a proprietary software package. Any seed vectors you provide will be used to help initiate test generation. TGEN will check for hazards and race conditions, monitor fault coverage and systematically add vectors until test coverage goals are met.

Monolithic Memories has a test quality standard that sets as a minimum goal 90% coverage of all stuck-at faults. Lower coverage patterns can sometimes be processed as HAL devices, or it is possible to handle them as ProPAL devices only, but your approval will be needed. If acceptable coverage cannot be obtained, ways of increasing the testability of the design may have to be considered before Monolithic Memories can process the pattern.

*Floppy disks are accepted in standard DEC® AT-11 (RX01) or RSX-11M®, files 11 format, or an IBM PC™ 5-1/4 in. diskette. IBM compatible (800 or 1600 BPI) nine track magnetic tapes are accepted in unlabeled (card image), files-11, or VAX VMS® backup format. If magnetic media absolutely cannot be provided, legible printouts (signed and dated) from PALASM will be accepted. Please note that magnetic media are required if you have more than 50 vectors.

For more detail on exactly how the test coverage is determined, refer to the article "PAL Design Function and Test Vectors" in this Handbook.

When suitable test coverage is obtained, as is normally the case, there is no need for you to be involved with vector generation. If, however, you wish to approve the test vectors before production units are generated, the vectors will be made available to you.

5. MMI Will Generate Production Units

After an acceptable test sequence has been generated, Monolithic Memories will generate the appropriate HAL or ZHAL mask and build the devices. Or, in the case of a ProPAL device, Monolithic Memories will arrange to program and test blank units.

Having Your Devices Marked

The standard Monolithic Memories mark consists of the device type, the package type, the date code, and the Monolithic Memories logo.

If you wish, you can have the standard marking replaced by a custom marking. The logo and date code are standard, but any other markings can be as you desire. The character and line limitations for the most common packages are in Table 1.

If the package you want is not listed, your local representative can help you determine the guidelines you need.

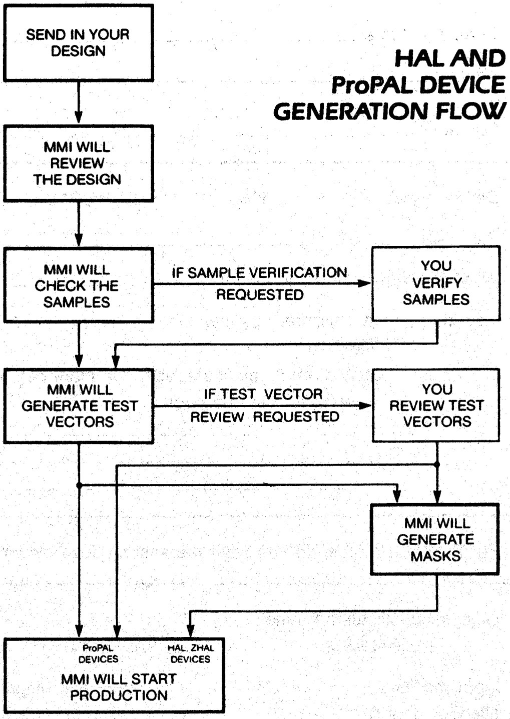
Whom to Contact

When you are ready to put the power of the Monolithic Memories factory to work for you, just contact your local Monolithic Memories sales representative. And let Monolithic Memories take care of your production programming, testing, and marking needs.

PLASTIC	20 pin (300 mil)	2 lines/13 characters per line
	24 pin (300 mil)	2 lines/17 characters per line
CERDIP	20 pin (300 mil)	2 lines/16 characters per line
	24 pin (300 mil)	2 lines/17 characters per line
PLCC	20 lead	4 lines/11 characters per line
	28 lead	5 lines/12 characters per line

Table 1

HAL AND ProPAL DEVICE GENERATION FLOW


SEND IN YOUR DESIGN
MMI WILL REVIEW THE DESIGN
MMI WILL CHECK THE SAMPLES
IF SAMPLE VERIFICATION REQUESTED
YOU VERIFY SAMPLES
MMI WILL GENERATE TEST VECTORS
IF TEST VECTOR REVIEW REQUESTED
YOU REVIEW TEST VECTORS
MMI WILL GENERATE MASKS
ProPAL DEVICES
HAL, ZHAL DEVICES
MMI WILL START PRODUCTION

BD00310M

PAL, HAL and PALASM are registered trademarks of Monolithic Memories, Inc.
ProPAL, ZHAL, MegaPAL, MegaHAL and TGEN are trademarks of Monolithic Memories, Inc.

ABEL is a trademark of Data I/O Corp.

DEC, RSX, VAX, and VMS are registered trademarks of Digital Equipment Corp.

IBM PC is a trademark of International Business Machines, Inc.

HAL/ProPAL CHECKLIST

Company name: ______________________________

Address: ______________________________

____

Contact: ______________________________ Phone Number: ______________

Do you want: ___ HAL ___ ProPAL ___ ZHAL

Part Type: ______________________________ Customer Part Number: ______________

What package type do you want? ______________________________

How do you want the devices marked?

___ Standard

___ Custom mark: please specify the mark below.

Refer to previous page

____

____

____

For ProPAL devices, do you want the security fuse blown? ______________________________

Design Specification Format:

___ PALASM ___ PALASM 2 ___ ABEL

Input medium:
(Choose 1 of 3)

1. 9-track Magnetic Tape
   - ___ card image
   - ___ files-11
   - ___ VAX VMS backup
2. Floppy Disk
   - ___ RT-11
   - ___ RSX-11M
   - ___ IBM PC
3. ___PALASM printout (signed and dated)

The following items are requested but not required. Please check if provided:

___ Seed vectors

___ Master PAL device

OPTIONS

A. ___ I want to start production immediately (or upon submittal of purchase order) if:

1. Design is acceptable;
2. MMI samples match my master device fuse for fuse;
3. Minimum test coverage goals are met.

(Master device must be provided.)

B. ___ I want to verify the MMI generated sample devices prior to production implementation.

___ I want to approve the test vectors prior to production implementation.

Please complete this form for each pattern submitted to Monolithic Memories, and include it in your design package.

Submitted by: ______________________________ Date: ______________

Title: ______________________________

Zero Power CMOS Hard Array Logic ZHAL™ 20A Series

Features/Benefits

- Zero standby power
- 25-ns maximum propagation delay
- HC and HCT compatible
- Space saving PLCC available
- Low power alternative for Small and Medium 20-pin PAL® devices, including 16L8/16R8/16R6/16R4

Description

The Zero Hard Array Logic (ZHAL) devices are ideal in low-power applications that require high-speed operation. These attributes are achieved through the use of Monolithic Memories' advanced high-speed CMOS process. Now system designers have the option of using a ZHAL device that matches fast PAL device speeds, but with the added advantage of zero standby power. These features are ideal for power-critical areas such as portable digital equipment or lap-top computers.

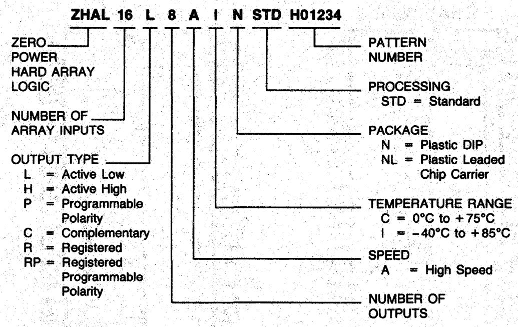
This family of ZHAL devices utilizes a unique architecture that is designed for a high degree of flexibility in implementing most patterns of the listed 20-pin PAL/HAL® devices. Prototyping can be done using standard PAL devices before converting to ZHAL circuits for production. ZHAL devices are fabricated by Monolithic Memories with custom metallization masks defined by a user-supplied HAL Design Specification.

The ZHAL option in the PALASM®2 CAD package will confirm whether a design specification will fit within the ZHAL20 architecture. For more information on the ZHAL software, refer to the PALASM 2 User Manual.

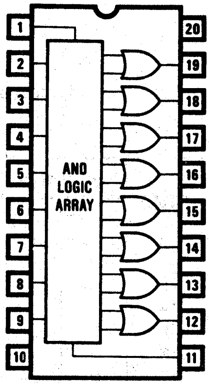
For evaluation of the ZHAL20A circuit, sample patterns are available. See page 4-13 for details.

Ordering Information

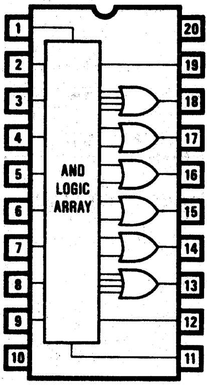
PART NUMBER	PACKAGE	ARRAY	OUTPUTS	
			COMB	REG
ZHAL10H8A	N, NL	10	8	—
ZHAL12H6A		12	6	—
ZHAL14H4A		14	4	—
ZHAL16H2A		16	2	—
ZHAL16C1A		16	2	—
ZHAL10L8A		10	8	—
ZHAL12L6A		12	6	—
ZHAL14L4A		14	4	—
ZHAL16L2A		16	2	—
ZHAL16L8A	N, NL	16	8	—
ZHAL16R8A		16	—	8
ZHAL16R6A		16	2	6
ZHAL16R4A		16	4	4
ZHAL16P8A	N, NL	16	8	—
ZHAL16RP8A		16	—	8
ZHAL16RP6A		16	2	6
ZHAL16RP4A		16	4	4



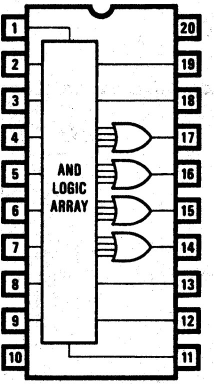
ZHAL10H8A



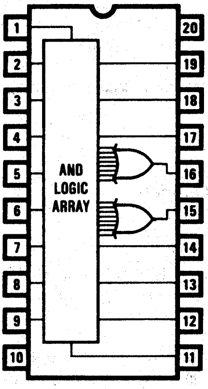
ZHAL12H6A



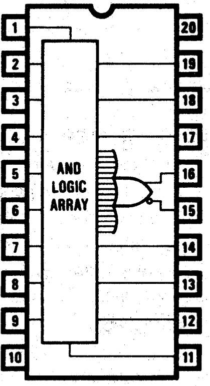
ZHAL14H4A



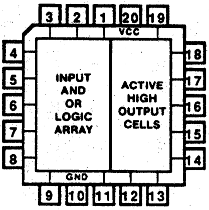
ZHAL16H2A



ZHAL16C1A

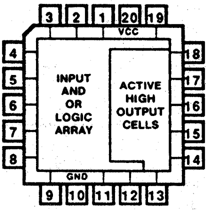


ZHAL10H8A



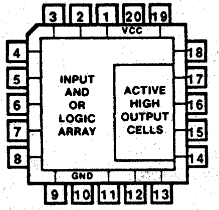
Plastic Chip Carrier

ZHAL12H6A



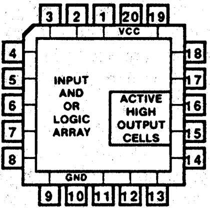
Plastic Chip Carrier

ZHAL14H4A



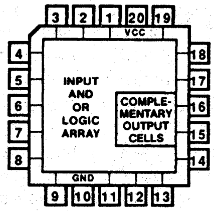
Plastic Chip Carrier

ZHAL16H2A



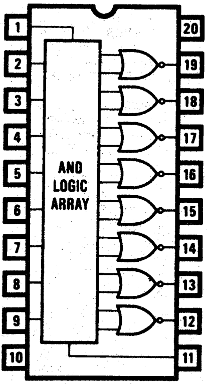
Plastic Chip Carrier

ZHAL16C1A

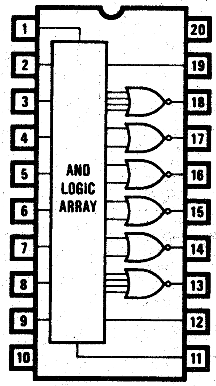


Plastic Chip Carrier

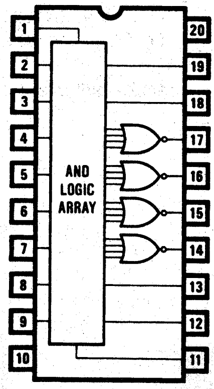
ZHAL10L8A



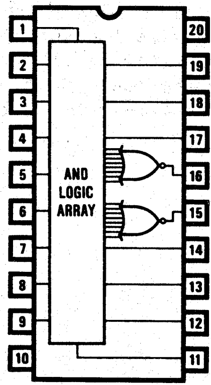
ZHAL12L6A



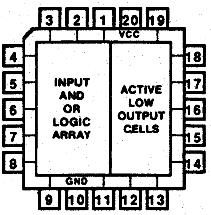
ZHAL14L4A



ZHAL16L2A

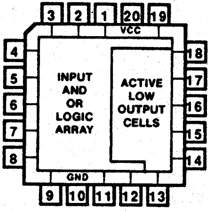


ZHAL10L8A



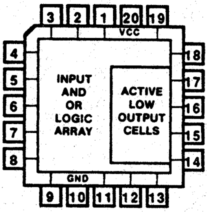
Plastic Chip Carrier

ZHAL12L6A



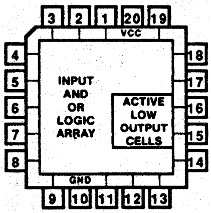
Plastic Chip Carrier

ZHAL14L4A



Plastic Chip Carrier

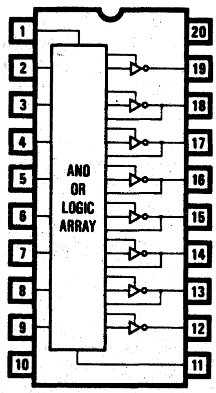
ZHAL16L2A



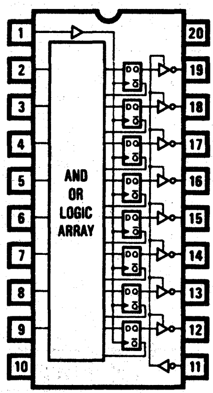
Plastic Chip Carrier

CD01030M

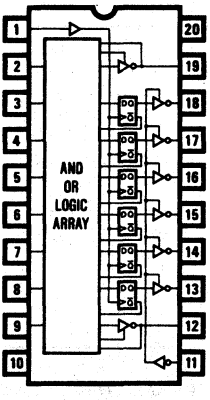
ZHAL16L8A



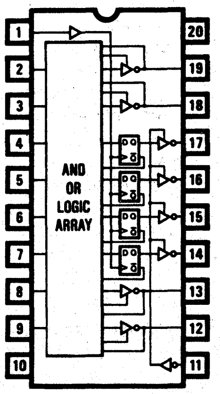
ZHAL16R8A



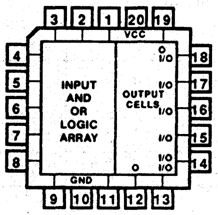
ZHAL16R6A



ZHAL16R4A

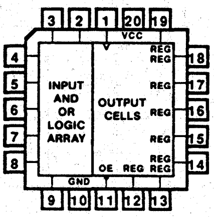


ZHAL16L8A



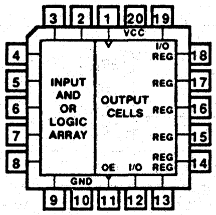
Plastic Chip Carrier

ZHAL16R8A



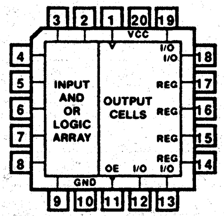
Plastic Chip Carrier

ZHAL16R6A



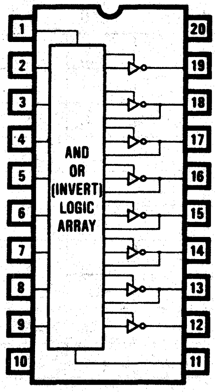
Plastic Chip Carrier

ZHAL16R4A

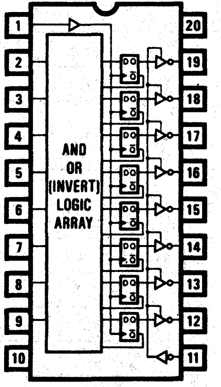


Plastic Chip Carrier

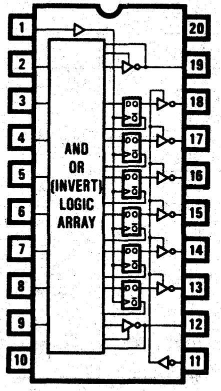
ZHAL16P8A



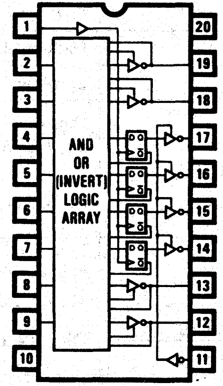
ZHAL16RP8A



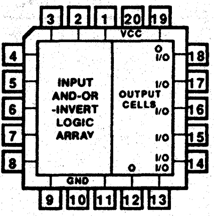
ZHAL16RP6A



ZHAL16RP4A

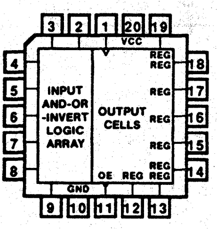


ZHAL16P8A



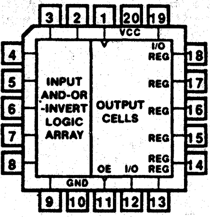
Plastic Chip Carrier

ZHAL16RP8A



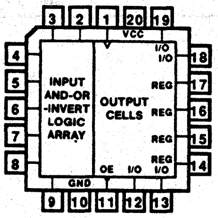
Plastic Chip Carrier

ZHAL16RP6A



Plastic Chip Carrier

ZHAL16RP4A



Plastic Chip Carrier

CD01040M

Absolute Maximum Ratings

Supply voltage, V_{CC}	−0.5V to 7V
DC input voltage, V_I	−0.5V to V_{CC} +0.5V
DC output voltage, V_O	−0.5V to V_{CC} +0.5V
DC output source/sink current per output pin, I_O	± 35mA
DC V_{CC} or ground current, I_{CC} or I_{GND}	± 100mA
Input diode current, I_{IK}:	
$V_I < 0$	−20mA
$V_I > V_{CC}$	+20mA
Output diode current, I_{OK}:	
$V_O < 0$	−20mA
$V_O > V_{CC}$	+20mA
Storage temperature	−65°C to 150°C

Operating Conditions

SYMBOL	PARAMETER		INDUSTRIAL			COMMERCIAL			UNIT
			MIN	TYP	MAX	MIN	TYP	MAX	
V_{CC}	Supply voltage		4.5	5	5.5	4.75	5	5.25	V
t_w	Width of clock	16R4A, 16R6A, 16R8A, 16RP4A, 16RP6A, 16RP8A	15	10		15	10		ns
t_{su}	Setup time from input or feedback to clock		20	13		20	13		ns
t_h	Hold time		0	−10		0	−10		ns
t_A	Operating free-air temperature		−40	25	85	0	25	75	°C

Electrical Characteristics Over Operating Conditions

SYMBOL	PARAMETER		TEST CONDITIONS		MIN	TYP	MAX	UNIT
V_{IL}[1]	Low-level input voltage				0		0.8	V
V_{IH}[1]	High-level input voltage				2		V_{CC}	V
I_{IL}	Low-level input current		V_{CC} = MAX	V_I = GND			−1	µA
I_{IH}	High-level input current	Pin 8[2]	V_{CC} = MAX	V_I = GND		1	10	µA
		All other pins					1	µA
V_{OL}	Low-level output voltage		V_{CC} = MIN	I_{OL} = 8mA		0.1	0.4	V
			V_{CC} = 5V	I_{OL} = 1µA			0.05	
V_{OH}	High-level output voltage		V_{CC} = MIN	I_{OH} = −6mA	3.76[3]	4.1		V
			V_{CC} = 5V	I_{OH} = −1µA	4.95			
I_{OZL}[4]	Off-state output current		V_{CC} = MAX	V_O = GND		0	−10	µA
I_{OZH}[4]				V_O = V_{CC}		0	10	µA
I_{CC}	Standby supply current[5]		I_O = 0mA, V_I = GND or V_{CC}			0	100	µA
	Operating supply current		f = 1MHz, I_O = 0mA, V_I = GND or V_{CC}			2	5[6]	mA

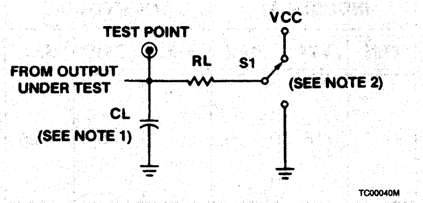
Switching Characteristics Over Operating Conditions

SYMBOL	PARAMETER		TEST CONDITIONS (See Test Load)	INDUSTRIAL			COMMERCIAL			UNIT
				MIN	TYP	MAX	MIN	TYP	MAX	
t_{PD}	Input or feedback to output 10H8A, 12H6A, 14H4A, 16H2A, 16C1A, 10L8A, 12L6A, 14L4A, 16L2A, 16L8A, 16R6A, 16R4A, 16P8A, 16RP6A, 16RP8A		R_L = 1KΩ C_L = 50pF		15	25		15	25	ns
t_{CLK}	Clock to output or feedback 16R4A, 16R6A, 16R8A, 16RP4A, 16RP6A, 16RP8A				10	15		10	15	ns
t_{PZX}	Input to output enable	16L8A, 16R4A, 16R6A, 16P8A, 16RP4A, 16RP6A			12	25		12	25	ns
t_{PXZ}[7]	Input to output disable				14	25		14	25	ns
t_{PZX}[7] t_{PZX}	Pin 11 to output disable/enable	16R4A, 16R6A, 16R8A, 16RP4A, 16RP6A, 16RP8A			12	15		12	15	ns
f_{MAX}	Maximum frequency			28.5	40		28.5	40		MHz

Notes: Apply to electrical and switching characteristics.

1. These are absolute voltages with respect to the ground pin on the device and include all overshoots due to system and/or tester noise. Do not attempt to test these values without suitable equipment.
2. Pin 8 (PRELOAD pin). Applies to all devices whether registered or non-registered.
3. JEDEC standard no. 7 for high-speed CMOS devices.
4. Applies to pins 12 to 19.
5. Disable output pins = V_{CC} or GND.
6. Add 3mA per additional 1.0MHz of operation over 1MHz.
7. C_L = 5pF.

Switching Test Load



Notes: 1. CL includes probe and jig capacitance.

2. When measuring t_{PLZ} and t_{PZL}, S1 is tied to V_{CC}. When measuring t_{PHZ} and t_{PZH}, S1 is tied to ground. t_{PZX} is measured with C_L = 50pF. t_{PXZ} is measured with C_L = 5pF. When measuring propagation delay times of 3-state outputs, S1 is open, i.e., not connected to V_{CC} or ground.

3. Waveform 1 is for an output with internal conditions such that the output is Low except when disabled by the output control. Waveform 2 is for an output with internal conditions such that the output is High except when disabled by the output control.

Output Register PRELOAD†

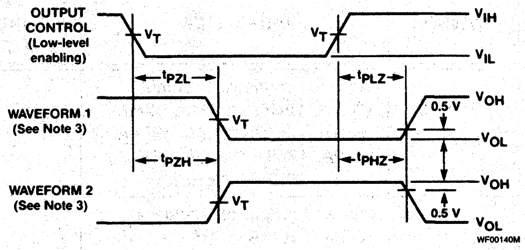
The PRELOAD function allows the register to be loaded from data placed on the output pins. This feature aids functional testing of state sequencer designs by allowing direct setting of output states for improved test coverage. The procedure for PRELOAD is as follows:

1. Raise V_{CC} to 4.5V.
2. Disable output registers by setting pin 11 to V_{IH}. Set pin 1 to 0V.
3. Apply V_{IL}/V_{IH} to all registered output pins.
4. Pulse pin 8 to V_p (12V), then back to 0V.
5. Remove V_{IL}/V_{IH} from all registered output pins.
6. Lower pin 11 to V_{IL} to enable the output registers.
7. Verify for V_{OL}/V_{OH} at all registered output pins.

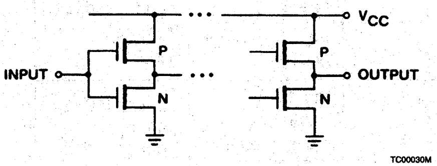
† Note: Only applies to parts with output registers.

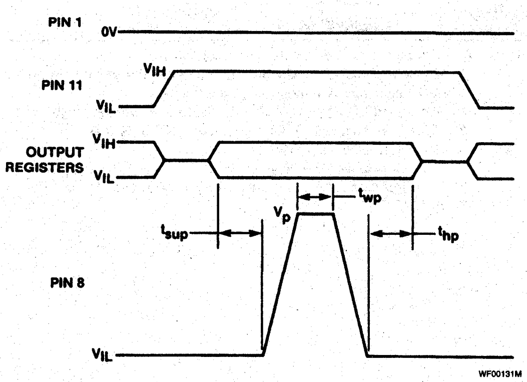
Typical	t_{sup}	= 50ns
	t_{wp}	= 100ns
	t_{hp}	= 50ns
	I_{IH}	= 30μA (Pin 8)

Enable/Disable Delay



Schematic of Inputs and Outputs





ZHAL20A Evaluation #3

Features/Benefits

- Demonstrates all features of ZHAL20A product
- 4-bit up/down counter with reset
- 3-bit shifter
- 25-ns maximum propagation delay
- Zero standby power

Description

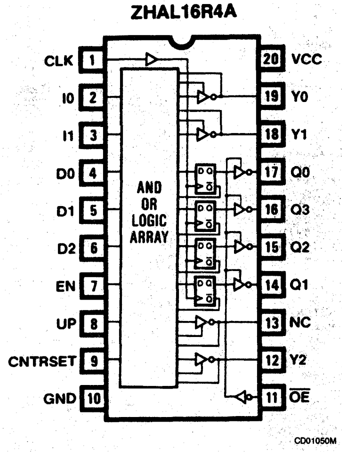
The ZHAL20A Evaluation Pattern is provided as an example of the features and characteristics of the ZHAL20A Series products.

This design consists of two functionally independent patterns: a 4-bit up/down counter and a 3-bit shifter. The 4-bit counter can count up or count down and has reset capability. These features are controlled by two control signals: UP and CNTRSET (Count Reset). When UP is high, the counter counts up. When UP is low, the counter counts down. CNTRSET overrides the count function and resets the counter to all ones, synchronous with the clock.

The 3-bit shifter shifts data bits by 0, 1 or 2 positions. The three bits of the shifter are enabled when EN (enable) is high, and are disabled (high-Z) when EN is low.

The PALASM®2 software file and simulation results are shown on the next page. Below are the function tables that summarize the functions of the counter and the shifter.

Logic Symbol



Counter Function Table

$\overline{\text{OE}}$	UP	CNTRSET	CLK	Q3-Q0	OPERATION
H	X	X	X	Z	High-Z
L	H	L	↑	Q plus 1	Increment
L	L	L	↑	Q minus 1	Decrement
L	X	H	↑	High	Reset

H=HIGH voltage level
L=LOW voltage level
X=Don't care
Z=High impedance (off) state
↑=LOW-to-HIGH clock transition

Shifter Function Table

EN	I1	I0	Y2	Y1	Y0	OPERATION
L	X	X	Z	Z	Z	High-Z
H	L	L	$\overline{D2}$	$\overline{D1}$	$\overline{D0}$	No operation
H	L	H	$\overline{D0}$	$\overline{D2}$	$\overline{D1}$	Shift by one
H	H	L	$\overline{D1}$	$\overline{D0}$	$\overline{D2}$	Shift by two

PALASM Design Specification

```
TITLE    PDS CONVERSION FILE
PATTERN  EXAMPLE
REVISION 1.00
AUTHOR   JOHN DOE
COMPANY  MONOLITHIC MEMORIES, INC
DATE     9/23/85

CHIP zzz PAL16RP4 CLK I0 I1 D0 D1 D2 EN UP CNTRSET GND
                  /OE Y2 NC Q1 Q2 Q3 Q0 Y1 Y0 VCC

EQUATIONS

 Y0 = /I1*/I0*/D0
    + /I1* I0*/D1
    +  I1*/I0*/D2
 Y0.TRST = EN

 Y1 = /I1*/I0*/D1
    + /I1* I0*/D2
    +  I1*/I0*/D0
 Y1.TRST = EN

 Y2 = /I1*/I0*/D2
    + /I1* I0*/D0
    +  I1*/I0*/D1
 Y2.TRST = EN


Q0 :=/Q0
    + CNTRSET

Q1 :=/Q1* Q0* UP
    + Q1* Q0*/UP
    + Q1*/Q0* UP
    +/Q1*/Q0*/UP
    + CNTRSET

Q2 := CNTRSET
    +/Q2*/Q1*/Q0*/UP
    + Q2*/Q1*     UP
    + Q2* Q1*/Q0
    +/Q2* Q1* Q0* UP
    + Q2*     Q0*/UP

Q3 := CNTRSET
    + Q3*         Q0*/UP
    +/Q3*/Q2*/Q1*/Q0*/UP
    +/Q3* Q2* Q1* Q0* UP
    + Q3* Q2*/Q1
    + Q3*    /Q1*     UP
    + Q3*/Q2* Q1
    + Q3*     Q1*/Q0
```

Simulation Results

```
PALASM SIMULATION HISTORY LISTING

ZZZ
Page :  1
         g g  cg cg  cg cgcgcg cgcgcgcgcg cgcgcgcgc
 CLK     LLLLHLLHLL HLLHLHLHLH LHLHLHLHLH LHLHLHLHL
 I0      XXLLLLLLLH HHLLLLLLLL LLLLLLLLLL LLLLLLLLL
 I1      XXLLLLLLLL LLHHHHHHHH HHHHHHHHHH HHHHHHHHH
 D0      XXLLLLLLLH HHLLLLLLLL LLLLLLLLLL LLLLLLLLL
 D1      XXLLLLLLLL LLHHHHHHHH HHHHHHHHHH HHHHHHHHH
 D2      XXLLLLLLLH HHLLLLLLLL LLLLLLLLLL LLLLLLLLL
 EN      LLHHHHHHHH HHHHHHHHHH HHHHHHHHHH HHHHHHHHH
 UP      XXXXXXHHHH HHHHHHHHHH HHHHHHHHHH HHHHHHHHH
 CNTRSET XXHHHHLLLL LLLLLLLLLL LLLLLLLLLL LLLLLLLLL
 GND     LLLLLLLLLL LLLLLLLLLL LLLLLLLLLL LLLLLLLLL
/OE      HHLLLLLLLL LLLLLLLLLL LLLLLLLLLL LLLLLLLLL
 Y2      XZHHHHHHHL LLLLLLLLLL LLLLLLLLLL LLLLLLLLL
 Q1      ZZXXXHHHLL LLLLHHHHLL LLHHHHLLLL HHHHLLLLH
 Q2      ZZXXXHHHLL LLLLLLLLHH HHHHHHLLLL LLLLHHHHH
 Q3      ZZXXXHHHLL LLLLLLLLLL LLLLLLHHHH HHHHHHHHH
 Q0      ZZXXXHHHLL LHHHLLHHLL HHLLHHLLHH LLHHLLHHL
 Y1      XZHHHHHHHL LLHHHHHHHH HHHHHHHHHH HHHHHHHHH
 Y0      XZHHHHHHHH HHHHHHHHHH HHHHHHHHHH HHHHHHHHH
 VCC     HHHHHHHHHH HHHHHHHHHH HHHHHHHHHH HHHHHHHHH
```

Simulation File

```
SIMULATION

TRACE_ON CLK I0 I1 D0 D1 D2 EN UP CNTRSET /OE Y2 Q1 Q2 Q3 Q0 Y1 Y0

SETF /CLK /OE /EN

SETF OE EN /I1 /I0 /D2 /D1 /D0 Y2 Y1 Y0 CNTRSET
CLOCKF CLK

CHECK Q3 Q2 Q1 Q0

SETF /I1 /I0 /D2 /D1 /D0 Y2 Y1 Y0 /CNTRSET UP
CLOCKF CLK

CHECK /Q3 /Q2 /Q1 /Q0

SETF /I1 I0 D2 /D1 D0 /Y2 /Y1 Y0
CLOCKF CLK

CHECK /Q3 /Q2 /Q1 Q0

SETF I1 /I0 /D2 D1 /D0 /Y2 Y1 Y0
CLOCKF CLK

CHECK /Q3 /Q2 Q1 /Q0

SETF OE /CNTRSET UP
CLOCKF CLK

CHECK /Q3 /Q2 Q1 Q0

SETF OE /CNTRSET UP
CLOCKF CLK

CHECK /Q3 Q2 /Q1 /Q0

SETF OE /CNTRSET UP
CLOCKF CLK

CHECK /Q3 Q2 /Q1 Q0

SETF OE /CNTRSET UP
CLOCKF CLK

CHECK /Q3 Q2 Q1 /Q0

SETF OE /CNTRSET UP
CLOCKF CLK

CHECK /Q3 Q2 Q1 Q0

SETF OE /CNTRSET UP
CLOCKF CLK

CHECK Q3 /Q2 /Q1 /Q0

SETF OE /CNTRSET UP
CLOCKF CLK

CHECK Q3 /Q2 /Q1 Q0

SETF OE /CNTRSET UP
CLOCKF CLK

CHECK Q3 /Q2 Q1 /Q0

SETF OE /CNTRSET UP
CLOCKF CLK

CHECK Q3 /Q2 Q1 Q0

SETF OE /CNTRSET UP
CLOCKF CLK

CHECK Q3 Q2 /Q1 /Q0

SETF OE /CNTRSET UP
CLOCKF CLK

CHECK Q3 Q2 /Q1 Q0

SETF OE /CNTRSET UP
CLOCKF CLK

CHECK Q3 Q2 Q1 /Q0

TRACE_OFF
```

Features/Benefits

- Cost-effective mask-programmable complement to PAL64R32 user-programmable device
- CMOS technology provides zero standby power
- High speed with 55ns maximum propagation delay
- High density with 32 highly-flexible macrocells and global connectivity
- Product term sharing, selectable output polarity, and register bypass for high logic efficiency
- Individual clocks for 4 banks of 8 registers
- Register preload for easier test generation
- HC/HCT level compatible for use in CMOS/TTL systems

Description

The ZHAL64R32 circuit is a high-density logic device with thirty-two flexible macrocells. Each macrocell consists of a registered sum of products with feedback, forming a one-bit state machine. The PAL/ZHAL64R32 device can implement over 1500 equivalent logic gates.

The MegaHAL™ circuit features product term sharing between output pairs. This allows sixteen product terms to be shared mutually exclusively between outputs. Selectable polarity allows the output to be either active low or active high, depending on the sense of the equation. Registers can be bypassed in banks of eight, leaving combinatorial outputs. Each register bank has its own clock, preset, preload, and enable controls for independent operation. The register preload pin allows test vectors to be loaded directly into the registers for control of present state conditions for testing.

Design Procedures

The zero-power ZHAL64R32 device is a CMOS, mask-programmable version of the bipolar PAL64R32 circuit. Prototyping can be done with the user-programmable PAL® circuit before committing to a dedicated mask. Thus the MegaPAL™/MegaHAL products combine the instant prototyping of the PAL circuit with the cost-effective, zero power ZHAL™ circuit.

To initiate a design with the ZHAL64R32 device, the PAL64R32 circuit is used to program and debug the design with PALASM®2 software. The resulting "PAL Device Design Specification" is submitted to Monolithic Memories, and the ZHAL circuit option is produced. A ZHAL32R16 option is also available. For details contact a Monolithic Memories representative.

Ordering Information

ZHAL 64 R 32 C NL STD H01234

ZHAL = Zero-Power Hard Array Logic

ARRAY INPUTS

OUTPUT TYPE
R = Registered

NUMBER OF OUTPUTS

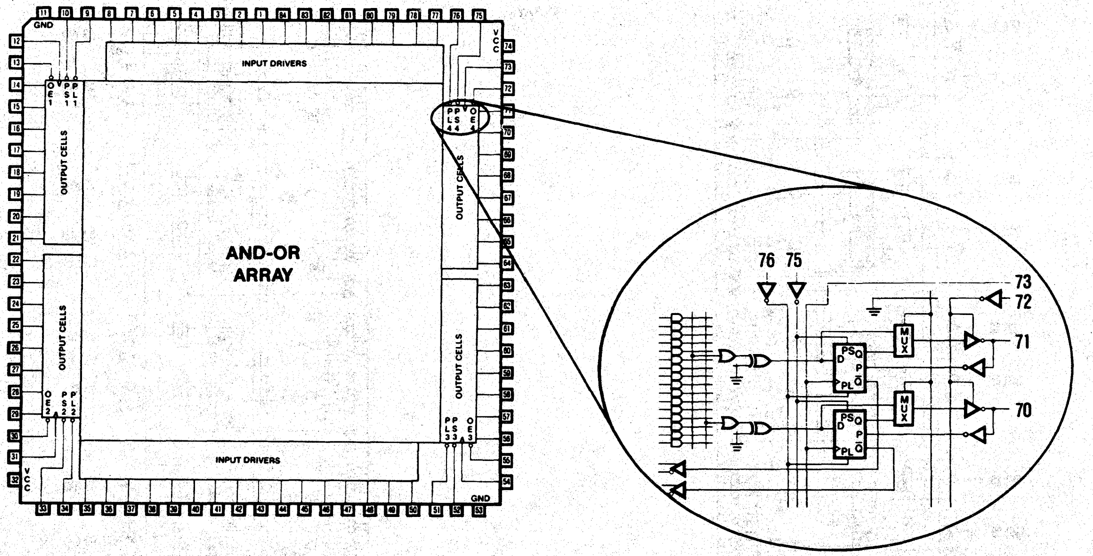
TEMPERATURE RANGE
C = 0°C to 75°C I = -40°C to 85°C

BIT PATTERN NUMBER

PROCESSING
STD = Standard
XXXX = Other

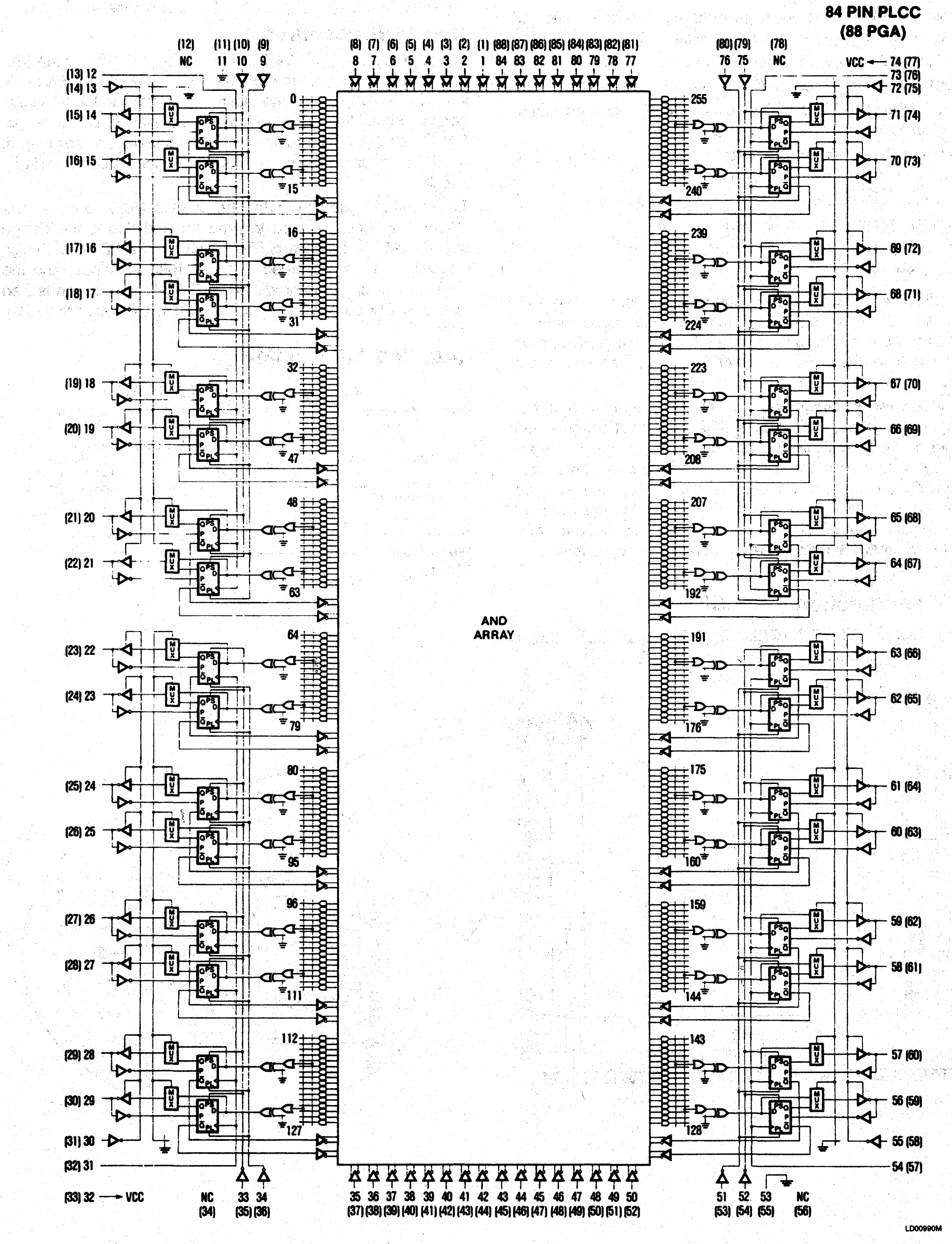
PACKAGE TYPE
NL = Plastic Leaded Chip Carrier
P = Pin Grid Array

Pin Configuration — PLCC



CD01060M

Logic Diagram and Pinout



Operating Conditions

SYMBOL	PARAMETER		COMMERCIAL			INDUSTRIAL			UNIT
			MIN	TYP	MAX	MIN	TYP	MAX	
V_{CC}	Supply voltage		4.75	5	5.25	4.5	5	5.5	V
t_w	Width of clock	Low	20	10		20	10		ns
		High							
t_{su}[1]	Setup time from input or feedback to clock		40	24		45	24		ns
t_h	Hold time		0	−10		0	−10		ns
T_A	Operating free-air temperature		0	25	75	−40	25	85	°C

Electrical Characteristics Over Operating Conditions

SYMBOL	PARAMETER	TEST CONDITIONS		MIN	TYP	MAX	UNIT
V_{IL}[2]	Low-level input voltage			0		0.8	V
V_{IH}[2]	High-level input voltage			2		V_{CC}	V
I_{IL}[5]	Low-level input current	V_{CC} = MAX	V_I = GND			−1	μA
I_{IH}[5]	High-level input current	V_{CC} = MAX	$V_I = V_{CC}$			1	μA
V_{OL}	Low-level output voltage	V_{CC} = MIN	I_{OL} = 4mA		0.1	0.4	V
		V_{CC} = 5V	I_{OL} = 1μA			0.05	
V_{OH}	High-level output voltage	V_{CC} = MIN	I_{OH} = −4mA	3.76[3]	4.1		V
		V_{CC} = 5V	I_{OH} = −1μA	4.95			
I_{OZL}[5]	Off-state output current	V_{CC} = MAX	V_O = GND		0	−10	μA
I_{OZH}[5]			$V_O = V_{CC}$		0	10	
I_{CC}	Standby supply current[4]	I_O = 0mA, V_I = GND or V_{CC}[4]			0	100	
	Operating supply current	I_O = 0mA, V_I = GND or V_{CC}		See graph on page 4-20			

Switching Characteristics Over Operating Conditions

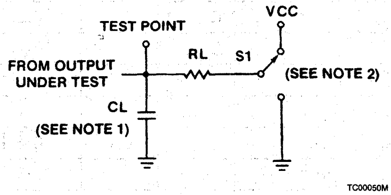
SYMBOL	PARAMETER		TEST CONDITIONS (See Test Load)	COMMERCIAL			INDUSTRIAL			UNIT
				MIN	TYP	MAX	MIN	TYP	MAX	
t_{PD}[1]	Input or feedback to output	Active low	R_L = 1KΩ C_L = 50pF		34	55		34	58	ns
		Active high								
t_{CLK}	Clock to output or feedback				14	22		14	25	ns
t_{PZX}	Output enable				19	30		19	30	ns
t_{PZX}	Output disable				18	30		18	30	ns
t_{PRH}	Preset to output				23	35		23	40	ns
f_{MAX}[1]	Maximum frequency, $f = 1/(t_{SU} + t_{CLK})$			16	25		14	25		MHz

Notes:
1. Maximum 32 inputs per product term.
2. V_{IL} and V_{IH} are absolute voltages with respect to the ground on the device and includes all overshoots due to system and/ or tester noise. Do not attempt to test these values without suitable equipment.
3. Per JEDEC standard no. 7 for high-speed CMOS devices.
4. Disable output pins = V_{CC} or GND.
5. I/O pin leakage is the worst case of I_{OZX} or I_{IX}, i.e., I_{OZH} and I_{IL}.

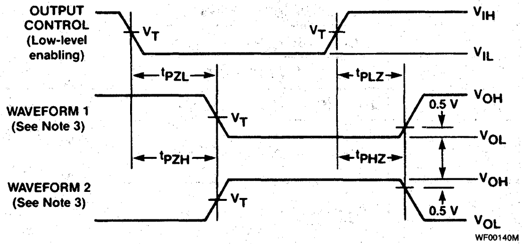
Absolute Maximum Ratings

Supply voltage, V_{CC}	−0.5V to 7.0V
DC input voltage, V_I	−0.5V to V_{CC} +0.5V
DC output voltage, V_O	−0.5V to V_{CC} +0.5V
DC output source/sink current per output pin, I_O	± 35mA
DC V_{CC} or ground current, I_{CC} or I_{GND}	± 100mA
Input diode current, I_{IK}:	
$V_I < 0$	−20mA
$V_I > V_{CC}$	+20mA
Output diode current, I_{OK}:	
$V_O < 0$	−20mA
$V_O > V_{CC}$	+20mA
Storage temperature	−65°C to +150°C
Input rise and fall times	0 to 500ns

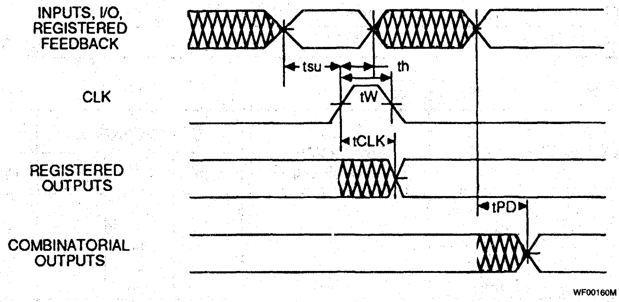
Switching Test Load



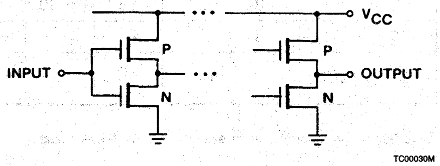
Enable/Disable Delay



Switching Waveforms



Schematic of Inputs and Outputs



Notes: 1. CL includes probe and jig capacitance.

2. When measuring t_{PLZ} and t_{PZL}, S1 is tied to V_{CC}. When measuring t_{PHZ} and t_{PZH}, S1 is tied to ground. When measuring propagation delay times of 3-state outputs, S1 is open, i.e., not connected to V_{CC} or ground.

3. Waveform 1 is for an output with internal conditions such that the output is Low except when disabled by the output control. Waveform 2 is for an output with internal conditions such that the output is High except when disabled by the output control.

ZHAL64R32 Testability Features
Testing Conditions

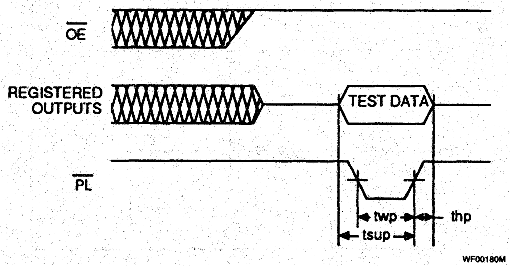
SYMBOL	PARAMETER	COMMERCIAL			INDUSTRIAL			UNIT
		MIN	TYP	MAX	MIN	TYP	MAX	
t_{wp}	Preload pulse width	35			45			ns
t_{sup}	Preload setup time	50			60			ns
t_{hp}	Preload hold time	5			10			ns
t_{PRW}	Preset pulse width	25			30			ns
t_{PRR}*	Preset recovery time	45			50			ns

* Includes setup time for preset data feedback, Maximum 32 inputs per product term.

Preload Feature

Preload pins are provided to enable the testability of each state in state-machine design. Typically for a state machine there are many unreachable states for the registers. These states, and the logic which controls them, are untestable without a way to "set-in" the desired starting state of the registers. In addition, long test sequences are sometimes needed to test a state machine simply to reach those starting states which are legal. Since complete logic verification is needed to ensure the proper exit from "illegal" or unused states, a way to enter these states must be provided. The ability to preload each bank of registers independently to any state is provided in this device.

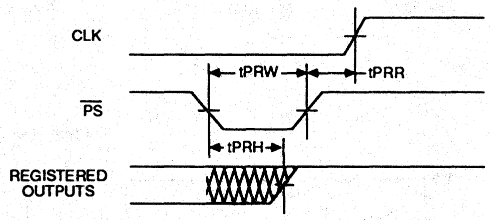
To use the preload feature, several steps must be followed. First, a high level on an assertive-low output enable pin disables the outputs for that bank of registers. Next, the data to be loaded is presented at the output pins. This data is then loaded into the register by placing a low level on the PRELOAD pin. PRELOAD is asynchronous with respect to the clock.



WF00180M

Preset Feature

Register banks of eight may be PRESET to all highs on the outputs by setting the PRESET pin ($\overline{PS}$) to a low level. Note from the Logic Diagram that when the state of an output is high, the state of the register is low due to the inverting three-state buffer.

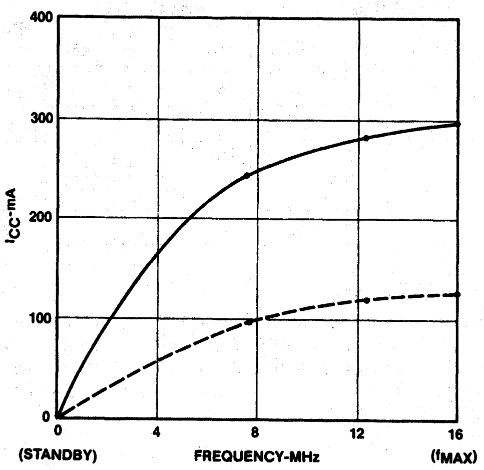


WF00170M

ʾreload data must be held constant while the PRELOAD pin is low.

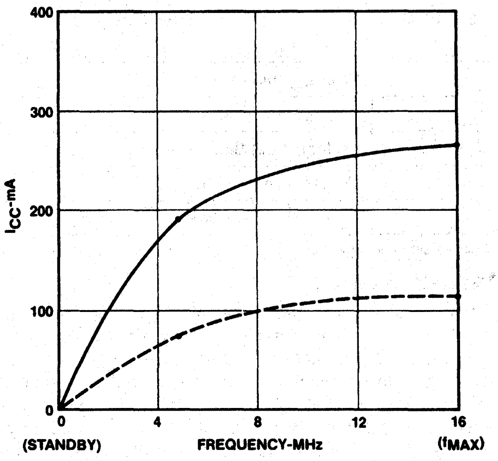
I_{CC} Characteristics vs. Frequency

Commercial Operating Conditions



$T_A = 0°C$

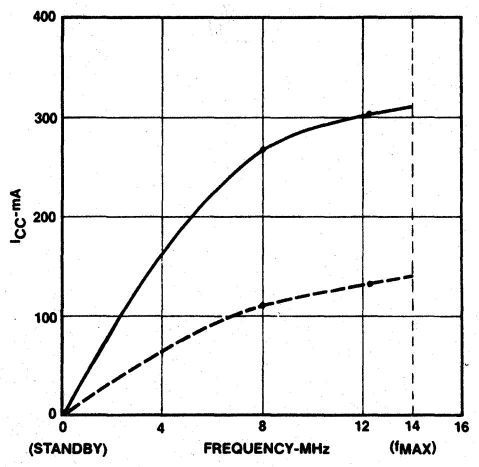
OP00020M



$T_A = 75°C$

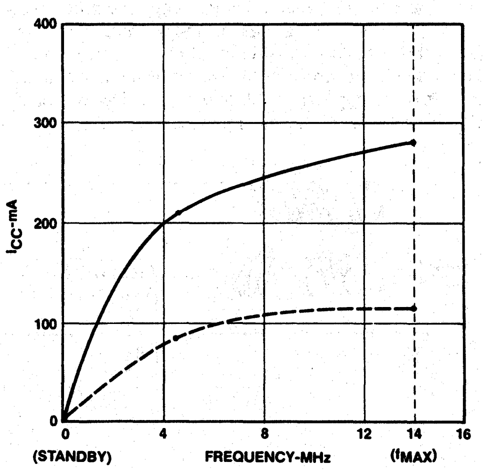
OP00030M

Industrial Operating Conditions



$T_A = -40°C$

OP00040M



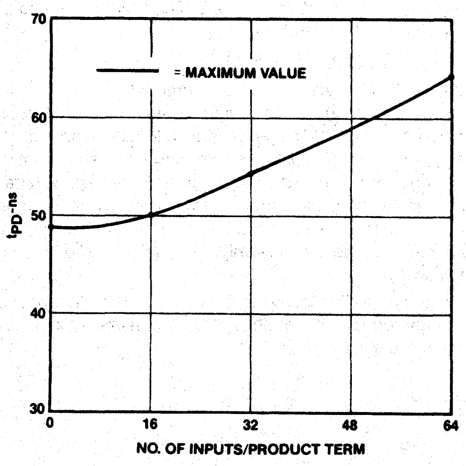
$T_A = 85°C$

OP00050M

——— = MAXIMUM VALUE
– – – = V_{CC} = 5V WITH 32 INPUTS MAX PER PRODUCT TERM

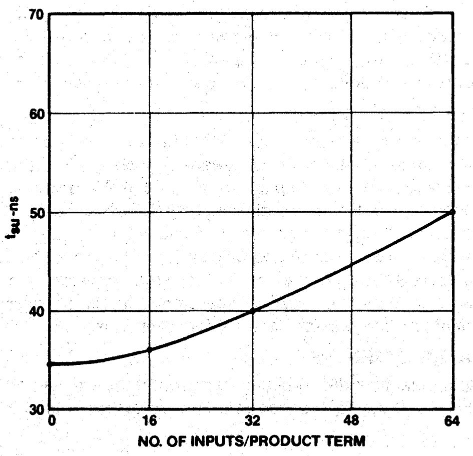
AC Characteristics vs. Loading

Commercial Operating Conditions



$T_A = 75°C$

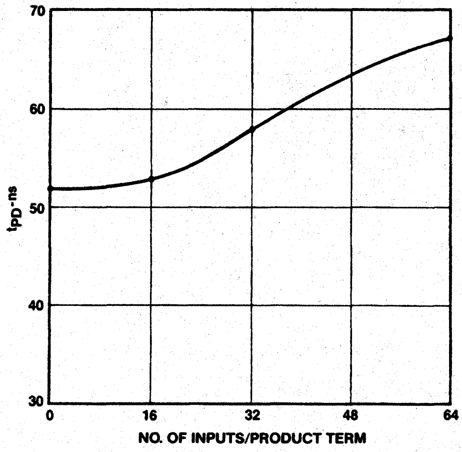
OP00060M



$T_A = 75°C$

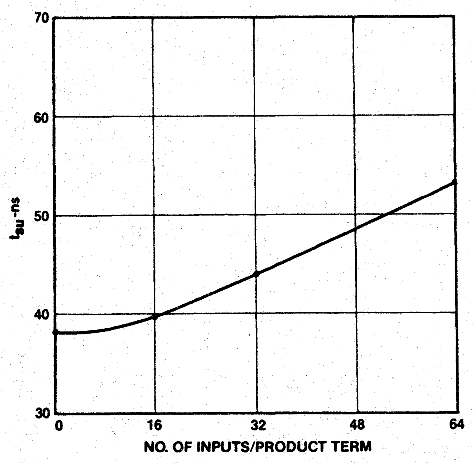
OP00070M

Industrial Operating Conditions



$T_A = 85°C$

OP00080M



$T_A = 85°C$

OP00090M

How To Use The PAL/ZHAL64R32

The following description and example demonstrate the functionality of the PAL/ZHAL64R32, using PALASM 2 software. Conventions for writing equations conform with the PAL Design Specification format. Features to be programmed into the PAL device are completely specified by the equations and automatically configured by PAL device assemblers.

Register Bypass

Outputs within a bank must either be all registered or all combinatorial. Whether or not a bank of registers is bypassed depends on how the outputs are defined in the equations. A colon followed by an equal sign [:=] specifies a registered output with feedback which is updated after the low-to-high transition of the clock. An equal sign [=] defines a combinatorial output which bypasses the register. Registers are bypassed in banks of eight. Bypassing a bank of registers eliminates the feedback lines for those outputs.

Output Polarity

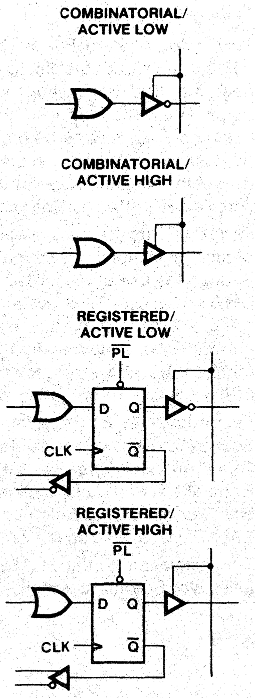
Output polarity is defined by comparison of the pin list and the equations. If the logic sense of a specific output in the pin list is different from the logic sense of that output as defined by its equation, the output is inverted or active low polarity. If the logic sense of a specific output in the pin list is the same as the logic sense of that output as defined by its equation, the output is active high polarity.

Product Term Sharing

The basic configuration is sixteen product terms shared between two output cells. For a typical output pair, each product term can be used by either output; but, since product term sharing is exclusive, a product term can be used by only one output, not both. If equations call for both outputs to use the same product term, two product terms are generated, one for each output. This should be taken into account when writing equations. PAL device assemblers configure product terms automatically.

This example on the following page uses the 84-pin package. Four output equations are shown to demonstrate functionality. Pin names are arbitrary.

```
TITLE     ZHAL64R32 DESIGN EXAMPLE USING PALASM 2
PATTERN   EXAMPLE.PDS
REVISION  PALASM 2 VERSION
AUTHOR    NAME
COMPANY   MONOLITHIC MEMORIES, INC.
DATE      1986

        CHIP FUNCTIONALITY_EXAMPLE PAL64R32

        I1 I2 I3 I4 I5 I6 I7 I8 /PL1/PS1 GND CLK1 /OE1
        Q1 Q2 Q3 Q4 Q5 Q6 Q7 Q8 Q9 Q10 Q11 Q12 Q13 Q14 Q15 Q16
        /OE2 CLK2 VCC /PS2 /PL2
        I9 I10 I11 I12 I13 I14 I15 I16 I17 I18 I19 I20 I21 I22 I23 I24
        /PL3 /PS3 GND CLK3 /OE3
        Q17 Q18 Q19 Q20 Q21 Q22 Q23 Q24 Q25 Q26 Q27 Q28 Q29 Q30 Q31 Q32
        /OE4 CLK4 VCC /PS4 /PL4 I25 I26 I27 I28 I29 I30 I31 I32

        EQUATIONS

        /Q1  = I1*I2*I3                   ;ACTIVE LOW COMBINATORIAL
             + I11*I27*/I32*Q9            ;OUTPUT.  SINCE THE
             + I4*/I9*Q10                 ;REGISTERS ARE BYPASSED
             + I1*I2*I7*/Q27*Q15*/Q32     ;FOR THIS BANK, THE
             + I8*/Q13*/Q32               ;OUTPUTS CANNOT FEED
                                          ;BACK INTO THE ARRAY.

         Q2  = I1*I2*I3                   ;ACTIVE HIGH
             + I27*/Q12                   ;COMBINATORIAL OUTPUT.
             + I15*/I32*/Q9               ;Q2 USES THE REMAINING
             +/I1*I2*/I3*I4*Q27           ;11 PRODUCT TERMS
             + Q10*Q11                    ;AFTER /Q1 USED 5.
             +/Q32                        ;NOTE THAT THE FIRST
             + I4*I9*I11*Q12*/Q27         ;PRODUCT TERM IS
             + Q9*Q27*/Q15                ;CREATED TWICE.
             + I6*I7*/I8*/I9*/I10*/I12
             + I6*Q12*Q15
             + I11*/Q15

        /Q9 := I2*/I4*Q15                 ;ACTIVE LOW REGISTERED
             +/I1*I4*I9*I15*I32           ;OUTPUT.  SOME OF THE
             + Q32*/Q15                   ;OUTPUT SIGNALS ARE FED
             +/Q9*I3                      ;BACK INTO THE ARRAY,
             + I8*/Q13*/Q32               ;SINCE THE REGISTER IS
             + I5*/I6*I11*/Q15            ;USED.
             + I12*I13*I27

        Q10 := I3*/I8                     ;ACTIVE HIGH REGISTERED
             + I9*Q15*/Q32                ;OUTPUT.
             + I7*I8*/I9*/Q11*/Q27*/Q32
```



TB02010M

Features/Benefits

- Zero standby power
- Low power operation
- High-speed CMOS technology
- HC and HCT compatible
- 24-pin SKINNYDIP® and 28-pin PLCC packages save space
- Low power alternative for most 24-pin PAL® devices, including 20L8/20R8/20R6/20R4

Description

This family of Zero Power Hard Array Logic (ZHAL) devices utilizes a unique architecture that is designed for a high degree of flexibility in implementing most patterns of the listed 24-pin PAL/HAL® devices. Prototyping should be done using standard PAL devices before converting to ZHAL circuits for production. ZHAL devices are fabricated by Monolithic Memories with custom metallization masks defined by a user-supplied HAL Design Specification.

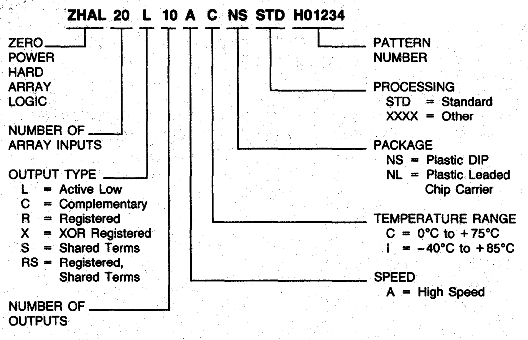
The ZHAL devices are ideal in low-power applications that require high-speed operation. These attributes are achieved through the use of Monolithic Memories' advanced high-speed CMOS process. Now system designers have the option of using a ZHAL device that matches fast PAL device speeds, but with the added feature of zero standby power. These features are needed in power-critical areas such as portable digital equipment or lap-top computers.

The procedures for designing with Monolithic Memories' ZHAL devices are shown in the flow chart on the next page. The ZHAL option in the PALASM®2 CAD package will confirm whether a design specification will fit within the ZHAL architecture. For more information on the ZHAL software, refer to the PALASM 2 User Manual.

For evaluation of the ZHAL24A circuit, sample patterns are available. See the description in this document for details.

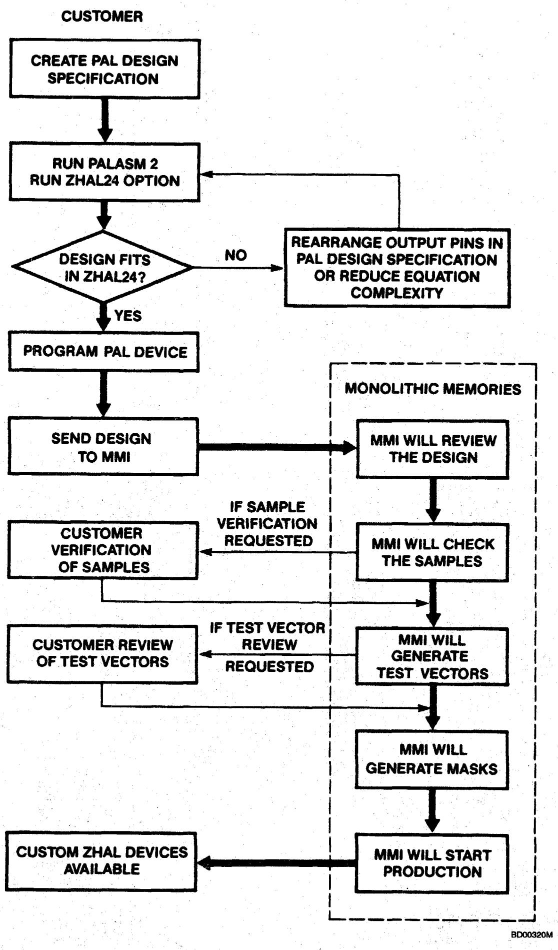
Ordering Information

PART NUMBER	PACKAGE	ARRAY INPUTS	OUTPUTS	
			COMB	REG
ZHAL12L10A	NS, NL	12	10	—
ZHAL14L8A	NS, NL	14	8	—
ZHAL16L6A	NS, NL	16	6	—
ZHAL18L4A	NS, NL	18	4	—
ZHAL20L2A	NS, NL	20	2	—
ZHAL20C1A	NS, NL	20	2	—
ZHAL20L8A	NS, NL	20	8	—
ZHAL20R8A	NS, NL	20	—	8
ZHAL20R6A	NS, NL	20	2	6
ZHAL20R4A	NS, NL	20	4	4
ZHAL20L10A	NS, NL	20	10	—
ZHAL20X10A	NS, NL	20	—	10
ZHAL20X8A	NS, NL	20	2	8
ZHAL20X4A	NS, NL	20	6	4
ZHAL20S10A	NS, NL	20	10	—
ZHAL20RS10A	NS, NL	20	—	10
ZHAL20RS8A	NS, NL	20	2	8
ZHAL20RS4A	NS, NL	20	6	4



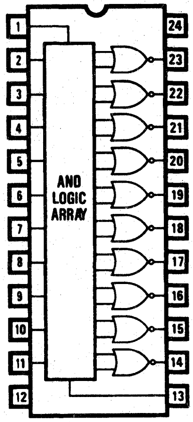
ZHAL 24A Device Generation Flow

ZHAL24A DEVICE GENERATION FLOW

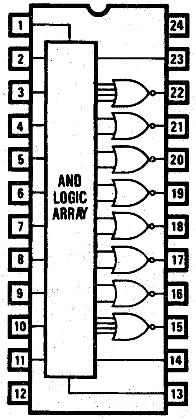

CUSTOMER
CREATE PAL DESIGN SPECIFICATION
RUN PALASM 2 RUN ZHAL24 OPTION
DESIGN FITS IN ZHAL24?
NO
REARRANGE OUTPUT PINS IN PAL DESIGN SPECIFICATION OR REDUCE EQUATION COMPLEXITY
YES
PROGRAM PAL DEVICE
MONOLITHIC MEMORIES
SEND DESIGN TO MMI
MMI WILL REVIEW THE DESIGN
IF SAMPLE VERIFICATION REQUESTED
CUSTOMER VERIFICATION OF SAMPLES
MMI WILL CHECK THE SAMPLES
IF TEST VECTOR REVIEW REQUESTED
CUSTOMER REVIEW OF TEST VECTORS
MMI WILL GENERATE TEST VECTORS
MMI WILL GENERATE MASKS
CUSTOM ZHAL DEVICES AVAILABLE
MMI WILL START PRODUCTION
BD00320M

Pin Configurations – DIP and PLCC

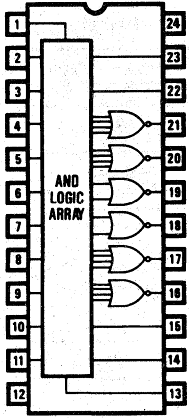
ZHAL12L10A



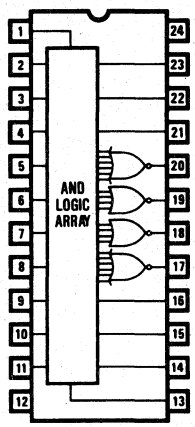
ZHAL14L8A



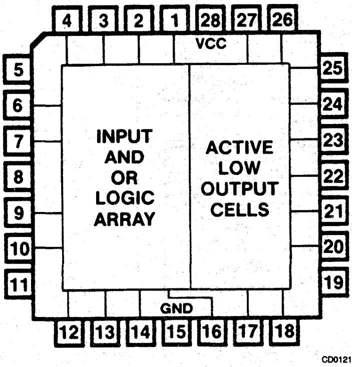
ZHAL16L6A



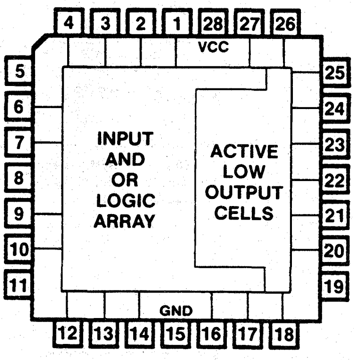
ZHAL18L4A



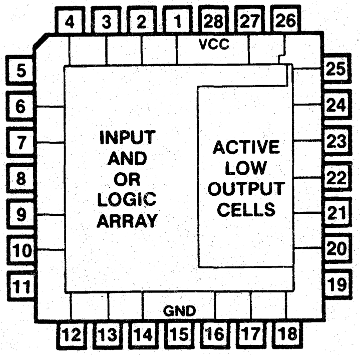
ZHAL12L10A



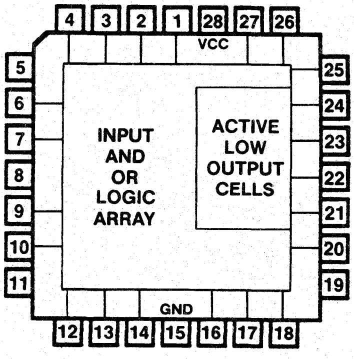
ZHAL14L8A



ZHAL16L6A

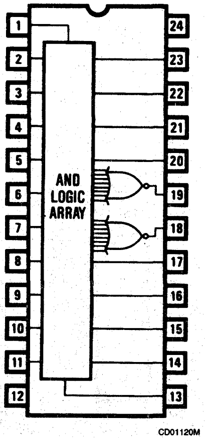


ZHAL18L4A

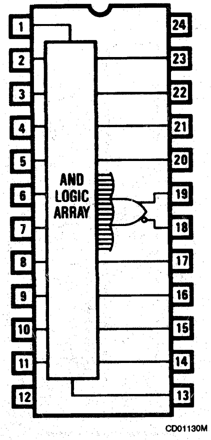


Pin Configurations – DIP and PLCC

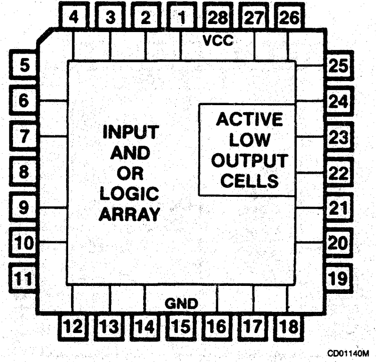
ZHAL20L2A



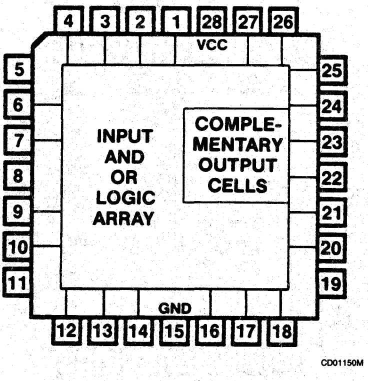
ZHAL20C1A



ZHAL20L2A

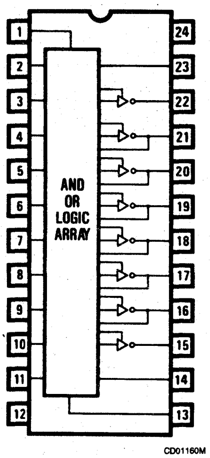


ZHAL20C1A

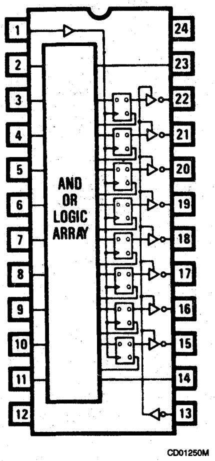


Pin Configurations – DIP and PLCC

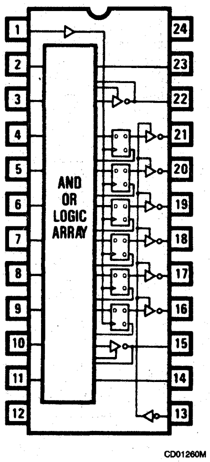
ZHAL20L8A



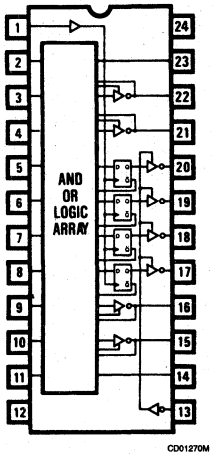
ZHAL20R8A



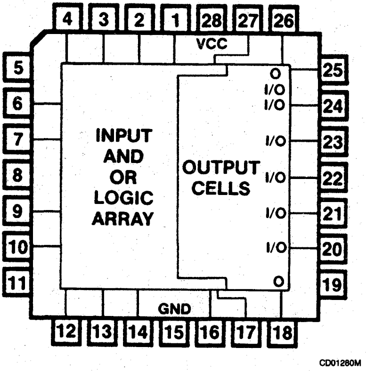
ZHAL20R6A



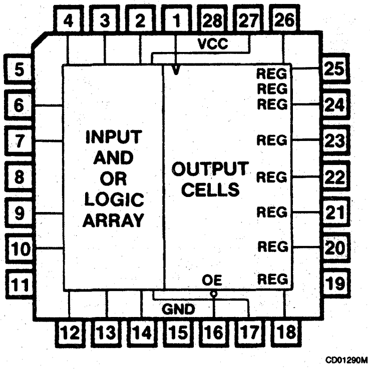
ZHAL20R4A



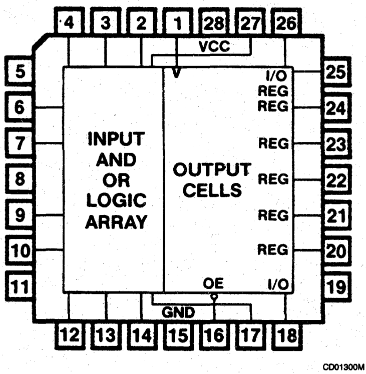
ZHAL20L8A



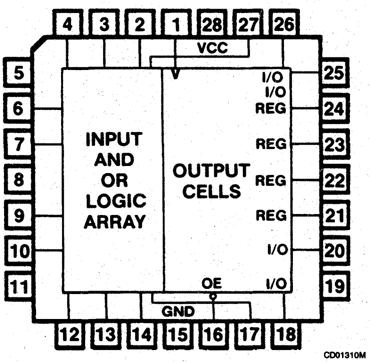
ZHAL20R8A



ZHAL20R6A

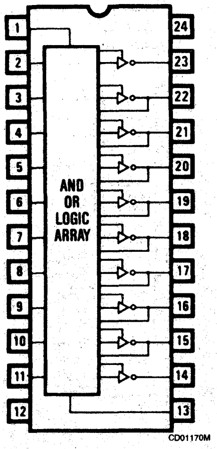


ZHAL20R4A



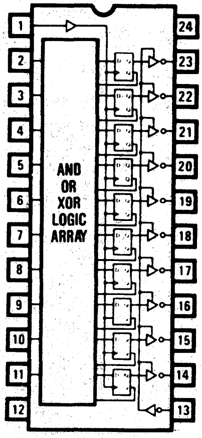
Pin Configurations – DIP and PLCC

ZHAL20L10A



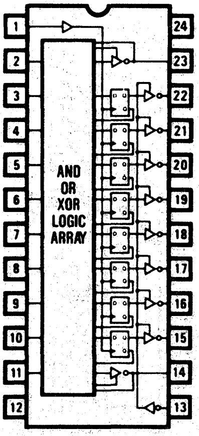
CD01170M

ZHAL20X10A



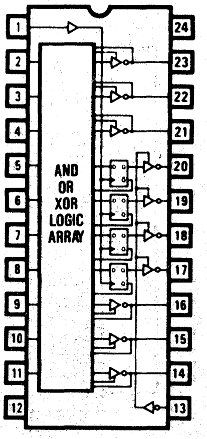
CD01320M

ZHAL20X8A



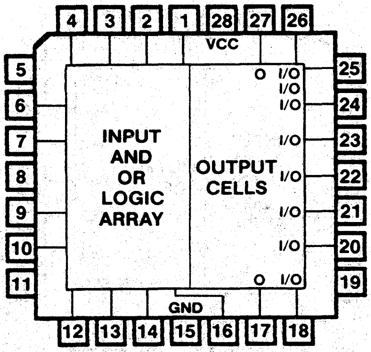
CD01330M

ZHAL20X4A



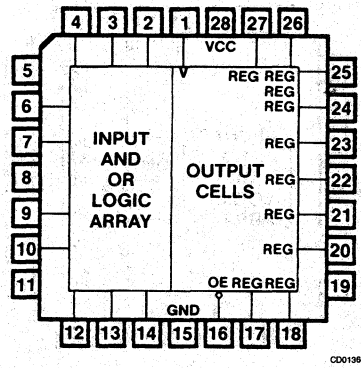
CD01340M

ZHAL20L10A



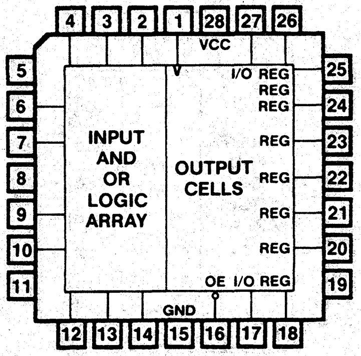
CD01350M

ZHAL20X10A



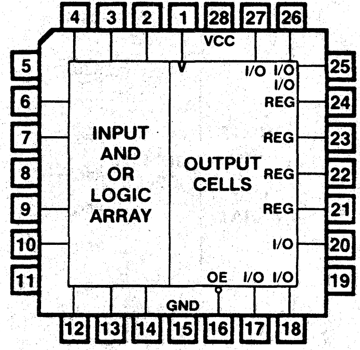
CD01360M

ZHAL20X8A



CD01370M

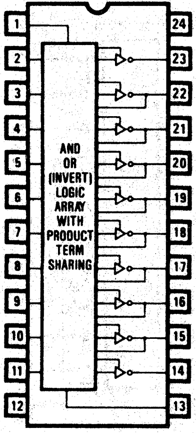
ZHAL20X4A



CD01380M

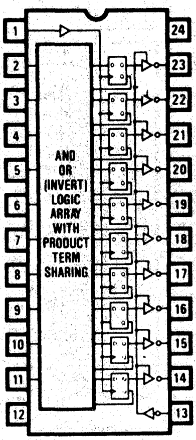
Pin Configurations – DIP and PLCC

ZHAL20S10A



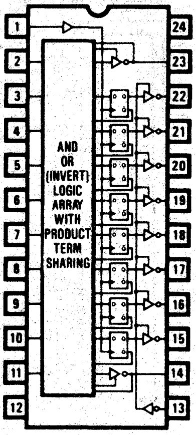
CD01180M

ZHAL20RS10A



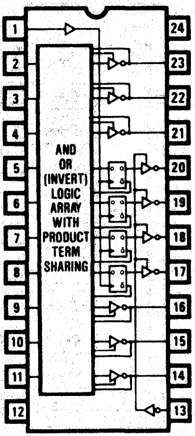
CD01390M

ZHAL20RS8A



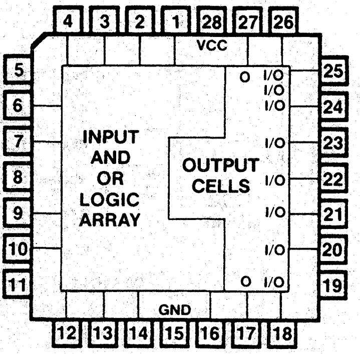
CD01400M

ZHAL20RS4A



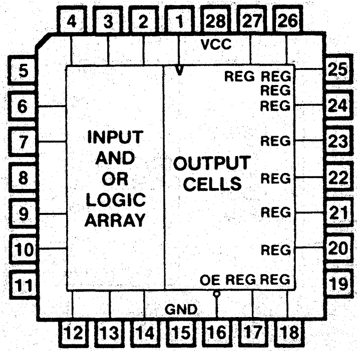
CD01410M

ZHAL20S10A



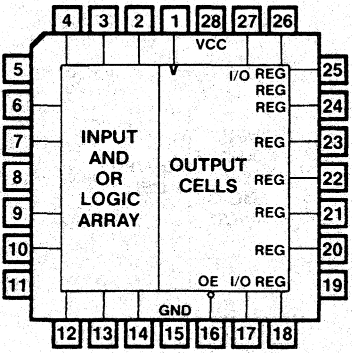
CD01420M

ZHAL20RS10A



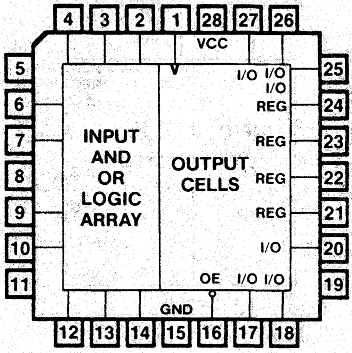
CD01430M

ZHAL20RS8A



CD01440M

ZHAL20RS4A



CD01450M

Operating Conditions

SYMBOL	PARAMETER	INDUSTRIAL			COMMERCIAL			UNIT
		MIN	TYP	MAX	MIN	TYP	MAX	
V_{CC}	Supply voltage	4.5	5	5.5	4.75	5	5.25	V
T_A	Operating free-air temperature	−40	25	85	0	25	75	°C

Electrical Characteristics Over Operating Conditions

SYMBOL	PARAMETER	TEST CONDITIONS		MIN	TYP	MAX	UNIT
V_{IL}[1]	Low-level input voltage			0		0.8	V
V_{IH}[1]	High-level input voltage			2		V_{CC}	V
I_{IL}	Low-level input current	V_{CC} = MAX	V_I = GND			−1	μA
I_{IH}	High-level input current	V_{CC} = MAX	$V_I = V_{CC}$			1	μA
V_{OL}	Low-level output voltage	V_{CC} = MIN	I_{OL} = 8mA		0.1	0.4	V
		V_{CC} = 5V	I_{OL} = 1μA			0.05	
V_{OH}	High-level output voltage	V_{CC} = MIN	I_{OH} = −6mA	3.76[2]	4.1		V
		V_{CC} = 5V	I_{OH} = −1μA	4.95			
I_{OZL}[3]	Off-state output current	V_{CC} = MAX	V_O = GND		0	−10	μA
I_{OZH}[3]			$V_O = V_{CC}$		0	10	μA
I_{CC}	Standby supply current[4]	I_O = 0mA, V_I = GND or V_{CC}			0	100	μA
	Operating supply current	f = MHz, I_O = 0mA, V_I = GND or V_{CC}			2	5[5]	mA

Switching Characteristics Over Operating Conditions

SYMBOL	PARAMETER	TEST CONDITIONS (See Test Load)	INDUSTRIAL			COMMERCIAL			UNIT
			MIN	TYP	MAX	MIN	TYP	MAX	
t_{PD}	Input to output	R_L = 1KΩ C_L = 50pF		13	25[6]		13	25[6]	ns

Notes:
1. These are absolute voltages with respect to the the ground pin on the device and includes all overshoots due to system and/ or tester noise. Do not attempt to test these values without suitable equipment.
2. JEDEC standard no. 7 for high-speed CMOS devices.
3. Applies to pins 14-23 for DIP (pins 17, 18, 20-27 for PLCC).
4. Disable output pins = V_{CC} or GND.
5. Add 3mA per additional 1.0MHz of operation over 1MHz.
6. For outputs with more than 12 inputs in a product term, t_{PD} = 30ns.

Operating Conditions

SYMBOL	PARAMETER		INDUSTRIAL			COMMERCIAL			UNIT
			MIN	TYP	MAX	MIN	TYP	MAX	
V_{CC}	Supply voltage		4.5	5	5.5	4.75	5	5.25	V
t_w	Width of clock	20R8A, 20R6A, 20R4A	15	4		15	4		ns
t_{su}	Setup time from input or feedback to clock		20[1]	11		20[1]	11		ns
t_h	Hold time		0	−10		0	−10		ns
T_A	Operating free-air temperature		−40	25	85	0	25	75	°C

Electrical Characteristics Over Operating Conditions

SYMBOL	PARAMETER		TEST CONDITIONS		MIN	TYP	MAX	UNIT
V_{IL}[2]	Low-level input voltage				0		0.8	V
V_{IH}[2]	High-level input voltage				2		V_{CC}	V
I_{IL}	Low-level input current		V_{CC} = MAX	V_I = GND			−1	μA
I_{IH}	High-level input current	Pin 10[3]	V_{CC} = MAX	V_I = V_{CC}		8	30	μA
		All other pins					1	μA
V_{OL}	Low-level output voltage		V_{CC} = MIN	I_{OL} = 8mA		0.1	0.4	V
			V_{CC} = 5V	I_{OL} = 1μA			0.05	
V_{OH}	High-level output voltage		V_{CC} = MIN	I_{OH} = −6mA	3.76[4]	4.1		V
			V_{CC} = 5V	I_{OH} = −1μA	4.95			
I_{OZL}[5]	Off-state output current		V_{CC} = MAX	V_O = GND		0	−10	μA
I_{OZH}[5]				V_O = V_{CC}		0	10	μA
I_{CC}	Standby supply current[6]		I_O = 0mA, V_I = GND or V_{CC}			0	100	μA
	Operating supply current		f = 1MHz, I_O = 0mA, V_I = GND or V_{CC}			2	5[7]	mA

Switching Characteristics Over Operating Conditions

SYMBOL	PARAMETER		TEST CONDITIONS (See Test Load)	INDUSTRIAL			COMMERCIAL			UNIT
				MIN	TYP	MAX	MIN	TYP	MAX	
t_{PD}	Input or feedback to output 20L8A, 20R6A, 20R4A		R_L = 1KΩ C_L = 50pF		13	25[1]		13	25[1]	ns
t_{CLK}	Clock to output or feedback 20R8A, 20R6A, 20R4A				8	15		8	15	ns
t_{PZX}	Input to output enable	20L8A, 20R6A, 20R4A			12	25		12	25	ns
t_{PXZ}[8]	Input to output disable				12	25		12	25	ns
t_{PXZ}[8] t_{PZX}	Pin 13 (DIP) to ouutput disable/enable	20R8A, 20R6A, 20R4A			10	20		10	20	ns
f_{MAX}	Maximum frequency			28.5	40		28.5	40		MHz

Notes:
1. For outputs with more than 12 inputs in a product term, t_{SU} = 25ns and t_{PD} = 30ns.
2. These are absolute voltages with respect to the ground pin on the device and include all overshoots due to system and/or tester noise. Do not attempt to test these values without suitable equipment.
3. Pin 10 (PRELOAD pin), pin 13PLCC (PRELOAD pin). Applies to registered devices only.
4. JEDEC standard no. 7 for high-speed CMOS devices.
5. Applies to pins 14-23 for DIP (pins 17, 18, 20-27 for PLCC).
6. Disable output pins = V_{CC} or GND.
7. Add 3mA per additional 1.0MHz of operation over 1MHz.
8. C_L = 5pF.

Operating Conditions

SYMBOL	PARAMETER		INDUSTRIAL			COMMERCIAL			UNIT
			MIN	TYP	MAX	MIN	TYP	MAX	
V_{CC}	Supply Voltage		4.5	5	5.5	4.75	5	5.25	V
t_w	Width of clock	20X10A, 20X8A, 20X4A	15	5		15	5		ns
t_{su}	Setup time from input or feedback to clock		25[1]	15		25[1]	15		ns
t_h	Hold time		0	−10		0	−10		ns
T_A	Operating free-air temperature		−40	25	85	0	25	75	°C

Electrical Characteristics Over Operating Conditions

SYMBOL	PARAMETER		TEST CONDITIONS		MIN	TYP	MAX	UNIT
V_{IL}[2]	Low-level input voltage				0		0.8	V
V_{IH}[2]	High-level input voltage				2		V_{CC}	V
I_{IL}	Low-level input current		V_{CC} = MAX	V_I = GND			−1	μA
I_{IH}	High-level input current	Pin 10[3]	V_{CC} = MAX	$V_I = V_{CC}$		8	30	μA
		All other pins					1	μA
V_{OL}	Low-level output voltage		V_{CC} = MIN	I_{OL} = 8mA		0.1	0.4	V
			V_{CC} = 5V	I_{OL} = 1μA			0.05	
V_{OH}	High-level output voltage		V_{CC} = MIN	I_{OH} = −6mA	3.76[4]	4.1		V
			V_{CC} = 5V	I_{OH} = −1μA	4.95			
I_{OZL}[5]	Off-state output current		V_{CC} = MAX	V_O = GND		0	−10	μA
I_{OZH}[5]				$V_O = V_{CC}$		0	10	μA
I_{CC}	Standby supply current[6]		I_O = 0mA, V_I = GND or V_{CC}			0	100	μA
	Operating supply current		f = 1MHz, I_O = 0mA, V_I = GND or V_{CC}			2	5[7]	mA

Switching Characteristics Over Operating Conditions

SYMBOL	PARAMETER		TEST CONDITIONS (See Test Load)	INDUSTRIAL			COMMERCIAL			UNIT
				MIN	TYP	MAX	MIN	TYP	MAX	
t_{PD}	Input or feedback to output 20L10A, 20X8A, 20X4A		R_L = 1KΩ C_L = 50pF		13	25[1]		13	25[1]	ns
t_{CLK}	Clock to output or feedback 20X10A, 20X8A, 20X4A				10	15		10	15	ns
t_{PZX}	Input to output enable	20L10A, 20X8A, 20X4A			12	25		12	25	ns
t_{PXZ}[8]	Input to output disable				12	25		12	25	ns
t_{PXZ}[8]	Pin 13 (DIP) to output disable/enable	20X10A, 20X8A, 20X4A			15	20		15	20	ns
t_{PZX}										
f_{MAX}	Maximum frequency			22.2	32		22.2	32		MHz

Notes:
[1] For outputs with more than 12 inputs in a product term, t_{SU} = 30ns and t_{PD} = 30ns.
[2] These are absolute voltages with respect to the ground pin on the device and include all overshoots due to system and/or tester noise. Do not attempt to test these values without suitable equipment.
[3] Pin 10 (PRELOAD pin), pin 13PLCC (PRELOAD pin). Applies to registered devices only.
[4] JEDEC standard no. 7 for high-speed CMOS devices.
[5] Applies to pins 14-23 for DIP (pins 17, 18, 20-27 for PLCC).
[6] Disable output pins = V_{CC} or GND.
[7] Add 3mA per additional 1.0MHz of operation over 1MHz.
[8] C_L = 5pF.

Operating Conditions

SYMBOL	PARAMETER		INDUSTRIAL			COMMERCIAL			UNIT
			MIN	TYP	MAX	MIN	TYP	MAX	
V_{CC}	Supply voltage		4.5	5	5.5	4.75	5	5.25	V
t_w	Width of clock	20RS10A, 20RS8A, 20S4A	15	4		15	4		ns
t_{su}	Setup time for input clock		20[1]	11		20[1]	11		ns
t_h	Hold time		0	−10		0	−10		
T_A	Operating free-air temperature		−40	25	85	0	25	75	°C

Electrical Characteristics Over Operating Conditions

SYMBOL	PARAMETER		TEST CONDITIONS		MIN	TYP	MAX	UNIT
V_{IL}[2]	Low-level input voltage				0		0.8	V
V_{IH}[2]	High-level input voltage				2		V_{CC}	V
I_{IL}	Low-level input current		V_{CC} = MAX	V_I = GND			−1	μA
I_{IH}	High-level input current	Pin 10[3]	V_{CC} = MAX	V_I = V_{CC}		8	30	μA
		All other pins					1	μA
V_{OL}	Low-level output voltage		V_{CC} = MIN	I_{OL} = 8mA		0.1	0.4	V
			V_{CC} = 5V	I_{OL} = 1μA			0.05	
V_{OH}	High-level output voltage		V_{CC} = MIN	I_{OH} = −6mA	3.76[4]	4.1		V
			V_{CC} = 5V	I_{OH} = −1μA	4.95			
I_{OZL}[5]	Off-state output current		V_{CC} = MAX	V_O = GND		0	−10	μA
I_{OZH}[5]				V_O = V_{CC}		0	10	μA
I_{CC}	Standby supply current[6]		I_O = 0mA, V_I = GND or V_{CC}			0	100	μA
	Operating supply current		f = 1MHz, I_O = 0mA, V_I = GND or V_{CC}			2	5[7]	mA

Switching Characteristics Over Operating Conditions

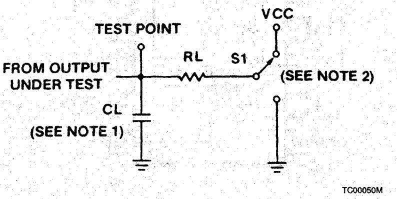
SYMBOL	PARAMETER		TEST CONDITIONS (See Test Load)	INDUSTRIAL			COMMERCIAL			UNIT
				MIN	TYP	MAX	MIN	TYP	MAX	
t_{PD}	Input or feedback to output 20S10A, 20RS8A, 20RS4A		R_L = 1KΩ C_L = 50pF		13	25[1]		13	25[1]	ns
t_{CLK}	Clock to output or feedback 20RS10A, 20RS8A, 20RS4A				8	15		8	15	ns
t_{PZX}	Input to output enable	20S10A, 20RS8A, 20RS4A			12	25		12	25	ns
t_{PXZ}[8]	Input to output disable				12	25		12	25	ns
t_{PXZ}[8]	Pin 13 (DIP) to output disable/ enable	20RS10A, 20RS8A, 20RS4A			10	20		10	20	ns
t_{PZX}										
f_{MAX}	Maximum frequency			28.5	40		28.5	40		MHz

Notes:
1. For outputs with more than 12 inputs in a product term, t_{SU} = 25ns and t_{PD} = 30ns.
2. These are absolute voltages with respect to the ground pin on the device and include all overshoots due to system and/or tester noise. Do not attempt to test these values without suitable equipment.
3. Pin 10 (PRELOAD pin), pin 13PLCC (PRELOAD pin). Applies to registered devices only.
4. JEDEC standard no. 7 for high-speed CMOS devices.
5. Applies to pins 14-23 for DIP (pins 17, 18, 20-27 for PLCC).
6. Disable output pins = V_{CC} or GND.
7. Add 3mA per additional 1.0MHz of operation over 1MHz.
8. C_L = 5pF.

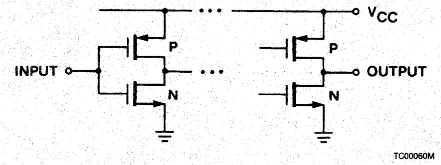
Absolute Maximum Ratings

Supply voltage, V_{CC}	−0.5V to 7V
DC input voltage, V_I	−0.5V to V_{CC} +0.5V
DC output voltage, V_O	−0.5V to V_{CC} +0.5V
DC output source/sink current per output pin, I_O	± 35mA
DC V_{CC} or ground current, I_{CC} or I_{GND}	± 100mA
Input diode current, I_{IK}:	
$V_I < 0$	−20mA
$V_I > V_{CC}$	+20mA
Output diode current, I_{OK}:	
$V_O < 0$	−20mA
$V_O > V_{CC}$	+20mA
Storage temperature	−65°C to +150°C

Switching Test Load



Schematic of Inputs and Outputs



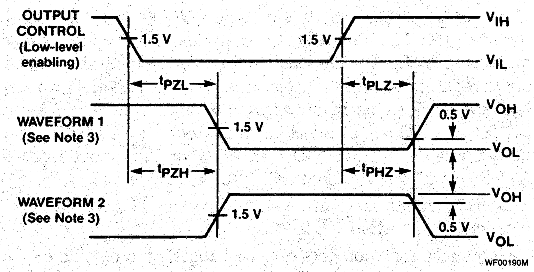
Output Register PRELOAD†

The PRELOAD function allows the register to be loaded from data placed on the output pins. This feature aids functional testing of state sequencer designs by allowing direct setting of output states for improved test coverage. The procedure for PRELOAD using DIP pin numbers, is as follows:

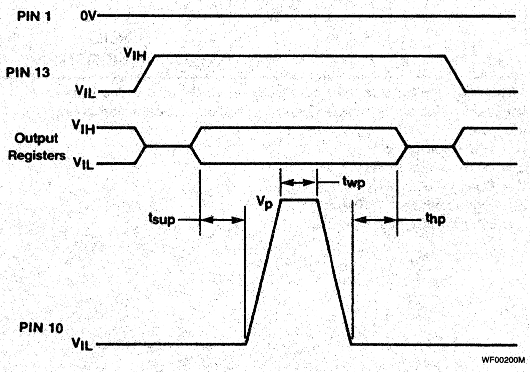
1. Raise V_{CC} to 4.5V.
2. Disable output registers by setting pin 13 to V_{IH}. Set pin 1 to 0V.
3. Apply V_{IL}/V_{IH} to all registered outputs.
4. Pulse pin 10 to V_p (12V), then back to 0V.
5. Remove V_{IL}/V_{IH} from all registered outputs.
6. Lower pin 13 to V_{IL} to enable the output registers.
7. Verify for V_{OL}/V_{OH} at all registered outputs.

† Note: Only applies to parts with output registers.
Typical t_{sup} = 50ns
t_{wp} = 100ns
t_{hp} = 50ns
I_{IH} = 30μA (Pin 10)

Enable/Disable Delay



Notes:
1. CL includes probe and jig capacitance.
2. When measuring t_{PLZ} and t_{PZL}, S1 is tied to V_{CC}. When measuring t_{PHZ} and t_{PZH}, S1 is tied to ground. t_{PZX} is measured with C_L = 50pF. t_{PXZ} is measured with C_L = 5pF.
When measuring propagation delay times of three-state outputs, S1 is open, i.e., not connected to V_{CC} or ground.
3. Waveform 1 is for an output with internal conditions such that the output is LOW except when disabled by the output control.
Waveform 2 is for an output with internal conditions such that the output is HIGH except when disabled by the output control.



Features/Benefits

- Demonstration pattern for ZHAL24A Series (ZHAL20X8A)
- 8-bit counter
- Three-state output
- Expandable in 8-bit increments
- Equivalent to 74ACT461

Description

The ZHAL24A Evaluation #4 pattern is provided as an example of the features and characteristics of the ZHAL24A Series products. The design consists of an 8-bit synchronous counter with parallel load, clear, and hold capability. Two function select inputs (I0, I1) provide one of four operations which occur synchronously on the rising edge of the clock (CK).

The LOAD operation loads the inputs (D7-D0) into the output register (Q7-Q0). The CLEAR operation resets the output register to all LOWs. The HOLD operation holds the previous value regardless of clock transitions. The INCREMENT operation adds one to the output register when the carry-in input is TRUE($\overline{\text{CI}}$ = LOW), otherwise the operation is a HOLD. The carry-out ($\overline{\text{CO}}$) is TRUE ($\overline{\text{CO}}$ = LOW) when the output register (Q7-Q0) is all HIGHs, otherwise FALSE ($\overline{\text{CO}}$ = HIGH).

The data output pins are enabled when $\overline{\text{OE}}$ is LOW, and disabled (HI-Z) when $\overline{\text{OE}}$ is HIGH.

Two or more 8-bit counters may be cascaded to provide larger counters. The operation codes were chosen such that when I1 is HIGH, I0 may be used to select between LOAD and INCREMENT as in a program counter (JUMP/INCREMENT).

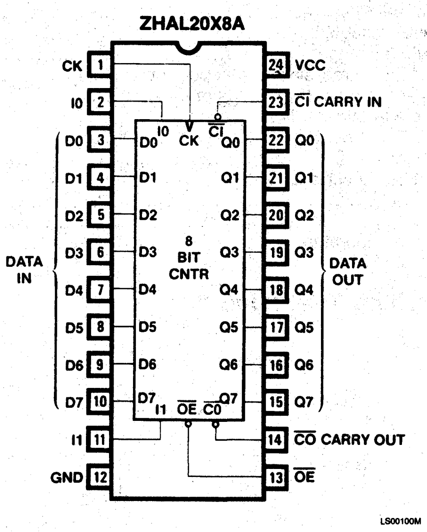
Function Table

$\overline{\text{OE}}$	CK	I1	I0	$\overline{\text{CI}}$	D7-D0	Q7-Q0	OPERATION
H	*	*	*	*	*	Z	HI-Z*
L	↑	L	L	X	X	L	CLEAR
L	↑	L	H	X	X	Q	HOLD
L	↑	H	L	X	D	D	LOAD
L	↑	H	H	H	X	Q	HOLD
L	↑	H	H	L	X	Q plus 1	INCREMENT

*When $\overline{\text{OE}}$ is HIGH, the three-state outputs are disabled to the high-impedance states; however, sequential operation of the counter is not affected.

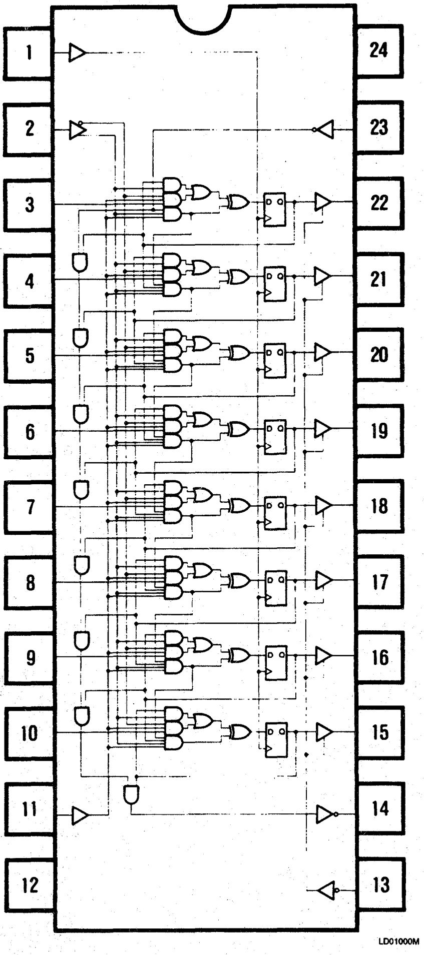
H = HIGH voltage level
L = LOW voltage level
X = Don't care
Z = High impedance (off) state
↑ = LOW-to-HIGH clock transition

Logic Symbol



Logic Diagram

ZHAL20X8A



ZHAL24A Evaluation #4

```
Title     ZHAL24A Evaluation 4 (74ACT461)
Pattern   P7023
Revision  B
Author    Birkner/Kazmi/Blasco
Company   Monolithic Memories, Inc.
Date      1986

CHIP ZHAL24A_Evaluation_4 PAL20X8

CK   I0 D0 D1 D2 D3 D4 D5 D6 D7  I1 GND
/OE /CO Q7 Q6 Q5 Q4 Q3 Q2 Q1 Q0 /CI VCC

EQUATIONS

/Q0 :=  /I1*/I0                                     ;CLEAR LSB
     +       I0*/Q0                                 ;COUNT/HOLD
    :+:  I1*/I0*/D0                                 ;LOAD D0 (LSB)
     +   I1* I0* CI                                 ;COUNT

/Q1 :=  /I1*/I0                                     ;CLEAR
     +       I0*/Q1                                 ;COUNT/HOLD
    :+:  I1*/I0*/D1                                 ;LOAD D1
     +   I1* I0* CI*Q0                              ;COUNT

/Q2 :=  /I1*/I0                                     ;CLEAR
     +       I0*/Q2                                 ;COUNT/HOLD
    :+:  I1*/I0*/D2                                 ;LOAD D2
     +   I1* I0* CI*Q0*Q1                           ;COUNT

/Q3 :=  /I1*/I0                                     ;CLEAR
     +       I0*/Q3                                 ;COUNT/HOLD
    :+:  I1*/I0*/D3                                 ;LOAD D3
     +   I1* I0* CI*Q0*Q1*Q2                        ;COUNT

/Q4 :=  /I1*/I0                                     ;CLEAR
     +       I0*/Q4                                 ;COUNT/HOLD
    :+:  I1*/I0*/D4                                 ;LOAD D4
     +   I1* I0* CI*Q0*Q1*Q2*Q3                     ;COUNT

/Q5 :=  /I1*/I0                                     ;CLEAR
     +       I0*/Q5                                 ;COUNT/HOLD
    :+:  I1*/I0*/D5                                 ;LOAD D5
     +   I1* I0* CI*Q0*Q1*Q2*Q3*Q4                  ;COUNT

/Q6 :=  /I1*/I0                                     ;CLEAR
     +       I0*/Q6                                 ;COUNT/HOLD
    :+:  I1*/I0*/D6                                 ;LOAD D6
     +   I1* I0* CI*Q0*Q1*Q2*Q3*Q4*Q5               ;COUNT

/Q7 :=  /I1*/I0                                     ;CLEAR MSB
     +       I0*/Q7                                 ;COUNT/HOLD
    :+:  I1*/I0*/D7                                 ;LOAD D7 (MSB)
     +   I1* I0* CI*Q0*Q1*Q2*Q3*Q4*Q5*Q6            ;COUNT

 IF (VCC)  CO = CI*Q0*Q1*Q2*Q3*Q4*Q5*Q6*Q7          ;CARRY OUT
```

TB02020M

Logic Cell Array™

Introduction

The Logic Cell Array (LCA)™ is a CMOS integrated circuit with a flexible, uncommitted architecture and VLSI-level density. The LCA is manufactured on Monolithic Memories' 1.6 micron CMOS process. The device architecture as is shown in Figure 1, similar to that of a gate array, with an interior matrix of programmable logic blocks, a surrounding ring of I/O interface blocks and programmable interconnect used to define the overall device structure.

Unlike gate arrays, Logic Cell Array functionality is defined by the user simply by loading the internal writable storage cells with the configuration data. An additional benefit, reprogrammability in system, allows in-circuit-emulation to be used for design verification.

The M2064 family of Logic Cell Arrays has been developed to allow Monolithic Memories to offer a device and technology that offer both the density benefits of gate arrays and the programmability benefits of user-configurable devices. These parts have been designed for maximum flexibility in system applications and are easy to use.

Using the XACT™ software development system, the designer can define and interconnect logic blocks to build larger-scale, multi-level logic functions. These are then connected to external circuitry. Interconnections throughout the Array are defined automatically by the development system, unless otherwise specified by the designer. Because the Logic Cell Array's logic functions and interconnections are established with memory cells, the array is never physically altered; instead it is simply reprogrammed.

XACT Evaluation Kit (LCA-MEK01)

Monolithic Memories offers evaluation software and documentation that will allow a designer to determine if his or her logic design fits and assess its performance as a Logic Cell Array. All that is needed is an IBM PC-XT, AT, or compatible, a three-button Mouse System or compatible mouse.

XACT Development System (LCA-MDS21)

The XACT Development System is the "power behind the machine." It will allow a designer to sit down with a concept and walk away with a completely tested, completely finished part.

The reason is that XACT functions as both a CAE and CAM system. The CAE part of the system allows the designer to simply draw out the design using a sophisticated graphics-based design editor. The CAM part then converts the drawing to code, similar to a PALASM-generated JEDEC file, that allows programming with conventional programming hardware of an EPSOM containing the configuration data for system phototyping.

The XACT Development System currently has 113 macros and, in addition, allows the user to define his or her own macro. To insure that internal timing constraints are satisfied, a Timing Analyzer is included to calculate propagation delays along any path within the Array.

As the design is being entered, the Automatic Design Checker insures that no design rules are violated, and when the design is completed, a final design rule check is performed.

The XACT Development System

Contents:

- Editor
- Macro Library
- Design Checker
- Timing Analyzer
- Configuration File Generator
- Configuration File Formatter

In-Circuit-Emulator (LCA-MDS24)

The In-Circuit-Emulator is a software/hardware package that enables a designer to connect his or her target system to the work-station where a design has just been completed. The emulator package allows:

- User control and monitoring of device function
- Interactive or file-driven setup and configuration
- Daisy-chain configuration capabilities for up to seven LCAs in a chain
- Simultaneous in-circuit emulation of up to four devices
- Single step capability for device clocks
- Readback display of device internal register states
- Dynamic reconfiguration capability.

The In-Circuit-Emulator comes with a single "pod". Up to three additional pods (LCA-MDS25) may be ordered for each emulator.

P-SILOS Simulation Package (LCA-MDS22)

After a design is completed, the next step is to simulate. Monolithic Memories offers an integrated simulation package manufactured by Simucad, called P-SILOS.

PART NUMBER	DESCRIPTION
LCA-MEK01	LCA Evaluation Kit
LCA-MDS21	LCA Development system
LCA-MDS22	LCA Simulator-P-SILOS
LCA-MDS24	LCA In-Circuit-Emulator
LCA-MDS25	LCA In-Circuit-Emulator Pod
LCA-MTB01	LCA Demonstration Board

Table 1

The introduction of the Logic Cell Array will allow customers to reduce inventories of discrete components, reduce the time to market and development cost for new products, save money in manufacturing and spare parts inventory management, reduce test costs and improve system reliability.

Configurable Logic Block

The core of the Logic Cell Array is an 8 x 8 matrix of Configurable Logic Blocks. Each CLB provides four logic inputs, a clock input, a combinatorial logic section, two logic outputs, and a programmable storage element.

The inputs drive a combinatorial logic section that can perform any logic function from a simple gate to a three-out-of-four majority decoder.

The combinatorial portion accepts and generates both positive- and negative-true logic, eliminating the need for inverters or the routing of complementary signals.

The storage element can serve as a flip-flop (D-type) and can be programmed to have clock enable, synchronous set and reset, and various gated inputs. In addition, since all these options can be specified independently for each logic block, designers can mix asynchronous and synchronous logic in any combination.

Interconnect

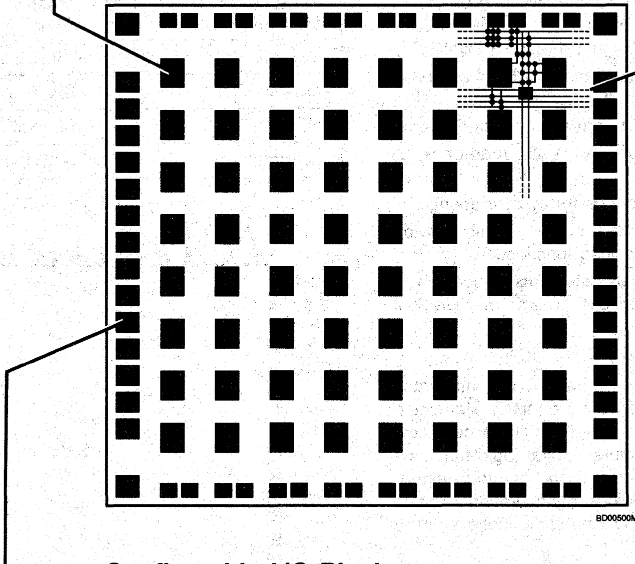
The Array's extraordinary flexibility is also the result of a two-layer metal network of lines that run horizontally and vertically between the logic and I/O blocks, and a variety of user-definable interconnection elements.

Definable interconnection points connect the inputs and outputs of logic and I/O blocks to nearby metal lines.

Crosspoint switches and interchanges are clustered at the intersection of every row and column of logic blocks. They link horizontal and vertical paths and allow signals to be switched from one path to another.

Finally, "long lines" run the length and breadth of the chip, bypassing interchanges but tying into logic blocks and other lines and distributing clocks and other critical signals with a minimum of propagation delay.

Interchanges and interconnection point assignments, as well as all routing are handled automatically by the XACT Development System Software. In addition, special graphics-based design tools are included to facilitate any necessary designer interaction.



Configurable I/O Block

External signals enter and leave the chip through general-purpose, user-definable I/O blocks positioned around the periphery of the array. Each block can be programmed independently to be an input, output or bidirectional pin with a tristate control on the output. When configured as an input, the designer can select TTL or CMOS thresholds. In addition, each I/O bloc contains an input register option whose clock line is common to all the other I/O blocks along the same edge of the die.

I/O blocks can also handle more than input and output functions. For example, the Input registers of unused I/O blocks can be used for read/write storage registers or as stages of a shift register.

Figure 1

Logic Cell Array™ M2064

Features/Benefits

- CMOS programmable Logic Cell Array (LCA™) for replacement of standard logic
- Completely reconfigurable by the user in the final system
- High performance
  - 20MHz flip-flop toggle rate (–20 speed grade)
  - 33MHz flip-flop toggle rate (–33 speed grade)
  - 50MHz flip-flop toggle rate (–50 speed grade)
- User-configurable logic functions, interconnect and I/O for maximum flexibility
- 64 user-Configurable Logic Blocks (CLBs) providing usable gate equivalency of up to 1500 gates
- 58 individually-configurable I/O pins allowing any mix of inputs, outputs or bidirectional signals (68-pin package)
- User-selectable TTL or HCMOS input threshold levels
- Multiple configuration modes for greatest flexibility and ease of use
- Verification feature allows user to check configuration data
- User-selectable security feature prevents read-back of configuration data
- Read-back of internal register states for system debug
- On-chip clock oscillator and clock buffer circuits provide flexible internal and external clocking functions
- Master reset of all internal register elements in addition to user-configurable Reset and/or Set control of individual CLB storage elements

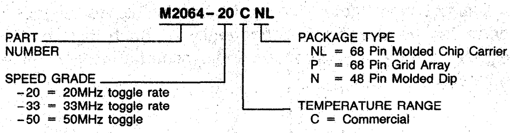
General Description

The M2064 is the first member of a family of configurable Logic Cell Arrays (LCAs) available from Monolithic Memories. These general purpose CMOS integrated circuit devices allow the user to rapidly implement complex digital logic functions directly without the requirement for masking or other vendor performed programming steps. Unique configuration circuitry allows complete reconfiguration within a user's final system to allow system changes "on-the-fly."

User configuration is controlled by internal storage elements. These are loaded with data bits which control definition of logic functions, configuration of I/O blocks, routing of internal signals, and other options. Configuration data can be loaded in one of several methods to minimize impact on overall system design.

CMOS technology optimized for system level performance provides LS-TTL compatible speeds with power consumption less than 10% of equivalent TTL systems. The use of innovative I/O buffers providing either TTL or CMOS input switching levels insures lowest possible power consumption in totally CMOS systems without any compromise in performance.

Ordering Information


M2064 - 20 C NL

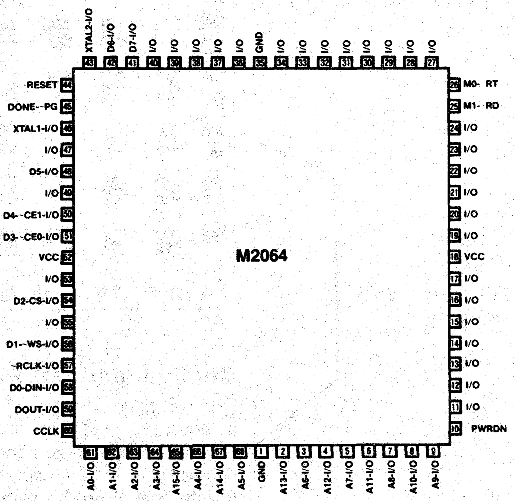
PART NUMBER

SPEED GRADE
-20 = 20MHz toggle rate
-33 = 33MHz toggle rate
-50 = 50MHz toggle

PACKAGE TYPE
NL = 68 Pin Molded Chip Carrier
P = 68 Pin Grid Array
N = 48 Pin Molded Dip

TEMPERATURE RANGE
C = Commercial

PART NUMBER	DESCRIPTION
LCA-MDS21	XACT Development System
LCA-MDS22	P-SILOS Simulation Package
LCA-MDS24	LCA In-Circuit Emulator
LCA-MDS25	In-Circuit Emulator Pod
LCA-MEK01	XACT Evaluation Kit



CD01470M

Portions of this Data Sheet reproduced with the permission of XILINX Inc.

Pin Description

I/O

User-configurable Input/Output pins.

~PWRDN

Input forces device into low power mode; operation is suspended.

M0-~RT

Dual function input. During initial power up, the state of M0 and M1 determines the configuration mode. After configuration, a rising edge on ~RT initiates a configuration read operation.

M1-~RD

Dual function input/output. During initial power up, the state of M1 and M0 determines the configuration mode. After configuration is complete, ~RD outputs configuration data during a configuration read back operation synchronously with the toggling of the CCLK input.

~RESET

Input. A low level on this input after configuration causes all register elements internal to the LCA to be forced to 0. If asserted prior to the start of configuration, causes the LCA to remain in the initialization state (configuration is not started). If asserted during configuration the LCA returns to the initialization state.

DONE-~PG

Dual function output/input. During configuration the LCA pulls DONE low and releases it when configuration is complete (output is open drain). After configuration is complete, a falling edge on ~PG initiates an LCA programming cycle (if enabled in the current configuration). This pin has an internal user-enabled pull-up resistor.

XTAL1-I/O

Dual function input and I/O. This pin may be configured by the user to be a normal I/O pin equivalent to any of the general purpose I/O pins. Alternatively, this pin and XTAL2 my be used to connect a crystal for use with the internal crystal oscillator configuration.

XTAL2-I/O

Dual function output and I/O. This pin may be configured by the user to be a normal I/O pin equivalent to any of the general purpose I/O pins. Alternatively, this pin and XTAL1 may be used to connect a crystal for use with the internal crystal oscillator configuration.

CCLK

Configuration mode dependent input/output. CCLK is the master configuration clock used to configure the LCA. In slave mode it is an input; in all other modes it is an output designed to provide the input clocking of additional slave mode daisy chain connected LCA devices. During a configuration read back operation, CCLK serves as the clock input used to read the internal configuration data.

DOUT-I/O

Dual function output and I/O. General purpose user-configurable I/O pin during normal operation. During configuration, the serial data stream supplied from the first LCA to LCAs down the serial daisy chain is output on DOUT.

~RCLK-I/O

Dual function output and I/O. General purpose user-configurable I/O pin during normal operation. During master mode configuration, a low level output on ~RCLK indicates that the external memory device is being accessed.

D0-DIN-I/O

Multi-function input and I/O. General purpose user-configurable I/O pin during normal operation. During master mode configuration, this pin is bit 0 of the 8-bit parallel input data bus (D0). During slave mode or peripheral mode configuration, this pin is the serial input data pin (DIN).

D1-~WS-I/O

Multi-function input and I/O. General purpose user-configurable I/O pin during normal operation. During master mode configuration, this pin is bit 1 of the 8-bit parallel input data bus (D1). During peripheral mode configuration, a low level on ~WS indicates that a write operation is being performed by the controlling processor. See note.

D2-CS-I/O

Multi-function input and I/O. General purpose user-configurable I/O pin during normal operation. During master mode configuration, this pin is bit 2 of the 8-bit parallel input data bus (D2). During peripheral mode configuration, a high level on CS indicates that a write operation is being performed by the controlling processor. See note.

D3-~CE0-I/O

Multi-function input and I/O. General purpose user-configurable I/O pin during normal operation. During master mode configuration, this pin is bit 3 of the 8-bit parallel input data bus (D3). During peripheral mode configuration, a low level on ~CE0 indicates that a write operation is being performed by the controlling processor. See note.

D4-~CE1-I/O

Multi-function input and I/O. General purpose user-configurable I/O pin during normal operation. During master mode configuration, this pin is bit 4 of the 8-bit parallel input data bus (D4). During peripheral mode configuration, a low level on ~CE1 indicates that a write operation is being performed by the controlling processor. See note.

D5-I/O to D7-I/O

Input and I/O. General purpose user-configurable I/O pins during normal operation. During master mode configuration, these pins are bits 5 through 7 of the 8-bit parallel input data bus (D5-D7).

A0-I/O to A15-I/O

Output and I/O. General purpose user-configurable I/O pins during normal operation. During master mode configuration these pins are address output pins (A0-A15) used to address the external storage element used for configuration data.

Note: to perform a peripheral mode write, the following logical combination is necessary:~WS·CS·CE0·CE1.

Functional Description

The M2064 is a high-performance CMOS Logic Cell Array providing superior system performance with greatest user flexibility. Complete user-configurability provides an optimized solution to logic implementation problems.

The M2064 utilizes a unique Configurable Logic Block (CLB) structure as the basic functional building block of the device. Each CLB is a combination of a programmable logic function and a storage element. The CLB has the capability of performing any function of its inputs with the option of the output of the storage element included in the input field. User-defined logic is implemented in a matrix of sixty-four CLBs which are interconnected with user-configurable interconnect resources.

Fifty-eight independently configurable I/O Blocks; each of which can be a direct or latched input, a direct or open drain output, or a bidirectional I/O buffer; provide the interface to external circuits. Input voltage levels are user definable and may be either standard TTL or CMOS for all I/O Blocks, depending on the user's configuration choice.

User-definable path selector or multiplexers are utilized to select configuration options for the CLBs and I/O Blocks. These selectors are set in the desired state by the configuration data loaded into the device upon power up.

Logic

User logic is implemented in one or more CLBs which are general purpose 4-input, 2-output elements. Figure 1 shows a block diagram of a single CLB. Each element is composed of a 4-input combinational logic module with two outputs, a general purpose storage element, and routing selection logic. The module can generate any combinational logic function of the four inputs, or it can generate any two independent functions of any three of the four inputs. If a function of four inputs is selected, that same function will be available on both of the outputs of the combinational module. The inputs to the combinational module are three of the four inputs to the CLB (A, B and C) and either the D input to the CLB, or the Q output of the storage element.

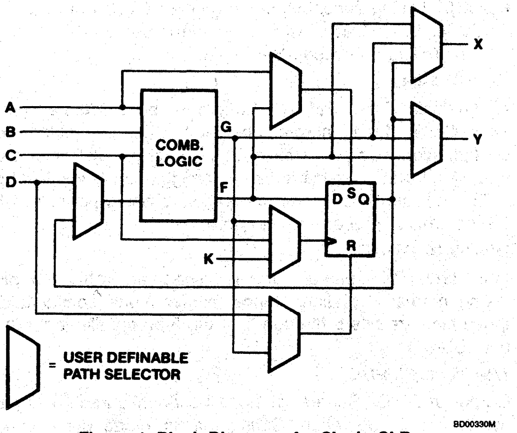


Figure 1. Block Diagram of a Single CLB

The general purpose storage element has a data input, a clock input, a set direct input, a reset direct input and an output, Q. The storage characteristic may be defined as either a transparent latch or as an edge-triggered flip-flop. The data input is connected to one of the outputs of the combinational logic module. The set direct and reset direct inputs may be individually enabled or disabled.

The reset direct input, if enabled, may come from either the D input to the CLB, or from the G output of the combinational logic module. Set direct control, if enabled, can come from either the A input to the CLB or from the F output of the combinational logic module.

Clock for the storage element may be individually enabled or disabled and can be driven by the clock input, K, to the CLB, the C input to the CLB, or the G output of the combinational logic module. Final outputs, X and Y, from the CLB can be selected to be either of the two outputs, F and G, of the combinational logic module, or the Q output from the storage element.

I/O Elements

The M2064 contains fifty-eight user-configurable I/O blocks for connection to external circuits. Each block is a general purpose device containing a three-state output buffer, an input buffer, and an input flip-flop as shown in Figure 2. The input buffer always reflects the status of the I/O pin or the contents of the input flip-flop. If the flip-flop is selected, data present on the I/O pin will be clocked to the input buffer by the I/O block clock signal. All I/O blocks on a particular edge of the device share a common I/O clock signal. The output buffer may be enabled, disabled, or under the control of the three-state connection.

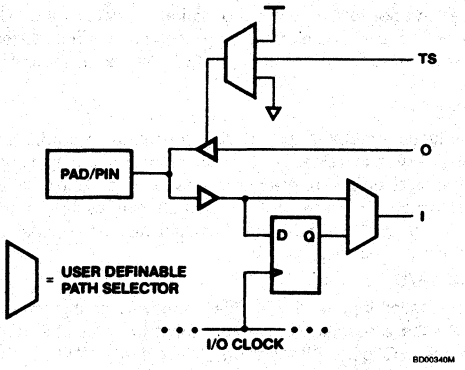


Figure 2. Block Diagram of an I/O Element

Interconnect

There are nine rows and nine columns of metal interconnect resources with one row or column located between each row or column of CLBs or I/O blocks. Each row or column of interconnect resources contains local use lines, long lines and programmable interconnect points between I/O blocks, CLBs and interconnect resources. Local lines run either vertically from one row of resources to the adjacent row or horizontally from one column to the adjacent column. Long lines run the full height or width of the device. At the intersection of every row and column of interconnect resources are user configurable interconnect elements which allow multiple combinations of connections between local lines in adjacent rows and columns. In addition, selected intersections of local and long lines can be connected by user programmable interconnect points.

Inputs and outputs from each CLB or I/O block have programmable connections to the interconnect resources in the adjacent rows and columns. These connections allow CLBs or I/O block connections to be made for proper signal routing to or from the I/O blocks or CLBs. In addition to the programmable connections to adjacent interconnect resources, there are direct connection paths which do not utilize the general interconnect resources. These paths allow selected connection between some I/O blocks and CLBs and between adjacent CLBs. For example, the outputs of a CLB in the interior of the matrix of CLBs may be connected to adjacent CLBs without using any interconnect resources.

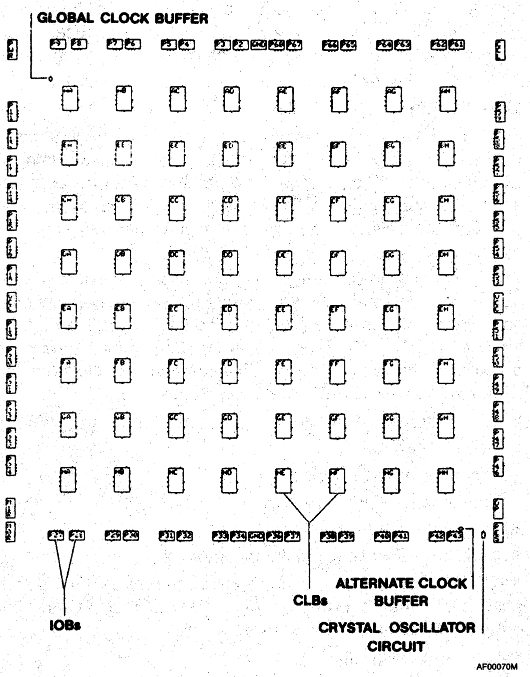
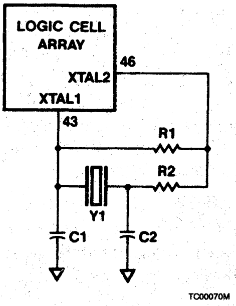


Figure 3. Overview Functional Layout of the M2064

Clock Generation and Buffering

The M2064 contains two special purpose clock buffers for generating and driving clock signals to multiple CLBs or I/O blocks with negligible skew. The Global Clock Buffer, is dedicated to driving a matrix of long lines which have configurable connections to the K input of each CLB register. This clock buffer may be driven from an internally generated register source, or configured with a connection directly to an I/O block for driving it with an external clock signal. The output from the Global Clock Buffer may be configured to directly drive an I/O block for driving clock signals off the device.

The alternate clock buffer can be configured either as a simple buffer or as a buffer for the crystal oscillator. In the crystal oscillator mode, an externally connected crystal and optional passive components form a clock generator for use on the chip or for driving other external circuits (see Figure 4). When configured in the buffer mode, the alternate clock buffer can have either one or both of its input and output configured to directly drive, or be driven by, an I/O block. The output of the alternate clock buffer can drive long lines in any column of CLBs as well as local interconnect.



Suggested component values:

R1	1 - 4MΩ
R2	0 - 1KΩ (may be required for low frequency, phase shift and/or compensate level for crystal Q)
C1, C2	5 - 20pF
Y1	1 - 10MHz AT cut

Figure 4. Crystal Oscillator

Each CLB has a special clock input (K) which can be selected as the clock input of the storage element. Clock inputs to user selected CLBs can be configured to be driven from either the Global Clock Buffer, the oscillator/buffer or from other local interconnect. Clocks to the I/O blocks can be configured from either of the clock buffers or the local interconnect.

Programming

Configuration of the device may be performed in any one of three modes. The desired configuration mode is set by the state of the mode pins M0 and M1 at power up (see Table 1). All configuration data relating to CLB function definition, interconnect resource utilization, and I/O block programming must be loaded into the device prior to use. In the peripheral and slave modes the data is supplied in a serial stream in conjunction with the configuration clock signal, CCLK. In master mode, the device automatically loads data from an external memory device by supplying addresses and reading bytes of data. In all modes the data patterns required to create a specific configuration are the same.

MODE SELECT PINS	M0	M1
Master LOW mode	0	0
Master HIGH mode	0	1
Peripheral mode	1	0
Slave mode	1	1

Note: During configuration, Pin 27 on the 68-pin package or Pin 7 on the 48-pin package must be held HIGH.

Table 1. Modes

Data patterns for a specific user-configuration are created with the Monolithic Memories XACT LCA Development System and can be output to a standard EPROM programmer or saved on disk for inclusion with other software. Users who are using the Monolithic Memories XACT debugging system can directly access the configuration data and load the device directly during a debug session. Because of the complexity of the data patterns and difficulty in generating them without a thorough knowledge of the device, users are discouraged from attempting to generate data patterns on their own. Data pattern files for M2064 devices contain 1536 bytes.

Special Features

The M2064 contains several special features which enhance its capacity for use in a wide variety of applications. Among these are the following:

Data Security

The M2064 configuration data contains special controls bits which enable or disable configuration data security control. If enabled, the security control will prevent the read-back of configuration data after the initial configuration. There are two possible modes of operation under security control. One mode allows a single read-back after configuration to allow verification of the data. In the second mode, all access to the configuration data is prevented.

Reprogrammability

Configuration data changes are controlled by reprogramming control bits in the configuration data supplied to the device. If reprogramming is enabled, the user may supply new configuration data at any time by asserting the correct control sequence on the DONE-~PG and M0 and M1 mode control pins. Alternatively, the user may elect to prevent reconfiguration of the device. When operating in this mode, the only method to remove the configuration is to remove all power from the device.

Inactive Power-down

In a system which is to remain in its current configuration through power loss, the M2064 may be forced into a low power inactive state by using the ~PWRDN pin. When held low, the LCA will retain all configuration data but will not operate. All clocks will be stopped and all outputs put into a high-impedance state. Power is reduced to a very low level, allowing a simple external battery arrangement to supply the required configuration data saving power (see electrical characteristics).

Configuration Data Read-back

A mechanism is provided in the M2064 to provide verification of stored configuration data. The configuration read-back is initiated by toggling the ~RT pin and clocking the CCLK pin. Each clock applied to CCLK will read out a configuration data bit on the ~RD pin. When all configuration data has been read out, the ~RD pin will return to its inactive state. The configuration data may be read at any time with no effect on the operation of the device. Once a configuration read-back has been initiated, all data must be read out of the device to insure that subsequent read-back operations will begin at the start of the configuration data.

Master Reset

After device configuration, the ~RESET pin becomes a master reset for all CLB and IOB storage elements in the device. Asserting this control signal will asynchronously reset all of the internal storage registers regardless of the operating condition of the circuit.

Development System

The Monolithic Memories Design System is an integrated package of design tools for developing configuration data for LCAs. All aspects of configuration are specified through interactive graphics software. Facilities to verify functionality and timing of the designed configuration insure that designs operate as desired.

XACT is a graphic design system used to specify LCA designs. It contains several standard and several optional software and hardware packages. The basic package runs on an IBM PC/XT or AT compatible computer with 640K memory, a color monitor and a mouse. The tools accessible from the executive, including the optional packages, are:

- LCA Editor and Macros
- Timing Analyzer
- Simulator (P-Silos, optional)
- Configuration-File Generator
- Configuration-File Formatter
- XACTOR 2 In-circuit-emulator (optional).

XACTOR 2 consists of a software program plus a hardware attachment that allows control of up to four LCAs. The program contains commands for:

- Loading configuration data
- Activating the Master Reset input
- Reconfiguring
- Single stepping the device clock
- Reading back configuration data and state of all 122 internal registers on any clock cycle.
- Real time system debug.

An evaluation kit is available which includes:

- Complete documentation of the Development System
- A sample LCA design
- XACT software package.

Contact your local Monolithic Memories Representative or Distributor for more information.

Absolute Maximum Ratings*

Supply voltage V_{CC}	−0.5V to 7V
Power down V_{CC}	2V to 7V
Input voltage	−0.5V to V_{CC} 0.5V
Voltage applied to three-state output	−0.5V to V_{CC} 0.5V
Storage temperature range	−65°C to +150°C
Lead temperature (soldering, 10 seconds)	260°C

Note: Stresses beyond those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. These are stress ratings only and functional operation of the device at these or any other conditions beyond those listed under "Recommended Operating Conditions" is not implied. Exposure to "Absolute Maximum Ratings" conditions for extended periods of time may affect device reliability.

Operating Conditions

SYMBOL	PARAMETER	MIN	TYP	MAX	UNIT
V_{CC}	Supply voltage relative to GND	4.75		5.25	V
V_{IHT}	High level input voltage — TTL configuration	2.0			V
V_{IHC}	High level input voltage — CMOS configuration	0.7			V
V_{ILT}	Low level input voltage — TTL configuration	0		0.8	V
V_{ILC}	Low level input voltage — CMOS configuration	0		0.2	V
I_{IT}	Input leakage current — TTL configuration			± 1	μA
I_{IC}	Input leakage current — CMOS configuration			± 1	μA
I_{OZ}	Three-state output off current (V_{CC} = 5.5V)			± 10	μA
t_{OP}	Operating free-air temperature	0		70	°C

Electrical Characteristics Over Operating Conditions

SYMBOL	PARAMETER		TEST CONDITION		MIN	TYP	MAX	UNIT
V_{OH}	High level output voltage		V_{CC} = 4.75V	I_{OH} = −4.0mA	3.86			V
V_{OL}	Low level output voltage		V_{CC} = 4.75V	I_{OL} = 4.0mA			0.32	V
I_{CCO}	Quiescent operating power supply current	CMOS inputs	V_{CC}=5.0V				5	mA
		TTL inputs	V_{CC}=5.0V				10	mA
I_{CCPD}	Power down supply current		V_{CC}=2.0V		Consult factory			V

Note: All switching characteristics are at worst case conditions.

Power on Timing

The M2064 contains on-chip reset timing logic for power-up operation. To insure proper master mode system operation, VCC must rise from 2.0V to minimum specification level in 10ms or less. For other modes, initiation of configuration must be delayed for 60ms after VCC reaches the minimum specified level.

Switching Characteristics — CLB

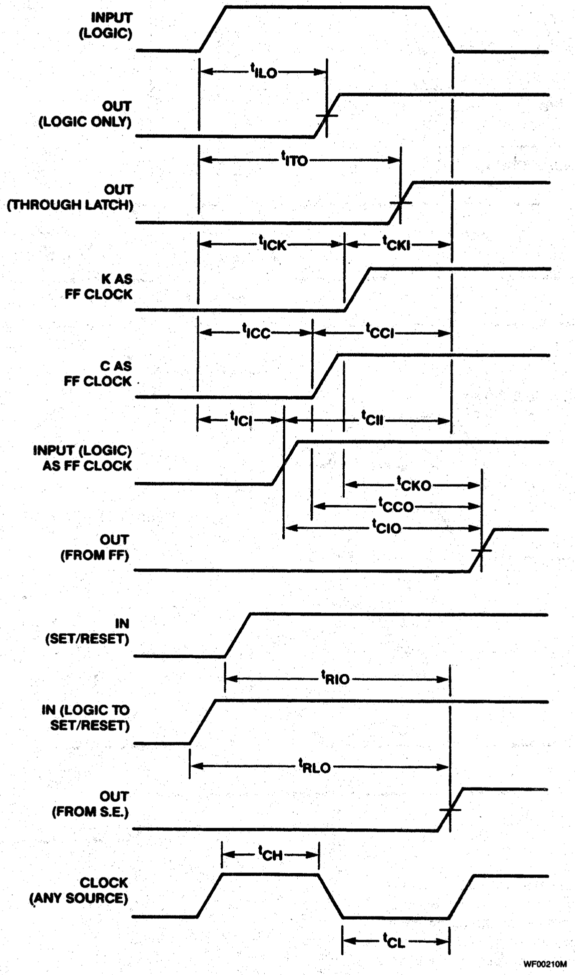
ITEM	DESCRIPTION	M2064 - 50			M2064 - 33			M2064 - 20			UNIT
		MIN	TYP	MAX	MIN	TYP	MAX	MIN	TYP	MAX	
t_{ILO}	Input through logic to X or Y			15			20			35	ns
t_{ITO}	Input through logic and latch to X or Y			20			25			45	ns
t_{CKO}	Storage element clock from K to output			15			20			35	ns
t_{CCO}	Storage element clock from C to output			19			25			45	ns
t_{CIO}	Logic to storage element clock to output			27			37			65	ns
t_{QLO}	Storage element Q to logic to output			8			13			30	ns
t_{ICK}	Input setup to K clock to storage element	8			12			22			ns
t_{CKI}	Input hold to K clock to storage element	0			0			0			ns
t_{ICC}	Input setup to C clock to storage element	9			12			18			ns
t_{CCI}	Input hold to C clock to storage element	0			6			10			ns
t_{ICI}	Input setup to input clock to storage element	4			6			10			ns
t_{CII}	Input hold to input clock to storage element	5			9			15			ns
t_{RIO}	Input to storage element Reset/Set to output			22			25			45	ns
t_{RLO}	Logic to storage element Reset/Set to output			28			37			65	ns
t_{RPW}	Reset/Set pulse width	9			12			20			ns
t_{RS}	Storage element control separation	9			17			30			ns
t_{CH}	Storage element clock high time	9			12			20			ns
t_{CL}	Storage element clock low time	9			12			20			ns
f_{CLK}	Storage element clock frequency			50			33			20	MHz
t_{MRQ}	Master reset to storage element Q to output			25			35			60	ns

Cross Reference Guide

XILINX	MONOLITHIC MEMORIES	V_{CC}		F_{MAX}
		MIN	MAX	
XC-2064-1		4.5V	5.5V	20MHz
	M2064-20	4.75V	5.25V	20MHz
XC-2064-2		4.5V	5.5V	33MHz
XC-2064-33	M2064-33	4.75V	5.25V	33MHz
XC-2064-50	M2064-50	4.75V	5.25V	50MHz

*For commercial product only.

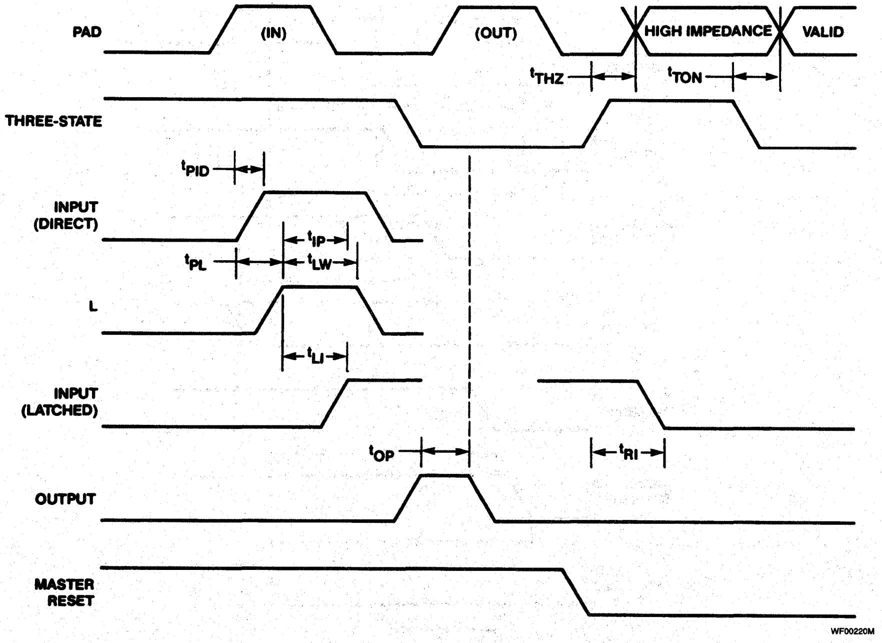
Switching Characteristics — CLB



Notes: 1. Input refers to CLB inputs A, B, C or D
2. Output refers to CLB output X or Y
3. Clock refers to the CLB storage element clock
4. FF (Flip-Flop or L (Latch) refers to the CLB storage element
5. Set and Reset refer to CLB storage element controls

Switching Characteristics — IOB

ITEM	DESCRIPTION	M2064 - 50			M2064 - 33			M2064 - 20			UNIT
		MIN	TYP	MAX	MIN	TYP	MAX	MIN	TYP	MAX	
t_{PID}	Pad to input direct			8			12			20	ns
t_{PL}	Pad input setup to I/O clock	8			12			20			ns
t_{LP}	Pad input hold to I/O clock	0			0			0			ns
f_L	Frequency, I/O clock			50			33			20	MHz
t_{LW}	Pulse width input latch clock	9			12			20			ns
t_{LI}	Input latch clock to input			15			20			30	ns
t_{OP}	Output to pad output (three-state enabled)			12			15			25	ns
t_{THZ}	Three-state inactive to high impedance			20			25			35	ns
t_{TON}	Three-state active to output on			20			25			40	ns
t_{RI}	Master Reset to latched input reset			25			30			50	ns



Notes: 1. Output (O) refers to the output connection on the IOB
2. Input (I) refers to the input connection on the IOB
3. Three-state (T) refers to the three-state control on the IOB
4. Pad or Pin (P) refers to the device pin connected to the IOB
5. Latch (L) refers to the input Flip-Flop clock connection

Switching Characteristics — Programming — Slave Mode

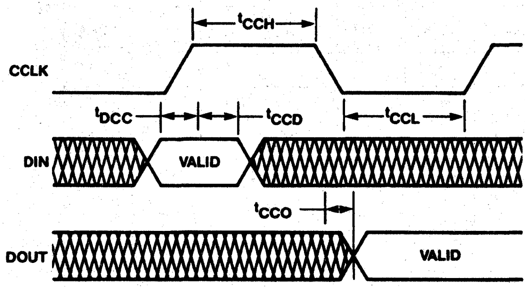
ITEM	DESCRIPTION	M2064 - 50			M2064 - 33			M2064 - 20			UNIT
		MIN	TYP	MAX	MIN	TYP	MAX	MIN	TYP	MAX	
t_{CCH}	CCLK high time	300			3000			500			ns
t_{CCL}	CCLK low time	200			200			300			ns
t_{CCL}	CCLK low time			5000			5000			10000	ns
t_{DCC}	DIN data setup to CCLK rising edge	25			25			50			ns
t_{CCD}	DIN data hold from CCLK rising edge	40			40			75			ns
t_{CCO}	DOUT data delay from CCLK falling edge			65			65			100	ns
f_{CC}	CCLK maximum frequency			2			2			1	MHz

Switching Characteristics — Programming — Peripheral Mode

ITEM	DESCRIPTION	M2064 - 50			M2064 - 33			M2064 - 20			UNIT
		MIN	TYP	MAX	MIN	TYP	MAX	MIN	TYP	MAX	
t_{CA}	Control input active period	200			200			300			ns
t_{CA}	Control input active period			5000			5000			10000	ns
t_{CI}	Control input inactive period	150			150			250			ns
t_{CCC}	CCLK delay from control input edge			75			75			100	ns
t_{DC}	DIN setup to control transition	35			35			50			ns
t_{CD}	DIN hold from control transition	5			5			10			ns

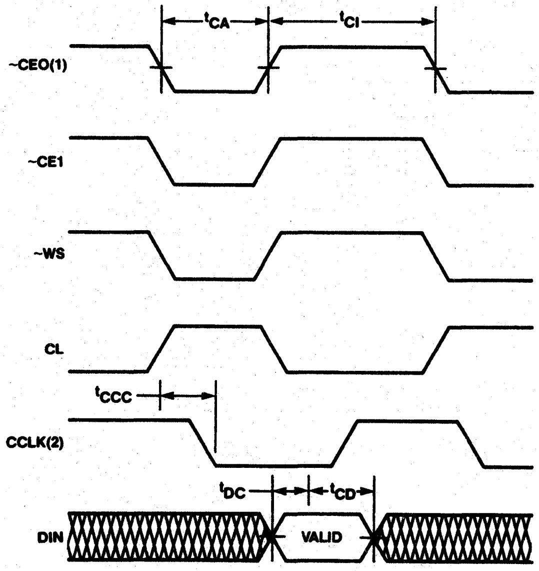
Notes: 1. Peripheral mode timing determined from last control signal (~CEO,~WS, CS) to transition to active or inactive state.
2. CCLK timing minima and maxima same as for slave mode.
3. CCLK and DOUT timing same as for slave mode.

Slave Mode



WF00230M

Peripheral Mode

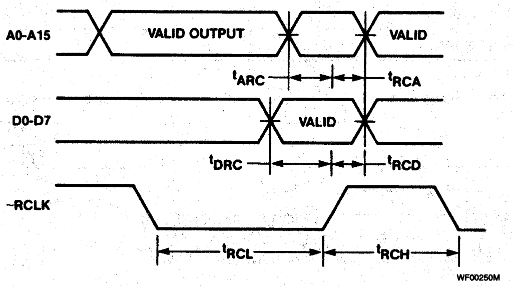


WF00240M

Switching Characteristics — Programming — Master Mode

ITEM	DESCRIPTION	M2064 - 50			M2064 - 33			M2064 - 20			UNIT
		MIN	TYP	MAX	MIN	TYP	MAX	MIN	TYP	MAX	
t_{ARC}	Address invalid prior to ~RCLK edge			0			0			0	ns
t_{RCA}	Address valid from ~RCLK edge			200			200			300	ns
t_{DRC}	Data bus setup to ~RCLK edge	60			60			100			ns
t_{RCD}	Data bus hold from ~RCLK edge	0			0			0			ns
t_{RCH}	~RCLK high	600			600			600			ns
t_{RCL}	~RCLK low	4000			4000			4000			ns

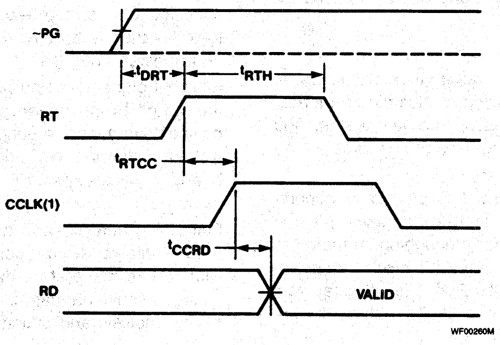
Note: Timing for DOUT and CCLK out is same as for slave mode.



Switching Characteristics — Program Readback

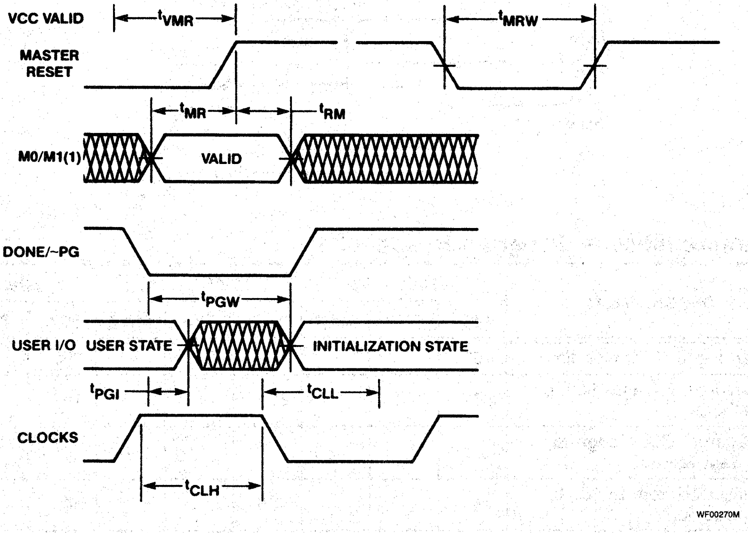
ITEM	DESCRIPTION	M2064 - 50			M2064 - 33			M2064 - 20			UNIT
		MIN	TYP	MAX	MIN	TYP	MAX	MIN	TYP	MAX	
t_{RTH}	Read trigger (RT) high time	150			150			250			ns
t_{RTCC}	Delay from RT assertion to first CCLK	60			60			100			ns
t_{CCRD}	Delay from CCLK edge to RD data valid			60			60			100	ns
t_{DRT}	Wait period from DONE to RT assertion	75			175			300			ns

Notes: 1. Timing for CCLK is same as for slave programming mode.
2. DONE/~PG output/input must be high (device programmed) prior to assertion of ~PG.



Switching Characteristics — General

ITEM	DESCRIPTION	M2064 - 50			M2064 - 33			M2064 - 20			UNIT
		MIN	TYP	MAX	MIN	TYP	MAX	MIN	TYP	MAX	
t_{VMR}	Master Reset delay from valid V_{CC}	150			150			250			ns
t_{MRW}	Master Reset pulse width	150			150			250			ns
t_{MR}	Mode control setup to Master Reset	60			60			100			ns
t_{RM}	Mode control hold from Master Reset	7			7			10			ns
t_{PGW}	Program control pulse width	6000			6000			6000			ns
t_{PGI}	Program control to I/O initialized			7000			7000			7000	ns
t_{CLH}	Clock buffer input high time	9			12			20			ns
t_{CLL}	Clock buffer input low time	9			12			20			ns
f_{CL}	Clock buffer input frequency			50			33			20	MHz



Test Conditions:

Outputs loaded with rated DC current and 50-pF capacitance to GND

Design Aids

XACT™ provides complete design automation for users to specify and implement designs utilizing Monolithic Memories' LCA products. Configuration of CLBs, internal routing, I/O block definitions and global routing are all handled in an integrated, easy-to-use system.

Placement and routing of logic and I/O blocks is accomplished using interactive graphics. Final programming bit patterns are automatically produced for debugging, transfer to other systems, or downloading to standard EPROM programmers. Debugging with the XACTOR 2™ emulation system allows full device emulation and operation analysis in the target system.

The user inputs data via a graphics-oriented physical editing environment. User functions are translated into CLB logic specifications and interconnections automatically. Standard logic libraries and other macro capabilities can be utilized for rapid design entry.

Physical placement and hard connections are performed with the graphics placement and connection capabilities. Final device layout and routing are visible and can be modified without disturbing the logical arrangement. Logical connectivity and physical layout rules checking are performed automatically.

Full timing analysis and functional simulation of user configurations allows device performance and functional checking without external test hardware. In addition, point-to-point path timing calculation capability is provided to simplify general timing analysis and critical path determination.

The debugging system provides full emulation in the user's target system. User configurations may be transferred directly from the design phase into the target system and tested through the emulation system.

Additional capabilities include a tool for automatic hard copy generation of the user's logic design and physical implementation and a tool for transferring programming bit patterns to EPROM programming systems.

Recommended Sockets

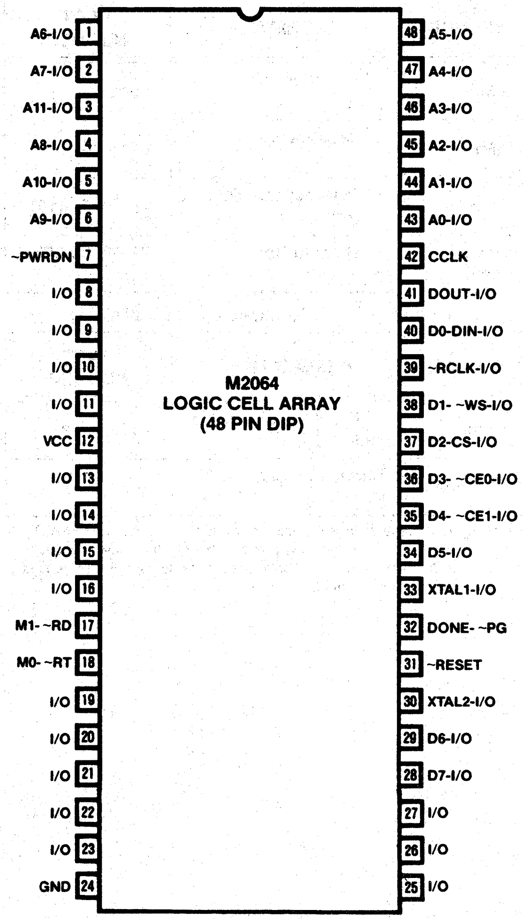
The following 68-pin PLCC sockets have been evaluated by Monolithic Memories and have been found to exhibit acceptable connectivity, device retention and device extraction characteristics:

VENDOR	DESCRIPTION	PART NUMBER
AMP	PCB solder tail, tin plate	821574-1
AMP	Surface mount, tin plate	821542-1
Burndy	PCB solder tail, tin plate	QILE68P-410T
Midland-Ross	PCB solder tail, tin plate	709-2000-068-4-1-1

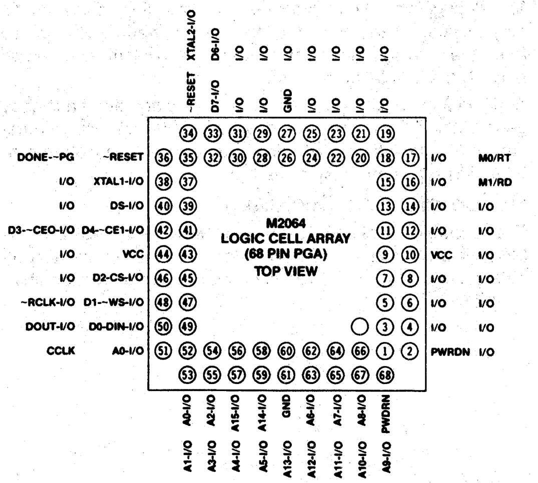
The following 68-pin PLCC sockets are also available:

VENDOR	DESCRIPTION	PART NUMBER
Methode	PCB solder tail, tin plate	213-068-001
Methode	Surface mount, tin plate	213-068-002

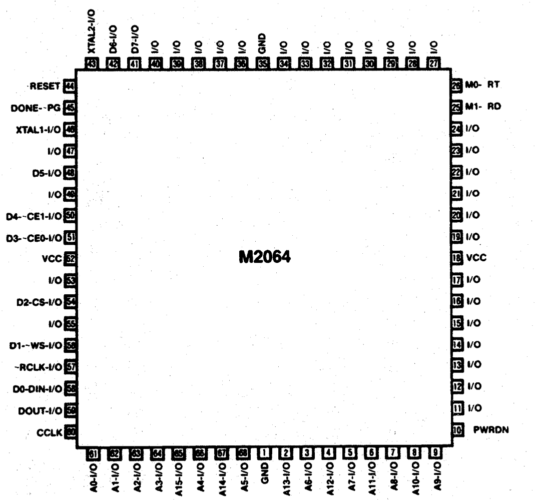
Notes:
1. The standard device extraction tools supplied with the XACT development system will work with all of the above sockets.
2. All of the above PCB solders tail sockets have the same hold pattern and pinout.



CD01480M



CD01490M



CD01470M

PALASM® 2 Software

Introducing PALASM®2 Software

PALASM 2 software is a package that turns PAL device Design Specification (PDS) files into input files for PAL device programmers. PDS is a format for specifying a PAL circuit and for creating inputs to a logic circuit. Using a text editor, you create a PDS file that describes a PAL circuit. PALASM 2 software accepts the file as input and performs a number of functions under your control including:

- Assembling PAL Design Specifications.
- Generating PAL device fuse patterns in JEDEC format.
- Reporting errors in syntax and assembly.
- Allowing concise mnemonic names for long, frequently used logic expressions through string substitution.

Functional differences from PALASM 1

PALASM 2 software is quite different from PALASM 1 in implementation. It is composed of several interacting programs coupled by disk files. (Floppy based files slow interaction. We recommend RAM or hard disks for production use.) The principal benefit of the reorganization is the freedom from fixed limits within the design file.

The syntax of PALASM 2 software is significantly different from that of PALASM 1. PALASM 2 software allows description of asynchronous devices like the 20RA10 and devices of much higher complexity such as MegaPAL devices.

The current version of PALASM 2 software omits several features provided within PALASM 1. They are:

- Fault coverage prediction for test vectors.
- Automatic generation of documentation.
- Device signal/pinout display.
- Support of the security fuse.
- Printing of logic equations for each product term in a fuse plot.

Some of these omissions represent a change in philosophy; others will be provided in later versions of the program.

Required Equipment

T ıis section describes computers and PAL device programmers supported by PALASM 2 software and provides information about necessary and optional PALASM 2 software.

Computers

PALASM 2 software operates with no user modification on the following CPUs, provided certain minimum system requirements are satisfied. It is usually provided as an executable program, ready to run on any of these systems:

Minicomputers: VAX™ under VMS™

Microcomputers: IBM-PC™, -XT™, -AT™ under MS-DOS™ (256K RAM)

All systems must have a serial port (RS-232) for communication with the PAL device programmer. We also recommend that floppy disk based systems be equipped with two disk drives.

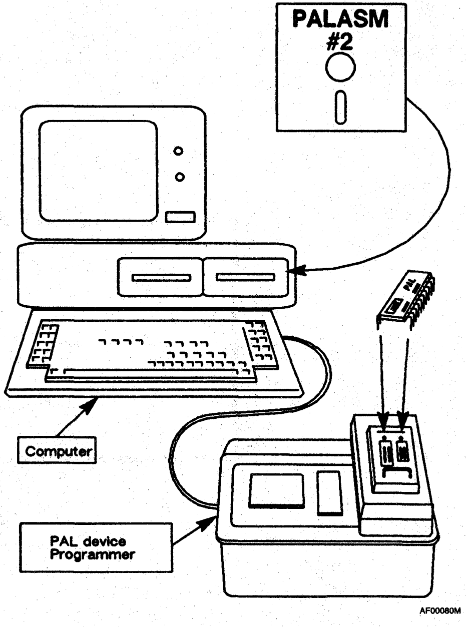


Figure 1. Typical Computer Configuration

NOTE

Please refer to a PALASM 2 software order form for the correct part number of the version of PALASM 2 software designed for your CPU.

Software

Following is a summary of all currently available programs.

1. PDSCNVT	PALASM 1 to PALASM 2 syntax conversion
2. PALASM 2	PALASM 2 syntax parser
3. XPLOT	PALASM 2 fuse map and JEDEC output
4. SIM	PALASM 2 simulator
5. ZHAL	ZHAL device fit

Supplementary Software

1. MENU	Simplified PALASM 2 user interface
2. PC2	Programmer interface program
3. VTRACE	Graph display simulator trace output

Files

1.	.PDS	User PALASM 2 PLD design description
2. PALASM 2	.TRE	PLD intermediate design description
3.	.PDF	PLD architecture description data
4.	.XPT	Contains PLD fuse map data
5.	.JED	Contains PLD fuse JEDEC data
6.	.HST	Contains full simulation history data
7.	.TRF	Contains user simulation trace data
8.	.JDC	Contains both PLD fuse JEDEC data and JEDEC test vectors

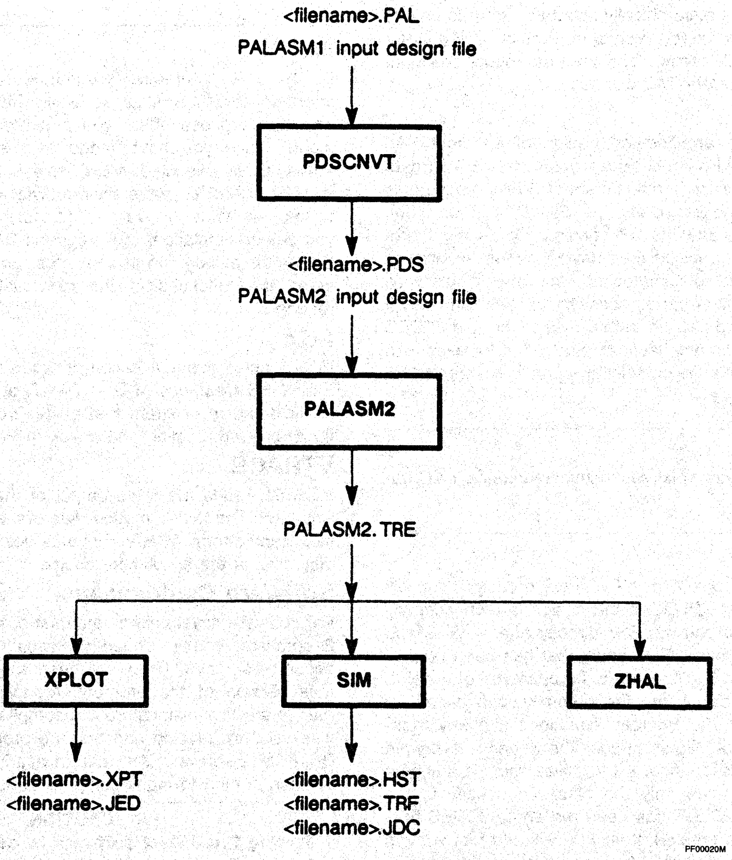


Figure 2. PALASM Software Flow

PDSCNVT

PDSCNVT allows you to interactively convert PAL device design specifications from the PALASM 1 format to PALASM 2 software. Input is a PALASM 1 formatted specification file, and output is the equivalent design in PALASM 2 software syntax.

PALASM 2

PALASM 2 is the first program you will use in the PALASM 2 software suite. It reads and validates your input — a PAL device design specification — for correct design syntax. If an error is detected, the program attempts to indicate where in the input description the error has occurred. Recovery is attempted after each error in order to catch as many errors as possible on a single run. Only if no error is detected is an intermediate specification file generated. This file contains the input specification in a hierarchically structured form to enable easy processing by follow-on programs. Further, it is guaranteed to be syntactically correct. This program recognizes input descriptions for all current PAL devices.

XPLOT

XPLOT validates the architectural design of an input PAL device description and produces fuse maps and JEDEC data for a specified PAL device. Input is a set of Boolean equations that has been preprocessed by the PALASM 2 program. XPLOT checks the equations for consistency among themselves and with the specified PAL device. When an error is detected, XPLOT attempts immediate recovery. In this way, XPLOT spots as many errors as possible on each run. Only if no errors are detected will the output fuse maps and JEDEC data be generated. The architectural information for each PAL device is read in from a file containing a profile description for the specific PAL device.

NOTE

XPLOT will check only valid Monolithic Memories PAL devices.

SIM

SIM checks the functionality of a PAL device design. You will run this program after XPLOT. If the design is architecturally correct, however, you can run SIM directly after PALASM 2. SIM reads a special simulation syntax that has been preprocessed by PALASM 2. It will simulate the operation of the PAL device you specify, calculating the output values based on input signals through the Boolean equations and any feedback. Output is a history file that traces the values of every pin through a simulation sequence. A trace file, which is a subset of the history file, traces only the pins you specify in the simulation syntax. If XPLOT has been run and a JEDEC fuse address file has been created, then SIM will add test vectors to the JEDEC file that duplicate the simulation sequence when the device is tested on a programmer. All JEDEC checksums are recalculated.

NOTE

SIM will test only valid Monolithic Memories PAL devices.

ZHAL

ZHAL makes sure that a PAL device description will fit into the ZHAL device architecture. ZHAL reads the description that has been preprocessed by the PALASM 2 software program, and a YES or NO answer is output. If the design fits, you must send the PAL device Design Specification to Monolithic Memories for mask processing. Before a description is rejected, the program will attempt to minimize the input equations to make use of some sharing features in the ZHAL architecture. ZHAL will also indicate what the error is before a no fit answer is output.

NOTE

ZHAL currently accepts designs only for valid Monolithic Memories PAL devices.

MENU

MENU is an interactive program designed to simplify user interface to PALASM 2 software. MENU's multiple choice selection process offers you a number of options at each stage. Once you have made a selection, your choice is automatically executed. MENU makes the modular design of the PALASM 2 software system invisible to you. All you need to know is what you want to do, not how-to. For example, if you select simulation, the programs PALASM 2, XPLOT and SIM automatically run in sequence, and your need to understand the functions of the individual programs is greatly reduced.

PC2

PC2 enables communication between PLD programmers and IBM^{TM} PC machines (-PC, -XT, -AT, etc.). It is a menu-driven multiple-choice program that guides you through various options for programming and checking PLD devices.

VTRACE

VTRACE reads the trace output of the PALASM 2 software simulator. The text-formatted data of the trace file is converted into graphic form. VTRACE output looks very much like timing diagrams of the simulation results.

Software Customization

For software development and user customization of PALASM 2 software, you will need a Pascal compiler/linker and a second disk drive. These are necessary to create the executable version of the program. Monolithic Memories recommends the Professional Pascal compiler (Microtek Inc.) which was used to develop and test the programs on the IBM-PC. PALASM 2 software is written in nearly ISO Standard Pascal to ease porting to many computer systems.

CAUTION

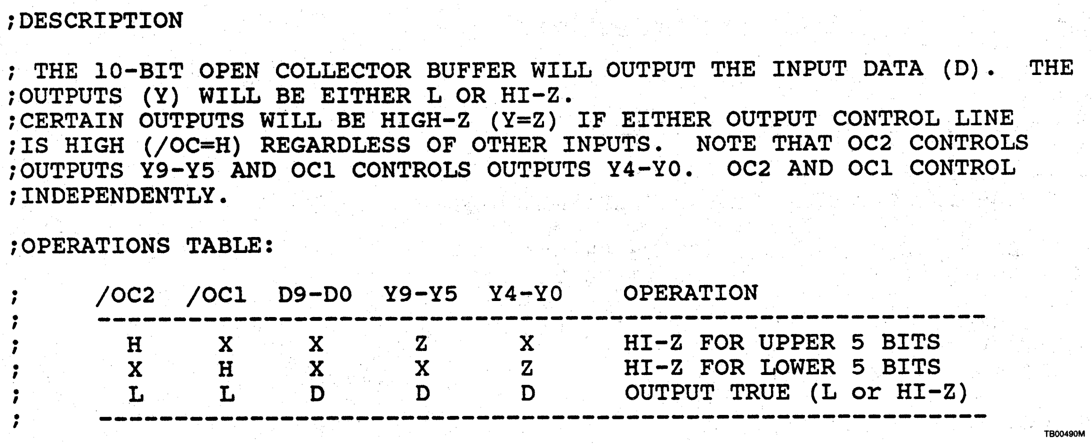
Porting PALASM 2 software to other computer systems will require you to install and modify the original source code. Allow at least two to four weeks of software engineering time to complete this task. Monolithic Memories makes no guarantee of the portability of PALASM 2 software, and does not provide support for such efforts or other user modification.

Table of Contents

10-Bit Open Collector Buffer

```
Title    10-BIT OPEN COLLECTOR BUFFER
Pattern  P7070
Revision A
Author   VINCENT COLI
Company  MMI SUNNYVALE, CALIFORNIA
Date     10/21/81

CHIP 10-BIT_OPEN_COLLECTOR_BUFFER PAL20L10

/OC1 D0 D1 D2 D3 D4 D5 D6 D7 D8 D9 GND
/OC2 Y9 Y8 Y7 Y6 Y5 Y4 Y3 Y2 Y1 Y0 VCC

EQUATIONS

IF(OC1*/D0)  /Y0 = VCC                ;Y0= D0 (TRUE)

IF(OC1*/D1)  /Y1 = VCC                ;Y1= D1 (TRUE)

IF(OC1*/D2)  /Y2 = VCC                ;Y2= D2 (TRUE)

IF(OC1*/D3)  /Y3 = VCC                ;Y3= D3 (TRUE)

IF(OC1*/D4)  /Y4 = VCC                ;Y4= D4 (TRUE)

IF(OC2*/D5)  /Y5 = VCC                ;Y5= D5 (TRUE)

IF(OC2*/D6)  /Y6 = VCC                ;Y6= D6 (TRUE)

IF(OC2*/D7)  /Y7 = VCC                ;Y7= D7 (TRUE)

IF(OC2*/D8)  /Y8 = VCC                ;Y8= D8 (TRUE)

IF(OC2*/D9)  /Y9 = VCC                ;Y9= D9 (TRUE)

;FUNCTION TABLE FOR PALASM 1
;/OC2 /OC1 D9 D8 D7 D6 D5 D4 D3 D2 D1 D0
;Y9 Y8 Y7 Y6 Y5 Y4 Y3 Y2 Y1 Y0

;       INPUT DATA    OUTPUT DATA
;/OC    DDDDDDDDDD    YYYYYYYYYY
;2 1    9876543210    9876543210     COMMENTS
;-------------------------------------------------------------------
;L L    LLLLLLLLLL    LLLLLLLLLL     TEST L's (TRUE D)
;L L    HHHHHHHHHH    ZZZZZZZZZZ     TEST H's (TRUE D)
;L L    HLHLHLHLHL    ZLZLZLZLZL     TEST EVEN CHECKERBOARD (TRUE D)
;L L    LHLHLHLHLH    LZLZLZLZLZ     TEST ODD  CHECKERBOARD (TRUE D)
;H L    XXXXXLLLLL    ZZZZZLLLLL     TEST HI-Z FOR /OC2=H
;L H    LLLLLXXXXX    LLLLLZZZZZ     TEST HI-Z FOR /OC1=H
;--------------------------------------------------------------------
```

TB00480M



```
;DESCRIPTION

; THE 10-BIT OPEN COLLECTOR BUFFER WILL OUTPUT THE INPUT DATA (D).  THE
;OUTPUTS (Y) WILL BE EITHER L OR HI-Z.
;CERTAIN OUTPUTS WILL BE HIGH-Z (Y=Z) IF EITHER OUTPUT CONTROL LINE
;IS HIGH (/OC=H) REGARDLESS OF OTHER INPUTS.  NOTE THAT OC2 CONTROLS
;OUTPUTS Y9-Y5 AND OC1 CONTROLS OUTPUTS Y4-Y0.  OC2 AND OC1 CONTROL
;INDEPENDENTLY.

;OPERATIONS TABLE:

;     /OC2  /OC1  D9-D0  Y9-Y5  Y4-Y0    OPERATION
;     -----------------------------------------------------------
;      H     X     X      Z      X       HI-Z FOR UPPER 5 BITS
;      X     H     X      X      Z       HI-Z FOR LOWER 5 BITS
;      L     L     D      D      D       OUTPUT TRUE (L or HI-Z)
;     -----------------------------------------------------------
```

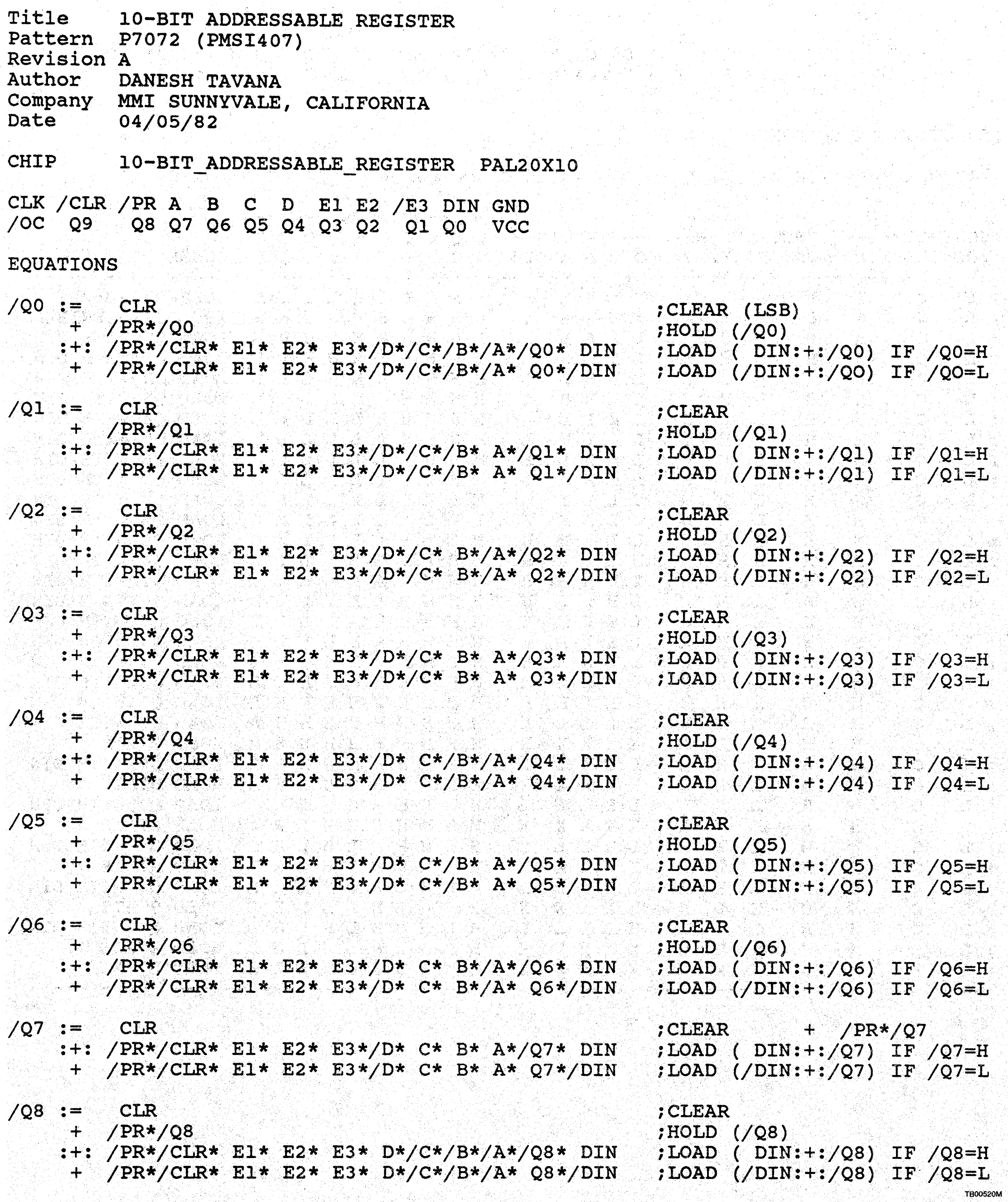
TB00490M

```
Title    10-BIT OPEN COLLECTOR INVERTING BUFFER
Pattern  P7071
Revision A
Author   VINCENT COLI
Company  MMI SUNNYVALE, CALIFORNIA
Date     10/21/81

CHIP 10-BIT_OPEN_COLLECTOR_INVERTING_BUFFER PAL20L10

/OC1 D0 D1 D2 D3 D4 D5 D6 D7 D8 D9 GND
/OC2 Y9 Y8 Y7 Y6 Y5 Y4 Y3 Y2 Y1 Y0 VCC

EQUATIONS

IF(OC1* D0)  /Y0 = VCC                  ;Y0= /D0 (COMP)

IF(OC1* D1)  /Y1 = VCC                  ;Y1= /D1 (COMP)

IF(OC1* D2)  /Y2 = VCC                  ;Y2= /D2 (COMP)

IF(OC1* D3)  /Y3 = VCC                  ;Y3= /D3 (COMP)
IF(OC1* D4)  /Y4 = VCC                  ;Y4= /D4 (COMP)

IF(OC2* D5)  /Y5 = VCC                  ;Y5= /D5 (COMP)

IF(OC2* D6)  /Y6 = VCC                  ;Y6= /D6 (COMP)

IF(OC2* D7)  /Y7 = VCC                  ;Y7= /D7 (COMP)

IF(OC2* D8)  /Y8 = VCC                  ;Y8= /D8 (COMP)

IF(OC2* D9)  /Y9 = VCC                  ;Y9= /D9 (COMP)

;FUNCTION TABLE FOR PALASM 1

;/OC2 /OC1 D9 D8 D7 D6 D5 D4 D3 D2 D1 D0
;          Y9 Y8 Y7 Y6 Y5 Y4 Y3 Y2 Y1 Y0

;       INPUT DATA    OUTPUT DATA;/OC    DDDDDDDDDD    YYYYYYYYYY
;2 1    9876543210    9876543210     COMMENTS
;----------------------------------------------------------------------
;L L    LLLLLLLLLL    ZZZZZZZZZZ     TEST L's (COMP D)
;L L    HHHHHHHHHH    LLLLLLLLLL     TEST H's (COMP D)
;L L    HLHLHLHLHL    LZLZLZLZLZ     TEST EVEN CHECKERBOARD (COMP D)
;L L    LHLHLHLHLH    ZLZLZLZLZL     TEST ODD  CHECKERBOARD (COMP D)
;H L    XXXXXHHHHH    ZZZZZLLLLL     TEST HI-Z FOR /OC2=H
;L H    HHHHHXXXXX    LLLLLZZZZZ     TEST HI-Z FOR /OC1=H
;----------------------------------------------------------------------
```

TB00500M

```
;DESCRIPTION

;THE 10-BIT INVERTING OPEN COLLECTOR BUFFER WILL OUTPUT THE
;COMPLEMENT OF THE INPUT DATA (D).  THE OUTPUTS (Y) WILL BE
;EITHER L OR HI-Z.
;CERTAIN OUTPUTS WILL BE HIGH-Z (Y=Z) IF EITHER OUTPUT CONTROL
;LINE IS HIGH;(/OC=H) REGARDLESS OF OTHER INPUTS.  NOTE THAT
;OC2 CONTROLS OUTPUTS Y9-Y5 AND OC1 CONTROLS OUTPUTS Y4-Y0.
;OC2 AND OC1 CONTROL INDEPENDENTLY.

;OPERATIONS TABLE:

;     /OC2  /OC1  D9-D0  Y9-Y5  Y4-Y0    OPERATION
;     ------------------------------------------------------------
;       H     X     X      Z      X      HI-Z FOR UPPER 5 BITS
;       X     H     X      X      Z      HI-Z FOR LOWER 5 BITS
;       L     L     D     /D     /D      OUTPUT COMP (L or HI-Z)
;     ------------------------------------------------------------
```

TB00510M



```
Title    10-BIT ADDRESSABLE REGISTER
Pattern  P7072 (PMSI407)
Revision A
Author   DANESH TAVANA
Company  MMI SUNNYVALE, CALIFORNIA
Date     04/05/82

CHIP     10-BIT_ADDRESSABLE_REGISTER  PAL20X10

CLK /CLR /PR A  B  C  D  E1 E2 /E3 DIN GND
/OC  Q9   Q8 Q7 Q6 Q5 Q4 Q3 Q2  Q1 Q0  VCC

EQUATIONS

/Q0 :=   CLR                                        ;CLEAR (LSB)
     +  /PR*/Q0                                     ;HOLD (/Q0)
    :+: /PR*/CLR* E1* E2* E3*/D*/C*/B*/A*/Q0* DIN   ;LOAD ( DIN:+:/Q0)  IF /Q0=H
     +  /PR*/CLR* E1* E2* E3*/D*/C*/B*/A* Q0*/DIN   ;LOAD (/DIN:+:/Q0)  IF /Q0=L

/Q1 :=   CLR                                        ;CLEAR
     +  /PR*/Q1                                     ;HOLD (/Q1)
    :+: /PR*/CLR* E1* E2* E3*/D*/C*/B* A*/Q1* DIN   ;LOAD ( DIN:+:/Q1)  IF /Q1=H
     +  /PR*/CLR* E1* E2* E3*/D*/C*/B* A* Q1*/DIN   ;LOAD (/DIN:+:/Q1)  IF /Q1=L

/Q2 :=   CLR                                        ;CLEAR
     +  /PR*/Q2                                     ;HOLD (/Q2)
    :+: /PR*/CLR* E1* E2* E3*/D*/C* B*/A*/Q2* DIN   ;LOAD ( DIN:+:/Q2)  IF /Q2=H
     +  /PR*/CLR* E1* E2* E3*/D*/C* B*/A* Q2*/DIN   ;LOAD (/DIN:+:/Q2)  IF /Q2=L

/Q3 :=   CLR                                        ;CLEAR
     +  /PR*/Q3                                     ;HOLD (/Q3)
    :+: /PR*/CLR* E1* E2* E3*/D*/C* B* A*/Q3* DIN   ;LOAD ( DIN:+:/Q3)  IF /Q3=H
     +  /PR*/CLR* E1* E2* E3*/D*/C* B* A* Q3*/DIN   ;LOAD (/DIN:+:/Q3)  IF /Q3=L

/Q4 :=   CLR                                        ;CLEAR
     +  /PR*/Q4                                     ;HOLD (/Q4)
    :+: /PR*/CLR* E1* E2* E3*/D* C*/B*/A*/Q4* DIN   ;LOAD ( DIN:+:/Q4)  IF /Q4=H
     +  /PR*/CLR* E1* E2* E3*/D* C*/B*/A* Q4*/DIN   ;LOAD (/DIN:+:/Q4)  IF /Q4=L

/Q5 :=   CLR                                        ;CLEAR
     +  /PR*/Q5                                     ;HOLD (/Q5)
    :+: /PR*/CLR* E1* E2* E3*/D* C*/B* A*/Q5* DIN   ;LOAD ( DIN:+:/Q5)  IF /Q5=H
     +  /PR*/CLR* E1* E2* E3*/D* C*/B* A* Q5*/DIN   ;LOAD (/DIN:+:/Q5)  IF /Q5=L

/Q6 :=   CLR                                        ;CLEAR
     +  /PR*/Q6                                     ;HOLD (/Q6)
    :+: /PR*/CLR* E1* E2* E3*/D* C* B*/A*/Q6* DIN   ;LOAD ( DIN:+:/Q6)  IF /Q6=H
     +  /PR*/CLR* E1* E2* E3*/D* C* B*/A* Q6*/DIN   ;LOAD (/DIN:+:/Q6)  IF /Q6=L

/Q7 :=   CLR                                        ;CLEAR      +  /PR*/Q7
    :+: /PR*/CLR* E1* E2* E3*/D* C* B* A*/Q7* DIN   ;LOAD ( DIN:+:/Q7)  IF /Q7=H
     +  /PR*/CLR* E1* E2* E3*/D* C* B* A* Q7*/DIN   ;LOAD (/DIN:+:/Q7)  IF /Q7=L

/Q8 :=   CLR                                        ;CLEAR
     +  /PR*/Q8                                     ;HOLD (/Q8)
    :+: /PR*/CLR* E1* E2* E3* D*/C*/B*/A*/Q8* DIN   ;LOAD ( DIN:+:/Q8)  IF /Q8=H
     +  /PR*/CLR* E1* E2* E3* D*/C*/B*/A* Q8*/DIN   ;LOAD (/DIN:+:/Q8)  IF /Q8=L
```

TB00520M

```
/Q9 :=    CLR                                         ;CLEAR (MSB)
     +   /PR*/Q9                                      ;HOLD (/Q9)
    :+: /PR*/CLR* E1* E2* E3* D*/C*/B* A*/Q9* DIN    ;LOAD ( DIN:+:/Q9) IF /Q9=H
     +  /PR*/CLR* E1* E2* E3* D*/C*/B* A* Q9*/DIN    ;LOAD (/DIN:+:/Q9) IF /Q9=L


;FUNCTION TABLE FOR PALASM 1

;/OC CLK /CLR /PR /E3 E2 E1 D C B A DIN Q9 Q8 Q7 Q6 Q5 Q4 Q3 Q2 Q1 Q0

;                                           -----OUTPUTS-------
;CONTROL -----FUNCTIONS----  ---INPUTS--  Q Q Q Q Q Q Q Q Q Q
;/OC CLK /CLR /PR /E3 E2 E1  D C B A DIN  9 8 7 6 5 4 3 2 1 0  COMMENTS
;---------------------------------------------------------------------------
; L   C    L    L   X  X  X   X X X X  X   L L L L L L L L L L  /CLR OVRRD /PR
; L   C    H    L   X  X  X   X X X X  X   H H H H H H H H H H  /PR OVRRD ENABLE
; L   C    H    H   L  H  H   L L L L  L   H H H H H H H H H L  LOAD Q0 WITH DIN
; L   C    H    H   L  H  H   L L L H  L   H H H H H H H H L L  LOAD Q1 WITH DIN
; L   C    H    H   L  H  H   L L H L  L   H H H H H H H L L L  LOAD Q2 WITH DIN
; L   C    H    H   L  H  H   L L H H  L   H H H H H H L L L L  LOAD Q3 WITH DIN
; L   C    H    H   L  H  H   L H L L  L   H H H H H L L L L L  LOAD Q4 WITH DIN
; L   C    H    H   L  H  H   L H L H  L   H H H H L L L L L L  LOAD Q5 WITH DIN
; L   C    H    H   L  H  H   L H H L  L   H H H L L L L L L L  LOAD Q6 WITH DIN
; L   C    H    H   L  H  H   L H H H  L   H H L L L L L L L L  LOAD Q7 WITH DIN
; L   C    H    H   L  H  H   H L L L  L   H L L L L L L L L L  LOAD Q8 WITH DIN
; L   C    H    H   L  H  H   H L L H  L   L L L L L L L L L L  LOAD Q9 WITH DIN
; L   C    H    H   L  H  H   H L L H  H   H L L L L L L L L L  LOAD Q9 WITH DIN
; L   C    H    H   L  H  H   H L L L  H   H H L L L L L L L L  LOAD Q8 WITH DIN
; L   C    H    H   L  H  H   L H H H  H   H H H L L L L L L L  LOAD Q7 WITH DIN
; L   C    H    H   L  H  H   L H H L  H   H H H H L L L L L L  LOAD Q6 WITH DIN
; L   C    H    H   L  H  H   L H L H  H   H H H H H L L L L L  LOAD Q5 WITH DIN
; L   C    H    H   L  H  H   L H L L  H   H H H H H H L L L L  LOAD Q4 WITH DIN
; L   C    H    H   L  H  H   L L H H  H   H H H H H H H L L L  LOAD Q3 WITH DIN
; L   C    H    H   L  H  H   L L H L  H   H H H H H H H H L L  LOAD Q2 WITH DIN
; L   C    H    H   L  H  H   L L L H  H   H H H H H H H H H L  LOAD Q1 WITH DIN
; L   C    H    H   L  H  H   L L L L  H   H H H H H H H H H H  LOAD Q0 WITH DIN
; L   C    H    H   L  L  L   X X X X  X   H H H H H H H H H H  HOLD STATE
; L   C    H    H   L  H  H   L L L L  L   H H H H H H H H H L  LOAD Q0 WITH DIN
; L   C    H    H   L  L  H   X X X X  X   H H H H H H H H H L  HOLD STATE
; L   C    H    H   L  H  H   L L H L  L   H H H H H H H L H L  LOAD Q2 WITH DIN
; L   C    H    H   L  H  L   X X X X  X   H H H H H H H L H L  HOLD STATE
; L   C    H    H   L  H  H   L H L L  L   H H H H H L H L H L  LOAD Q4 WITH DIN
; L   C    H    H   H  L  L   X X X X  X   H H H H H L H L H L  HOLD STATE
; L   C    H    H   L  H  H   L H H L  L   H H H L H L H L H L  LOAD Q6 WITH DIN
; L   C    H    H   H  L  H   X X X X  X   H H H L H L H L H L  HOLD STATE
; L   C    H    H   L  H  H   H L L L  L   H L H L H L H L H L  LOAD Q8 WITH DIN
; L   C    H    H   H  H  H   X X X X  X   H L H L H L H L H L  HOLD STATE
; H   X    X    X   X  X  X   X X X X  X   Z Z Z Z Z Z Z Z Z Z  TEST HI-Z
;---------------------------------------------------------------------------
```

TB00530M

```
;DESCRIPTION

;THE 10-BIT ADDRESSABLE REGISTER IS A SYNCHRONOUS GENERAL PURPOSE ADDRESSABLE
;REGISTER WITH CLEAR, PRESET, AND ENABLE.  THE OUTPUT REGISTER (Q) IS SELECTED
;BY THE INPUT ADDRESS PINS (A,B,C,D).  THE DATA (DIN) IS LOADED INTO THE
;SELECTED OUTPUT REGISTER ON THE RISING EDGE OF THE CLOCK (CLK) IF THE CHIP IS
;ENABLED (E1=HIGH,E2=HIGH,/E3=LOW).  ALL OTHER OUTPUTS HOLD THEIR PREVIOUS
;STATES DURING THE LOAD OPERATION.  ANY OTHER COMBINATION OF THE ENABLE PINS
;(E1,E2,/E3) WILL DISABLE THE REGISTER AND ALL OUTPUTS WILL HOLD THEIR PREVIOUS
;STATES.  CLEAR (/CLR) AND PRESET (/PR) ARE ACTIVE LOW PINS WHICH SET THE
;REGISTERS TO ALL HIGH OR LOW RESPECTIVELY.

;CLEAR OVERRIDES PRESET AND ENABLE, PRESET OVERRIDES ENABLE.

;THESE FUNCTIONS ARE EXERCISED IN THE FUNCTION TABLE AND SUMMARIZED IN THE
;OPERATIONS TABLE:

;/OC CLK  /CLR /PR /E3 E2 E1  D C B A  DIN  Q9-Q0  OPERATION
;-------------------------------------------------------------------------------
; H   X    X    X   X  X  X   X X X X   X     Z    HI-Z
; L   C    L    L   X  X  X   X X X X   X     L    CLEAR
; L   C    H    L   X  X  X   X X X X   X     H    PRESET
; L   C    H    H   L  L  L   X X X X   X     Q    HOLD PREVIOUS STATES
; L   C    H    H   L  L  H   X X X X   X     Q    HOLD PREVIOUS STATES
; L   C    H    H   L  H  L   X X X X   X     Q    HOLD PREVIOUS STATES
; L   C    H    H   L  H  H   D C B A  DIN   DIN   LOAD DIN TO ADDRESSED OUTPUT
; L   C    H    H   H  L  L   X X X X   X     Q    HOLD PREVIOUS STATES
; L   C    H    H   H  L  H   X X X X   X     Q    HOLD PREVIOUS STATES
; L   C    H    H   H  H  L   X X X X   X     Q    HOLD PREVIOUS STATES
; L   C    H    H   H  H  H   X X X X   X     Q    HOLD PREVIOUS STATES
;-------------------------------------------------------------------------------


;                        OUTPUT SELECT TABLE;

;     D C B A  DIN   Q9  Q8  Q7  Q6  Q5  Q4  Q3  Q2  Q1  Q0
;     ---------------------------------------------------------
;     L L L L  DIN   Q9  Q8  Q7  Q6  Q5  Q4  Q3  Q2  Q1  DIN
;     L L L H  DIN   Q9  Q8  Q7  Q6  Q5  Q4  Q3  Q2  DIN Q0
;     L L H L  DIN   Q9  Q8  Q7  Q6  Q5  Q4  Q3  DIN Q1  Q0
;     L L H H  DIN   Q9  Q8  Q7  Q6  Q5  Q4  DIN Q2  Q1  Q0
;     L H L L  DIN   Q9  Q8  Q7  Q6  Q5  DIN Q3  Q2  Q1  Q0
;     L H L H  DIN   Q9  Q8  Q7  Q6  DIN Q4  Q3  Q2  Q1  Q0
;     L H H L  DIN   Q9  Q8  Q7  DIN Q5  Q4  Q3  Q2  Q1  Q0
;     L H H H  DIN   Q9  Q8  DIN Q6  Q5  Q4  Q3  Q2  Q1  Q0
;     H L L L  DIN   Q9  DIN Q7  Q6  Q5  Q4  Q3  Q2  Q1  Q0
;     H L L H  DIN   DIN Q8  Q7  Q6  Q5  Q4  Q3  Q2  Q1  Q0
;     H L H L  DIN   Q9  Q8  Q7  Q6  Q5  Q4  Q3  Q2  Q1  Q0
;     H L H H  DIN   Q9  Q8  Q7  Q6  Q5  Q4  Q3  Q2  Q1  Q0
;     H H X X  DIN   Q9  Q8  Q7  Q6  Q5  Q4  Q3  Q2  Q1  Q0
;     ---------------------------------------------------------
```

TB00540M

```
Title    MC6800 MICROPROCESSOR INTERFACE
Pattern  P7073
Revision A
Author   COLI/SACKETT
Company  MMI SUNNYVALE, CALIFORNIA
Date     09/14/82

CHIP MC6800_MICROPROCESSOR_INTERFACE PAL20L10

A15  A14 A13  A12   A11    NC    /S    /R     PH2    VMA  /AR GND
ENIN EN  /IO /RAM4 /RAM3 /RAM2 /RAM1 /PROM2 /PROM1 /RSET VUA VCC

EQUATIONS

IF(VCC)  PROM1 =  A15* A14* A13* A12      * VMA* PH2*/RSET    ;PROM1, F000-FFFF

IF(VCC)  PROM2 =  A15* A14* A13*/A12      * VMA* PH2*/RSET    ;PROM2, E000-EFFF

IF(VCC)  RAM1  = /A15*/A14*/A13*/A12*/A11* VMA* PH2*/RSET    ;RAM1,  0000-07FF

IF(VCC)  RAM2  = /A15*/A14*/A13*/A12* A11* VMA* PH2*/RSET    ;RAM2,  0800-0FFF

IF(VCC)  RAM3  = /A15*/A14*/A13* A12*/A11* VMA* PH2*/RSET    ;RAM3,  1000-17FF

IF(VCC)  RAM4  = /A15*/A14*/A13* A12* A11* VMA* PH2*/RSET    ;RAM4,  1800-1FFF

IF(VCC)  IO    =  A15* A14*/A13* A12* A11* VMA* PH2*/RSET    ;I/O,   D800-DFFF

IF(VCC) /EN    = /PROM1*/PROM2*/RAM1*/RAM2*/RAM3*/RAM4*/IO* VMA*/RSET ;EN=/VUA

IF(VCC) /VUA   =  ENIN          ;ASSERTIVE HIGH VUA SIGNAL (INVERT EN FEEDBACK)

IF(VCC)  RSET =  S              ;SET
              + /R * RSET       ;RESET
              + /AR* RSET       ;AUTO RESET


;FUNCTION TABLE FOR PALASM 1

;A15 A14 A13 A12 A11 /S /R /AR /RSET PH2 VMA /PROM1 /PROM2 /RAM1 /RAM2 /RAM3
;/RAM4 /IO EN ENIN VUA

;ADDR1  S-R       /RE            PROM  --RAM--       ENABLE
;54321 /S /R /AR  SET  PH2 VMA   1  2  1 2 3 4  I/O  OUT IN VUA    COMMENT
;-------------------------------------------------------------------------------
;HHHHX  L  H  L    L    L   H    H  H  H H H H   H    H  H   L     RESET (/S=L)
;HHHHX  H  L  H    L    L   H    H  H  H H H H   H    H  H   L     AUTO-RESET
;HHHHX  H  L  L    H    L   H    H  H  H H H H   H    L  L   H     NO SELECT PH2=L
;HHHHX  H  L  L    H    H   L    H  H  H H H H   H    H  H   L     NO SELECT VMA=L
;HHHHX  H  L  L    H    H   H    L  H  H H H H   H    H  H   L     SELECT PROM1
;HHHLX  H  L  L    H    H   H    H  L  H H H H   H    H  H   L     SELECT PROM2
;LLLLL  H  L  L    H    H   H    H  H  L H H H   H    H  H   L     SELECT RAM1
;LLLLH  H  L  L    H    H   H    H  H  H L H H   H    H  H   L     SELECT RAM2
;LLLHL  H  L  L    H    H   H    H  H  H H L H   H    H  H   L     SELECT RAM3
;LLLHH  H  L  L    H    H   H    H  H  H H H L   H    H  H   L     SELECT RAM4
;HHLHH  H  L  L    H    H   H    H  H  H H H H   L    H  H   L     SELECT I/O PORT
;-------------------------------------------------------------------------------
```

TB00550M

```
;DESCRIPTION

;THIS PAL20L10 INTERFACES BETWEEN THE MOTOROLA MC6800 MICROPROCESSOR AND ITS
;SYSTEM COMPONENTS ON A SINGLE BOARD COMPUTER.  THE FUNCTIONS IT PERFORMS,
;PREVIOUSLY DONE WITH RANDOM LOGIC ARE:  ADDRESS DECODING, MEMORY AND I/O
;SELECT, RESET SIGNAL GENERATION, AND CONTROL OF THE BUFFER WHICH INTERFACES
;THE DATA BUS TO OTHER BOARDS IN THE SYSTEM.
```

TB00560M

```
;FUNCTION TABLE FOR PALASM 1

;B A 1C0 1C1 1C2 2C0 2C1 2C2 3C0 3C1 3C2 4C0 4C1 4C2 1Y 2Y 3Y 4Y

; SEL  ------INPUTS------  --OUTPUTS--      COMMENTS
;       1C   2C   3C   4C
; B A  012  012  012  012  1Y 2Y 3Y 4Y
;---------------------------------------------------------
; L L  LHH  HHH  HHH  HHH  L  H  H  H       1C0=0
; L L  HHH  LHH  HHH  HHH  H  L  H  H       2C0=0
; L L  HHH  HHH  LHH  HHH  H  H  L  H       3C0=0
; L L  HHH  HHH  HHH  LHH  H  H  H  L       4C0=0
; L L  HLL  LLL  LLL  LLL  H  L  L  L       1C0=1
; L L  LLL  HLL  LLL  LLL  L  H  L  L       2C0=1
; L L  LLL  LLL  HLL  LLL  L  L  H  L       3C0=1
; L L  LLL  LLL  LLL  HLL  L  L  L  H       4C0=1
; L L  HHH  HHH  HHH  HHH  H  H  H  H       TOGGLE LINES
; L H  HLH  HHH  HHH  HHH  L  H  H  H       1C1=0
; L H  HHH  HLH  HHH  HHH  H  L  H  H       2C1=0
; L H  HHH  HHH  HLH  HHH  H  H  L  H       3C1=0
; L H  HHH  HHH  HHH  HLH  H  H  H  L       4C1=0
; L H  LHL  LLL  LLL  LLL  H  L  L  L       1C1=1
; L H  LLL  LHL  LLL  LLL  L  H  L  L       2C1=1
; L H  LLL  LLL  LHL  LLL  L  L  H  L       3C1=1
; L H  LLL  LLL  LLL  LHL  L  L  L  H       4C1=1
; L H  HHH  HHH  HHH  HHH  H  H  H  H       TOGGLE LINES
; H L  HHL  HHH  HHH  HHH  L  H  H  H       1C2=0
; H L  HHH  HHL  HHH  HHH  H  L  H  H       2C2=0
; H L  HHH  HHH  HHL  HHH  H  H  L  H       3C2=0
; H L  HHH  HHH  HHH  HHL  H  H  H  L       4C2=0
; H L  LLH  LLL  LLL  LLL  H  L  L  L       1C2=1
; H L  LLL  LLH  LLL  LLL  L  H  L  L       2C2=1
; H L  LLL  LLL  LLH  LLL  L  L  H  L       3C2=1
; H L  LLL  LLL  LLL  LLH  L  L  L  H       4C2=1
; H L  HHH  HHH  HHH  HHH  H  H  H  H       TOGGLE LINES
; H H  LLL  LLL  LLL  LLL  H  H  H  H       SELECT = 4
; H H  HHH  HHH  HHH  HHH  H  H  H  H       TOGGLE LINES
;--------------------------------------------------------


;DESCRIPTION

;THIS IS AN EXAMPLE OF A QUAD 3-TO-1 MULTIPLEXER USING A PAL18L4.  SELECT
;LINES A,B ARE ENCODED IN BINARY, WITH A REPRESENTING THE LSB.  THE OUTPUTS
;(Y) ARE ALL HIGH IF THE SELECT LINES ARE BOTH HIGH (B,A=H).

;       OPERATIONS TABLE:

;       INPUT    OUTPUTS
;       SELECT
;        B  A       Y
;       ---------------
;        L  L       C0
;        L  H       C1
;        H  L       C2
;        H  H       H
;       ---------------
```

TB00570M

Quad 3:1 Multiplexer

```
Title    QUAD 3:1 MULTIPLEXER
Pattern  P7074
Revision A
Author   VINCENT COLI
Company  MMI SUNNYVALE, CALIFORNIA
Date     01/05/82

CHIP QUAD_3:1_MULTIPLEXER PAL14L4

1C0 1C1 1C2 2C0 2C1 2C2 3C0 3C1 3C2 GND
4C0 4C1 4C2  4Y  3Y  2Y  1Y  B   A  VCC

EQUATIONS

/1Y  =  /B*/A * /1C0                          ;SELECT INPUT 1C0
     +  /B* A * /1C1                          ;SELECT INPUT 1C1
     +   B*/A * /1C2                          ;SELECT INPUT 1C2

/2Y  =  /B*/A * /2C0                          ;SELECT INPUT 2C0
     +  /B* A * /2C1                          ;SELECT INPUT 2C1
     +   B*/A * /2C2                          ;SELECT INPUT 2C2

/3Y  =  /B*/A * /3C0                          ;SELECT INPUT 3C0
     +  /B* A * /3C1                          ;SELECT INPUT 3C1
     +   B*/A * /3C2                          ;SELECT INPUT 3C2

/4Y  =  /B*/A * /4C0                          ;SELECT INPUT 4C0
     +  /B* A * /4C1                          ;SELECT INPUT 4C1
     +   B*/A * /4C2                          ;SELECT INPUT 4C2
```

TB00580M

```
Title    4-BIT COUNTER WITH REGISTER
Pattern  P7075
Revision A
Author   MIKE VOLPIGNO
Company  MONOLITHIC MEMORIES INC NEWTON, MASSACHUSETTS
Date     11/20/81

CHIP 4-BIT_COUNTER_WITH_REGISTER PAL20X8

CLK D0 D1 D2 D3 /RCLR /RLD SEL /CCLR /CLD EP GND
/OC RCO C3 C2 C1 C0 R3 R2 R1 R0 ET VCC

EQUATIONS

/C0 :=    CCLR                                ;CLEAR COUNTER
     +   /CLD*/CCLR* EP* ET                   ;COUNT
    :+: /CLD*/CCLR*/C0                        ;COUNT
     +    CLD*/CCLR*/R0                       ;LOAD COUNTER FROM REGISTER

/C1 :=    CCLR                                ;CLEAR COUNTER
     +   /CLD*/CCLR* EP* ET* C0               ;COUNT
    :+: /CLD*/CCLR*/C1                        ;COUNT
     +    CLD*/CCLR*/R1                       ;LOAD COUNTER FROM REGISTER

/C2 :=    CCLR                                ;CLEAR COUNTER
     +   /CLD*/CCLR* EP* ET* C0* C1           ;COUNT
    :+: /CLD*/CCLR*/C2                        ;COUNT
     +    CLD*/CCLR*/R2                       ;LOAD COUNTEF FROM REGISTER

/C3 :=    CCLR                                ;CLEAR COUNTER
     +   /CLD*/CCLR* EP* ET* C0* C1* C2       ;COUNT
    :+: /CLD*/CCLR*/C3                        ;COUNT
     +    CLD*/CCLR*/R3                       ;LOAD COUNTER FROM REGISTER

/R0 :=    RCLR                                ;CLEAR REGISTER
     +   /RCLR* RLD*/SEL*/D0                  ;LOAD REGISTER FROM DATA
    :+: /RCLR* RLD* SEL*/C0                   ;LOAD REGISTER FROM COUNTER
     +   /RCLR*/RLD*/R0                       ;HOLD

/R1 :=    RCLR                                ;CLEAR REGISTER
     +   /RCLR* RLD*/SEL*/D1                  ;LOAD REGISTER FROM DATA
    :+: /RCLR* RLD* SEL*/C1                   ;LOAD REGISTER FROM COUNTER
     +   /RCLR*/RLD*/R1                       ;HOLD

/R2 :=    RCLR                                ;CLEAR REGISTER
     +   /RCLR* RLD*/SEL*/D2                  ;LOAD REGISTER FROM DATA
    :+: /RCLR* RLD* SEL*/C2                   ;LOAD REGISTER FROM COUNTER
     +   /RCLR*/RLD*/R2                       ;HOLD

/R3 :=    RCLR                                ;CLEAR REGISTER
     +   /RCLR* RLD*/SEL*/D3                  ;LOAD REGISTER FROM DATA
    :+: /RCLR* RLD* SEL*/C3                   ;LOAD REGISTER FROM COUNTER
     +   /RCLR*/RLD*/R3                       ;HOLD

IF(VCC) /RCO =  ET* C0* C1* C2* C3            ;TERMINAL COUNT
```

TB00590M

```
;FUNCTION TABLE FOR PALASM 1

;CLK /RCLR /RLD SEL /CCLR /CLD EP ET D3 D2 D1 D0
;R3 R2 R1 R0 C3 C2 C1 C0 /RCO

;   /     /
;   R /   C /
; C C R S C C                   R
; L L L E L L E E D--D R--R C--C C
; K R D L R D P T 3  0 3  0 3  0 O          COMMENTS
;-----------------------------------------------------------------------
; C L X X X X X X XXXX LLLL XXXX X      CLEAR REGISTER
; C H L L X X X X HHHH HHHH XXXX X      LOAD REGISTER HI FROM DATA
; C H H X H L X X XXXX HHHH HHHH X      LOAD COUNTER FROM REGISTER
; C H H X H H L H XXXX HHHH HHHH H      ENABLE RCO AND HOLD COUNT
; C H H X H H H H XXXX HHHH LLLL L      COUNT AND ROLLOVER
; C H H X H H H H XXXX HHHH LLLH L      INCREMENT COUNTER
; C H H X H H H H XXXX HHHH LLHL L          "        "
; C H H X H H H H XXXX HHHH LLHH L          "        "
; C H H X H H H H XXXX HHHH LHLL L          "        "
; C H H X H H H H XXXX HHHH LHLH L          "        "
; C H H X H H H H XXXX HHHH LHHL L          "        "
; C H H X H H H H XXXX HHHH LHHH L          "        "
; C H H X H H H H XXXX HHHH HLLL L          "        "
; C H H X H H H H XXXX HHHH HLLH L          "        "
; C H H X H H H H XXXX HHHH HLHL L          "        "
; C H H X H H H H XXXX HHHH HLHH L          "        "
; C H H X H H H H XXXX HHHH HHLL L          "        "
; C H H X H H H H XXXX HHHH HHLH L          "        "
; C H H X H H H H XXXX HHHH HHHL L          "        "
; C H L H H H H L XXXX HHHL HHHL L      LOAD REGISTER FROM COUNTER
; C H H X L X X X XXXX HHHL LLLL L      HOLD REGISTER, CLEAR COUNTER
;-----------------------------------------------------------------------

;DESCRIPTION

;THIS PAL DESIGN SPECIFICATION DESCRIBES A 4-BIT SYNCHRONOUS COUNTER WITH
;4-BIT REGISTER.  DATA CAN BE LOADED TO THE COUNTER FROM THE REGISTER.  IT
;CAN ALSO BE SYNCHRONOUSLY CLEARED.  THE REGISTER CAN BE LOADED FROM EITHER
;THE COUNTER OR THE DATA INPUTS UNDER CONTROL OF THE SEL INPUT.  THE REGISTER
;CAN ALSO BE SYNCHRONOUSLY CLEARED.  THE COUNTER AND REGISTER HAVE A COMMON
;CLOCK FOR SYNCHRONOUS OPERATION.
```

TB00600M

```
Title    9-BIT DOWN COUNTER
Pattern  P7076
Revision A
Author   BIRKNER/COLI/LEE
Company  MMI SUNNYVALE, CALIFORNIA
Date     01/12/83

CHIP 9-BIT_DOWN_COUNTER PAL20X10

CLK  D0 D1 D2 D3 D4 D5 D6 D7 D8 /LD GND
/OC /BO Q8 Q7 Q6 Q5 Q4 Q3 Q2 Q1  Q0 VCC

EQUATIONS

/Q0 :=  /LD*/Q0                                     ;HOLD Q0
     +   LD*/D0                                     ;LOAD D0 (LSB)
    :+: /LD                                         ;COUNT DOWN

/Q1 :=  /LD*/Q1                                     ;HOLD Q1
     +   LD*/D1                                     ;LOAD D1
    :+: /LD*/Q0                                     ;COUNT DOWN

/Q2 :=  /LD*/Q2                                     ;HOLD Q2
     +   LD*/D2                                     ;LOAD D2
    :+: /LD*/Q0*/Q1                                 ;COUNT DOWN

/Q3 :=  /LD*/Q3                                     ;HOLD Q3
     +   LD*/D3                                     ;LOAD D3
    :+: /LD*/Q0*/Q1*/Q2                             ;COUNT DOWN

/Q4 :=  /LD*/Q4                                     ;HOLD Q4
     +   LD*/D4                                     ;LOAD D4
    :+: /LD*/Q0*/Q1*/Q2*/Q3                         ;COUNT DOWN

/Q5 :=  /LD*/Q5                                     ;HOLD Q5
     +   LD*/D5                                     ;LOAD D5
    :+: /LD*/Q0*/Q1*/Q2*/Q3*/Q4                     ;COUNT DOWN

/Q6 :=  /LD*/Q6                                     ;HOLD Q6
     +   LD*/D6                                     ;LOAD D6
    :+: /LD*/Q0*/Q1*/Q2*/Q3*/Q4*/Q5                 ;COUNT DOWN

/Q7 :=  /LD*/Q7                                     ;HOLD Q7
     +   LD*/D7                                     ;LOAD D7
    :+: /LD*/Q0*/Q1*/Q2*/Q3*/Q4*/Q5*/Q6             ;COUNT DOWN

/Q8 :=  /LD*/Q8                                     ;HOLD Q8
     +   LD*/D8                                     ;LOAD D8 (MSB)
    :+: /LD*/Q0*/Q1*/Q2*/Q3*/Q4*/Q5*/Q6*/Q7         ;COUNT DOWN

 BO :=  /LD* Q0*/Q1*/Q2*/Q3*/Q4*/Q5*/Q6*/Q7*/Q8     ;CARRY OUT (ANTICIPATE COUNT)
     +   LD*/D0*/D1*/D2*/D3*/D4*/D5*/D6*/D7*/D8     ;CARRY OUT (ANTICIPATE LOAD)
```

TB00610M

```
                                                                              ;F

;CLK /OC /LD D8 D7 D6 D5 D4 D3 D2 D1 D0 /BO Q8 Q7 Q6 Q5 Q4 Q3 Q2 Q1 Q0

;               DATA IN        DATA  OUT
;  CONTROL     DDDDDDDDD       QQQQQQQQQ
;CLK /OC /LD   876543210  /BO  876543210    COMMENT
-----------------------------------------------------------------
; C   L   L    LLLLLLLLL   L   LLLLLLLLL    LOAD (BORROW)
; C   L   H    XXXXXXXXX   H   HHHHHHHHH    DECREMENT
; C   L   L    LLLLLLLLH   H   LLLLLLLLH    LOAD
; C   L   H    XXXXXXXXX   L   LLLLLLLLL    DECREMENT (BORROW)
; C   L   L    LLLLLLLHH   H   LLLLLLLHH    LOAD
; C   L   H    XXXXXXXXX   H   LLLLLLLHL    DECREMENT
; C   L   L    LLLLLLHHH   H   LLLLLLHHH    LOAD
; C   L   H    XXXXXXXXX   H   LLLLLLHHL    DECREMENT
; C   L   L    LLLLLHHHH   H   LLLLLHHHH    LOAD
; C   L   H    XXXXXXXXX   H   LLLLLHHHL    DECREMENT
; C   L   L    LLLLHHHHH   H   LLLLHHHHH    LOAD
; C   L   H    XXXXXXXXX   H   LLLLHHHHL    DECREMENT
; C   L   L    LLLHHHHHH   H   LLLHHHHHH    LOAD
; C   L   H    XXXXXXXXX   H   LLLHHHHHL    DECREMENT
; C   L   L    LLHHHHHHH   H   LLHHHHHHH    LOAD
; C   L   H    XXXXXXXXX   H   LLHHHHHHL    DECREMENT
; C   L   L    LHHHHHHHH   H   LHHHHHHHH    LOAD
; C   L   H    XXXXXXXXX   H   LHHHHHHHL    DECREMENT
; C   L   L    HHHHHHHHH   H   HHHHHHHHH    LOAD
; C   L   H    XXXXXXXXX   H   HHHHHHHHL    DECREMENT
; C   L   L    HHHHHHHLL   H   HHHHHHHLL    LOAD
; C   L   H    XXXXXXXXX   H   HHHHHHLHH    DECREMENT
; L   L   H    XXXXXXXXX   H   HHHHHHLHH    HOLD
; C   L   H    XXXXXXXXX   H   HHHHHHLHL    DECREMENT
; C   L   H    XXXXXXXXX   H   HHHHHHLLH    DECREMENT
; C   L   H    XXXXXXXXX   H   HHHHHHLLL    DECREMENT
; X   H   X    XXXXXXXXX   Z   ZZZZZZZZZ    TEST HI-Z
;----------------------------------------------------------------

;DESCRIPTION

;THE 9-BIT SYNCHRONOUS COUNTER HAS PARALLEL LOAD, DECREMENT, AND HOLD
;CAPABILITIES.  DATA (D8-D0) IS LOADED INTO THE OUTPUT REGISTER (Q8-Q0) WHEN THE
;LOAD INPUT IS TRUE (/LD=L) AND A POSITIVE EDGE PULSE IS RECEIVED ON THE CLOCK
;PIN (CLK).  THE COUNTER WILL DECREMENT IF A CLOCK PULSE IS RECEIVED WITH THE
;LOAD INPUT BEING FALSE (/LD=H).  THE OPERATION IS A HOLD IF NO CLOCK PULSE IS
;RECEIVED REGARDLESS OF ANY OTHER INPUTS.

;THE BORROW OUT PIN (/BO) SHOWS HOW TO IMPLEMENT A BORROW OUT USING A REGISTER
;BY ANTICIPATED ONE COUNT BEFORE THE TERMINAL COUNT IF COUNTING AND THE TERMINAL
;COUNT IF LOADING.

;OPERATIONS TABLE:

;     /OC   CLK   /LD   D8-D0   Q8-Q0     OPERATION
;     ----------------------------------------------
;      H     X     X      X       Z       HI-Z
;      L     L     X      X      /Q       HOLD
;      L     C     L      D       D       LOAD
;      L     C     H      X   Q MINUS 1   DECREMENT
;     ----------------------------------------------
```

TB00620M

```
Title    REFRESH CLOCK GENERATOR
Pattern  P7077
Revision A
Author   FRANK LEE
Company  MMI SUNNYVALE, CALIFORNIA
Date     08/03/82

CHIP REFRESH CLOCK GENERATOR PAL20X10

CLK /LD  F3 F2 F1 F0 M3 M2 M1 M0 /HOLD GND
/OC RFCK Q0 Q1 Q2 Q3 Q4 Q5 Q6 Q7 /INIT VCC

EQUATIONS

INIT    :=  /Q7*/Q6*/Q5*/Q4*/Q3*/Q2*/Q1*/Q0         ;END OF COUNT
         +  LD                                      ;LOAD

/Q0     :=  /INIT*/Q0                               ;DECREMENT
        :+: /INIT*/HOLD                             ;HOLD

/Q1     :=  /INIT*/Q1                               ;DECREMENT
        :+: /INIT*/HOLD*/Q0                         ;HOLD

/Q2     :=   INIT                                   ;LOAD IN 0
         +  /INIT*/Q2                               ;DECREMENT
        :+: /INIT*/HOLD*/Q0*/Q1                     ;HOLD

/Q3     :=  /INIT*/Q3                               ;DECREMENT
        :+: /INIT*/HOLD*/Q0*/Q1*/Q2                 ;HOLD

/Q4     :=   INIT*/F0                               ;LOAD IN F0
         +  /INIT*/Q4                               ;DECREMENT
        :+: /INIT*/HOLD*/Q0*/Q1*/Q2*/Q3             ;HOLD

/Q5     :=   INIT*/F1                               ;LOAD IN F1
         +  /INIT*/Q5                               ;DECREMENT
        :+: /INIT*/HOLD*/Q0*/Q1*/Q2*/Q3*/Q4         ;HOLD

/Q6     :=   INIT*/F2                               ;LOAD IN F2
         +  /INIT*/Q6                               ;DECREMENT
        :+: /INIT*/HOLD*/Q0*/Q1*/Q2*/Q3*/Q4*/Q5     ;HOLD

/Q7     :=   INIT*/F3                               ;LOAD IN F3
         +  /INIT*/Q7                               ;DECREMENT
        :+: /INIT*/HOLD*/Q0*/Q1*/Q2*/Q3*/Q4*/Q5*/Q6 ;HOLD

/RFCK   :=  M3*/Q7*/Q6* Q5                          ;32 LOW STATES
         +  M2*/Q7*/Q6*/Q5* Q4                      ;16 LOW STATES
        :+: M1*/Q7*/Q6*/Q5*/Q4* Q3                  ;8 LOW STATES
         +  M0*/Q7*/Q6*/Q5*/Q4*/Q3* Q2              ;4 LOW STATES
```

TB00630M

```
;FUNCTION TABLE FOR PALASM 1

;CLK /OC /LD /HOLD F3 F2 F1 F0 M3 M2 M1 M0 /INIT Q7 Q6 Q5 Q4 Q3 Q2 Q1 Q0 RFCK

;      /             /
;      H             I          R
;C / / O             N          F
;L O L L FFFF MMMM   I QQQQQQQQ C
;K C D D 3210 3210   T 76543210 K    COMMEMTS
;--------------------------------------------------------------------------
;C L L X LLHH LLHH   L XXXXXXXX X    READY TO LOAD COUNTER
;C L L H LLHH LLHH   L LLHHHLHH X    READY TO LOAD COUNTER
;C L H H LLHH LLHH   H LLHHHLHH H    COUNTER LOADED, READY TO COUNT DOWN
;C L H H LLHH LLHH   H LLHHHLHL H    COUNT DOWN
;C L H H LLHH LLHH   H LLHHHLLH H    COUNT DOWN
;C L L H LLLH LHHL   L LLHHHLLL H    READY TO LOAD COUNTER, COUNT DOWN
;C L H H LLLH LHHL   H LLLHHLHH H    COUNTER LOADED
;C L H H LLLH LHHL   H LLLHHLHL L    COUNT DOWN
;C L H H LLLH LHHL   H LLLHHLLH L    COUNT DOWN
;C L H H LLLH LHHL   H LLLHHLLL L    COUNT DOWN
;C L L H LLLH LHHL   L LLLHLHHH L    READY TO LOAD COUNTER, COUNT DOWN
;C L H H LLLL LHHL   H LLLLHLHH L    COUNTER LOADED
;C L H H LLLL LHHL   H LLLLHLHL L    COUNT DOWN
;C L H H LLLL LHHL   H LLLLHLLH L    COUNT DOWN
;C L H H LLLL LHHL   H LLLLHLLL L    COUNT DOWN
;C L H H LLLL LHHL   H LLLLLHHH L    COUNT DOWN
;C L H H LLLL LHHL   H LLLLLHHL H    COUNT DOWN, RFCK HIGH
;C L H H LLLL LHHL   H LLLLLHLH H    COUNT DOWN, RFCK HIGH
;C L H L LLLL LHHL   H LLLLLHLH H    HOLD
;C L H H LLLL LHHL   H LLLLLHLL H    COUNT DOWN
;C L H H LLLL LHHL   H LLLLLLHH H    COUNT DOWN
;C L H H LLLL LHHL   H LLLLLLHL H    COUNT DOWN
;C L H H LLLL LHHL   H LLLLLLLH H    COUNT DOWN
;C L H H LLLL LHHL   H LLLLLLLL H    COUNT DOWN
;C L H H LLLL LHHL   L HHHHHHHH H    COUNT DOWN, READY TO LOAD COUNTER
;C H X X XXXX XXXX   Z ZZZZZZZZ Z    HIGH IMPEDANCE TEST
;--------------------------------------------------------------------
```

TB00640M

```
;DESCRIPTION

;THE REFRESH CLOCK GENERATOR CAN GENERATE REFRESH CLOCK (RFCK) FOR THE DYNAMIC
;RAM CONTROLLERS.  THE PERIOD OF RFCK DEPENDS ON F3-F0 WHILE THE DURATION OF
;RFCK BEING LOW DEPENDS ON M3-M0.

;         FFFF  RFCK PERIOD                MMMM  RFCK LOW DURATION
;         3210    (CYCLES)                 3210       (CYCLES)
;         -----------------                -----------------------
;         0000        12                   0000            0
;         0001        28                   0001            4
;         0010        44                   0010            8
;         0011        60                   0011           12
;         0100        76                   0100           16
;         0101        92                   0101           20*
;         0110       108                   0110           24
;         0111       124                   0111           28
;         1000       140                   1000           32
;         1001       156                   1001           36*
;         1010       172                   1010           40*
;         1011       188                   1011           44*
;         1100       204                   1100           48
;         1101       220                   1101           52*
;         1110       236                   1110           56
;         1111       252                   1111           60
;         -----------------                -----------------------

;*NOT ALLOWED DUE TO BAD WAVEFORMS
```

TB00650M

```
Title    OCTAL ADDRESSABLE REGISTER
Pattern  P7078
Revision A
Author   DANESH TAVANA
Company  MMI SUNNYVALE, CALIFORNIA
Date     02/02/82

CHIP OCTAL_ADDRESSABLE_REGISTER PAL16R8

CLK /CLR /PR A  B  C  E1 /E2 DIN GND
/OC  Q7   Q6 Q5 Q4 Q3 Q2  Q1 Q0  VCC

EQUATIONS

/Q0 :=  CLR                              ;CLEAR (LSB)
     + /PR* E1* E2*/DIN*/C*/B*/A         ;DATA IN ADDRESS 0
     + /PR* E1* E2*/Q0 *       A         ;LOAD PREVIOUS STATES
     + /PR* E1* E2*/Q0 *    B            ;LOAD PREVIOUS STATES
     + /PR* E1* E2*/Q0 * C               ;LOAD PREVIOUS STATES
     + /PR* E1*/E2*/Q0                   ;HOLD IF NOT LOADING (E1=H,/E2=H)
     + /PR*/E1*/E2*/Q0                   ;HOLD IF NOT LOADING (E1=L,/E2=H)
     + /PR*/E1* E2*/Q0                   ;HOLD IF NOT LOADING (E1=L,/E2=L)

/Q1 :=  CLR                              ;CLEAR
     + /PR* E1* E2*/DIN*/C*/B* A         ;DATA IN ADDRESS 1
     + /PR* E1* E2*/Q1 *      /A         ;LOAD PREVIOUS STATES
     + /PR* E1* E2*/Q1 *    B            ;LOAD PREVIOUS STATES
     + /PR* E1* E2*/Q1 * C               ;LOAD PREVIOUS STATES
     + /PR* E1*/E2*/Q1                   ;HOLD IF NOT LOADING (E1=H,/E2=H)
     + /PR*/E1*/E2*/Q1                   ;HOLD IF NOT LOADING (E1=L,/E2=H)
     + /PR*/E1* E2*/Q1                   ;HOLD IF NOT LOADING (E1=L,/E2=L)

/Q2 :=  CLR                              ;CLEAR
     + /PR* E1* E2*/DIN*/C* B*/A         ;DATA IN ADDRESS 2
     + /PR* E1* E2*/Q2 *       A         ;LOAD PREVIOUS STATES
     + /PR* E1* E2*/Q2 *   /B            ;LOAD PREVIOUS STATES
     + /PR* E1* E2*/Q2 * C               ;LOAD PREVIOUS STATES
     + /PR* E1*/E2*/Q2                   ;HOLD IF NOT LOADING (E1=H,/E2=H)
     + /PR*/E1*/E2*/Q2                   ;HOLD IF NOT LOADING (E1=L,/E2=H)
     + /PR*/E1* E2*/Q2                   ;HOLD IF NOT LOADING (E1=L,/E2=L)

/Q3 :=  CLR                              ;CLEAR
     + /PR* E1* E2*/DIN*/C* B* A         ;DATA IN ADDRESS 3
     + /PR* E1* E2*/Q3 *      /A         ;LOAD PREVIOUS STATES
     + /PR* E1* E2*/Q3 *   /B            ;LOAD PREVIOUS STATES
     + /PR* E1* E2*/Q3 * C               ;LOAD PREVIOUS STATES
     + /PR* E1*/E2*/Q3                   ;HOLD IF NOT LOADING (E1=H,/E2=H)
     + /PR*/E1*/E2*/Q3                   ;HOLD IF NOT LOADING (E1=L,/E2=H)
     + /PR*/E1* E2*/Q3                   ;HOLD IF NOT LOADING (E1=L,/E2=L)

/Q4 :=  CLR                              ;CLEAR
     + /PR* E1* E2*/DIN* C*/B*/A         ;DATA IN ADDRESS 4
     + /PR* E1* E2*/Q4 *       A         ;LOAD PREVIOUS STATES
     + /PR* E1* E2*/Q4 *    B            ;LOAD PREVIOUS STATES
     + /PR* E1* E2*/Q4 */C               ;LOAD PREVIOUS STATES
     + /PR* E1*/E2*/Q4                   ;HOLD IF NOT LOADING (E1=H,/E2=H)
     + /PR*/E1*/E2*/Q4                   ;HOLD IF NOT LOADING (E1=L,/E2=H)
     + /PR*/E1* E2*/Q4                   ;HOLD IF NOT LOADING (E1=L,/E2=L)
```

TB00660M

```
/Q5 :=  CLR                              ;CLEAR
     + /PR* E1* E2*/DIN* C*/B* A         ;DATA IN ADDRESS 5
     + /PR* E1* E2*/Q5 *      /A         ;LOAD PREVIOUS STATES
     + /PR* E1* E2*/Q5 *    B            ;LOAD PREVIOUS STATES
     + /PR* E1* E2*/Q5 */C               ;LOAD PREVIOUS STATES
     + /PR* E1*/E2*/Q5                   ;HOLD IF NOT LOADING (E1=H,/E2=H)
     + /PR*/E1*/E2*/Q5                   ;HOLD IF NOT LOADING (E1=L,/E2=H)
     + /PR*/E1* E2*/Q5                   ;HOLD IF NOT LOADING (E1=L,/E2=L)

/Q6 :=  CLR                              ;CLEAR
     + /PR* E1* E2*/DIN* C* B*/A         ;DATA IN ADDRESS 6
     + /PR* E1* E2*/Q6 *       A         ;LOAD PREVIOUS STATES
     + /PR* E1* E2*/Q6 *   /B            ;LOAD PREVIOUS STATES
     + /PR* E1* E2*/Q6 */C               ;LOAD PREVIOUS STATES
     + /PR* E1*/E2*/Q6                   ;HOLD IF NOT LOADING (E1=H,/E2=H)
     + /PR*/E1*/E2*/Q6                   ;HOLD IF NOT LOADING (E1=L,/E2=H)
     + /PR*/E1* E2*/Q6                   ;HOLD IF NOT LOADING (E1=L,/E2=L)

/Q7 :=  CLR                              ;CLEAR (MSB)
     + /PR* E1* E2*/DIN* C* B* A         ;DATA IN ADDRESS 7
     + /PR* E1* E2*/Q7 *      /A         ;LOAD PREVIOUS STATES
     + /PR* E1* E2*/Q7 *   /B            ;LOAD PREVIOUS STATES
     + /PR* E1* E2*/Q7 */C               ;LOAD PREVIOUS STATES
     + /PR* E1*/E2*/Q7                   ;HOLD IF NOT LOADING (E1=H,/E2=H)
     + /PR*/E1*/E2*/Q7                   ;HOLD IF NOT LOADING (E1=L,/E2=H)
     + /PR*/E1* E2*/Q7                   ;HOLD IF NOT LOADING (E1=L,/E2=L)

;FUNCTION TABLE FOR PALASM 1
;/OC CLK /CLR /PR /E2 E1 C B A DIN Q7 Q6 Q5 Q4 Q3 Q2 Q1 Q0
; CONTROL   FUNCTIONS     INPUTS        OUTPUTS
;/OC CLK /CLR /PR /E2 E1  C B A DIN Q Q Q Q Q Q Q Q
;                                   7 6 5 4 3 2 1 0    COMMENTS
;-------------------------------------------------------------------------------
; L   C   L    L   X   X  X X X  X  L L L L L L L L    CLEAR (OVERRD /PR)
; L   C   H    L   X   X  X X X  X  H H H H H H H H    PRESET (OVERRD ENABLES)
; L   C   H    H   L   H  L L L  L  H H H H H H H L    LOAD Q0 WITH DIN
; L   C   H    H   L   H  L L H  L  H H H H H H L L    LOAD Q1 WITH DIN
; L   C   H    H   L   H  L H L  L  H H H H H L L L    LOAD Q2 WITH DIN
; L   C   H    H   L   H  L H H  L  H H H H L L L L    LOAD Q3 WITH DIN
; L   C   H    H   L   H  H L L  L  H H H L L L L L    LOAD Q4 WITH DIN
; L   C   H    H   L   H  H L H  L  H H L L L L L L    LOAD Q5 WITH DIN
; L   C   H    H   L   H  H H L  L  H L L L L L L L    LOAD Q6 WITH DIN
; L   C   H    H   L   H  H H H  L  L L L L L L L L    LOAD Q7 WITH DIN
; L   C   H    H   L   H  H H H  H  H L L L L L L L    LOAD Q7 WITH DIN
; L   C   H    H   L   H  H H L  H  H H L L L L L L    LOAD Q6 WITH DIN
; L   C   H    H   L   H  H L H  H  H H H L L L L L    LOAD Q5 WITH DIN
; L   C   H    H   L   H  H L L  H  H H H H L L L L    LOAD Q4 WITH DIN
; L   C   H    H   L   H  L H H  H  H H H H H L L L    LOAD Q3 WITH DIN
; L   C   H    H   L   H  L H L  H  H H H H H H L L    LOAD Q2 WITH DIN
; L   C   H    H   L   H  L L H  H  H H H H H H H L    LOAD Q1 WITH DIN
; L   C   H    H   L   H  L L L  H  H H H H H H H H    LOAD Q0 WITH DIN
; L   C   H    H   L   H  H H H  L  L H H H H H H H    LOAD Q7 WITH DIN
; L   C   H    H   H   H  X X X  X  L H H H H H H H    HOLD Q5 WITH DIN
; L   C   H    H   L   H  H L H  L  L H L H H H H H    LOAD LINE 5 WITH 0
; L   C   H    H   H   L  X X X  X  L H L H H H H H    HOLD Q3 WITH DIN
; L   C   H    H   L   H  L H H  L  L H L H L H H H    LOAD LINE 3 WITH 0
; L   C   H    H   L   L  X X X  X  L H L H L H H H    HOLD Q1 WITH DIN
; L   C   H    H   L   H  L L H  L  L H L H L H L H    LOAD LINE 1 WITH 0
; H   X   X    X   X   X  X X X  X  Z Z Z Z Z Z Z Z    HI-Z
;-------------------------------------------------------------------------------
```

TB00670M

```
;DESCRIPTION

;THE 8-BIT ADDRESSABLE REGISTER LOADS THE DATA (DIN) INTO THE APPROPRIATE
;ADDRESS LINE REGISTER (Q) ON THE RISING EDGE OF THE CLOCK (CLK).
;THE INPUT ADDRESSING PINS (C,B,A) CHANNEL THE DATA (DIN) INTO ITS CORRESPONDING
;OUTPUT REGISTER (Q) WHEN THE ENABLE PINS (/E2,E1) ARE (LOW,HIGH) RESPECTIVELY;
;ANY OTHER COMBINATION OF INPUTS FOR THE ENABLE PINS HOLDS THE PREVIOUS STATE OF
;THE REGISTERS (Q).

;CLEAR OVERRIDES PRESET, PRESET OVERRIDES LOAD ENABLE.

;THESE FUNCTIONS ARE EXERCISED IN THE FUNCTION TABLE AND SUMMARIZED IN THE
;OPERATIONS TABLE:

;      /OC CLK  /CLR /PR  E1 /E2  C B A  DIN  Q7-Q0  OPERATION
;      -------------------------------------------------------
;       H   X    X    X   X   X     X     X     Z    HI-Z
;       L   C    L    X   X   X     X     X     L    CLEAR
;       L   C    H    L   X   X     X     X     H    PRESET
;       L   C    H    H   H   L   C B A   D     D    ENABLE
;       L   C    H    H   H   H     X     X     Q    HOLD
;       L   C    H    H   L   H     X     X     Q    HOLD
;       L   C    H    H   L   L     X     X     Q    HOLD
;      -------------------------------------------------------
```

TB00680M

```
Title    OCTAL ADDRESSABLE REGISTER WITH DEMULTIPLEXER/ENABLES
Pattern  P7079
Revision A
Author   DANESH TAVANA
Company  MMI SUNNYVALE, CALIFORNIA
Date     02/08/82

CHIP  OCTAL_ADDRESSABLE_REGISTER_WITH_DEMULTIPLEXER/ENABLES PAL16R8

CLK MODE /PR A  B  C  E1 /E2 DIN GND
/OC  Q7   Q6 Q5 Q4 Q3 Q2  Q1 Q0  VCC

EQUATIONS

/Q0 := /PR* E1* E2*/DIN*/C*/B*/A* MODE   ;LOAD /Q0 WITH /DIN (MODE=H)
     + /PR* E1* E2*     /C*/B*/A*/MODE   ;LOAD /Q0 WITH HIGH (MODE=L)
     + /PR* E1* E2*/Q0 *          A* MODE   ;/Q0 IS EITHER PREVIOUS STATE OR LOW
     + /PR* E1* E2*/Q0 *     B    * MODE   ;/Q0 IS EITHER PREVIOUS STATE OR LOW
     + /PR* E1* E2*/Q0 * C        * MODE   ;/Q0 IS EITHER PREVIOUS STATE OR LOW
     + /PR* E1*/E2*/Q0                   ;HOLD IF NOT LOADING (E1=H,/E2=H)
     + /PR*/E1*/E2*/Q0                   ;HOLD IF NOT LOADING (E1=L,/E2=H)
     + /PR*/E1* E2*/Q0                   ;HOLD IF NOT LOADING (E1=L,/E2=L)

/Q1 := /PR* E1* E2*/DIN*/C*/B* A* MODE   ;LOAD /Q1 WITH /DIN (MODE=H)
     + /PR* E1* E2*     /C*/B* A*/MODE   ;LOAD /Q1 WITH HIGH (MODE=L)
     + /PR* E1* E2*/Q1 *         /A* MODE   ;/Q1 IS EITHER PREVIOUS STATE OR LOW
     + /PR* E1* E2*/Q1 *     B    * MODE   ;/Q1 IS EITHER PREVIOUS STATE OR LOW
     + /PR* E1* E2*/Q1 * C        * MODE   ;/Q1 IS EITHER PREVIOUS STATE OR LOW
     + /PR* E1*/E2*/Q1                   ;HOLD IF NOT LOADING (E1=H,/E2=H)
     + /PR*/E1*/E2*/Q1                   ;HOLD IF NOT LOADING (E1=L,/E2=H)
     + /PR*/E1* E2*/Q1                   ;HOLD IF NOT LOADING (E1=L,/E2=L)

/Q2 := /PR* E1* E2*/DIN*/C* B*/A* MODE   ;LOAD /Q2 WITH /DIN (MODE=H)
     + /PR* E1* E2*     /C* B*/A*/MODE   ;LOAD /Q2 WITH HIGH (MODE=L)
     + /PR* E1* E2*/Q2 *          A* MODE   ;/Q2 IS EITHER PREVIOUS STATE OR LOW
     + /PR* E1* E2*/Q2 *    /B    * MODE   ;/Q2 IS EITHER PREVIOUS STATE OR LOW
     + /PR* E1* E2*/Q2 * C        * MODE   ;/Q2 IS EITHER PREVIOUS STATE OR LOW
     + /PR* E1*/E2*/Q2                   ;HOLD IF NOT LOADING (E1=H,/E2=H)
     + /PR*/E1*/E2*/Q2                   ;HOLD IF NOT LOADING (E1=L,/E2=H)
     + /PR*/E1* E2*/Q2                   ;HOLD IF NOT LOADING (E1=L,/E2=L)

/Q3 := /PR* E1* E2*/DIN*/C* B* A* MODE   ;LOAD /Q3 WITH /DIN (MODE=H)
     + /PR* E1* E2*     /C* B* A*/MODE   ;LOAD /Q3 WITH HIGH (MODE=L)
     + /PR* E1* E2*/Q3 *         /A* MODE   ;/Q3 IS EITHER PREVIOUS STATE OR LOW
     + /PR* E1* E2*/Q3 *    /B    * MODE   ;/Q3 IS EITHER PREVIOUS STATE OR LOW
     + /PR* E1* E2*/Q3 * C        * MODE   ;/Q3 IS EITHER PREVIOUS STATE OR LOW
     + /PR* E1*/E2*/Q3                   ;HOLD IF NOT LOADING (E1=H,/E2=H)
     + /PR*/E1*/E2*/Q3                   ;HOLD IF NOT LOADING (E1=L,/E2=H)
     + /PR*/E1* E2*/Q3                   ;HOLD IF NOT LOADING (E1=L,/E2=L)

/Q4 := /PR* E1* E2*/DIN* C*/B*/A* MODE   ;LOAD /Q4 WITH /DIN (MODE=H)
     + /PR* E1* E2*      C*/B*/A*/MODE   ;LOAD /Q4 WITH HIGH (MODE=L)
     + /PR* E1* E2*/Q4 *          A* MODE   ;/Q4 IS EITHER PREVIOUS STATE OR LOW
     + /PR* E1* E2*/Q4 *     B    * MODE   ;/Q4 IS EITHER PREVIOUS STATE OR LOW
     + /PR* E1* E2*/Q4 */C        * MODE   ;/Q4 IS EITHER PREVIOUS STATE OR LOW
     + /PR* E1*/E2*/Q4                   ;HOLD IF NOT LOADING (E1=H,/E2=H)
     + /PR*/E1*/E2*/Q4                   ;HOLD IF NOT LOADING (E1=L,/E2=H)
     + /PR*/E1* E2*/Q4                   ;HOLD IF NOT LOADING (E1=L,/E2=L)
```

TB00690M

```
/Q5 := /PR* E1* E2*/DIN* C*/B* A* MODE   ;LOAD /Q5 WITH /DIN (MODE=H)
     + /PR* E1* E2*      C*/B* A*/MODE   ;LOAD /Q5 WITH HIGH (MODE=L)
     + /PR* E1* E2*/Q5 *      /A* MODE   ;/Q5 IS EITHER PREVIOUS STATE OR LOW
     + /PR* E1* E2*/Q5 *    B    * MODE   ;/Q5 IS EITHER PREVIOUS STATE OR LOW
     + /PR* E1* E2*/Q5 */C       * MODE   ;/Q5 IS EITHER PREVIOUS STATE OR LOW
     + /PR* E1*/E2*/Q5                   ;HOLD IF NOT LOADING (E1=H,/E2=H)
     + /PR*/E1*/E2*/Q5                   ;HOLD IF NOT LOADING (E1=L,/E2=H)
     + /PR*/E1* E2*/Q5                   ;HOLD IF NOT LOADING (E1=L,/E2=L)

/Q6 := /PR* E1* E2*/DIN* C* B*/A* MODE   ;LOAD /Q6 WITH /DIN (MODE=H)
     + /PR* E1* E2*      C* B*/A*/MODE   ;LOAD /Q6 WITH HIGH (MODE=L)
     + /PR* E1* E2*/Q6 *       A* MODE   ;/Q6 IS EITHER PREVIOUS STATE OR LOW
     + /PR* E1* E2*/Q6 *   /B    * MODE   ;/Q6 IS EITHER PREVIOUS STATE OR LOW
     + /PR* E1* E2*/Q6 */C       * MODE   ;/Q6 IS EITHER PREVIOUS STATE OR LOW
     + /PR* E1*/E2*/Q6                   ;HOLD IF NOT LOADING (E1=H,/E2=H)
     + /PR*/E1*/E2*/Q6                   ;HOLD IF NOT LOADING (E1=L,/E2=H)
     + /PR*/E1* E2*/Q6                   ;HOLD IF NOT LOADING (E1=L,/E2=L)

/Q7 := /PR* E1* E2*/DIN* C* B* A* MODE   ;LOAD /Q7 WITH /DIN (MODE=H)
     + /PR* E1* E2*      C* B* A*/MODE   ;LOAD /Q7 WITH HIGH (MODE=L)
     + /PR* E1* E2*/Q7 *      /A* MODE   ;/Q7 IS EITHER PREVIOUS STATE OR LOW
     + /PR* E1* E2*/Q7 *   /B    * MODE   ;/Q7 IS EITHER PREVIOUS STATE OR LOW
     + /PR* E1* E2*/Q7 */C       * MODE   ;/Q7 IS EITHER PREVIOUS STATE OR LOW
     + /PR* E1*/E2*/Q7                   ;HOLD IF NOT LOADING (E1=H,/E2=H)
     + /PR*/E1*/E2*/Q7                   ;HOLD IF NOT LOADING (E1=L,/E2=H)
     + /PR*/E1* E2*/Q7                   ;HOLD IF NOT LOADING (E1=L,/E2=L)
```

TB00700M

```
;FUNCTION TABLE

;/OC CLK MODE /PR /E2 E1 C B A DIN Q7 Q6 Q5 Q4 Q3 Q2 Q1 Q0

;CONTROL    FUNCTIONS        INPUTS      OUTPUTS
;/OC CLK MODE /PR /E2 E1   C B A DIN  Q Q Q Q Q Q Q Q
;                                     7 6 5 4 3 2 1 0      COMMENTS
-----------------------------------------------------------------------------------
; L   C    X    L    X  X   X X X  X   H H H H H H H H    PRESET (OVERRD ENABLES)
; L   C    H    H    L  H   L L L  L   H H H H H H H L    LOAD Q0 WITH DIN
; L   C    H    H    L  H   L L H  L   H H H H H H L L    LOAD Q1 WITH DIN
; L   C    H    H    L  H   L H L  L   H H H H H L L L    LOAD Q2 WITH DIN
; L   C    H    H    L  H   L H H  L   H H H H L L L L    LOAD Q3 WITH DIN
; L   C    H    H    L  H   H L L  L   H H H L L L L L    LOAD Q4 WITH DIN
; L   C    H    H    L  H   H L H  L   H H L L L L L L    LOAD Q5 WITH DIN
; L   C    H    H    L  H   H H L  L   H L L L L L L L    LOAD Q6 WITH DIN
; L   C    H    H    L  H   H H H  L   L L L L L L L L    LOAD Q7 WITH DIN
; L   C    H    H    L  H   H H H  H   H L L L L L L L    LOAD Q7 WITH DIN
; L   C    H    H    L  H   H H L  H   H H L L L L L L    LOAD Q6 WITH DIN
; L   C    H    H    L  H   H L H  H   H H H L L L L L    LOAD Q5 WITH DIN
; L   C    H    H    L  H   H L L  H   H H H H L L L L    LOAD Q4 WITH DIN
; L   C    H    H    L  H   L H H  H   H H H H H L L L    LOAD Q3 WITH DIN
; L   C    H    H    L  H   L H L  H   H H H H H H L L    LOAD Q2 WITH DIN
; L   C    H    H    L  H   L L H  H   H H H H H H H L    LOAD Q1 WITH DIN
; L   C    H    H    L  H   L L L  H   H H H H H H H H    LOAD Q0 WITH DIN
; L   C    L    H    L  H   L L L  X   H H H H H H H L    DECODE ADDRESS LINE 0
; L   C    L    H    L  H   L L H  X   H H H H H H L H    DECODE ADDRESS LINE 1
; L   C    L    H    L  H   L H L  X   H H H H H L H H    DECODE ADDRESS LINE 2
; L   C    L    H    L  H   L H H  X   H H H H L H H H    DECODE ADDRESS LINE 3
; L   C    L    H    L  H   H L L  X   H H H L H H H H    DECODE ADDRESS LINE 4
; L   C    L    H    L  H   H L H  X   H H L H H H H H    DECODE ADDRESS LINE 5
; L   C    L    H    L  H   H H L  X   H L H H H H H H    DECODE ADDRESS LINE 6
; L   C    L    H    L  H   H H H  X   L H H H H H H H    DECODE ADDRESS LINE 7
; L   C    X    H    L  L   X X X  X   L H H H H H H H    HOLD PREVIOUS STATE
; L   C    H    H    L  H   H L H  L   L H L H H H H H    LOAD Q5 WITH DIN
; L   C    X    H    H  L   X X X  X   L H L H H H H H    HOLD PREVIOUS STATE
; L   C    H    H    L  H   L H H  L   L H L H L H H H    LOAD Q3 WITH DIN
; L   C    X    H    H  H   X X X  X   L H L H L H H H    HOLD PREVIOUS STATE
; L   C    L    H    L  H   L L H  X   H H H H H H L H    DECODE ADDRESS LINE 1
; H   X    X    X    X  X   X X X  X   Z Z Z Z Z Z Z Z    HI-Z
;----------------------------------------------------------------------------------
```

TB00710M

```
;DESCRIPTION

;THE 8-BIT ADDRESSABLE REGISTER AND DEMULTIPLEXER PERFORMS TWO FUNCTIONS ON ONE
;MSI PACKAGE. IF MODE=0 THE PART PERFORMS A 3 TO 8 DE-MULTIPLEXER FUNCTION WITH
;PRESET, LOAD, AND HOLD; IF MODE=1 IT IS AN 8-BIT ADRESSABLE REGISTER WITH
;PRESET, LOAD, AND HOLD.

;WHEN THE CONTROL PINS (/E2,E1) ARE (LOW,HIGH) THE UNIT IS ENABLED AND THE INPUT
;ADDRESS PINS (C,B,A) EITHER CHANNEL THE INPUT DATA (DIN) TO ITS APPROPRIATE
;OUTPUT REGISTER (Q) WITH MODE=1, OR SELECT THE ADDRESSED OUTPUT REGISTER (Q)
;WITH MODE=0.  ANY OTHER COMBINATION FOR THE INPUT CONTROL PINS HOLDS THE
;PREVIOUS STATE OF THE OUTPUTS.

;PRESET OVERRIDES ENABLE.

;THESE FUNCTIONS ARE EXERCISED IN THE FUNCTION TABLE AND SUMMARIZED IN THE
;OPERATIONS TABLE:

;       /OC CLK  MODE /PR /E2  E1  C B A  DIN  Q7-Q0  OPERATION
;       -------------------------------------------------------------------
;        H   X    X    X   X    X   X X X   X     Z    HI-Z
;        L   C    X    L   X    X   X X X   X     H    PRESET
;        L   C    L    H   L    H   C B A   X    MUX   ADDRESSED OUTPUT=LOW
;        L   C    H    H   L    H   C B A   D    REG   ADDRESSED OUTPUT=D
;        L   C    X    H   H    L   X X X   X     Q    HOLD
;        L   C    X    H   L    L   X X X   X     Q    HOLD
;        L   C    X    H   H    H   X X X   X     Q    HOLD
;       -------------------------------------------------------------------
```

TB00720M

```
Title    OCTAL ADDRESSABLE REGISTER WITH DEMULTIPLEXER/CLEAR
Pattern  P7080 (PMSI002)
Revision A
Author   DANESH TAVANA
Company  MMI SUNNYVALE, CALIFORNIA
Date     04/27/82

CHIP OCTAL_ADDRESSABLE_REGISTER_WITH_DEMULTIPLEXER/CLEAR PAL16R8

EQUATIONS

CLK /CLR /PR A  B  C  MODE /E DIN GND
/OC  Q7   Q6 Q5 Q4 Q3  Q2  Q1 Q0  VCC

/Q0 :=  CLR                              ;CLEAR (LSB)
     + /PR* E*/MODE*/C*/B*/A             ;DEMULTIPLEX OUTPUT  /Q0=H
     + /PR* E* MODE*/C*/B*/A*/DIN        ;REGISTER OUTPUT /Q0=/DIN
     + /PR* E* MODE*       A*/Q0         ;LOAD PREVIOUS STATE (/Q0) OR A LOW
     + /PR* E* MODE*    B   */Q0         ;LOAD PREVIOUS STATE (/Q0) OR A LOW
     + /PR* E* MODE* C      */Q0         ;LOAD PREVIOUS STATE (/Q0) OR A LOW
     + /PR*/E               */Q0         ;HOLD IF NOT LOADING (/E=H)

/Q1 :=  CLR                              ;CLEAR
     + /PR* E*/MODE*/C*/B* A             ;DEMULTIPLEX OUTPUT  /Q1=H
     + /PR* E* MODE*/C*/B* A*/DIN        ;REGISTER OUTPUT /Q1=/DIN
     + /PR* E* MODE*      /A*/Q1         ;LOAD PREVIOUS STATE (/Q1) OR A LOW
     + /PR* E* MODE*    B   */Q1         ;LOAD PREVIOUS STATE (/Q1) OR A LOW
     + /PR* E* MODE* C      */Q1         ;LOAD PREVIOUS STATE (/Q1) OR A LOW
     + /PR*/E               */Q1         ;HOLD IF NOT LOADING (/E=H)

/Q2 :=  CLR                              ;CLEAR
     + /PR* E*/MODE*/C* B*/A             ;DEMULTIPLEX OUTPUT  /Q2=H
     + /PR* E* MODE*/C* B*/A*/DIN        ;REGISTER OUTPUT /Q2=/DIN
     + /PR* E* MODE*       A*/Q2         ;LOAD PREVIOUS STATE (/Q2) OR A LOW
     + /PR* E* MODE*   /B   */Q2         ;LOAD PREVIOUS STATE (/Q2) OR A LOW
     + /PR* E* MODE* C      */Q2         ;LOAD PREVIOUS STATE (/Q2) OR A LOW
     + /PR*/E               */Q2         ;HOLD IF NOT LOADING (/E=H)

/Q3 :=  CLR                              ;CLEAR
     + /PR* E*/MODE*/C* B* A             ;DEMULTIPLEX OUTPUT  /Q3=H
     + /PR* E* MODE*/C* B* A*/DIN        ;REGISTER OUTPUT /Q3=/DIN
     + /PR* E* MODE*      /A*/Q3         ;LOAD PREVIOUS STATE (/Q3) OR A LOW
     + /PR* E* MODE*   /B   */Q3         ;LOAD PREVIOUS STATE (/Q3) OR A LOW
     + /PR* E* MODE* C      */Q3         ;LOAD PREVIOUS STATE (/Q3) OR A LOW
     + /PR*/E               */Q3         ;HOLD IF NOT LOADING (/E=H)

/Q4 :=  CLR                              ;CLEAR
     + /PR* E*/MODE* C*/B*/A             ;DEMULTIPLEX OUTPUT  /Q4=H
     + /PR* E* MODE* C*/B*/A*/DIN        ;REGISTER OUTPUT /Q4=/DIN
     + /PR* E* MODE*       A*/Q4         ;LOAD PREVIOUS STATE (/Q4) OR A LOW
     + /PR* E* MODE*    B   */Q4         ;LOAD PREVIOUS STATE (/Q4) OR A LOW
     + /PR* E* MODE*/C      */Q4         ;LOAD PREVIOUS STATE (/Q4) OR A LOW
     + /PR*/E               */Q4         ;HOLD IF NOT LOADING (/E=H)
```

TB00730M

```
/Q5 :=  CLR                                  ;CLEAR
     + /PR* E*/MODE* C*/B* A                 ;DEMULTIPLEX OUTPUT  /Q5=H
     + /PR* E* MODE* C*/B* A*/DIN            ;REGISTER OUTPUT /Q5=/DIN
     + /PR* E* MODE*      /A*/Q5             ;LOAD PREVIOUS STATE (/Q5) OR A LOW
     + /PR* E* MODE*    B   */Q5             ;LOAD PREVIOUS STATE (/Q5) OR A LOW
     + /PR* E* MODE*/C      */Q5             ;LOAD PREVIOUS STATE (/Q5) OR A LOW
     + /PR*/E               */Q5             ;HOLD IF NOT LOADING (/E=H)

/Q6 :=  CLR                                  ;CLEAR
     + /PR* E*/MODE* C* B*/A                 ;DEMULTIPLEX OUTPUT  /Q6=H
     + /PR* E* MODE* C* B*/A*/DIN            ;REGISTER OUTPUT /Q5=/DIN
     + /PR* E* MODE*       A*/Q6             ;LOAD PREVIOUS STATE (/Q6) OR A LOW
     + /PR* E* MODE*   /B   */Q6             ;LOAD PREVIOUS STATE (/Q6) OR A LOW
     + /PR* E* MODE*/C      */Q6             ;LOAD PREVIOUS STATE (/Q6) OR A LOW
     + /PR*/E               */Q6             ;HOLD IF NOT LOADING (/E=H)

/Q7 :=  CLR                                  ;CLEAR (MSB)
     + /PR* E*/MODE* C* B* A                 ;DEMULTIPLEX OUTPUT  /Q7=H
     + /PR* E* MODE* C* B* A*/DIN            ;REGISTER OUTPUT /Q7=/DIN
     + /PR* E* MODE*      /A*/Q7             ;LOAD PREVIOUS STATE (/Q7) OR A LOW
     + /PR* E* MODE*   /B   */Q7             ;LOAD PREVIOUS STATE (/Q7) OR A LOW
     + /PR* E* MODE*/C      */Q7             ;LOAD PREVIOUS STATE (/Q7) OR A LOW
     + /PR*/E               */Q7             ;HOLD IF NOT LOADING (/E=H)

;FUNCTION TABLE FOR PALASM 1
;/OC CLK /CLR /PR /E MODE C B A DIN Q7 Q6 Q5 Q4 Q3 Q2 Q1 Q0
;                                      ----OUTPUTS----
;CONTROL ---FUNCTIONS----  -INPUTS--  Q Q Q Q Q Q Q Q
;/OC CLK /CLR /PR /E MODE  C B A DIN  7 6 5 4 3 2 1 0   COMMENTS
;------------------------------------------------------------------------------
; L   C    L    L   X   X   X X X  X   L L L L L L L L   CLEAR (OVERRD /PR)
; L   C    H    L   X   X   X X X  X   H H H H H H H H   PRESET (OVERRD ENABLES)
; L   C    H    H   L   H   L L L  L   H H H H H H H L   LOAD Q0 WITH DIN
; L   C    H    H   L   H   L L H  L   H H H H H H L L   LOAD Q1 WITH DIN
; L   C    H    H   L   H   L H L  L   H H H H H L L L   LOAD Q2 WITH DIN
; L   C    H    H   L   H   L H H  L   H H H H L L L L   LOAD Q3 WITH DIN
; L   C    H    H   L   H   H L L  L   H H H L L L L L   LOAD Q4 WITH DIN
; L   C    H    H   L   H   H L H  L   H H L L L L L L   LOAD Q5 WITH DIN
; L   C    H    H   L   H   H H L  L   H L L L L L L L   LOAD Q6 WITH DIN
; L   C    H    H   L   H   H H H  L   L L L L L L L L   LOAD Q7 WITH DIN
; L   C    H    H   L   H   H H H  H   H L L L L L L L   LOAD Q7 WITH DIN
; L   C    H    H   L   H   H H L  H   H H L L L L L L   LOAD Q6 WITH DIN
; L   C    H    H   L   H   H L H  H   H H H L L L L L   LOAD Q5 WITH DIN
; L   C    H    H   L   H   H L L  H   H H H H L L L L   LOAD Q4 WITH DIN
; L   C    H    H   L   H   L H H  H   H H H H H L L L   LOAD Q3 WITH DIN
; L   C    H    H   L   H   L H L  H   H H H H H H L L   LOAD Q2 WITH DIN
; L   C    H    H   L   H   L L H  H   H H H H H H H L   LOAD Q1 WITH DIN
; L   C    H    H   L   H   L L L  H   H H H H H H H H   LOAD Q0 WITH DIN
; L   C    H    H   L   L   L L L  X   H H H H H H H L   DECODE ADDRESS LINE 0
; L   C    H    H   L   L   L L H  X   H H H H H H L H   DECODE ADDRESS LINE 1
; L   C    H    H   L   L   L H L  X   H H H H H L H H   DECODE ADDRESS LINE 2
; L   C    H    H   L   L   L H H  X   H H H H L H H H   DECODE ADDRESS LINE 3
; L   C    H    H   L   L   H L L  X   H H H L H H H H   DECODE ADDRESS LINE 4
; L   C    H    H   L   L   H L H  X   H H L H H H H H   DECODE ADDRESS LINE 5
; L   C    H    H   L   L   H H L  X   H L H H H H H H   DECODE ADDRESS LINE 6
; L   C    H    H   L   L   H H H  X   L H H H H H H H   DECODE ADDRESS LINE 7
; L   C    H    H   H   X   X X X  X   L H H H H H H H   HOLD PREVIOUS STATE
; H   X    X    X   X   X   X X X  X   Z Z Z Z Z Z Z Z   TEST HI-Z
;------------------------------------------------------------------------------
```

TB00740M

```
;DESCRIPTION

;THE OCTAL ADDRESSABLE REGISTER AND DEMULTIPLEXER PERFORMS ONE OF TWO MSI
;FUNCTIONS DEPENDING ON THE STATE OF THE MODE SELECT PIN. IF MODE=HIGH THEN THE
;PART PERFORMS THE FUNCTION OF AN ADDRESSABLE REGISTER WITH 8 OUTPUTS (Q7-Q0)
;AND ONE DATA INPUT (DIN).  THE REGISTERED OUTPUT IS SELECTED BY THREE INPUT
;ADDRESS PINS (A,B,C).  WITH MODE=LOW CONVERTS THIS CHIP INTO AN ACTIVE LOW
;3-TO-8 DEMULTIPLEXER, WHERE THE ADDRESSED OUTPUT IS LOW AND ALL OTHER OUTPUTS
;REMAIN HIGH.  CLEAR (/CLR) AND PRESET (/PR) ARE ACTIVE LOW OUTPUTS WHICH SET
;ALL OUTPUTS TO LOW OR HIGH RESPECTIVELY. WHEN ENABLE (/E) IS HIGH, THE CHIP IS
;DISABLED AND THE OUTPUTS RETAIN THEIR PREVIOUS STATES.

;CLEAR OVERRIDES PRESET AND ENABLE, PRESET OVERRIDES ENABLE.

;THESE FUNCTIONS ARE EXERCISED IN THE FUNCTION TABLE AND SUMMARIZED IN THE
;OPERATIONS TABLE:

;/OC  CLK  /CLR  /PR  /E  MODE  C B A  DIN  Q7-Q0   OPERATION
;----------------------------------------------------------------------------
; H    X     X    X    X    X    X X X   X     Z     HI-Z
; L    C     L    X    X    X    X X X   X     L     CLEAR

; L    C     H    L    X    X    X X X   X     H     PRESET
; L    C     H    H    L    L    C B A   X    MUX    ADDRESSED OUTPUT=LOW
; L    C     H    H    L    H    C B A  DIN   REG    ADDRESSED OUTPUT=DIN
; L    C     H    H    H    X    X X X   X     Q     HOLD PREVIOUS STATES
;----------------------------------------------------------------------------


;                        OUTPUT SELECT TABLE

;      C B A   DIN   Q7  Q6  Q5  Q4  Q3  Q2  Q1  Q0
;      ---------------------------------------------
;      L L L   DIN   Q7  Q6  Q5  Q4  Q3  Q2  Q1  DIN
;      L L H   DIN   Q7  Q6  Q5  Q4  Q3  Q2  DIN Q0
;      L H L   DIN   Q7  Q6  Q5  Q4  Q3  DIN Q1  Q0
;      L H H   DIN   Q7  Q6  Q5  Q4  DIN Q2  Q1  Q0
;      H L L   DIN   Q7  Q6  Q5  DIN Q3  Q2  Q1  Q0
;      H L H   DIN   Q7  Q6  DIN Q4  Q3  Q2  Q1  Q0
;      H H L   DIN   Q7  DIN Q5  Q4  Q3  Q2  Q1  Q0
;      H H H   DIN   DIN Q6  Q5  Q4  Q3  Q2  Q1  Q0
;     ----------------------------------------------
```

TB00750M

Rounding – Control Logic

```
Title    ROUNDING-CONTROL LOGIC
Pattern  P7081
Revision A
Author   VINCENT COLI
Company  MMI SUNNYVALE, CALIFORNIA
Date     03/18/82

CHIP ROUNDING-CONTROL_LOGIC PAL16C1


D00 D01 D02 D03 D04 D05 D06 D07 D08 GND
D09 D10 D11 D12 TRU COM NC  NC  NC  VCC

EQUATIONS

COM =  D12* D11
    +       D11* D10
    +       D11     *D09
    +       D11         * D08
    +       D11              * D07
    +       D11                   * D06
    +       D11                        * D05
    +       D11                             * D04
    +       D11                                  * D03
    +       D11                                       * D02
    +       D11                                            * D01
    +       D11                                                 * D00


;FUNCTION TABLE

;D12 D11 D10 D09 D08 D07 D06 D05 D04 D03 D02 D01 D00 COM TRU
;
; COM RESULTS HAVE BEEN REMOVED FROM SIMULATION RESULTS
;
;D  D  D  D  D  D  D  D  D  D  D  D  D   ROUND-CONTROL
;12 11 10
```

TB00760M

4-Bit Counter With Terminal Count Lock

```
Title    4-BIT COUNTER WITH TERMINAL COUNT LOCK
Pattern  P7082
Revision A
Author   COLI/VOLDAN
Company  MMI SUNNYVALE, CALIFORNIA
Date     04/05/82

CHIP 4-BIT_COUNTER_WITH_TERMINAL_COUNT_LOCK PAL16R6

CLK NC  NC D0 D1 D2 D3 CLEAR /LOAD GND
/OC NC /EP Q3 Q2 Q1 Q0  NC     NC  VCC

EQUATIONS

/Q0 :=  CLEAR                         ;CLEAR Q0 (LSB)
     +  LOAD*/D0                      ;LOAD D0
     + /LOAD*/Q0*/EP                  ;HOLD IF NO EP
     + /LOAD* EP* Q0                  ;COUNT IF EP AND Q0=H

/Q1 :=  CLEAR                         ;CLEAR Q1
     +  LOAD*/D1                      ;LOAD D1
     + /LOAD*/Q1*/EP                  ;HOLD IF NO EP
     + /LOAD*/Q1*/Q0                  ;HOLD IF Q1,Q0=L
     + /LOAD* EP* Q1* Q0              ;COUNT IF EP AND Q1,Q0=H

/Q2 :=  CLEAR                         ;CLEAR Q2
     +  LOAD*/D2                      ;LOAD D2
     + /LOAD*/Q2*/EP                  ;HOLD IF NO EP
     + /LOAD*/Q2*/Q0                  ;HOLD IF Q2,Q0=L
     + /LOAD*/Q2*/Q1                  ;HOLD IF Q2,Q1=L
     + /LOAD* EP* Q2* Q1* Q0          ;COUNT IF EP AND Q2,Q1,Q0=H

/Q3 :=  CLEAR                         ;CLEAR Q3 (MSB)
     +  LOAD*/D3                      ;LOAD D3
     + /LOAD*/Q3*/EP                  ;HOLD IF NO EP
     + /LOAD*/Q3*/Q0                  ;HOLD IF Q3,Q0=L
     + /LOAD*/Q3*/Q1                  ;HOLD IF Q3,Q1=L
     + /LOAD*/Q3*/Q2                  ;HOLD IF Q3,Q2=L
     + /LOAD* EP* Q3* Q2* Q1* Q0      ;COUNT IF EP AND Q3,Q2,Q1,Q0=H

 EP :=   CLEAR                        ;INITIALIZE REGISTER
     +  /Q3 + /Q2 + /Q1               ;HOLD IF Q3,Q2,Q1=H
```

TB00770M

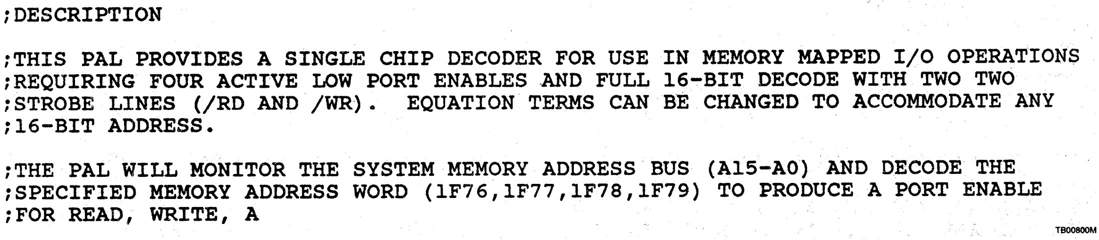
```
;FUNCTION TABLE FOR PALASM 1

;CLK /OC CLEAR /LOAD /EP D3 D2 D1 D0 Q3 Q2 Q1 Q0

;CONTROL   ---OPERATION---    DDDD   QQQQ     COMMENTS
;CLK /OC   CLEAR /LOAD /EP    3210   3210    (HEX VALUES)
;------------------------------------------------------------------
; C   L      H     H    L     XXXX   LLLL    CLEAR (0)
; C   L      L     H    L     XXXX   LLLH    COUNT (1)
; C   L      L     H    L     XXXX   LLHL    COUNT (2)
; C   L      L     H    L     XXXX   LLHH    COUNT (3)
; C   L      L     H    L     XXXX   LHLL    COUNT (4)
; C   L      L     H    L     XXXX   LHLH    COUNT (5)
; C   L      L     H    L     XXXX   LHHL    COUNT (6)
; C   L      L     H    L     XXXX   LHHH    COUNT (7)
; C   L      L     H    L     XXXX   HLLL    COUNT (8)
; C   L      L     H    L     XXXX   HLLH    COUNT (9)
; C   L      L     H    L     XXXX   HLHL    COUNT (A)
; C   L      L     H    L     XXXX   HLHH    COUNT (B)
; C   L      L     H    L     XXXX   HHLL    COUNT (C)
; C   L      L     H    L     XXXX   HHLH    COUNT (D)
; C   L      L     H    L     XXXX   HHHL    COUNT (E)
; C   L      L     H    H     XXXX   HHHH    COUNT (F)
; C   L      L     H    H     XXXX   HHHH    HOLD AT TERMINAL COUNT
; C   L      L     L    X     LLLL   LLLL    LOAD (0)
; C   L      L     L    X     LLLH   LLLH    LOAD (1)
; C   L      L     L    X     LLHL   LLHL    LOAD (3)
; C   L      L     L    X     LHLL   LHLL    LOAD (4)
; C   L      L     L    X     HLLL   HLLL    LOAD (8)
; C   L      L     L    X     HHLL   HHLL    LOAD (C)
; C   L      L     L    X     HHHL   HHHL    LOAD (E)
; C   L      L     L    X     HHHH   HHHH    LOAD (F)
; X   H      X     X    Z     XXXX   ZZZZ    TEST HI-Z
;------------------------------------------------------------------

;DESCRIPTION

;THE 4-BIT COUNTER LOADS DATA, INCREMENTS, OR HOLDS ON THE RISING EDGE
;OF THE CLOCK (CLK).

;THE COUNTER WILL HOLD WHEN THE TERMINAL COUNT IS REACHED UNTIL NEW DATA
;IS LOADED OR THE REGISTERS ARE CLEARED.

;THESE OPERATIONS ARE EXERCISED IN THE FUNCTION TABLE AND SUMMARIZED IN THE
;IN THE OPERATIONS TABLE:

;    /OC  CLK  CLEAR  /LOAD  /EP  D3-D0  Q3-Q0     OPERATION
;   ----------------------------------------------------------
;     H    X     X      X     X     X       Z      HI-Z
;     L    C     H      X     X     L       L      CLEAR
;     L    C     L      L     X     A       A      LOAD
;     L    C     L      H     H     X       Q      HOLD
;     L    C     L      H     L     X    Q PLUS 1  INCREMENT
;   ----------------------------------------------------------
```

TB00780M

7

```
Title   MEMORY MAPPED I/O
Pattern P7083
Revison A
Author  VINCENT COLI
Company MMI SUNNYVALE, CALIFORNIA
Date    04/19/82

CHIP MEMORY_MAPPED_I/O PAL18L4

A0 A1 A2 A3 A4 A5 A6 A7 A8 /RD /WR GND
A9 A10 A11 A12 /READ /WRITE /AUX2 /AUX1 A13 A14 A15 VCC

EQUATIONS

READ  = /A15*/A14*/A13* A12* A11* A10* A9* A8  ;SELECT PORT1
      * /A7 * A6 * A5 * A4 */A3 * A2 * A1*/A0  ;    (1F76)
      *  RD                                    ;READ STROBE

WRITE = /A15*/A14*/A13* A12* A11* A10* A9* A8  ;SELECT PORT1
      * /A7 * A6 * A5 * A4 */A3 * A2 * A1* A0  ;    (1F77)
      *  WR                                    ;WRITE STROBE

AUX1  = /A15*/A14*/A13* A12* A11* A10* A9* A8  ;SELECT PORT0
      * /A7 * A6 * A5 * A4 * A3 */A2 */A1*/A0  ;    (1F78)

AUX2  = /A15*/A14*/A13* A12* A11* A10* A9* A8  ;SELECT PORT1
      * /A7 * A6 * A5 * A4 * A3 */A2 */A1* A0  ;    (1F79)

;FUNCTION TABLE FOR PALASM 1
;A15 A14 A13 A12 A11 A10 A9 A8 A7 A6 A5 A4 A3 A2 A1 A0 /RD /WR
;/READ /WRITE /AUX1 /AUX2

;-ADDRESS INPUTS-
;111111           STROBES  --PORT ENABLE OUTPUTS---
;5432109876543210 /RD /WR  /READ /WRITE /AUX1 /AUX2   COMMENT
;----------------------------------------------------------------
;LLLLHHHHLHHHHLLL  L   L     H      H     H     H     TEST 0F78
;LLLHHHHLLHHHHLLL  L   L     H      H     H     H     TEST 1E78
;LLLHHHHHHHHHHLLL  L   L     H      H     H     H     TEST 1FF8
;LLLHHHHHLHHHLLLL  L   L     H      H     H     H     TEST 1F70
;LLLHHHHHLHHHLHHL  H   L     H      H     H     H     TEST 1F76 (NO RD STROBE)
;LLLHHHHHLHHHLHHL  L   L     L      H     H     H     TEST 1F76 (READ ENABLE)
;LLLHHHHHLHHHLHHH  L   H     H      H     H     H     TEST 1F77 (NO WR STROBE)
;LLLHHHHHLHHHLHHH  L   L     H      L     H     H     TEST 1F77 (WRITE ENABLE)
;LLLHHHHHLHHHHLLL  L   L     H      H     L     H     TEST 1F78 (AUX1 ENABLE)
;LLLHHHHHLHHHHLLH  L   L     H      H     H     L     TEST 1F79 (AUX2 ENABLE)
;LLLHHHHHLHHHHLHL  L   L     H      H     H     H     TEST 1F7A
;LLLHHHHHHHHHHLLH  L   L     H      H     H     H     TEST 1FF9
;LLLHHHHLLHHHHLLH  L   L     H      H     H     H     TEST 1E79
;LLHHHHHHLHHHHLLH  L   L     H      H     H     H     TEST 3F79
;LLLLLLLLLLLLLLLL  L   L     H      H     H     H     TEST ALL L'S
;HHHHHHHHHHHHHHHH  L   L     H      H     H     H     TEST ALL H'S
;LHLHLHLHLHLHLHLH  L   L     H      H     H     H     TEST ODD  CHECKERBOARD
;HLHLHLHLHLHLHLHL  L   L     H      H     H     H     TEST EVEN CHECKERBOARD
;-------------------------------------------------------------------
```

TB00790M



```
;DESCRIPTION

;THIS PAL PROVIDES A SINGLE CHIP DECODER FOR USE IN MEMORY MAPPED I/O OPERATIONS
;REQUIRING FOUR ACTIVE LOW PORT ENABLES AND FULL 16-BIT DECODE WITH TWO TWO
;STROBE LINES (/RD AND /WR).  EQUATION TERMS CAN BE CHANGED TO ACCOMMODATE ANY
;16-BIT ADDRESS.

;THE PAL WILL MONITOR THE SYSTEM MEMORY ADDRESS BUS (A15-A0) AND DECODE THE
;SPECIFIED MEMORY ADDRESS WORD (1F76,1F77,1F78,1F79) TO PRODUCE A PORT ENABLE
;FOR READ, WRITE, A
```

TB00800M

74S508 Memory Map Interface #1

```
Title    SN54/74S508 MEMORY MAP INTERFACE WITH THE INTEL 8085 (DESIGN # 1)
Pattern  P7085
Revision A
Author   VINCENT COLI
Company  MMI SUNNYVALE, CALIFORNIA
Date     05/15/82

CHIP SN54/74S508_MEMORY_MAP_INTERFACE_WITH_THE_INTEL_8085_(DESIGN_#1) PAL16R4

CLK AD0 AD1 AD2 AD3 AD4 AD5 AD6 AD7 GND
/OE /E1 ALE I2  I1  I0  /GO E2  E3  VCC

EQUATIONS

GO := /AD3*/AD4*/AD5* AD6*/AD7     ;MONITOR ADDRESS/DATA BUS (AD3-AD7)
    *  E1 * E2*  E3                ;MONITOR ENABLES (FROM A8-A15)
    *  ALE                         ;MONITOR ADDRESS/DATA CONTROL

/I0 := /AD0     ;REGISTER INSTRUCTION INPUT I0

/I1 := /AD1     ;REGISTER INSTRUCTION INPUT I1

/I2 := /AD2     ;REGISTER INSTRUCTION INPUT I2

;FUNCTION TABLE FOR PALASM 1
;/OE CLK AD0 AD1 AD2 AD3 AD4 AD5 AD6 AD7 /E1 E2 E3 ALE I0 I1 I2 /GO

;          A/D BUS    ADD BUS       INSTR          COMMENTS
;/OE CLK  01234567  /E1 E2 E3   ALE  012   /GO   (OCTAL ADDRESS)
;-------------------------------------------------------------------------
; L   C   XXXLLLLH   L  H  H     H   XXX    H    (200 TO 207) INACTIVE
; L   C   XXXLLLLH   H  H  H     H   XXX    H    (100 TO 107) INACTIVE (/E1=H)
; L   C   XXXLLLHL   L  L  L     H   XXX    H    (100 TO 107) INACTIVE (E2,E3=L)
; L   C   XXXLLLHL   L  H  H     L   XXX    H    (100 TO 107) INACTIVE (ALE=L)
; L   C   XXXLLLHL   L  H  H     H   XXX    L    (100 TO 107) (ACTIVE RANGE)
; L   C   LHHLLLHL   L  H  H     H   LHH    L    (106) LOAD X
; L   C   LHHLLLHL   L  H  H     L   LHH    H    (106) NOP (ALE=L)
; L   C   LLLLLLHL   L  H  H     H   LLL    L    (100) LOAD Y
; H   X   XXXXXXXX   X  X  X     X   ZZZ    Z    TEST HI-Z
;-------------------------------------------------------------------------
```

TB00830M

74S508 Memory Map Interface #1

```
;DESCRIPTION

;THIS PAL PROVIDES THE DECODE LOGIC FOR INTERFACING THE MMI SN54/74S508, 8-BIT
;SEQUENTIAL MULTIPLIER/DIVIDER, WITH THE INTEL 8085 MICROPROCESSOR.  THE PAL16R4
;MONITORS THE LOWER 8 BIT ADDRESS/DATA BUS (AD0-AD7), THE ADDRESS LATCH ENABLE
;(ALE), AND THREE OF THE UPPER 8 BIT ADDRESS BUS (A8-A15) WHICH IS LABELED
;/E1,E2,M THE 8085 IN ORDER TO
;DECODE AN ACTIVE LOW CHIP-ACTIVATION SIGNAL (/GO) FOR THE 74S508.  THE
;INSTRUCTION LINES AND CHIP-ACTIVATION SIGNAL ARE REGISTERED IN ORDER TO INSURE
;THAT INSTRUCTION INPUTS WILL NOT CHANGE WHEN THE CLOCK IS LOW (CLK=L).  BY
;MONITORING THE MACHINE STATUS CYCLE, THE 74S508 CAN BE ADDRESS MAPPED BY THE
;8085 AS IF IT WERE AN I/O DEVICE, THUS NOT USING ANY MEMORY MAP ADDRESS SPACE.
;THE MACHINE CYCLE STATUS FROM THE 8085 IS AVAILABLE ON THE FALLING EDGE OF ALE.

;FOR THIS PARTICULAR DESIGN, THE THREE INSTRUCTION INPUTS TO THE 74S508 (I0-I2)
;ARE ASSIGNED TO THE THREE LSB BITS OF ADDRESS BUS (A8-A10), WHILE THE REMAINING
;ADDRESS BITS (A11-A15) ARE DECODED BY THE PAL TO DETERMINE IF THE 74S508 IS
;SELECTED.  ALSO, ADDRESS 100 TO 107 IS RESERVED FOR THE 74S508.  THE ADDRESS
;SPACE CAN BE CHANGED BY SIMPLY EDITING THE LOGIC
```

TB00840M

74S508 Memory Map Interface #2

```
Title SN54/74S508 I/O DEVICE INTERFACE WITH THE INTEL 8085 (DESIGN # 2)
Pattern  P7086
Revision A
Author   VINCENT COLI
Company  MMI SUNNYVALE, CALIFORNIA
Date     05/15/82

CHIP SN54/74S508_I/O_DEVICE_INTERFACE_WITH_THE_INTEL_8085_(DESIGN_#_2) PAL16R4

CLK A15 A14 A13 A12 A11 A10 A9 A8 GND
/OE IOM ALE I2  I1  I0  /GO S1 S0 VCC

EQUATIONS

GO := /A11*/A12*/A13* A14*/A15    ;MONITOR ADDRESS BUS (A11-A15)
    *  IOM*/S1 * S0               ;MONITOR MACHINE CYCLE STATUS (I/O WRITE)
    * /ALE                        ;MONITOR ADDRESS/DATA CONTROL (FALLING EDGE)

/I0 := /A8      ;REGISTER INSTRUCTION INPUT I0

/I1 := /A9      ;REGISTER INSTRUCTION INPUT I1

/I2 := /A10     ;REGISTER INSTRUCTION INPUT I2


;FUNCTION TABLE FOR PALASM 1

;/OE CLK A8 A9 A10 A11 A12 A13 A14 A15 IOM S1 S0 ALE I0 I1 I2 /GO

;         ADDR BUS   MACH STATUS      INSTR           COMMENTS
;/OE CLK  89012345   IOM S1 S0   ALE  012     /GO   (OCTAL ADDRESS)
;-----------------------------------------------------------------------
; L   C   XXXLLLLH    H  L  H     L   XXX     H     (200 TO 207) INACTIVE
; L   C   XXXLLLLH    H  H  L     L   XXX     H     (100 TO 107) I/O REA
; L   C   XXXLLLHL    H  H  H     L   XXX     H     (100 TO 107) INTERRUPT ACKNOW
; L   C   XXXLLLHL    L  H  H     H   XXX     H     (100 TO 107) INACTIVE (ALE=H)
; L   C   XXXLLLHL    H  L  H     L   XXX     L     (100 TO 107) (ACTIVE RANGE)
; L   C   LHHLLLHL    H  L  H     L   LHH     L     (106) LOAD X
; L   C   LHHLLLHL    H  L  H     H   LHH     H     (106) NOP (ALE=H)
; L   C   LLLLLLHL    H  L  H     L   LLL     L     (100) LOAD Y
; H   X   XXXXXXXX    X  X  X     X   ZZZ     Z     TEST HI-Z
;-----------------------------------------------------------------------


;DESCRIPTION

;THIS PAL PROVIDES THE DECODE LOGIC FOR INTERFACING THE MMI SN54/74S508, 8-BIT
;SEQUENTIAL MULTIPLIER/DIVIDER, WITH THE INTEL 8085 MICROPROCESSOR.  THE PAL16R4
;MONITORS THE UPPER 8 BIT ADDRESS BUS (A8-A15), THE ADDRESS LATCH ENABLE (ALE),
;AND THE THREE MACHINE CYCLE STATUS LINES (IOM,S1,S0) FRO
```

TB00850M

16 Input Registered Priority Encoder #1

```
Title     16 INPUT REGISTERED PRIORITY ENCODER (CHIP No. 1)
Pattern   P7090  COLI/VOLPIGNO 10/13/82
Revision  A
Author    COLI/VOLPIGNO
Company   MMI SUNNYVALE, CALIFORNIA
Date      10/13/82

CHIP 16_INPUT_REGISTERED_PRIORITY_ENCODER_(CHIP_No._1) PAL20R4

CLK I0  I1  I2  I3 I4 I5 I6 I7  I8  I9  GND
/OC I10 I11 I12 Q3 Q2 Q1 Q0 I13 I14 I15 VCC

EQUATIONS

/Q0 := /I0* I1
     + /I0*/I1*/I2* I3
     + /I0*/I1*/I2*/I3*/I4* I5
     + /I0*/I1*/I2*/I3*/I4*/I5*/I6* I7
     + /I0*/I1*/I2*/I3*/I4*/I5*/I6*/I7*/I8* I9
     + /I0*/I1*/I2*/I3*/I4*/I5*/I6*/I7*/I8*/I9*/I10* I11
     + /I0*/I1*/I2*/I3*/I4*/I5*/I6*/I7*/I8*/I9*/I10*/I11*/I12* I13
     + /I0*/I1*/I2*/I3*/I4*/I5*/I6*/I7*/I8*/I9*/I10*/I11*/I12*/I13*/I14* I15

/Q1 := /I0*/I1* I2
     + /I0*/I1*/I2* I3
     + /I0*/I1*/I2*/I3*/I4*/I5* I6
     + /I0*/I1*/I2*/I3*/I4*/I5*/I6* I7
     + /I0*/I1*/I2*/I3*/I4*/I5*/I6*/I7*/I8*/I9* I10
     + /I0*/I1*/I2*/I3*/I4*/I5*/I6*/I7*/I8*/I9*/I10* I11
     + /I0*/I1*/I2*/I3*/I4*/I5*/I6*/I7*/I8*/I9*/I10*/I11*/I12*/I13* I14
     + /I0*/I1*/I2*/I3*/I4*/I5*/I6*/I7*/I8*/I9*/I10*/I11*/I12*/I13*/I14* I15

/Q2 := /I0*/I1*/I2*/I3* I4
     + /I0*/I1*/I2*/I3*/I4* I5
     + /I0*/I1*/I2*/I3*/I4*/I5* I6
     + /I0*/I1*/I2*/I3*/I4*/I5*/I6* I7
     + /I0*/I1*/I2*/I3*/I4*/I5*/I6*/I7*/I8*/I9*/I10*/I11* I12
     + /I0*/I1*/I2*/I3*/I4*/I5*/I6*/I7*/I8*/I9*/I10*/I11*/I12* I13
     + /I0*/I1*/I2*/I3*/I4*/I5*/I6*/I7*/I8*/I9*/I10*/I11*/I12*/I13* I14
     + /I0*/I1*/I2*/I3*/I4*/I5*/I6*/I7*/I8*/I9*/I10*/I11*/I12*/I13*/I14* I15

/Q3 := /I0*/I1*/I2*/I3*/I4*/I5*/I6*/I7* I8
     + /I0*/I1*/I2*/I3*/I4*/I5*/I6*/I7*/I8* I9
     + /I0*/I1*/I2*/I3*/I4*/I5*/I6*/I7*/I8*/I9* I10
     + /I0*/I1*/I2*/I3*/I4*/I5*/I6*/I7*/I8*/I9*/I10* I11
     + /I0*/I1*/I2*/I3*/I4*/I5*/I6*/I7*/I8*/I9*/I10*/I11* I12
     + /I0*/I1*/I2*/I3*/I4*/I5*/I6*/I7*/I8*/I9*/I10*/I11*/I12* I13
     + /I0*/I1*/I2*/I3*/I4*/I5*/I6*/I7*/I8*/I9*/I10*/I11*/I12*/I13* I14
     + /I0*/I1*/I2*/I3*/I4*/I5*/I6*/I7*/I8*/I9*/I10*/I11*/I12*/I13*/I14* I15
```

TB00860M

```
;FUNCTION TABLE FOR PALASM 1
;CLK /OC I15 I14 I13 I12 I11 I10 I9 I8 I7 I6 I5 I4 I3 I2 I1 I0 Q3 Q2 Q1 Q0

; CHIP        SIXTEEN INPUTS    OUTPUTS
;CONTROL     111111             QQQQ
;CLK /OC     5432109876543210   3210       COMMENTS
;--------------------------------------------------------------------------------
; C   L      XXXXXXXXXXXXXXXH   HHHH       I0  INTERRUPT (HIGHEST PRIORITY INPUT)
; C   L      XXXXXXXXXXXXXXHL   HHHL       I1  INTERRUPT
; C   L      XXXXXXXXXXXXXHLL   HHLH       I2  INTERRUPT
; C   L      XXXXXXXXXXXXHLLL   HHLL       I3  INTERRUPT
; C   L      XXXXXXXXXXXHLLLL   HLHH       I4  INTERRUPT
; C   L      XXXXXXXXXXHLLLLL   HLHL       I5  INTERRUPT
; C   L      XXXXXXXXXHLLLLLL   HLLH       I6  INTERRUPT
; C   L      XXXXXXXXHLLLLLLL   HLLL       I7  INTERRUPT
; C   L      XXXXXXXHLLLLLLLL   LHHH       I8  INTERRUPT
; C   L      XXXXXXHLLLLLLLLL   LHHL       I9  INTERRUPT
; C   L      XXXXXHLLLLLLLLLL   LHLH       I10 INTERRUPT
; C   L      XXXXHLLLLLLLLLLL   LHLL       I11 INTERRUPT
; C   L      XXXHLLLLLLLLLLLL   LLHH       I12 INTERRUPT
; C   L      XXHLLLLLLLLLLLLL   LLHL       I13 INTERRUPT
; C   L      XHLLLLLLLLLLLLLL   LLLH       I14 INTERRUPT
; C   L      HLLLLLLLLLLLLLLL   LLLL       I15 INTERRUPT (LOWEST PRIORITY INPUT)
; X   H      XXXXXXXXXXXXXXXX   ZZZZ       TEST HI-Z
;-------------------------------------------------------------------------------

;DESCRIPTION

;THE 16 INPUT REGISTERED PRIORITY ENCODER ACCEPTS SIXTEEN ACTIVE-LOW INPUTS
;(I0-I15) TO LOAD THE BINARY WEIGHTED CODE OF THE PRIORITY ORDER INTO THE OUTPUT
;REGISTER (Q3-Q0) ON THE RISING EDGE OF THE CLOCK (CLK).  A PRIORITY IS ASSIGNED
;TO EACH INPUT SO THAT WHEN TWO INPUTS ARE SIMULTANEOUSLY ACTIVE, THE INPUT WITH
;THE HIGHEST PRIORITY IS LOADED INTO THE OUTPUT REGISTER.  THEREFORE THE HIGHEST
;PRIORITY INPUT (I0=H) PRODUCES HHHH IN THE OUTPUT REGISTER AND THE LOWEST
;PRIORITY INPUT (I15=H) PRODUCES LLLL IN THE OUTPUT REGISTER.

;OPERATIONS TABLE

;      /OC  CLK  I15-I0   Q3-Q0    OPERATION
;      -------------------------------------------------------------
;       H    X      X        Z      HI-Z
;       L    C    I0 =H     15      I0  INTERRUPT (HIGHEST PRIORITY INPUT)
;       L    C    I1 =H     14      I1  INTERRUPT
;       L    C    I1 =H     13      I2  INTERRUPT
;       L    C    I1 =H     12      I3  INTERRUPT
;       L    C    I1 =H     11      I4  INTERRUPT
;       L    C    I1 =H     10      I5  INTERRUPT
;       L    C    I1 =H      9      I6  INTERRUPT
;       L    C    I1 =H      8      I7  INTERRUPT
;       L    C    I1 =H      7      I8  INTERRUPT
;       L    C    I1 =H      6      I9  INTERRUPT
;       L    C    I1 =H      5      I10 INTERRUPT
;       L    C    I1 =H      4      I11 INTERRUPT
;       L    C    I1 =H      3      I12 INTERRUPT
;       L    C    I1 =H      2      I13 INTERRUPT
;       L    C    I1 =H      1      I14 INTERRUPT
;       L    C    I1 =H      0      I15 INTERRUPT (LOWEST PRIORITY INPUT)
;      -------------------------------------------------------------
```

TB00870M

16 Input Priority Encoder Interrupt Flag #2

```
Title    16 INPUT PRIORITY ENCODER INTERRUPT FLAG (CHIP No. 2)
Pattern  P7091
Revision A
Author   VINCENT COLI
Company  MMI SUNNYVALE, CALIFORNIA
Date     11/13/82


CHIP 16_INPUT_PRIORITY_ENCODER_INTERRUPT_FLAG_(CHIP_No._2) PAL16C1

I0 I1  I2  I3    I4      I5     I6  I7  I8  GND
I9 I10 I11 I12 POS_INT NEG_INT I13 I14 I15 VCC

EQUATIONS

NEG_INT = /I0*/I1*/I2*/I3*/I4*/I5*/I6*/I7*/I8*/I9*/I10*/I11*/I12*/I13*/I14*/I15


;FUNCTION TABLE FOR PALASM 1

;I15 I14 I13 I12 I11 I10 I9 I8 I7 I6 I5 I4 I3 I2 I1 I0 NEG_INT POS_INT

; SIXTEEN INPUTS    INTERRUPT
;111111               FLAG
;5432109876543210    NEG POS     COMMENTS
;-------------------------------------------------------------------------
;XXXXXXXXXXXXXXXH     L   H      I0  INTERRUPT (HIGHEST PRIORITY INPUT)
;XXXXXXXXXXXXXXHL     L   H      I1  INTERRUPT
;XXXXXXXXXXXXXHLL     L   H      I2  INTERRUPT
;XXXXXXXXXXXXHLLL     L   H      I3  INTERRUPT
;XXXXXXXXXXXHLLLL     L   H      I4  INTERRUPT
;XXXXXXXXXXHLLLLL     L   H      I5  INTERRUPT
;XXXXXXXXXHLLLLLL     L   H      I6  INTERRUPT
;XXXXXXXXHLLLLLLL     L   H      I7  INTERRUPT
;XXXXXXXHLLLLLLLL     L   H      I8  INTERRUPT
;XXXXXXHLLLLLLLLL     L   H      I9  INTERRUPT
;XXXXXHLLLLLLLLLL     L   H      I10 INTERRUPT
;XXXXHLLLLLLLLLLL     L   H      I11 INTERRUPT
;XXXHLLLLLLLLLLLL     L   H      I12 INTERRUPT
;XXHLLLLLLLLLLLLL     L   H      I13 INTERRUPT
;XHLLLLLLLLLLLLLL     L   H      I14 INTERRUPT
;HLLLLLLLLLLLLLLL     L   H      I15 INTERRUPT (LOWEST PRIORITY INPUT)
;LLLLLLLLLLLLLLLL     H   L      NO  INTERRUPT INPUT
;-------------------------------------------------------------------------


;DESCRIPTION
;THE 16 INPUT PRIORITY ENCODER INTERRUPT FLAG ACCEPTS SIXTEEN ACTIVE-LOW INPUTS
;(I0-I15) IN ORDER TO GENERATE AN ACTIVE LOW (NEG_INT) AND ACTIVE HIGH (POS_INT)
;INTERRUPT FLAG.

;UP TO FOUR INTERRUPT ENABLE PINS OF ANY POLARITY CAN BE ADDED IF A PAL20C1 IS
;SUBSTITUTED.
```

TB00880M

```
Title     15 INPUT REGISTERED PRIORITY ENCODER
Pattern   P7092 COLI/VOLPIGNO 10/13/82
Revision  A
Author    COLI/VOLPIGNO
Company   MMI SUNNYVALE, CALIFORNIA
Date      10/13/82


CHIP 15_INPUT_REGISTERED_PRIORITY_ENCODER PAL20R4

CLK I0  I1   I2   I3 I4 I5 I6 I7  I8  I9  GND
/OC I10 I11 FLAG Q3 Q2 Q1 Q0 I12 I13 I14 VCC

EQUATIONS

/Q0 := /I0* I1
     + /I0*/I1*/I2* I3
     + /I0*/I1*/I2*/I3*/I4* I5
     + /I0*/I1*/I2*/I3*/I4*/I5*/I6* I7
     + /I0*/I1*/I2*/I3*/I4*/I5*/I6*/I7*/I8* I9
     + /I0*/I1*/I2*/I3*/I4*/I5*/I6*/I7*/I8*/I9*/I10* I11
     + /I0*/I1*/I2*/I3*/I4*/I5*/I6*/I7*/I8*/I9*/I10*/I11*/I12* I13
     + /I0*/I1*/I2*/I3*/I4*/I5*/I6*/I7*/I8*/I9*/I10*/I11*/I12*/I13*/I14

/Q1 := /I0*/I1* I2
     + /I0*/I1*/I2* I3
     + /I0*/I1*/I2*/I3*/I4*/I5* I6
     + /I0*/I1*/I2*/I3*/I4*/I5*/I6* I7
     + /I0*/I1*/I2*/I3*/I4*/I5*/I6*/I7*/I8*/I9* I10
     + /I0*/I1*/I2*/I3*/I4*/I5*/I6*/I7*/I8*/I9*/I10* I11
     + /I0*/I1*/I2*/I3*/I4*/I5*/I6*/I7*/I8*/I9*/I10*/I11*/I12*/I13* I14
     + /I0*/I1*/I2*/I3*/I4*/I5*/I6*/I7*/I8*/I9*/I10*/I11*/I12*/I13*/I14

/Q2 := /I0*/I1*/I2*/I3* I4
     + /I0*/I1*/I2*/I3*/I4* I5
     + /I0*/I1*/I2*/I3*/I4*/I5* I6
     + /I0*/I1*/I2*/I3*/I4*/I5*/I6* I7
     + /I0*/I1*/I2*/I3*/I4*/I5*/I6*/I7*/I8*/I9*/I10*/I11* I12
     + /I0*/I1*/I2*/I3*/I4*/I5*/I6*/I7*/I8*/I9*/I10*/I11*/I12* I13
     + /I0*/I1*/I2*/I3*/I4*/I5*/I6*/I7*/I8*/I9*/I10*/I11*/I12*/I13* I14
     + /I0*/I1*/I2*/I3*/I4*/I5*/I6*/I7*/I8*/I9*/I10*/I11*/I12*/I13*/I14

/Q3 := /I0*/I1*/I2*/I3*/I4*/I5*/I6*/I7* I8
     + /I0*/I1*/I2*/I3*/I4*/I5*/I6*/I7*/I8* I9
     + /I0*/I1*/I2*/I3*/I4*/I5*/I6*/I7*/I8*/I9* I10
     + /I0*/I1*/I2*/I3*/I4*/I5*/I6*/I7*/I8*/I9*/I10* I11
     + /I0*/I1*/I2*/I3*/I4*/I5*/I6*/I7*/I8*/I9*/I10*/I11* I12
     + /I0*/I1*/I2*/I3*/I4*/I5*/I6*/I7*/I8*/I9*/I10*/I11*/I12* I13
     + /I0*/I1*/I2*/I3*/I4*/I5*/I6*/I7*/I8*/I9*/I10*/I11*/I12*/I13* I14
     + /I0*/I1*/I2*/I3*/I4*/I5*/I6*/I7*/I8*/I9*/I10*/I11*/I12*/I13*/I14

/FLAG = /I0*/I1*/I2*/I3*/I4*/I5*/I6*/I7*/I8*/I9*/I10*/I11*/I12*/I13*/I14
```

TB00890M

```
;FUNCTION TABLE FOR PALASM 1
;CLK /OC I14 I13 I12 I11 I10 I9 I8 I7 I6 I5 I4 I3 I2 I1 I0 Q3 Q2 Q1 Q0 FLAG

; CHIP     FIFTEEN  INPUTS    --OUTPUTS--
;CONTROL  11111               QQQQ
;CLK /OC  432109876543210     3210   FLAG   COMMENTS
;----------------------------------------------------------------------------
; C    L  XXXXXXXXXXXXXXH     HHHH    H     I0  INTERRUPT (HIGHEST PRIORITY INPUT)
; C    L  XXXXXXXXXXXXXHL     HHHL    H     I1  INTERRUPT
; C    L  XXXXXXXXXXXXHLL     HHLH    H     I2  INTERRUPT
; C    L  XXXXXXXXXXXHLLL     HHLL    H     I3  INTERRUPT
; C    L  XXXXXXXXXXHLLLL     HLHH    H     I4  INTERRUPT
; C    L  XXXXXXXXXHLLLLL     HLHL    H     I5  INTERRUPT;
; C    L  XXXXXXXXHLLLLLL     HLLH    H     I6  INTERRUPT
; C    L  XXXXXXXHLLLLLLL     HLLL    H     I7  INTERRUPT
; C    L  XXXXXXHLLLLLLLL     LHHH    H     I8  INTERRUPT
; C    L  XXXXXHLLLLLLLLL     LHHL    H     I9  INTERRUPT
; C    L  XXXXHLLLLLLLLLL     LHLH    H     I10 INTERRUPT
; C    L  XXXHLLLLLLLLLLL     LHLL    H     I11 INTERRUPT
; C    L  XXHLLLLLLLLLLLL     LLHH    H     I12 INTERRUPT
; C    L  XHLLLLLLLLLLLLL     LLHL    H     I13 INTERRUPT
; C    L  HLLLLLLLLLLLLLL     LLLH    H     I14 INTERRUPT (LOWEST PRIORITY INPUT)
; C    L  LLLLLLLLLLLLLLL     LLLL    L     NO  INTERRUPT
; X    H  XXXXXXXXXXXXXXX     ZZZZ    X     TEST HI-Z
;---------------------------------------------------------------------------

;DESCRIPTION

;THE 15 INPUT REGISTERED PRIORITY ENCODER ACCEPTS FIFTEEN ACTIVE-LOW INPUTS
;(I0-I14) TO LOAD THE BINARY WEIGHTED CODE OF THE PRIORITY ORDER INTO THE OUTPUT
;REGISTER (Q3-Q0) ON THE RISING EDGE OF THE CLOCK (CLK).  A PRIORITY IS ASSIGNED
;TO EACH INPUT SO THAT WHEN TWO INPUTS ARE SIMULTANEOUSLY ACTIVE, THE INPUT WITH
;THE HIGHEST PRIORITY IS LOADED INTO THE OUTPUT REGISTER.  THEREFORE THE HIGHEST
;PRIORITY INPUT (I0=H) PRODUCES HHHH IN THE OUTPUT REGISTER AND THE LOWEST
;PRIORITY INPUT (I14=H) PRODUCES LLLH IN THE OUTPUT REGISTER.  THE NON-REGIST-
;ERED INTERRUPT FLAG (FLAG) GOES HIGH WHEN AN INTERRUPT IS PRESENT AND REMAINS
;LOW WHEN THERE IS NO INTERRUPT PRESENT.

;OPERATIONS TABLE

;      /OC  CLK  I14-I0   Q3-Q0   FLAG     OPERATION
;      ---------------------------------------------------------------------
;      H    X      X        Z      X      HI-Z
;      L    C      L        0      L      NO INTERRUPT
;      L    C    I0 =H     15      H      I0  INTERRUPT (HIGHEST PRIORITY INPUT)
;      L    C    I1 =H     14      H      I1  INTERRUPT
;      L    C    I2 =H     13      H      I2  INTERRUPT
;      L    C    I3 =H     12      H      I3  INTERRUPT
;      L    C    I4 =H     11      H      I4  INTERRUPT
;      L    C    I5 =H     10      H      I5  INTERRUPT
;      L    C    I6 =H      9      H      I6  INTERRUPT
;      L    C    I7 =H      8      H      I7  INTERRUPT
;      L    C    I8 =H      7      H      I8  INTERRUPT
;      L    C    I9 =H      6      H      I9  INTERRUPT
;      L    C    I10=H      5      H      I10 INTERRUPT
;      L    C    I11=H      4      H      I11 INTERRUPT
;      L    C    I12=H      3      H      I12 INTERRUPT
;      L    C    I13=H      2      H      I13 INTERRUPT
;      L    C    I14=H      1      H      I14 INTERRUPT (LOWEST PRIORITY INPUT)
;     ----------------------------------------------------------------------
```

TB00900M

```
Title    8 INPUT REGISTERED PRIORITY ENCODER WITH INTERRUPT FLAG
Pattern  P7093
Revision A
Author   VINCENT COLI
Company  MMI SUNNYVALE, CALIFORNIA
Date     06/21/82

CHIP 8_INPUT_REGISTERED_PRIORITY_ENCODER_WITH_INTERRUPT_FLAG PAL16R4

CLK  I0  I1 I2 I3 I4 I5 I6 I7 GND
/OC /E4 /E3 Q3 Q2 Q1 Q0 E2 E1 VCC

EQUATIONS

/Q0 := /I0* I1                             ;DECODE I0=L   AND I1=H
     + /I0*/I1*/I2* I3                     ;DECODE I0-2=L AND I3=H
     + /I0*/I1*/I2*/I3*/I4* I5             ;DECODE I0-4=L AND I5=H
     + /I0*/I1*/I2*/I3*/I4*/I5*/I6* I7     ;DECODE I0-6=L AND I7=H

/Q1 := /I0*/I1* I2                         ;DECODE I0-1=L AND I2=H
     + /I0*/I1*/I2* I3                     ;DECODE I0-2=L AND I3=H
     + /I0*/I1*/I2*/I3*/I4*/I5* I6         ;DECODE I0-5=L AND I6=H
     + /I0*/I1*/I2*/I3*/I4*/I5*/I6* I7     ;DECODE I0-6=L AND I7=H

/Q2 := /I0*/I1*/I2*/I3* I4                 ;DECODE I0-3=L AND I4=H
     + /I0*/I1*/I2*/I3*/I4* I5             ;DECODE I0-4=L AND I5=H
     + /I0*/I1*/I2*/I3*/I4*/I5* I6         ;DECODE I0-5=L AND I6=H
     + /I0*/I1*/I2*/I3*/I4*/I5*/I6* I7     ;DECODE I0-6=L AND I7=H

/Q3 :=  E1* E2* E3* E4* I0                 ;INTERRUPT FLAG (I0)
     +  E1* E2* E3* E4* I1                 ;INTERRUPT FLAG (I1)
     +  E1* E2* E3* E4* I2                 ;INTERRUPT FLAG (I2)
     +  E1* E2* E3* E4* I3                 ;INTERRUPT FLAG (I3)
     +  E1* E2* E3* E4* I4                 ;INTERRUPT FLAG (I4)
     +  E1* E2* E3* E4* I5                 ;INTERRUPT FLAG (I5)
     +  E1* E2* E3* E4* I6                 ;INTERRUPT FLAG (I6)
     +  E1* E2* E3* E4* I7                 ;INTERRUPT FLAG (I7)
```

TB00910M

```
;FUNCTION TABLE FOR PALASM 1

;CLK /OC E1 E2 /E3 /E4 I7 I6 I5 I4 I3 I2 I1 I0 Q3 Q2 Q1 Q0

; CHIP      ----FOUR-----    8 INPUTS   OUTPUTS
;CONTROL    INPUT ENABLES    IIIIIIII    Q QQQ
;CLK /OC    E1 E2 /E3 /E4    76543210    3 210    COMMENTS
;------------------------------------------------------------------------------
; C    L    L  H   L    L    XXXXXXXH    H HHH    I0 INTERRUPT NOT ENABLED (E1=L)
; C    L    H  L   L    L    XXXXXXXH    H HHH    I0 INTERRUPT NOT ENABLED (E2=L)
; C    L    H  H   H    L    XXXXXXXH    H HHH    I0 INTERRUPT NOT ENABLED(/E3=H)
; C    L    H  H   L    H    XXXXXXXH    H HHH    I0 INTERRUPT NOT ENABLED(/E4=H)
; C    L    H  H   L    L    XXXXXXXH    L HHH    I0 INTERRUPT (HIGHEST PRIORITY)
; C    L    L  H   L    L    XXXXXXHL    H HHL    I1 INTERRUPT NOT ENABLED (E1=L)
; C    L    H  L   L    L    XXXXXXHL    H HHL    I1 INTERRUPT NOT ENABLED (E2=L)
; C    L    H  H   H    L    XXXXXXHL    H HHL    I1 INTERRUPT NOT ENABLED(/E3=H)
; C    L    H  H   L    H    XXXXXXHL    H HHL    I1 INTERRUPT NOT ENABLED(/E4=H)
; C    L    H  H   L    L    XXXXXXHL    L HHL    I1 INTERRUPT
; C    L    L  H   L    L    XXXXXHLL    H HLH    I2 INTERRUPT NOT ENABLED (E1=L)
; C    L    H  L   L    L    XXXXXHLL    H HLH    I2 INTERRUPT NOT ENABLED (E2=L)
; C    L    H  H   H    L    XXXXXHLL    H HLH    I2 INTERRUPT NOT ENABLED(/E3=H)
; C    L    H  H   L    H    XXXXXHLL    H HLH    I2 INTERRUPT NOT ENABLED(/E4=H)
; C    L    H  H   L    L    XXXXXHLL    L HLH    I2 INTERRUPT
; C    L    L  H   L    L    XXXXHLLL    H HLL    I3 INTERRUPT NOT ENABLED (E1=L)
; C    L    H  L   L    L    XXXXHLLL    H HLL    I3 INTERRUPT NOT ENABLED (E2=L)
; C    L    H  H   H    L    XXXXHLLL    H HLL    I3 INTERRUPT NOT ENABLED(/E3=H)
; C    L    H  H   L    H    XXXXHLLL    H HLL    I3 INTERRUPT NOT ENABLED(/E4=H)
; C    L    H  H   L    L    XXXXHLLL    L HLL    I3 INTERRUPT
; C    L    L  H   L    L    XXXHLLLL    H LHH    I4 INTERRUPT NOT ENABLED (E1=L)
; C    L    H  L   L    L    XXXHLLLL    H LHH    I4 INTERRUPT NOT ENABLED (E2=L)
; C    L    H  H   H    L    XXXHLLLL    H LHH    I4 INTERRUPT NOT ENABLED(/E3=H)
; C    L    H  H   L    H    XXXHLLLL    H LHH    I4 INTERRUPT NOT ENABLED(/E4=H)
; C    L    H  H   L    L    XXXHLLLL    L LHH    I4 INTERRUPT
; C    L    L  H   L    L    XXHLLLLL    H LHL    I5 INTERRUPT NOT ENABLED (E1=L)
; C    L    H  L   L    L    XXHLLLLL    H LHL    I5 INTERRUPT NOT ENABLED (E2=L)
; C    L    H  H   H    L    XXHLLLLL    H LHL    I5 INTERRUPT NOT ENABLED(/E3=H)
; C    L    H  H   L    H    XXHLLLLL    H LHL    I5 INTERRUPT NOT ENABLED(/E4=H)
; C    L    H  H   L    L    XXHLLLLL    L LHL    I5 INTERRUPT
; C    L    L  H   L    L    XHLLLLLL    H LLH    I6 INTERRUPT NOT ENABLED (E1=L)
; C    L    H  L   L    L    XHLLLLLL    H LLH    I6 INTERRUPT NOT ENABLED (E2=L)
; C    L    H  H   H    L    XHLLLLLL    H LLH    I6 INTERRUPT NOT ENABLED(/E3=H)
; C    L    H  H   L    H    XHLLLLLL    H LLH    I6 INTERRUPT NOT ENABLED(/E4=H)
; C    L    H  H   L    L    XHLLLLLL    L LLH    I6 INTERRUPT
; C    L    L  H   L    L    HLLLLLLL    H LLL    I7 INTERRUPT NOT ENABLED (E1=L)
; C    L    H  L   L    L    HLLLLLLL    H LLL    I7 INTERRUPT NOT ENABLED (E2=L)
; C    L    H  H   H    L    HLLLLLLL    H LLL    I7 INTERRUPT NOT ENABLED(/E3=H)
; C    L    H  H   L    H    HLLLLLLL    H LLL    I7 INTERRUPT NOT ENABLED(/E4=H)
; C    L    H  H   L    L    HLLLLLLL    L LLL    I7 INTERRUPT (LOWEST PRIORITY)
; C    L    H  H   L    L    LLLLLLLL    H HHH    NO INTERRUPT INPUT (Q3=H)
; X    H    X  X   X    X    XXXXXXXX    Z ZZZ    TEST HI-Z
;------------------------------------------------------------------------------
```

TB00920M

```
;DESCRIPTION

;THE 8 INPUT REGISTERED PRIORITY ENCODER ACCEPTS SIXTEEN ACTIVE-LOW INPUTS
;(I0-I7) TO LOAD THE BINARY WEIGHTED CODE OF THE PRIORITY ORDER INTO THE OUTPUT
;REGISTER (Q2-Q0) ON THE RISING EDGE OF THE CLOCK (CLK) PROVIDING THE FOUR
;ENABLE INPUTS ARE TRUE (E1=H,E2=H,/E3=L,/E4=L).  A PRIORITY IS ASSIGNED TO EACH
;INPUT SO THAT WHEN TWO INPUTS ARE SIMULTANEOUSLY ACTIVE, THE INPUT WITH THE
;HIGHEST PRIORITY IS LOADED INTO THE OUTPUT REGISTER.  THEREFORE THE HIGHEST
;PRIORITY INPUT (I0=H) PRODUCES HHH IN THE OUTPUT REGISTER AND THE LOWEST
;PRIORITY INPUT (I7=H) PRODUCES LLL IN THE OUTPUT REGISTER.

;THE PRIORITY INTERRUPT ENCODER REGISTERS (Q3-Q0) ARE UPDATED ON THE RISING EDGE
;OF THE CLOCK (CLK) PROVIDING THE FOUR ENABLE INPUTS ARE TRUE (E1=H,E2=H,/E3=L,
;/E4=L).  THE PREVIOUS DATA IS HELD IN THE PRIORITY ENCODER REGISTERS IF ANY OF
;THE ENABLE INPUTS ARE FALSE (E1=L,E2=L,/E3=H,/E4=H) REGARDLESS OF CLOCK
;TRANSITIONS.  NOTE THAT THE POLARITY OF THE ENABLES CAN BE CHANGED BY MERELY
;EDITING THE LOGIC EQUATIONS.

;OUTPUT Q4 SERVES AS THE INTERRUPT FLAG AND IS TRUE (Q4=L) WHEN ANY OF THE 8
;INPUTS ARE ACTIVE (I=H) ON THE RISING EDGE OF THE CLOCK (CLK) PROVIDING THE
;FOUR ENABLE INPUTS ARE TRUE (E1=H,E2=H,/E3=L,/E4=L).  THE INTERRUPT FLAG IS
;FALSE (Q4=H) WHEN ALL INPUTS ARE INACTIVE (I=L) OR WHEN ANY ONE OF THE FOUR
;ENABLE INPUTS ARE FALSE (E1=L,E2=L,/E3=H,/E4=H).

;OPERATIONS TABLE

;/OC  CLK  E1 E2 /E3 /E4  I7-I0  Q4  Q3-Q0   OPERATION
;-----------------------------------------------------------------------
; H    X    X  X   X   X     X    Z    Z     HI-Z
; L    C    L  X   X   X     X    H    Q     NOT ENABLED  (E1=L)
; L    C    X  L   X   X     X    H    Q     NOT ENABLED  (E2=L)
; L    C    X  X   H   X     X    H    Q     NOT ENABLED (/E3=H)
; L    C    X  X   X   H     X    H    Q     NOT ENABLED (/E4=H)
; L    C    H  H   L   L     L    H    X     NO INTERRUPT FLAG
; L    C    H  H   L   L   I0=H   L    7     I0  INTERRUPT (HIGHEST PRIORITY)
; L    C    H  H   L   L   I1=H   L    6     I1  INTERRUPT
; L    C    H  H   L   L   I1=H   L    5     I2  INTERRUPT
; L    C    H  H   L   L   I1=H   L    4     I3  INTERRUPT
; L    C    H  H   L   L   I1=H   L    3     I4  INTERRUPT
; L    C    H  H   L   L   I1=H   L    2     I5  INTERRUPT
; L    C    H  H   L   L   I1=H   L    1     I6  INTERRUPT
; L    C    H  H   L   L   I1=H   L    0     I7  INTERRUPT (LOWEST PRIORITY)
;-----------------------------------------------------------------------
```

TB00930M

```
Title    DUAL STEPPER MOTOR CONTROLLER
Pattern  P7094
Revision A
Author   COLI/SACKETTE
Company  MMI SUNNYVALE, CALIFORNIA
Date     12/07/82

CHIP DUAL_STEPPER_MOTOR_CONTROLLER PAL16R8

CLK /ENA   SETA  ROTA MODEA /ENB   SETB  ROTB MODEB GND
/OC /SW1B /SW2B /SW3B /SW4B /SW4A /SW3A /SW2A /SW1A VCC

EQUATIONS

SW1A :=  SW1A                  */ENA                       ;HOLD SW1A (DISABLE)
      +       /SW2A      */SW4A* ENA* ROTA* MODEA          ;HALF-STEP MOTOR CW
      +              SW3A      * ENA* ROTA*/MODEA          ;FULL-STEP MOTOR CW
      +       /SW2A*/SW3A      * ENA*/ROTA* MODEA          ;HALF-STEP MOTOR CCW
      +                    SW4A* ENA*/ROTA*/MODEA          ;FULL-STEP MOTOR CCW
      +                          ENA             * SETA    ;SET A TO STEP 1

SW2A :=        SW2A            */ENA                       ;HOLD SW2A (DISABLE)
      + /SW1A      */SW3A      * ENA* ROTA* MODEA*/SETA    ;HALF-STEP MOTOR CW
      +                    SW4A* ENA* ROTA*/MODEA*/SETA    ;FULL-STEP MOTOR CW
      + /SW1A            */SW4A* ENA*/ROTA* MODEA*/SETA    ;HALF-STEP MOTOR CCW
      +              SW3A      * ENA*/ROTA*/MODEA*/SETA    ;FULL-STEP MOTOR CCW

SW3A :=              SW3A      */ENA                       ;HOLD SW3A (DISABLE)
      + /SW1A            */SW4A* ENA* ROTA* MODEA          ;HALF-STEP MOTOR CW
      +        SW2A            * ENA* ROTA*/MODEA          ;FULL-STEP MOTOR CW
      +       /SW2A      */SW4A* ENA*/ROTA* MODEA          ;HALF-STEP MOTOR CCW
      +  SW1A                  * ENA*/ROTA*/MODEA          ;FULL-STEP MOTOR CCW
      +                          ENA             * SETA    ;SET A TO STEP 1

SW4A :=                    SW4A*/ENA                       ;HOLD SW4A (DISABLE)
      +       /SW2A*/SW3A      * ENA* ROTA* MODEA*/SETA    ;HALF-STEP MOTOR CW
      +  SW1A                  * ENA* ROTA*/MODEA*/SETA    ;FULL-STEP MOTOR CW
      + /SW1A      */SW3A      * ENA*/ROTA* MODEA*/SETA    ;HALF-STEP MOTOR CCW
      +        SW2A            * ENA*/ROTA*/MODEA*/SETA    ;FULL-STEP MOTOR CCW

SW1B :=  SW1B                  */ENB                       ;HOLD SW1B (DISABLE)
      +       /SW2B      */SW4B* ENB* ROTB* MODEB          ;HALF-STEP MOTOR CW
      +              SW3B      * ENB* ROTB*/MODEB          ;FULL-STEP MOTOR CW
      +       /SW2B*/SW3B      * ENB*/ROTB* MODEB          ;HALF-STEP MOTOR CCW
      +                    SW4B* ENB*/ROTB*/MODEB          ;FULL-STEP MOTOR CCW
      +                          ENB             * SETB    ;SET B TO STEP 1

SW2B :=        SW2B            */ENB                       ;HOLD SW2B (DISABLE)
      + /SW1B      */SW3B      * ENB* ROTB* MODEB*/SETB    ;HALF-STEP MOTOR CW
      +                    SW4B* ENB* ROTB*/MODEB*/SETB    ;FULL-STEP MOTOR CW
      + /SW1B            */SW4B* ENB*/ROTB* MODEB*/SETB    ;HALF-STEP MOTOR CCW
      +              SW3B      * ENB*/ROTB*/MODEB*/SETB    ;FULL-STEP MOTOR CCW

SW3B :=              SW3B      */ENB                       ;HOLD SW3B (DISABLE)
      + /SW1B            */SW4B* ENB* ROTB* MODEB          ;HALF-STEP MOTOR CW
      +        SW2B            * ENB* ROTB*/MODEB          ;FULL-STEP MOTOR CW
      +       /SW2B      */SW4B* ENB*/ROTB* MODEB          ;HALF-STEP MOTOR CCW
      +  SW1B                  * ENB*/ROTB*/MODEB          ;FULL-STEP MOTOR CCW
      +                          ENB             * SETB    ;SET B TO STEP 1
```

TB00940M

```
SW4B :=                       SW4B*/ENB                   ;HOLD SW4B (DISABLE)
      +        /SW2B*/SW3B        * ENB* ROTB* MODEB*/SETB    ;HALF-STEP MOTOR CW
      +  SW1B                     * ENB* ROTB*/MODEB*/SETB    ;FULL-STEP MOTOR CW
      + /SW1B      */SW3B         * ENB*/ROTB* MODEB*/SETB    ;HALF-STEP MOTOR CCW
      +         SW2B              * ENB*/ROTB*/MODEB*/SETB    ;FULL-STEP MOTOR CCW

;FUNCTION TABLE FOR PALASM 1

;CLK /OC /ENA SETA ROTA MODEA SW1A SW2A SW3A SW4A /ENB SETB ROTB MODEB SW1B
;SW2B SW3B SW4B

; CHIP     STEPPER MOTOR A  SSSS   STEPPER MOTOR B  SSSS
;CONTROL                    WWWW                    WWWW
;CLK /OC   /EN SET ROT MODE 1234   /EN SET ROT MODE 1234   COMMENTS
;-----------------------------------------------------------------------------
; C   L     L   H   X   X   HLHL    L   H   X   X   HLHL   SET A AND B TO STEP 1
; C   L     L   L   H   H   HLLL    L   L   L   H   LLHL   FULL STEP A CW, B CCW
; C   L     L   L   H   H   HLLH    L   L   L   H   LHHL   FULL STEP A CW, B CCW
; C   L     H   L   H   H   HLLH    H   L   L   H   LHHL   HOLD MOTOR POSITION
; C   L     L   L   H   H   LLLH    L   L   L   H   LHLL   FULL STEP A CW, B CCW
; C   L     L   L   H   H   LHLH    L   L   L   H   LHLH   FULL STEP A CW, B CCW
; C   L     L   L   H   H   LHLL    L   L   L   H   LLLH   FULL STEP A CW, B CCW
; C   L     L   L   H   H   LHHL    L   L   L   H   HLLH   FULL STEP A CW, B CCW
; C   L     L   L   H   H   LLHL    L   L   L   H   HLLL   FULL STEP A CW, B CCW
; C   L     L   L   H   H   HLHL    L   L   L   H   HLHL   FULL STEP A CW, B CCW
; C   L     H   L   L   L   HLHL    H   L   H   L   HLHL   HOLD MOTOR POSITION
; C   L     L   L   L   L   LHHL    L   L   H   L   HLLH   HALF STEP A CCW, B CW
; C   L     L   L   L   L   LHLH    L   L   H   L   LHLH   HALF STEP A CCW, B CW
; C   L     L   L   L   L   HLLH    L   L   H   L   LHHL   HALF STEP A CCW, B CW
; C   L     L   L   L   L   HLHL    L   L   H   L   HLHL   HALF STEP A CCW, B CW
; X   H     X   X   X   X   ZZZZ    X   X   X   X   ZZZZ   TEST HI-Z
;-----------------------------------------------------------------------------


;DESCRIPTION
;CLK  /OC  /EN  SET  ROT  MODE   SW1-SW4    COMMENTS
;------------------------------------------------------------------------------
;X     H    X    X    X    X        Z       HI-Z
;C     L    H    X    X    X      HOLD      HOLD MOTOR POSITION
;C     L    L    H    X    X        1       SET MOTOR POSITION TO STEP 1
;C     L    L    L    H    H    SW PLUS  1  HALF-STEP MOTOR CLOCKWISE
;C     L    L    L    H    L    SW PLUS  2  FULL-STEP MOTOR CLOCKWISE
;C     L    L    L    L    H    SW MINUS 1  HALF-STEP MOTOR COUNTERCLOCKWISE
;C     L    L    L    L    L    SW MINUS 2  FULL-STEP MOTOR COUNTERCLOCKWISE
;------------------------------------------------------------------------------
```

TB00950M

```
Title    CLEAN OCTAL LATCH
Pattern  P7096
Revision A
Author   VINCENT COLI
Company  MMI SUNNYVALE, CALIFORNIA
Date     03/10/83

CHIP CLEAN_OCTAL_LATCH PAL20L10

 G  NC D0 D1 D2 D3 D4 D5 D6 D7 NC GND
/OC NC Q7 Q6 Q5 Q4 Q3 Q2 Q1 Q0 NC VCC

EQUATIONS

IF(OC)  /Q0 =   G*/D0      ;LOAD LATCH (Q0)
            +  /G*/Q0      ;LATCH OUTPUT
            + /D0*/Q0      ;COVER ALWAYS HIGH HAZARD

IF(OC)  /Q1 =   G*/D1      ;LOAD LATCH (Q1)
            +  /G*/Q1      ;LATCH OUTPUT
            + /D1*/Q1      ;COVER ALWAYS HIGH HAZARD

IF(OC)  /Q2 =   G*/D2      ;LOAD LATCH (Q2)
            +  /G*/Q2      ;LATCH OUTPUT
            + /D2*/Q2      ;COVER ALWAYS HIGH HAZARD

IF(OC)  /Q3 =   G*/D3      ;LOAD LATCH (Q3)
            +  /G*/Q3      ;LATCH OUTPUT
            + /D3*/Q3      ;COVER ALWAYS HIGH HAZARD

IF(OC)  /Q4 =   G*/D4      ;LOAD LATCH (Q4)
            +  /G*/Q4      ;LATCH OUTPUT
            + /D4*/Q4      ;COVER ALWAYS HIGH HAZARD

IF(OC)  /Q5 =   G*/D5      ;LOAD LATCH (Q5)
            +  /G*/Q5      ;LATCH OUTPUT
            + /D5*/Q5      ;COVER ALWAYS HIGH HAZARD

IF(OC)  /Q6 =   G*/D6      ;LOAD LATCH (Q6)
            +  /G*/Q6      ;LATCH OUTPUT
            + /D6*/Q6      ;COVER ALWAYS HIGH HAZARD

IF(OC)  /Q7 =   G*/D7      ;LOAD LATCH (Q7)
            +  /G*/Q7      ;LATCH OUTPUT
            + /D7*/Q7      ;COVER ALWAYS HIGH HAZARD
```

TB00960M

```
;FUNCTION TABLE FOR PALASM 1

/OC G D7 D6 D5 D4 D3 D2 D1 D0 Q7 Q6 Q5 Q4 Q3 Q2 Q1 Q0

;CONTROL    DDDDDDDD    QQQQQQQQ
;/OC   G    76543210    76543210      COMMENTS
;--------------------------------------------------------------
; L    H    LLLLLLLL    LLLLLLLL      LOAD  ALL ZEROS
; L    L    XXXXXXXX    LLLLLLLL      LATCH ALL ZEROS
; L    H    HHHHHHHH    HHHHHHHH      LOAD  ALL ONES
; L    L    XXXXXXXX    HHHHHHHH      LATCH ALL ONES
; L    H    HLHLHLHL    HLHLHLHL      LOAD  EVEN CHECKERBOARD
; L    L    XXXXXXXX    HLHLHLHL      LATCH EVEN CHECKERBOARD
; L    H    LHLHLHLH    LHLHLHLH      LOAD  ODD  CHECKERBOARD
; L    L    XXXXXXXX    LHLHLHLH      LATCH ODD  CHECKERBOARD
; H    X    XXXXXXXX    ZZZZZZZZ      TEST HI-Z
;---------------------------------------------------------------


;DESCRIPTION

;THIS PAL16L8 IMPLEMENTS AN 8-BIT LATCH FUNCTION WITH THREE-STATE OUTPUTS.  THE
;LATCH PASSES EIGHT BITS OF DATA (D7-D0) TO THE EIGHT OUTPUTS (Q7-Q0) WHEN LATCH
;ENABLE IS TRUE (G=HIGH).  THE DATA IS LATCHED WHEN LATCH ENABLE IS FALSE
;(G=LOW).  THE OUTPUTS WILL BE DISABLED (HI-Z) WHEN OUTPUT ENABLE IS TRUE
;(/OC=TRUE) REGARDLESS OF ANY OTHER INPUTS.


;/OC  G  D7-D0  Q7-Q0    COMMENTS
;-------------------------------------
; H   X    X      Z      HI-Z
; L   L    X      Q      LATCH OUTPUT
; L   H    D      D      LOAD LATCH
;-------------------------------------


;THIS DESIGN SHOWS HOW TO IMPLEMENT A "CLEAN" LATCH SINCE THERE ARE NO OUTPUT
;GLITCHES AS THE DEVICE CHANGES STATE.  THE KARNAUGH MAP BELOW FOR O+
;ILLUSTRATES THIS.  THE TWO HORIZONTAL CIRCLES REPRESENT THE "LOAD LATCH" AND
;"LATCH OUTPUT" PRODUCT LINES.  THE VERTICAL CIRCLE LINKS TOGETHER THE OTHER
;CIRCLES IN ORDER TO COVER A POTENTIAL SWITCHING HAZARD WHICH WOULD OCCUR WHEN
;THE OUTPUTS ARE ALWAYS HIGH.

;G !_
;    !_
;      !- 00   01   11   10
;       !----!----!----!----!
;     0 !  0 !  1 !  1 !  0 !
;       !-------------------!
;     1 !  0 !  1 !  1 !  1 !
;       !----!----!----!----!
```

TB00970M

```
Title    SHAFT ENCODER No. 1PAL16R4
Pattern  P7097
Revision A
Author   WILLY VOLDAN
Company  MMI GMBH MUNICH
Date     09/09/82

CHIP SHAFT_ENCODER_No._1 PAL16R4

CLK PHIO PHI90 X4 NC NC NC NC /SSET GND
/OC DOWN  NC   S4 S3 S2 S1 NC  UP   VCC

EQUATIONS

/S1 :=  /PHIO                                              ;CHECK FOR PHIO
     +   SSET                                              ;INITIALIZE S1=L

/S2 :=  /S1                                                ;CHECK FOR S1
     +   SSET                                              ;INITIALIZE S2=L

/S3 :=  /PHI90                                             ;CHECK FOR PHI90
     +   SSET                                              ;INITIALIZE S3=L

/S4 :=  /S3                                                ;CHECK FOR S3
     +   SSET                                              ;INITIALIZE S4=L

IF (VCC) /DOWN =  S1* S2* S3*/S4* PHIO* PHI90              ;PHIO LEADS PHI90
               + /S1*/S2*/S3* S4*/PHIO*/PHI90              ;PHIO LEADS PHI90
               +  S1*/S2*/S3*/S4* PHIO*/PHI90              ;PHIO LEADS PHI90
               + /S1* S2* S3* S4*/PHIO* PHI90              ;PHIO LEADS PHI90

IF (VCC) /UP   = /S1*/S2* S3*/S4*/PHIO* PHI90              ;PHI90 LEADS PHIO
               +  S1* S2*/S3* S4* PHIO*/PHI90              ;PHI90 LEADS PHIO
               +  S1*/S2* S3* S4* PHIO* PHI90              ;PHI90 LEADS PHIO
               + /S1* S2*/S3*/S4*/PHIO*/PHI90              ;PHI90 LEADS PHIO

;FUNCTION TABLE FOR PALASM 1

;CLK /OC /SSET PHIO PHI90 S4 S3 S2 S1 UP DOWN
```

TB00980M

Shaft Encoder #1

```
;---CONTROLS---        --INPUTS--      SSSS     OUTPUTS
;CLK  /OC  /SSET       PHI0 PHI90      4321     UP DOWN       COMMENTS
;-----------------------------------------------------------------------------
; C    L    L          X     X         LLLL     H    H        CLEAR REGISTERS
; C    L    H          L     L         LLLL     H    H
; C    L    H          L     L         LLLL     H    H
; C    L    H          L     H         LHLL     L    H        COUNT UP
; C    L    H          L     H         HHLL     H    H
; C    L    H          H     H         HHLH     L    H        COUNT UP
; C    L    H          H     H         HHHH     H    H
; C    L    H          H     L         HLHH     L    H        COUNT UP
; C    L    H          H     L         LLHH     H    H
; C    L    H          L     L         LLHL     L    H        COUNT UP
; C    L    H          L     L         LLLL     H    H
; C    L    H          L     H         LHLL     L    H        COUNT UP
; C    L    H          L     H         HHLL     H    H
; C    L    H          H     H         HHLH     L    H        COUNT UP
; C    L    H          H     H         HHHH     H    H
; C    L    H          H     L         HLHH     L    H        COUNT UP
; C    L    H          H     L         LLHH     H    H
; C    L    H          L     L         LLHL     L    H        COUNT UP
; C    L    H          L     L         LLLL     H    H
; C    L    H          L     H         LHLL     L    H        COUNT UP
; C    L    H          L     H         HHLL     H    H
; C    L    H          H     H         HHLH     L    H        COUNT UP
; C    L    H          H     H         HHHH     H    H
; C    L    H          H     L         HLHH     L    H        COUNT UP
; C    L    H          H     L         LLHH     H    H
; C    L    H          L     L         LLHL     L    H        COUNT UP
; C    L    H          L     L         LLLL     H    H
; C    L    L          X     X         LLLL     H    H        CLEAR REGISTERS
; C    L    H          L     L         LLLL     H    H
; C    L    H          L     L         LLLL     H    H
; C    L    H          H     L         LLLH     H    L        COUNT DOWN
; C    L    H          H     L         LLHH     H    H
; C    L    H          H     H         LHHH     H    L        COUNT DOWN
; C    L    H          H     H         HHHH     H    H
; C    L    H          L     H         HHHL     H    L        COUNT DOWN
; C    L    H          L     H         HHLL     H    H
; C    L    H          L     L         HLLL     H    L        COUNT DOWN
; C    L    H          L     L         LLLL     H    H
; C    L    H          H     L         LLLH     H    L        COUNT DOWN
; C    L    H          H     L         LLHH     H    H
; C    L    H          H     H         LHHH     H    L        COUNT DOWN
; C    L    H          H     H         HHHH     H    H
; C    L    H          L     H         HHHL     H    L        COUNT DOWN
; C    L    H          L     H         HHLL     H    H
; C    L    H          L     L         HLLL     H    L        COUNT DOWN
; C    L    H          L     L         LLLL     H    H
; C    L    H          H     L         LLLH     H    L        COUNT DOWN
; C    L    H          H     L         LLHH     H    H
; C    L    H          H     H         LHHH     H    L        COUNT DOWN
; C    L    H          H     H         HHHH     H    H
; C    L    H          L     H         HHHL     H    L        COUNT DOWN
; C    L    H          L     H         HHLL     H    H
; C    L    H          L     L         HLLL     H    L        COUNT DOWN
; C    L    H          L     L         LLLL     H    H
; X    H    X          X     X         ZZZZ     X    X        TEST HI-Z
;-----------------------------------------------------------------------------
```

TB00990M

Shaft Encoder #1

```
;DESCRIPTION:

;THIS PAL16R4 IMPLEMENTS A TWO CHANNEL SHAFT ENCODER OF THE TYPE USED IN SPEED
;CONTROLLERS AND OPTICAL DEVICES.

;BOTH THE "UP" AND "DOWN" OUTPUTS OF THE PAL ARE NORMALLY HIGH.

;WHEN THE SIGNAL AT THE "PHI0" INPUT LEADS THE SIGNAL AT THE "PHI90" INPUT, THE
;"DOWN" OUTPUT ALTERNATES BETWEEN HIGH AND LOW LEVELS AT HALF THE "CLK"
;FREQUENCY RATE.  ALSO, WHEN THE SIGNAL AT THE "PHI0" INPUT LAGS THE SIGNAL AT
;THE "PHI90" INPUT, THE "UP" OUTPUT ALTERNATES BETWEEN HIGH AND LOW LEVELS AT
;HALF THE "CLK" FREQUENCY RATE.

;THE SHAFT ENCODER FEATURES THE CONFIGURATION AND OUTPUT POLARITY TO DRIVE AN
;74S193 TYPE UP/DOWN COUNTER.

;THIS DESIGN WITH GLITCHFREE OUTPUTS WILL BE EXTREMELY USEFUL IN ELECTRICALLY
;NOISY ENVIRONMENTS.  THE PINNING IS GIVEN AS A FIRST PROPOSAL AND CAN BE
;CHANGED ACCORDING TO THE PC-BOARD LAYOUT.
```

TB01000M

Shaft Encoder #2

```
Title     SHAFT ENCODER No. 2
Pattern   P7098
Revision  A
Author    WILLY VOLDAN
Company   MMI GMBH MUNICH
Date      09/09/82

CHIP SHAFT_ENCODER_No._2 PAL16R8

CLK PHI0 PHI90 X4 NC NC NC NC /SSET GND
/OC UD NC S4 S3 S2 S1 NC COUNT VCC

EQUATIONS

/S1     := /PHI0                                     ;CHECK FOR PHI0
         +  SSET                                     ;INITIALIZE S1=L

/S3     := /PHI90                                    ;CHECK FOR PHI90
         +  SSET                                     ;INITIALIZE S3=L

/S2     :=  S1                                       ;CHECK FOR /S1
         +  SSET                                     ;INITIALIZE S2=L

/S4     :=  S3                                       ;CHECK FOR /S3
         +  SSET                                     ;INITIALIZE S4=L

/COUNT :=  S1* S2*/S3* S4                            ;THIS OUTPUT ALTERNATES
         + /S1*/S2* S3*/S4                           ;BETWEEN HIGH AND LOW WITH
         + /S1* S2*/S3*/S4* X4                       ;HALF OR QUARTER THE
         +  S1*/S2* S3* S4* X4                       ;CLK FREQUENCY
         +  S1* S2* S3*/S4
         + /S1*/S2*/S3* S4
         + /S1* S2* S3* S4* X4
         +  S1*/S2*/S3*/S4* X4

/UD     := /S1* S2*/S3* S4                           ;THIS OUTPUT DETERMINES
         + /S1* S2* S3* S4                           ;IF SIGNAL PHI0 LEADS
         + /S1* S2* S3*/S4                           ;OR LAGS SIGNAL PHI90
         +  S1* S2* S3*/S4
         +  S1*/S2* S3*/S4
         +  S1*/S2*/S3*/S4
         +  S1*/S2*/S3* S4
         + /S1*/S2*/S3* S4
```

TB01010M

```
;FUNCTION TABLE FOR PALASM 1

;CLK /OC /SSET PHI0 PHI90 X4 S1 S2 S3 S4 COUNT UD

;---CONTROLS---      --INPUTS--    X     SSSS    -OUTPUTS-
;CLK  /OC  /SSET     PHI0 PHI90    4     1234    COUNT  UD       COMMENTS
;-----------------------------------------------------------------------------
; C    L     L        X     X      X     LLLL      H    H        CLEAR REGISTERS
; C    L     H        L     L      L     LHLH      H    H        COUNT UP X4=L
; C    L     H        H     L      L     HHLH      H    L
; C    L     H        H     L      L     HLLH      L    H
; C    L     H        H     H      L     HLHH      H    L
; C    L     H        H     H      L     HLHL      H    H
; C    L     H        L     H      L     LLHL      H    L
; C    L     H        L     H      L     LHHL      L    H
; C    L     H        L     L      L     LHLL      H    L
; C    L     H        L     L      L     LHLH      H    H
; C    L     H        H     L      L     HHLH      H    L
; C    L     H        H     L      L     HLLH      L    H
; C    L     H        H     H      L     HLHH      H    L
; C    L     H        H     H      L     HLHL      H    H
; C    L     H        L     H      L     LLHL      H    L
; C    L     H        L     H      L     LHHL      L    H
; C    L     H        L     L      H     LHLL      H    L        COUNT UP X4=L
; C    L     H        L     L      H     LHLH      L    H
; C    L     H        H     L      H     HHLH      H    L
; C    L     H        H     L      H     HLLH      L    H
; C    L     H        H     H      H     HLHH      H    L
; C    L     H        H     H      H     HLHL      L    H
; C    L     H        L     H      H     LLHL      H    L
; C    L     H        L     H      H     LHHL      L    H
; C    L     H        L     L      H     LHLL      H    L
; C    L     H        L     L      H     LHLH      L    H
; C    L     H        H     L      H     HHLH      H    L
; C    L     H        H     L      H     HLLH      L    H
; C    L     H        H     H      H     HLHH      H    L
; C    L     H        H     H      H     HLHL      L    H
; C    L     L        X     X      X     LLLL      H    L        CLEAR REGISTERS
; C    L     H        L     L      L     LHLH      H    H        COUNT DOWN X4=L
; C    L     H        L     H      L     LHHH      H    L
; C    L     H        L     H      L     LHHL      H    L
; C    L     H        H     H      L     HHHL      H    L
; C    L     H        H     H      L     HLHL      L    L
; C    L     H        H     L      L     HLLL      H    L
; C    L     H        H     L      L     HLLH      H    L
; C    L     H        L     L      L     LLLH      H    L
; C    L     H        L     L      L     LHLH      L    L
; C    L     H        L     H      L     LHHH      H    L
; C    L     H        L     H      L     LHHL      H    L
; C    L     H        H     H      L     HHHL      H    L
; C    L     H        H     H      L     HLHL      L    L
; C    L     H        H     L      L     HLLL      H    L
; C    L     H        H     L      L     HLLH      H    L
; C    L     H        L     L      H     LLLH      H    L        COUNT DOWN X4=H
; C    L     H        L     L      H     LHLH      L    L
; C    L     H        L     H      H     LHHH      H    L
; C    L     H        L     H      H     LHHL      L    L
; C    L     H        H     H      H     HHHL      H    L
; C    L     H        H     H      H     HLHL      L    L
```

TB01020M

```
; C    L     H      H     L     H     HLLL     H   L
; C    L     H      H     L     H     HLLH     L   L
; C    L     H      L     L     H     LLLH     H   L
; C    L     H      L     L     H     LHLH     L   L
; C    L     H      L     H     H     LHHH     H   L
; C    L     H      L     H     H     LHHL     L   L
; C    L     H      H     H     H     HHHL     H   L
; C    L     H      H     H     H     HLHL     L   L
; X    H     X      X     X     X     ZZZZ     Z   Z      TEST HI-Z
;-------------------------------------------------------------------------

;DESCRIPTION

;THIS PAL16R8 IMPLEMENTS A TWO CHANNEL SHAFT ENCODER OF THE TYPE USED IN SPEED
;CONTROLLERS AND OPTICAL DEVICES.

;THE "COUNT" OUTPUT OF THE PAL IS NORMALLY HIGH.  DURING SHAFT ENCODING THIS
;OUTPUT ALTERNATES BETWEEN HIGH AND LOW.

;INPUT "X4" SELECTS BETWEEN HALF (X4=H) OR QUARTER (X4=L) CLK FREQUENCY OF THE
;"COUNTER" OUTPUT.

;OUTPUT "UD" DETERMINES WHETHER SIGNAL PHI0 LEADS (UD=H) OR LAGS (UD=L) SIGNAL
;PHI90.

;THE SHAFT ENCODER FEATURES THE CONFIGURATION AND OUTPUT POLARITY TO DRIVE AN
;74S697 TYPE UP/DOWN COUNTER.

;THIS DESIGN WITH GLITCHFREE OUTPUTS WILL BE EXTREMELY USEFUL IN ELECTRICALLY
;NOISY ENVIRONMENTS.  THE PINNING IS GIVEN AS A FIRST PROPOSAL AND CAN BE
;CHANGED ACCORDING TO THE PC-BOARD LAYOUT.
```

TB01030M

```
Title     SHAFT ENCODER No. 3 (WITH INTERNAL 4-BIT UP/DOWN COUNTER)
Pattern   P7099
Revision  A
Author    WILLY VOLDAN
Company   MMI GMBH MUNICH
Date      09/09/82

CHIP SHAFT_ENCODER_No._3_(WITH_INTERNAL_4-BIT_UP/DOWN_COUNTER) PAL20X10

EQUATIONS

CLK PHI0 PHI90 X4 /LD NC D3 D2 D1 D0 /SSET GND
/OC DOWN S4 S3 S2 S1 Q3 Q2 Q1 Q0 UP VCC

  /S1 :=  /PHI0                             ;CHECK FOR PHI0
       +   SSET                             ;INITIALIZE S1=L

  /S2 :=  /S1                               ;CHECK FOR S1
       +   SSET                             ;INITIALIZE S2=L

  /S3 :=  /PHI90                            ;CHECK FOR PHI90
       +   SSET                             ;INITIALIZE S3=L

  /S4 :=  /S3                               ;CHECK FOR S3
       +   SSET                             ;INITIALIZE S4=L

/DOWN :=   S1* S2* S3*/S4* PHI0* PHI90* X4  ;PHI0 LEADS PHI90 - COUNT=FREQ/2
       +  /S1*/S2*/S3* S4*/PHI0*/PHI90* X4  ;PHI0 LEADS PHI90 - COUNT=FREQ/2
      :+:  S1*/S2*/S3*/S4* PHI0*/PHI90      ;PHI0 LEADS PHI90 - COUNT=FREQ/4
       +  /S1* S2* S3* S4*/PHI0* PHI90      ;PHI0 LEADS PHI90 - COUNT=FREQ/4

/UP   :=  /S1*/S2* S3*/S4*/PHI0* PHI90      ;PHI90 LEADS PHI0 - COUNT=FREQ/4
       +   S1* S2*/S3* S4* PHI0*/PHI90      ;PHI90 LEADS PHI0 - COUNT=FREQ/4
      :+:  S1*/S2* S3* S4* PHI0* PHI90* X4  ;PHI90 LEADS PHI0 - COUNT=FREQ/2
       +  /S1* S2*/S3*/S4*/PHI0*/PHI90* X4  ;PHI90 LEADS PHI0 - COUNT=FREQ/2

/Q0   :=  /SSET* LD*/D0                      ;LOAD D0 (LSB)
       +  /SSET*/LD*/Q0                      ;HOLD Q0
      :+: /SSET*/LD* UP*/DOWN                ;DECREMENT
       +  /SSET*/LD*/UP* DOWN                ;INCREMENT

/Q1   :=  /SSET* LD*/D1                      ;LOAD D1
       +  /SSET*/LD*/Q1                      ;HOLD Q1
      :+: /SSET*/LD* UP*/DOWN*/Q0            ;DECREMENT
       +  /SSET*/LD*/UP* DOWN* Q0            ;INCREMENT

/Q2   :=  /SSET* LD*/D2                      ;LOAD D2
       +  /SSET*/LD*/Q2                      ;HOLD Q2
      :+: /SSET*/LD* UP*/DOWN*/Q0*/Q1        ;DECREMENT
       +  /SSET*/LD*/UP* DOWN* Q0* Q1        ;INCREMENT

/Q3   :=  /SSET* LD*/D3                      ;LOAD D3 (MSB)
       +  /SSET*/LD*/Q3                      ;HOLD Q3
      :+: /SSET*/LD* UP*/DOWN*/Q0*/Q1*/Q2    ;DECREMENT
       +  /SSET*/LD*/UP* DOWN* Q0* Q1* Q2    ;INCREMENT
```

TB01040M

```
;FUNCTION TABLE FOR PALASM 1

;CLK /OC /SSET /LD X4 PHI0 PHI90 S1 S2 S3 S4 UP DOWN D3 D2 D1 D0 Q3 Q2 Q1 Q0

;----CONTROLS----  INPUT
;    /  /   /       PHI    SSSS            DDDD  QQQQ     COMMENTS
;CLK OC SSET LD X4  0  90  1234  UP DOWN  3210  3210    (Q HEX VALUE)
;----------------------------------------------------------------------------------
; C   L  L  X  X    X X   LLLL  H   H     XXXX  HHHH   INITIALIZE REGISTERS (F)
; C   L  H  L  X    X X   XXXX  X   X     HLHL  HLHL   LOAD (A)
; C   L  H  H  L    L L   LLLL  H   H     XXXX  HLHL   HOLD (A)
; C   L  H  H  L    H L   HLLL  H   H     XXXX  HLHL   HOLD (A) PHI0 LEADS PHI90
; C   L  H  H  L    H L   HHLL  H   L     XXXX  HLHL   HOLD (A) X4=H - FREQ/4
; C   L  H  H  L    H H   HHHL  H   H     XXXX  HLLH   DECREMENT (9)
; C   L  H  H  L    H H   HHHH  H   H     XXXX  HLLH   HOLD (9)
; C   L  H  H  L    L H   LHHH  H   H     XXXX  HLLH   HOLD (9)
; C   L  H  H  L    L H   LLHH  H   L     XXXX  HLLH   HOLD (9)
; C   L  H  H  L    L L   LLLH  H   H     XXXX  HLLL   DECREMENT (8)
; C   L  H  H  L    L L   LLLL  H   H     XXXX  HLLL   HOLD (8)
; C   L  H  H  L    H L   HLLL  H   H     XXXX  HLLL   HOLD (8)
; C   L  H  H  L    H L   HHLL  H   L     XXXX  HLLL   HOLD (8)
; C   L  H  H  L    H H   HHHL  H   H     XXXX  LHHH   DECREMENT (7)
; C   L  H  H  L    H H   HHHH  H   H     XXXX  LHHH   HOLD (7)
; C   L  H  H  L    L H   LHHH  H   H     XXXX  LHHH   HOLD (7)
; C   L  H  H  L    L H   LLHH  H   L     XXXX  LHHH   HOLD (7)
; C   L  H  H  H    L L   LLLH  H   H     XXXX  LHHL   DECREMENT (6)
; C   L  H  H  H    L L   LLLL  H   L     XXXX  LHHL   HOLD (6) X4=H - FREQ/2
; C   L  H  H  H    H L   HLLL  H   H     XXXX  LHLH   DECREMENT (5)
; C   L  H  H  H    H L   HHLL  H   L     XXXX  LHLH   HOLD (5)
; C   L  H  H  H    H H   HHHL  H   H     XXXX  LHLL   DECREMENT (4)
; C   L  H  H  H    H H   HHHH  H   L     XXXX  LHLL   HOLD (4)
; C   L  H  H  H    L H   LHHH  H   H     XXXX  LLHH   DECREMENT (3)
; C   L  H  H  H    L H   LLHH  H   L     XXXX  LLHH   HOLD (3)
; C   L  H  H  H    L L   LLLH  H   H     XXXX  LLHL   DECREMENT (2)
; C   L  H  H  H    L L   LLLL  H   L     XXXX  LLHL   HOLD (2)
; C   L  H  H  H    H L   HLLL  H   H     XXXX  LLLH   DECREMENT (1)
; C   L  H  H  H    H L   HHLL  H   L     XXXX  LLLH   HOLD (1)
; C   L  H  H  H    H H   HHHL  H   H     XXXX  LLLL   DECREMENT (0)
; C   L  H  H  H    H H   HHHH  H   L     XXXX  LLLL   HOLD (0)
; C   L  H  H  H    L H   LHHH  H   H     XXXX  HHHH   DECREMENT (F)(ROLL UNDER)
; C   L  H  H  H    L H   LLHH  H   L     XXXX  HHHH   HOLD (F)
; C   L  H  H  H    L L   LLLH  H   H     XXXX  HHHL   DECREMENT (E)
; C   L  H  L  X    X X   XXXX  X   X     LHLH  LHLH   LOAD (5)
; C   L  H  H  L    L L   LLLL  H   H     XXXX  LHLH   HOLD (5)
; C   L  H  H  L    L H   LLHL  H   H     XXXX  LHLH   HOLD (5) PHI90 LEADS PHI0
; C   L  H  H  L    L H   LLHH  L   H     XXXX  LHLH   HOLD (5) X4=L - FREQ/4
; C   L  H  H  L    H H   HLHH  H   H     XXXX  LHHL   INCREMENT (6)
; C   L  H  H  L    H H   HHHH  H   H     XXXX  LHHL   HOLD (6)
; C   L  H  H  L    H L   HHLH  H   H     XXXX  LHHL   HOLD (6)
; C   L  H  H  L    H L   HHLL  L   H     XXXX  LHHL   HOLD (6)
; C   L  H  H  L    L L   LHLL  H   H     XXXX  LHHH   INCREMENT (7)
; C   L  H  H  L    L L   LLLL  H   H     XXXX  LHHH   HOLD (7)
; C   L  H  H  L    L H   LLHL  H   H     XXXX  LHHH   HOLD (7)
; C   L  H  H  L    L H   LLHH  L   H     XXXX  LHHH   HOLD (7)
; C   L  H  H  L    H H   HLHH  H   H     XXXX  HLLL   INCREMENT (8)
; C   L  H  H  L    H H   HHHH  H   H     XXXX  HLLL   HOLD (8)
; C   L  H  H  L    H L   HHLH  H   H     XXXX  HLLL   HOLD (8)
; C   L  H  H  L    H L   HHLL  L   H     XXXX  HLLL   HOLD (8)
; C   L  H  H  H    L L   LHLL  H   H     XXXX  HLLH   INCREMENT (9)
```

TB01050M

```
; C   L  H  H  H    L L  LLLL  L   H    XXXX  HLLH   HOLD (9)  X4=H - FREQ/2
; C   L  H  H  H    L H  LLHL  H   H    XXXX  HLHL   INCREMENT (A)
; C   L  H  H  H    L H  LLHH  L   H    XXXX  HLHL   HOLD (A)
; C   L  H  H  H    H H  HLHH  H   H    XXXX  HLHH   INCREMENT (B)
; C   L  H  H  H    H H  HHHH  L   H    XXXX  HLHH   HOLD (B)
; C   L  H  H  H    H L  HHLH  H   H    XXXX  HHLL   INCREMENT (C)
; C   L  H  H  H    H L  HHLL  L   H    XXXX  HHLL   HOLD (C)
; C   L  H  H  H    L L  LHLL  H   H    XXXX  HHLH   INCREMENT (D)
; C   L  H  H  H    L L  LLLL  L   H    XXXX  HHLH   HOLD (D)
; C   L  H  H  H    L H  LLHL  H   H    XXXX  HHHL   INCREMENT (E)
; C   L  H  H  H    L H  LLHH  L   H    XXXX  HHHL   HOLD (E)
; C   L  H  H  H    H H  HLHH  H   H    XXXX  HHHH   INCREMENT (F)
; C   L  H  H  H    H H  HHHH  L   H    XXXX  HHHH   HOLD (F)
; C   L  H  H  H    H L  HHLH  H   H    XXXX  LLLL   INCREMENT (0) (ROLL OVER)
; C   L  H  H  H    H L  HHLL  L   H    XXXX  LLLL   HOLD (0)
; C   L  H  H  H    L L  LHLL  H   H    XXXX  LLLH   INCREMENT (1)
; C   L  H  L  X    X X  XXXX  X   X    LLLH  LLLH   LOAD (1)
; C   L  H  L  X    X X  XXXX  X   X    LLHH  LLHH   LOAD (3)
; C   L  H  L  X    X X  XXXX  X   X    LHLH  LHLH   LOAD (5)
; C   L  H  L  X    X X  XXXX  X   X    LHHH  LHHH   LOAD (7)
; C   L  H  L  X    X X  XXXX  X   X    HLLH  HLLH   LOAD (9)
; C   L  H  L  X    X X  XXXX  X   X    HLHH  HLHH   LOAD (B)
; C   L  H  L  X    X X  XXXX  X   X    HHLH  HHLH   LOAD (D)
; C   L  H  L  X    X X  XXXX  X   X    HHHH  HHHH   LOAD (F)
; X   H  X  X  X    X X  ZZZZ  Z   Z    XXXX  ZZZZ   TEST HI-Z
;-----------------------------------------------------------------------------

;DESCRIPTION

;THIS PAL20X10 IMPLEMENTS A TWO CHANNEL SHAFT ENCODER WITH AN INTERNAL 4-BIT
;UP/DOWN COUNTER.

;BOTH THE "UP" AND "DOWN" OUTPUTS OF THE PAL ARE NORMALLY AT HIGH.

;WHEN THE SIGNAL AT THE "PHI0" INPUT LEADS THE SIGNAL AT THE "PHI90" INPUT, THE
;"DOWN" OUTPUT ALTERNATES BETWEEN HIGH AND LOW LEVELS AND THE COUNTER WILL COUNT
;DOWN.  WHEN THE SIGNAL AT THE "PHI0" INPUT LEADS THE SIGNAL AT THE "PHI90"
;INPUT, THE "UP" OUTPUT ALTERNATES BETWEEN HIGH AND LOW LEVELS AND THE COUNTER
;WILL COUNT UP.

;INPUT "X4" SELECTS BETWEEN HALF (X4=H) OR QUARTER (X4=L) CLK FREQUENCY OF THE
;COUNTER OUTPUTS.

;THE INTERNAL 4-BIT SYNCHRONOUS COUNTER HAS COUNT UP, COUNT DOWN CAPABILITIES.
;ALSO, THE COUNTER CAN PARALLEL LOAD AND HOLD DATA INDEPENDENTLY OF THE SHAFT
;ENCODER SECTION.  THE REGISTERS ARE SYNCHRONOUSLY INITIALIZED WHEN /SSET IS HEL
;LOW.

;THE CONTROL INPUTS PROVIDE THESE OPERATIONS WHICH OCCUR SYNCHRONOUSLY AT THE
;RISING EDGE OF THE CLOCK.
```

TB01060M

PAL Device Design Specification

```
Title      4to16 Decoder
Pattern    4-16DEC.PDS
Revision   A
Author     Mehrnaz Hada
Company    Monolithic Memories, Santa Clara, CA
Date       1/9/85

CHIP Decoder PAL6L16

Q0 Q1 Q2 A B C D EN1 EN2 Q3 Q4 GND Q5
Q6 Q7 Q8 Q9 Q10 Q11 Q12 Q13 Q14 Q15 VCC

EQUATIONS

/Q0  = /D*/C*/B*/A* EN1* EN2            ;Decode 0000
/Q1  = /D*/C*/B* A* EN1* EN2            ;Decode 0001
/Q2  = /D*/C* B*/A* EN1* EN2            ;Decode 0010
/Q3  = /D*/C* B* A* EN1* EN2            ;Decode 0011
/Q4  = /D* C*/B*/A* EN1* EN2            ;Decode 0100
/Q5  = /D* C*/B* A* EN1* EN2            ;Decode 0101
/Q6  = /D* C* B*/A* EN1* EN2            ;Decode 0110
/Q7  = /D* C* B* A* EN1* EN2            ;Decode 0111
/Q8  =  D*/C*/B*/A* EN1* EN2            ;Decode 1000
/Q9  =  D*/C*/B* A* EN1* EN2            ;Decode 1001
/Q10 =  D*/C* B*/A* EN1* EN2            ;Decode 1010
/Q11 =  D*/C* B* A* EN1* EN2            ;Decode 1011
/Q12 =  D* C*/B*/A* EN1* EN2            ;Decode 1100
/Q13 =  D* C*/B* A* EN1* EN2            ;Decode 1101
/Q14 =  D* C* B*/A* EN1* EN2            ;Decode 1110
/Q15 =  D* C* B* A* EN1* EN2            ;Decode 1111


SIMULATION

TRACE_ON  D B C A Q0 Q1 Q2 Q3 Q4 Q5 Q6 Q7 Q8 Q9 Q10
          Q11 Q12 Q13 Q14 Q15

SETF /D /C /B /A EN1 EN2
SETF A
SETF B
SETF C
SETF D
SETF /D
SETF /C
SETF /B
SETF /A
SETF /EN1                                   ;Set outputs to high
SETF EN1 /EN2                               ;Set outputs to high
SETF /EN1                                   ;Set outputs to high

;The 4 to 16 decoder, decodes four binary decoded inputs
;into one of 16 mutually exclusive outputs, whenever the
;two enable lines EN1 and EN2 are high. When one or both
;of the enable lines are low the outputs are all set to
;high values.
```

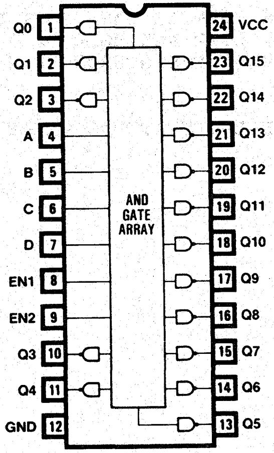
Simulation Results

```
Page :  1
     gggggggggg gg
 D   LLLLHLLLLL LL
 B   LLHHHHHLLL LL
 C   LLLHHHLLLL LL
 A   LHHHHHHHLL LL
 Q0  LHHHHHHHLH HH
 Q1  HLHHHHHLHH HH
 Q2  HHHHHHHHHH HH
 Q3  HHLHHHLHHH HH
 Q4  HHHHHHHHHH HH
 Q5  HHHHHHHHHH HH
 Q6  HHHHHHHHHH HH
 Q7  HHHLHLHHHH HH
 Q8  HHHHHHHHHH HH
 Q9  HHHHHHHHHH HH
 Q10 HHHHHHHHHH HH
 Q11 HHHHHHHHHH HH
 Q12 HHHHHHHHHH HH
 Q13 HHHHHHHHHH HH
 Q14 HHHHHHHHHH HH
 Q15 HHHHLHHHHH HH
```

XPLOT Output

```
Title    :     4to16 Decoder      Author  :    Mehrnaz Hada
Pattern  :   4-16DEC.PDS          Company :  Monolithic Memories,
Revision :  A                     Date    :      1/9/85

PAL6L16
DECODER

                     11
     0123 4567 8901

  0 -X-X -X-X X-X-
  1 X-X- X-X- X-X-
  2 X--X -X-X X-X-
  3 -XX- -X-X X-X-
  4 -XX- X-X- X-X-
  5 X--X X-X- X-X-
  6 -X-X X-X- X-X-
  7 X-X- -XX- X-X-
  8 -XX- -XX- X-X-
  9 X--X -XX- X-X-
 10 -X-X -XX- X-X-
 11 X-X- -X-X X-X-
 12 X-X- X--X X-X-
 13 -XX- X--X X-X-
 14 -X-X X--X X-X-
 15 X--X X--X X-X-


 TOTAL FUSES BLOWN:     96
```

Logic Symbol



TB01690M

PAL Device Design Specification

```
Title    PC I/O Mapper
Pattern  MemIO.pds
Revision A
Author   A G Gilbert
Company  Monolithic Memories Inc., Santa Clara,CA
Date     1/8/85

;Personal computers which are hardware compatible with the
;ubiquitous IBM PC share this I/O map.

CHIP PC_IO PAL8L14

NC NC A9 A8 A7 A6 A5 A4 A3 AEN /CSMONOCHRMAD GND
/CSGAMEIOAD /CSCOLORAD /CSPRINTERAD /CS5FLOPPYAD /CSRS232AD /CSNMIMKRG
/CSDMAPGRG /CSPPICHIP /CSTIMERCHIP /CSINTCCHIP /CSDMACCHIP VCC

Equations

CSDMACCHIP = /A9*/A8*/A7*/A6*/A5       ;DMA controller
           * /A4*/AEN                  ;Chip select
                                       ;HEX address 000-00F

CSINTCCHIP = /A9*/A8*/A7*/A6*A5        ;Interupt controller
           * /A4*/A3*/AEN              ;Chip select
                                       ;HEX address 020-021

CSTIMERCHIP = /A9*/A8*/A7*A6*/A5       ;Timer
            * /A4*/A3*/AEN             ;Chip select
                                       ;HEX address 040-043

CSPPICHIP = /A9*/A8*/A7*A6*A5          ;Parallel peripheral interface
          * /A4*/A3*/AEN               ;Chip select
                                       ;HEX address 060-063

CSDMAPGRG = /A9*/A8*A7*/A6*/A5         ;DMA page register
          * /A4*/A3*/AEN               ;Chip select
                                       ;HEX address 080-083

CSNMIMKRG = /A9*/A8*A7*/A6*A5          ;NMI mask register
          * /A4*/AEN                   ;Chip select
                                       ;HEX address 0AX

CSRS232AD = A9*A8*A7*A6*A5             ;RS 232 module
          * A4*A3*/AEN                 ;Device select
                                       ;HEX address 3F8-3FF

CS5FLOPPYAD = A9*A8*A7*A6*A5           ;5.25 floppy disk module
            * A4*/A3*/AEN              ;Device select
                                       ;HEX address 3F0-3F7

CSPRINTERAD = A9*/A8*/A7*A6*A5         ;Parallel printer module
            * A4*A3*/AEN               ;Device select
                                       ;HEX address 378-37F

CSCOLORAD = A9*A8*A7*A6*/A5            ;Color graphics video module
          * A4*/AEN                    ;Device select
                                       ;HEX address 3D0-3DF

CSGAMEIOAD = A9*/A8*/A7*/A6*/A5        ;Game I/O module
           * /A4*/AEN                  ;Device select
                                       ;HEX address 200-20F

CSMONOCHRMAD = A9*A8*A7*/A6*A5         ;Monochrome video module
             * A4*/AEN                 ;Device select
                                       ;HEX address 3B0-3BF


SIMULATION

TRACE_ON  A9 A8 A7 A6 A5 A4 A3 AEN /CSMONOCHRMAD
         /CSGAMEIOAD /CSCOLORAD /CSPRINTERAD /CS5FLOPPYAD
         /CSRS232AD /CSNMIMKRG /CSDMAPGRG /CSPPICHIP
         /CSTIMERCHIP /CSINTCCHIP /CSDMACCHIP
SETF AEN
SETF /A9 /A8 /A7 /A6 /A5 /A4 /A3 /AEN
SETF  A5
SETF  A6
SETF /A5  A7
SETF  A5
SETF  A4 /A6
SETF  A9  A8  A7  A6  A5  A4  A3
SETF /A3
SETF /A5
SETF /A4 /A6 /A7 /A8
SETF  A9  A8  A7 /A6  A5  A4
```

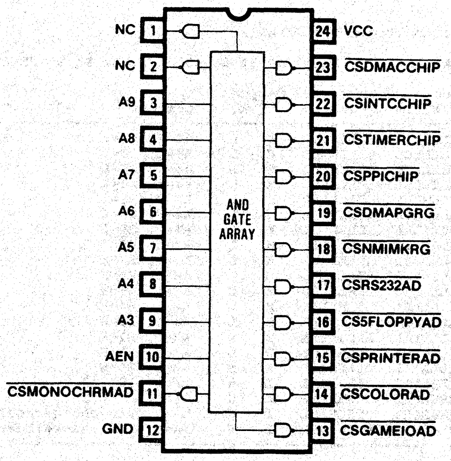
Simulation Results

```
  Page :  1
                      gggggggggg gg
  A9                  XLLLLLLHHH HH
  A8                  XLLLLLLHHH LH
  A7                  XLLLHHHHHH LH
  A6                  XLLHHHLHHH LL
  A5                  XLHHLHHHHL LH
  A4                  XLLLLLHHHH LH
  A3                  XLLLLLLHLL LL
  AEN                 HLLLLLLLLL LL
 /CSMONOCHRMAD        HHHHHHHHHH HL
 /CSGAMEIOAD          HHHHHHHHHH LH
 /CSCOLORAD           HHHHHHHHHL HH
 /CSPRINTERAD         HHHHHHHHHH HH
 /CS5FLOPPYAD         HHHHHHHHLH HH
 /CSRS232AD           HHHHHHHLHH HH
 /CSNMIMKRG           HHHHHHHHHH HH
 /CSDMAPGRG           HHHHHHHHHH HH
 /CSPPICHIP           HHHLHHHHHH HH
 /CSTIMERCHIP         HHHHHHHHHH HH
 /CSINTCCHIP          HHLHHHHHHH HH
 /CSDMACCHIP          HLHHHHHHHH HH
```

XPLOT Output

```
Title     : PC I/O Mapper
Pattern   : MemIO.pds
Revision  : A
Author    : A G Gilbert
Company   : Monolithic Memories Inc
Date      : 1/8/85


PAL8L14
PC_IO

               11 1111
   0123 4567 8901 2345

 0 XXXX XXXX XXXX XXXX
 1 -X-X -X-X -X-X ---X
 2 XXXX XXXX XXXX XXXX
 3 -X-X -X-X X--X -X-X
 4 -X-X -XX- -X-X -X-X
 5 -X-X -XX- X--X -X-X
 6 -X-X X--X -X-X -X-X
 7 -X-X X--X X--X ---X
 8 X-X- X-X- X-X- X--X
 9 X-X- X-X- X-X- -X-X
10 X--X -XX- X-X- X--X
11 X-X- X-X- -XX- ---X
12 X-X- X--X X-X- ---X
13 X--X -X-X -X-X ---X


TOTAL FUSES BLOWN:   101
```

Logic Symbol



TB01700M

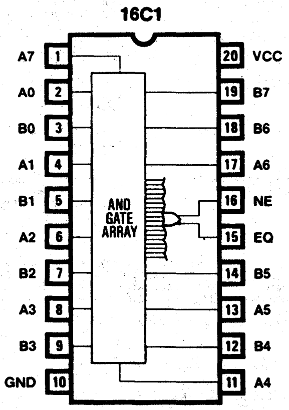
PAL Device Design Specification

```
Title    Octal_Comparator
Pattern  OctComp.pds
Revision A
Author   Mehrnaz Hada
Company  Monolithic Memories Inc., Santa Clara,CA
Date     1/29/85

;The octal comparator establishes when two 8-bit data
;strings (A7-A0) and (B7-B0) are equivalent (EQ=H) or
;equivalent (NE=H).

CHIP OctalComparato PAL16C1

A7 A0 B0 A1 B1 A2 B2 A3 B3 GND
A4 B4 A5 B5 EQ NE A6 B6 B7 VCC

EQUATIONS

NE =  A0*/B0  +  /A0* B0                      ;A0 :+: B0
   +  A1*/B1  +  /A1* B1                      ;A1 :+: B1
   +  A2*/B2  +  /A2* B2                      ;A2 :+: B2
   +  A3*/B3  +  /A3* B3                      ;A3 :+: B3
   +  A4*/B4  +  /A4* B4                      ;A4 :+: B4
   +  A5*/B5  +  /A5* B5                      ;A5 :+: B5
   +  A6*/B6  +  /A6* B6                      ;A6 :+: B6
   +  A7*/B7  +  /A7* B7                      ;A7 :+: B7


SIMULATION

TRACE_ON A7 A6 A5 A4 A3 A2 A1 A0 NE
         B7 B6 B5 B4 B3 B2 B1 B0

SETF  A7 /A6 /A5 /A4 /A3 /A2 /A1 /A0         ;A7=H, B7=L
     /B7 /B6 /B5 /B4 /B3 /B2 /B1 /B0
SETF /A7 A6                                  ;A6=H, B6=L
SETF /A6 A5                                  ;A5=H, B5=L
SETF /A5 A4                                  ;A4=H, B4=L
SETF /A4 A3                                  ;A3=H, B3=L
SETF /A3 A2                                  ;A2=H, B2=L
SETF /A2 A1                                  ;A1=H, B1=L
SETF /A1 A0                                  ;A0=H, B0=L
SETF /A7 /A6 /A5 /A4 /A3 /A2 /A1 /A0         ;A7=L, B7=H
      B7
SETF /B7 B6                                  ;A6=L, B6=H
SETF /B6 B5                                  ;A5=L, B5=L
SETF /B5 B4                                  ;A4=L, B4=H
SETF /B4 B3                                  ;A3=L, B3=H
SETF /B3 B2                                  ;A2=L, B2=H
SETF /B2 B1                                  ;A1=L, B1=H
SETF /B1 B0                                  ;A0=L, B0=H
SETF /B0                                     ;Test all L's
SETF A7 A6 A5 A4 A3 A2 A1 A0                 ;Test all H's
      B7 B6 B5 B4 B3 B2 B1 B0
SETF /A7 A6 /A5 A4 /A3 A2 /A1 A0             ;Test even ones
      /B7 B6 /B5 B4 /B3 B2 /B1 B0
SETF A7 /A6 A5 /A4 A3 /A2 A1 /A0             ;Test odd ones
      B7 /B6 B5 /B4 B3 /B2 B1 /B0


;Function Table for PALASM1

;A7 A6 A5 A4 A3 A2 A1 A0 B7 B6 B5 B4 B3 B2 B1 B0 NE EQ

; Input  A  Input  B  Outputs
; 76543210  76543210   NE EQ    Comments
;-----------------------------------------------
; HLLLLLLL  LLLLLLLL   H   L    A7=H, B7=L
; LHLLLLLL  LLLLLLLL   H   L    A6=H, B6=L
; LLHLLLLL  LLLLLLLL   H   L    A5=H, B5=L
; LLLHLLLL  LLLLLLLL   H   L    A4=H, A5=L
; LLLLHLLL  LLLLLLLL   H   L    A3=H, B3=L
; LLLLLHLL  LLLLLLLL   H   L    A2=H, B2=L
; LLLLLLHL  LLLLLLLL   H   L    A1=H, B1=L
; LLLLLLLH  LLLLLLLL   H   L    A0=H, B0=L
; LLLLLLLL  HLLLLLLL   H   L    A7=L, B7=H
; LLLLLLLL  LHLLLLLL   H   L    A6=L, B6=H
; LLLLLLLL  LLHLLLLL   H   L    A5=L, B5=H
; LLLLLLLL  LLLHLLLL   H   L    A4=L, B4=H
; LLLLLLLL  LLLLHLLL   H   L    A3=L, B3=H
; LLLLLLLL  LLLLLHLL   H   L    A2=L, B2=H
; LLLLLLLL  LLLLLLHL   H   L    A1=L, B1=H
; LLLLLLLL  LLLLLLLH   H   L    A0=L, B0=H
; LLLLLLLL  LLLLLLLL   L   H    Test all L'S
; HHHHHHHH  HHHHHHHH   L   H    Test all H'S
; HLHLHLHL  HLHLHLHL   L   H    Test even checkerboard
; LHLHLHLH  LHLHLHLH   L   H    Test odd  checkerboard
;-----------------------------------------------------
```

Simulation Results

```
Page :  1
    gggggggggg gggggggggg
 A7 HLLLLLLLLL LLLLLLLHLH
 A6 LHLLLLLLLL LLLLLLLHHL
 A5 LLHLLLLLLL LLLLLLLHLH
 A4 LLLHLLLLLL LLLLLLLHHL
 A3 LLLLHLLLLL LLLLLLLHLH
 A2 LLLLLHLLLL LLLLLLLHHL
 A1 LLLLLLHLLL LLLLLLLHLH
 A0 LLLLLLLHLL LLLLLLLHHL
 NE HHHHHHHHHH HHHHHHLLLL
 B7 LLLLLLLLHL LLLLLLLHLH
 B6 LLLLLLLLLH LLLLLLLHHL
 B5 LLLLLLLLLL HLLLLLLHLH
 B4 LLLLLLLLLL LHLLLLLHHL
 B3 LLLLLLLLLL LLHLLLLHLH
 B2 LLLLLLLLLL LLLHLLLHHL
 B1 LLLLLLLLLL LLLLHLLHLH
 B0 LLLLLLLLLL LLLLLHLHHL
```

Logic Symbol



TB01710M

PAL Device Design Specification

```
Title    3to8_Dmux
Pattern  3to8Dmux.pds
Revision A
Author   Mehrnaz Hada
Company  Monolithic Memories Inc., Santa Clara, CA
Date     1/29/85

;The 3-to-8 demultiplexer with control storage provides a
;conventional 8-bit demux function combined with control
;storage functions:load true, load complement, hold, toggle,
;polarity, clear and preset. Five inputs(/LD,/CLR,/PR,POL,
;TOG) select one of six operations. The six operations are
;summarized in the following operations table:

;Control   Functions   Polarity Inputs Outputs
;/OC CLK /CLR /PR /LD  POL  TOG  ABC    Q7-Q0  Operation
;-------------------------------------------------------------
; H   X    X    X   X    X    X    X       Z   HI-Z
; L   C    L    X   X    X    X    X       L   Clear
; L   C    H    L   X    X    X    X       H   PRESET
; L   C    H    H   L    H    X    I      MUX  Load true
; L   C    H    H   L    L    X    I     /MUX  Load COMP
; L   C    H    H   H    X    L    X       Q   Hold
; L   C    H    H   H    X    H    X      /Q   Tog polarity
;-------------------------------------------------------------

CHIP 3to8Dmux PAL16R8

CLK /CLR /PR A  B  C  /LD POL TOG GND
/OC  Q7   Q6 Q5 Q4 Q3  Q2 Q1  Q0  VCC

EQUATIONS

/Q0 :=  CLR                                   ;Clear Q0
     + /PR* LD*/POL*/C*/B*/A                  ;Decode 000
     + /PR* LD* POL*       A                  ;Load true
     + /PR* LD* POL*    B                     ;Load true
     + /PR* LD* POL* C                        ;Load true
     + /PR*/LD*/TOG*/Q0                       ;Hold
     + /PR*/LD* TOG* Q0                       ;Toggle polarity

/Q1 :=  CLR                                   ;Clear Q1
     + /PR* LD*/POL*/C*/B* A                  ;Decode 001
     + /PR* LD* POL*      /A                  ;Load true
     + /PR* LD* POL*    B                     ;Load true
     + /PR* LD* POL* C                        ;Load true
     + /PR*/LD*/TOG*/Q1                       ;Hold
     + /PR*/LD* TOG* Q1                       ;Toggle polarity

/Q2 :=  CLR                                   ;Clear Q2
     + /PR* LD*/POL*/C* B*/A                  ;Decode 010
     + /PR* LD* POL*       A                  ;Load true
     + /PR* LD* POL*   /B                     ;Load true
     + /PR* LD* POL* C                        ;Load true
     + /PR*/LD*/TOG*/Q2                       ;Hold
     + /PR*/LD* TOG* Q2                       ;Toggle polarity

/Q3 :=  CLR                                   ;Clear Q3
     + /PR* LD*/POL*/C* B* A                  ;Decode 011
     + /PR* LD* POL*      /A                  ;Load true
     + /PR* LD* POL*   /B                     ;Load true
     + /PR* LD* POL* C                        ;Load true
     + /PR*/LD*/TOG*/Q3                       ;Hold
     + /PR*/LD* TOG* Q3                       ;Toggle polarity

/Q4 :=  CLR                                   ;Clear Q4
     + /PR* LD*/POL* C*/B*/A                  ;Decode 100
     + /PR* LD* POL*       A                  ;Load true
     + /PR* LD* POL*    B                     ;Load true
     + /PR* LD* POL*/C                        ;Load true
     + /PR*/LD*/TOG*/Q4                       ;Hold
     + /PR*/LD* TOG* Q4                       ;Toggle polarity

/Q5 :=  CLR                                   ;Clear Q5
     + /PR* LD*/POL* C*/B* A                  ;Decode 101
     + /PR* LD* POL*      /A                  ;Load true
     + /PR* LD* POL*    B                     ;Load true
     + /PR* LD* POL*/C                        ;Load true
     + /PR*/LD*/TOG*/Q5                       ;Hold
     + /PR*/LD* TOG* Q5                       ;Toggle polarity

/Q6 :=  CLR                                   ;Clear Q6
     + /PR* LD*/POL* C* B*/A                  ;Decode 110
     + /PR* LD* POL*       A                  ;Load true
     + /PR* LD* POL*   /B                     ;Load true
     + /PR* LD* POL*/C                        ;Load true
     + /PR*/LD*/TOG*/Q6                       ;Hold
     + /PR*/LD* TOG* Q6                       ;Toggle polarity

/Q7 :=  CLR                                   ;Clear Q7
     + /PR* LD*/POL* C* B* A                  ;Decode 111
     + /PR* LD* POL*      /A                  ;Load true
     + /PR* LD* POL*   /B                     ;Load true
     + /PR* LD* POL*/C                        ;Load true
     + /PR*/LD*/TOG*/Q7                       ;Hold
     + /PR*/LD* TOG* Q7                       ;Toggle polarity

 SIMULATION

TRACE_ON /OC /CLR /PR /LD POL TOG C B A
         Q7 Q6 Q5 Q4 Q3 Q2 Q1 Q0

SETF OC CLR PR LD POL /TOG                    ;Clear
CLOCKF CLK
SETF /CLR                                     ;Preset
CLOCKF CLK
SETF /PR /C /B /A                             ;Load 0
CLOCKF CLK
SETF A                                        ;Load 1
CLOCKF CLK
SETF B /A                                     ;Load 2
CLOCKF CLK
SETF A                                        ;Load 3
CLOCKF CLK
SETF /LD                                      ;Hold
SETF TOG                                      ;Toggle polarity
CLOCKF CLK
CLOCKF CLK                                    ;Toggle polarity

SETF /POL LD /C /B /A                         ;Load 0 complement
CLOCKF CLK
SETF A                                        ;Load 1 complement
CLOCKF CLK
SETF /OC                                      ;Test HI-Z
CLOCKF CLK

;Function Table for PALASM1

;/OC CLK /CLR /PR /LD POL TOG C B A Q7 Q6 Q5 Q4 Q3 Q2 Q1 Q0

;Control   Functions   Polarity  Input  Output
;/OC CLK /CLR /PR /LD  POL  TOG   CBA  Q7----Q0  Comments
;-------------------------------------------------------------
; L   C    L    L   L    H    L   XXX  LLLLLLLL  Clear
; L   C    H    L   L    H    L   XXX  HHHHHHHH  PRESET
; L   C    H    H   L    H    X   LLL  LLLLLLLH  Load 0
; L   C    H    H   L    H    X   LLH  LLLLLLHL  Load 1
; L   C    H    H   L    H    X   LHL  LLLLLHLL  Load 2
; L   C    H    H   L    H    X   LHH  LLLLHLLL  Load 3
; L   C    H    H   L    H    X   HLL  LLLHLLLL  Load 4
; L   C    H    H   L    H    X   HLH  LLHLLLLL  Load 5
; L   C    H    H   L    H    X   HHL  LHLLLLLL  Load 6
; L   C    H    H   L    H    X   HHH  HLLLLLLL  Load 7
; L   C    H    H   H    X    L   XXX  HLLLLLLL  Hold 7
; L   C    H    H   H    X    H   XXX  LHHHHHHH  Hold
; L   C    H    H   H    X    H   XXX  HLLLLLLL  Hold
; L   C    H    H   L    L    X   LLL  HHHHHHHL  Load 0
; L   C    H    H   L    L    X   LLH  HHHHHHLH  Load 1
; L   C    H    H   L    L    X   LHL  HHHHHLHH  Load 2
; L   C    H    H   L    L    X   LHH  HHHHLHHH  Load 3
; L   C    H    H   L    L    X   HLL  HHHLHHHH  Load 4
; L   C    H    H   L    L    X   HLH  HHLHHHHH  Load 5
; L   C    H    H   L    L    X   HHL  HLHHHHHH  Load 6
; L   C    H    H   L    L    X   HHH  LHHHHHHH  Load 7
; L   C    H    H   H    X    L   XXX  LHHHHHHH  Hold 7
; L   C    H    H   H    X    H   XXX  HLLLLLLL  Hold
; L   C    H    H   H    X    H   XXX  LHHHHHHH  Hold
; H   X    X    X   X    X    X   XXX  ZZZZZZZZ  Test HI-Z
;-------------------------------------------------------------
```

Simulation Results

```
Page :  1
     g  cgcgcgc gcgcgc cgc gcg    c
/OC  LLLLLLLLLL LLLLLLLLLL LLHHHHH
/CLR LLLLHHHHHH HHHHHHHHHH HHHHHHH
/PR  LLLLLLHHHH HHHHHHHHHH HHHHHHH
/LD  LLLLLLLLLL LLLLHHHHLL LLLLLLL
 POL HHHHHHHHHH HHHHHHHHLL LLLLLLL
 TOG LLLLLLLLLL LLLLHHHHHH HHHHHHH
 C   XXXXXXLLLL LLLLLLLLLL LLLLLLL
 B   XXXXXXLLLL HHHHHHHHLL LLLLLLL
 A   XXXXXXLLHH LLHHHHHHLL HHHHHHH
 Q7  XXXLLHHLLL LLLLLHHLLH HHHZZZZ
 Q6  XXXLLHHLLL LLLLLHHLLH HHHZZZZ
 Q5  XXXLLHHLLL LLLLLHHLLH HHHZZZZ
 Q4  XXXLLHHLLL LLLLLHHLLH HHHZZZZ
 Q3  XXXLLHHLLL LLLHHLLHHH HHHZZZZ
 Q2  XXXLLHHLLL LHHLLHHLLH HHHZZZZ
 Q1  XXXLLHHLLH HLLLLHHLLH HLLZZZZ
 Q0  XXXLLHHHHL LLLLLHHLLL LHHZZZZ
```

TB01720M

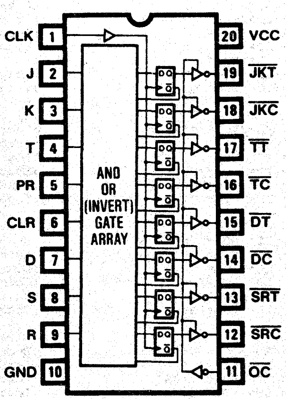
PAL Device Design Specification

```
Title    Basic Flip Flops
Pattern  FlipFlop.pds
Revision A
Author   Vincent Coli
Company  Monolithic Memories Inc., Santa Clara, CA
Date     2/28/85

CHIP FlipFlop PAL16RP8

CLK J K T PR CLR D S R GND
/OC /SRC /SRT /DC /DT /TC /TT /JKC /JKT VCC

EQUATIONS

JKT :=  J*/JKT*/CLR                          ;JK Flio-Flop
      + /K* JKT*/CLR                         ;(JKC = /Q)
      +  PR                                  ;Preset Q

JKC := /J* K  */PR                           ;JK Flip-Flop
      + /J*/JKT*/PR                          ;(JK = /Q)
      +  K* JKT*/PR
      +  CLR                                 ;Clear /Q

TT   :=  T*/TT*/CLR                          ;T Flip-Flop
      + /T* TT*/CLR                          ;(TT = Q)
      +  PR                                  ;Preset Q

TC   := /T*/TT*/CLR                          ;T Flip-Flop
      +  T* TT*/PR                           ;(TC = /Q)
      +  CLR                                 ;Clear /Q

DT   :=  D*/CLR                              ;D Flip-Flop
      +  PR                                  ;Preset Q

DC   := /D*/PR                               ;D Flip-Flop
      +  CLR                                 ;Clear /Q

SRT :=  S*     /CLR                          ;Set-Reset Flip-Flop
      + /R* SRT*/CLR                         ;(SRT = Q)
      +  PR                                  ;Preset Q

SRC := /S* R  */PR                           ;Set-Reset Flip-Flop
      + /S*/SRT*/PR                          ;(SRC = /Q)
      +  CLR                                 ;Clear /Q

SIMULATION

TRACE_ON /OC PR CLR J K JKT T TT D DT S R SRT


SETF OC /PR CLR                              ; Clear
CLOCKF
SETF /CLR /J /K
CLOCKF
SETF K
CLOCKF
SETF J                                       ; Toggle
CLOCKF
SETF /K
CLOCKF
SETF /J
CLOCKF
SETF K
CLOCKF
SETF PR                                      ; Preset
CLOCKF
SETF /PR J K                                 ; Toggle
CLOCKF
SETF /K
CLOCKF

SETF CLR                                     ; Clear
CLOCKF
SETF /CLR /T
CLOCKF
SETF T                                       ; Toggle
CLOCKF
SETF /T                                      ; Toggle
CLOCKF


SETF CLR                                     ; Clear
CLOCKF
SETF /D
CLOCKF
SETF D
CLOCKF
SETF /D
CLOCKF
SETF CLR                                     ; Clear
CLOCKF
SETF /CLR /S /R
CLOCKF
SETF S                                       ; Set
CLOCKF
SETF /S R                                    ; Reset
CLOCKF
SETF /S R                                    ; Hold
CLOCKF
SETF /OC                                     ; HI-Z
CLOCKF
```

Simulation Results

```
     g   cg  cg  c  g  cg  cg  cg   cg  cg  cg   cg  cg  cg  c
/OC  LLLLLLLLLL LLLLLLLLLL LLLLLLLLLL LLLLLLLLLL
 PR  LLLLLLLLLL LLLLLLLLLL LLHHHLLLLL LLLLLLLLLL
 CLR HHHHLLLLLL LLLLLLLLLL LLLLLLLLLL LHHHLLLLLL
 J   XXXXLLLLLL HHHHHHLLLL LLLLLHHHHH HHHHHHHHHH
 K   XXXXLLLHHH HHHLLLLLLH HHHHHHHHLL LLLLLLLLLL
 JKT XXXLLLLLLL LLHHHHHHHH HLLLHHHLLL HHHLLLHHHH
 T   XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXXLLLHHH
 TT  XXXLLLXXXX XXXXXXXXXX XXXXHHHXXX XXXLLLLLLH
 D   XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX
 DT  XXXLLLXXXX XXXXXXXXXX XXXXHHHXXX XXXLLLXXXX
 S   XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX
 R   XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX
 SRT XXXLLLXXXX XXXXXXXXXX XXXXHHHXXX XXXLLLXXXX

Page :   2
     g  cg  cg  cg   cg  cgcg  c  g  cg  cgcg
/OC  LLLLLLLLLL LLLLLLLLLL LLLLLLLLHH
 PR  LLLLLLLLLL LLLLLLLLLL LLLLLLLLLL
 CLR LLLHHHHHHH HHHHHHHLLL LLLLLLLLLL
 J   HHHHHHHHHH HHHHHHHHHH HHHHHHHHHH
 K   LLLLLLLLLL LLLLLLLLLL LLLLLLLLLL
 JKT HHHHHLLLLL LLLLLLLLLH HHHHHHHHHZ
 T   LLLLLLLLLL LLLLLLLLLL LLLLLLLLLL
 TT  HHHHHLLLLL LLLLLLLLLL LLLLLLLLLZ
 D   XXXXXXLLLH HHLLLLLLLL LLLLLLLLLL
 DT  XXXXXLLLLL LLLLLLLLLL LLLLLLLLLZ
 S   XXXXXXXXXX XXXXXXXLLL HHHLLLLLLL
 R   XXXXXXXXXX XXXXXXXLLL LLLHHHHHHH
 SRT XXXXXLLLLL LLLLLLLLLL LLHHHLLLLZ
```

Logic Symbol



TB01730M

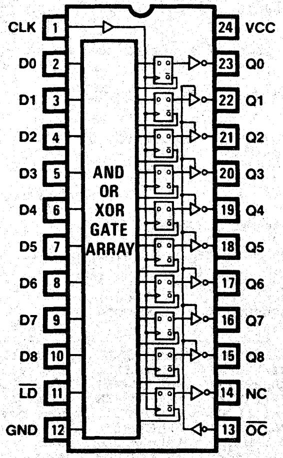
PAL Device Design Specification

```
Title    9BitRegister
Pattern  9BitReg.pds
Revision A
Author   Vincent Coli/Mehrnaz Hada
Company  Monolithic Memories Inc., Santa Clara, CA
Date     1/30/85

;This is a design of a 9-bit register with parallel load
;and hold capabilities. The operations of this register are
;summarized in the following operations table:
;
;       /OC  CLK  /LD  D8-D0  Q8-Q0    Operation
;       -----------------------------------------
;        H    X    X     X      Z      HI-Z
;        L    1    H     X      Q      Hold
;        L    1    L     D      D      Load
;       -----------------------------------------


CHIP 9BitRegister PAL20X10

CLK D0 D1 D2 D3 D4 D5 D6 D7 D8 /LD GND
/OC NC Q8 Q7 Q6 Q5 Q4 Q3 Q2 Q1  Q0 VCC

EQUATIONS

/Q0 := /D0* LD                          ;Load D0
     + /Q0*/LD                          ;Hold Q0

/Q1 := /D1* LD                          ;Load D1
     + /Q1*/LD                          ;Hold Q1

/Q2 := /D2* LD                          ;Load D2
     + /Q2*/LD                          ;Hold Q2

/Q3 := /D3* LD                          ;Load D3
     + /Q3*/LD                          ;Hold Q3

/Q4 := /D4* LD                          ;Load D4
     + /Q4*/LD                          ;Hold Q4

/Q5 := /D5* LD                          ;Load D5
     + /Q5*/LD                          ;Hold Q5

/Q6 := /D6* LD                          ;Load D6
     + /Q6*/LD                          ;Hold Q6

/Q7 := /D7* LD                          ;Load D7
     + /Q7*/LD                          ;Hold Q7

/Q8 := /D8* LD                          ;Load D8
     + /Q8*/LD                          ;Hold Q8

SIMULATION

TRACE_ON /OC CLK /LD D8 D7 D6 D5 D4 D3 D2 D1 D0
         Q8 Q7 Q6 Q5 Q4 Q3 Q2 Q1 Q0

SETF  OC LD /D8 /D7 /D6 /D5 /D4 /D3     ;Load zeros
    /D2 /D1 /D0
CLOCKF CLK

SETF /LD                                ;Hold zeros
CLOCKF CLK

SETF LD D8 D7 D6 D5 D4 D3 D2 D1 D0      ;Load ones
CLOCKF CLK

SETF /LD                                ;Hold ones
CLOCKF CLK

SETF LD /D8 D7 /D6 D5 /D4 D3 /D2 D1 /D0
CLOCKF CLK

SETF /LD                                ;Hold even ones
CLOCKF CLK

SETF  LD D8 /D7 D6 /D5 D4 /D3 D2 /D1 D0
CLOCKF CLK

SETF /LD                                ;Hold odd ones
CLOCKF CLK

SETF OC
CLOCKF CLK                              ;Test HI-Z

;Function Table for PALASM1

;/OC CLK D9 D8 D7 D6 D5 D4 D3 D2 D1 D0
;Q9 Q8 Q7 Q6 Q5 Q4 Q3 Q2 Q1 Q0
```

Simulation Results

```
;             Data In      Data Out
;Control     DDDDDDDDDD   QQQQQQQQQQ
;/OC CLK     9876543210   9876543210    Comment
;--------------------------------------------------------------
; L   C      LLLLLLLLLL   LLLLLLLLLL    Load all zeros
; L   L      XXXXXXXXXX   LLLLLLLLLL    Hold all zeros
; L   C      HHHHHHHHHH   HHHHHHHHHH    Load all ones
; L   L      XXXXXXXXXX   HHHHHHHHHH    Hold all ones
; L   C      HLHLHLHLHL   HLHLHLHLHL    Load even checkerboard
; L   L      XXXXXXXXXX   HLHLHLHLHL    Hold even checkerboard
; L   C      LHLHLHLHLH   LHLHLHLHLH    Load odd  checkerboard
; L   L      XXXXXXXXXX   LHLHLHLHLH    Hold odd  checkerboard
; H   X      XXXXXXXXXX   ZZZZZZZZZZ    Test HI-Z
;--------------------------------------------------------------
```

```
Page :  1
      g  cgcgcgc gcgcgcgcg    c
/OC  LLLLLLLLLL LLLLLLLLHH HHH
 CLK XXHLHLHLHL HLHLHLHLLL HHL
/LD  LLLLHHLLHH LLHHLLHHHH HHH
 D8  LLLLLLHHHH LLLLHHHHHH HHH
 D7  LLLLLLHHHH HHHHLLLLLL LLL
 D6  LLLLLLHHHH LLLLHHHHHH HHH
 D5  LLLLLLHHHH HHHHLLLLLL LLL
 D4  LLLLLLHHHH LLLLHHHHHH HHH
 D3  LLLLLLHHHH HHHHLLLLLL LLL
 D2  LLLLLLHHHH LLLLHHHHHH HHH
 D1  LLLLLLHHHH HHHHLLLLLL LLL
 D0  LLLLLLHHHH LLLLHHHHHH HHH
 Q8  XXXLLLLHHH HLLLLHHHHZ ZZZ
 Q7  XXXLLLLHHH HHHHHLLLLZ ZZZ
 Q6  XXXLLLLHHH HLLLLHHHHZ ZZZ
 Q5  XXXLLLLHHH HHHHHLLLLZ ZZZ
 Q4  XXXLLLLHHH HLLLLHHHHZ ZZZ
 Q3  XXXLLLLHHH HHHHHLLLLZ ZZZ
 Q2  XXXLLLLHHH HLLLLHHHHZ ZZZ
 Q1  XXXLLLLHHH HHHHHLLLLZ ZZZ
 Q0  XXXLLLLHHH HLLLLHHHHZ ZZZ
```



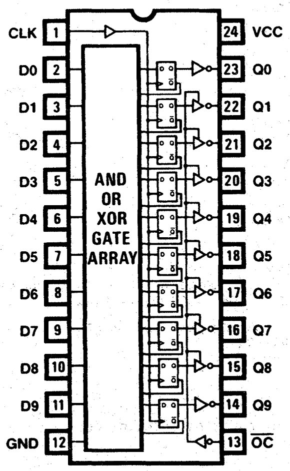
TB01740M

PAL Device Design Specification

```
Title    10BitRegister
Pattern  10BitReg.pds
Revision A
Author   Vincent Coli/Mehrnaz Hada
Company  Monolithic Memories Inc., Santa Clara, CA
Date     1/28/85

;The 10-bit register loads the data (D9-D0) on the rising
;edge of the clock(CLK) into the register(Q9-Q0). The data
;is held in the register until the next posiyive edge of
;the clock.

;
;      /OC  CLK  D9-D0  Q9-Q0    Operation
;     -----------------------------------
;      H    X     X      Z      HI-Z
;      L    C     D      D      Load
;      L    L     X      Q      Hold
;     -----------------------------------

CHIP 10BitReg PAL20X10

CLK D0 D1 D2 D3 D4 D5 D6 D7 D8 D9 GND
/OC Q9 Q8 Q7 Q6 Q5 Q4 Q3 Q2 Q1 Q0 VCC

EQUATIONS

/Q0  := /D0                               ;Load D0

/Q1  := /D1                               ;Load D1

/Q2  := /D2                               ;Load D2

/Q3  := /D3                               ;Load D3

/Q4  := /D4                               ;Load D4

/Q5  := /D5                               ;Load D5

/Q6  := /D6                               ;Load D6

/Q7  := /D7                               ;Load D7

/Q8  := /D8                               ;Load D8

/Q9  := /D9                               ;Load D9

SIMULATION

TRACE_ON /OC CLK Q9 Q8 Q7 Q6 Q5 Q4 Q3 Q2 Q1 Q0
         D9 D8 D7 D6 D5 D4 D3 D2 D1 D0

SETF OC /D9 /D8 /D7 /D6 /D5 /D4 /D3 /D2 /D1 /D0
CLOCKF CLK                                ;Load all zeros

SETF D9 D8 D7 D6 D5 D4 D3 D2 D1 D0
CLOCKF CLK                                ;Load all ones

SETF D9 /D8 D7 /D6 D5 /D4 D3 /D2 D1 /D0
CLOCKF CLK

SETF /D9 D8 /D7 D6 /D5 D4 /D3 D2 /D1 D0
CLOCKF CLK

SETF /OC
CLOCKF CLK                                ;Test HI-Z

;Function Table for PALASM1

;/OC CLK D9 D8 D7 D6 D5 D4 D3 D2 D1 D0
;Q9 Q8 Q7 Q6 Q5 Q4 Q3 Q2 Q1 Q0

;            Data In     Data Out
;Control    DDDDDDDDDD  QQQQQQQQQQ
;/OC CLK    9876543210  9876543210   Comment
;-------------------------------------------------------
; L   C     LLLLLLLLLL  LLLLLLLLLL   Load all zeros
; L   L     XXXXXXXXXX  LLLLLLLLLL   Hold all zeros
; L   C     HHHHHHHHHH  HHHHHHHHHH   Load all ones
; L   L     XXXXXXXXXX  HHHHHHHHHH   Hold all ones
; L   C     HLHLHLHLHL  HLHLHLHLHL   Load even checkerboard
; L   L     XXXXXXXXXX  HLHLHLHLHL   Hold even checkerboard
; L   C     LHLHLHLHLH  LHLHLHLHLH   Load odd  checkerboard
; L   L     XXXXXXXXXX  LHLHLHLHLH   Hold odd  checkerboard
; H   X     XXXXXXXXXX  ZZZZZZZZZZ   Test HI-Z
;-------------------------------------------------------
```

Simulation Results

```
Page :  1
       g   cg cg    cg cg    c
/OC   LLLLLLLLLL LLLLLHHHHH
 CLK  XXHHLLHLLH HLLHLLLHHL
 Q9   XXXLLLLHHH HHHHLLZZZZ
 Q8   XXXLLLLHHH LLLLHHZZZZ
 Q7   XXXLLLLHHH HHHHLLZZZZ
 Q6   XXXLLLLHHH LLLLHHZZZZ
 Q5   XXXLLLLHHH HHHHLLZZZZ
 Q4   XXXLLLLHHH LLLLHHZZZZ
 Q3   XXXLLLLHHH HHHHLLZZZZ
 Q2   XXXLLLLHHH LLLLHHZZZZ
 Q1   XXXLLLLHHH HHHHLLZZZZ
 Q0   XXXLLLLHHH LLLLHHZZZZ
 D9   LLLLLHHHHH HHLLLLLLLL
 D8   LLLLLHHHLL LLHHHHHHHH
 D7   LLLLLHHHHH HHLLLLLLLL
 D6   LLLLLHHHLL LLHHHHHHHH
 D5   LLLLLHHHHH HHLLLLLLLL
 D4   LLLLLHHHLL LLHHHHHHHH
 D3   LLLLLHHHHH HHLLLLLLLL
 D2   LLLLLHHHLL LLHHHHHHHH
 D1   LLLLLHHHHH HHLLLLLLLL
 D0   LLLLLHHHLL LLHHHHHHHH
```



TB01750M

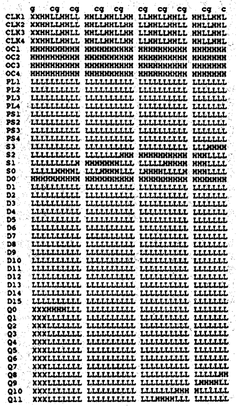
PAL Device Design Specification

```
Title     Barrel_Shifter
Pattern   Barrel.pds
Revision  A
Author    Mehrnaz Hada
Company   Monolithic Memories Inc. Santa Clara, CA
Date      1/15/85

;The 16-bit barrel shifter will shift 16 bits of data
;(D15-D0) a number of locations into the output pins, as
;specified by the binary encoded input. A compacted
;equation can be used to specify this design. It can be
;specified as following:
;
;Q[J=0..15] :=
;      OR[K=0..15](D[(J+K)-((J+K)/16)*16]*BIN[K,I=3..0]S(I))
;
;Inputs are shown by D. Si are shift amount inputs and
;Qj are outputs. 16 product terms in each output pair
;are directed to one output; thus only 16 out of 32
;output pins are used.

CHIP BarrelShift PAL64R32

D7 D6 D5 D4 D3 D2 D1 D0 /PL1 /PS1 GND CLK1
/OC1 Q0 NC Q1 NC Q2 NC Q3 NC Q4 NC Q5 NC Q6
NC Q7 NC /OC2 CLK2 VCC /PS2 /PL2 NC NC NC
NC NC S0 S1 S2 S3 NC NC NC NC NC NC NC
/PL3 /PS3 GND CLK3 /OC3 NC Q8 NC Q9 NC Q10
NC Q11 NC Q12 NC Q13 NC Q14 NC Q15 /OC4
CLK4 VCC /PS4 /PL4 D15 D14 D13 D12 D11 D10
D9 D8

EQUATIONS

Q0 := /S3 * /S2 * /S1 * /S0 * D0          ; No shift
   +  /S3 * /S2 * /S1 *  S0 * D1          ; Shift 1 space
   +  /S3 * /S2 *  S1 * /S0 * D2          ; Shift 2 spaces
   +  /S3 * /S2 *  S1 *  S0 * D3          ; Shift 3 spaces
   +  /S3 *  S2 * /S1 * /S0 * D4          ; Shift 4 spaces
   +  /S3 *  S2 * /S1 *  S0 * D5          ; Shift 5 spaces
   +  /S3 *  S2 *  S1 * /S0 * D6          ; Shift 6 spaces
   +  /S3 *  S2 *  S1 *  S0 * D7          ; Shift 7 spaces
   +   S3 * /S2 * /S1 * /S0 * D8          ; Shift 8 spaces
   +   S3 * /S2 * /S1 *  S0 * D9          ; Shift 9 spaces
   +   S3 * /S2 *  S1 * /S0 * D10         ; Shift 10 spaces
   +   S3 * /S2 *  S1 *  S0 * D11         ; Shift 11 spaces
   +   S3 *  S2 * /S1 * /S0 * D12         ; Shift 12 spaces
   +   S3 *  S2 * /S1 *  S0 * D13         ; Shift 13 spaces
   +   S3 *  S2 *  S1 * /S0 * D14         ; Shift 14 spaces
   +   S3 *  S2 *  S1 *  S0 * D15         ; Shift 15 spaces

Q1 := /S3 * /S2 * /S1 * /S0 * D1          ; No shift
   +  /S3 * /S2 * /S1 *  S0 * D2          ; Shift 1 space
   +  /S3 * /S2 *  S1 * /S0 * D3          ; Shift 2 spaces
   +  /S3 * /S2 *  S1 *  S0 * D4          ; Shift 3 spaces
   +  /S3 *  S2 * /S1 * /S0 * D5          ; Shift 4 spaces
   +  /S3 *  S2 * /S1 *  S0 * D6          ; Shift 5 spaces
   +  /S3 *  S2 *  S1 * /S0 * D7          ; Shift 6 spaces
   +  /S3 *  S2 *  S1 *  S0 * D8          ; Shift 7 spaces
   +   S3 * /S2 * /S1 * /S0 * D9          ; Shift 8 spaces
   +   S3 * /S2 * /S1 *  S0 * D10         ; Shift 9 spaces
   +   S3 * /S2 *  S1 * /S0 * D11         ; Shift 10 spaces
   +   S3 * /S2 *  S1 *  S0 * D12         ; Shift 11 spaces
   +   S3 *  S2 * /S1 * /S0 * D13         ; Shift 12 spaces
   +   S3 *  S2 * /S1 *  S0 * D14         ; Shift 13 spaces
   +   S3 *  S2 *  S1 * /S0 * D15         ; Shift 14 spaces
   +   S3 *  S2 *  S1 *  S0 * D0          ; Shift 15 spaces

Q2 := /S3 * /S2 * /S1 * /S0 * D2          ; No shift
   +  /S3 * /S2 * /S1 *  S0 * D3          ; Shift 1 space
   +  /S3 * /S2 *  S1 * /S0 * D4          ; Shift 2 spaces
   +  /S3 * /S2 *  S1 *  S0 * D5          ; Shift 3 spaces
   +  /S3 *  S2 * /S1 * /S0 * D6          ; Shift 4 spaces
   +  /S3 *  S2 * /S1 *  S0 * D7          ; Shift 5 spaces
   +  /S3 *  S2 *  S1 * /S0 * D8          ; Shift 6 spaces
   +  /S3 *  S2 *  S1 *  S0 * D9          ; Shift 7 spaces
   +   S3 * /S2 * /S1 * /S0 * D10         ; Shift 8 spaces
   +   S3 * /S2 * /S1 *  S0 * D11         ; Shift 9 spaces
   +   S3 * /S2 *  S1 * /S0 * D12         ; Shift 10 spaces
   +   S3 * /S2 *  S1 *  S0 * D13         ; Shift 11 spaces
   +   S3 *  S2 * /S1 * /S0 * D14         ; Shift 12 spaces
   +   S3 *  S2 * /S1 *  S0 * D15         ; Shift 13 spaces
   +   S3 *  S2 *  S1 * /S0 * D0          ; Shift 14 spaces
   +   S3 *  S2 *  S1 *  S0 * D1          ; Shift 15 spaces

Q3 := /S3 * /S2 * /S1 * /S0 * D3          ; No shift
   +  /S3 * /S2 * /S1 *  S0 * D4          ; Shift 1 space
   +  /S3 * /S2 *  S1 * /S0 * D5          ; Shift 2 spaces
   +  /S3 * /S2 *  S1 *  S0 * D6          ; Shift 3 spaces
   +  /S3 *  S2 * /S1 * /S0 * D7          ; Shift 4 spaces
   +  /S3 *  S2 * /S1 *  S0 * D8          ; Shift 5 spaces
   +  /S3 *  S2 *  S1 * /S0 * D9          ; Shift 6 spaces
   +  /S3 *  S2 *  S1 *  S0 * D10         ; Shift 7 spaces
   +   S3 * /S2 * /S1 * /S0 * D11         ; Shift 8 spaces
   +   S3 * /S2 * /S1 *  S0 * D12         ; Shift 9 spaces
   +   S3 * /S2 *  S1 * /S0 * D13         ; Shift 10 spaces
   +   S3 * /S2 *  S1 *  S0 * D14         ; Shift 11 spaces
   +   S3 *  S2 * /S1 * /S0 * D15         ; Shift 12 spaces
   +   S3 *  S2 * /S1 *  S0 * D0          ; Shift 13 spaces
   +   S3 *  S2 *  S1 * /S0 * D1          ; Shift 14 spaces
   +   S3 *  S2 *  S1 *  S0 * D2          ; Shift 15 spaces

Q4 := /S3 * /S2 * /S1 * /S0 * D4          ; No shift
   +  /S3 * /S2 * /S1 *  S0 * D5          ; Shift 1 space
   +  /S3 * /S2 *  S1 * /S0 * D6          ; Shift 2 spaces
   +  /S3 * /S2 *  S1 *  S0 * D7          ; Shift 3 spaces
   +  /S3 *  S2 * /S1 * /S0 * D8          ; Shift 4 spaces
   +  /S3 *  S2 * /S1 *  S0 * D9          ; Shift 5 spaces
   +  /S3 *  S2 *  S1 * /S0 * D10         ; Shift 6 spaces
   +  /S3 *  S2 *  S1 *  S0 * D11         ; Shift 7 spaces
   +   S3 * /S2 * /S1 * /S0 * D12         ; Shift 8 spaces
   +   S3 * /S2 * /S1 *  S0 * D13         ; Shift 9 spaces
   +   S3 * /S2 *  S1 * /S0 * D14         ; Shift 10 spaces
   +   S3 * /S2 *  S1 *  S0 * D15         ; Shift 11 spaces
   +   S3 *  S2 * /S1 * /S0 * D0          ; Shift 12 spaces
   +   S3 *  S2 * /S1 *  S0 * D1          ; Shift 13 spaces
   +   S3 *  S2 *  S1 * /S0 * D2          ; Shift 14 spaces
   +   S3 *  S2 *  S1 *  S0 * D3          ; Shift 15 spaces

Q5 := /S3 * /S2 * /S1 * /S0 * D5          ; No shift
   +  /S3 * /S2 * /S1 *  S0 * D6          ; Shift 1 space
   +  /S3 * /S2 *  S1 * /S0 * D7          ; Shift 2 spaces
   +  /S3 * /S2 *  S1 *  S0 * D8          ; Shift 3 spaces
   +  /S3 *  S2 * /S1 * /S0 * D9          ; Shift 4 spaces
   +  /S3 *  S2 * /S1 *  S0 * D10         ; Shift 5 spaces
   +  /S3 *  S2 *  S1 * /S0 * D11         ; Shift 6 spaces
   +  /S3 *  S2 *  S1 *  S0 * D12         ; Shift 7 spaces
   +   S3 * /S2 * /S1 * /S0 * D13         ; Shift 8 spaces
   +   S3 * /S2 * /S1 *  S0 * D14         ; Shift 9 spaces
   +   S3 * /S2 *  S1 * /S0 * D15         ; Shift 10 spaces
   +   S3 * /S2 *  S1 *  S0 * D0          ; Shift 11 spaces
   +   S3 *  S2 * /S1 * /S0 * D1          ; Shift 12 spaces
   +   S3 *  S2 * /S1 *  S0 * D2          ; Shift 13 spaces
   +   S3 *  S2 *  S1 * /S0 * D3          ; Shift 14 spaces
   +   S3 *  S2 *  S1 *  S0 * D4          ; Shift 15 spaces

Q6 := /S3 * /S2 * /S1 * /S0 * D6          ; No shift
   +  /S3 * /S2 * /S1 *  S0 * D7          ; Shift 1 space
   +  /S3 * /S2 *  S1 * /S0 * D8          ; Shift 2 spaces
   +  /S3 * /S2 *  S1 *  S0 * D9          ; Shift 3 spaces
   +  /S3 *  S2 * /S1 * /S0 * D10         ; Shift 4 spaces
   +  /S3 *  S2 * /S1 *  S0 * D11         ; Shift 5 spaces
   +  /S3 *  S2 *  S1 * /S0 * D12         ; Shift 6 spaces
   +  /S3 *  S2 *  S1 *  S0 * D13         ; Shift 7 spaces
   +   S3 * /S2 * /S1 * /S0 * D14         ; Shift 8 spaces
   +   S3 * /S2 * /S1 *  S0 * D15         ; Shift 9 spaces
   +   S3 * /S2 *  S1 * /S0 * D0          ; Shift 10 spaces
   +   S3 * /S2 *  S1 *  S0 * D1          ; Shift 11 spaces
   +   S3 *  S2 * /S1 * /S0 * D2          ; Shift 12 spaces
   +   S3 *  S2 * /S1 *  S0 * D3          ; Shift 13 spaces
   +   S3 *  S2 *  S1 * /S0 * D4          ; Shift 14 spaces
   +   S3 *  S2 *  S1 *  S0 * D5          ; Shift 15 spaces

Q7 := /S3 * /S2 * /S1 * /S0 * D7          ; No shift
   +  /S3 * /S2 * /S1 *  S0 * D8          ; Shift 1 space
   +  /S3 * /S2 *  S1 * /S0 * D9          ; Shift 2 spaces
   +  /S3 * /S2 *  S1 *  S0 * D10         ; Shift 3 spaces
   +  /S3 *  S2 * /S1 * /S0 * D11         ; Shift 4 spaces
   +  /S3 *  S2 * /S1 *  S0 * D12         ; Shift 5 spaces
   +  /S3 *  S2 *  S1 * /S0 * D13         ; Shift 6 spaces
   +  /S3 *  S2 *  S1 *  S0 * D14         ; Shift 7 spaces
   +   S3 * /S2 * /S1 * /S0 * D15         ; Shift 8 spaces
   +   S3 * /S2 * /S1 *  S0 * D0          ; Shift 9 spaces
   +   S3 * /S2 *  S1 * /S0 * D1          ; Shift 10 spaces
   +   S3 * /S2 *  S1 *  S0 * D2          ; Shift 11 spaces
   +   S3 *  S2 * /S1 * /S0 * D3          ; Shift 12 spaces
   +   S3 *  S2 * /S1 *  S0 * D4          ; Shift 13 spaces
   +   S3 *  S2 *  S1 * /S0 * D5          ; Shift 14 spaces
   +   S3 *  S2 *  S1 *  S0 * D6          ; Shift 15 spaces

Q8 := /S3 * /S2 * /S1 * /S0 * D8          ; No shift
   +  /S3 * /S2 * /S1 *  S0 * D9          ; Shift 1 space
   +  /S3 * /S2 *  S1 * /S0 * D10         ; Shift 2 spaces
   +  /S3 * /S2 *  S1 *  S0 * D11         ; Shift 3 spaces
   +  /S3 *  S2 * /S1 * /S0 * D12         ; Shift 4 spaces
   +  /S3 *  S2 * /S1 *  S0 * D13         ; Shift 5 spaces
   +  /S3 *  S2 *  S1 * /S0 * D14         ; Shift 6 spaces
   +  /S3 *  S2 *  S1 *  S0 * D15         ; Shift 7 spaces
   +   S3 * /S2 * /S1 * /S0 * D0          ; Shift 8 spaces
   +   S3 * /S2 * /S1 *  S0 * D1          ; Shift 9 spaces
   +   S3 * /S2 *  S1 * /S0 * D2          ; Shift 10 spaces
   +   S3 * /S2 *  S1 *  S0 * D3          ; Shift 11 spaces
   +   S3 *  S2 * /S1 * /S0 * D4          ; Shift 12 spaces
   +   S3 *  S2 * /S1 *  S0 * D5          ; Shift 13 spaces
   +   S3 *  S2 *  S1 * /S0 * D6          ; Shift 14 spaces
   +   S3 *  S2 *  S1 *  S0 * D7          ; Shift 15 spaces

Q9 := /S3 * /S2 * /S1 * /S0 * D9          ; No shift
   +  /S3 * /S2 * /S1 *  S0 * D10         ; Shift 1 space
   +  /S3 * /S2 *  S1 * /S0 * D11         ; Shift 2 spaces
   +  /S3 * /S2 *  S1 *  S0 * D12         ; Shift 3 spaces
   +  /S3 *  S2 * /S1 * /S0 * D13         ; Shift 4 spaces
   +  /S3 *  S2 * /S1 *  S0 * D14         ; Shift 5 spaces
   +  /S3 *  S2 *  S1 * /S0 * D15         ; Shift 6 spaces
```

TB01760M

```
     + /S3 *  S2 *  S1 *  S0 * D0           ; Shift 7 spaces
     +  S3 * /S2 * /S1 * /S0 * D1           ; Shift 8 spaces
     +  S3 * /S2 * /S1 *  S0 * D2           ; Shift 9 spaces
     +  S3 * /S2 *  S1 * /S0 * D3           ; Shift 10 spaces
     +  S3 * /S2 *  S1 *  S0 * D4           ; Shift 11 spaces
     +  S3 *  S2 * /S1 * /S0 * D5           ; Shift 12 spaces
     +  S3 *  S2 * /S1 *  S0 * D6           ; Shift 13 spaces
     +  S3 *  S2 *  S1 * /S0 * D7           ; Shift 14 spaces
     +  S3 *  S2 *  S1 *  S0 * D8           ; Shift 15 spaces

Q10 :=/S3 * /S2 * /S1 * /S0 * D10           ; No shift
     + /S3 * /S2 * /S1 *  S0 * D11          ; Shift 1 space
     + /S3 * /S2 *  S1 * /S0 * D12          ; Shift 2 spaces
     + /S3 * /S2 *  S1 *  S0 * D13          ; Shift 3 spaces
     + /S3 *  S2 * /S1 * /S0 * D14          ; Shift 4 spaces
     + /S3 *  S2 * /S1 *  S0 * D15          ; Shift 5 spaces
     + /S3 *  S2 *  S1 * /S0 * D0           ; Shift 6 spaces
     + /S3 *  S2 *  S1 *  S0 * D1           ; Shift 7 spaces
     +  S3 * /S2 * /S1 * /S0 * D2           ; Shift 8 spaces
     +  S3 * /S2 * /S1 *  S0 * D3           ; Shift 9 spaces
     +  S3 * /S2 *  S1 * /S0 * D4           ; Shift 10 Spaces
     +  S3 * /S2 *  S1 *  S0 * D5           ; Shift 11 spaces
     +  S3 *  S2 * /S1 * /S0 * D6           ; Shift 12 spaces
     +  S3 *  S2 * /S1 *  S0 * D7           ; Shift 13 spaces
     +  S3 *  S2 *  S1 * /S0 * D8           ; Shift 14 spaces
     +  S3 *  S2 *  S1 *  S0 * D9           ; Shift 15 spaces

Q11 :=/S3 * /S2 * /S1 * /S0 * D11           ; No shift
     + /S3 * /S2 * /S1 *  S0 * D12          ; Shift 1 space
     + /S3 * /S2 *  S1 * /S0 * D13          ; Shift 2 spaces
     + /S3 * /S2 *  S1 *  S0 * D14          ; Shift 3 spaces
     + /S3 *  S2 * /S1 * /S0 * D15          ; Shift 4 spaces
     + /S3 *  S2 * /S1 *  S0 * D0           ; Shift 5 spaces
     + /S3 *  S2 *  S1 * /S0 * D1           ; Shift 6 spaces
     + /S3 *  S2 *  S1 *  S0 * D2           ; Shift 7 spaces
     +  S3 * /S2 * /S1 * /S0 * D3           ; Shift 8 spaces
     +  S3 * /S2 * /S1 *  S0 * D4           ; Shift 9 spaces
     +  S3 * /S2 *  S1 * /S0 * D5           ; Shift 10 spaces
     +  S3 * /S2 *  S1 *  S0 * D6           ; Shift 11 spaces
     +  S3 *  S2 * /S1 * /S0 * D7           ; Shift 12 spaces
     +  S3 *  S2 * /S1 *  S0 * D8           ; Shift 13 spaces
     +  S3 *  S2 *  S1 * /S0 * D9           ; Shift 14 spaces
     +  S3 *  S2 *  S1 *  S0 * D10          ; Shift 15 spaces

Q12 :=/S3 * /S2 * /S1 * /S0 * D12           ; No shift
     + /S3 * /S2 * /S1 *  S0 * D13          ; Shift 1 space
     + /S3 * /S2 *  S1 * /S0 * D14          ; Shift 2 spaces
     + /S3 * /S2 *  S1 *  S0 * D15          ; Shift 3 spaces
     + /S3 *  S2 * /S1 * /S0 * D0           ; Shift 4 spaces
     + /S3 *  S2 * /S1 *  S0 * D1           ; Shift 5 spaces
     + /S3 *  S2 *  S1 * /S0 * D2           ; Shift 6 spaces
     + /S3 *  S2 *  S1 *  S0 * D3           ; Shift 7 spaces
     +  S3 * /S2 * /S1 * /S0 * D4           ; Shift 8 spaces
     +  S3 * /S2 * /S1 *  S0 * D5           ; Shift 9 spaces
     +  S3 * /S2 *  S1 * /S0 * D6           ; Shift 10 spaces
     +  S3 * /S2 *  S1 *  S0 * D7           ; Shift 11 spaces
     +  S3 *  S2 * /S1 * /S0 * D8           ; Shift 12 spaces
     +  S3 *  S2 * /S1 *  S0 * D9           ; Shift 13 spaces
     +  S3 *  S2 *  S1 * /S0 * D10          ; Shift 14 spaces
     +  S3 *  S2 *  S1 *  S0 * D11          ; Shift 15 spaces

Q13 :=/S3 * /S2 * /S1 * /S0 * D13           ; No shift
     + /S3 * /S2 * /S1 *  S0 * D14          ; Shift 1 space
     + /S3 * /S2 *  S1 * /S0 * D15          ; Shift 2 spaces
     + /S3 * /S2 *  S1 *  S0 * D0           ; Shift 3 spaces
     + /S3 *  S2 * /S1 * /S0 * D1           ; Shift 4 spaces
     + /S3 *  S2 * /S1 *  S0 * D2           ; Shift 5 spaces
     + /S3 *  S2 *  S1 * /S0 * D3           ; Shift 6 spaces
     + /S3 *  S2 *  S1 *  S0 * D4           ; Shift 7 spaces
     +  S3 * /S2 * /S1 * /S0 * D5           ; Shift 8 spaces
     +  S3 * /S2 * /S1 *  S0 * D6           ; Shift 9 spaces
     +  S3 * /S2 *  S1 * /S0 * D7           ; Shift 10 spaces
     +  S3 * /S2 *  S1 *  S0 * D8           ; Shift 11 spaces
     +  S3 *  S2 * /S1 * /S0 * D9           ; Shift 12 spaces
     +  S3 *  S2 * /S1 *  S0 * D10          ; Shift 13 spaces
     +  S3 *  S2 *  S1 * /S0 * D11          ; Shift 14 spaces
     +  S3 *  S2 *  S1 *  S0 * D12          ; Shift 15 spaces

Q14 :=/S3 * /S2 * /S1 * /S0 * D14           ; No shift
     + /S3 * /S2 * /S1 *  S0 * D15          ; Shift 1 space
     + /S3 * /S2 *  S1 * /S0 * D0           ; Shift 2 spaces
     + /S3 * /S2 *  S1 *  S0 * D1           ; Shift 3 spaces
     + /S3 *  S2 * /S1 * /S0 * D2           ; Shift 4 spaces
     + /S3 *  S2 * /S1 *  S0 * D3           ; Shift 5 spaces
     + /S3 *  S2 *  S1 * /S0 * D4           ; Shift 6 spaces
     + /S3 *  S2 *  S1 *  S0 * D5           ; Shift 7 spaces
     +  S3 * /S2 * /S1 * /S0 * D6           ; Shift 8 spaces
     +  S3 * /S2 * /S1 *  S0 * D7           ; Shift 9 spaces
     +  S3 * /S2 *  S1 * /S0 * D8           ; Shift 10 spaces
     +  S3 * /S2 *  S1 *  S0 * D9           ; Shift 11 spaces
     +  S3 *  S2 * /S1 * /S0 * D10          ; Shift 12 spaces
     +  S3 *  S2 * /S1 *  S0 * D11          ; Shift 13 spaces
     +  S3 *  S2 *  S1 * /S0 * D12          ; Shift 14 spaces
     +  S3 *  S2 *  S1 *  S0 * D13          ; Shift 15 spaces

Q15 :=/S3 * /S2 * /S1 * /S0 * D15           ; No shift
     + /S3 * /S2 * /S1 *  S0 * D0           ; Shift 1 space
     + /S3 * /S2 *  S1 * /S0 * D1           ; Shift 2 spaces
     + /S3 * /S2 *  S1 *  S0 * D2           ; Shift 3 spaces
     + /S3 *  S2 * /S1 * /S0 * D3           ; Shift 4 spaces
     + /S3 *  S2 * /S1 *  S0 * D4           ; Shift 5 spaces
     + /S3 *  S2 *  S1 * /S0 * D5           ; Shift 6 spaces
     + /S3 *  S2 *  S1 *  S0 * D6           ; Shift 7 spaces
     +  S3 * /S2 * /S1 * /S0 * D7           ; Shift 8 spaces
     +  S3 * /S2 * /S1 *  S0 * D8           ; Shift 9 spaces
     +  S3 * /S2 *  S1 * /S0 * D9           ; Shift 10 spaces
     +  S3 * /S2 *  S1 *  S0 * D10          ; Shift 11 spaces
     +  S3 *  S2 * /S1 * /S0 * D11          ; Shift 12 spaces
     +  S3 *  S2 * /S1 *  S0 * D12          ; Shift 13 spaces
     +  S3 *  S2 *  S1 * /S0 * D13          ; Shift 14 spaces
     +  S3 *  S2 *  S1 *  S0 * D14          ; Shift 15 spaces

SIMULATION

TRACE_ON CLK1 CLK2 CLK3 CLK4 OC1 OC2 OC3 OC4
         PL1 PL2 PL3 PL4 PS1 PS2 PS3 PS4 S3
         S2 S1 S0 D0 D1 D2 D3 D4 D5 D6 D7 D8
         D9 D10 D11 D12 D13 D14 D15 Q0 Q1 Q2
         Q3 Q4 Q5 Q6 Q7 Q8 Q9 Q10 Q11 Q12 Q13
         Q14 Q15

SETF OC1 OC2 OC3 OC4 /PS1 /PS2 /PS3 /PS4
    /PL1 /PL2 /PL3 /PL4 /S3 /S2 /S1 /S0 D0
    /D1 /D2 /D3 /D4 /D5 /D6 /D7 /D8 /D9 /D10
    /D11 /D12 /D13 /D14 /D15

CLOCKF CLK1 CLK2 CLK3 CLK4               ;Clock

SETF /S3 /S2 /S1 S0                      ;Shift 1
CLOCKF CLK1 CLK2 CLK3 CLK4

SETF /S3 /S2 S1 /S0                      ;Shift 2
CLOCKF CLK1 CLK2 CLK3 CLK4

SETF /S3 /S2 S1 S0                       ;Shift 3
CLOCKF CLK1 CLK2 CLK3 CLK4

SETF /S3 S2 /S1 /S0                      ;Shift 4
CLOCKF CLK1 CLK2 CLK3 CLK4

SETF /S3 S2 /S1 S0                       ;Shift 5
CLOCKF CLK1 CLK2 CLK3 CLK4

SETF /S3 S2 S1 /S0                       ;Shift 6
CLOCKF CLK1 CLK2 CLK3 CLK4

SETF /S3 S2 S1 S0                        ;Shift 7
CLOCKF CLK1 CLK2 CLK3 CLK4

SETF S3 /S2 /S1 /S0                      ;Shift 8
CLOCKF CLK1 CLK2 CLK3 CLK4

;The 16-bit barrel shifter will shift 16 bits of data
;(D15-D0) a number of locations into the output pins, as
;specified by the binary encoded input. A compacted
;equation can be used to specify this design. It can be
;specified as following:
;
;Q[J=0..15] :=
;  OR[K=0..15](D[(J+K)-((J+K)/16)*16]*BIN[K,I=3..0]S(I))
;
;Inputs are shown by D. Si are shift amount inputs and
;Qj are outputs. 16 product terms in each output pair
;are directed to one output; thus only 16 out of 32
;output pins are used.
```

Simulation Results



```
      g   cg   cg    cg   cg    cg   cg   cg    cg  c
CLK1 XXHHLLHHLL HHLLHHLLHH LLHHLLHHLL HHLLHHL
CLK2 XXHHLLHHLL HHLLHHLLHH LLHHLLHHLL HHLLHHL
CLK3 XXHHLLHHLL HHLLHHLLHH LLHHLLHHLL HHLLHHL
CLK4 XXHHLLHHLL HHLLHHLLHH LLHHLLHHLL HHLLHHL
OC1  HHHHHHHHHH HHHHHHHHHH HHHHHHHHHH HHHHHHH
OC2  HHHHHHHHHH HHHHHHHHHH HHHHHHHHHH HHHHHHH
OC3  HHHHHHHHHH HHHHHHHHHH HHHHHHHHHH HHHHHHH
OC4  HHHHHHHHHH HHHHHHHHHH HHHHHHHHHH HHHHHHH
PL1  LLLLLLLLLL LLLLLLLLLL LLLLLLLLLL LLLLLLL
PL2  LLLLLLLLLL LLLLLLLLLL LLLLLLLLLL LLLLLLL
PL3  LLLLLLLLLL LLLLLLLLLL LLLLLLLLLL LLLLLLL
PL4  LLLLLLLLLL LLLLLLLLLL LLLLLLLLLL LLLLLLL
PS1  LLLLLLLLLL LLLLLLLLLL LLLLLLLLLL LLLLLLL
PS2  LLLLLLLLLL LLLLLLLLLL LLLLLLLLLL LLLLLLL
PS3  LLLLLLLLLL LLLLLLLLLL LLLLLLLLLL LLLLLLL
PS4  LLLLLLLLLL LLLLLLLLLL LLLLLLLLLL LLLLLLL
S3   LLLLLLLLLL LLLLLLLLLL LLLLLLLLLL LLLHHHH
S2   LLLLLLLLLL LLLLLLLHHH HHHHHHHHHH HHHLLLL
S1   LLLLLLLLLH HHHHHHHLLL LLLLLHHHHH HHHLLLL
S0   LLLLLHHHHL LLLHHHHLLL LHHHHLLLLH HHHLLLL
D0   HHHHHHHHHH HHHHHHHHHH HHHHHHHHHH HHHHHHH
D1   LLLLLLLLLL LLLLLLLLLL LLLLLLLLLL LLLLLLL
D2   LLLLLLLLLL LLLLLLLLLL LLLLLLLLLL LLLLLLL
D3   LLLLLLLLLL LLLLLLLLLL LLLLLLLLLL LLLLLLL
D4   LLLLLLLLLL LLLLLLLLLL LLLLLLLLLL LLLLLLL
D5   LLLLLLLLLL LLLLLLLLLL LLLLLLLLLL LLLLLLL
D6   LLLLLLLLLL LLLLLLLLLL LLLLLLLLLL LLLLLLL
D7   LLLLLLLLLL LLLLLLLLLL LLLLLLLLLL LLLLLLL
D8   LLLLLLLLLL LLLLLLLLLL LLLLLLLLLL LLLLLLL
D9   LLLLLLLLLL LLLLLLLLLL LLLLLLLLLL LLLLLLL
D10  LLLLLLLLLL LLLLLLLLLL LLLLLLLLLL LLLLLLL
D11  LLLLLLLLLL LLLLLLLLLL LLLLLLLLLL LLLLLLL
D12  LLLLLLLLLL LLLLLLLLLL LLLLLLLLLL LLLLLLL
D13  LLLLLLLLLL LLLLLLLLLL LLLLLLLLLL LLLLLLL
D14  LLLLLLLLLL LLLLLLLLLL LLLLLLLLLL LLLLLLL
D15  LLLLLLLLLL LLLLLLLLLL LLLLLLLLLL LLLLLLL
Q0   XXXHHHHLLL LLLLLLLLLL LLLLLLLLLL LLLLLLL
Q1   XXXLLLLLLL LLLLLLLLLL LLLLLLLLLL LLLLLLL
Q2   XXXLLLLLLL LLLLLLLLLL LLLLLLLLLL LLLLLLL
Q3   XXXLLLLLLL LLLLLLLLLL LLLLLLLLLL LLLLLLL
Q4   XXXLLLLLLL LLLLLLLLLL LLLLLLLLLL LLLLLLL
Q5   XXXLLLLLLL LLLLLLLLLL LLLLLLLLLL LLLLLLL
Q6   XXXLLLLLLL LLLLLLLLLL LLLLLLLLLL LLLLLLL
Q7   XXXLLLLLLL LLLLLLLLLL LLLLLLLLLL LLLLLLL
Q8   XXXLLLLLLL LLLLLLLLLL LLLLLLLLLL LLLLLHH
Q9   XXXLLLLLLL LLLLLLLLLL LLLLLLLLLL LHHHHLL
Q10  XXXLLLLLLL LLLLLLLLLL LLLLLLLHHH HLLLLLL
Q11  XXXLLLLLLL LLLLLLLLLL LLLHHHHLLL LLLLLLL
```

TB01770M

PAL Device Design Specification

```
TITLE     16-BIT ADDRESSABLE REGISTER
PATTERN   ADREG16.PDS
REVISION  A
AUTHOR    John Birkner
COMPANY   Monolithic Memories Inc. Santa Clara, CA
DATE      2/11/85

;       The 16-bit addressable register loads one of 16 registers
;       selected by ADDR[0..3] with data input, DATA.

CHIP ADREG16 PAL32R16
Q0 Q1 Q2 Q3 /E1 NC NC A0 A1 VCC A2 A3 DATA NC /PRLD2 CLK2
Q4 Q5 Q6 Q7 Q8 Q9 Q10 Q11 /E2 NC NC NC NC GND NC NC NC NC /PRLD1 CLK1
Q12 Q13 Q14 Q15

EQUATIONS

Q0      :=  A0            *Q0           ;hold
        +       A1        *Q0           ;hold
        +           A2    *Q0           ;hold
        +               A3*Q0           ;hold
        +  /A0*/A1*/A2*/A3*DATA         ;load

Q1      := /A0            *Q1           ;hold
        +       A1        *Q1           ;hold
        +           A2    *Q1           ;hold
        +               A3*Q1           ;hold
        +   A0*/A1*/A2*/A3*DATA         ;load

Q2      :=  A0            *Q2           ;hold
        +      /A1        *Q2           ;hold
        +           A2    *Q2           ;hold
        +               A3*Q2           ;hold
        +  /A0* A1*/A2*/A3*DATA         ;load

Q3      := /A0            *Q3           ;hold
        +      /A1        *Q3           ;hold
        +           A2    *Q3           ;hold
        +               A3*Q3           ;hold
        +   A0* A1*/A2*/A3*DATA         ;load

Q4      :=  A0            *Q4           ;hold
        +       A1        *Q4           ;hold
        +          /A2    *Q4           ;hold
        +               A3*Q4           ;hold
        +  /A0*/A1* A2*/A3*DATA         ;load

Q5      := /A0            *Q5           ;hold
        +       A1        *Q5           ;hold
        +          /A2    *Q5           ;hold
        +               A3*Q5           ;hold
        +   A0*/A1* A2*/A3*DATA         ;load

Q6      :=  A0            *Q6           ;hold
        +      /A1        *Q6           ;hold
        +          /A2    *Q6           ;hold
        +               A3*Q6           ;hold
        +  /A0* A1* A2*/A3*DATA         ;load

Q7      := /A0            *Q7           ;hold
        +      /A1        *Q7           ;hold
        +          /A2    *Q7           ;hold
        +               A3*Q7           ;hold
        +   A0* A1* A2*/A3*DATA         ;load

Q8      :=  A0            *Q8           ;hold
        +       A1        *Q8           ;hold
        +           A2    *Q8           ;hold
        +              /A3*Q8           ;hold
        +  /A0*/A1*/A2* A3*DATA         ;load

Q9      := /A0            *Q9           ;hold
        +       A1        *Q9           ;hold
        +           A2    *Q9           ;hold
        +              /A3*Q9           ;hold
        +   A0*/A1*/A2* A3*DATA         ;load

Q10     :=  A0            *Q10          ;hold
        +      /A1        *Q10          ;hold
        +           A2    *Q10          ;hold
        +              /A3*Q10          ;hold
        +  /A0* A1*/A2* A3*DATA         ;load

Q11     := /A0            *Q11          ;hold
        +      /A1        *Q11          ;hold
        +           A2    *Q11          ;hold
        +              /A3*Q11          ;hold
        +   A0* A1*/A2* A3*DATA         ;load

Q12     :=  A0            *Q12          ;hold
        +       A1        *Q12          ;hold
        +          /A2    *Q12          ;hold
        +              /A3*Q12          ;hold
        +  /A0*/A1* A2* A3*DATA         ;load

Q13     := /A0            *Q13          ;hold
        +       A1        *Q13          ;hold
        +          /A2    *Q13          ;hold
        +              /A3*Q13          ;hold
        +   A0*/A1* A2* A3*DATA         ;load

Q14     :=  A0            *Q14          ;hold
        +      /A1        *Q14          ;hold
        +          /A2    *Q14          ;hold
        +              /A3*Q14          ;hold
        +  /A0* A1* A2* A3*DATA         ;load

Q15     := /A0            *Q15          ;hold
        +      /A1        *Q15          ;hold
        +          /A2    *Q15          ;hold
        +              /A3*Q15          ;hold
        +   A0* A1* A2* A3*DATA         ;load

SIMULATION

TRACE_ON Q0 Q1 Q2 Q3 Q4 Q5 Q6 Q7 Q8 Q9 Q10 Q11 Q12 Q13 Q14 Q15
         A0 A1 A2 A3 DATA

SETF E1 E2 /DATA /PRLD1 /PRLD2

        SETF /A0 /A1 /A2 /A3
        CLOCKF CLK1 CLK2

        SETF  A0 /A1 /A2 /A3
        CLOCKF CLK1 CLK2

        SETF /A0  A1 /A2 /A3
        CLOCKF CLK1 CLK2

        SETF  A0  A1 /A2 /A3
        CLOCKF CLK1 CLK2

        SETF /A0 /A1  A2 /A3
        CLOCKF CLK1 CLK2

        SETF  A0 /A1  A2 /A3
        CLOCKF CLK1 CLK2

        SETF /A0  A1  A2 /A3
        CLOCKF CLK1 CLK2

        SETF  A0  A1  A2 /A3
        CLOCKF CLK1 CLK2

        SETF /A0 /A1 /A2  A3
        CLOCKF CLK1 CLK2

        SETF  A0 /A1 /A2  A3
        CLOCKF CLK1 CLK2

        SETF /A0  A1 /A2  A3
        CLOCKF CLK1 CLK2

        SETF  A0  A1 /A2  A3
        CLOCKF CLK1 CLK2

        SETF /A0 /A1  A2  A3
        CLOCKF CLK1 CLK2

        SETF  A0 /A1  A2  A3
        CLOCKF CLK1 CLK2

        SETF /A0  A1  A2  A3
        CLOCKF CLK1 CLK2

        SETF  A0  A1  A2  A3
        CLOCKF CLK1 CLK2

        SETF DATA

        SETF /A0 /A1 /A2 /A3
        CLOCKF CLK1 CLK2
```

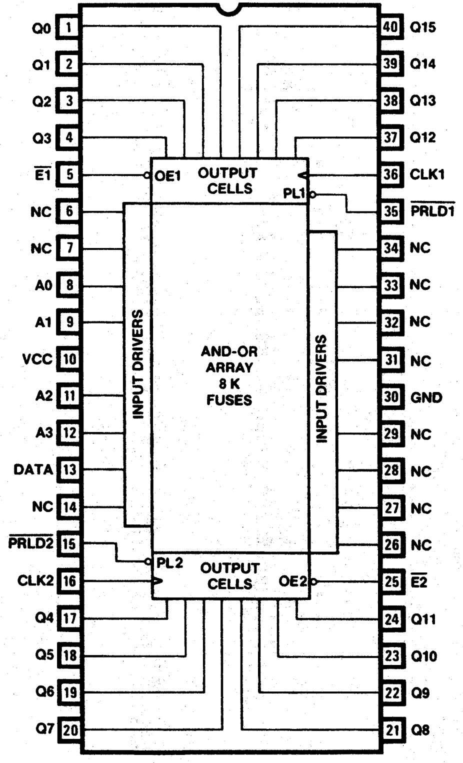
Simulation Results

```
Page :  1
       g g cgcgcg cgcgcgcgcg cgcgcgcgcg cgcgcgc
 Q0    XXXXLLLLLL LLLLLLLLLL LLLLLLLLLL LLLLLLH
 Q1    XXXXXXLLLL LLLLLLLLLL LLLLLLLLLL LLLLLLL
 Q2    XXXXXXXXLL LLLLLLLLLL LLLLLLLLLL LLLLLLL
 Q3    XXXXXXXXXX LLLLLLLLLL LLLLLLLLLL LLLLLLL
 Q4    XXXXXXXXXX XXLLLLLLLL LLLLLLLLLL LLLLLLL
 Q5    XXXXXXXXXX XXXXLLLLLL LLLLLLLLLL LLLLLLL
 Q6    XXXXXXXXXX XXXXXXLLLL LLLLLLLLLL LLLLLLL
 Q7    XXXXXXXXXX XXXXXXXXLL LLLLLLLLLL LLLLLLL
 Q8    XXXXXXXXXX XXXXXXXXXX LLLLLLLLLL LLLLLLL
 Q9    XXXXXXXXXX XXXXXXXXXX XXLLLLLLLL LLLLLLL
 Q10   XXXXXXXXXX XXXXXXXXXX XXXXLLLLLL LLLLLLL
 Q11   XXXXXXXXXX XXXXXXXXXX XXXXXXLLLL LLLLLLL
 Q12   XXXXXXXXXX XXXXXXXXXX XXXXXXXXLL LLLLLLL
 Q13   XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX LLLLLLL
 Q14   XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXLLLLL
 Q15   XXXXXXXXXX XXXXXXXXXX XXXXXXXXXX XXXXLLL
 A0    XXLLLHHLLH HLLHHLLHHL LHHLLHHLLH HLLHHLL
 A1    XXLLLLLHHH HLLLLHHHHL LLLHHHHLLL LHHHHLL
 A2    XXLLLLLLLL LHHHHHHHHL LLLLLLLHHH HHHHHLL
 A3    XXLLLLLLLL LLLLLLLLLH HHHHHHHHHH HHHHHLL
 DATA  LLLLLLLLLL LLLLLLLLLL LLLLLLLLLL LLLLLHH
```

TB01780M

XPLOT Output

```
PALASM XPLOT, V2.06 - BETA RELEASE
(C) - COPYRIGHT MONLITHIC MEMORIES INC., 1984

Title    : 16-BIT Addressable Register
Pattern  : ADREG16.PDS
Revision : A
Author   : John Birkner
Company  : Monolithic Memories Inc
Date     : 2/11/85

PAL32R16
ADREG16

             111111 11112222 22222233 33333333 44444444 44555555 55556
    01234567 89012345 67890123 45678901 23.56789 01234567 89012345 67890

  0 -------- -------- -------- -------- -------- -------- -------- -X---
  1 -------- -------- -------- -------- -------- -------- -------- -----
  2 X------- -------- -------- -------- -------- -------- -------- -----
  3 ----X--- -------- -------- -------- -------- -------- -------- -----
  4 -X---X-- X------- -------- -------- -------- -------- -------- X---X
  5 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXX
  6 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXX
  7 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXX
  8 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXX
  9 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXX
 10 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXX
 11 -X---X-- X------- -------- -------- -------- -------- -------- -X--X
 12 ----X--- -------- -------- -------- -------- -------- -------- --X--
 13 X------- -------- -------- -------- -------- -------- -------- --X--
 14 -------- -------- -------- -------- -------- -------- -------- --X--
 15 -------- -------- -------- -------- -------- -------- -------- X-X--

 16 -------- -------- -------- -------- -------- -------- ------X- -X---
 17 -------- -------- -------- -------- -------- -------- ------X- ----X
 18 X------- -------- -------- -------- -------- -------- ------X- -----
 19 ----X--- -------- -------- -------- -------- -------- ------X- -----
 20 -X---X-- X------- -------- -------- -------- -------- -------- X----
 21 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXX
 22 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXX
 23 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXX
 24 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXX
 25 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXX
 26 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXX
 27 -X---X-- X------- -------- -------- -------- -------- -------- -X---
 28 ----X--- -------- -------- -------- -------- -------- --X----- -----
 29 X------- -------- -------- -------- -------- -------- --X----- -----
 30 -------- -------- -------- -------- -------- -------- --X----- ----X
 31 -------- -------- -------- -------- -------- -------- --X----- X----

 32 -------- -------- -------- -------- -------- ------X- -------- -X---
 33 -------- -------- -------- -------- -------- ------X- -------- -----
 34 -X------ -------- -------- -------- -------- ------X- -------- -----
 35 -----X-- -------- -------- -------- -------- ------X- -------- -----
 36 X---X--- X------- -------- -------- -------- -------- -------- X---X
 37 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXX
 38 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXX
 39 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXX
 40 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXX
 41 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXX
 42 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXX
 43 X---X--- X------- -------- -------- -------- -------- -------- -X--X
 44 -----X-- -------- -------- -------- -------- --X----- -------- -----
 45 -X------ -------- -------- -------- -------- --X----- -------- -----
 46 -------- -------- -------- -------- -------- --X----- -------- -----
 47 -------- -------- -------- -------- -------- --X----- -------- X----

 48 -------- -------- -------- -------- ------X- -------- -------- -X---
 49 -------- -------- -------- -------- ------X- -------- -------- ----X
 50 -X------ -------- -------- -------- ------X- -------- -------- -----
 51 -----X-- -------- -------- -------- ------X- -------- -------- -----
 52 X---X--- X------- -------- -------- -------- -------- -------- X----
 53 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXX
 54 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXX
 55 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXX
 56 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXX
 57 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXX
 58 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXX
 59 X---X--- X------- -------- -------- -------- -------- -------- -X---
 60 -----X-- -------- -------- -------- --X----- -------- -------- -----
 61 -X------ -------- -------- -------- --X----- -------- -------- -----
 62 -------- -------- -------- -------- --X----- -------- -------- ----X
 63 -------- -------- -------- -------- --X----- -------- -------- X----

 64 -------- -------- -------- ------X- -------- -------- -------- -X---
 65 -------- -------- -------- ------X- -------- -------- -------- -----
 66 X------- -------- -------- ------X- -------- -------- -------- -----
 67 -----X-- -------- -------- ------X- -------- -------- -------- -----
 68 -X--X--- X------- -------- -------- -------- -------- -------- X---X
 69 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXX
 70 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXX
 71 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXX
 72 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXX
 73 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXX
 74 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXX
 75 -X--X--- X------- -------- -------- -------- -------- -------- -X--X
 76 -----X-- -------- -------- --X----- -------- -------- -------- -----
 77 X------- -------- -------- --X----- -------- -------- -------- -----
 78 -------- -------- -------- --X----- -------- -------- -------- -----
 79 -------- -------- -------- --X----- -------- -------- -------- X----

 80 -------- -------- ------X- -------- -------- -------- -------- -X---
 81 -------- -------- ------X- -------- -------- -------- -------- ----X
 82 X------- -------- ------X- -------- -------- -------- -------- -----
 83 -----X-- -------- ------X- -------- -------- -------- -------- -----
 84 -X--X--- X------- -------- -------- -------- -------- -------- X----
 85 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXX
 86 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXX
 87 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXX
 88 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXX
 89 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXX
 90 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXX
 91 -X--X--- X------- -------- -------- -------- -------- -------- -X---
 92 -----X-- -------- --X----- -------- -------- -------- -------- -----
 93 X------- -------- --X----- -------- -------- -------- -------- -----
 94 -------- -------- --X----- -------- -------- -------- -------- ----X
 95 -------- -------- --X----- -------- -------- -------- -------- X----

 96 -------- ------X- -------- -------- -------- -------- -------- -X---
 97 -------- ------X- -------- -------- -------- -------- -------- -----
 98 -X------ ------X- -------- -------- -------- -------- -------- -----
 99 ----X--- ------X- -------- -------- -------- -------- -------- -----
100 X----X-- X------- -------- -------- -------- -------- -------- X---X
101 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXX
102 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXX
103 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXX
104 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXX
105 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXX
106 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXX
107 X----X-- X------- -------- -------- -------- -------- -------- -X--X
108 ----X--- --X----- -------- -------- -------- -------- -------- -----
109 -X------ --X----- -------- -------- -------- -------- -------- -----
110 -------- --X----- -------- -------- -------- -------- -------- -----
111 -------- --X----- -------- -------- -------- -------- -------- X----

112 ------X- -------- -------- -------- -------- -------- -------- -X---
113 ------X- -------- -------- -------- -------- -------- -------- ----X
114 -X----X- -------- -------- -------- -------- -------- -------- -----
115 ----X-X- -------- -------- -------- -------- -------- -------- -----
116 X----X-- X------- -------- -------- -------- -------- -------- X----
117 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXX
118 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXX
119 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXX
120 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXX
121 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXX
122 XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXX
123 X----X-- X------- -------- -------- -------- -------- -------- -X---
124 --X-X--- -------- -------- -------- -------- -------- -------- -----
125 -XX----- -------- -------- -------- -------- -------- -------- -----
126 --X----- -------- -------- -------- -------- -------- -------- ----X
127 --X----- -------- -------- -------- -------- -------- -------- X----


     OUTPUT PINS:       111222223334
                    1234789012347890
  POLARITY FUSE:    ----------------


     OUTPUT  BANK:   4-40    17-24
       FLUSH FUSE:     X        X

TOTAL FUSES BLOWN:  5008
```



TB01790M

State Machine Design Example

Figure 1 illustrates a simple traffic intersection consisting of two one-way streets, direction 1 and direction 2. Each direction has a signal consisting of red, yellow, and green lamps which are activated with appropriately named active high signals. Also each direction has a sensor which provides an active high signal indicating the presence of an oncoming vehicle. Our controller is to manage this intersection with the sensors as inputs and the lamps as outputs, as shown in Figure 2.

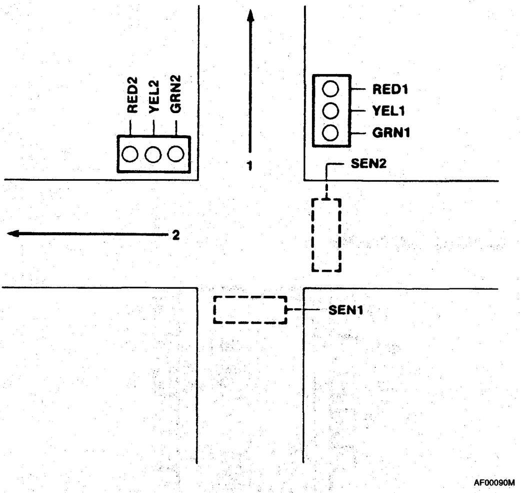


Figure 1. Traffic Intersection

Figure 2 also includes the system clock and an initialize (or reset) signal, which drives the controller to a predefined initial state. This raises two important issues in designing sequential logic with PAL devices. First, all circuit implementations of sequential logic with PAL devices are totally synchronous. This implies that all state variables (flip-flops) change at the same time, precisely after the rising edge of the clock. Second, PAL sequential logic designs should include a means for initialization to implement test programs and ensure reliable circuit operation. The specifics of the controller operations are detailed with a state diagram shown in Figure 3.

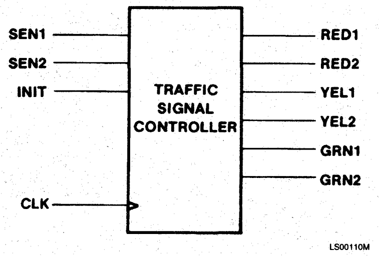


Figure 2

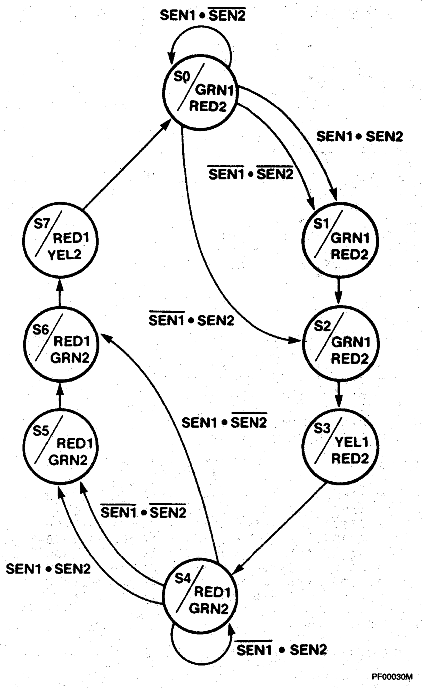

SEN1 • $\overline{\text{SEN2}}$
S0 GRN1 RED2
SEN1 • SEN2
$\overline{\text{SEN1}}$ • $\overline{\text{SEN2}}$
S1 GRN1 RED2
$\overline{\text{SEN1}}$ • SEN2
S2 GRN1 RED2
S3 YEL1 RED2
S7 RED1 YEL2
S6 RED1 GRN2
S5 RED1 GRN2
SEN1 • $\overline{\text{SEN2}}$
$\overline{\text{SEN1}}$ • $\overline{\text{SEN2}}$
SEN1 • SEN2
S4 RED1 GRN2
$\overline{\text{SEN1}}$ • SEN2
PF00030M

Figure 3. State Diagram — Traffic Signal Controller

Each circle in Figure 3 represents a stable state, i.e. an output configuration lasting at least one clock cycle. Inside the circles is the name of the state (S0 – S7) and the outputs associated with that state. For the sake of simplicity in the state diagram, the transitions involving INIT are omitted; INIT simply drives the circuit to S0 from any state, regardless of other inputs.

Since RED1 = /RED2, RED1 is implemented with one flip-flop and RED2 with an external inverter.

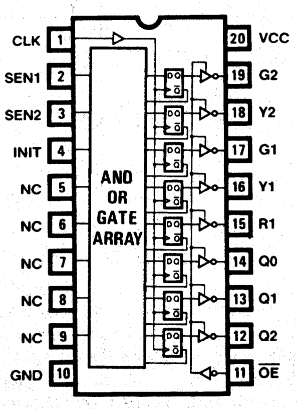
PAL Device Design Specification

```
TITLE       TRAFFIC SIGNAL CONTROLLER
PATTERN     TRAFFIC1.PDS
REVISION    A
AUTHOR      KELVIN CHOW
COMPANY     MONOLITHIC MEMORIES INC., SANTA CLARA
DATE        2/28/85

CHIP TRAFFIC PAL16RP8

CLK SEN1 SEN2 INIT NC NC NC NC NC GND
/OE Q2 Q1 Q0 R1 Y1 G1 Y2 G2 VCC

STRING I1 ' /SEN1*/SEN2*/INIT '
STRING I2 ' /SEN1*SEN2*/INIT  '
STRING I3 ' SEN1*/SEN2*/INIT  '
STRING I4 ' SEN1*SEN2*/INIT   '
STRING I5 ' INIT              '

STATE

S0      = BIN[4](R1,Y1,G1,Y2,G2)
S1      = BIN[4](R1,Y1,G1,Y2,G2)
S2      = BIN[4](R1,Y1,G1,Y2,G2)
S3      = BIN[8](R1,Y1,G1,Y2,G2)
S4      = BIN[17](R1,Y1,G1,Y2,G2)
S5      = BIN[17](R1,Y1,G1,Y2,G2)
S6      = BIN[17](R1,Y1,G1,Y2,G2)
S7      = BIN[18](R1,Y1,G1,Y2,G2)

EQUATIONS

S0      = I1*S1 + I2*S2 + I3*S0 + I4*S1 + I5*S0
S1      = I1*S2 + I2*S2 + I3*S2 + I4*S2 + I5*S0
S2      = I1*S3 + I2*S3 + I3*S3 + I4*S3 + I5*S0
S3      = I1*S4 + I2*S4 + I3*S4 + I4*S4 + I5*S0
S4      = I1*S5 + I2*S4 + I3*S6 + I4*S5 + I5*S0
S5      = I1*S6 + I2*S6 + I3*S6 + I4*S6 + I5*S0
S6      = I1*S7 + I2*S7 + I3*S7 + I4*S7 + I5*S0
S7      = I1*S0 + I2*S0 + I3*S0 + I4*S0 + I5*S0

SIMULATION

TRACE_ON CLK INIT SEN1 SEN2 R1 Y1 G1 Y2 G2

SETF OE INIT
CLOCKF
CLOCKF
CHECK /R1 /Y1 G1 /Y2 /G2

SETF /INIT /SEN1 /SEN2
CLOCKF

SETF SEN1 /SEN2
CLOCKF
CHECK /R1 /Y1 G1 /Y2 /G2

SETF /SEN1 SEN2
CLOCKF
CHECK /R1 /Y1 G1 /Y2 /G2

SETF SEN1 SEN2
CLOCKF
CHECK /R1 Y1 /G1 /Y2 /G2

SETF /SEN1 /SEN2
CLOCKF
CHECK R1 /Y1 /G1 /Y2 G2

SETF /SEN1 SEN2
CLOCKF
CHECK R1 G2

CLOCKF
CHECK R1 G2

CLOCKF
CHECK R1 /Y1 /G1 Y2 /G2

CLOCKF
CHECK /R1 /Y1 G1 /Y2 /G2

CLOCKF
CLOCKF
CLOCKF
CLOCKF

; This simulation was done using the alpha release version
; of Palasm2 software.
```

Simulation Results

```
      g  c  cg c g cg cg cg  cg c c c  c c c  c
CLK  XXHLHHLLHL LHLLHLLHLL HLLHLHLHLH LHLHLHHL
INIT HHHHHHHLLL LLLLLLLLLL LLLLLLLLLL LLLLLLLL
SEN1 XXXXXXXLLL HHHLLLHHHL LLLLLLLLLL LLLLLLLL
SEN2 XXXXXXXLLL LLLHHHHHHL LLHHHHHHHH HHHHHHHH
R1   XXXXXLLLLL LLLLLLLLLL LHHHHHHHHH LLLLLLHH
Y1   XXXXXLLLLL LLLLLLLLHH HLLLLLLLLL LLLLHHLL
G1   XXXXXHHHHH HHHHHHHHLL LLLLLLLLLL HHHHLLLL
Y2   XXXXXLLLLL LLLLLLLLLL LLLLLLLLHH LLLLLLLL
G2   XXXXXLLLLL LLLLLLLLLL LHHHHHHHLL LLLLLLHH
```

Logic Symbol



TB01810M

State Machine Design Example

A typical control logic problem is the memory-to-processor handshake on memory transfer used in many computer architectures. The processor makes a transfer request by activating a request line (REQ) and specifies a read or write operation on a Read/Write line (R/W).

During a read operation, the processor waits for a Data Available signal at which time the data bus is sampled and the request line lowered, thus completing the cycle. During a write operation, the processor places data on the bus and waits for a Write Complete signal after the write cycle is finished. Upon write complete, the request line is lowered, hence completing the cycle. Table 1 shows the state assignments and the appropriate outputs. The state diagram is shown in Figure 1. Also the handshaking operation is illustrated in the timing diagram of Figure 2.

The memory-board logic to implement this function may be designed with gates and edge-triggered flip-flops as shown in Figure 3. This particular design would require about five SSI/MSI packages, but the same design can be implemented by a single PAL16RP6. The PAL design specification using state equations is shown on the next page.

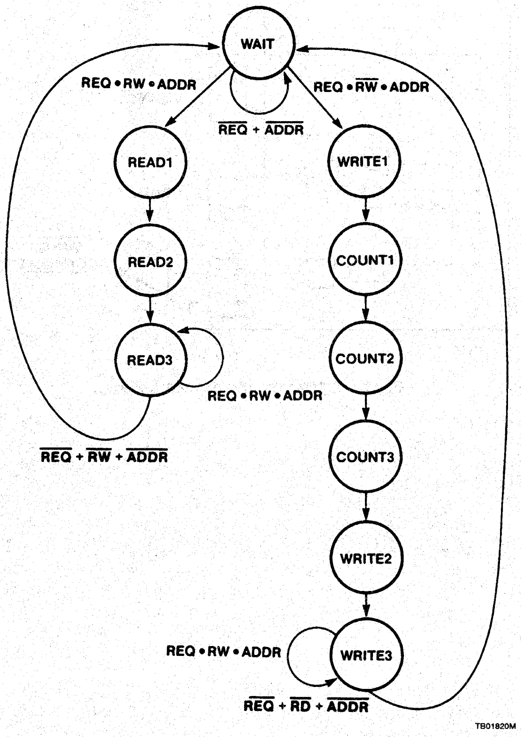

WAIT
REQ • RW • ADDR
REQ • $\overline{RW}$ • ADDR
$\overline{REQ}$ + $\overline{ADDR}$
READ1
READ2
READ3
REQ • RW • ADDR
$\overline{REQ}$ + $\overline{RW}$ + $\overline{ADDR}$
WRITE1
COUNT1
COUNT2
COUNT3
WRITE2
WRITE3
REQ • RW • ADDR
$\overline{REQ}$ + $\overline{RD}$ + $\overline{ADDR}$
TB01820M

Figure 1. State Diagram – Memory Handshake Logic

STATE	DOUT	DA	WE	WC	C0	C1
WAIT	0	0	0	0	0	0
READ1	1	0	0	0	0	0
READ2	1	1	0	0	0	0
READ3	0	0	0	0	0	0
COUNT1	0	0	1	0	1	0
COUNT2	0	0	1	0	0	1
COUNT3	0	0	1	0	1	1
WRITE1	0	0	1	0	0	0
WRITE2	0	0	1	1	0	0
WRITE3	0	0	0	1	0	0

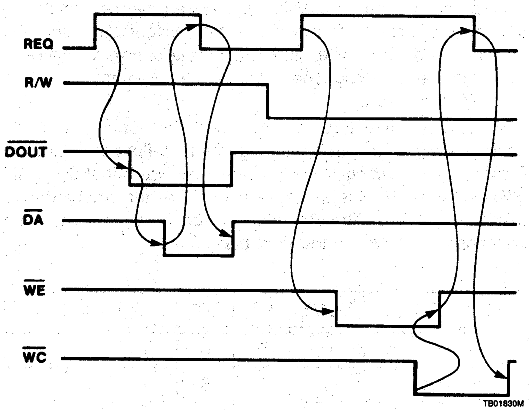


Figure 2. Memory Handshake Timing

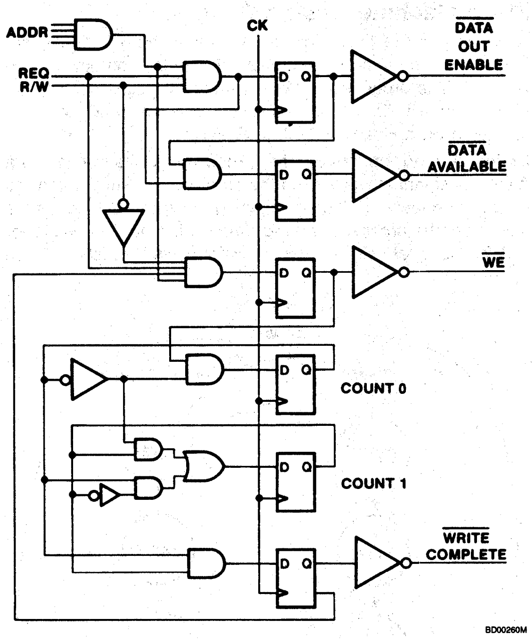


Figure 3. Memory Handshake Logic

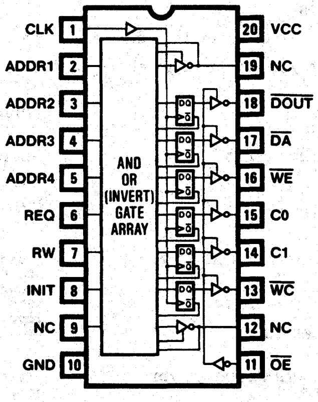
PAL Device Design Specification

```
TITLE          MEMORY HANDSHAKE LOGIC
PATTERN        MEMORY1.PDS
REVISION       A
AUTHOR         KELVIN CHOW
COMPANY        MONOLITHIC MEMORIES INC., SANTA CLARA, CA
DATE           2/28/85

CHIP MEMORY PAL16RP6

CLK ADDR1 ADDR2 ADDR3 ADDR4 REQ RW INIT NC GND
/OE NC /WC C1 C0 /WE /DA /DOUT NC VCC

STRING I1 ' REQ*RW*ADDR1*ADDR2*ADDR3*ADDR4*/INIT '
STRING I2 ' REQ*/RW*ADDR1*ADDR2*ADDR3*ADDR4*/INIT'
STRING I3 ' (/REQ+/ADDR1+/ADDR2+/ADDR3+/ADDR4) * /INIT '
STRING I4 ' (/REQ+/RW+/ADDR1+/ADDR2+/ADDR3+/ADDR4) * /INIT '
STRING I5 ' INIT '

STATE

WAIT    = BIN[0](DOUT,DA,WE,WC,C0,C1)
READ1   = BIN[32](DOUT,DA,WE,WC,C0,C1)
READ2   = BIN[48](DOUT,DA,WE,WC,C0,C1)
READ3   = BIN[0](DOUT,DA,WE,WC,C0,C1)
WRITE1  = BIN[8](DOUT,DA,WE,WC,C0,C1)
COUNT1  = BIN[10](DOUT,DA,WE,WC,C0,C1)
COUNT2  = BIN[9](DOUT,DA,WE,WC,C0,C1)
COUNT3  = BIN[11](DOUT,DA,WE,WC,C0,C1)
WRITE2  = BIN[12](DOUT,DA,WE,WC,C0,C1)
WRITE3  = BIN[4](DOUT,DA,WE,WC,C0,C1)

EQUATIONS

WAIT    := I2*WRITE1 + I1*READ1 + I3*WAIT
         + I5*WAIT
READ1   := I4*READ2 + I3*READ2 + I2*READ2 + I1*READ2
         + I5*WAIT
READ2   := I4*READ3 + I3*READ3 + I2*READ3 + I1*READ3
         + I5*WAIT
READ3   := I1*READ3 + I4*WAIT
         + I5*WAIT
WRITE1  := I4*COUNT1 + I3*COUNT1 + I2*COUNT1 + I1*COUNT1
         + I5*WAIT
COUNT1  := I4*COUNT2 + I3*COUNT2 + I2*COUNT2 + I1*COUNT2
         + I5*WAIT
COUNT2  := I4*COUNT3 + I3*COUNT3 + I2*COUNT3 + I1*COUNT3
         + I5*WAIT
COUNT3  := I4*WRITE2 + I3*WRITE2 + I2*WRITE2 + I1*WRITE2
         + I5*WAIT
WRITE2  := I4*WRITE3 + I3*WRITE3 + I2*WRITE3 + I1*WRITE3
         + I5*WAIT
WRITE3  := I1*WRITE3 + I4*WAIT
         + I5*WAIT

SIMULATION

TRACE_ON REQ RW CLK /DOUT /DA /WE /WC

SETF INIT /REQ OE RW ADDR1 ADDR2 ADDR3 ADDR4
CLOCKF CLK
CHECK /DOUT

SETF REQ /INIT
CLOCKF
CLOCKF
CLOCKF
CLOCKF
CHECK DOUT DA

SETF /REQ
CLOCKF
CHECK /DOUT /DA

SETF REQ /RW
CLOCKF
CLOCKF
CLOCKF
CLOCKF
CLOCKF
CLOCKF

SETF /REQ
CLOCKF
CLOCKF
CLOCKF

; This simulation was done using the alpha release version of
; Palasm2 software.
```

Simulation Results

```
Page :  1
      g  cg   c   c c cg  cg   c c c c  c cg c c
 CLK  XXHLLHHLHH LHLHLLHHLL HHLHLHLHLH LHLLHLHL
 REQ  LLLLHHHHHH HHHHHLLLLH HHHHHHHHHH HHHLLLLL
 RW   HHHHHHHHHH HHHHHHHHHL LLLLLLLLLL LLLLLLLL
/DOUT XXXHHHLLLL LLLLLLLHHH HHHHHHHHHH HHHHHHHH
/DA   XXXHHHHHHL LLLLLLLHHH HHHHHHHHHH HHHHHHHH
/WE   XXXHHHHHHH HHHHHHHHHH HLLLLLLLLL LLHHHHHH
/WC   XXXXXXHHHH HHHHHHHHHH HHHHHHHHHH LLHHHHHH
```

Logic Symbol



TB01840M

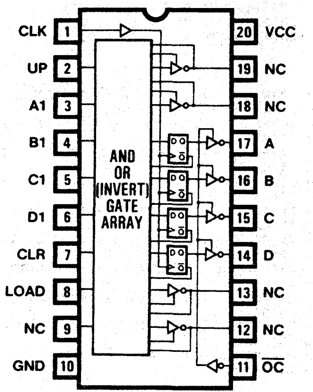
PAL Device Design Specification

```
Title      4Bit_Counter
Pattern    4cnt.pds
Revision   A
Author     Mehrnaz Hada
Company    Monolithic Memories Inc. Santa Clara, CA
Date       1/14/85

CHIP 4BitCounter PAL16RP4

CLK UP AI BI CI DI CLR LOAD NC GND
/OC NC NC D C B A NC NC VCC

EQUATIONS

A   := /A*/B*/C*/D*/UP*/LOAD*/CLR      ;When CLR=1, A=0.
     + /A* B* C* D* UP*/LOAD*/CLR      ;Else it will count
     +  A* B*   /D*/UP*/LOAD*/CLR      ;UP or DOWN.
     +  A*/B* C*    UP*/LOAD*/CLR
     +  A*   /C*    UP*/LOAD*/CLR
     +  A*        D*/UP*/LOAD*/CLR
     +                 LOAD*/CLR* AI  ;New value is loaded
                                       ;when LOAD=1, CLR=0.

B   :=    /B*/C*/D*/UP*/LOAD*/CLR      ;When CLR=1, B=0.
     +    /B* C* D* UP*/LOAD*/CLR      ;Else it will count.
     +     B* C*/D*    /LOAD*/CLR
     +     B*/C*    UP*/LOAD*/CLR
     +     B*     D*/UP*/LOAD*/CLR
     +                 LOAD*/CLR* BI  ;New value is loaded
                                       ;when LOAD=1, CLR=0.

C   :=       /C*/D*/UP*/LOAD*/CLR      ;When CLR=1, C=0.
     +       /C* D* UP*/LOAD*/CLR      ;Else it will count.
     +        C*/D* UP*/LOAD*/CLR
     +        C* D*/UP*/LOAD*/CLR
     +                 LOAD*/CLR* CI  ;New value is loaded
                                       ;when LOAD=1, CLR=0.

D   :=          /D*    /LOAD*/CLR      ;Count
     +                 LOAD*/CLR* DI  ;New value is loaded
                                       ;when LOAD=1, CLR=0.

SIMULATION

TRACE_ON  AI BI CI DI LOAD CLR UP A B C D

SETF LOAD /CLR AI BI CI DI OC             ;Load all registers
CLOCKF CLK                                 ;to HIGH and count up

SETF CLR                                   ;Clear all registers
CLOCKF CLK

SETF /CLR UP /LOAD                         ;Start Counting up

 FOR I:= 1 TO 16 DO                       ;Count up 16 clock
  BEGIN                                   ;cycles
    CLOCKF CLK
  END

SETF LOAD /CLR /UP AI BI CI DI            ;Load all registers
CLOCKF                                    ;to HIGH and count
SETF /LOAD                                ;down
  FOR I:= 1 TO 16 DO                      ;Count down 16 clock
   BEGIN                                  ;cycles
     CLOCKF CLK
   END

SETF LOAD CLR AI /BI CI /DI               ;Test setting LOAD
CLOCKF CLK                                ;and CLR on at the
                                          ;same time.
SETF /OC

;The 4-bit counter counts up or down and has the clear and
;load capability. The clear operation overrides count and
;load. The counter counts up when CLR=low, LOAD=low, and
;UP=high. It counts down whenever CLR=low, LOAD=low, and
```

Simulation Results

```
Page :  1
      g  cgcgc c  c c c c c  c c c c c  c c c cgc
 AI   HHHHHHHHHH HHHHHHHHHH HHHHHHHHHH HHHHHHHHHH
 BI   HHHHHHHHHH HHHHHHHHHH HHHHHHHHHH HHHHHHHHHH
 CI   HHHHHHHHHH HHHHHHHHHH HHHHHHHHHH HHHHHHHHHH
 DI   HHHHHHHHHH HHHHHHHHHH HHHHHHHHHH HHHHHHHHHH
 LOAD HHHHHHLLLL LLLLLLLLLL LLLLLLLLLL LLLLLLLLHH
 CLR  LLLLHHLLLL LLLLLLLLLL LLLLLLLLLL LLLLLLLLLL
 UP   XXXXXXHHHH HHHHHHHHHH HHHHHHHHHH HHHHHHHHLL
 A    XXXHHLLLLL LLLLLLLLLL LHHHHHHHHH HHHHHLLHHH
 B    XXXHHLLLLL LLLHHHHHHH HLLLLLLLLH HHHHHHHLLH
 C    XXXHHLLLLH HHHLLLLHHH HLLLLHHHHL LLLHHHHLLH
 D    XXXHHLLHHL LHHLLHHLLH HLLHHLLHHL LHHLLHHLLH
```

Logic Symbol



TB01850M

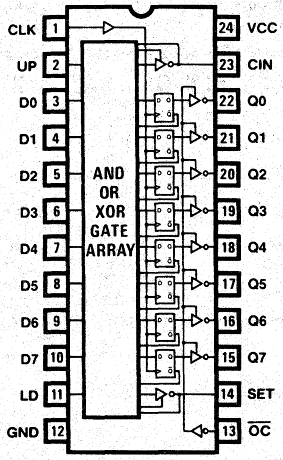
PAL Device Design Specification

```
Title     8Count
Pattern   8count.pds
Revision  A
Author    Mehrnaz Hada
Company   Monolithic Memories Inc. Santa Clara, CA
Date      1/15/85

;This 8-bit up/down counter has the hold and load
;capabilities. It sets all the outputs high if SET=high.
;It loads new value when SET=low and LOAD=high. Else it
;counts up if UP=high and counts down if UP=low.

CHIP 8BitCounter PAL20X8

CLK UP D0 D1 D2 D3 D4 D5 D6 D7 LD GND
/OC SET Q7 Q6 Q5 Q4 Q3 Q2 Q1 Q0 CIN VCC

EQUATIONS

/Q0 := /SET* LD*/D0                          ;Load D0
    +  /SET*/LD*/Q0                          ;Hold
   :+: /SET*/LD*CIN* UP                      ;Increment
    +  /SET*/LD*CIN*/UP                      ;Decrement

/Q1 := /SET* LD*/D1                          ;Load D1
    +  /SET*/LD*/Q1                          ;Hold
   :+: /SET*/LD*CIN* UP* Q0                  ;Increment
    +  /SET*/LD*CIN*/UP*/Q0                  ;Decrement

/Q2 := /SET* LD*/D2                          ;Load D2
    +  /SET*/LD*/Q2                          ;Hold
   :+: /SET*/LD*CIN* UP* Q0* Q1              ;Increment
    +  /SET*/LD*CIN*/UP*/Q0*/Q1              ;Decrement

/Q3 := /SET* LD*/D3                          ;Load D3
    +  /SET*/LD*/Q3                          ;Hold
   :+: /SET*/LD*CIN* UP* Q0* Q1* Q2          ;Increment
    +  /SET*/LD*CIN*/UP*/Q0*/Q1*/Q2          ;Decrement

/Q4 := /SET* LD*/D4                          ;Load D4
    +  /SET*/LD*/Q4                          ;Hold
   :+: /SET*/LD*CIN* UP* Q0* Q1* Q2* Q3      ;Increment
    +  /SET*/LD*CIN*/UP*/Q0*/Q1*/Q2*/Q3      ;Decrement

/Q5 := /SET* LD*/D5                          ;Load D5
    +  /SET*/LD*/Q5                          ;Hold
   :+: /SET*/LD*CIN* UP* Q0* Q1* Q2* Q3
        * Q4                                 ;Increment
    +  /SET*/LD*CIN*/UP*/Q0*/Q1*/Q2*/Q3
        */Q4                                 ;Decrement

/Q6 := /SET* LD*/D6                          ;Load D6
    +  /SET*/LD*/Q6                          ;Hold
   :+: /SET*/LD*CIN* UP* Q0* Q1* Q2* Q3
        * Q4* Q5                             ;Increment
    +  /SET*/LD*CIN*/UP*/Q0*/Q1*/Q2*/Q3
        */Q4*/Q5                             ;Decrement

/Q7 := /SET* LD*/D7                          ;Load D7
    +  /SET*/LD*/Q7                          ;Hold
   :+: /SET*/LD*CIN* UP* Q0* Q1* Q2* Q3
        * Q4* Q5* Q6                         ;Increment
    +  /SET*/LD*CIN*/UP*/Q0*/Q1*/Q2*/Q3
        */Q4*/Q5*/Q6                         ;Decrement

SIMULATION

TRACE_ON  SET LD CIN UP
          D0 D1 D2 D3 D4 D5 D6 D7
          Q0 Q1 Q2 Q3 Q4 Q5 Q6 Q7

SETF OC SET
CLOCKF CLK
CHECK Q7 Q6 Q5 Q4 Q3 Q2 Q1 Q0                ;All outputs high
SETF /SET UP CIN /LD                         ;Counting up

FOR I:=1 TO 9 DO
 BEGIN
  CLOCKF CLK
  IF I=8 THEN                                ;Checking after 8
   BEGIN                                     ;clock pulses
    CHECK /Q7 /Q6 /Q5 /Q4 /Q3 Q2 Q1 Q0
   END
 END

SETF /CIN                                    ;Holding
CLOCKF CLK
CLOCKF CLK                                   ;The outputs hold to
                                             ;their values

SETF /UP CIN                                 ;Counting down

SETF LD /D7 D6 /D5 D4 /D3 D2 /D1 D0          ;Loading some data
CLOCKF CLK
CHECK /Q7 Q6 /Q5 Q4 /Q3 Q2 /Q1 Q0            ;Checking the output
                                             ;for the loaded data
SETF /LD UP                                  ;Counting up after
FOR I:=1 TO 5 DO                             ;removing HOLD, count
 BEGIN                                       ;up 3 cycles, count
  CLOCKF CLK                                 ;down for 2 cycles.
  IF I=3 THEN
   BEGIN
    SETF /UP
   END
 END
```

Simulation Results

```
 Page :  1
     g  cgc c c  c c c c c  cgc cgcgc  c cgc c
 SET HHHHLLLLLL LLLLLLLLLL LLLLLLLLLL LLLLLLLL
 LD  XXXXLLLLLL LLLLLLLLLL LLLLLLHHLL LLLLLLLL
 CIN XXXXHHHHHH HHHHHHHHHH HHLLLLHHHH HHHHHHHH
 UP  XXXXHHHHHH HHHHHHHHHH HHHHHHLLHH HHHHLLLL
 D0  XXXXXXXXXX XXXXXXXXXX XXXXXXHHHH HHHHHHHH
 D1  XXXXXXXXXX XXXXXXXXXX XXXXXXLLLL LLLLLLLL
 D2  XXXXXXXXXX XXXXXXXXXX XXXXXXHHHH HHHHHHHH
 D3  XXXXXXXXXX XXXXXXXXXX XXXXXXLLLL LLLLLLLL
 D4  XXXXXXXXXX XXXXXXXXXX XXXXXXHHHH HHHHHHHH
 D5  XXXXXXXXXX XXXXXXXXXX XXXXXXLLLL LLLLLLLL
 D6  XXXXXXXXXX XXXXXXXXXX XXXXXXHHHH HHHHHHHH
 D7  XXXXXXXXXX XXXXXXXXXX XXXXXXLLLL LLLLLLLL
 Q0  XXXHHLLHHL LHHLLHHLLH HLLLLLLHHL LHHLLHHL
 Q1  XXXHHLLLLH HHHLLLLHHH HLLLLLLLLH HHHLLHHH
 Q2  XXXHHLLLLL LLLHHHHHHH HLLLLLLHHH HHHLLHHH
 Q3  XXXHHLLLLL LLLLLLLLLL LHHHHHHLLL LLLHHLLL
 Q4  XXXHHLLLLL LLLLLLLLLL LLLLLLLHHH HHHHHHHH
 Q5  XXXHHLLLLL LLLLLLLLLL LLLLLLLLLL LLLLLLLL
 Q6  XXXHHLLLLL LLLLLLLLLL LLLLLLLHHH HHHHHHHH
 Q7  XXXHHLLLLL LLLLLLLLLL LLLLLLLLLL LLLLLLLL
```

Logic Symbol



TB01860M

9-Bit Counter

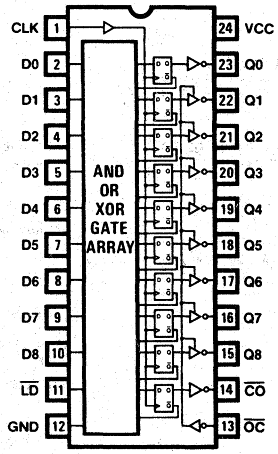
PAL Device Design Specification

```
Title    9BitCounter
Pattern  9BitCnt.pds
Revision A
Author   Mehrnaz Hada
Company  Monolithic Memories Inc., Santa Clara, CA
Date     1/28/85

;The 9-bit synchronous counter has parallel load, increment,
;and hold capabilities. The carry out pin (/CO) shows how to
;implement a carry out using a register by anticipated one
;count before the terminal count if counting and the terminal
;count if loading.

;Operations Table

;     /OC   CLK   /LD   D8-D0   Q8-Q0      Operation
;     ----------------------------------------------
;      H     X     X      X       Z        HI-Z
;      L     L     X      X       Q        Hold
;      L     C     L      D       D        Load
;      L     C     H      X     Q PLUS 1   Increment
;     ----------------------------------------------

CHIP 9BitCounter PAL20X10

CLK  D0 D1 D2 D3 D4 D5 D6 D7 D8 /LD GND
/OC /CO Q8 Q7 Q6 Q5 Q4 Q3 Q2 Q1  Q0 VCC

EQUATIONS

/Q0 :=  /LD*/Q0                                   ;Hold Q0
     +   LD*/D0                                   ;Load D0 (LSB)
    :+: /LD                                       ;Count

/Q1 :=  /LD*/Q1                                   ;Hold Q1
     +   LD*/D1                                   ;Load D1
    :+: /LD* Q0                                   ;Count

/Q2 :=  /LD*/Q2                                   ;Hold Q2
     +   LD*/D2                                   ;Load D2
    :+: /LD* Q0* Q1                               ;Count

/Q3 :=  /LD*/Q3                                   ;Hold Q3
     +   LD*/D3                                   ;Load D3
    :+: /LD* Q0* Q1* Q2                           ;Count

/Q4 :=  /LD*/Q4                                   ;Hold Q4
     +   LD*/D4                                   ;Load D4
    :+: /LD* Q0* Q1* Q2* Q3                       ;Count

/Q5 :=  /LD*/Q5                                   ;Hold Q5
     +   LD*/D5                                   ;Load D5
    :+: /LD* Q0* Q1* Q2* Q3* Q4                   ;Count

/Q6 :=  /LD*/Q6                                   ;Hold Q6
     +   LD*/D6                                   ;Load D6
    :+: /LD* Q0* Q1* Q2* Q3* Q4* Q5               ;Count

/Q7 :=  /LD*/Q7                                   ;Hold Q7
     +   LD*/D7                                   ;Load D7
    :+: /LD* Q0* Q1* Q2* Q3* Q4* Q5* Q6           ;Count

/Q8 :=  /LD*/Q8                                   ;Hold Q8
     +   LD*/D8                                   ;Load D8 (MSB)
    :+: /LD* Q0* Q1* Q2* Q3* Q4* Q5* Q6* Q7       ;Count

 CO :=  /LD*/Q0* Q1* Q2* Q3* Q4* Q5* Q6* Q7* Q8   ;Carry out (Anticipate
     +   LD* D0* D1* D2* D3* D4* D5* D6* D7* D8   ;Carry out (Anticipate

SIMULATION

TRACE_ON /OC /LD D8 D7 D6 D5 D4 D3 D2 D1 D0
         /CO Q8 Q7 Q6 Q5 Q4 Q3 Q2 Q1 Q0

SETF OC LD /D8 /D7 /D6 /D5 /D4 /D3 /D2 /D1 /D0    ;Load
CLOCKF CLK

SETF /LD                                          ;Increment
CLOCKF CLK

SETF LD /D8 /D7 /D6 /D5 /D4 /D3 /D2 /D1 D0        ;Load
CLOCKF CLK

SETF /LD                                          ;Increment
CLOCKF CLK

SETF LD /D8 /D7 /D6 /D5 /D4 /D3 D2 D1 D0          ;Load
CLOCKF CLK

SETF /LD                                          ;Increment
CLOCKF CLK

SETF LD /D8 /D7 /D6 /D5 /D4 D3 D2 D1 D0           ;Load
CLOCKF CLK

SETF /LD                                          ;Increment
CLOCKF CLK

SETF LD /D8 /D7 /D6 /D5 D4 D3 D2 D1 D0            ;Load
CLOCKF CLK

SETF /LD                                          ;Increment
CLOCKF CLK

SETF LD /D8  D7  D6  D5 D4 D3 D2 D1 D0            ;Load
CLOCKF CLK

SETF /LD                                          ;Increment
CLOCKF CLK


;Function Table

;CLK /OC /LD D8 D7 D6 D5 D4 D3 D2 D1 D0 /CO
;            Q8 Q7 Q6 Q5 Q4 Q3 Q2 Q1 Q0

;             Data In        Data  Out
;  Control   DDDDDDDDD       QQQQQQQQQ
;CLK /OC /LD 876543210  /CO  876543210  Comment
;------------------------------------------------------
; C   L   L  LLLLLLLLL   H   LLLLLLLLL  Load
; C   L   H  XXXXXXXXX   H   LLLLLLLLH  Increment
; C   L   L  LLLLLLLLH   H   LLLLLLLLH  Load
; C   L   H  XXXXXXXXX   H   LLLLLLLHL  Increment
; C   L   L  LLLLLLLHH   H   LLLLLLLHH  Load
; C   L   H  XXXXXXXXX   H   LLLLLLHLL  Increment
; C   L   L  LLLLLLHHH   H   LLLLLLHHH  Load
; C   L   H  XXXXXXXXX   H   LLLLLHLLL  Increment
; C   L   L  LLLLLHHHH   H   LLLLLHHHH  Load
; C   L   H  XXXXXXXXX   H   LLLLHLLLL  Increment
; C   L   L  LLLLHHHHH   H   LLLLHHHHH  Load
; C   L   H  XXXXXXXXX   H   LLLHLLLLL  Increment
; C   L   L  LLLHHHHHH   H   LLLHHHHHH  Load
; C   L   H  XXXXXXXXX   H   LLHLLLLLL  Increment
; C   L   L  LLHHHHHHH   H   LLHHHHHHH  Load
; C   L   H  XXXXXXXXX   H   LHLLLLLLL  Increment
; C   L   L  LHHHHHHHH   H   LHHHHHHHH  Load
; C   L   H  XXXXXXXXX   H   HLLLLLLLL  Increment
; C   L   L  HHHHHHHHH   L   HHHHHHHHH  Load(Carry out)
; C   L   H  XXXXXXXXX   H   LLLLLLLLL  Increment(Roll over)
; C   L   L  HHHHHHHLL   H   HHHHHHHLL  Load
; C   L   H  XXXXXXXXX   H   HHHHHHHLH  Increment
; L   L   H  XXXXXXXXX   H   HHHHHHHLH  Hold
; C   L   H  XXXXXXXXX   H   HHHHHHHHL  Increment
; C   L   H  XXXXXXXXX   L   HHHHHHHHH  Increment(Carry out)
; C   L   H  XXXXXXXXX   H   LLLLLLLLL  Increment(Roll over)
; X   H   X  XXXXXXXXX   Z   ZZZZZZZZZ  Test HI-Z
;------------------------------------------------------
```

Simulation Results

```
Page :  1
    g  cgcgcgc gcgcgcgcgc gcgcg c
/OC LLLLLLLLLL LLLLLLLLLL LLLLLLL
/LD LLLLHHLLHH LLHHLLHHLL HHLLHHH
 D8 LLLLLLLLLL LLLLLLLLLL LLLLLLL
 D7 LLLLLLLLLL LLLLLLLLLL LLHHHHH
 D6 LLLLLLLLLL LLLLLLLLLL LLHHHHH
 D5 LLLLLLLLLL LLLLLLLLLL LLHHHHH
 D4 LLLLLLLLLL LLLLLLLLHH HHHHHHH
 D3 LLLLLLLLLL LLLLHHHHHH HHHHHHH
 D2 LLLLLLLLLL HHHHHHHHHH HHHHHHH
 D1 LLLLLLLLLL HHHHHHHHHH HHHHHHH
 D0 LLLLLLHHHH HHHHHHHHHH HHHHHHH
/CO XXXHHHHHHH HHHHHHHHHH HHHHHHH
 Q8 XXXLLLLLLL LLLLLLLLLL LLLLLHH
 Q7 XXXLLLLLLL LLLLLLLLLL LLLHHLL
 Q6 XXXLLLLLLL LLLLLLLLLL LLLHHLL
 Q5 XXXLLLLLLL LLLLLLLLLL LHHHHLL
 Q4 XXXLLLLLLL LLLLLLLHHH HLLHHLL
 Q3 XXXLLLLLLL LLLHHHHLLH HLLHHLL
 Q2 XXXLLLLLLL LHHLLHHLLH HLLHHLL
 Q1 XXXLLLLLLH HHHLLHHLLH HLLHHLL
 Q0 XXXLLHHHHL LHHLLHHLLH HLLHHLL
```

Logic Symbol



TB01870M

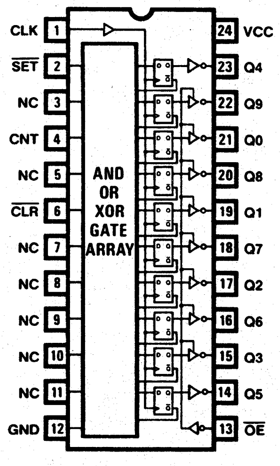
PAL Device Design Specification

```
Title      10Bit_Counter
Pattern    10count.pds
Revision   A
Author     Mehrnaz Hada
Company    Monolithic Memories Inc. Santa Clara, CA
Date       1/15/85

;The 10-bit counter increments on the rising edge of the
;clock input (CLK), if CNT input is high. The outputs are
;HIGH-Z when the enable line (/OE) is high and enabled
;when the enable line (/OE) is low. The counter is
;cleared (all lows) if CLR=HIGH.

CHIP 10BitCount PAL20RS10

CLK /SET NC CNT NC /CLR NC NC NC NC NC GND
/OE Q5 Q3 Q6 Q2 Q7 Q1 Q8 Q0 Q9 Q4 VCC

EQUATIONS

/Q0  := /SET *  CNT * /CLR * Q0          ;Toggle
      + /SET * /CNT * /CLR * /Q0         ;Hold
      + CLR                              ;CLR

/Q1  := /SET * CNT * /CLR * Q0 * Q1      ;Toggle
      + /SET * CNT * /CLR * /Q0 * /Q1    ;Toggle
      + /SET * /CNT * /CLR * /Q1         ;Hold
      + CLR                              ;CLR

/Q2  := /SET * CNT * /CLR * Q0 * Q1 *    ;Toggle
        Q2
      + /SET * CNT * /CLR * /Q0 * /Q2    ;Toggle
      + /SET * CNT * /CLR * /Q1 * /Q2    ;Toggle
      + /SET * /CNT * /CLR * /Q2         ;Hold
      + CLR                              ;CLR

/Q3  := /SET * CNT * /CLR * Q0 * Q1 *    ;Toggle
        Q2 * Q3
      + /SET * CNT * /CLR * /Q0 * /Q3    ;Toggle
      + /SET * CNT * /CLR * /Q1 * /Q3    ;Toggle
      + /SET * CNT * /CLR * /Q2 * /Q3    ;Toggle
      + /SET * /CNT * /CLR * /Q3         ;Hold
      + CLR                              ;CLR

/Q4  := /SET * CNT * /CLR * Q0 * Q1 *    ;Toggle
        Q2 * Q3 * Q4
      + /SET * CNT * /CLR * /Q0 * /Q4    ;Toggle
      + /SET * CNT * /CLR * /Q1 * /Q4    ;Toggle
      + /SET * CNT * /CLR * /Q2 * /Q4    ;Toggle
      + /SET * CNT * /CLR * /Q3 * /Q4    ;Toggle
      + /SET * /CNT * /CLR * /Q4         ;Hold
      + CLR                              ;CLR

/Q5  := /SET * CNT * /CLR * Q0 * Q1 *    ;Toggle
        Q2 * Q3 * Q4 * Q5
      + /SET * CNT * /CLR * /Q0 * /Q5    ;Toggle
      + /SET * CNT * /CLR * /Q1 * /Q5    ;Toggle
      + /SET * CNT * /CLR * /Q2 * /Q5    ;Toggle
      + /SET * CNT * /CLR * /Q3 * /Q5    ;Toggle
      + /SET * CNT * /CLR * /Q4 * /Q5    ;Toggle
      + /SET * /CNT * /CLR * /Q5         ;Hold
      + CLR                              ;CLR

/Q6  := /SET * CNT * /CLR * Q0 * Q1 *    ;Toggle
        Q2 * Q3 * Q4 * Q5 * Q6
      + /SET * CNT * /CLR * /Q0 * /Q6    ;Toggle
      + /SET * CNT * /CLR * /Q1 * /Q6    ;Toggle
      + /SET * CNT * /CLR * /Q2 * /Q6    ;Toggle
      + /SET * CNT * /CLR * /Q3 * /Q6    ;Toggle
      + /SET * CNT * /CLR * /Q4 * /Q6    ;Toggle
      + /SET * CNT * /CLR * /Q5 * /Q6    ;Toggle
      + /SET * /CNT * /CLR * /Q6          ;Hold
      + CLR                               ;CLR

/Q7  := /SET * CNT * /CLR * Q0 * Q1 *    ;Toggle
        Q2 * Q3 * Q4 * Q5 * Q6 * Q7
      + /SET * CNT * /CLR * /Q0 * /Q7    ;Toggle
      + /SET * CNT * /CLR * /Q1 * /Q7    ;Toggle
      + /SET * CNT * /CLR * /Q2 * /Q7    ;Toggle
      + /SET * CNT * /CLR * /Q3 * /Q7    ;Toggle
      + /SET * CNT * /CLR * /Q4 * /Q7    ;Toggle
      + /SET * CNT * /CLR * /Q5 * /Q7    ;Toggle
      + /SET * CNT * /CLR * /Q6 * /Q7    ;Toggle
      + /SET * /CNT * /CLR * /Q7         ;Hold
      + CLR                              ;CLR

/Q8  := /SET * CNT * /CLR * Q0 * Q1 *    ;Toggle
        Q2 * Q3 * Q4 * Q5 * Q6 * Q7 *
        Q8
      + /SET * CNT * /CLR * /Q0 * /Q8    ;Toggle
      + /SET * CNT * /CLR * /Q1 * /Q8    ;Toggle
      + /SET * CNT * /CLR * /Q2 * /Q8    ;Toggle
      + /SET * CNT * /CLR * /Q3 * /Q8    ;Toggle
      + /SET * CNT * /CLR * /Q4 * /Q8    ;Toggle
      + /SET * CNT * /CLR * /Q5 * /Q8    ;Toggle
      + /SET * CNT * /CLR * /Q6 * /Q8    ;Toggle
      + /SET * CNT * /CLR * /Q7 * /Q8    ;Toggle
      + /SET * /CNT * /CLR * /Q8         ;Hold
      + CLR                              ;CLR

/Q9  := /SET * CNT * /CLR * Q0 * Q1 *    ;Toggle
        Q2 * Q3 * Q4 * Q5 * Q6 * Q7 *
        Q8 * Q9
      + /SET * CNT * /CLR * /Q0 * /Q9    ;Toggle
      + /SET * CNT * /CLR * /Q1 * /Q9    ;Toggle
      + /SET * CNT * /CLR * /Q2 * /Q9    ;Toggle
      + /SET * CNT * /CLR * /Q3 * /Q9    ;Toggle
      + /SET * CNT * /CLR * /Q4 * /Q9    ;Toggle
      + /SET * CNT * /CLR * /Q5 * /Q9    ;Toggle
      + /SET * CNT * /CLR * /Q6 * /Q9    ;Toggle
      + /SET * CNT * /CLR * /Q7 * /Q9    ;Toggle
      + /SET * CNT * /CLR * /Q8 * /Q9    ;Toggle
      + /SET */CNT * /CLR * /Q9          ;Hold
      + CLR                              ;CLR

SIMULATION

TRACE_ON OE CLK SET CLR CNT Q0 Q1 Q2 Q3
         Q4 Q5 Q6 Q7 Q8 Q9

SETF OE /CLR SET                          ;Set high all the
CLOCKF CLK                                ;registers

SETF CLR                                  ;Clear all the
CLOCKF CLK                                ;registers

SETF /SET CNT /CLR                        ;Start counting

FOR I:= 1 TO 5 DO                         ;Count for five
  BEGIN                                   ;cycles. At the count
    CLOCKF CLK                            ;of four check for
                                          ;four on the output.
    IF I=4 THEN
      BEGIN
        CHECK /Q0 /Q1 Q2
      END
  END

SETF /CNT                                  ;Hold
CLOCKF CLK
```

Simulation Results

```
Page :  1
      g   cg cg c    c c c cg    c
 OE   HHHHHHHHHH HHHHHHHHHH H
 CLK  XXHLLHLLHL HLHLHLHLLH L
 SET  HHHHHHHLLL LLLLLLLLLL L
 CLR  LLLLHHHLLL LLLLLLLLLL L
 CNT  XXXXXXXHHH HHHHHHHHLL L
 Q0   XXXHHHLLLH HLLHHLLHHH H
 Q1   XXXHHHLLLL LHHHHLLLLL L
 Q2   XXXHHHLLLL LLLLLHHHHH H
 Q3   XXXHHHLLLL LLLLLLLLLL L
 Q4   XXXHHHLLLL LLLLLLLLLL L
 Q5   XXXHHHLLLL LLLLLLLLLL L
 Q6   XXXHHHLLLL LLLLLLLLLL L
 Q7   XXXHHHLLLL LLLLLLLLLL L
 Q8   XXXHHHLLLL LLLLLLLLLL L
 Q9   XXXHHHLLLL LLLLLLLLLL L
```

Logic Symbol

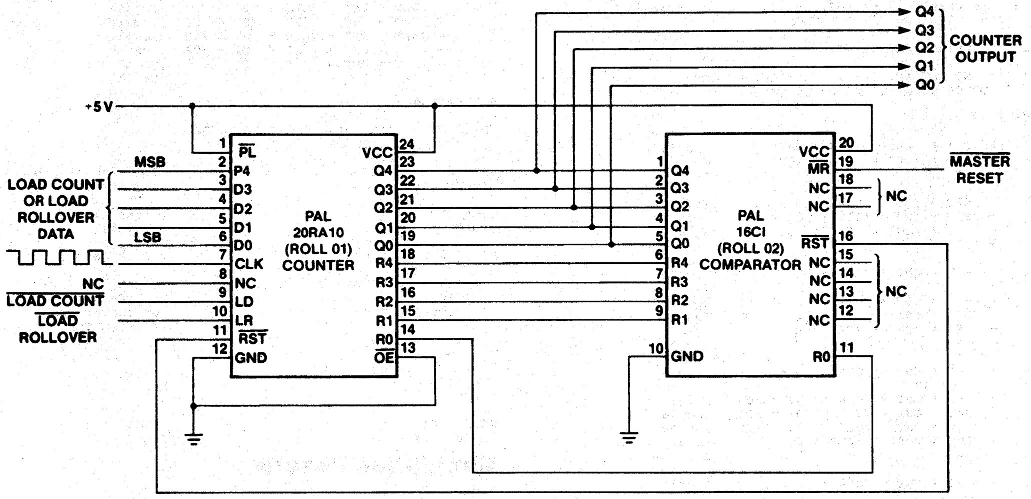


TB01880M

Functional Description

Shown below is a schematic of two PAL devices implementing a 5-bit up asynchronous ring counter with programmable rollover, asynchronous load, and reset. Initial count point can be loaded by asserting /LD low. Rollover point is loaded by asserting /LR low. Q4...Q0 and R4...R0 are compared in the PAL16C1 which is implemented as a comparator. The result of the comparison is fed back from the PAL16C1 to the PAL20RA10 device through the /RST line. Note that a Master Reset must be executed first before an initial count point can be loaded.

Block Diagram



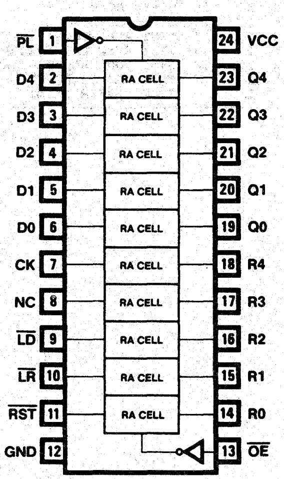
TC00020M

PAL Device Design Specification

```
Title    5-Bit Up Counter
Pattern  UpCount.pds
Revision A
Author   Bill Karkula
Company  Monolithic Memories Inc., Santa Clara, CA
Date     7/19/84

CHIP UpCounter PAL20RA10

/PL D4 D3 D2 D1 D0 CK NC /LD /LR /RST GND
/OE R0 R1 R2 R3 R4 Q0 Q1 Q2 Q3 Q4 VCC

EQUATIONS

/Q4             := Q4                   ;Toggle if lower MSB
Q4.CLKF          = /Q3                  ;becomes a zero
Q4.RSTF          = RST                  ;Rollover/master RST
Q4.SETF          = D4*LD                ;Load initial Count

/Q3             := Q3                   ;Toggle when Q2
Q3.CLKF          = /Q2                  ;becomes a zero
Q3.RSTF          = RST                  ;Rollover/master RST
Q3.SETF          = D3*LD                ;Load initial count

/Q2             := Q2                   ;Toggle when Q1
Q2.CLKF          = /Q1                  ;becomes a zero
Q2.RSTF          = RST                  ;Rollover/master RST
Q2.SETF          = D2*LD                ;Load initial count

/Q1             := Q1                   ;Toggle when Q0
Q1.CLKF          = /Q0                  ;becomes a zero
Q1.RSTF          = RST                  ;Rollover/master RST
Q1.SETF          = D3*LD                ;Load initial count

/Q0             := Q0                   ;Toggle LSB
Q0.CLKF          = CK                   ;External CLKF input
Q0.RSTF          = RST                  ;Rollover/master RST
Q0.SETF          = D0*LD                ;Load initial count

/R4             := /D4                  ;Load rollover point
R4.CLKF          = LR                   ;if /LR is low
/R3             := /D3                  ;Load rollover bit 3
R3.CLKF          = LR                   ;if /LR is asserted
/R2             := /D2                  ;Load rollover bit 2
R2.CLKF          = LR                   ;if /LR is asserted
/R1             := /D1                  ;Load rollover bit 1
R1.CLKF          = LR                   ;if /LR is asserted
/R0             := /D0                  ;Load rollover bit 0
R0.CLKF          = LR                   ;if /LR is asserted

SIMULATION

TRACE_ON PL OE CK Q0 Q1 Q2 Q3 Q4

SETF RST /LD                            ;Test SET function
                                        ;of registers
CHECK  Q0  Q1  Q2  Q3  Q4               ;Check for high
                                        ;on reg. outputs
SETF /RST                               ;Deassert SET funct

SETF D0 D1 D2 D3 D4 LD                  ;Test RESET function
CHECK /Q0 /Q1 /Q2 /Q3 /Q4

SETF /OE Q4 Q3 Q2 Q1 /Q0 /LD            ;Disable RESET(LD=0)
                                        ;load registers w/
                                        ;HHHHL (Q4..Q0),
                                        ;tristate registers
SETF PL                                 ;Load reg.'s w/ data
                                        ;on output bus.
SETF OE /PL                             ;Disable PRELOAD &
                                        ;TRISTATE funct.
SETF CK
SETF /CK

SETF CK
SETF /CK

SETF CK
SETF /CK

SETF CK
SETF /CK
```

Simulation Results

```
   g gg g ggg    gg gg    gg
PL XXXXXXXXHLL LLLLLLLLLL LLL
OE XXXXXLLLHH HHHHHHHHHH HHH
CK XXXXXXXXXH HHLHHLHHHH LHH
Q0 HHHLLZZLHH LLLLHHHLLL LLH
Q1 HHHLLZZHLL LHHHHHHHLL LLL
Q2 HHHLLZZHLL LLLLLLLLLH HHH
Q3 HHHLLZZHLL LLLLLLLLLL LLL
Q4 HHHLLZZHLL LLLLLLLLLL LLL
```



TB01900M

PAL Device Design Specification

```
PAL16C1
ROLL02
COMPARATOR
MONOLITHIC MEMORIES INC., SANTA CLARA, CA
Q4 Q3 Q2 Q1 Q0 R4 R3 R2 R1 GND
R0 NC NC NC NC /RST NC NC /MR VCC

EQUATIONS

/RST     = Q4*/R4*/MR + /Q4*R4*/MR        ;COMPARE Q4 & R4 MSB
         + Q3*/R3*/MR + /Q3*R3*/MR        ;COMPARE Q3 & R3
         + Q2*/R2*/MR + /Q2*R2*/MR        ;COMPARE Q2 & R2
         + Q1*/R1*/MR + /Q1*R1*/MR        ;COMPARE Q1 & R1
         + Q0*/R0*/MR + /Q0*R0*/MR        ;COMPARE Q0 & R0 LSB

FUNCTION TABLE

Q4 Q3 Q2 Q1 Q0 R4 R3 R2 R1 R0 /MR /RST

;               /
;             / R
;QQQQQ RRRRR M  S
;43210 43210 R  T
-----------------------------------------------------
 HLLLL LLLLL H  H      1  Q4=H, R4=L
 LHLLL LLLLL H  H      2  Q3=H, R3=L
 LLHLL LLLLL H  H      3  Q2=H, R2=L
 LLLHL LLLLL H  H      4  Q1=H, R1=L
 LLLLH LLLLL H  H      5  Q0=H, R0=L
 LLLLL HLLLL H  H      6  Q4=L, R4=H
 LLLLL LHLLL H  H      7  Q3=L, R3=H
 LLLLL LLHLL H  H      8  Q2=L, R2=H
 LLLLL LLLHL H  H      9  Q1=L, R1=H
 LLLLL LLLLH H  H     10  Q0=L, R0=H
 LLLLL LLLLL H  L     11  TEST ALL LOWS
 HHHHH HHHHH H  L     12  TEST ALL HIGHS
 LHLHL LHLHL H  L     13  TEST EVEN CHECKERBOARD
 HLHLH HLHLH H  L     14  TEST ODD CHECKERBOARD
 LLLLL HHHHH L  L     15  TEST MASTER RESET
 HHHHH LLLLL L  L     16  TEST MASTER RESET
---------------------------------------------------------
```

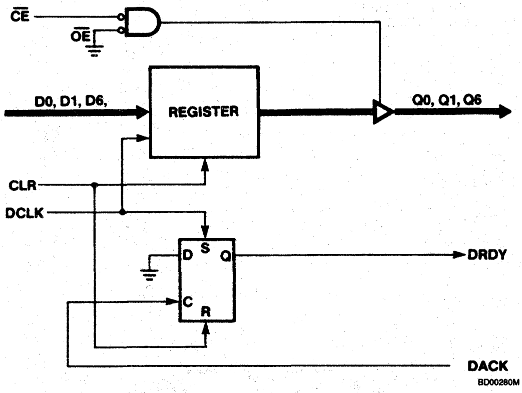
```
DESCRIPTION

PAL16C1 DESIGN SPECIFICATION
5-BIT COMPARATOR WITH MASTER RESET OVERRIDE
(SECOND OF THE TWO PALS SOLUTION ON THE 5-BIT COUNTER WITH
PROGRAMMABLE ROLLOVER, ASYNCHRONOUS LOAD AND RESET.)
MONOLITHIC MEMORIES INC., SANTA CLARA, CA
BILL KARKULA                    7/19/84

THIS DEVICE COMPARES 5 BITS OF DATA (R4...R0) WITH Q4...Q0
AND ASSERTS /RST IF THEY ARE EQUAL.  THEREFORE /RST
GOES LOW WHEN PROGRAMMED ROLLOVER POINT R4...R0
MATCHES COUNT Q4...Q0.  /RST ALSO GOES LOW WHEN /MR
GOES LOW, INDICATING A MASTER RESET.

NOTE:  THIS PAL DESIGN SPEC WAS ASSEMBLED ON PALASM V1.7.
```

TB01890M

Functional Description

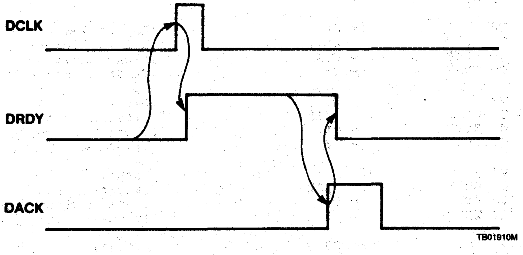
This application is for a seven-bit register with handshake logic. The chip can be used for interfacing between a microprocessor and its peripheral I/O. The on-chip flag flip-flop provides the handshaking capability required in typical demand-response-based data transfer. Both the register and the flag flip-flops are asynchronously cleared by CLR signal.

Block Diagram



Input data is stored in the register when DCLK signal is applied, and at the same time, the event is signified by asserting DRDY signal. The DRDY signal indicates that the data is available in the register. By monitoring the DRDY signal when it is high, the stored input data can be transferred to Q output port by asserting /OE three-state control signal. After moving the data, DACK signal should be applied to clear the flag flip-flop.

Handshake Operation

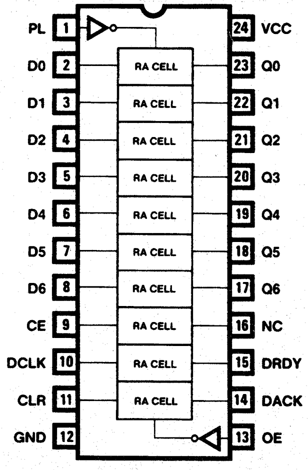


PAL Device Design Specification

```
Title      7-Bit I/O Port with Handshake Logic
Pattern    Port.pds
Revision   A
Author     Sadahiro Horiko / Kelvin Chow
Company    Monolithic Memories Inc., Santa Clara, Ca
Date       3/1/85

CHIP IOPORT PAL20RA10

PL D0 D1 D2 D3 D4 D5 D6 CE DCLK CLR GND
OE DACK DRDY NC Q6 Q5 Q4 Q3 Q2 Q1 Q0 VCC

EQUATIONS

Q0                := D0                      ;LSB of 7-bit regs
Q0.CLKF            = DCLK                    ;External clock
Q0.SETF            = CLR                     ;Clear register
Q0.TRST            = CE                      ;Tristate control

Q1                := D1                      ;Data 1
Q1.CLKF            = DCLK                    ;External clock
Q1.SETF            = CLR                     ;Clear register
Q1.TRST            = CE                      ;Tristate control

Q2                := D2                      ;Data 2
Q2.CLKF            = DCLK                    ;External clock
Q2.SETF            = CLR                     ;Clear register
Q2.TRST            = CE                      ;Tristate control

Q3                := D3                      ;Data 3
Q3.CLKF            = DCLK                    ;External clock
Q3.SETF            = CLR                     ;Clear register
Q3.TRST            = CE                      ;Tristate control

Q4                := D4                      ;Data 4
Q4.CLKF            = DCLK                    ;External clock
Q4.SETF            = CLR                     ;Clear register
Q4.TRST            = CE                      ;Tristate control

Q5                := D5                      ;Data 5
Q5.CLKF            = DCLK                    ;External clock
Q5.SETF            = CLR                     ;Clear register
Q5.TRST            = CE                      ;Tristate control

Q6                := D6                      ;Data 6
Q6.CLKF            = DCLK                    ;External clock
Q6.SETF            = CLR                     ;Clear register
Q6.TRST            = CE                      ;Tristate control

DRDY              := GND                     ;Handshake logic
DRDY.CLKF          = DACK                    ;Cleared by DACK
DRDY.RSTF          = DCLK                    ;Clear
DRDY.SETF          = CLR                     ;Asserted by DCLK
DRDY.TRST          = VCC                     ;(External clock)

SIMULATION

TRACE_ON CLR Q0 Q1 Q2 Q3 Q4 Q5 Q6 DCLK DRDY DACK

SETF PL /CE /OE /D0 D1 /D2 D3 /D4 D5 /D6 CLR /DCLK /DACK
                                             ;Set input values
                                             ;Tristate outputs

SETF CE OE CLR                               ;Remove the tri-
                                             ;states on the
                                             ;outputs and clear
                                             ;registers

SETF CLR
SETF CLR

SETF /CLR                                    ;Clock the data &
SETF DCLK                                    ;set DRDY register
SETF DCLK

SETF /DCLK                                   ;Remove the clock

SETF DACK                                    ;Assert DACK
SETF DACK

SETF /DACK                                   ;Lower DACK
SETF /DACK
```

Simulation Results

```
Page :   1
          g g gg gg  g
 CLR      HHHHLLLLLL L
 Q0       XZZLLLLLLL L
 Q1       XZZLLLHHHH H
 Q2       XZZLLLLLLL L
 Q3       XZZLLLHHHH H
 Q4       XZZLLLLLLL L
 Q5       XZZLLLHHHH H
 Q6       XZZLLLLLLL L
 DCLK     LLLLLHLLL L
 DRDY     XLZZLHHHHL L
 DACK     LLLLLLLLHH L
```



TB01920M

Functional Description

Original application was developed by LTT, Conflans Ste. Honorine, FRANCE. Part of the schematics, reprinted with courtesy of LTT, is used to control a serial data link based upon a specialized LSI chip.

Originally designed with six standard SSI/MSI circuits, this same function can now be implemented, not only into a single PAL20RA10 device, but with even more features and better performance. The function can be divided into three subfunctions:

1. Address Decoding
2. Control Flags
3. Transmission Speed Selection

Up to four address lines are allowed (eight were actually used), plus two extra lines which are special decoding controls (MEM/IO selection, Enable Control...). Two flip-flop load flag conditions, from the address bus (A1 and A2), providing handshake between the 6850 UART and the communication lines. They have a common clock which also serves as Chip Select (CSO) for the UART.

The UART Transmit clock (TXCLK) can be directly connected to the Receive Clock (CK or RXCLK) or represents the Receive Clock value divided by sixteen. This function was performed by four D-type Flip-Flops connected as a 4-stage Asynchronous Divider. Since each basic cell, used in a PAL20RA10 device has four Product Terms available, this function could be implemented either asynchronously or synchronously. In the PAL Design Specification example, a 4-bit synchronous divider was used instead of the asynchronous circuit shown in the schematic.

Pin Description

1.	TEST	Allows preload function for testing.
2.	SYSRESET	Reset line from microprocessor.
3.	A2	Address line from address bus.
4.	A1	Address line from address bus.
5.	HDSHAKE	Handshake line (CTS/RTS).
6.	CK	External clock.
7.	E	Enable line from microprocessor.
8.	AUXDECOD	Extra decoding line (e.g. board level decoding).
9.	A3	Address line from address bus.
10.	A4	Address line from address bus.
11.	A5	Address line from address bus.
12.	GND	Reference power supply ground.
13.	/OE	Output enable line.
14.	A6	Address line from address bus.
15.	SPEEDSEL	Speed selection line.
16.	DIV4	MSB 4-bit synchronous counter.
17.	DIV3	3rd stage synchronous counter.
18.	DIV2	2nd stage synchronous counter.
19.	DIV1	LSB 4-bit synchronous counter.
20.	CS0	UART chip select line (CSO).
21.	BLOCREC	Bloc receive line.
22.	DIR DIV	Direct or divided clock.
23.	/TPH	External use flag.
24.	VCC	5V power supply.

Before

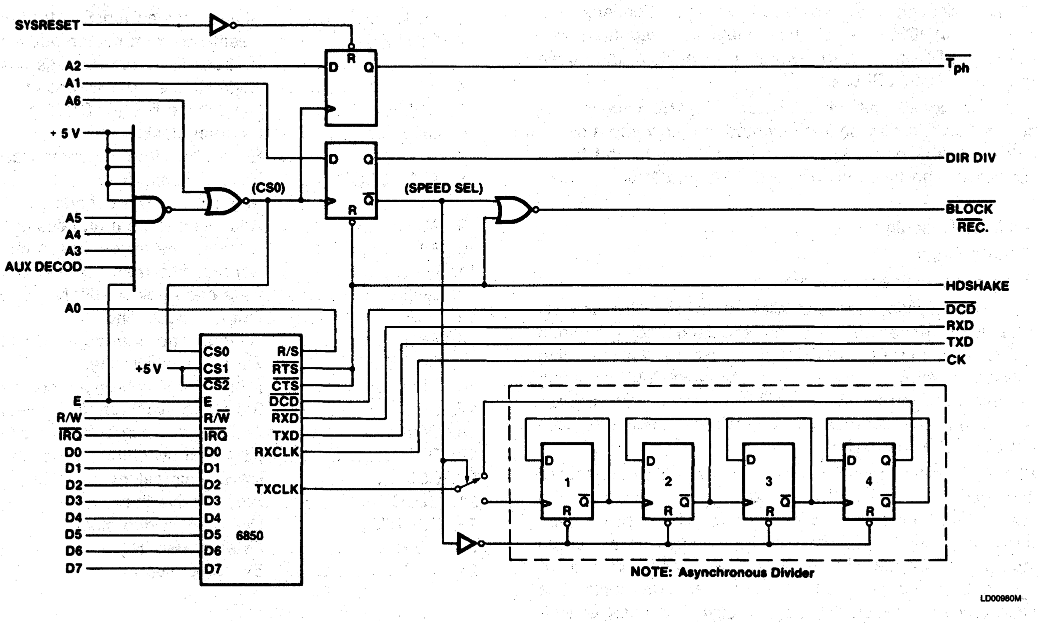


Figure 6.

After

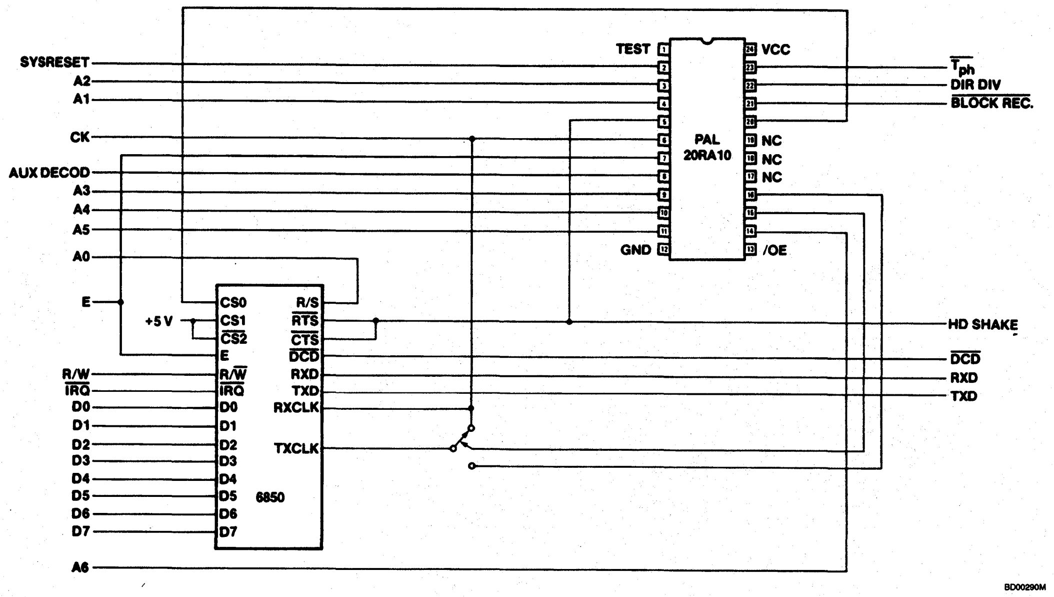


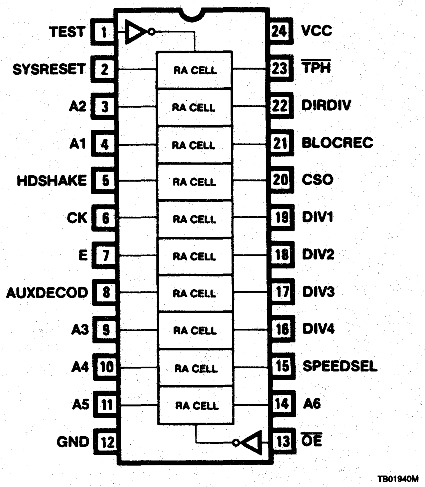
Figure 7.

PAL Device Design Specification

```
Title      Serial Data Link Controller
Pattern    Link.pds
Revision   A
Author     Jose Juntas / Kelvin Chow
Company    Monolithic Memories Inc., Santa Clara, Ca
Date       3/1/85

CHIP SE_CH_CNTRL  PAL20RA10

TEST SYSRESET A2 A1 HDSHAKE CK E AUXDECOD A3 A4 A5 GND
/OE A6 SPEEDSEL DIV4 DIV3 DIV2 DIV1 CSO BLOCREC DIRDIV
/TPH VCC

EQUATIONS

/TPH             := A2                      ;Load A2 as flag
/TPH.CLKF         = CSO                     ;CLK W/ ADDR. decode
/TPH.SETF         = SYSRESET                ;global system reset

DIRDIV           := A1                      ;Load speed ratio
DIRDIV.CLKF       = CSO                     ;CLK W/ ADDR. decode
DIRDIV.SETF       = /HDSHAKE                ;CLR by CTS/RTS line

/BLOCREC          = /DIRDIV                 ;Controlled by speed
                  + HDSHAKE                 ;option and CTS/RTS
                                            ;line
CSO               = /A6*A5*A4*A3*AUXDECOD*E
                                            ;UART address valid
/DIV1            := DIV1                    ;4-bit synchronous
                                            ;divider LSB
/DIV1.CLKF        = CK                      ;CLK by CK(external)
/DIV1.SETF        = /DIRDIV                 ;CLR by speed option

/DIV2            := /DIV1*/DIV2             ;2ND stage of
                  + DIV1*DIV2               ;divider
/DIV2.CLKF        = CK                      ;CLK by CK(external)
/DIV2.SETF        = /DIRDIV                 ;CLR by speed option

/DIV3            := /DIV2*/DIV3             ;3RD stage of
                  + /DIV1*/DIV3             ;divider
                  + DIV1*DIV2*DIV3
/DIV3.CLKF        = CK                      ;CLK by CK(external)
/DIV3.SETF        = /DIRDIV                 ;CLR by speed option

/DIV4            := /DIV3*/DIV4             ;4TH stage of
                  + /DIV2*/DIV4             ;divider MSB
                  + /DIV1*/DIV4
                  + DIV1*DIV2*DIV3*DIV4
/DIV4.CLKF        = CK                      ;CLK by CK(external)
/DIV4.SETF        = /DIRDIV                 ;CLR by speed option

SPEEDSEL         := /A1                     ;Load speed choice
SPEEDSEL.CLKF     = CSO                     ;CLK W/ ADDR. decode

SPEEDSEL.SETF     = /HDSHAKE                ;CLR by CTS/RTS line

SIMULATION

TRACE_ON A1,A2,A3,A4,A5,A6,E,               ;Signals to be
         AUXDECOD,SYSRESET,/TPH,HDSHAKE,             ;observed
         CSO,SPEEDSEL,DIRDIV,CK,
         DIV1,DIV2,DIV3,DIV4
SETF SYSRESET,/HDSHAKE                      ;Reset all regs

CHECK /SPEEDSEL,/DIRDIV,TPH
SETF /SYSRESET,A1,A2,A3,A4,A5,/A6,HDSHAKE,   ;Set decode
    E,AUXDECOD                               ;condition

CHECK /SPEEDSEL,DIRDIV                      ;Check SPEEDSEL and
                                            ;DIRDIV regs

FOR I:=1 TO 15 DO                           ;This portion
 BEGIN                                      ;simulates divide
  SETF CK                                   ;by four counter

  SETF /CK
 END
```

Simulation Results

```
Page :  1
          g g   g gg  gg gg gg g g gg gg gg  gg gg gg
 A1       XXHHHHHHHH HHHHHHHHHH HHHHHHHHHH HHHHHHHHHH
 A2       XXHHHHHHHH HHHHHHHHHH HHHHHHHHHH HHHHHHHHHH
 A3       XXHHHHHHHH HHHHHHHHHH HHHHHHHHHH HHHHHHHHHH
 A4       XXHHHHHHHH HHHHHHHHHH HHHHHHHHHH HHHHHHHHHH
 A5       XXHHHHHHHH HHHHHHHHHH HHHHHHHHHH HHHHHHHHHH
 A6       XXLLLLLLLL LLLLLLLLLL LLLLLLLLLL LLLLLLLLLL
 E        XXHHHHHHHH HHHHHHHHHH HHHHHHHHHH HHHHHHHHHH
 AUXDECOD XXHHHHHHHH HHHHHHHHHH HHHHHHHHHH HHHHHHHHHH
 SYSRESET HHLLLLLLLL LLLLLLLLLL LLLLLLLLLL LLLLLLLLLL
/TPH      LLLLHHHHHH HHHHHHHHHH HHHHHHHHHH HHHHHHHHHH
 HDSHAKE  LLHHHHHHHH HHHHHHHHHH HHHHHHHHHH HHHHHHHHHH
 CSO      XXHHHHHHHH HHHHHHHHHH HHHHHHHHHH HHHHHHHHHH
 SPEEDSEL LLLLLLLLLL LLLLLLLLLL LLLLLLLLLL LLLLLLLLLL
 DIRDIV   LLLLHHHHHH HHHHHHHHHH HHHHHHHHHH HHHHHHHHHH
 CK       XXXXXHHLHH LHHLHHLHHL HHLHHLHHLH HLHHLHHLHH
 DIV1     XLLLLLHHHL LLHHHLLLHH HLLLHHHLLL HHHLLLHHHL
 DIV2     XLLLLLLLLH HHHHHLLLLL LHHHHHHLLL LLLHHHHHHL
 DIV3     XLLLLLLLLL LLLLLHHHHH HHHHHHHLLL LLLLLLLLLH
 DIV4     XLLLLLLLLL LLLLLLLLLL LLLLLLLHHH HHHHHHHHHH


Page :  2
          gg gg gg g
 A1       HHHHHHHHHH
 A2       HHHHHHHHHH
 A3       HHHHHHHHHH
 A4       HHHHHHHHHH
 A5       HHHHHHHHHH
 A6       LLLLLLLLLL
 E        HHHHHHHHHH
 AUXDECOD HHHHHHHHHH
 SYSRESET LLLLLLLLLL
/TPH      HHHHHHHHHH
 HDSHAKE  HHHHHHHHHH
 CSO      HHHHHHHHHH
 SPEEDSEL LLLLLLLLLL
 DIRDIV   HHHHHHHHHH
 CK       LHHLHHLHHL
 DIV1     LLHHHLLLHH
 DIV2     LLLLLHHHHH
 DIV3     HHHHHHHHHH
 DIV4     HHHHHHHHHH
```



Functional Description

One of the more widely used computer families is the Digital Equipment Corp.'s PDP-11 series. This family of computers uses the DEC unibus to communicate between cards. A specific protocol is required to interface a card to the unibus. This protocol is described in the available DEC literature.

Since the unibus is an asynchronous bus, much of the interface circuitry consists of combinational logic to generate specific signals and flip-flops which are set and reset as flags. This tends to use a lot of SSI and MSI logic packages. Using Monolithic Memories' PAL devices, much of this logic can be condensed into a few packages. Figure 2 is the schematic diagram for an interrupt Controller to be used on the unibus. (p.6 – 30 of the 1976 DEC PDP-11 Peripherals Handbook.)

Many cards communicate over the bus by taking control of the unibus with an interrupt request, and then do whatever they require before releasing control. As can be seen, this interrupt controller takes six special interface ICs, (380 and 8881 bus drivers and receivers) eight MSI, SSIIC, (7400, 7402 and 7474s) along with some transistors and discrete parts. This parts count can be considerably reduced by using PAL20RA10 and PAL20L10 devices.

Figure 1 shows how the circuit with the PAL devices would look. The two PAL devices allow almost all of the 7400, 7402 and 7474 packages to be removed. (Almost a 4-1 saving in chip count.) In addition the preload pin (PRLD) on the 20RA10 allows the flip-flops to be easily set to a known state on power up, or when re-initializing. So the PAL devices reduce the logic package count from eight chips to three.

This shows that by using PAL devices substantial space and circuit savings can be realized when interfacing to the unibus.

In the schematic shown, there are three VLSI devices, three MSIs and two SSIs. Using a PAL20RA10 logic circuit, it is possible to replace three MSIs and one SSI device, thereby reducing the chip count by a factor of two. The ICs inside the enclosed loop were replaced.

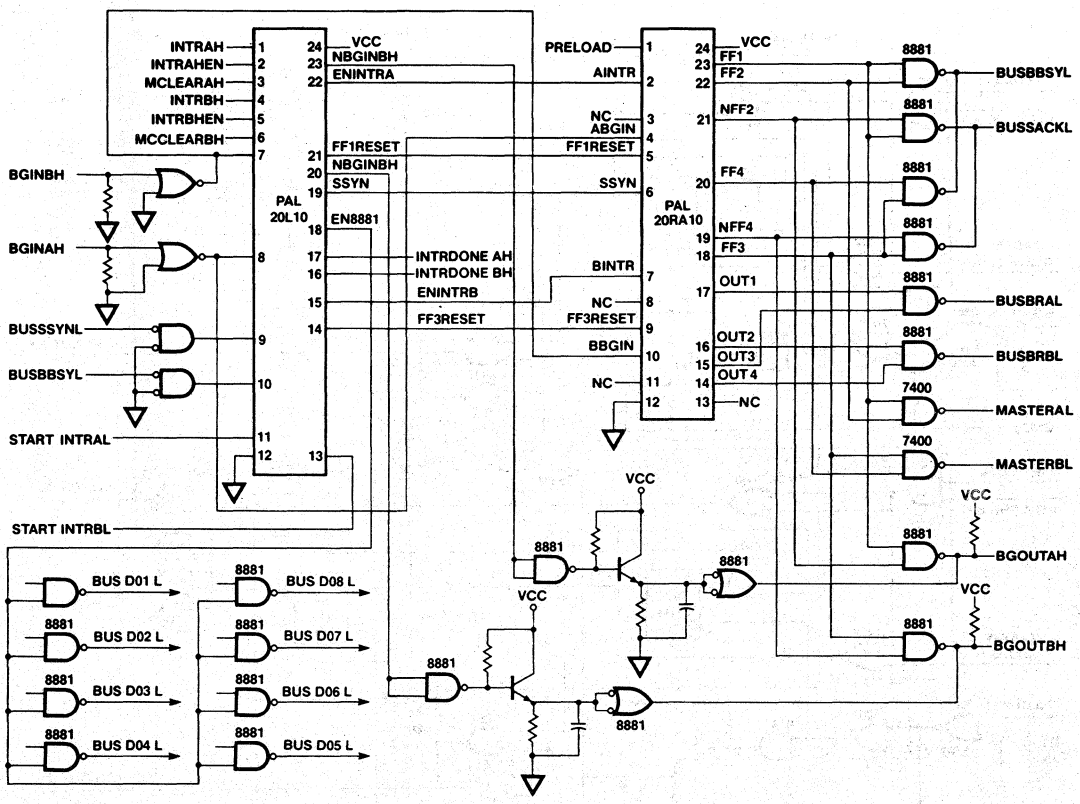


Figure 1.

PAL Design Specification

```
PAL20L10
INTRP01
INTERRUPT LOGIC
MONOLITHIC MEMORIES INC., SANTA CLARA, CA
INTRAH INTRAHEN MCLEARAH INTRBH INTRBHEN MCLEARBH BGINBH
BGINAH BUSSSYNL BUSBBSYL STARTINTRAL GND
STARTINTRBL FF3RESET ENINTRB INTRDONEBH INTRDONEAH EN8881
SSYN NBGINBH FF1RESET ENINTRA NBGINAH VCC

EQUATIONS

/NBGINAH         =  BGINAH              ;FF1 CLK CONTROL
                                        ; BLOCK A
/FF1RESET        =  MCLEARAH+ENINTRA    ;SET FF1 CONTROL
                                        ; BLOCK A
/ENINTRA         =  INTRAHEN*INTRAH     ;ENABLE INTERRUPT A

/NBGINBH         =  BGINBH              ;FF3 CLOCK CONTROL
                                        ; BLOCK B
/SSYN            =  BUSSSYNL*BUSBBSYL   ;SYNCHRONIZE FF2 &
                                        ; FF4
/EN8881          = STARTINTRAL*STARTINTRBL
                                        ;INTERRUPT BUS
/INTRDONEAH      = BUSSSYNL+STARTINTRAL ;SIGNAL INTERRUPT
                                        ; DONE
/ENINTRB         = INTRBH*INTRBHEN      ;ENABLE INTERRUPT B

/FF3RESET        = MCLEARBH+ENINTRB     ;SET FF3 CONTROL
                                        ; BLOCK B
/INTRDONEBH      = BUSSSYNL+STARTINTRBL ;SIGNAL INTERRUPT
                                        ; DONE
```

```
DESCRIPTION

COMBINATORIAL LOGIC FOR PAL20RA10 INTERRUPT CONTROLLER
(1ST PART OF THE TWO PALS SOLUTION: PAL20L10 & PAL20RA10)
MONOLITHIC MEMORIES INC., SANTA CLARA, CA
DAN KINSELLA             7/19/84

NOTE: THIS PAL DESIGN SPEC WAS ASSEMBLED ON PALASM V1.7.
```

TB01960M

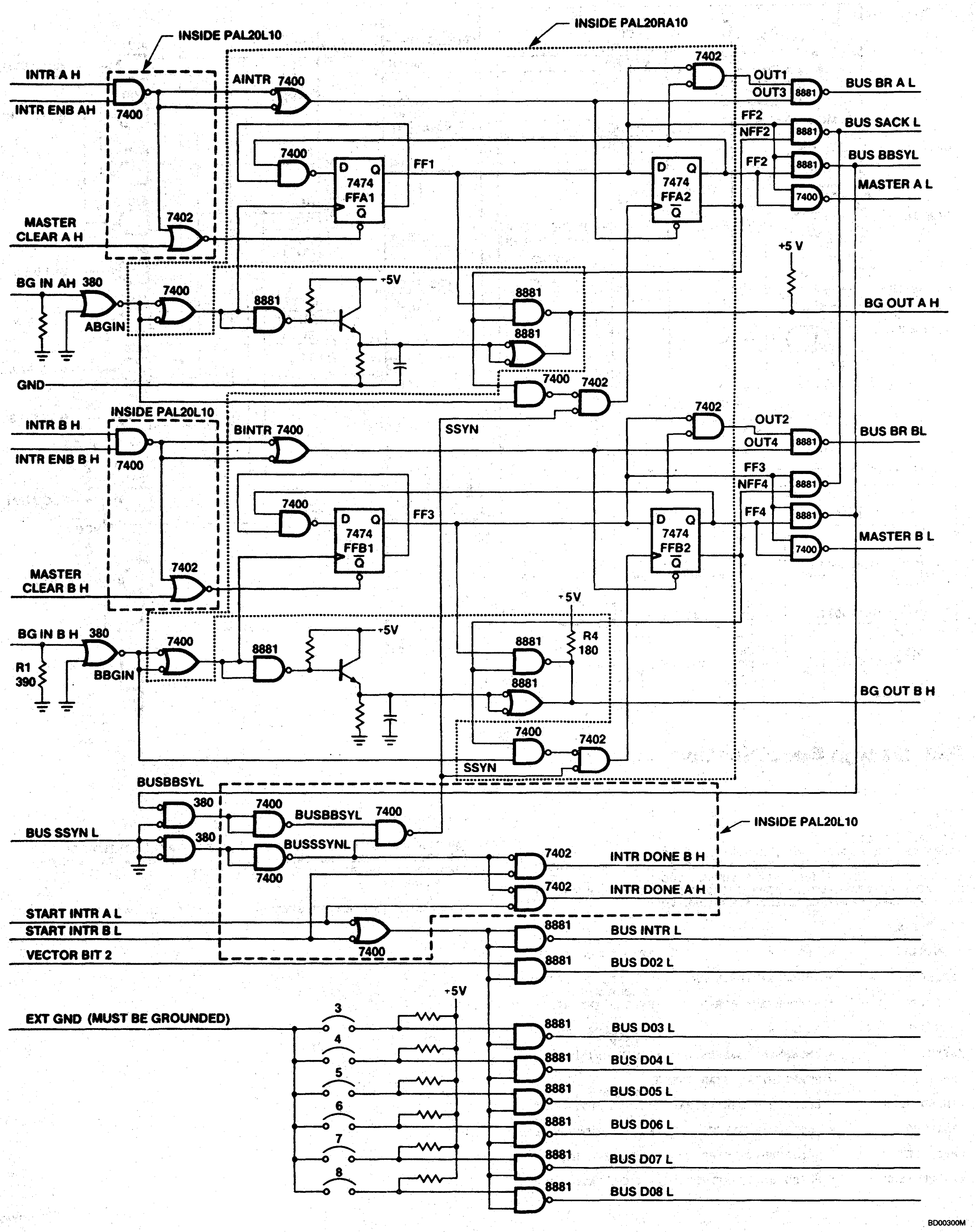

INSIDE PAL20L10
INSIDE PAL20RA10
INTR A H
INTR ENB AH
7400
AINTR 7400
7402
OUT1
OUT3
8881
BUS BR A L
FF2
NFF2
BUS SACK L
BUS BBSYL
MASTER A L
FF1
7474 FFA1
7474 FFA2
MASTER CLEAR A H
+5 V
BG IN AH 380
ABGIN
8881
+5V
BG OUT A H
GND
7400 7402
SSYN
INSIDE PAL20L10
INTR B H
INTR ENB B H
BINTR 7400
OUT2
OUT4
BUS BR BL
FF3
NFF4
FF4
MASTER B L
7474 FFB1
7474 FFB2
MASTER CLEAR B H
BG IN B H 380
R1 390
BBGIN
R4 180
BG OUT B H
BUSBBSYL
BUS SSYN L
BUSSSYNL
INSIDE PAL20L10
INTR DONE B H
INTR DONE A H
START INTR A L
START INTR B L
VECTOR BIT 2
BUS INTR L
BUS D02 L
EXT GND (MUST BE GROUNDED)
3
4
5
6
7
8
BUS D03 L
BUS D04 L
BUS D05 L
BUS D06 L
BUS D07 L
BUS D08 L
BD00300M

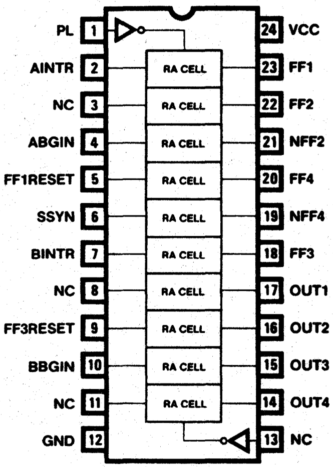
Figure 2.

PAL Device Design Specification

```
Title       DEC PDP-11 unibus interrupt controller
Pattern     Control.pds
Revision    A
Author      Dan Kinsella
Company     Monolithic Memories Inc., Santa Clara, CA
Date        3/1/85

CHIP INTR_CONTROL PAL20RA10

PL AINTR NC ABGIN FF1RESET SSYN BINTR NC FF3RESET BBGIN
NC GND
NC OUT4 OUT3 OUT2 OUT1 FF3 NFF4 FF4 NFF2 FF2 FF1 VCC

EQUATIONS

/FF1            := /FF1*FF2             ;Master control
FF1.SETF         = /FF1RESET            ;block A
FF1.CLKF         = /ABGIN

FF2             := FF1                  ;Bus Busy Signal
FF2.SETF         = /AINTR
FF2.CLKF         = ABGIN*FF2*/SSYN

/NFF2           := FF1                  ;Bus sack signal
NFF2.SETF        = /AINTR
NFF2.CLKF        = ABGIN*NFF2*/SSYN

/FF3            := /FF3*FF4             ;Master control
FF3.SETF         = /FF3RESET            ;block B
FF3.CLKF         = /BBGIN

FF4             := FF4                  ;Bus busy signal
FF4.SETF         = /BINTR
FF4.CLKF         = BBGIN*FF4*/SSYN

/NFF4           := FF3                  ;Bus sack signal
NFF4.SETF        = /BINTR
NFF4.CLKF        = BBGIN*NFF4*/SSYN

/OUT1            = FF1+FF2              ;Bus request signal
                                        ;block A
/OUT2            = FF4+FF3              ;Bus request signal
                                        ;block B
/OUT3            = AINTR                ;Intr. signal for
                                        ;bus req. block A
/OUT4            = BINTR                ;Intr. signal for
                                        ;bus req. block B

SIMULATION

TRACE_ON FF1RESET FF3RESET AINTR BINTR SSYN ABGIN BBGIN
         FF1 FF3 NFF2 NFF4 OUT1 OUT2 OUT3 OUT4

SETF /FF1RESET /FF3RESET AINTR BINTR    ;Reset all regs

SETF FF1RESET FF3RESET /AINTR /BINTR    ;Clock FF1 and FF3
     ABGIN BBGIN                        ;regs

SETF /SSYN                              ;Clock NFF and NFF3
                                        ;regs
```

Simulation Results

```
Page :   1
                gg g
 FF1RESET LHHH
 FF3RESET LHHH
 AINTR    HLLL
 BINTR    HLLL
 SSYN     XXXL
 ABGIN    XHHH
 BBGIN    XHHH
 FF1      LLLL
 FF3      LLLL
 NFF2     XLLL
 NFF4     XLLL
 OUT1     XHHH
 OUT2     XHHH
 OUT3     LHHH
 OUT4     LHHH
```



TB01970M

Notes

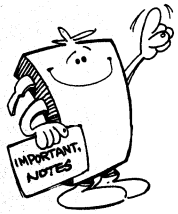


Table of Contents

1. Boolean Algebra

1.1 The Language of Logic

Although you may not be aware of it, you are already an expert at forming, simplifying and comprehending Boolean equations and expressions. Boolean algebra, in its most common application, is concerned with the truth or falsity of statements. Any time you describe what circumstances would make something true or false, you have made a Boolean equation.

For example, suppose A is true only if B and C are true. These three letters may represent anything you like — A may be whether or not you may become president, B may be whether or not you are elected, and C may be whether or not you are a citizen of the U.S.A. You may become president only if you are elected and you are a citizen of the United States. If we wrote that statement in equation form, it might look like this:

$$A = B*C$$

where the * is a shorthand notation for the word 'AND'. A, B and C are all Boolean variables, since they represent some value which may be either true or false. You either are a citizen of the United States, or you are not — there is no in between. Examining the relationship between these three variables, we find that:

1) if you are elected and you are citizen then you may become president;
2) if you are elected but you are not a citizen then you cannot become president;
3) if you are not elected, but you are a citizen, you still can't become president, and;
4) if you are neither elected nor a citizen, then you definitely cannot become president.

This same relationship, which may be expressed in terms of an English sentence, may also be represented by a table of all the possibilities, called a truth table. If we let "1' stand for true, and "0' stand for false, we can make the following table:

B	C	A
0	0	0
0	1	0
1	0	0
1	1	1

Figure 1-1. The AND Operator

The table above is a standard way of expressing logical relationships. Our truth table lists the possibilities one-by-one. If B and C are false, then A will be false. If B is true and C is false, then A will still be false. If B is false, and C is true then A will again be false. However, if B and C are both true, then A will be true.

1.2 AND, OR and NOT

The fact is that every time you have an equation of the form:

$$A = B*C$$

you will have a truth table in the form of Figure 1-1 because the table and the word 'AND' are just two ways of expressing the same relationship between two Boolean variables.

Now let's look at the operator 'OR'. Suppose A is true if B or C is true. This equation can be written:

$$A = B + C$$

Do not confuse the + with the addition sign of arithmetic; in Boolean algebra, it is shorthand notation for the word 'OR'. A truth table for this equation would be:

B	C	A
0	0	0
0	1	1
1	0	1
1	1	1

Figure 1-2. The OR Operator

This table expresses a different relationship between the variables than AND does; AND requires that both of its operands be true for the expression to be true. OR only requires that one of its operands be true for the expression to be true. From the table above, we can see that:

1) if both B and C are false, then A is false:
2) if B is false, and C is true, then A is true:
3) if B is true and C is false, then A is true and:
4) if both B and C are true, then A is true.

Finally, let's look at the operator 'NOT'. If A equals NOT B, then the value of A is the inverse of B. This equation would be:

$$A = /B$$

Again, the '/' should not be mistaken for the division sign of arithmetic. It is a shorthand notation for the Boolean operator, 'NOT'. The truth table for this equation is:

B	A
0	1
1	0

Figure 1-3. The NOT Operator.

which is to say that:

1) if B is false, then A is true and:
2) if B is true then A is false.

The elementary operators are summarized in Figure 1-4.

OPERATOR	SYMBOL
AND	*
OR	+
NOT	/

Figure 1-4. Elementary Boolean Operators

1.3 Precedence

In arithmetic, the multiplication sign is always evaluated before the addition sign. For example:

3 + 4 x 7

is 31, not 49. Similarly, the AND sign is always evaluated before the OR sign. Another way to say this is that AND has a higher precedence than OR.

Of course, in arithmetic, the precedence of operators may be changed with parentheses. If you wish the expression:

3 + 4 x 7

to be evaluated as 49, then you should write it as:

(3 + 4) x 7

The parentheses enclose a subexpression that should be evaluated before the expression as a whole can be evaluated.

Of the three Boolean operators we have seen so far, NOT has the highest precedence, then AND, then OR.

1.4 Associativity and Commutativity

Both the AND and OR operators (and, in fact, all binary Boolean operators) have the property of associativity. The property of associativity says that in an expression with more than one operator of the same kind, it does not matter which you evaluate first. In terms of equations, this would be:

B*(C*D) = (B*C)*D

or

B + (C + D) = (B + C) + D

All binary Boolean operators are also commutative. This means that the order in which the operands appear is not important. In equations, that would be

B*C = C*B

or

B + C = C + B

1.5 Postulates and Theorems

In 1854, the mathematician and philosopher George Boole published his book, 'An Investigation of The Laws of Thought', in which he demonstrated how classical logic could be defined with algebraic terminology and operations. Then, in 1938, C. E. Shannon published his paper "A Symbolic Analysis of Relay and Switching Circuits", which demonstrated a Boolean algebra of two values called "switching algebra", which could be used to represent the properties of bistable electric switching circuits. A minimal set of formal postulates is needed in order to define this Boolean algebra. Here we will define Boolean algebra to be an algebra defined over the set B, where B = (False, True) and over the operators AND (*), OR (+) and NOT (/), such that:

1) All operators are closed (which means that it is impossible to create a Boolean expression that has a value other than True or False),
2) Postulates 1 through 4 in Figure 1-5 are true, and
3) NOT is an operator which, when applied to a Boolean variable x, creates its complement such that if x = True then /x = False, and if x = False then /x = True.

Given this basic set of rules, it is possible to derive any of the theorems in Figure 1-5. For example:

Theorem 1a: x + x = x

x + x	= (x + x)*True	by Postulate 1b
	= (x + x)*(x + /x)	by Postulate 2a
	= x + (x*/x)	by Postulate 4b
	= x + False	by Postulate 2b
	= x	by Postulate 1a

1.5.1 Duality

One of the most important properties of Boolean algebra is the *duality principle*. This principle states that any algebraic expression that may be deduced from the postulates of Boolean algebra has a dual which is also true. The dual of an expression is obtained by replacing all Trues with Falses, all Falses with Trues, all ANDs with ORs, and all ORs with ANDs. For example:

Theorem 2a: x + True = True

has the dual: x*False = False

which is theorem 2b. All postulates and theorems listed in Figure 1-5 are listed as pairs of duals. Of course, any of these theorems could also be derived without using the duality principle. For example:

Theorem 2b: x*False = False

x*False	= False + (x*False)	by Postulate 1a
	= (x*/x) + (x*False)	by Postulate 2b
	= x*(/x + False)	by Postulate 4a
	= x*/x	by Postulate 1a
	= False	by Postulate 2b

Postulate 1	(a) x + False = x (b) x*True = x
Postulate 2	(a) x + /x = True (b) x*/x = False
Postulate 3	(a) x + y = y + x (b) x*y = y*x
Postulate 4	(a) x*(y + z) = (x*y) + (x*z) (b) x + (y*z) = (x + y)*(x + z)
Theorem 1	(a) x + x = x (b) x*x = x
Theorem 2	(a) x + True = True (b) x*False = False
Theorem 3	/(/x) = x
Theorem 4	(a) x + (y + z) = (x + y) + z (b) x*(y*z) = (x*y)*z
Theorem 5	(a)/(x + y) = /x*/y (b)/(x*y) = /x + /y
Theorem 6	(a) x + (x*y) = x (b) x*(x + y) = x
Theorem 7	(a) (x*y) + (x*/y) = x (b)(x + y)*(x + /y) = x
Theorem 8	(a) x + (/x*y) = x + y (b) x*(/x + y) = x*y
Theorem 9	(x*y) + (/x*z) + (y*z) = (x*y) + (/x*z)

Figure 1-5. Postulates and Theorems of Boolean Algebra

1.5.2 Using Truth Tables

Finally, theorems may be demonstrated with truth tables. A theorem always holds true if it holds true for all cases. Since two variables can only have two values each, there are only four possible cases, so it is reasonable to look at a theorem on a case-by-case basis. For example, we can prove Theorem 5a with the following truth table:

x	y	/(x + y)	(/x*/y)
F	F	T	T
F	T	F	F
T	F	F	F
T	T	F	F

Figure 1-6

It can be seen from Figure 1-6 that, in every case, $/(x + y)$ is equal to $(/x*/y)$.

1.5.3 Complement of a Boolean Function

A *Boolean expression* is some mixture of Boolean variables and operators that has a value. For example:

$$x + y*z*/a$$

is a Boolean expression. A *Boolean function* is a statement in which two expressions are equated. For example:

$$a = b*c$$
$$/(c*d) = /c + /d$$

are Boolean functions. The difference is the presence of an equal sign. It is worth noting that 'equals', or equivalence, is also a Boolean function, since two expressions either are or are not equal. However, in this book we will attempt to present only true equations, so the Boolean value of an equal sign may be ignored in functions.

So far, we have talked about a Boolean expression's value as True or False. More frequently, these values are written as 1 and 0, with 1 standing for True, and 0 standing for False. From here on, we will also adopt this standard.

The complement of an expression may be written easily by placing the NOT operator in front of the enclosed expression:

$$/(x + y*z*/a),$$

but it is also possible to complement a function. The *complement* of a function is obtained by complementing both sides of an equation. For example, given the equation:

$$/a = b*c + 1$$

the complement would be:

$$/(/a) = /(b*c + 1)$$

which could be simplified:

$$\begin{aligned} a &= /(b*c + 1) && \text{by Theorem 3} \\ a &= /(b*c)*/1 && \text{by Theorem 5a} \\ a &= /(b*c)*0 && \text{def. of complement} \\ a &= 0 && \text{by Theorem 2b} \end{aligned}$$

Note the differences between obtaining the complement of a function, and obtaining the dual of a function. The complement is obtained by complementing the entire expression on both sides of the equation, and manipulating it from there with the given postulates and theorems. The dual of a function is obtained by replacing all 1's with 0's, all 0's with 1's, all ANDs with ORs, and all ORs with ANDs.

In fact, the easiest way in which to obtain the complement of a function is by taking the dual of the function and complementing each individual variable (called a *literal*). For example, the complement of:

$$F = (x + /y)*[W*(x + z)]$$

can be found by

1) taking the dual:

$$F^D = (x*/y) + [W + (x*z)]$$

2) complementing each literal:

$$/F = (/x*y) + [/W + (/x*/z)]$$

1.6 Algebraic Simplification

A *literal* is a complemented (/x) or uncomplemented (x) variable. A *term* is a subexpression, often enclosed in parentheses. The equation:

$$F = (x + /y)*/x$$

has three literals and two terms. *Simplifying* a Boolean equation is an attempt to minimize the number of literals or the number of terms in an equation. Unfortunately, in many situations, one can only be minimized at the expense of the other, so it is important to decide from the outset whether you are minimizing literals or terms. Literals can be minimized by repeated applications of the postulates and theorems of Boolean algebra (Table 1-6), but there is no algorithm; it is a trial and error process, and the result may not be unique. For example, the equation:

$$F = (x*/z) + [(x + y)*/z]$$

may be simplified through the following steps:

$$\begin{aligned} F &= (x*/z) + [(x + y)*/z] && \\ &= (/z*x) + [/z*(x + y)] && \text{Postulate 3b} \\ &= /z*[x + (x + y)] && \text{Postulate 4a} \\ &= /z*[(x + x) + y] && \text{Theorem 4a} \\ &= /z*(x + y) && \text{Theorem 1a} \end{aligned}$$

The equation is now simplified because there are no postulates or theorems which, when applied, will serve to reduce the number of items further.

1.6.1 Sum-of-Products and Product-of-Sums

When an equation is in the form:

$$F = (a*b) + (c*/a) + d$$

for example, it is said to be in *sum of products* form. This is because the equation is composed of a number of product terms (AND) that are summed (ORed) together. The subexpression resulting from two operands being ANDed together is referred to as a product because of the resemblance of the AND operator to the multiplication operator of arithmetic; the result of OR is referred to as a sum because of the resemblance of the OR operator to the addition operator of arithmetic.

When an equation is in the form:

$$F = (a + b)*(a + /c)$$

for example, it is said to be in *product of sums* form, because it is composed of a number of sum terms (OR) that are ANDed together. Both sum of products and product of sums forms are called *standard forms.*

1.6.2 Canonical Forms

If an equation has three variables that are complemented or uncomplemented, then there are a limited number of ways in which these variables can be ANDed or ORed together. Referring to Figure 1-7, under the column 'Minterms', and the subcolumn 'Term', there are eight different ways in which three variables could be ANDed together. Each combination has been given a name: the letter 'm' and a number. For example, the expression:

(/x*y*z) is m_3

while the expression:

(x*y*/z) is m_7

Minterms		Maxterms	
Term	Name	Term	Name
x*/y*/z	m_0	x + y + z	M_0
x*/y*z	m_1	x + y + /z	M_1
x*y*/z	m_2	x + /y + z	M_2
x*y*z	m_3	x + /y + /z	M_3
x*/y*/z	m_4	/x + y + z	M_4
x*/y*z	m_5	/x + y + /z	M_5
x*y*/z	m_6	x + /y + z	M_6
x*y*z	m_7	/x + /y + /z	M_7

Figure 1-7

Using these shorthand notations for expressions, we can refer to the equation:

F = (/x*y*/z) + (x*/y*/z) + (x*/y*z)

as:

F = $m_2 + m_4 + m_5$

which is much more compact. When an equation is expressed in terms of these named AND subexpressions, or *minterms* it is said to be in *sum-of-minterms* form.

Similarly, there are eight ways in which three variables may be ORed together. Such an OR subexpression is called a *maxterm.* The equation:

F = (x + y + z)*(x + /y + /z)*(/x + /y + /z)

could also be written as:

F = M_0*M_3*M_7

since each OR subexpression has been given a name consisting of an 'M' and a number (see the column 'Maxterms' in Figure 1-7). An equation expressed in this way is said to be written in *product-of-maxterms* form. Both sum of minterms and product of maxterms forms are called *canonical forms.*

In many equations, not every variable is represented in every term, but it is still possible to write them in canonical form. A little algebraic manipulation will produce the missing terms that are needed. For example, the equation:

F = (x*y*z) + (/x*y)

is missing a 'z' in its second term. In order to write this equation in sum of minterms form, we must first take the following steps:

F = (x*y*z) + (/x*y*1) Postulate 1b
= (x*y*z) + [/x*y*(z + /z)] Postulate 2a
= (x*y*z) + (/x*y*z) + (/x*y*/z) Postulate 4a
= $m_2 + m_3 + m_7$

To create a missing variable in a maxterm, use the duals of the postulates used above. To create more than one missing variable, expand the equation as many times as is needed by following the steps above.

1.6.3 Conversion Between Canonical Forms

Canonical forms do not only exist because they are more compact. With canonical forms, it is a trivial matter to invert an expression, or to convert between sum-of-product and product-of-sum representations.

Given the equation:

F = (/a*b*/c) + (a*/b*c) + (a* b*c)
= $m_2 + m_5 + m_7$

we can take its complement by forming an equation from all the minterms that are NOT present in the equation:

/F = $m_0 + m_1 + m_3 + m_4 + m_6$
= (/a*/b*/c) + (/a*/b*c) + (/a*b*c) + (a*/b*/c) + (a*b*/c)

Finally, using the dual/complement method, we can take the complement again. Of course, by Theorem 3 (Figure 1-5), anything that is complemented twice returns to its original value:

F = (a + b + c)*(a + b + /c)*(a + /b + /c)
(/a + b + c)(/a + b + /c)

We have now expressed function F, originally a sum of products, in product-of-sums form. Any Boolean equation can be written in either form.

Thus a quick way of doing this conversion is to write a product-of-maxterm equation using the maxterm numbers which did not appear in the original equation. In our example, we used the numbers 2, 5 and 7 for the sum-of-minterms form. In our product-of-maxterms form, we would use the maxterms 0, 1, 3, 4 and 6.

F = M_0*M_1*M_3*M_4*M_6
= (a + b + c)*(a + b + /c)*(a + /b + /c)
(/a + b + c)(/a + /b + c)

This works because each maxterm is the dual of the minterm that has the same number.

Of course, any equation written in the canonical forms can likely be simplified. After converting from standard form to canonical form, then converting from one canonical form to another, you may wish to simplify your equation.

1.7 Exclusive-OR (XOR) and Equivalence (XNOR)

There are two other frequently used operators which are really just special combinations of the AND, OR, and NOT operators.

The first is the Exclusive OR operator. Its symbol and truth table is shown in Figure 1-8. Note that the simple OR function is true even when both conditions are true, which is somewhat unlike our conversational use of the word 'OR'. The XOR function is more like our normal use of 'OR': one or the other condition is true, but not both.

The XOR function

X :+: Y

is equivalent to

x*/y + /x*y.

A	B	A :+: B1
0	0	0
0	1	1
1	0	1
1	1	0

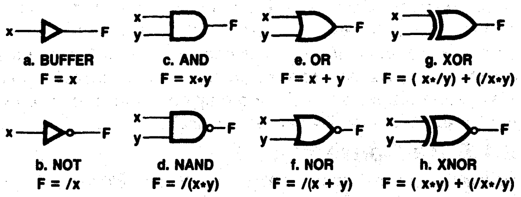
Figure 1-8. The XOR Operator

The Equivalence (XNOR) operators, symbols, and truth table are shown in Figure 1-9. It is true only when both conditions are the same. Thus XNOR is the complement of XOR
Here x :*: y is equivalent to x*y + /x*/y.

A	B	A :*: B
0	0	1
0	1	0
1	0	0
1	1	1

Figure 1-9. The XNOR Operator

1-8 Boolean Operator Summary



a. BUFFER
F = x

c. AND
F = x*y

e. OR
F = x + y

g. XOR
F = (x*/y) + (/x*y)

b. NOT
F = /x

d. NAND
F = /(x*y)

f. NOR
F = /(x + y)

h. XNOR
F = (x*y) + (/x*/y)

Figure 1-10. Logic Gates

X	Y	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11	F12	F13	F14	F15	F16
0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
0	1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
1	0	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
1	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
		0	*		X		Y	:+:	+		:*:	/Y		/X			1

Figure 1-11 Boolean Operators

2. Binary Systems

Binary numbers utilize a base 2 number system that consists of two digits: '0' and '1'. This number system is used in current digital computer systems because the outputs of most switching circuits can only be in one of two logical states. Also, when transistor circuits are only operating in one of two modes greater reliability can be obtained.

2.1 Base Conversion

Normally, decimal (base 10) numbers are written using a positional notation. In other words, the value of the number is determined by multiplying each digit by an appropriate power of 10 which is dependent on its relative position to the decimal point.

Example 2.1

$714.02 = 7 \times 10^2 + 1 \times 10^1 + 4 \times 10^0 + 0 \times 10^{-1} + 2 \times 10^{-2}$

2.1.1 Base-2 to Base-10 Conversion

Similarly, binary (base-2) numbers are also position-dependent relative to the binary point: each binary digit is multiplied by an appropriate power of 2 in order to obtain the decimal equivalent. The following example shows the conversion from a base-2 number to a base-10 number.

Example 2.2

$$\begin{aligned} 101.01_2 &= 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 + 0 \times 2^{-1} + 1 \times 2^{-2} \\ &= 4 + 0 + 1 + 0 + 1/4 \\ &= 5.25_{10} \end{aligned}$$

Notice that the binary point separates the positive and the negative powers of 2. This is similar to the case of the decimal point separating the positive and negative powers of 10.

2.1.2 Base-10 to Base-2 Conversion

Converting a base-10 integer to a base-2 integer requires utilizing the division method. To explain, let N represent the base-10 integer. Divide N by 2, since base-2 is desired. As a result, there should be a quotient, Q_0, and a remainder, R_0. Then divide Q_0 by 2 again and continue this process until the final quotient equals zero. The desired binary digits are the remainders resulting from each division step; the least significant bit starts with R_0.

Example 2.3

Converts 61_{10} to binary:

61/2 = 30	remainder = 1	LSB
30/2 = 15	remainder = 0	
15/2 = 7	remainder = 1	
7/2 = 3	remainder = 1	
3/2 = 1	remainder = 1	
1/2 = 0	remainder = 1	MSB

$61_{10} = 111101_2$

Converting decimal fractions to binary requires successive multiplications by 2. Let F be a decimal fraction. Multiply this number F by 2 and obtain an integer and a fraction result. Take the fraction and multiply once again by 2. Continue this process until the fraction becomes zero, or until a sufficient number of digits has been reached. The desired digits are the integer parts that were obtained at each multiplication step. The most significant digit is obtained first.

Example 2.4

Convert 0.375_{10} to binary

0.375	0.750	0.500	$I_0 = 0$	MSB
x 2	x 2	x 2	$I_1 = 1$	
0.750	1.500	1.000	$I_2 = 1$	LSB

$0.375_{10} = 0.011$

Note that if this procedure doesn't terminate, then the result must be a repeating fraction.

2.1.3 Base-2 to Base-8 or Base-16

To convert binary to *octal* (base-8) or vice versa is very simple and can be done by inspection. Each octal digit corresponds to three binary digits, since it can be in one of eight states (0 to 7). Therefore, the binary number should be divided into groups of three starting from the binary point. Each group on both sides of the binary point is replaced by an octal digit representation.

Example 2.5

$$\begin{aligned} 101110.011_2 &= 101 \quad 110 \;.\; 011 \\ &= 5 \quad 6 \;.\; 3_8 \end{aligned}$$

Similarly, binary to *hexadecimal* (base-16) and vice versa can also be done easily. This time, instead of three, the binary number is broken up into groups of four. The reason is because a hexadecimal digit can assume one of sixteen states (0 to 9, A, B, C, D, E and F). Again starting from the binary point, each group is replaced by its hexadecimal equivalent.

Example 2.6

$$\begin{aligned} 11100101.0011_2 &= 1110 \quad 0101 \;.\; 0011 \\ &= E \quad 5 \;.\; 3_{16} \end{aligned}$$

2.2 Simplicity of Binary Arithmetic

Due to the design of logic networks, it is much easier to do binary than decimal arithmetic in digital systems. Although binary arithmetic is implemented in about the same manner, the addition tables are much easier. Fortunately, numerical subtractions may be performed by addition operations between numbers. This property is of little use in the decimal system. However, much can be gained if used in the binary system. This is mainly due to the fact that in a binary system, complements of numbers are easily implemented, and the same hardware can be used for addition and subtraction operations. This allows for considerable savings in terms of system hardware design.

2.2.1 1's Complement

Finding the 1's complement of a binary number is easily done by inverting each digit (0 or 1).

Example 2.7

The 1's complement of:

01011.1101 = 10100.0010

2.2.2 Subtraction with 1's Complement

To subtract two positive binary numbers X and Y, (X-Y), the following procedures should be used:

1. Take the 1's complement of Y and add it to X.
2. Check results for overflow carry:
   a. If there is an overflow carry, add it to the least significant digit of the result.
   b. If there is no overflow carry, the result is negative. Then, complement this result and place a minus sign in front.

Example 2.8

```
a) 1010.11    -1000.01  = ?

               1010.11
              +0111.10   ←  1's complement of 1000.01
overflow 1     0010.01
              +      1   ←  add overflow carry
               0010.10   →  answer

b) 1001.10    - 1100.11  = ?

               1001.10
              +0011.00   ←  1's complement of 1100.11
no overflow    1100.10   →  -0011.01  →  answer
```

2.2.3 2's Complement

The most widely used numbering manipulation technique in current digital computers is the 2's complement method. This method is easily implemented with any decent computer instruction set. Using the same hardware for addition and subtraction in 2's complement makes system design simpler and can lead to savings in cost.

Finding the 2's complement of a binary number requires the following:

1. Take the logical complement by inverting each digit of the binary number.
2. Add 1 to the least significant digit.

Example 2.9

The 2's complement of 001100.01 is

```
step (1)  110011.10  ←  logical complement of 001100.01
step (2) +        1
          110011.11  →  answer
```

This technique can also be done by visual inspection. Start with the least significant digit of the number and visually scan to the left. Leave all digits unchanged until the first '1' is encountered. Then invert all the remaining digits to the left of this first '1'. Note that the binary point has no effect on this procedure.

2.2.4 Subtraction with 2's Complement

The steps for subtracting two binary numbers X and Y, (X-Y), are as follows:

1. Add X to the 2's complement of Y.
2. Check result for overflow carry:
   a. If there is an overflow carry, then throw it out. The result now represents (X-Y).
   b. If there is no overflow carry, the number is negative. Take the 2's complement of the result and place a negative sign in front of it.

Example 2.10

```
a) 1110.11 -1011.10 = ?

                  1110.11
overflow carry  + 0100.10  ←  2's complement of 1011.10
  throw out 1     0011.01  →  +0011.01  →  answer

b) 0001.11 -1000.10 = ?

                  0001.11
  no overflow   + 0111.10  ←  2's complement of 1000.10
        carry     1001.01  →  -0110.11  →  answer
```

Note that in computing systems which need to represent negative numbers, the MSB serves as a sign bit. When it is '1', the number is negative.

3. Karnaugh Maps

3.1 Karnaugh Map Technique

There exists a technique that allows the logic designer to minimize Sum of Product terms by utilizing *Karnaugh maps.* The Karnaugh map (sometimes referred to as *K-map*) graphically displays the implicants (*minterms*) of any sum-of-products expression in a matrix. It is derived directly from the truth table of this expression. K-maps are very useful for minimizing three, four, five and even six variable functions, but it gets too complicated beyond six. For expressions with more than six variables, the numerical manipulation should be done on a computer that uses a method such as the Quine-McCluskey method. This technique will not be discussed here.

3.1.1 Karnaugh Map Reading Procedure

Each minterm cell in the K-map has a value of '1' as determined by the truth table. Circle those single minterm cells that will combine with its adjacent cells to form larger groups of 1, 2, 4, 8, etc. If each single minterm cell is grouped individually, the map reading process should yield the original Sum of Product expression.

However, if two minterm cells are grouped together, at least one variable is dropped. This is because the theorem

$$X*Y + X*/Y = X$$

has been executed once. If a group of four adjacent minterm cells have been combined, then the theorem has been executed twice, and two variables are dropped. A group of eight adjacent cells result in three variables being dropped. Therefore, the main objective is to minimize the number of minterm cell groupings while maximizing the number of minterm cells in each grouping. By minimizing the number of cell groupings, the number of product terms is reduced. On the other hand, by maximizing the number of cells in each grouping, the number of literals in each product term is reduced.

For any product term, the variables which are included in the term are those whose values in the labels of the grouped cells are constant. The constant values give the polarity of the variables.

In example 3.1, product term 1 has cells with labels ABC = 010 and 110. B and C are constant here; B is 1 and C is 0, giving the product term B*/C.

3.1.2 Karnaugh Map Matrix Labels

In labeling the K-map matrix, the following rule should be followed:

Top to bottom or left to right:

Two-variable	Three-variable	Four-variable
00	000	Add a '0' MSB and use the three-variable chart for the first half. For the second half, add a '1' MSB and repeat the same chart in reverse order.
01	001	
11	011	
10	010	
	110	
	111	
	101	
	100	

Notice that the number of variables shown above is referring to one axis only (X or Y). However, this technique may be used for any number of variables that may be desired on each axis. For any axis greater than one variable, the second-half is a mirror image of the first-half with the MSB equal to a '1'. This can be seen above when comparing the three-variable list to the two-variable list.

3.1.3 Karnaugh Map Examples

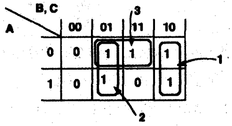
Examples of three- and four-variable K-maps are shown below. The corresponding truth tables for the examples are also shown to illustrate the derivation of the K-maps.

Example 3.1

Three-Variable K-map:

A	B	C	F
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	0

Truth Table



A \ B, C	00	01	11	10
0	0	1	1	1
1	0	1	0	1

Karnaugh Map

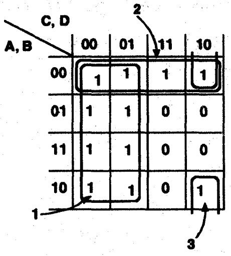
product term 1= B*/C
product term 2 = /B*C
product term 3 = /A*C
F = B*/C + /B*C + /A*C

Example 3.2

Four-Variable K-map:

A	B	C	D	F
0	0	0	0	1
0	0	0	1	1
0	0	1	0	1
0	0	1	1	1
0	1	0	0	1
0	1	0	1	1
0	1	1	0	0
0	1	1	1	0
1	0	0	0	1
1	0	0	1	1
1	0	1	0	1
1	0	1	1	0
1	1	0	0	1
1	1	0	1	1
1	1	1	0	0
1	1	1	1	0

Truth Table



A, B \ C, D	00	01	11	10
00	1	1	1	1
01	1	1	0	0
11	1	1	0	0
10	1	1	0	1

Karnaugh Map

product term 1 = /C
product term 2 = /A*/B
product term 3 = /B*C*/D
F = /C + /A*/B + /B*C*/D

4. Combinatorial Logic

4.1 Logic Design Introduction

Logic design is a combination of analysis, synthesis, minimization and implementation of Boolean functions. Boolean functions must originally come from worded statements. This is a very important part of logic design because the worded statement can be ambiguous and imprecise, while the Boolean equation must be unambiguous and exact. The conversion of words to equations is called synthesis. Engineers must be careful when synthesizing a problem because many times the originator of a problem is not a technical person. It is the responsibility of the logic designer to review the synthesis of the problem with the originator to make sure the solution is suitable.

4.2 Combinatorial Design

Combinatorial logic is a network whose output is solely dependent upon its inputs. It has no feedback loops or memory elements.

The first step in combinatorial design is to analyze the problem and then define it in an exact manner. This will make synthesizing a Boolean equation much easier.

Synthesis usually takes several steps. Using truth tables and K-maps are common ways of specifying a problem and putting it in the minimal Boolean form.

Example 4.1

A seven-segment decoder decodes a BCD number and turns on the appropriate segments of a seven-segment digit. Given the seven-segment digit in Figure 4-1, develop a minimal equation for each segment by using a truth table and K-maps.

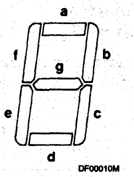
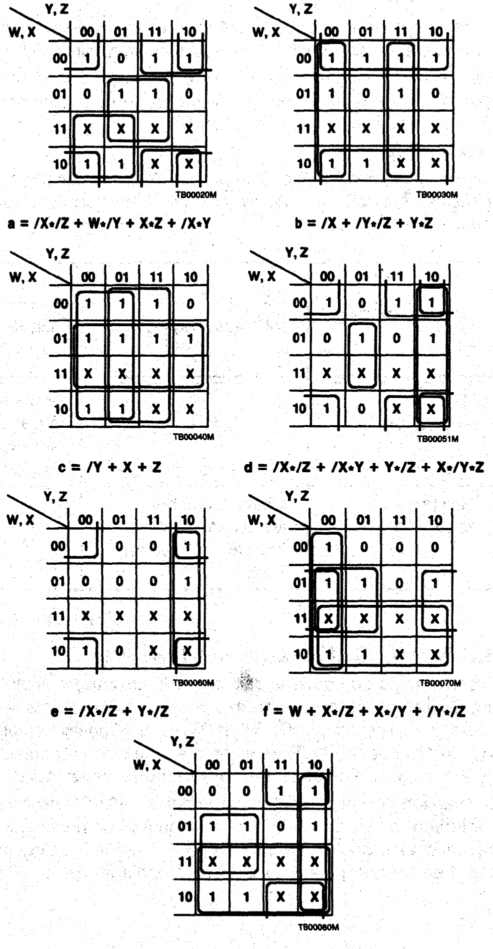


Figure 4-1. Seven-Segment Decoder

Forming a truth table from Figure 4-1 is done by writing all '10' possible inputs down, then determining which segments should be activated for each input. For example, segment 'a' is activated whenever; 2, 3, 5, 7, 8, 9 or 0 is input to the decoder. Once the table is formed, a K-map can be made for each segment. The K-maps are used to derive a Sum of Products logic equation for each segment.

K-maps are an excellent way of forming equations when three to six variables are involved in a problem. Either of the two standard algebraic forms of the function (sum-of-products or product-of-sums) can be derived. A network of AND and OR gates is obtained directly from either form.

DECIMAL	INPUTS				OUTPUTS						
	W	X	Y	Z	a	b	c	d	e	f	g
0	0	0	0	0	1	1	1	1	1	1	0
1	0	0	0	1	0	1	1	0	0	0	0
2	0	0	1	0	1	1	0	1	1	0	1
3	0	0	1	1	1	1	1	1	0	0	1
4	0	1	0	0	0	1	1	0	0	1	1
5	0	1	0	1	1	0	1	1	0	1	1
6	0	1	1	0	0	0	1	1	1	1	1
7	0	1	1	1	1	1	1	0	0	0	0
8	1	0	0	0	1	1	1	1	1	1	1
9	1	0	0	1	1	1	1	0	0	1	1



W, X \ Y, Z	00	01	11	10
00	1	0	1	1
01	0	1	1	0
11	X	X	X	X
10	1	1	X	X

a = /X•/Z + W•/Y + X•Z + /X•Y

W, X \ Y, Z	00	01	11	10
00	1	1	1	1
01	1	0	1	0
11	X	X	X	X
10	1	1	X	X

b = /X + /Y•/Z + Y•Z

W, X \ Y, Z	00	01	11	10
00	1	1	1	0
01	1	1	1	1
11	X	X	X	X
10	1	1	X	X

c = /Y + X + Z

W, X \ Y, Z	00	01	11	10
00	1	0	1	1
01	0	1	0	1
11	X	X	X	X
10	1	0	X	X

d = /X•/Z + /X•Y + Y•/Z + X•/Y•Z

W, X \ Y, Z	00	01	11	10
00	1	0	0	1
01	0	0	0	1
11	X	X	X	X
10	1	0	X	X

e = /X•/Z + Y•/Z

W, X \ Y, Z	00	01	11	10
00	1	0	0	0
01	1	1	0	1
11	X	X	X	X
10	1	1	X	X

f = W + X•/Z + X•/Y + /Y•/Z

W, X \ Y, Z	00	01	11	10
00	0	0	1	1
01	1	1	0	1
11	X	X	X	X
10	1	1	X	X

g = W + X•/Y + /X•Y + Y•/Z

Figure 4-2

ROW	A	B	C	MINTERMS	MAXTERMS
0	0	0	0	/A*/B*/C = m0	A + B + C = M0
1	0	0	1	/A*/B*C = m1	A + B + /C = M1
2	0	1	0	/A*B*/C = m2	A + /B + C = M2
3	0	1	1	/A*B*C = m3	A + /B + /C = M3
4	1	0	0	A*/B*/C = m4	/A + B + C = M4
5	1	0	1	A*/B*C = m5	/A + B + /C = M5
6	1	1	0	A*B*/C = m6	/A + /B + C = M6
7	1	1	1	A*B*C = m7	/A + /B + /C = M7

Figure 4-3 Minterm and Maxterm Expansions for 3 Variables

Since a minterm is a product term, an unminimized sum-of-products expression may also be written with minterms.

Each maxterm is a sum of variables. It is derived by solving a K-map for the *0-terms* instead of the *1-terms*. Maxterms are used in a product-of-sums solution.

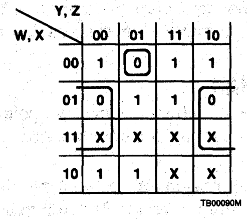
Example 4.2

Rewriting the Sum of Products equations from example 4.1 in minterm form can be done by inspecting the truth table or K-map.

$a = \Sigma m\ (0, 2, 3, 5, 7, 8, 9, 10, 11, 12, 13, 15)$
$b = \Sigma m\ (0, 1, 2, 3, 4, 7, 8, 9, 10, 11, 12, 15)$
$c = \Sigma m\ (0, 1, 3, 4, 5, 6, 7, 8, 9, 11, 12, 13, 14,15)$
$d = \Sigma m\ (0, 2, 3, 5, 6, 8, 10, 11, 13)$
$e = \Sigma m\ (0, 2, 6, 8, 10, 14)$
$f = \Sigma m\ (0, 4, 5, 6, 8, 9, 10, 11, 12, 13, 14, 15)$
$g = \Sigma m\ (2, 3, 4, 5, 6, 8, 9, 10, 11, 12, 13, 14,15)$

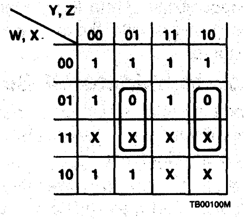
Example 4.3

We can easily rework the first two K-maps from Example 4.1 to get a maxterm solution.



W, X \ Y, Z	00	01	11	10
00	1	0	1	1
01	0	1	1	0
11	X	X	X	X
10	1	1	X	X

TB00090M



W, X \ Y, Z	00	01	11	10
00	1	1	1	1
01	1	0	1	0
11	X	X	X	X
10	1	1	X	X

TB00100M

/a = X*/Z + /W*/X*/Y*Z
a = (/X + Z)*(W + X + Y + /Z)
a = ΠM (1, 4, 6, 12, 14)

/b = X*/Y + X*/Z
b = (/X + Y)*(/X + Z)
b = ΠM (5, 6, 13, 14)

Given either camonical form, it is a simple matter of converting to the other form, or the inverse of either form.

GIVEN FORM	DESIRED FORM			
	Minterm expansion of F	**Maxterm expansion of F**	**Inverted Minterm expansion of F**	**Inverted Maxterm expansion of F**
Mintern expansion of F	—	Maxterm numbers are those numbers not on the minterm list for F	List minterms not present in F	Maxterm numbers are the same as minterm numbers of F
Maxterm expansion of F	Minterm numbers are those numbers not on the maxterm list for F	—	Minterm numbers are the same as maxterm numbers of F	List maxterms not present in F

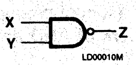
Figure 4-4 Conversion of Forms Table

4.3 NAND Gates and NOR Gates

A set of logic operators is said to be functionally complete if any Boolean function can be expressed in terms of this set of operations. The set {AND, OR, NOT} is functionally complete. The NAND and the NOR gates are each functionally complete by themselves. Therefore they are called *universal* gates.

Conversion of AND and OR networks to NAND networks is carried out by starting with a minimal sum of products expression and then applying the theorem; F = /(/F). This equation can then be manipulated using DeMorgan's theorem.

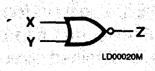
X	Y	Z
0	0	1
0	1	1
1	0	1
1	1	0



LD00010M

Z = /(A*B)

X	Y	Z
0	0	1
0	1	0
1	0	0
1	1	0



LD00020M

Z = /(X + Y)

Figure 4-5. Truth Tables

Example 4.4

From the K-map in Figure 4-6, we can find equations for an AND-OR, NAND-NAND, OR-NAND and NOR-OR networks.

A, B \ C, D	00	01	11	10
00	1	0	1	0
01	1	0	1	0
11	1	1	1	1
10	0	0	0	0

TB00110M

Figure 4-6. Karnaugh Map

F = A*B + /A*/C*/D + /A*C*D — AND-OR
= /[/(A*B + /A*/C*/D + /A*C*D)]
= /[/(A*B)*/(/A*/C*/D) + /(/A*C*D)] — NAND-NAND
= /[/A + /B)*(A + C + D)*(A + /C + /D)] — OR-NAND
= /(/A + /B) + /(A + C + D) + /(A + /C + /D) — NOR-OR

In order to get a network of NOR gates we must start with the minimum product-of-sums form of F.

Example 4.5

From the same K-map, we can also find equations for OR-AND, NOR-NOR, AND-NOR and NAND-AND networks.

F = (/A + B)*(A + C + /D)*(A + /C + D) — OR-AND
= /[/(/A + B) + /(A + C + /D) + /(A + /C + D)] — NOR-NOR
= /(A*/B + /A*/C*D + /A*C*/D) — AND-NOR
= /(A*/B)*/(/A*/C*D)*/(/A*C*/D) — NAND-AND

NAND-NAND and NOR-NOR networks are very common in industry because both the NAND and NOR gates are universal gates. Thus, these gates are made in great quantities, making them more available for designers.

A NAND-NAND network is made from a Sum of Products (SOP) solution. The AND and OR gates of the SOP solution are replaced by NAND gates with all the interconnections staying the same. Variables that are input directly to the output gate must be inverted.

A NOR-NOR network is made from a Product of Sums solution. The OR and AND gates are replaced by NOR gates with all interconnections staying the same. Any variables that are input directly to the output NOR gates must be inverted.

An easy way of forming either a NAND network from a Sum of Products solution or a NOR network from a Product of Sums solution is to place two inversion bubbles in series between the two levels as demonstrated in Figure 4-7.

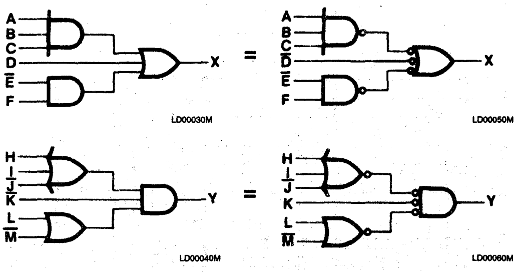


Figure 4-7. Network Conversion

4.4 Multiplexers

Multiplexers are circuits which select one of 2^n input lines using n selector lines. For example, an eight-input multiplexer (often called a 'MUX') selects one of 2^3 input lines using three select lines.

Example 4.6

Design an 8:1 multiplexer in SOP form by using a truth table.

SELECT			MULTIPLEXER INPUTS								OUTPUT
A	B	C	D0	D1	D2	D3	D4	D5	D6	D7	Y
0	0	0	0	X	X	X	X	X	X	X	0
0	0	0	1	X	X	X	X	X	X	X	1
0	0	1	X	0	X	X	X	X	X	X	0
0	0	1	X	1	X	X	X	X	X	X	1
0	1	0	X	X	0	X	X	X	X	X	0
0	1	0	X	X	1	X	X	X	X	X	1
0	1	1	X	X	X	0	X	X	X	X	0
0	1	1	X	X	X	1	X	X	X	X	1
1	0	0	X	X	X	X	0	X	X	X	0
1	0	0	X	X	X	X	1	X	X	X	1
1	0	1	X	X	X	X	X	0	X	X	0
1	0	1	X	X	X	X	X	1	X	X	1
1	1	0	X	X	X	X	X	X	0	X	0
1	1	0	X	X	X	X	X	X	1	X	1
1	1	1	X	X	X	X	X	X	X	0	0
1	1	1	X	X	X	X	X	X	X	1	1

Figure 4-8. Truth Table for 8:1 Multiplexer

As can be seen from the truth table A, B and C select one of the eight multiplexer inputs to appear on the output, Y. If A, B and C = 011, then Figure 4-9 shows that the D3 AND gate will be enabled while all the other AND gates will be disabled. This allows D3 to be 'ORed' with seven zeros and thus end up on the output Y.

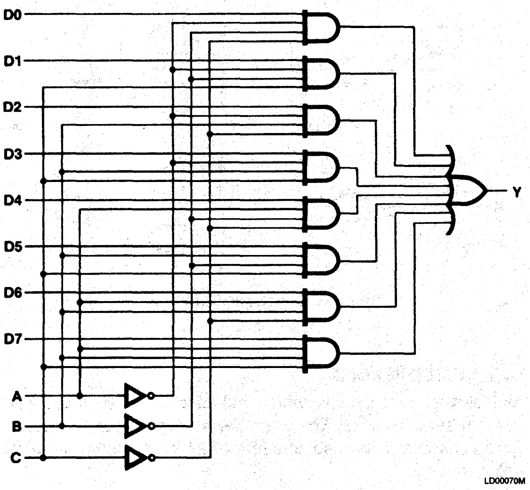


Figure 4-9. 8:1 Multiplexer Circuit

Example 4.7

We will design a dual 8:1 mux in a PAL device.

When selecting a PAL device, several things must be considered. Will the design need registers? How many inputs and outputs are there? Are the outputs active high or active low? For a dual 8:1 mux the select lines will be shared, but the eight data inputs to each mux are independent. Thus we need nineteen inputs and two outputs for the design. This narrows our choices down to one PAL device, the PAL20L2. The output of the PAL20L2 is active low but this causes no problems because an active high output will result by simply inverting all the data inputs. The Boolean equations are shown below; the complete logic diagram is on the next page.

Multiplexers have been widely used as logic devices as well as selector circuits. A 4:1 mux can be used to realize any three-variable function. An 8:1 mux can realize any four-variable function.

Example 4.8

Solve the K-map in Figure 4-10 and build the circuit with an 8:1 multiplexer.

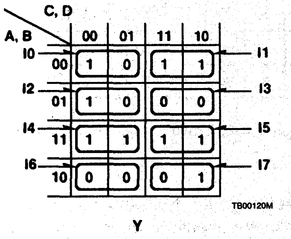


Figure 4-10. Eight One-Variable Karnaugh Maps

A, B and C are used as control inputs to the multiplexer, this leaves D as the only real variable in the problem. The 16-square K-map can thus be broken up into eight one-variable K-maps. Each map is solved for one of the eight data inputs to the 8:1 multiplexer. The solution is shown in Figure 4-11.

A	B	C	
0	0	0	I0 = /D
0	0	1	I1 = 1
0	1	0	I2 = /D
0	1	1	I3 = 0
1	0	0	I4 = 1
1	0	1	I5 = 1
1	1	0	I6 = 0
1	1	1	I7 = /D

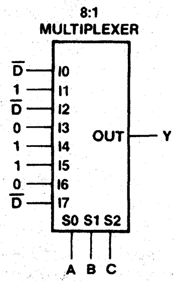


Figure 4-11

4.5 Decoders

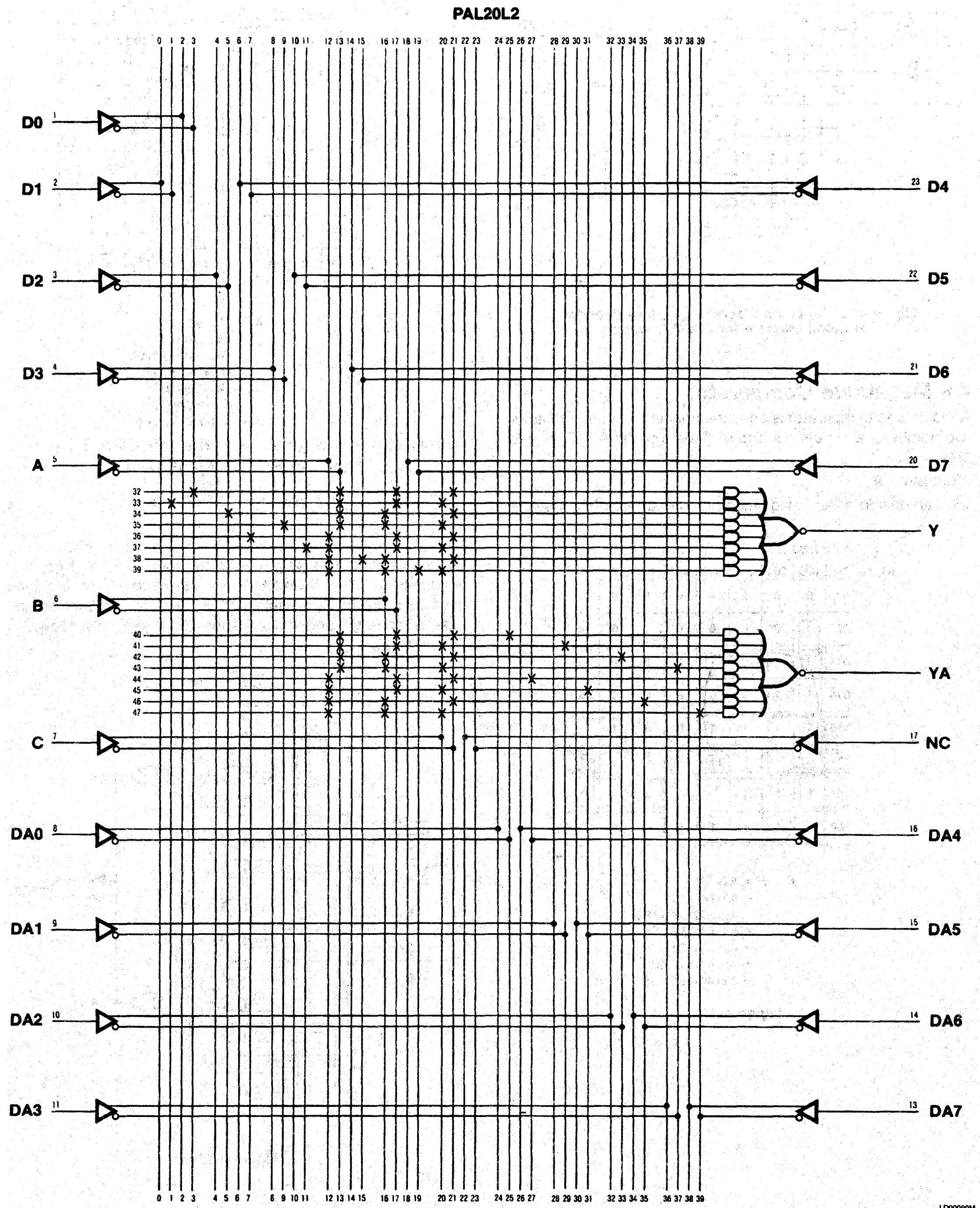
On a multiplexer with n address lines, one of the 2^n inputs is selected to be output. On a decoder with n address lines, one of the 2^n output lines is forced either high or low, depending on the design of the decoder. Figure 4-12 shows a truth table for an active high 3-to-8 decoder.

SELECT LINES			OUTPUT LINES							
A	B	C	f	g	h	i	j	k	l	m
0	0	0	1	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0	0	0	0
0	1	0	0	0	1	0	0	0	0	0
0	1	1	0	0	0	1	0	0	0	0
1	0	0	0	0	0	0	1	0	0	0
1	0	1	0	0	0	0	0	1	0	0
1	1	0	0	0	0	0	0	0	1	0
1	1	1	0	0	0	0	0	0	0	1

Figure 4-12. Decoder Truth Table

A decoder will have as many outputs as there are possible binary input combinations. It can be seen from Figure 4-12 that only one output can be equal to 1 at any time. The active output represents the minterm combination that was input to the decoder. It can also be noticed from Figure 4-12 that there is not a combination of inputs that will give all 0's on the outputs. Many designs actually need to be able to make all outputs inactive; this can be done simply by putting an enable line in all of the output AND gates. The logic design and block diagram for the 3-bit decoder in Figure 4-12 appears in Figure 4-13.

Logic Diagram



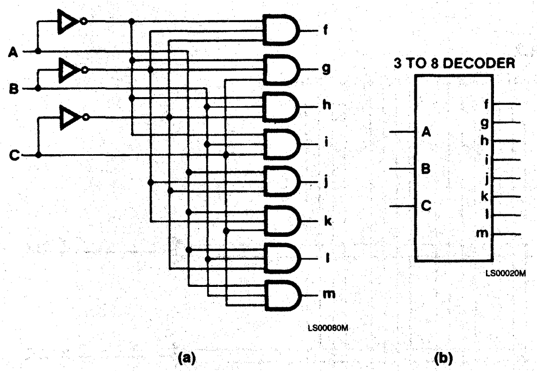


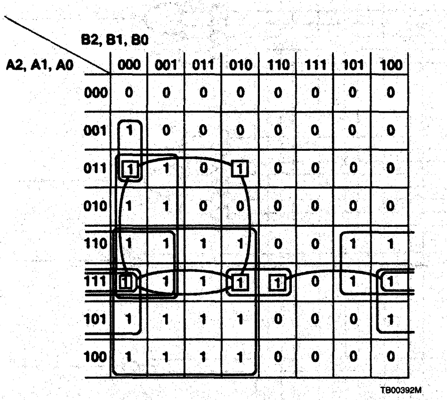
Figure 4-13. (a) Logic Diagram for 3-to-8 Decoder
(b) Block Diagram for 3-to-8 Decoder

4.6 Magnitude Comparator

A magnitude comparator is combinatorial circuit that compares two numbers, and puts out one of three signals: A > B, A = B or A < B.

Example 4.9

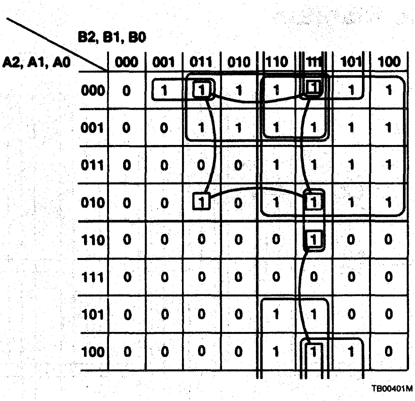
We can design a 3-bit magnitude comparator in a PAL device.



A2, A1, A0 \ B2, B1, B0	000	001	011	010	110	111	101	100
000	0	0	0	0	0	0	0	0
001	1	0	0	0	0	0	0	0
011	1	1	0	1	0	0	0	0
010	1	1	0	0	0	0	0	0
110	1	1	1	1	0	0	1	1
111	1	1	1	1	1	0	1	1
101	1	1	1	1	0	0	0	1
100	1	1	1	1	0	0	0	0

A > B = A2*/B2
+ A1*/B2*/B1
+ A0*/B2*/B1*/B0
+ A1*A0*/B2*/B0
+ A2*A1*/B1
+ A2*A1*A0*/B0
+ A2*A0*/B1*/B0

Figure 4-14



A2, A1, A0 \ B2, B1, B0	000	001	011	010	110	111	101	100
000	0	1	1	1	1	1	1	1
001	0	0	1	1	1	1	1	1
011	0	0	0	0	1	1	1	1
010	0	0	1	0	1	1	1	1
110	0	0	0	0	0	1	0	0
111	0	0	0	0	0	0	0	0
101	0	0	0	0	1	1	0	0
100	0	0	0	0	1	1	1	0

B > A = /A2*B2
+ /A2*/A1*B1
+ /A2*/A1*/A0*B0
+ /A2*/A0*B1*B0
+ /A1*B2*B1
+ /A0*B2*B1*B0
+ /A1*/A0*B2*B0

Of course, A = B only if not A < B and not A > B

(A = B) = /(B < A)*/(B > A)

Figure 4-14 (Continued)

The six-variable K-maps are used to produce Sum of Product equations for A > B and B < A. These equations are then used to form the two-level logic diagram of the 3-bit magnitude comparator and the equation for A = B, as shown in Figure 4-15.

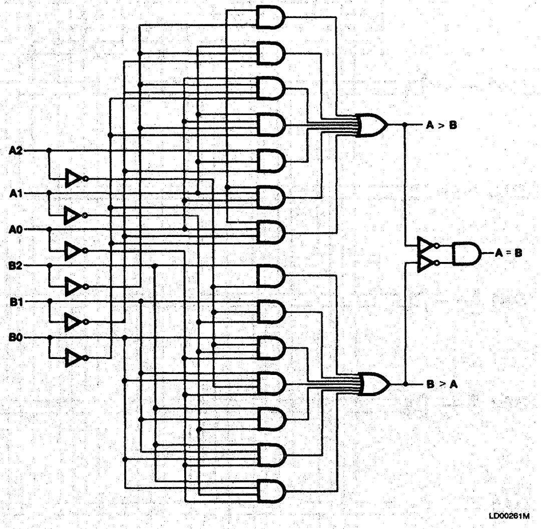


Figure 4-15

The logic diagram of the 3-bit comparator shows that there are six inputs and three outputs in the circuit. Each output is derived from at most seven product terms. This design can fit into a portion of a PAL16P8, as shown on page 8-18. Note that 5 outputs remain, which can be used for some other functions if needed.

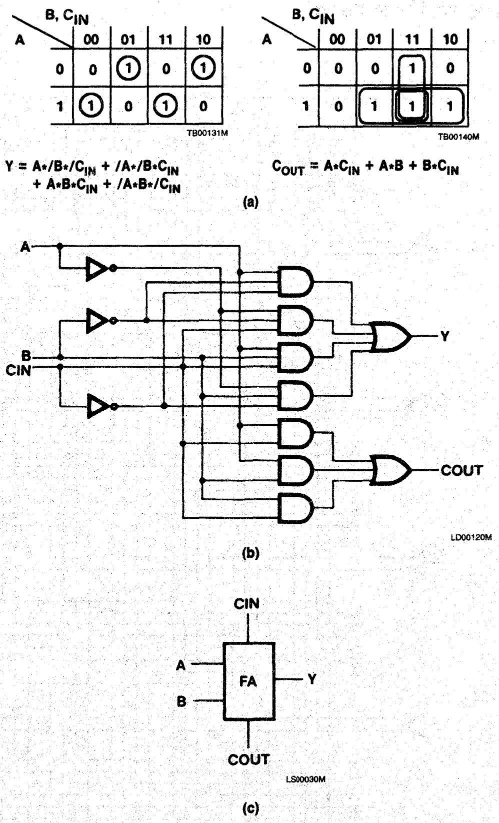
4.7 Adder

A binary adder takes two binary inputs, adds them, and generates the binary sum. A full adder is the basic building block of any adding network. A full adder is a 1-bit adder with a carry-in and a carry-out. The truth table is shown in Figure 4-16. The logic design and block diagram appear in Figure 4-17.

A	B	C_{IN}	Y	C_{OUT}
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Figure 4-16. Truth Table for Full Adder

The truth table is used to form K-maps for the outputs Y and C_{in}. These simple K-maps are solved to obtain equations for Y and C_{in}.



$Y = A{*}/B{*}/C_{IN} + /A{*}/B{*}C_{IN} + A{*}B{*}C_{IN} + /A{*}B{*}/C_{IN}$

$C_{OUT} = A{*}C_{IN} + A{*}B + B{*}C_{IN}$

(a)

Figure 4-17. (a) Karnaugh Maps for the Full Adder, (b) Logic Diagram, (c) Block Diagram

A parallel 4-bit adder can be designed using four full adders.

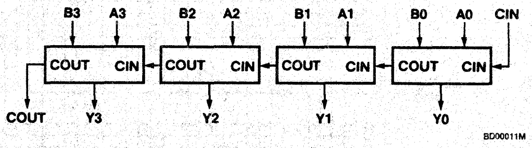
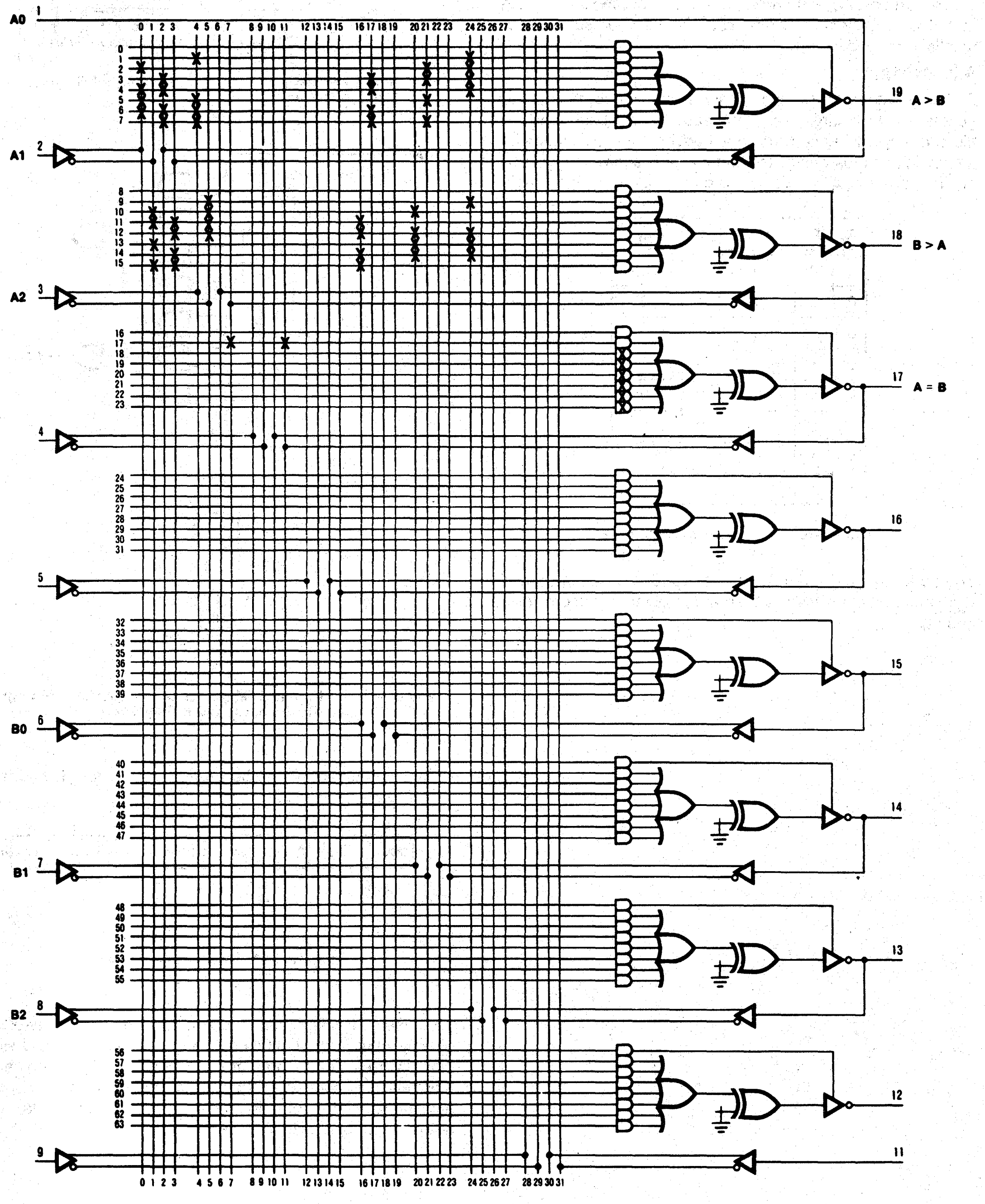


Figure 4-18. Parallel 4-Bit Adder

To implement this circuit in a PAL device, each carry-out is directly input to the next digit's carry-in. Nine inputs and eight outputs are needed. Three of the outputs (the first three carry-outs) are only needed so they can be fed back into the circuit as inputs. A PAL16L8 is the perfect PAL device for this design. The logic diagram is shown on page 8-19.

Logic Diagram

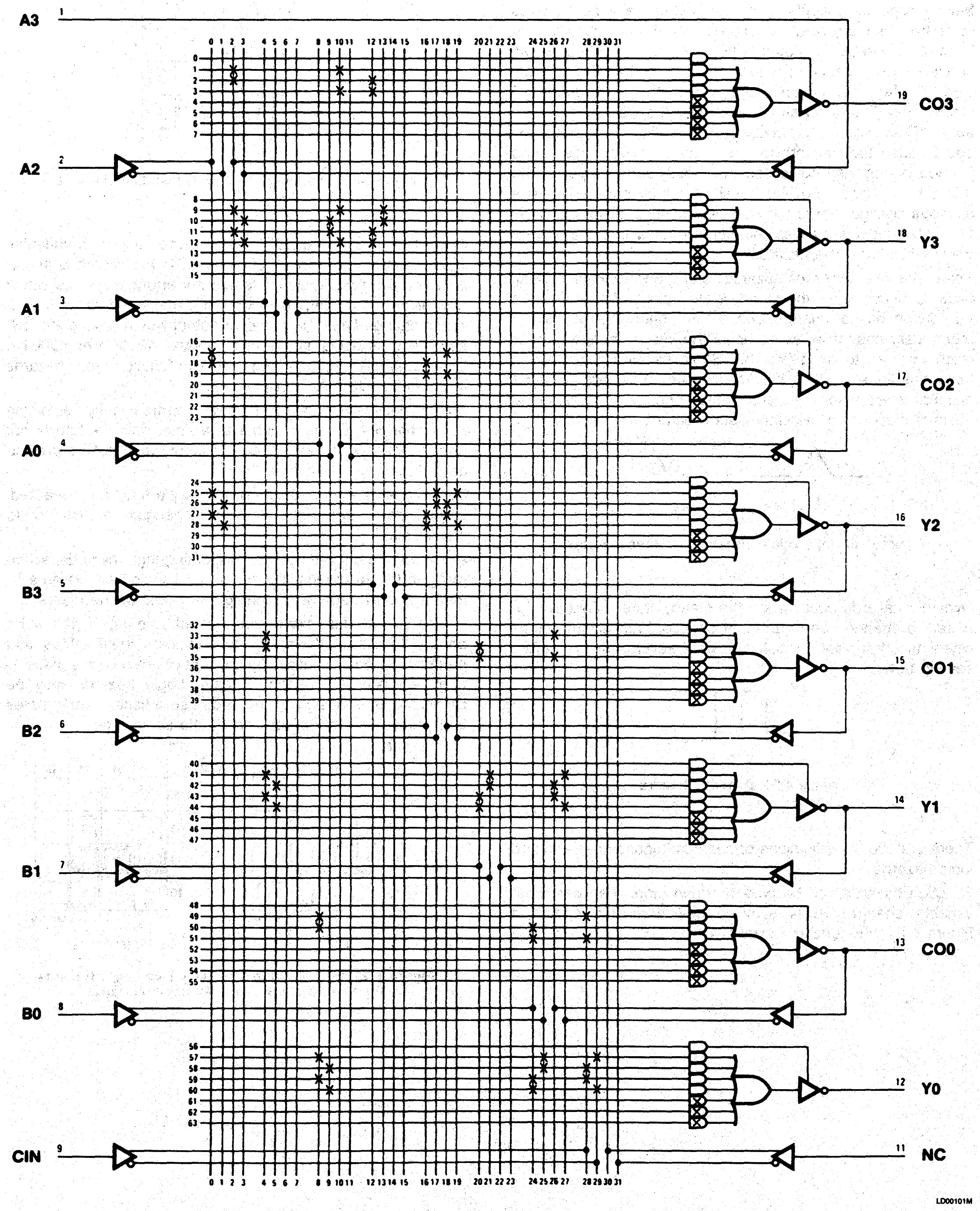
PAL16P8



LD00290M

Logic Diagram

PAL16L8



4.8 Hazards

Even though a digital network is designed correctly, it still may have erroneous outputs at times due to *hazards*. Hazards exist because physical circuits do not behave ideally. For example, a D-type flip-flop has two outputs; Q and /Q, which should always be complements of each other. In the real world Q may be switching from 1 to 0 and /Q from 0 to 1. Unless both Q and /Q switch at exactly the same time, Q will equal /Q for some finite amount of time. In some cases this could cause the network to malfunction. The change in the flip-flop output may not cause the steady-state output of the network to change, but the transient output may have had a spurious change due to the non-ideal flip-flop. If, for instance, the network's output was the set line of a latch, the latch would set due to the hazard.

There are two types of hazards, *static* and *dynamic*. Static hazards occur when the steady-state output of a network is not supposed to change due to an input change, but a momentary change does occur as the inputs change. This is often referred to as a "glitch." Static hazards are qualified further as either static 1 hazards or static 0 hazards. Static 1 hazards exist when the steady-state output is 1, static 0 hazards exist when the steady-state output is 0.

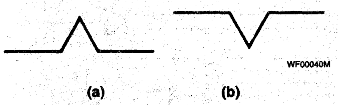


Figure 4-19. (a) Static 0 Hazard, (b) Static 1 Hazard

Dynamic hazards occur when the steady-state output is supposed to change due to an input change. The hazard occurs when the transient output changes several times before settling down.

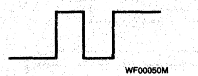
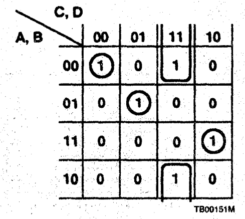


Figure 4-20. Dynamic Hazard

There are two classifications of hazards: function hazards and logic hazards.

Function hazards can be present when more than one input variable changes. It is easy to see from the K-map in Figure 4-21 why function hazards exist.



A, B \ C, D	00	01	11	10
00	1	0	1	0
01	0	1	0	0
11	0	0	0	1
10	0	0	1	0

Figure 4-21. Karnaugh Map with Function Hazards

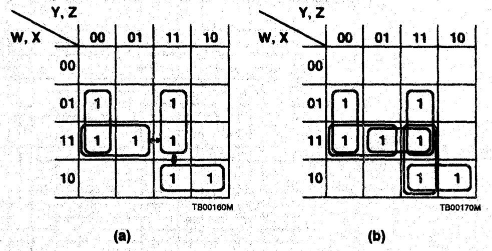
A function static 1 hazard is present when the input variables A, B, C and D go from <0000> to <0101>. If both B and D changed simultaneously no temporary erroneous pulse would appear on the output; however in the real world either B or D would change first. The transient output would have gone to 0 as a result of being momentarily in state <0100> or <0001>. Looking at the K-map, it is easy to see function static hazards and function dynamic hazards.

The easiest way to avoid function hazards is by restricting input changes to one variable at a time. This method is not always possible though, since the inputs may not be predictable.

Logic hazards exist because of the way a function is realized. Logic hazards can exist even if input changes are restricted to one variable at a time.

A K-map is a very good way of locating logic hazards. When trying to locate the static 0 and static 1 logic hazards on a K-map it is only necessary to map the *1-sets* or the *0-sets*.

A 1-set is a product expression derived from a grouping of 1's on the K-map. If there are two adjacent input states that produce a 1 on the output, but are not covered by the same 1-term, a static logic hazard exists. Logic hazards may be eliminated by redesigning the circuit so adjacent input states that produce ones are covered by the same 1-term.



(a)

W, X \ Y, Z	00	01	11	10
00				
01	1		1	
11	1	1	1	
10			1	1

(b)

W, X \ Y, Z	00	01	11	10
00				
01	1		1	
11	1	1	1	
10			1	1

Figure 4-22. (a) Karnaugh Map with Two Logic Hazards
(b) Karnaugh Map with No Logic Hazards

5. Sequential Logic

5.1 Introduction

In the previous section, combinatorial circuits were discussed — circuits whose outputs are determined completely by their present inputs. Outputs of some networks depend not only on their present inputs but also on the sequence of their past inputs. These circuits are called *sequential* switching networks. Sequential networks must be able to remember the past sequence of their inputs in order to be able to produce new outputs.

In order for a sequential circuit to remember the previous inputs, it must retain those values in some memory elements. The most basic memory element is called a *flip-flop.* Flip-flops are bistable devices with one or more inputs that affect their outputs. The term *bistable* means that the outputs are stable in either of two states.

There are two types of flip-flops: *unclocked* and *clocked.* Unclocked flip-flops are often referred to as *latches.* Clocked flip-flops are often grouped into *registers.*

5.2 Unclocked Flip-Flops – Latches

5.2.1 S-R Latch

The circuit in Figure 5-1a is called a *SET-RESET* or an S-R latch. An S-R latch has two inputs which are used to control the state of the latch. The rules for this type of latch are:

1. If SET = RESET = 0, then the latch remains in the same state, and the output does not change.
2. A '1' on the SET input and a '0' on the RESET input will make the latch SET to '1'.
3. A '0' on the SET input and a '1' on the RESET input causes the latch to RESET to '0' state.
4. If SET = RESET = 1, then Q and /Q will be '0' at the same time, which is meaningless. When designing with an S-R latch, we should remember that SET = RESET = 1 is forbidden.

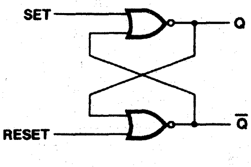
The S-R latch circuit, *state table, characteristic equation* and *waveforms* are shown in figure 5-1.

Latches can have more than two inputs. In Figure 5-2, we examine a latch circuit that has two SET terms instead of one. The logic diagram and the transition table are shown.

In some applications using latches, it is desirable for the input data to be effective only when another signal — usually referred to as a control signal — is active. For these applications, the S-R latch could be modified as shown in Figure 5-3.

It is apparent from Figure 5-3 that the values of SET and RESET are effective only when the control signal (C) is active. When C = 0, changes in the SET and RESET inputs do not have any effect on the output.

Of course in the input of the latch does not affect the output immediately; there is a short delay for this change to appear at the output. This delay, shown in figure 5-3(b), is caused because of the propagation delays of the gates between inputs and outputs.



(a)

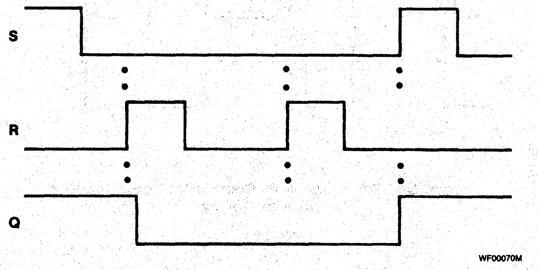
S_t	R_t	Q_t	Q_{t+1}
0	0	0	0
0	0	1	1
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	X
1	1	1	X

(b)

S_t, R_t \ Q_7	0	1
00	0	1
01	0	0
11	X	X
10	1	1

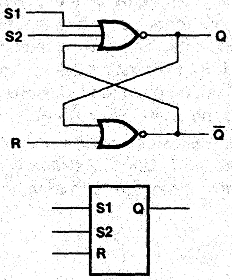
$Q_{t+1} = S_t + Q_t{*}/R_t$
(where $S_t{*}R_t = 0$)

(c)



(d)

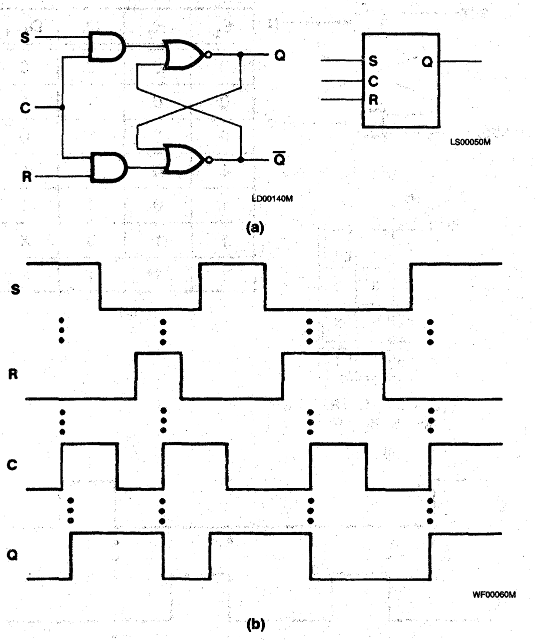
Figure 5-1. S-R Latch (a) Logic Circuit (b) State Table (c) Characteristic Equation (d) Waveforms



S_1	S_2	R	Q_{t+1}
0	0	0	Q_t
1	x	0	1
x	1	0	1
0	0	1	0
x	1	1	x
1	x	1	x

(a) (b)

Figure 5-2. S-R Latch with two SET inputs (a) Logic diagram (b) State Table



(a)

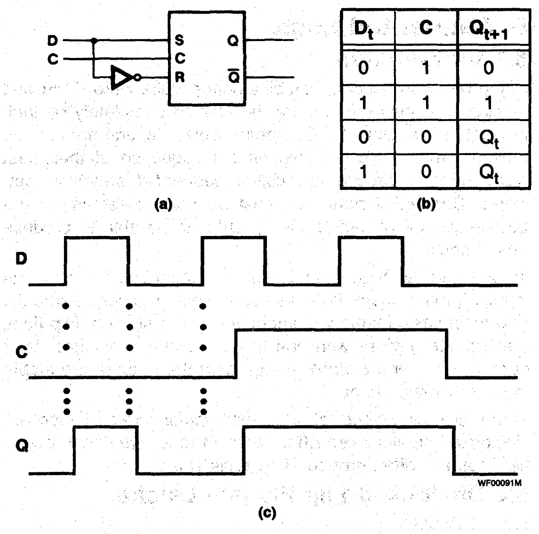
(b)

Figure 5-3. S-R Latch with Control (a) Logic Diagram (b) Waveforms

5.2.2 D-type Latch

Other kinds of latches are used in sequential circuits. One of the most popular latches is called a *delay latch*, or D-type latch. An S-R latch is modified to a D-type latch by inserting an inverter between S and R, and calling the input "D" instead of "S". The D-type latch will take the value of its input and transfer it to the output. The advantage of the D-type latch over the S-R latch is that in the former only one input is needed and there is no forbidden state. The only disadvantage of the D-type latch is that it does not have a "no change" condition. This condition can be provided by inserting a control signal, C, as an input to the latch (Figure 5-4).

This configuration is probably the most widely used one, with the control input commonly called the "Gate". Devices with active high gates and with active low gates are both commercially available.



(a)

D_t	C	Q_{t+1}
0	1	0
1	1	1
0	0	Q_t
1	0	Q_t

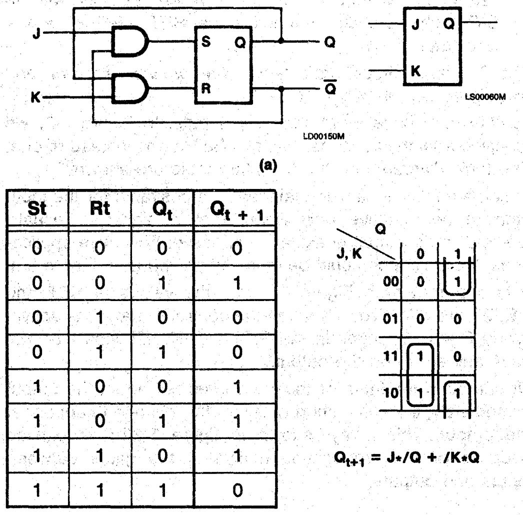
(b)

(c)

Figure 5-4. D-type Latch with Gate (a) Logic Circuit (b) State Table (c) Waveforms

5.2.3 J-K Latch

Another useful latch is the J-K latch which is shown in Figure 5-5. This latch consists of an S-R latch with two AND gates in front of the inputs. This is most useful because J-K latches act like S-R latches, and it is permissible to apply '1' to both inputs simultaneously. The state table and characteristic equation for a J-K latch are also shown in Figure 5-5.



(a)

St	Rt	Q_t	Q_{t+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
1	1	0	1
1	1	1	0

J, K \ Q	0	1
00	0	1
01	0	0
11	1	0
10	1	1

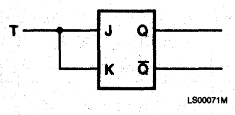
$Q_{t+1} = J*/Q + /K*Q$

(b)

Figure 5-5. J-K Latch (a) Logic Diagram (b) State Table and Characteristic Equation

5.2.4 T-type Latch

Another type of latch is a *triggered latch,* or T-type latch, which has only one input called T. Whenever T is high, the latch changes state. A T-type latch is realized by connecting both J and K to one input, T.



(a)

T	Q	Q_{t+1}
0	0	0
0	1	1
1	0	1
1	1	0

T \ Q	0	1
0	0	$\boxed{1}$
1	$\boxed{1}$	0

$$Q_{t+1} = /T{*}Q + T{*}/Q$$
$$= T :+: Q$$

(b)

Figure 5-6. T-type Latch (a) Logic Diagram (b) State Table and Characteristic Equation

5.3 Clocked Flip-Flops – Registers

In the previous section, different types of latches were discussed. Clocked flip-flops of the same types are available (S-R, J-K, D and T), and the state equations are the same, except that the output change in clocked flip-flops occurs on the clock edge.

In general, clocked flip-flops are more common than unclocked flip-flops, so the word "clocked" is often left out. Thus a D-type flip-flop is a clocked element, whereas a D-type latch is unclocked. Note, however, that this is common usage, not a definition.

Figure 5-7 illustrates the difference between a D-type latch and a D-type flip-flop.

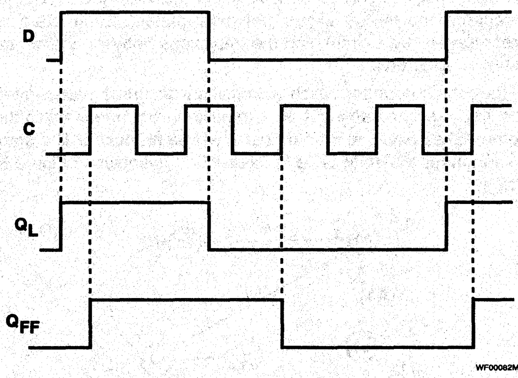


Figure 5-7. Comparison of a D-type latch and a positive edge-triggered D-type flip-flop

5.3.1 Characteristic Equations

The characteristic equations for various flip-flops are summarized as follows:

$Q_{t+1} = S + /R{*}Q_t$	S-R flip-flop
$Q_{t+1} = J{*}/Q_t + /K{*}Q_t$	J-K flip-flop
$Q_{t+1} = T + Q_t$	T-type flip-flop
$Q_{t+1} = D$	D-type flip-flop

In the above equations Q_{t+1} is the next state and Q_t is the present state. We can convert one flip-flop to another by inserting gates between the inputs and the actual flip-flop.

A group of clocked flip-flops forms a register. A register is a digital device that is used to hold several bits of information. A register may be a combination of flip-flops and gates. The gates control how and when the data from the flip-flops is transferred.

5.4 Designing Synchronous Sequential Circuits

In this section, we will familiarize ourselves with the analysis and synthesis of synchronous sequential circuits. Analysis will be covered first, since it will make the understanding of synthesis easier.

The present states of a sequential circuit depend not only on the present inputs, but also on the past states. A sequential circuit is constructed of flip-flops and gates. The gates form the combinatorial part of a sequential circuit; the flip-flops provide the memory of past states. A general block diagram of a sequential circuit is shown in Figure 5-8.

There are two basic types of sequential circuit: synchronous and asynchronous. Synchronous circuits are controlled by a common clock; these types of circuits are the easiest to design. Asynchronous circuits have independently clocked flip-flops whose clock rates may not have any relationships to each other. Such circuits may be faster than sychronous circuits, but are much harder to design.

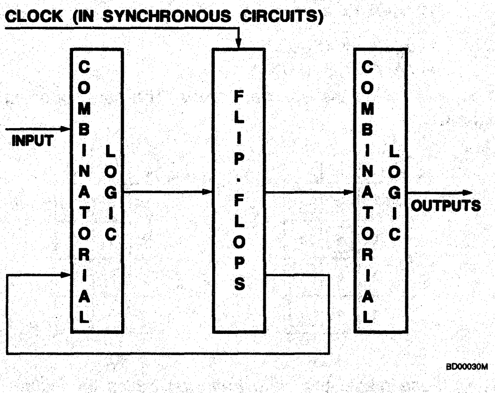


Figure 5-8. Sequential Circuit Block Diagram

The input combinatorial logic section receives external inputs and feeds information to the flip-flops. The flip-flops have a feedback path back to the input combinatorial circuit. At the rising edge of each clock pulse the information from the input combinatorial circuit is read into the flip-flops and the new outputs are generated through the output combinatorial circuit. The outputs do not change until the next clock edge.

Figure 5-9, shows an example of a sequential circuit.

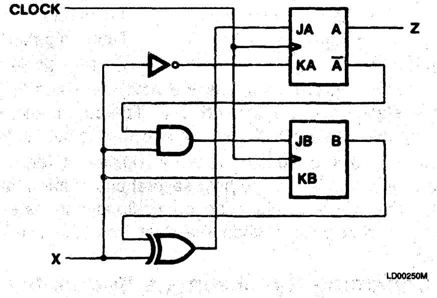


Figure 5-9. Example of a Sequential Circuit

This circuit consists of two J-K flip-flops, an inverter, an AND gate and an XOR gate. It has an external input, X, and an external output, Z.

5.4.1 State Transition Tables

The states of a sequential circuit are determined by its inputs, the outputs and states of the flip-flops. In order to examine these states, we should determine the input equations to the flip-flops. From Figure 5-9 we have:

$J_A = X :+: B \qquad J_B = X*/A \qquad Z = A$

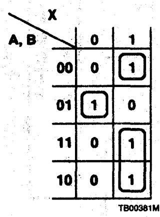
$K_A = /X \qquad K_B = X$

The characteristic equation for a J-K flip-flop gives:

$$\begin{aligned} A_{t+1} &= J_A*/Q + /K_A*Q \\ &= (X :+: B)*/A + X*A \\ &= (X*/B + /X*B)*/A + X*A \\ &= /A*/B*X + /A*B*/X + A*X \end{aligned}$$

$$\begin{aligned} B_{t+1} &= J_B*/Q + /K_B*Q \\ &= /A*/B*X + B*/X \end{aligned}$$

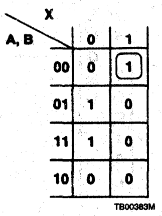
These are called the *state equations.* The corresponding K-maps are:



A, B \ X	0	1
00	0	1
01	1	0
11	0	1
10	0	1

TB00381M

A_{t+1}



A, B \ X	0	1
00	0	1
01	1	0
11	1	0
10	0	0

TB00383M

B_{t+1}

Using these maps, the *state transition tables* for Figure 5-9 can be derived. There are only four different combinations that A and B can have. The next state values are derived using these four possible combinations.

The present state is the state of the flip-flop before the clock pulse; the next state is the state of flip-flop after the clock pulse has been applied. The present output, Z, is the output of the sequential circuit after the clock pulse. As mentioned before, the circuit can have four states: AB = 00, 01, 11 and 10. At this point, we try to cover transitions for one input state. Suppose the circuit is in the 00 state. If an X = 0 input is applied, the next state will be 00. If an X = 1 is applied, the next state will be 11. The output for both cases is Z = 0.

PRESENT STATE	NEXT STATE A_{t+1} B_{t+1}		PRESENT OUTPUT
AB	X = 0	X = 1	Z
00	00	11	0
01	11	00	0
11	01	10	1
10	00	10	1

Figure 5-10. State Transition Table for Figure 5-9.

5.4.2 State Tables and State Diagrams

We can assign names to the four possible states: S_0 = 00, S_1 = 01, S_2 = 11 and S_3 = 10. Using these assignments the state transition table can be modified to the *state table* shown in Figure 5-11.

PRESENT STATE	NEXT STATE		PRESENT OUTPUT Z
	X = 0	X = 1	
S0	S0	S2	0
S1	S2	S0	0
S2	S1	S3	1
S3	S0	S3	1

Figure 5-11. State Table for Figure 5-9

A state diagram can be derived using the state table. A state diagram shows transitions between states. Each state is represented by a circle, and the transitions between states are shown by arrows.

The condition under which a transition occurs is represented by X/Z. Applying an input, X, a transition from one state to the other takes place, and an output Z will be produced. The state diagram for the state table in Figure 5-11 is shown in Figure 5-12.

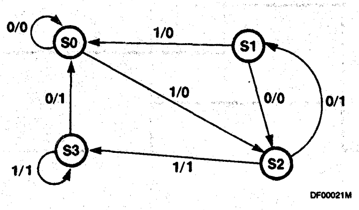


Figure 5-12. State Diagram for Figure 5-9

We have gone through a complete analysis procedure for the previous example. This process can be summarized as follows:

1. Using a given network, determine the input equations.
2. Derive the next state equations, using the flip-flop characteristic equations:

   $Q_{t+1} = D$ D-type flip-flop
   $Q_{t+1} = T :+: Q$ T-type flip-flop
   $Q_{t+1} = J*/Q + /K*Q$ J-K flip-flop
   $Q_{t+1} = S + /R*Q$ S-R flip-flop
3. Derive the corresponding K-maps and transition tables.
4. Assign state names and make the state table.
5. From the state table, draw the state diagram.

The synthesis of a sequential circuit uses the same steps, but in reverse order. A flow chart of the procedure is shown in Figure 5-13.

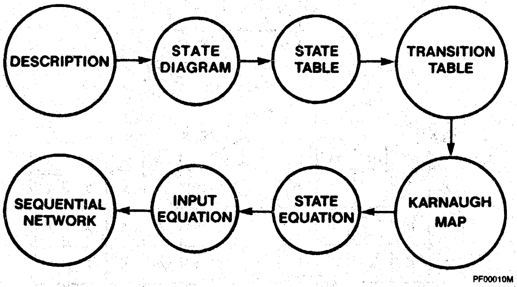


Figure 5-13. Flow Chart of Sequential Circuit Synthesis

5.4.3 Design Examples

Let's look at the procedure for designing sequential circuits in more detail by analyzing some design problems. The design steps will follow the flow diagram in Figure 5-13.

Example 5.1

We will design a clocked sequential circuit which receives an input, X, and will produce an output Z = 1 after it has received an input sequence of 0010 or 100.

Solution:

The first step is to make a state diagram. The state diagram will start in a state designated by S0. If X = 0 is input while in state S0, it could be the start for the 0010 sequence, so we move to state S1. If a 1 is receive, the circuit will go to state S4, which could be the start for the 100 sequence.

When in the S1 state, one of two inputs could be received: X = 0 or X = 1. If X = 0, then the circuit could still follow the 0010 sequence, because so far it has received the 00 sequence; this takes us to state S2. If X = 1, then the input sequence would be 01 so far, which cannot follow the 0010 any more, but could be a start for the 100 sequence. Therefore under this condition the circuit would have a transition from state S1 to S4.

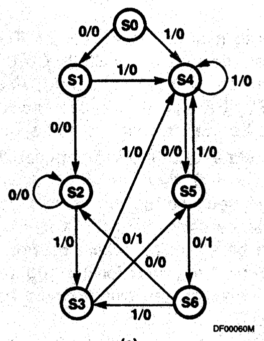
While in the S4 state, if an X = 1 is received, it would stay in the same state, because this of 11 could start a new 100 pattern. If X = 0 is received, the 100 pattern could continue, because 10 follows the 100 pattern.

When in the S2 state, if an X = 0 is received, it will stay in the S2 state; if X = 1, then it will have a transition from the S2 to the S3 state. While in the S3 state, we will have a transition to state S4 if X = 1, because the input sequence would be 0011, which cannot follow the 0010 sequence any more, but the 1 at the end could be a beginning for the 100 sequence. If X = 0 is received then the 0010 sequence is complete and an output Z = 1 is generated. At the same time in the 0010 sequence, the 10 at the end could be the start of the 100 sequence. Therefore we move from the S3 to the S5 state.

While in the S5 state an X = 1 will transfer the circuit from S5 to S4, and an X = 0 will cause a transition from S5 to S6. Under this condition a Z = 1 is generated for the output, because the 100 sequence for the inputs has occurred.

From the S6 state we will have a transition from S6 to S2 if X = 0, and a transition to S3 if X = 1. A state diagram of this design is shown in Figure 5-14a.

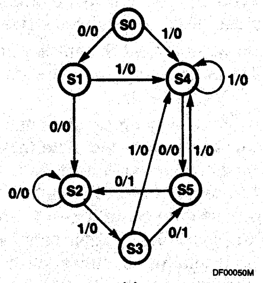
By analyzing the state diagram, a state table is generated (Figure 5-14b). A careful look at the state table will show that the S2 and the S6 states are equivalent, because under X = 0 they both have a transition to state S2 with Z = 0, and under X = 1 they would transfer to S3 with Z = 0. So the state table can be reduced to Figure 5-14c. After summarizing the state table, there will be a total of six states left, so at least three variables will be needed for the state assignments ($2^3 = 8$). In this case only six assignments will be used. The state variables are designated by A, B and C. The state assignments are illustrated in Figure 5-14d.



(a)

PRESENT STATE	NEXT STATE		PRESENT OUTPUT	
	X = 0	X = 1	X = 0	X = 1
S0	S1	S4	0	0
S1	S2	S4	0	0
S2	S2	S3	0	0
S3	S5	S4	1	0
S4	S5	S4	0	0
S5	S6	S4	1	0
S6	S2	S3	0	0

(b)



(c)

PRESENT STATE	NEXT STATE		PRESENT OUTPUT	
	X = 0	X = 1	X = 0	X = 1
S0	S1	S4	0	0
S1	S2	S4	0	0
S2	S2	S3	0	0
S3	S5	S4	1	0
S4	S5	S4	0	0
S5	S2	S4	1	0

(d)

Figure 5-14. Example 5.1
(a) State Diagram (b) State Transition Table (c) Reduced State Diagram (d) Reduced State Transition Table

From the assignment table, we derive the K-maps for each state. In this design the circuit is realized by J-K flip-flops, therefore two K-maps are needed for each state variable (shown in Figure 5-15). These can be derived from a K-map of the transition, and the J and K characteristics in figure 5-15(a).

$Q_t \rightarrow Q_{t+1}$	J	K
0 → 0	0	X
0 → 1	1	X
1 → 0	X	1
1 → 1	X	0

(a)

A, B \ C, X	00	01	11	10
00	0	1	1	0
01	0	0	1	1
11	X	X	X	X
10	1	1	1	0

TB00410M

A_{t+1}

A, B \ C, X	00	01	11	10
00	0	1	1	0
01	0	0	1	1
11	X	X	X	X
10	X	X	X	X

TB00440M

J_A = /B*X + B*C

A, B \ C, X	00	01	11	10
00	X	X	X	X
01	X	X	X	X
11	X	X	X	X
10	0	0	0	1

TB00450M

K_A = C*/X

(b)

A, B \ C, X	00	01	11	10
00	0	0	0	1
01	1	1	0	0
11	X	X	X	X
10	0	0	0	1

TB00420M

B_{t+1}

A, B \ C, X	00	01	11	10
00	0	0	0	1
01	X	X	X	X
11	X	X	X	X
10	0	0	0	1

TB00320M

J_B = C*/X

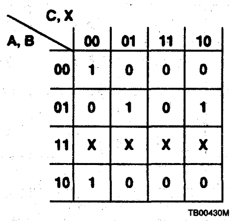
A, B \ C, X	00	01	11	10
00	X	X	X	X
01	0	0	1	1
11	X	X	X	X
10	X	X	X	X

TB00330M

K_B = C

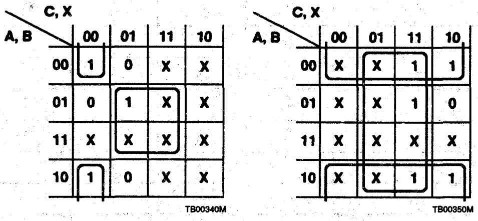
(c)

Figure 5-15. Karnaugh Maps for Example 5.1



A, B \ C, X	00	01	11	10
00	1	0	0	0
01	0	1	0	1
11	X	X	X	X
10	1	0	0	0

C_{t+1}



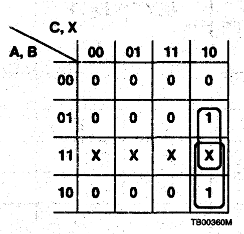
A, B \ C, X	00	01	11	10
00	1	0	X	X
01	0	1	X	X
11	X	X	X	X
10	1	0	X	X

A, B \ C, X	00	01	11	10
00	X	X	1	1
01	X	X	1	0
11	X	X	X	X
10	X	X	1	1

J_C = B*X + /B*/C*/X

K_C = /B + X

(d)



A, B \ C, X	00	01	11	10
00	0	0	0	0
01	0	0	0	1
11	X	X	X	X
10	0	0	0	1

Z = A*C*/X + B*C*/X

(e)

Figure 5-15 (Continued)

The state equations are derived from the K-maps. Using the state equations, the logic diagram in Figure 5-16 is obtained. The state equations are summarized as follows.

J_A = /B*X + B*C J_B = C*/X J_C = B*X + /B*/C*/X

K_A = /C*/X K_B = C K_C = /B + X

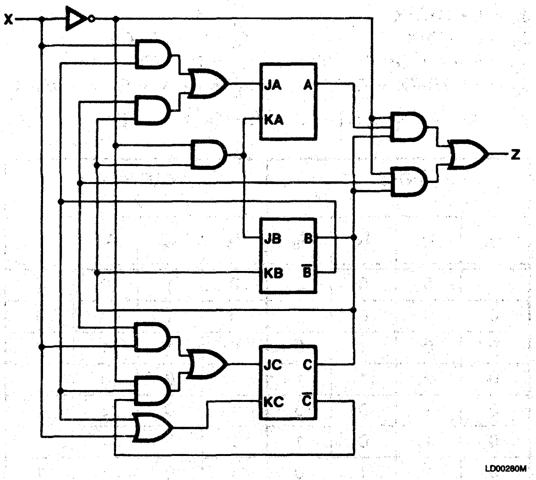


Figure 5-16. Circuit Diagram for Example 5.1

Example 5.2

We will derive the state diagram, state transition table, and state equations of a sequential circuit which adds five to a binary number in the range of 0000 to 1010 (decimal 0 to 10). The inputs and outputs will be serial with the least significant bit arriving first. This design will be realized with J-K flip-flops.

Solution:

In Figure 5-17a, all the possible combinations for the input and the output are shown. The sequence will start from state A. At time t_0 (when the first input is received), if X = 0, then we look at the map for all possible combinations and notice that at t_0 whenever X = 0, the output is a 1. Thus, if the present state is A, under X = 0 the output is Z = 1. On the other hand, if X = 1 the map shows that Z will be a 0. At time t_1, if X = 0 and the sequence of the inputs has been 00 then the output would be a 01. So in this transition Z = 0. Likewise, the remaining states of the state diagram can be derived by inspection of the table in Figure 5-17a. The state diagram is then completed as in Figure 5-17b. Note that the states have been labeled arbitrarily and that we have not yet determined how the states will be implemented.

The state table is determined from the state diagram. Inspecting the state table shows that some of the states are equivalent. States K, J, I, and H have the same next state under X = 0 and X = 1, and produce the same outputs; therefore K = J = I = H, and they can be replaced by the H state. This allows us to reduce the state table to the table shown in Figure 5-18b. There are now a total of nine states used, so four state variables will be needed. The state variables are called A, B, C and D. A complete state transition table is shown in Figure 5-18c.

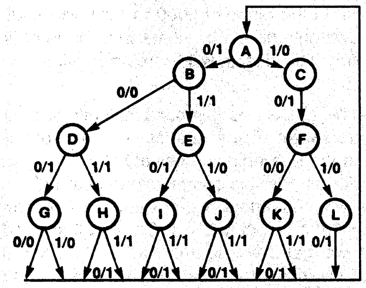
The K-maps are drawn from the state table. The state equations are derived using the K-maps.

J_A = B*/C*D*X K_A = 1
J_B = C + D*X K_B = C + D*X
J_C = B*/X + D*/X + /D*X K_C = B + /D
J_D = /A*/B*/X + B*/C K_D = B*/X + C*/X + /C*X

Z = /A*/B*C + /B*/D*/ X + /C*/D*/X + C*D

INPUT X				OUTPUT Z			
t_3	t_2	t_1	t_0	t_3	t_2	t_1	t_0
0	0	0	0	0	1	0	1
0	0	0	1	0	1	1	0
0	0	1	0	0	1	1	1
0	0	1	1	1	0	0	0
0	1	0	0	1	0	0	1
0	1	0	1	1	0	1	0
0	1	1	0	1	0	1	1
0	1	1	1	1	1	0	0
1	0	0	0	1	1	0	1
1	0	0	1	1	1	1	0
1	0	1	0	1	1	1	1

(a)



DF00041M

(b)

Figure 5-17. Example 5.2 (a) Truth Table (b) State Diagram

PRESENT STATE	NEXT STATE		Z	
	X = 0	X = 1	X = 0	X = 1
A	B	C	1	0
B	D	E	0	1
C	F	—	1	—
D	G	H	1	1
E	I	J	1	0
F	K	L	0	0
G	A	A	0	0
H	A	A	1	1
I	A	A	1	1
J	A	A	1	1
K	A	A	1	1
L	A	—	1	—

(a)

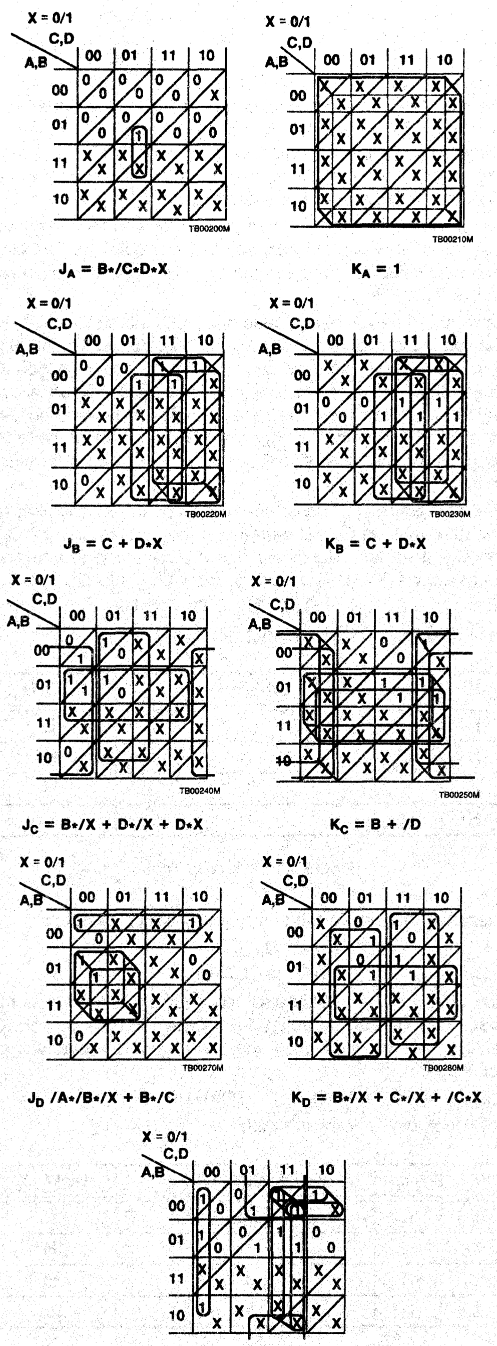
PRESENT STATE	NEXT STATE		Z	
	X = 0	X = 1	X = 0	X = 1
A	B	C	1	0
B	D	E	0	1
C	F	—	1	—
D	G	H	1	1
E	H	H	1	0
F	H	L	0	0
G	A	A	0	0
H	A	A	1	1
L	A	—	1	—

(b)

PRESENT STATE ABCD	NEXT STATE		Z	
	X = 0	X = 1	X = 0	X = 1
0000	0001	0010	1	0
0001	0011	0100	0	1
0010	0101	—	1	—
0011	0110	0111	1	1
0100	0111	0111	1	0
0101	0111	1000	0	0
0110	0000	0000	0	0
0111	0000	0000	1	1
1000	0000	—	1	—

(c)

Figure 5-18. Example 5.2 (a) State Transition Table (b) Reduced State Table (c) Reduced Transition Table



J_A = B*/C*D*X

K_A = 1

J_B = C + D*X

K_B = C + D*X

J_C = B*/X + D*/X + D*X

K_C = B + /D

J_D /A*/B*/X + B*/C

K_D = B*/X + C*/X + /C*X

Z = /A*/B*C + /B*/D*/X + /C*/D*/X + C*D

Figure 5-19. Karnaugh Maps for Example 5.2

5.5 Counters

Counters are among the most commonly used sequential circuits. In the following sections, they are covered in detail.

A register that goes through a predetermined state sequence upon receiving an input pulse is called a counter. Counters are one of the simplest sequential circuits, and are found in almost all equipment containing digital logic. The sequence of states followed determines the different types of counters (BCD, binary, etc).

The binary counter is one of the simplest counters. An n-bit binary counter is a register with n flip-flops and associated combinatorial logic that follows the sequence from 0 to 2^{n-1}.

Example 5.3

We will look at the design of a 3-bit up/down binary counter. There are three outputs from the binary counter: DA, DB and DC. The input to the counter is X; the counter will increment if X = 1, and decrement if X = 0. We will use D flip-flops for this design.

Solution:

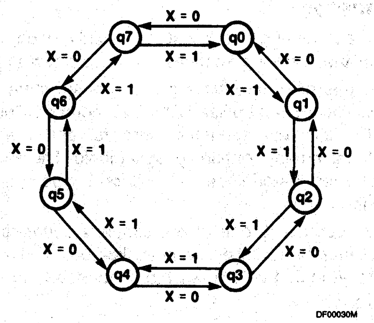
This circuit will have eight different states, because the 3-bit counter cycles through 2^3 states. The states are called q0, q1, q2, q3, q4, q5, q6 and q7. If X = 1 each state will have a transition to its next higher state. For example, q0 will go to q1, q1 will go to q2, and so on. If X = 0 then each state will change to its previous state. For example, q7 will go to q6, q5 to q4, and so on.

The state diagram of this design is shown in Figure 5-22a. State transition tables, derived from the state diagram, are shown in Figure 5-22b. The K-maps and the state variables are shown in Figure 5-22c. The state equations are derived from the K-maps. The equations are summarized as follows:

D_A: = /A*/B*/C*/X + /A*B*C*X + A*B*/C + A*/B*X + A*C*/X

D_B: = /B*/C*/X + /B*C*X + B*/C*X + B*C*/X

D_C: = /C



(a)

PRESENT STATE	NEXT STATE	
	X = 0	X = 1
q0	q7	q1
q1	q0	q2
q2	q1	q3
q3	q2	q4
q4	q3	q5
q5	q4	q6
q6	q5	q7
q7	q6	q0

PRESENT STATE	NEXT STATE	
	X = 0	X = 1
000	111	001
001	000	010
010	001	011
011	010	100
100	011	101
101	100	110
110	101	111
111	110	000

(b)

A, B \ C, X	00	01	11	10
00	1	0	0	0
01	0	0	1	0
11	1	1	0	1
10	0	1	1	1

TB00290M

D_A: = /A*/B*/C*/X + /A*B*C*X + A*B*/C + A*/B*X + A*C*/X

A, B \ C, X	00	01	11	10
00	1	0	1	0
01	0	1	0	1
11	0	1	0	1
10	1	0	1	0

TB00300M

D_B: = /B*/C*/X + /B*C*X + B*C*/X

A, B \ C, X	00	01	11	10
00	1	1	0	0
01	1	1	0	0
11	1	1	0	0
10	1	1	0	0

TB00310M

D_C: = /C

(c)

Figure 5-20. 3-Bit Up/Down Counter (a) State Diagram (b) State Transition Tables (c) Karnaugh Maps

We can easily implement this counter using a PAL device. The PAL device for this design will require at least three flip-flops. Since there is no PAL device available that has only three flip-flops, the best PAL device will be the PAL16R4, because it has four flip-flops and contains the AND-OR gates needed to realize the combinatorial circuit of the counter. The schematic of the 3-bit up/down counter using the PAL16R4 is shown on page 8-31. Note that there are plenty of unused inputs and outputs for other functions if desired.

Design of a binary counter can be complicated by adding new features to it. Counters can be made to count up or down, load new values in, or clear the present state of the counter and reset to 0's.

As we try to design bigger counters, using the state tables and K-maps becomes more difficult. In the design of the 3-bit counter, K-maps were used. If we try to design a counter bigger than this, summarizing the equations will be a very tedious process. Therefore, we try to find a general solution for solving the counter design problem. Let's try to write the equation for the most significant bit (MSB) of an n-bit binary counter (Q_n).

First we look at the case where the counter is counting up. The new value of Q_n will depend on the carry-in from bit Q_{n-1} into Q_n. If all less significant bits (LSBs) are high when we count up, we will have a carry-in from Q_{n-1} into Q_n.

$$C_{IN} := Q_{n-1}*Q_{n-2}*...Q1*Q0*UP$$

Now let's look at the following table:

UP	Q_n	CARRY-IN	NEW Q_n
H	L	L	L
H	L	H	H
H	H	L	H
H	H	H	L

Figure 5-21. Carry-In Table

Examining the above table, it is easily concluded that:

Q_n : = Q_n :+: carry-into Q_n

Q_n : = Q_n :+: $(Q_{n-1}*Q_{n-2}*...Q0*UP)$

Now that we have calculated the equation for the count-up case, let's look at the count-down case. Borrow-out from Q_n to Q_{n-1} will be high if all the LSBs are low and we are counting down.

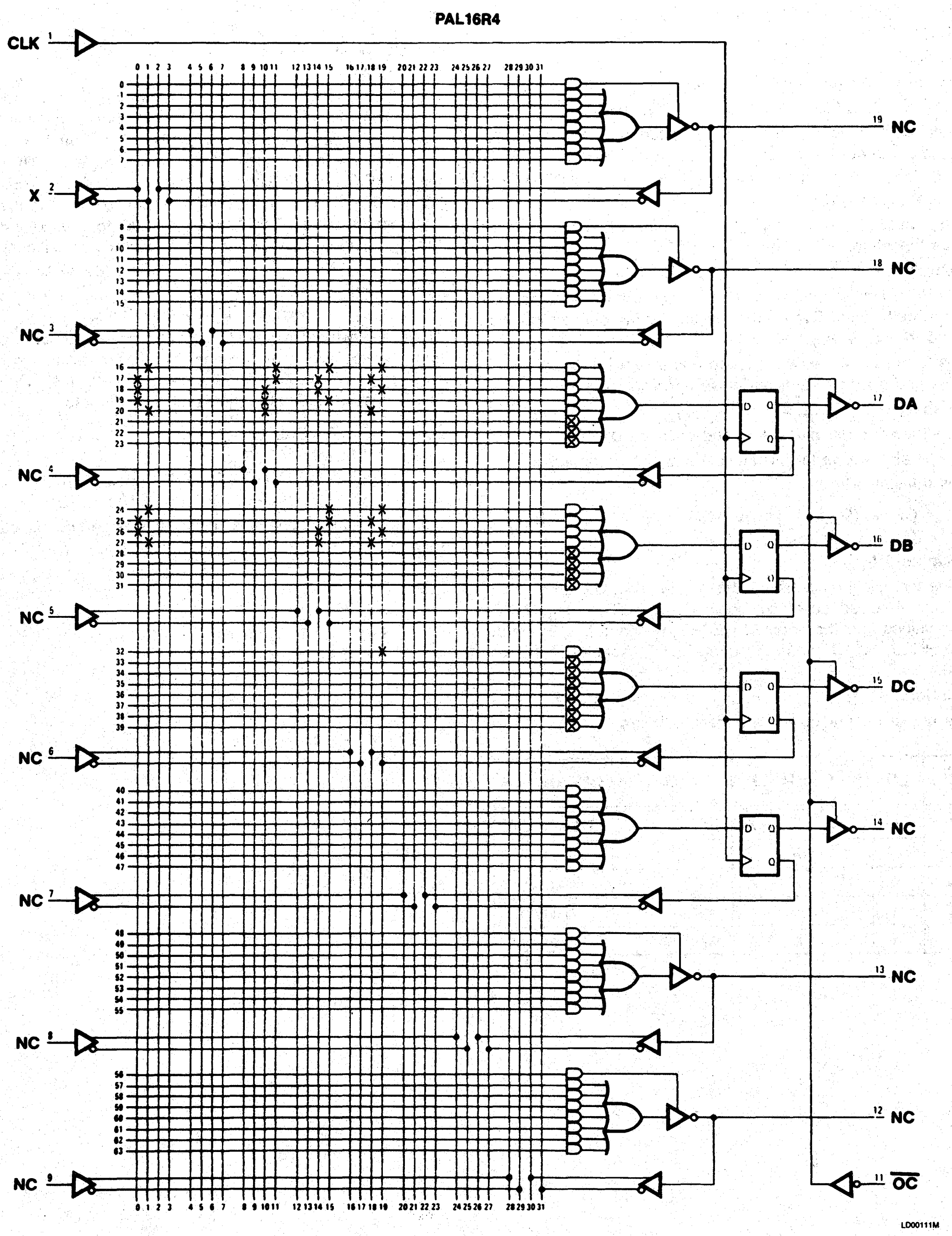
B_{OUT} := $/Q_n - 1*/Q_n - 2*.../Q0*/UP$

Let's look at the following table:

UP	Q_n	BORROW OUT	NEW Q_n
L	L	L	L
L	L	H	H
L	H	L	H
L	H	H	L

Figure 5-22. Borrow-out Table

Logic Diagram



So:

$Q_n := Q_n :+: \text{Borrow-out } Q_n$

$Q_n := Q_n :+: (/Q_{n-1}*/\ Q_{n-2}*.../Q_1*/Q_0*/UP)$

Therefore:

$Q_n := Q_n :+: (Q_{n-1}\ *Q_{n-2}*...Q_1*Q_0*Q_0*UP)$
$\quad + Q_n :+: (/Q_{n-1}*/\ Q_{n-2}*.../Q1*/Q0/*UP)$

Another way to look at a counter is to formulate the following statement about the nth bit of the counter:

> On should hold UNLESS all lower order bits are HI and we are counting UP.

The "UNLESS" operation is implemented by an XOR gate. Thus this statement translates directly into

$Qn := Qn :+: (Q_{n-1}*Q_{n-2}*...Q_1*Q_0*UP)$

This is, of course, the same equation we obtained by looking at the truth table. The expression for a down-counter can be found the same way.

This formulation makes it easy to combine these two expressions into a single equation for an up/down counter:

Qn should hold UNLESS:

> All lower order bits are HI and counting UP
>
> OR all lower order bits are LO and NOT counting UP.

This translates to

$Qn := Qn :+: (Q_{n-1}*Q_{n-2}*...*Q_1*Q_0*UP)$ Eq. 5.1
$\quad + (Q_{n-1}*/Q_{n-2}*...*/Q_{1*1}Q_0*/UP)$

Example 5.4

Using the solution discussed above, let's try to design an n-bit counter that can count up, count down, RESET and LOAD new values into the counter. RESET overrides LOAD, count and HOLD. LOAD overrides count. RESET sets all of the Qs to 0.

Solution:

The above operations are summarized below.

RE-SET	LD	HOLD	UP	D	Q	OPERATION
H	X	X	X	X	X	Set all LOW
L	L	X	X	D	D	Load D
L	H	H	X	X	Q	Hold Q
L	H	L	L	X	Q plus 1	Increment
L	H	L	H	X	Q minus 1	Decrement

We will try to generate a general equation that can be used for any bit in the counter. If we take the nth bit, we have D_n as input and Q_n as output. There are five operations that can happen for any given bit: LOAD, HOLD, RESET, count up or count down.

If RESET is HI, then all other operations are overridden. Overriding the other operations is sufficient to provide a LO into the flip-flop. So every product term will contain /RESET. We LOAD the new value in the counter's register if RESET is OFF (/RESET = H). Since Q_n is replaced by D_n if (/RESET = H, LOAD = H), the expression that will allow loading the new value will be $/RESET*LOAD*/D_n$. In order to be able to HOLD the Q_n value, we should have the following conditions: (RESET = L, LOAD = L, HOLD = H). So the expression for holding the old value is:

$$/RESET*/LOAD*/HOLD*Q_n$$

There are two more functions for the counter: count up and count down. These two functions have been calculated in Eq. 5.1. Using this equation and the calculation for the HOLD and LOAD cases, the final equation for the nth bit is:

$Q_n = \ /RESET*LOAD*D_n$ Eq. 5.2
$\quad + \ /RESET*/LOAD*/Q_n$
$\quad :+: \ /RESET*/LOAD*HOLD*UP*Q0*Q1*...\ Q_{n-1}$
$\quad + \ /RESET*/LOAD*HOLD*/UP*/Q0*/Q1*.../Q_{n-1}$

Using the above general equation, any large counter can be designed.

Table of Contents

PLE to PROM Cross Reference

TEMP. RANGE	PLE NUMBER	INPUTS	OUTPUTS	OUTPUT TYPE	MEMORY SIZE	PROM NUMBER	PACKAGE
Com.	PLE5P8C	5	8	Three-State	32 x 8	63S081	16N,J,(20),(NL)
	PLE5P8AC	5	8	Three-State	32 x 8	63S081A	16N,J,(20),(NL)
	PLE5P16C	5	16	Three-State	32 x 16	none	24NS,JS,(28),(NL)
	PLE6P16C	6	16	Two-State	64 x 16	none	24NS,JS,(28),(NL)
	PLE8P4C	8	4	Three-State	256 x 4	63S141A	16N,J,(20),(NL)
	PLE8P8C	8	8	Three-State	256 x 8	63S281A	20N,J,NL
	PLE9P4C	9	4	Three-State	512 x 4	63S241A	16N,J,(20),(NL)
	PLE9P8C	9	8	Three-State	512 x 8	63S481A	20N,J,NL
	PLE10P4C	10	4	Three-State	1024 x 4	63S441A	18N,J,(20),(NL)
	PLE10P8C	10	8	Three-State	1024 x 8	63S881A	24N,J,NS,JS,(28),(NL)
	PLE11P4C	11	4	Three-State	2048 x 4	63S841A	18N,J,(20),(NL)
	PLE11P8C	11	8	Three-State	2048 x 8	63S1681A	24N,J,NS,JS,(28),(NL)
	PLE12P4C	12	4	Three-State	4096 x 4	63S1641A	20N,J,NL
	PLE12P8C	12	8	Three-State	4096 x 8	63S3281A	24N,J,(28),(NL)
	PLE9R8C	9	8	Register	512 x 8	63RA481A	24NS,JS,(28),(NL)
	PLE10R8C	10	8	Register	1024 x 8	63RS881A	24NS,JS,(28),(NL)
	PLE11RA8C	11	8	Register	2048 x 8	63RA1681A	24NS,JS,(28),(NL)
	PLE11RS8C	11	8	Register	2048 x 8	63RS1681A	24NS,JS,(28),(NL)
Mil.	PLE5P8M	5	8	Three-State	32 x 8	53S081	16J,W,(20),(L)
	PLE8P4M	8	4	Three-State	256 x 4	53S141A	16J,W,(20),(L)
	PLE8P8M	8	8	Three-State	256 x 8	53S281A	20J,W,L
	PLE9P4M	9	4	Three-State	512 x 4	53S241A	16J,W,(20),(L)
	PLE9P8M	9	8	Three-State	512 x 8	53S481A	20J,F,L
	PLE10P4M	10	4	Three-State	1024 x 4	53S441A	18J,W,(20),(L)
	PLE10P8M	10	8	Three-State	1024 x 8	53S881A	24JS,J,W,(28),(L)
	PLE11P4M	11	4	Three-State	2048 x 4	53S841A	18J,W,(28),(L),(20),(L)*
	PLE11P8M	11	8	Three-State	2048 x 8	53S1681A	24JS,J,W,(28),(L)
	PLE12P4M	12	4	Three-State	4096 x 4	53S1641A	20J
	PLE12P8M	12	8	Three-State	4096 x 8	53S3281A	24J,W,(28),(L)
	PLE9R8M	9	8	Register	512 x 8	53RA481A	24JS,W,(28),(L)
	PLE10R8M	10	8	Register	1024 x 8	53RS881A	24JS,J,W,(28),(L)
	PLE11RA8M	11	8	Register	2048 x 8	53RA1681A	24JS,W,(28),(L)
	PLE11RS8M	11	8	Register	2048 x 8	53RS1681A	24JS,W,(28),(L)

*The PLE11P4M is available in a 20- or 28-pin Leadless Chip Carrier

Programmable Logic Element PLE™ Circuit Facility

Features/Benefits

- Programmable replacement for conventional TTL logic
- Reduces IC inventories and simplifies their control
- Expedites and simplifies prototyping and board layout
- Saves space with 0.3-inch SKINNYDIP® packages (except PLE12P8)
- Programmed on standard PROM programmers
- Test and simulation made simple with PLEASM™ software
- Low-current PNP inputs
- Three-state outputs
- Reliable TiW fuses guarantee > 98% programming yeld

Ordering Information

PLE 5 P 8 A C N STD

PREFIX
PLE = Programmable Logic Element

NUMBER OF INPUTS

OUTPUT TYPE
P = Non Registered
R = Registered
RA = Registered Asynchronous Enable
RS = Registered Synchronous Enable

NUMBER OF OUTPUTS

PERFORMANCE
Blank = Standard
A = Enhanced

PROCESSING
STD = Standard
XXXX = Other

PACKAGE TYPE
N = Plastic DIP
NS = Plastic SKINNYDIP
J = Ceramic DIP
JS = Ceramic SKINNYDIP
F = Ceramic Solder Seal Flat Pack
L = Leadless Chip Carrier
NL = Plastic Leaded Chip Carrier
W = Cerpack

TEMPERATURE CODE
C = 0°C to +75°C
M = −55°C to +125°C

PLE Circuit Selection Guide

PART NUMBER	INPUTS	OUTPUTS	PRODUCT TERMS	OUTPUT REGISTERS	t_{PD} (ns) MAX*
PLE5P8	5	8	32		25
PLE5P8A	5	8	32		15
PLE5P16	5	16	32		18
PLE6P16	6	16	64		20
PLE8P4	8	4	256		30
PLE8P8	8	8	256		28
PLE9P4	9	4	512		35
PLE9P8	9	8	512		30
PLE10P4	10	4	1024		35
PLE10P8	10	8	1024		30
PLE11P4	11	4	2048		35
PLE11P8	11	8	2048		35
PLE12P4	12	4	4096		35
PLE12P8	12	8	4096		35
PLE9R8	9	8	512	8	15
PLE10R8	10	8	1024	8	15
PLE11RA8	11	8	2048	8	15
PLE11RS8	11	8	2048	8	15

*Clock to output time for registered outputs.
Note: Commercial limits specified.

PLE Means Programmable Logic Element

Joining the world of IdeaLogic™ is a new generation of High-speed PROMs which the designer can use as *Programmable Logic Elements.* The combination of PLE circuits as logic elements with PAL devices can greatly enhance system speed while providing almost unlimited design freedom.

Basically, PLE circuits are ideal when a large number of product terms is required. On the other hand, a PAL device is best suited for situations when many inputs are needed.

The PLE circuit transfer function is the familiar sum of products. Like the PAL device, the PLE circuit has a single array of fusible links. Unlike the PAL device, the PLE circuits have a programmable OR array driven by a fixed AND array (the PAL device is a programmed AND array driving a fixed OR array).

PRODUCT TERM AND INPUT LINES

	PLE	PAL
Product Terms	32 to 4096	1 to 16
Input Lines	5 to 12	6 to 64

The PLE family features common electrical parameters and programming algorithms, low-current PNP inputs, full Schottky clamping and three-state outputs.

The entire PLE family is programmed on conventional PROM programmers with the appropriate personality cards and socket adapters.

Registered PLE Circuits

The registered PLE circuits have on-chip "D" type registers, versatile output enable control through synchronous and asynchronous enable inputs, and flexible start-up sequencing through programmable initialization.

Data is transferred into the output registers on the rising edge of the clock. Provided that the asynchronous ($\overline{\text{E}}$) and synchronous ($\overline{\text{ES}}$) enables are Low, the data will appear at the outputs. Prior to the positive clock edge, register data are not affected by changes in addressing or synchronous enable inputs.

Data control is made flexible with synchronous and asynchronous enable inputs. Outputs may be set to the high-impedance state at any time by setting $\overline{\text{E}}$ to a High or if $\overline{\text{ES}}$ is High when the rising clock edge occurs. When V_{CC} power is first applied the synchronous enable flip-flop will be in the set condition causing the outputs to be in the high-impedance state.

A flexible initialization feature allows start-up and time-out sequencing with 1:16 programmable words to be loaded into the output registers. With the synchronous INITIALIZE ($\overline{\text{IS}}$) pin Low, one of the 16 initialize words, addressed through pins 5, 6, 7 and 8 will be set in the output registers independent of all other input pins. The unprogrammed state of $\overline{\text{IS}}$ words are Low, presenting a CLEAR with $\overline{\text{IS}}$ pin Low. With all $\overline{\text{IS}}$ column words (A3-AO) programmed to the same pattern, the $\overline{\text{IS}}$ function will be independent of both row and column addressing and may be used as a single pin control. With all $\overline{\text{IS}}$ words programmed High a PRESET function is performed.

The PLE9R8 has asynchronous PRESET and CLEAR functions. With the chip enabled, a Low on the $\overline{\text{PR}}$ input will cause all outputs to be set to the High state. When the $\overline{\text{CLR}}$ input is set Low the output registers are reset and all outputs will be set to the Low state. The $\overline{\text{PR}}$ and $\overline{\text{CLR}}$ functions are common to all output registers and independent of all other data input states.

	AND	OR	OUTPUT OPTIONS
PLE	Fixed	Prog	TS, Registered Outputs, Fusible Polarity
PLA	Prog	Prog	TS, OC, Fusible Polarity
PAL	Prog	Fixed	TS, Registered Feedback I/O Fusible Polarity

PLEASM™ Software

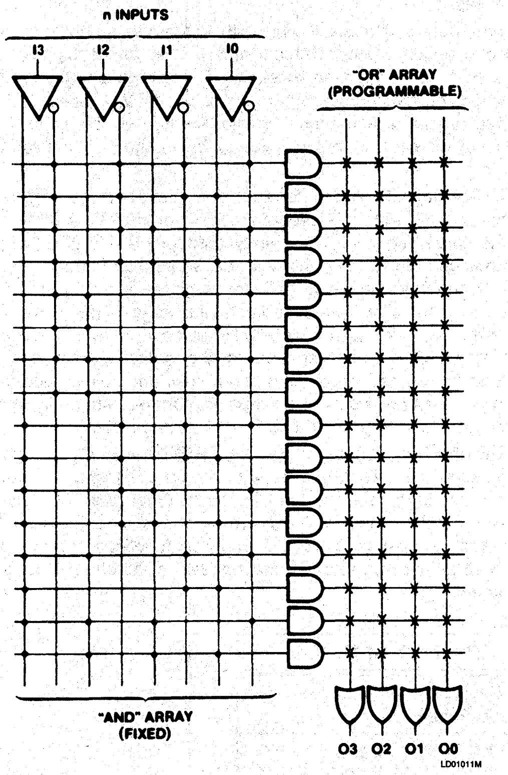
Software that makes programmable logic easy.

Monolithic Memories has developed a software tool to assist in designing and programming PROMs as PLE circuits. This package called "PLEASM" (PLE Assembler) is available for several computers including the VAX/VMS and IBM PC/DOS.

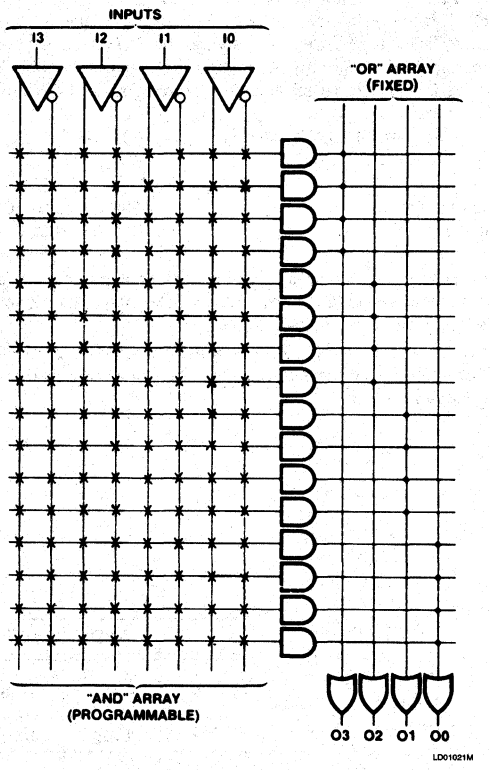
PLEASM software converts design equations (Boolean and arithmetic) into truth tables and formats compatible with PROM programmers. A simulator is also provided to test a design using a Function Table before actually programming the PLE circuit.

PLEASM software may be requested through the Monolithic Memories IdeaLogic Exchange.

PLE (PROM)



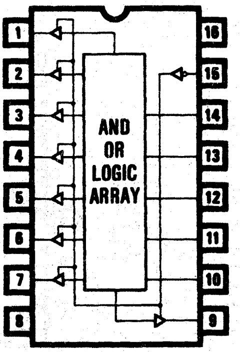
PAL Device



NOTE: • = hardwired connection
X = programmable connection

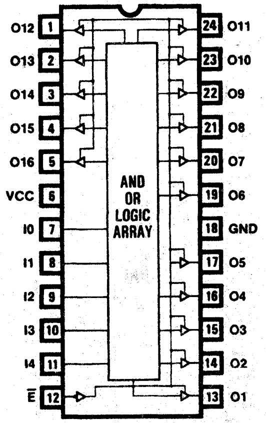
Logic Symbols

PLE5P8/A



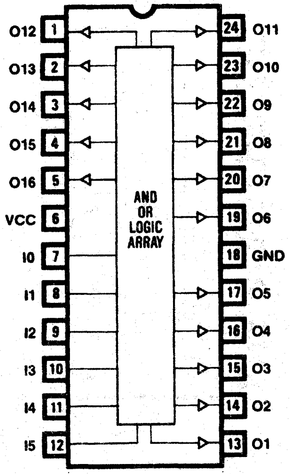
CD01500M

PLE5P16



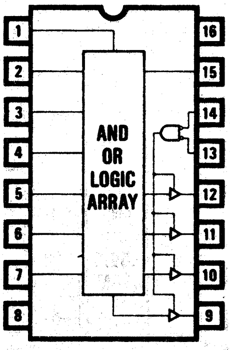
CD01070M

PLE6P16



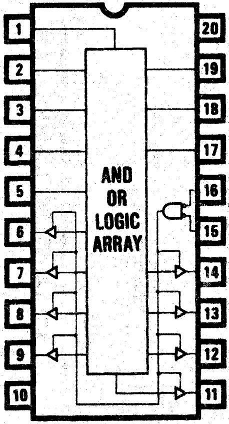
CD01090M

PLE8P4



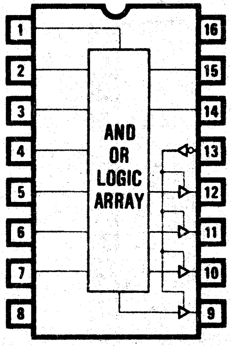
CD01510M

PLE8P8



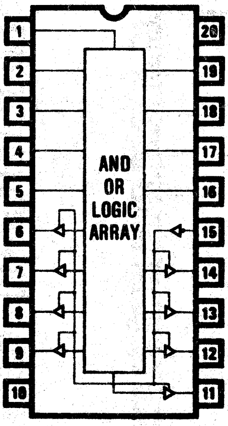
CD01520M

PLE9P4



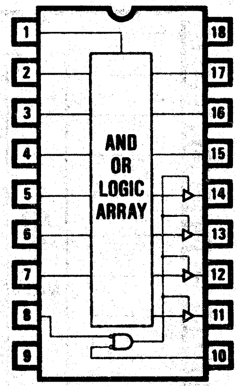
CD01530M

PLE9P8



CD01540M

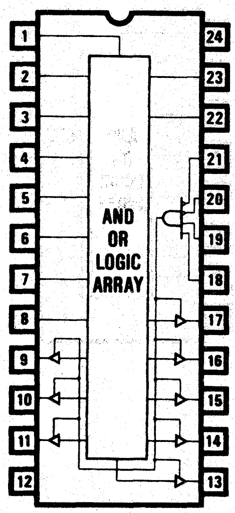
PLE10P4



CD01550M

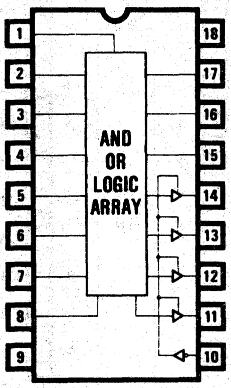
Logic Symbols

PLE10P8



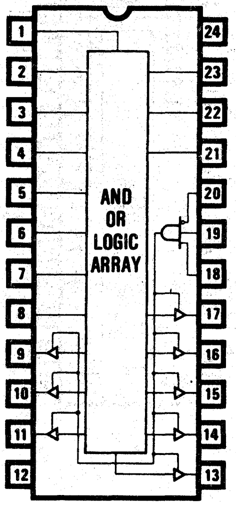
CD01560M

PLE11P4



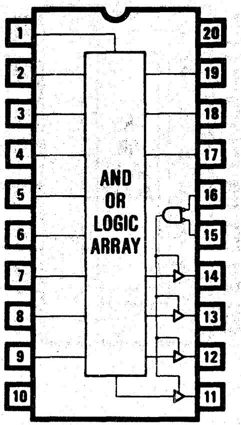
CD01570M

PLE11P8



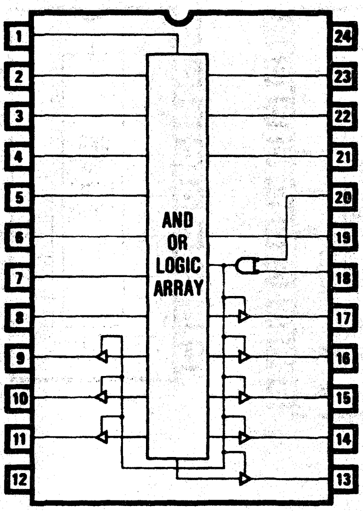
CD01580M

PLE12P4



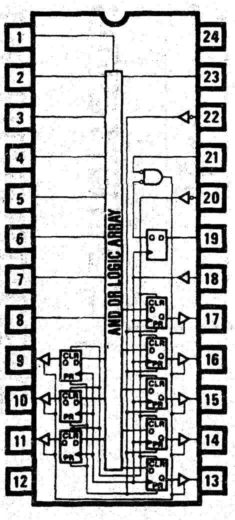
CD01590M

PLE12P8



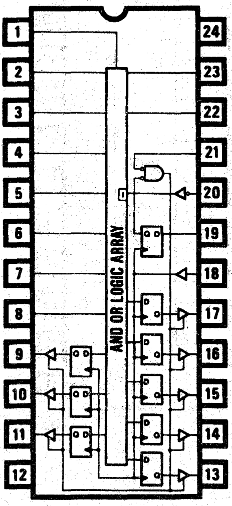
CD01600M

PLE9R8



CD01610M

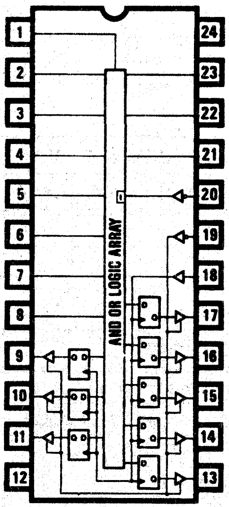
PLE10R8



CD01621M

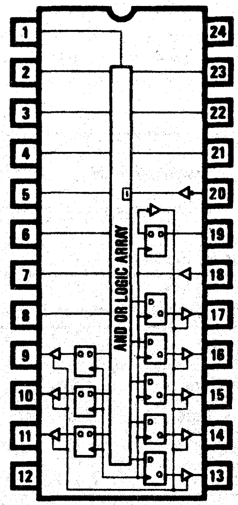
Logic Symbols

PLE11RA8



CD01631M

PLE11RS8



CD01641M

Absolute Maximum Ratings

	Operating	Programming
Supply voltage V_{CC}	−0.5V to 7V	12V
Input voltage	−1.5V to 7V	7V
Off-state output voltage	−0.5V to 5.5V	12V
Storage temperature	−65° to +150°C	

Operating Conditions

SYMBOL	PARAMETER	COMMERCIAL			MILITARY			UNIT
		MIN	NOM	MAX	MIN	NOM	MAX	
V_{CC}	Supply Voltage	4.75	5	5.25	4.5	5	5.5	V
T_A	Operating free-air temperature	0	25	75	−55	25	125	°C

Electrical Characteristics Over Operating Conditions

SYMBOL	PARAMETER	TEST CONDITIONS			MIN	TYP†	MAX	UNIT
V_{IL}	Low-level input voltage	Guaranteed logical low voltage for all inputs					0.8	V
V_{IH}	High-level input voltage	Guaranteed logical high voltage for all inputs			2			V
V_{IC}	Input clamp voltage	V_{CC} = MIN	I_I = −18mA				−1.5	V
I_{IL}	Low-level input current	V_{CC} = MAX	V_I = 0.4V				−0.25	mA
I_{IH}	High-level input current	V_{CC} = MAX	V_I = V_{CC}				40	μA
V_{OL}	Low-level output voltage	V_{CC} = MIN	I_{OL} = 16mA	Com			0.45	V
				Mil			0.5	
V_{OH}	High-level output voltage	V_{CC} = MIN	Com I_{OH} = −3.2mA		2.4			V
			Mil I_{OH} = −2mA					
I_{OZL}	Off-state output current	V_{CC} = MAX	V_O = 0.4V				−40	μA
I_{OZH}			V_O = 2.4V				40	
I_{OS}	Output short-circuit current*	V_{CC} = 5V	V_O = 0V		−20		−90	mA
I_{CC}	Supply current	V_{CC} = MAX All inputs grounded; all outputs open	5P8			90	125	mA
			5P8A			90	125	
			5P16			140	180	
			6P16			150	190	
			8P4			80	130	
			8P8			90	140	
			9P4			80	130	
			9P8			104	155	
			10P4			95	140	
			10P8			92	160	
			11P4			110	150	
			11P8			135	185	
			12P4			130	175	
			12P8			150	190	
		V_{CC} = MAX All inputs TTL; all outputs open	9R8			130	180	
			10R8			130	180	
			11RA8			140	185	
			11RS8			140	185	

† Typicals at 5.0V V_{CC} and 25°C T_A.

* Not more than one output should be shorted at a time and duration of the short circuit should not exceed one second.

Switching Characteristics Over Commercial Operating Conditions

DEVICE TYPE	t_{PD}(ns) PROPAGATION DELAY MAX	t_{PZX} AND t_{PXZ} (ns) INPUT TO OUTPUT ENABLE/DISABLE TIME MAX
5P8AC	15	20
5P8C	25	20
5P16C	18	15
6P16C	20	15
8P4C	30	20
8P8C	28	25
9P4C	35	20
9P8C	30	25
10P4C	35	25
10P8C	30	25
11P4C	35	25
11P8C	35	25
12P4C	35	25
12P8C	35	30

Switching Characteristics Over Military Operating Conditions

DEVICE TYPE	t_{PD}(ns) PROPAGATION DELAY MAX	t_{PZX} AND t_{PXZ} (ns) INPUT TO OUTPUT ENABLE/DISABLE TIME MAX
5P8M	35	30
8P4M	40	30
8P8M	40	30
9P4M	45	30
9P8M	40	30
10P4M	50	30
10P8M	45	30
11P4M	50	30
11P8M	50	30
12P4M	50	30
12P8M	40	35

Operating Conditions

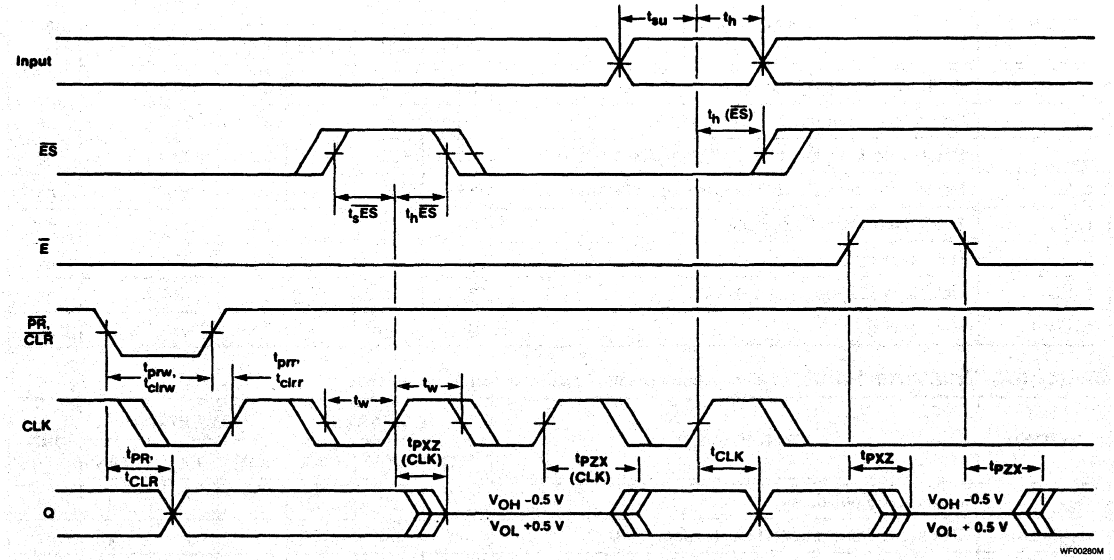
SYMBOL	PARAMETER	MILITARY			COMMERCIAL			UNIT
		MIN	TYP*	MAX	MIN	TYP*	MAX	
t_w	Width of clock (High or Low)	20	10		20	10		ns
t_{prw}	Width of preset or clear (LOW) to Output (High or Low)	20	10		20	10		ns
t_{clrw}								
t_{prr}	Recovery from preset or clear (LOW) to clock HIGH	25	11		20	11		ns
t_{clrr}								
t_{su}	Setup time from input to clock	35	22		30	22		ns
$t_s(\overline{ES})$	Setup time from $\overline{ES}$ to clock	15	7		10	7		ns
t_h	Hold time from input to clock	0	−5		0	−5		ns
$t_h(\overline{ES})$	Hold time from $\overline{ES}$ to clock	5	−3		5	−3		ns

Switching Characteristics Over Operating Conditions and using Standard Test Load

SYMBOL	PARAMETER	MILITARY			COMMERCIAL			UNIT
		MIN	TYP*	MAX	MIN	TYP*	MAX	
t_{CLK}	Clock to output delay		11	20		11	15	ns
t_{PR}	Preset to output delay		15	25		15	25	ns
t_{CLR}	Clear to output delay		18	35		18	25	ns
t_{PZX} (CLK)	Clock to output enable time		14	30		14	25	ns
t_{PXZ} (CLK)	Clock to output disable time		14	30		14	25	ns
t_{PZX}	Input to output enable time		10	25		10	20	ns
t_{PXZ}	Input to output disable time		10	25		10	20	ns

* Typicals at 5.0V V_{CC} and 25°C T_A.

Definition of Waveforms



NOTES:
1. Input pulse amplitude 0V to 3.0V.
2. Input rise and fall times 2-5ns from 0.8V to 2.0V.
3. Input access measured at the 1.5V level.
4. Switch S_1 is closed. C_L = 30pF and outputs measured at 1.5V level for all tests except t_{PXZ} and t_{PZX}.
5. t_{PZX} and $t_{PZX(CLK)}$ are measured at the 1.5V output level with C_L = 30pF. S_1 is open for high impedance to "1" test and closed for high impedance to "0" test.

   t_{PXZ} and $t_{PXZ(CLK)}$ are tested with C_L = 5pF. S_1 is open for "1" to high impedance test, measured at V_{OH} – 0.5V output level; S_1 is closed for "0" to high impedance test measured at V_{OL} + 0.5V output level.

Operating Conditions

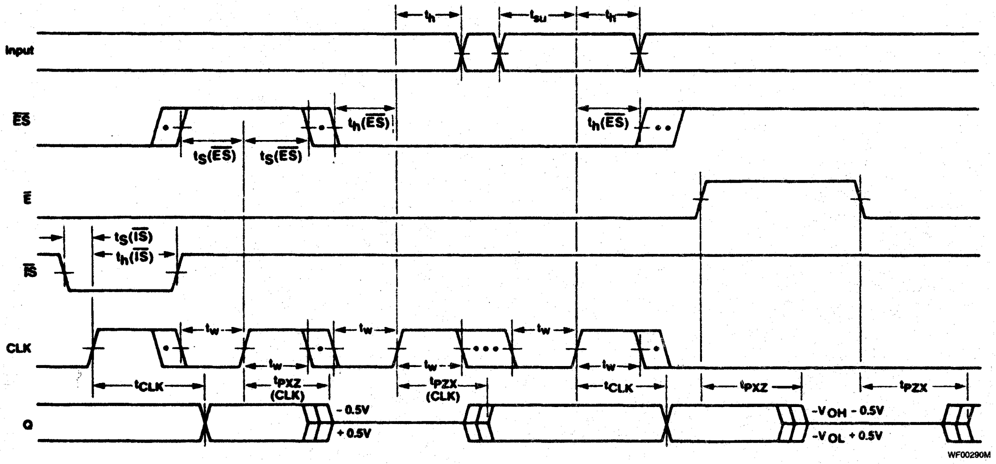
SYMBOL	PARAMETER	MILITARY			COMMERCIAL			UNIT
		MIN	TYP*	MAX	MIN	TYP*	MAX	
t_w	Width of clock (High or Low)	20	10		20	10		ns
t_{su}	Setup time from input to clock (10R8)	40	25		30	25		ns
t_{su}	Setup time from input to clock (11RA8, 11RS8)	40	28		35	28		ns
$t_s(\overline{ES})$	Setup time from $\overline{ES}$ to clock (except 11RA8)	15	8		15	8		ns
$t_s(\overline{IS})$	Setup time from $\overline{IS}$ to clock	30	20		25	20		ns
t_h	Hold time input to clock	0	-5		0	-5		ns
$t_h(\overline{ES})$	Hold time ($\overline{ES}$) (except 11RA8)	5	-3		5	-3		ns
$t_h(\overline{IS})$	Hold time ($\overline{IS}$)	0	-5		0	-5		ns

Switching Characteristics Over Operating Conditions and using Standard Test Load

SYMBOL	PARAMETER	MILITARY			COMMERCIAL			UNIT
		MIN	TYP*	MAX	MIN	TYP*	MAX	
t_{CLK}	Clock to output delay		10	20		10	15	ns
t_{PZX}(CLK)	Clock to output enable time (except 11RA8)		18	30		18	25	ns
t_{PXZ}(CLK)	Clock to output disable time (except 11RA8)		17	30		17	25	ns
t_{PZX}	Input to output enable time (except 11RS8)		17	30		17	25	ns
t_{PXZ}	Input to output disable time (except 11RS8)		17	30		17	25	ns

* Typicals at 5.0V V_{CC} and 25°C T_A.

Definition of Waveforms

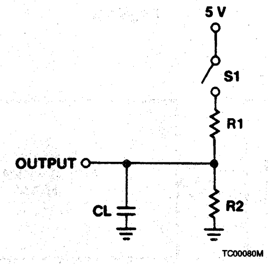


NOTES:

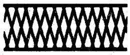
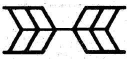
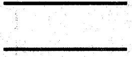
1. Input pulse amplitude 0V to 3.0V.
2. Input rise and fall times 2-5ns from 0.8V to 2.0V.
3. Input access measured at the 1.5V level.
4. Switch S_1 is closed. C_L = 30pF and outputs measured at 1.5V level for all tests except t_{PZX} and t_{PXZ}.
5. t_{PZX} and $t_{PZX(CLK)}$ are measured at the 1.5V output level with C_L = 30pF. S_1 is open for high impedance to "1" test and closed for high impedance to "0" test.

   t_{PXZ} and $t_{PXZ(CLK)}$ are tested with C_L = 5pF.S_1 is open for "1" to high impedance test, measured at V_{OH} - 0.5V output level; S_1 is closed for "0" to high impedance test measured at V_{OL} + 0.5V output level.

Switching Test Load

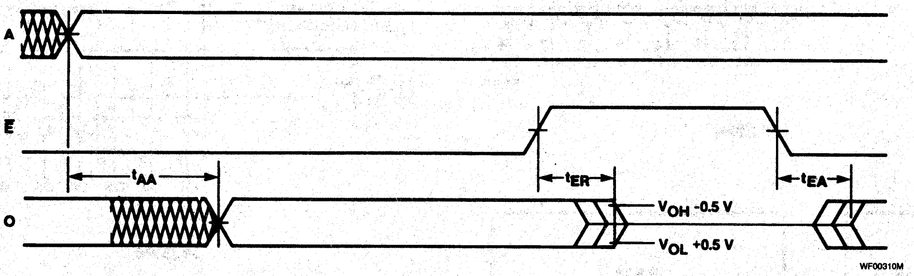


Definition of Timing Diagram

WAVEFORM	INPUTS	OUTPUTS
	DON'T CARE; CHANGE PERMITTED	CHANGING; STATE UNKNOWN
	NOT APPLICABLE	CENTER LINE IS HIGH IMPEDANCE STATE
	MUST BE STEADY	WILL BE STEADY

WF00300M

Definition of Waveforms

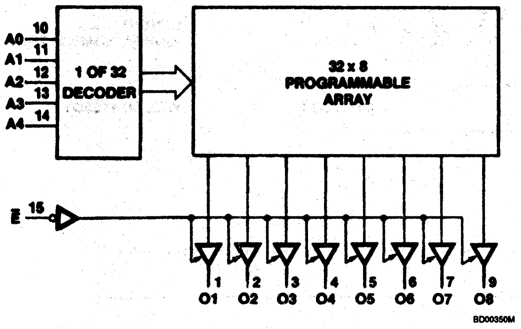


NOTES: Apply to electrical and switching characteristics. Typical at 5.0V V_{CC} and 25°C T_A. Measurements are absolute voltages with respect to the ground pin on the device and includes all overshoots due to system and/or tester noise. In all PLE devices unused inputs must be tied to either ground or V_{CC}. The series resistor required for unused inputs on standard TTL is NOT required for PLE devices, thus using less parts. *Not more than one output should be shorted at a time and duration of the short-circuit should not exceed one second. For commercial operating range R_1 = 200Ω, R_2 = 390Ω. For military operating range R_1 = 300Ω, R_2 = 600Ω.

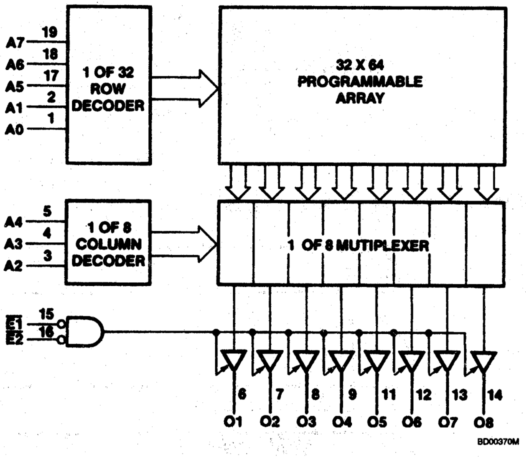
1. Input pulse amplitude 0V to 3.0V.
2. Input rise and fall times 2-5ns from 0.8 to 2.0V.
3. Input access measured at the 1.5V level.
4. Data delay is tested with switch S_1 closed. C_L = 30pF and measured at 1.5V output level.
5. t_{PZX} is measured at the 1.5V output level with C_L = 30pF. S_1 is open for high-impedance to "1" test and closed for high-impedance to "0" test. t_{PXZ} is measured C_L = 5pF. S_1 is open for "1" to high-impedance test, measured at V_{OH} - 0.5V output level; S_1 is closed for "0" to high-impedance test measured at V_{OL} + 0.5V output level.

Block Diagrams

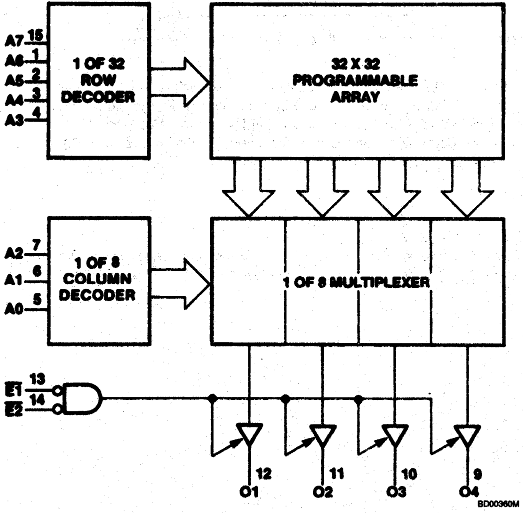
PLE5P8/A



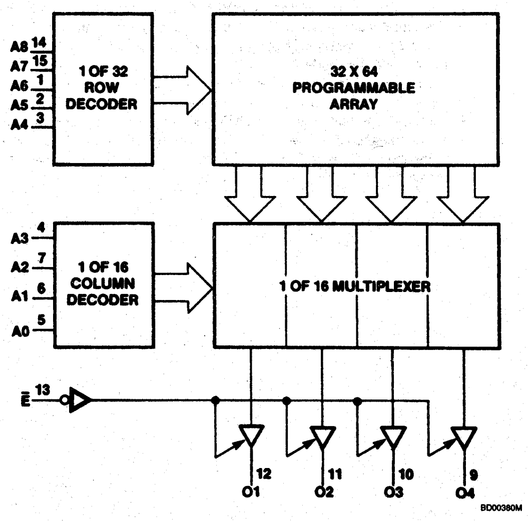
PLE8P8



PLE8P4

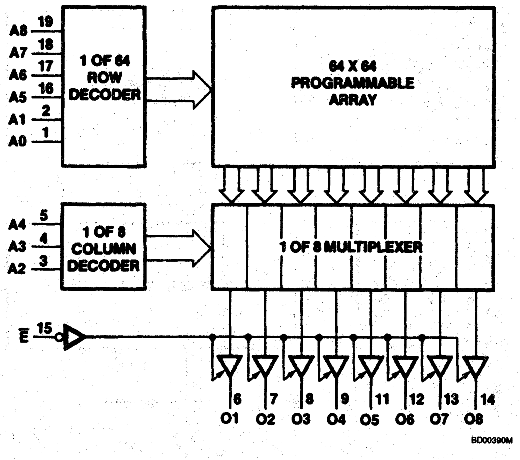


PLE9P4

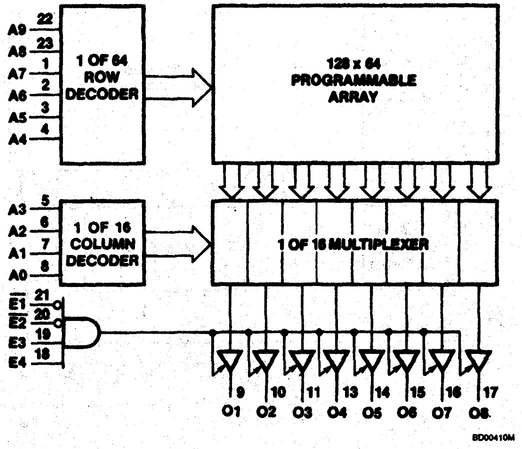


Block Diagrams

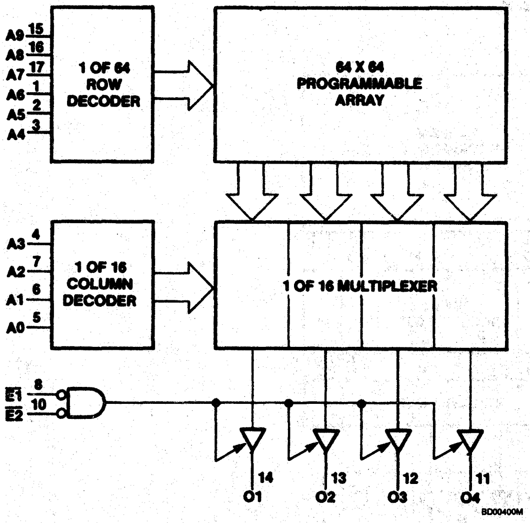
PLE9P8



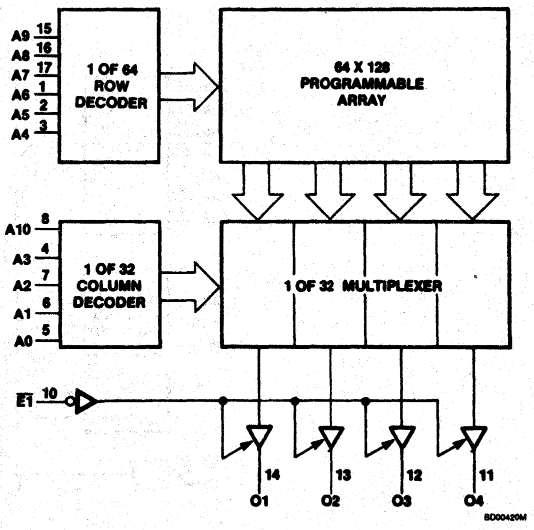
PLE10P8



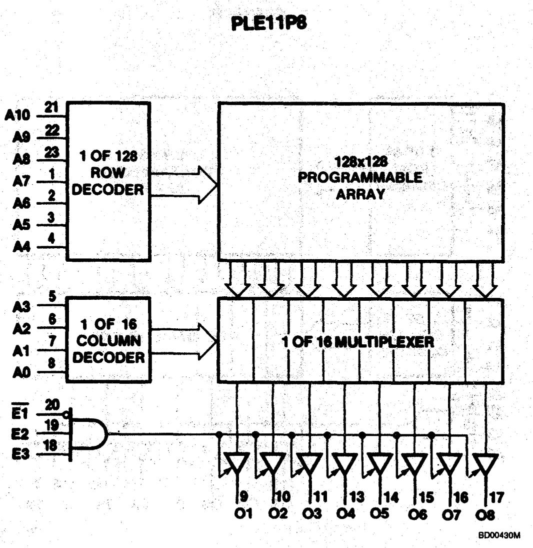
PLE10P4

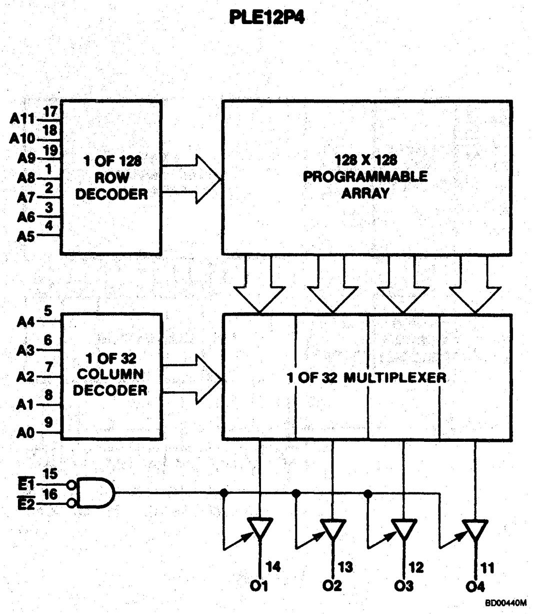


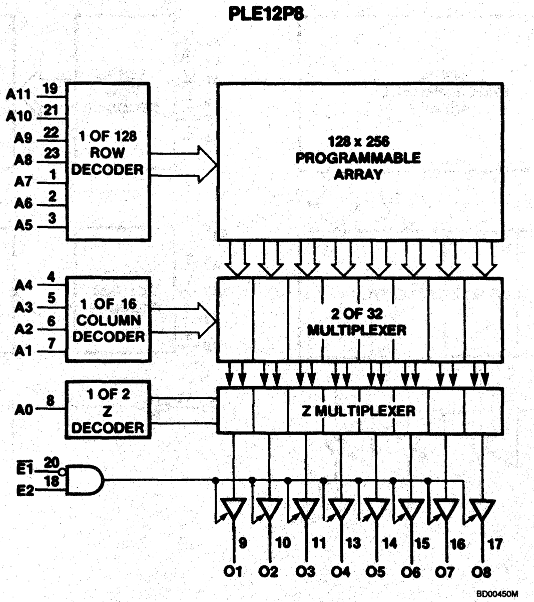
PLE11P4



Block Diagrams

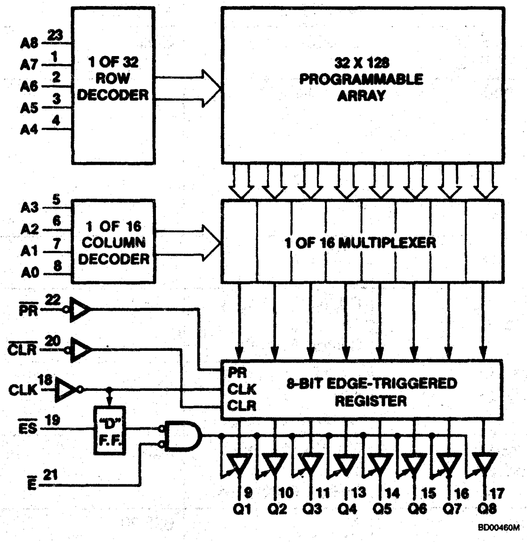




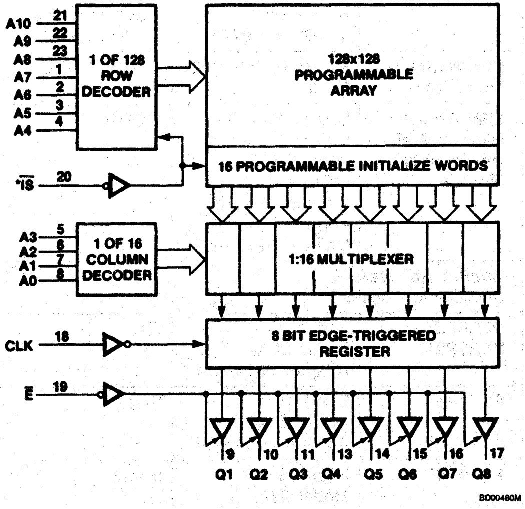


Block Diagrams

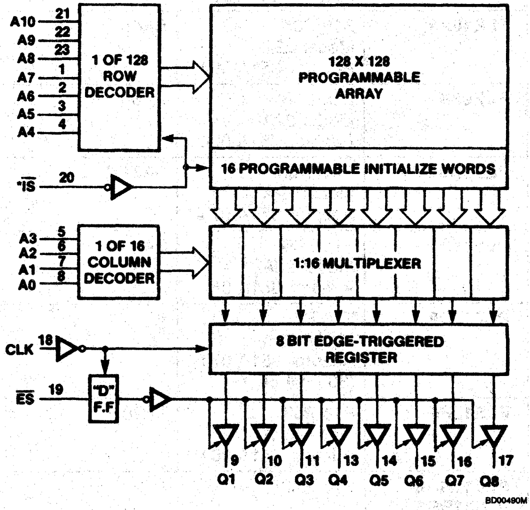
PLE9R8



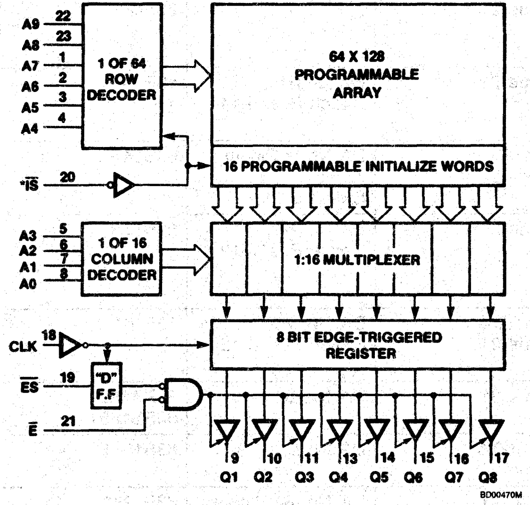
PLE11RA8



PLE11RS8



PLE10R8



Monolithic Memories PLE Programmer Reference Chart

SOURCE AND LOCATION	**Data I/O Corp. 10525 Willows Rd. N.E. P.O. Box 97046 Redmond WA 98073-9746**	**Kontron Electronics Inc. 1230 Charleston Rd. Mountain View CA 94039-7230**	**Stag Microsystems Inc. 528-5 Weddell Dr. Sunnyvale CA 94089**	**Digelec Inc. 1602 Lawrence Ave. Suite 113 Ocean NJ 07712**	**Varix Corp. 1210 E. Campbell Rd. Suite 100 Richardson TX 75081**
Programmer Model(s)	Model 19/29A/29B Model 22	Model MMP-80S	Model PPX Model PPZ	UP803	OMNI
MMI Generic Bipolar PLE Personality Module	UniPak Rev 10 UniPak II Rev 07 (Not all PLE devices are supported by earlier UniPak revisions)	MOD16	Zm 2000	FAM Mod. No. 12	
Socket Adapter(s) and Device Code					
PLE5P8/ PLE5P8A	F18 P02 Model 22A- Adapter 351A-064	SA3	AM110-2 Code 21	DA No. 2 Pinout 1A Switch Pos. O-6	63S081
PLE8P4	F18 P01 Model 22A- Adapter 351A-064	SA4-2	AM130-2 Code 21	DA No. 1 Pinout 1B Switch Pos. 0-6	63S141
PLE9P4	F18 P03 Model 22A- Adapter 351A-064	SA4-1	AM130-3 Code 21	DA No. 1 Pinout 1D Switch Pos. 2-14	63S241
PLE8P8	F18 P08 Model 22A- Adapter 351A-064	SA6-1	Code 21	†	63S281
PLE10P4	F18 P05 Model 22A- Adapter 351A-064	SA4	AM140-2 Code 21	DA No. 3 Pinout 1E Switch Pos. 0-6	63S441
PLE9P8	F18 P08 Model 22A- Adapter 351A-064	SA6	Code 21	†	63S481
PLE9R8	FEC P65 Model 22A- Adapter 351A-074	SA31-2	Code 21	Pinout 1H Switch Pos. 5-14	63RA481
PLE11P4	F18 P06 Model 22A- Adapter 351A-064	SA4-4	AM140-3 Code 21	DA No. 3 Pinout 1L Switch 5-14	63S841
PLE10R8	F18 P86 Model 22A- Adapter 351A-074 (300 mil pkg)	†	Code 21	DA No. 64† Switch Pos. 0-12	63RS881
PLE12P4	F18 P53 Model 22A- Adapter 351A-064	SA20	AM120-6 Code 21	DA No. 70 Switch Pos. 4-12	63S1641
PLE11RA8 PLE11RS8	F18 PA3	†	Code 21	DA No. 64	63RA1681 63RS1681
PLE11P8	F18P21	SA5-4	AM100-5 Code 21	DA No.7	63S1681
PLE12P8	F18P63	†	Code 21	DA No. 64 Pinout 47 Switch Pos. 0-4	63S3281

† Contact manufacturer for availability and programming information.

Table of Contents

Random Logic Replacement

PROMs, as logic elements, have been providing solutions as replacements of random logic. This is the concept of PROM as a Programmable Logic Element (PLE) device.

The usages of PLE devices include simple multiplexer/demultiplexer/encoder/decoder, control signal generators, data communications support like CRC, and arithmetic elements like ALUs, multipliers, sine and inverse look-up tables, and applications in signal processing.

The advantages of PLE devices over SSI/MSI logic devices are the flexibility of design and the fast turnaround time which non-programmable devices cannot offer. For example, if a decoder is used to select between memory pages and I/O ports, once a design is done, it will be fixed — it is not easy to find a part to be put just in the same place without modification of PC board layout in case the designer wants to expand the memory or to increase the I/O. For a PLE device, what is needed is to program another PLE device and place it in the same socket where the old part was placed. In addition, it can allow designers to define their logic functions in a component.

The AND-OR planar structure of the PLE circuit array lends itself naturally to being viewed as a two-level logic circuit. The fixed AND plane contains all possible combinations of the literals of its inputs. Each combination (product term) is fuse-connected to each output in the programmable OR plane.

A common PLE device application in the control path is to customize logic functions. An n input exclusive OR function is quite commonly required in comparator and adder circuits. It contains 2^{n-1} product terms, which becomes quite large for large values of n. Therefore, it is very convenient to implement large XOR functions in PLE devices.

The PLE logic circuit implementation of a 4-input XOR is shown below.

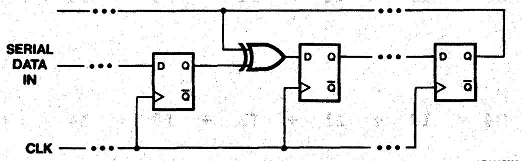
ab \ cd	00	01	11	10
00	0	1	0	1
01	1	0	1	0
11	0	1	0	1
10	1	0	1	0

TB01070M

Although it seems that XOR functions may be replaced by SSIs, in most applications, the XOR functions will not be alone by themselves, PLE circuits can provide the flexibility of adding in additional functions without using additional packages.

In the data path, a PLE device can be used to implement complex functions such as a Pseudo Random Number (PRN) Generator. Random number sequences are useful in encoding and decoding of information in signal processing and communications systems. They are used for data encryption, image quantization, waveform synchronization, and white noise generation, etc.

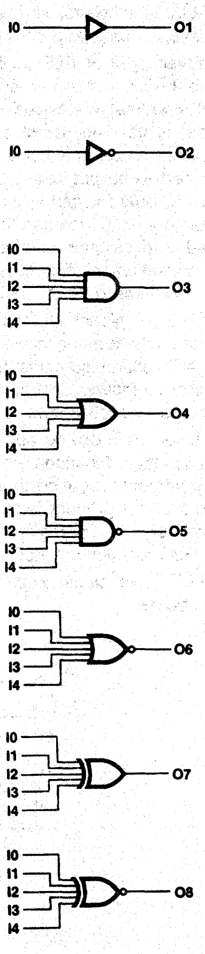
There are many techniques for generating PRN sequences. The most common technique, however, is to use 'n' stages of linear shift registers with feedback through a logic function. The function f is an arbitrary function chosen for a specific application. A most general linear function is an 'm' input XOR (m ≤ n).



LD00850M

There are a number of examples in the following session which shows how a PLE device can be used to replace SSI/MSI logic devices using PLEASM software.

```
PLE5P8                                           PLE CIRCUIT DESIGN SPECIFICATION
P5000                                                      VINCENT COLI 10/03/83
BASIC GATES
MMI SANTA CLARA, CALIFORNIA
.ADD I0 I1 I2 I3 I4
.DAT O1 O2 O3 O4 O5 O6 O7 O8

O1 =  I0                                        ; BUFFER


O2 = /I0                                        ; INVERTER


O3 =  I0   *  I1   *  I2   *  I3   *  I4        ; AND GATE


O4 =  I0   +  I1   +  I2   +  I3   +  I4        ; OR GATE


O5 = /I0   + /I1   + /I2   + /I3   + /I4        ; NAND GATE


O6 = /I0   * /I1   * /I2   * /I3   * /I4        ; NOR GATE


O7 =  I0 :+: I1 :+: I2 :+: I3 :+: I4            ; EXCLUSIVE OR GATE


O8 =  I0 :*: I1 :*: I2 :*: I3 :*: I4            ; EXCLUSIVE NOR GATE
```



TB01080M

```
FUNCTION TABLE

I0 I1 I2 I3 I4 O1 O2 O3 O4 O5 O6 O7 O8

;INPUT   - -  OUTPUTS  FROM  BASIC  GATES  - -
;01234   BUF  INV  AND  OR  NAND  NOR  XOR  XNOR   COMMENTS
-------------------------------------------------------------------
 LLLLL    L    H    L    L    H     H    L    L     ALL ZEROS
 HHHHH    H    L    H    H    L     L    H    H     ALL ONES
 HLHLH    H    L    L    H    H     L    H    H     ODD CHECKERBOARD
 LHLHL    L    H    L    H    H     L    L    L     EVEN CHECKERBOARD
-------------------------------------------------------------------
```

DESCRIPTION

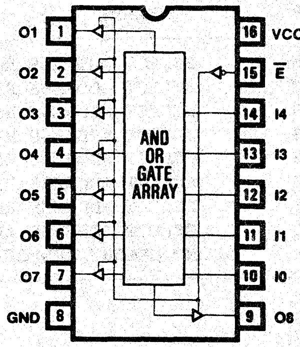
THIS EXAMPLE ILLUSTRATES THE USE OF PLE DEVICES TO IMPLEMENT THE BASIC GATES: BUFFER, INVERTER, AND GATE, OR GATE, NAND GATE, NOR GATE, EXCLUSIVE OR GATE, AND EXCLUSIVE NOR GATE.

NOTE ALSO THAT THREE-STATE OUTPUTS ARE PROVIDED WITH ONE ACTIVE LOW OUTPUT ENABLE CONTROL (/E).

PLEASM SOFTWARE GENERATES THE PROM TRUTH TABLE FROM THE LOGIC EQUATIONS AND SIMULATES THE FUNCTION TABLE IN THE LOGIC EQUATIONS.

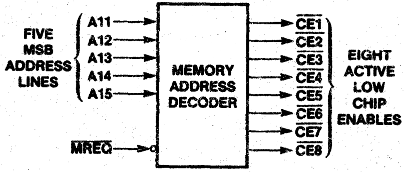
BASIC GATES

PLE5P8



TB01090M

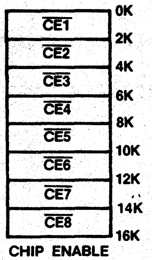
```
PLE8P8                                           PLE CIRCUIT DESIGN SPECIFICATION
P5001                                               ULRIK MUELLER 05/01/84
MEMORY ADDRESS DECODER
MMI SANTA CLARA, CALIFORNIA
.ADD A11 A12 A13 A14 A15 /MREQ
.DAT /CE1 /CE2 /CE3 /CE4 /CE5 /CE6 /CE7 /CE8

CE1 = /A11*/A12*/A13*/A14*/A15* MREQ      ; SELECTS ADDRESS RANGE 0K-2K

CE2 =  A11*/A12*/A13*/A14*/A15* MREQ      ; SELECTS ADDRESS RANGE 2K-4K

CE3 = /A11* A12*/A13*/A14*/A15* MREQ      ; SELECTS ADDRESS RANGE 4K-6K

CE4 =  A11* A12*/A13*/A14*/A15* MREQ      ; SELECTS ADDRESS RANGE 6K-8K

CE5 = /A11*/A12* A13*/A14*/A15* MREQ      ; SELECTS ADDRESS RANGE 8K-10K

CE6 =  A11*/A12* A13*/A14*/A15* MREQ      ; SELECTS ADDRESS RANGE 10K-12K

CE7 = /A11* A12* A13*/A14*/A15* MREQ      ; SELECTS ADDRESS RANGE 12K-14K

CE8 =  A11* A12* A13*/A14*/A15* MREQ      ; SELECTS ADDRESS RANGE 14K-16K

FUNCTION TABLE

A11 A12 A13 A14 A15 /MREQ /CE1 /CE2 /CE3 /CE4 /CE5 /CE6 /CE7 /CE8

; ADD LINES
;   11111               CHIP ENABLES
;   12345      /MREQ      12345678        COMMENTS
-----------------------------------------------------------------------
    LLLLL        L        LHHHHHHH        SELECT ADDRESS RANGE 0-2K
    HLLLL        L        HLHHHHHH        SELECT ADDRESS RANGE 2K-4K
    LHLLL        L        HHLHHHHH        SELECT ADDRESS RANGE 4K-6K
    HHLLL        L        HHHLHHHH        SELECT ADDRESS RANGE 6K-8K
    LLHLL        L        HHHHLHHH        SELECT ADDRESS RANGE 8K-10K
    HLHLL        L        HHHHHLHH        SELECT ADDRESS RANGE 10K-12K
    LHHLL        L        HHHHHHLH        SELECT ADDRESS RANGE 12K-14K
    HHHLL        L        HHHHHHHL        SELECT ADDRESS RANGE 14K-16K
    XXXXX        H        HHHHHHHH        NO MEMORY SELECT (/MREQ=H)
-----------------------------------------------------------------------
```



TB01100M

DESCRIPTION

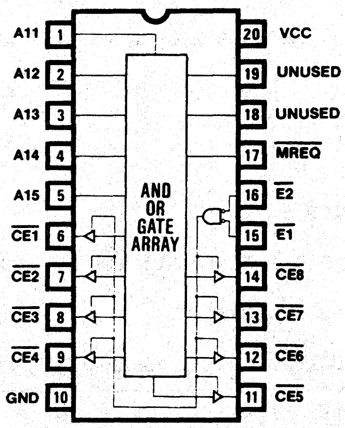
THIS PLE8P8 PROVIDES A SINGLE CHIP ADDRESS DECODER FOR USE WITH MANY POPULAR 8-BIT MICROPROCESSORS SUCH AS THE Z80 AND 8080. THE FIVE MSB ADDRESS LINES (A11-A15) AND THE MEMORY REQUEST LINE (/MREQ) FROM THE Z80 MICROPROCESSOR ARE DECODED TO PRODUCE EIGHT ACTIVE LOW CHIP ENABLES (/CE1-/CE8) TO SELECT A RANGE OF 2K BYTES FROM A BANK OF EIGHT 2Kx8 STATIC RAMS. THIS BANK OF STATIC RAMS WILL OCCUPY THE LOWEST 16K BYTES OF ADDRESS SPACE LEAVING THE UPPER 48K BYTE SPACE AVAILABLE FOR OTHER MEMORIES AND I/O. THE PLE8P8 HAS THREE ADDITIONAL INPUTS WHICH CAN BE RESERVED FOR FUTURE SYSTEM EXPANSION.

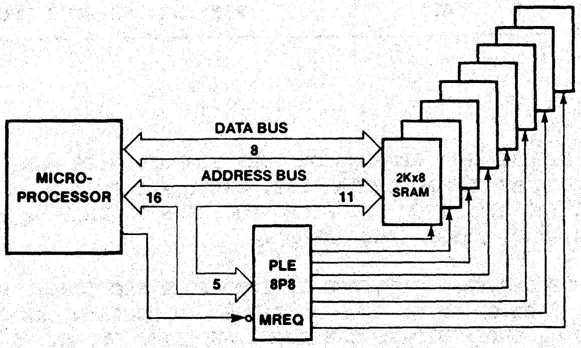


CHIP ENABLE ADDRESS MAP

MEMORY ADDRESS DECODER

PLE8P8



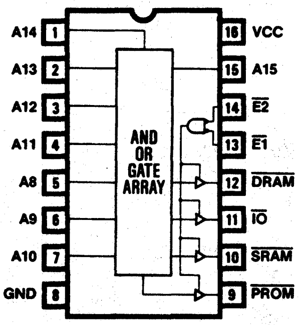


TB01110M

```
PLE8P4                                        PLE CIRCUIT DESIGN SPECIFICATION
P5029                                                      VINCENT COLI 10/13/84
6809 ADDRESS DECODER
MMI SANTA CLARA, CALIFORNIA
.ADD A8 A9 A10 A11 A12 A13 A14 A15
.DAT /DRAM /IO /SRAM /PROM


DRAM = /A8*                A12* A13*/A14* A15  ; SELECTS ADDRESS RANGE 0000-BEFF
     +     /A9*            A12* A13*/A14* A15
     +         /A10*       A12* A13*/A14* A15
     +              /A11*  A12* A13*/A14* A15
     +                    /A12*     /A14
     +                         /A13*/A14
     +                                   /A15

IO   =  A8* A9* A10* A11* A12* A13*/A14* A15   ; SELECTS ADDRESS RANGE BF00-BFFF

SRAM =                         /A13* A14* A15  ; SELECTS ADDRESS RANGE C000-DFFF

PROM =                          A13* A14* A15  ; SELECTS ADDRESS RANGE E000-FFFF


FUNCTION TABLE

A8 A9 A10 A11 A12 A13 A14 A15 /DRAM /IO /SRAM /PROM

; ADDRESS LINES
;    11 1111
;   8901 2345     /DRAM  /IO  /SRAM  /PROM     COMMENTS
-----------------------------------------------------------------------------
    LLLL LLLL       L     H     H      H       00XX HEX SELECTS DRAMS
    LLLL HHLH       L     H     H      H       B0XX HEX SELECTS DRAMS
    HHHH HHLH       H     L     H      H       BFXX HEX SELECTS I/O PORTS
    LLLL LLHH       H     H     L      H       C0XX HEX SELECTS SRAM
    LLLL HLHH       H     H     L      H       D0XX HEX SELECTS SRAM
    LLLL LHHH       H     H     H      L       E0XX HEX SELECTS PROM
    HHHH HHHH       H     H     H      L       FFXX HEX SELECTS PROM
-----------------------------------------------------------------------------


DESCRIPTION

THIS PLE8P4 PROVIDES A SINGLE CHIP ADDRESS DECODER FOR USE WITH MANY POPULAR
8-BIT MICROPROCESSORS SUCH AS THE MOTOROLA 6809.  THIS PLE DEVICE DECODES THE
EIGHT MSB ADDRESS LINES (A8-A15) FROM THE MICROPROCESSOR TO PROVIDE FOUR ACTIVE
LOW CHIP ENABLES (/DRAM, /IO, /SRAM, AND /PROM).

THE 64K MEMORY MAP OF THE SYSTEM IS DIVIDED UP INTO FOUR SECTIONS:  DRAM, IO
PORTS, SRAM, AND PROM.  EACH OF THESE FOUR SECTIONS CAN CONTAIN ONE OR MORE
BLOCKS OF MEMORY.  EACH OF THESE BLOCKS CAN START AND STOP ON ANY 256 BIT
BOUNDARY
```

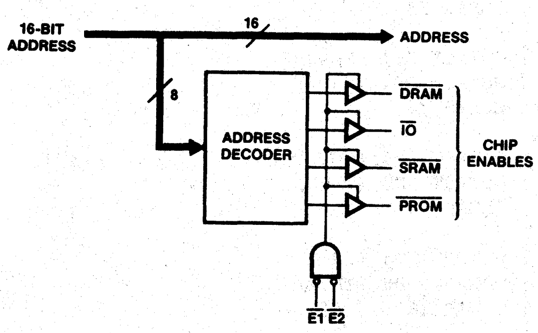
TB01120M

6809 Address Decoder
PLE8P4



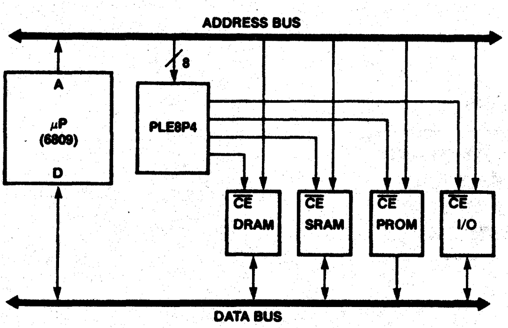
CD00890M

Block Diagram



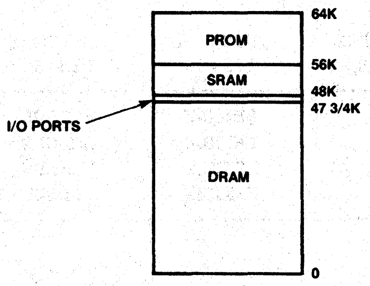
BD00050M

System Diagram



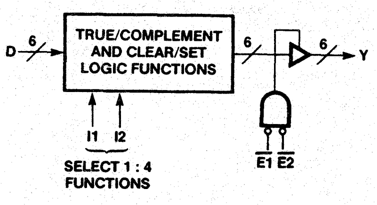
AF00010M

Memory Map



AF00020M

```
PLE8P8                                 PLE CIRCUIT DESIGN SPECIFICATION
P5002                                  JOEL ROSENBERG 10/26/83
6-BIT TRUE/COMPLEMENT AND CLEAR/SET LOGIC FUNCTIONS
MMI SANTA CLARA, CALIFORNIA
.ADD I1 I2 D1 D2 D3 D4 D5 D6
.DAT Y1 Y2 Y3 Y4 Y5 Y6


Y1 = /I1*/I2*/D1        ; OUTPUT /D1 (INVERT)
   + /I1* I2* D1        ; OUTPUT  D1 (TRUE)
   +  I1*/I2            ; CLEAR Y1

Y2 = /I1*/I2*/D2        ; OUTPUT /D2 (INVERT)
   + /I1* I2* D2        ; OUTPUT  D2 (TRUE)
   +  I1*/I2            ; CLEAR Y2

Y3 = /I1*/I2*/D3        ; OUTPUT /D3 (INVERT)
   + /I1* I2* D3        ; OUTPUT  D3 (TRUE)
   +  I1*/I2            ; CLEAR Y3

Y4 = /I1*/I2*/D4        ; OUTPUT /D4 (INVERT)
   + /I1* I2* D4        ; OUTPUT  D4 (TRUE)
   +  I1*/I2            ; CLEAR Y4

Y5 = /I1*/I2*/D5        ; OUTPUT /D5 (INVERT)
   + /I1* I2* D5        ; OUTPUT  D5 (TRUE)
   +  I1*/I2            ; CLEAR Y5

Y6 = /I1*/I2*/D6        ; OUTPUT /D6 (INVERT)
   + /I1* I2* D6        ; OUTPUT  D6 (TRUE)
   +  I1*/I2            ; CLEAR Y6


FUNCTION TABLE

I1 I2 D1 D2 D3 D4 D5 D6 Y1 Y2 Y3 Y4 Y5 Y6

;CONTROL    INPUT  D    OUTPUT Y
; LINES      123456      123456      COMMENTS
------------------------------------------------------
  LL         LHLHLH      HLHLHL      INVERT FUNCTION
  LH         LHLHLH      LHLHLH      TRUE FUNCTION
  HL         XXXXXX      HHHHHH      CLEAR FUNCTION
  HH         XXXXXX      LLLLLL      SET FUNCTION
------------------------------------------------------
```



TB01130M

DESCRIPTION

THIS PLE8P8 IS A 6-BIT TRUE/COMPLEMENT AND CLEAR/SET LOGIC FUNCTIONS. THE CONTROL LINES (I1 AND I2) SELECT ONE OF FOUR LOGIC FUNCTIONS FOR THE 6-BIT INPUT DATA (D1-D6) AND THE 6-BIT OUTPUT FUNCTION (Y1-Y6).

WHEN I1 IS FALSE (I1=LOW) THE FUNCTION IS INVERT IF I2 IS FALSE (I2=LOW) OR TRUE IF I2 IS TRUE (I2=HIGH).

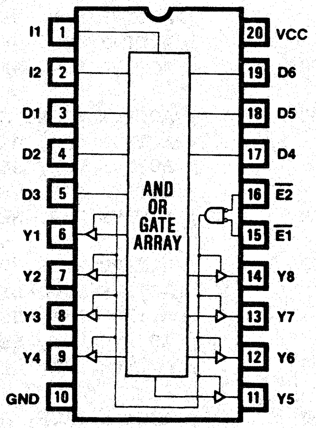
WHEN I1 IS TRUE (I1=HIGH) THE FUNCTION IS CLEAR IF I2 IS FALSE (I2=LOW) OR SET IF I2 IS TRUE (I2=HIGH).

THE PLE8P8 ALSO FEATURES THREE-STATE OUTPUTS WITH TWO ACTIVE LOW OUTPUT ENABLE CONTROLS (/E1 AND /E2).

I1	I2	D1-D6	Y1-Y6	FUNCTION
L	L	D	/D	INVERT
L	H	D	D	TRUE
H	L	X	H	CLEAR
H	H	X	L	SET

6-BIT TRUE/COMPLEMENT
ZERO/ONE LOGIC FUNCTIONS

PLE8P8



TB01140M

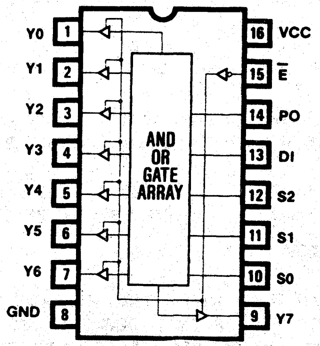
```
PLE5P8                                PLE CIRCUIT DESIGN SPECIFICATION
P5003                                             FRANK LEE 04/15/84
EXPANDABLE 3-TO-8 DEMULTIPLEXER
MMI SANTA CLARA, CALIFORNIA
.ADD S0 S1 S2 DI PO
.DAT Y0 Y1 Y2 Y3 Y4 Y5 Y6 Y7

Y0 =  PO *  DI * /S2 * /S1 * /S0         ; ACTIVE HIGH, SELECT 0
   + /PO *  DI                           ; ACTIVE LOW, DI INACTIVE
   + /PO       *  S2                     ; ACTIVE LOW, SELECT 4-7
   + /PO             *  S1               ; ACTIVE LOW, SELECT 2,3,6,7
   + /PO                   *  S0         ; ACTIVE LOW, SELECT 1,3,5,7

Y1 =  PO *  DI * /S2 * /S1 *  S0         ; ACTIVE HIGH, SELECT 1
   + /PO *  DI                           ; ACTIVE LOW, DI INACTIVE
   + /PO       *  S2                     ; ACTIVE LOW, SELECT 4,5,6,7
   + /PO             *  S1               ; ACTIVE LOW, SELECT 2,3,6,7
   + /PO                   * /S0         ; ACTIVE LOW, SELECT 0,2,4,6

Y2 =  PO *  DI * /S2 *  S1 * /S0         ; ACTIVE HIGH, SELECT 2
   + /PO *  DI                           ; ACTIVE LOW, DI INACTIVE
   + /PO       *  S2                     ; ACTIVE LOW, SELECT 4-7
   + /PO             * /S1               ; ACTIVE LOW, SELECT 0,1,4,5
   + /PO                   *  S0         ; ACTIVE LOW, SELECT 1,3,5,7

Y3 =  PO *  DI * /S2 *  S1 *  S0         ; ACTIVE HIGH, SELECT 3
   + /PO *  DI                           ; ACTIVE LOW, DI INACTIVE
   + /PO       *  S2                     ; ACTIVE LOW, SELECT 4-7
   + /PO             * /S1               ; ACTIVE LOW, SELECT 0,1,4,5
   + /PO                   * /S0         ; ACTIVE LOW, SELECT 0,2,4,6

Y4 =  PO *  DI *  S2 * /S1 * /S0         ; ACTIVE HIGH, SELECT 4
   + /PO *  DI                           ; ACTIVE LOW, DI INACTIVE
   + /PO       * /S2                     ; ACTIVE LOW, SELECT 0-3
   + /PO             *  S1               ; ACTIVE LOW, SELECT 2,3,6,7
   + /PO                   *  S0         ; ACTIVE LOW, SELECT 1,3,5,7

Y5 =  PO *  DI *  S2 * /S1 *  S0         ; ACTIVE HIGH, SELECT 5
   + /PO *  DI                           ; ACTIVE LOW, DI INACTIVE
   + /PO       * /S2                     ; ACTIVE LOW, SELECT 0-3
   + /PO             *  S1               ; ACTIVE LOW, SELECT 2,3,6,7
   + /PO                   * /S0         ; ACTIVE LOW, SELECT 0,2,4,6

Y6 =  PO *  DI *  S2 *  S1 * /S0         ; ACTIVE HIGH, SELECT 6
   + /PO *  DI                           ; ACTIVE LOW, DI INACTIVE
   + /PO       * /S2                     ; ACTIVE LOW, SELECT 0-3
   + /PO             * /S1               ; ACTIVE LOW, SELECT 0,1,4,5
   + /PO                   *  S0         ; ACTIVE LOW, SELECT 1,3,5,7

Y7 =  PO *  DI *  S2 *  S1 *  S0         ; ACTIVE HIGH, SELECT 7
   + /PO *  DI                           ; ACTIVE LOW, DI INACTIVE
   + /PO       * /S2                     ; ACTIVE LOW, SELECT 0-3
   + /PO             * /S1               ; ACTIVE LOW, SELECT 0,1,4,5
   + /PO                   * /S0         ; ACTIVE LOW, SELECT 0,2,4,6
```

TB01150M

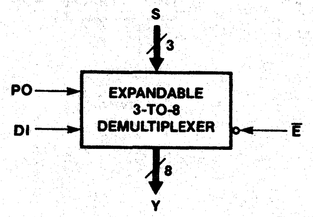
```
FUNCTION TABLE

PO DI S2 S1 S0 Y7 Y6 Y5 Y4 Y3 Y2 Y1 Y0

;       SSS     YYYYYYYY
;PO DI  210     76543210        COMMENTS
-----------------------------------------
 H  L   XXX     LLLLLLLL        DATA INPUT = 0
 H  H   LLL     LLLLLLLH        SELECT OUTPUT 0
 H  H   LLH     LLLLLLHL        SELECT OUTPUT 1
 H  H   LHL     LLLLLHLL        SELECT OUTPUT 2
 H  H   LHH     LLLLHLLL        SELECT OUTPUT 3
 H  H   HLL     LLLHLLLL        SELECT OUTPUT 4
 H  H   HLH     LLHLLLLL        SELECT OUTPUT 5
 H  H   HHL     LHLLLLLL        SELECT OUTPUT 6
 H  H   HHH     HLLLLLLL        SELECT OUTPUT 7
 L  H   XXX     HHHHHHHH        DATA INPUT = 0
 L  L   LLL     HHHHHHHL        SELECT OUTPUT 0
 L  L   LLH     HHHHHHLH        SELECT OUTPUT 1
 L  L   LHL     HHHHHLHH        SELECT OUTPUT 2
 L  L   LHH     HHHHLHHH        SELECT OUTPUT 3
 L  L   HLL     HHHLHHHH        SELECT OUTPUT 4
 L  L   HLH     HHLHHHHH        SELECT OUTPUT 5
 L  L   HHL     HLHHHHHH        SELECT OUTPUT 6
 L  L   HHH     LHHHHHHH        SELECT OUTPUT 7
-----------------------------------------
```

EXPANDABLE
3-TO-8 DEMULTIPLEXER

PLE5P8

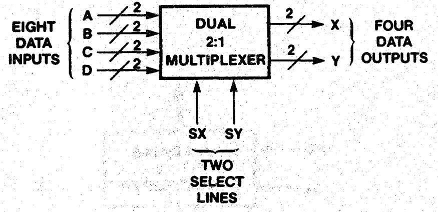


```
DESCRIPTION

THIS PLE5P8 IMPLEMENTS AN EXPANDABLE 3-TO-8 DEMULTIPLEXER.  THE DEVICE
DEMULTIPLEXES THREE SELECT INPUT SIGNALS (S2-S0) INTO EIGHT OUTPUTS (Y7-Y0)
USING THE INPUT DI WITH POLARITY SELECT PO.  SINCE THE DEVICE HAS THREE-STATE
OUTPUTS, IT CAN BE EXPANDED USING THE ACTIVE LOW ENABLE PIN (/E).

PIN ASSIGNMENTS:

1. PO       HIGH INDICATES OUTPUT IS ACTIVE HIGH.  LOW INDICATES OUTPUT IS
            ACTIVE LOW.
2. DI       DATA INPUT (DEMULTIPLEXING INPUT).  ACTIVE LOW IF PO IS LOW.
3. S2-S0    SELECT PINS.  S2 IS THE MOST SIGNIFICANT BIT. ACTIVE HIGH
            REGARDLESS OF PO.
4. Y7-Y0    OUTPUTS.  CAN BE ACTIVE HIGH OR ACTIVE LOW DEPENDING ON PO.
            ACTIVE LOW IF PO IS LOW.

OPERATIONS TABLE:

 PO  DI  S2-S0  Y7-Y0    OPERATION
------------------------------------------
 L   H     X      H      OUTPUTS HIGH
 H   H     S    DEMUX    DEMUX ACTIVE HIGH
 L   L     S   /DEMUX    DEMUX ACTIVE LOW
 H   L     X      L      OUTPUTS LOW
------------------------------------------
```



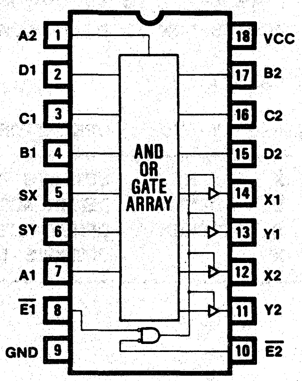
TB01160M

```
PLE10P4                                   PLE CIRCUIT DESIGN SPECIFICATION
P5004                                                 ULRIK MUELLER 04/01/83
DUAL 2:1 MULTIPLEXER
MMI SANTA CLARA, CALIFORNIA
.ADD SX SY A1 B1 C1 D1 A2 B2 C2 D2
.DAT X1 Y1 X2 Y2


X1 = /SX* A1        ; SELECT INPUT A1
   +  SX* B1        ; SELECT INPUT B1

Y1 = /SY* C1        ; SELECT INPUT C1
   +  SY* D1        ; SELECT INPUT D1

X2 = /SX* A2        ; SELECT INPUT A2
   +  SX* B2        ; SELECT INPUT B2

Y2 = /SY* C2        ; SELECT INPUT C2
   +  SY* D2        ; SELECT INPUT D2
DESCRIPTION

THIS IS AN EXAMPLE OF TWO INDEPENDENT 2-TO-1 MULTIPLEXERS USING A PLE10P4.
THE DEVICE WILL SWITCH BETWEEN TWO PAIRS OF 2-BIT INPUTS (A, B AND C, D), AS
DETERMINED BY THE TWO SELECT LINES (SX, SY), FOR OUTPUT THROUGH TWO PAIRS OF
2-BIT OUTPUTS (X AND Y).  THREE-STATE OUTPUTS ARE ALSO PROVIDED WITH TWO
ACTIVE LOW ENABLE PINS (/E1 AND /E2).  THE FUNCTIONS OF THE DEVICE ARE
SUMMARIZED IN THE TABLE BELOW:

SELECT    INPUT  A, B     INPUT  C, D          OUTPUT  X, Y
LINES
 S  S     A  A   B  B     C  C   D  D     !    X  Y   X  Y
 X  Y     1  2   1  2     1  2   1  2     !    1  1   2  2       FUNCTION
--------------------------------------------------------------------------
 L  L     A1 A2  X  X     C1 C2  X  X     !    A1 C1  A2 C2      SELECT A, C
 L  H     A1 A2  X  X     X  X   D1 D2    !    A1 D1  A2 D2      SELECT A, D
 H  L     X  X   B1 B2    C1 C2  X  X     !    B1 B2  C1 C2      SELECT B, C
 H  H     X  X   B1 B2    X  X   D1 D2    !    B1 D1  B2 D2      SELECT B, D
--------------------------------------------------------------------------
```



DUAL 2:1 MULTIPLEXER

PLE10P4



TB01170M

```
PLE10P4                          PLE CIRCUIT DESIGN SPECIFICATION
P5005                                           S. HORIKO 04/29/84
QUAD 2:1 MULTIPLEXER WITH POLARITY CONTROL
MMI JAPAN
.ADD SEL POL A0 A1 A2 A3 B0 B1 B2 B3
.DAT Y0 Y1 Y2 Y3


Y0 = /SEL*/POL*/A0             ; SELECT INPUT /A0 (COMP)
   + /SEL* POL* A0             ; SELECT INPUT  A0 (TRUE)
   +  SEL*/POL*/B0             ; SELECT INPUT /B0 (COMP)
   +  SEL* POL* B0             ; SELECT INPUT  B0 (TRUE)

Y1 = /SEL*/POL*/A1             ; SELECT INPUT /A1 (COMP)
   + /SEL* POL* A1             ; SELECT INPUT  A1 (TRUE)
   +  SEL*/POL*/B1             ; SELECT INPUT /B1 (COMP)
   +  SEL* POL* B1             ; SELECT INPUT  B1 (TRUE)

Y2 = /SEL*/POL*/A2             ; SELECT INPUT /A2 (COMP)
   + /SEL* POL* A2             ; SELECT INPUT  A2 (TRUE)
   +  SEL*/POL*/B2             ; SELECT INPUT /B2 (COMP)
   +  SEL* POL* B2             ; SELECT INPUT  B2 (TRUE)

Y3 = /SEL*/POL*/A3             ; SELECT INPUT /A3 (COMP)
   + /SEL* POL* A3             ; SELECT INPUT  A3 (TRUE)
   +  SEL*/POL*/B3             ; SELECT INPUT /B3 (COMP)
   +  SEL* POL* B3             ; SELECT INPUT  B3 (TRUE)


FUNCTION TABLE

SEL POL A0 A1 A2 A3 B0 B1 B2 B3 Y0 Y1 Y2 Y3

; SELECT    AAAA BBBB    YYYY
;SEL  POL   0123 0123    0123      COMMENTS
------------------------------------------------------------
  L    L    LLLL XXXX    HHHH      SELECT COMP INPUT /A=00
  L    L    LHLH XXXX    HLHL      SELECT COMP INPUT /A=05
  L    L    HLHL XXXX    LHLH      SELECT COMP INPUT /A=10
  L    L    HHHH XXXX    LLLL      SELECT COMP INPUT /A=15
  L    H    LLLL XXXX    LLLL      SELECT TRUE INPUT  A=00
  L    H    LHLH XXXX    LHLH      SELECT TRUE INPUT  A=05
  L    H    HLHL XXXX    HLHL      SELECT TRUE INPUT  A=10
  L    H    HHHH XXXX    HHHH      SELECT TRUE INPUT  A=15
  H    L    XXXX LLLL    HHHH      SELECT COMP INPUT /B=00
  H    L    XXXX LHLH    HLHL      SELECT COMP INPUT /B=05
  H    L    XXXX HLHL    LHLH      SELECT COMP INPUT /B=10
  H    L    XXXX HHHH    LLLL      SELECT COMP INPUT /B=15
  H    H    XXXX LLLL    LLLL      SELECT TRUE INPUT  B=00
  H    H    XXXX HLHL    HLHL      SELECT TRUE INPUT  B=05
  H    H    XXXX LHLH    LHLH      SELECT TRUE INPUT  B=10
  H    H    XXXX HHHH    HHHH      SELECT TRUE INPUT  B=15
------------------------------------------------------------
```

TB01180M

10

DESCRIPTION

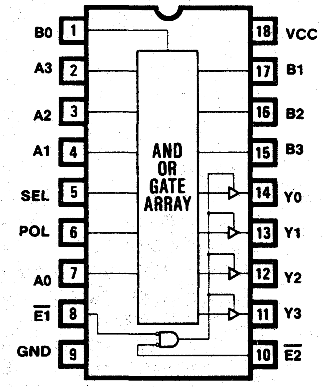
THIS IS AN EXAMPLE OF A QUAD 2:1 MULTIPLEXER WITH POLARITY CONTROL IMPLEMENTED IN A PLE10P4. THE DEVICE SELECTS BETWEEN TWO 4-BIT INPUTS (A1-A4 AND B1-B4) WHICH ARE DIRECTED TO ONE 4-BIT OUTPUT (Y1-Y4) AS DETERMINED BY ONE INPUT SELECT LINE (SEL) AND POLARITY CONTROL (POL). WHEN POLARITY IS TRUE (POL=HIGH), THE TRUE OF THE INPUT SIGNAL IS SELECTED. WHEN POLARITY IS FALSE (POL=LOW), THE COMPLEMENT OF THE INPUT SIGNAL IS SELECTED.

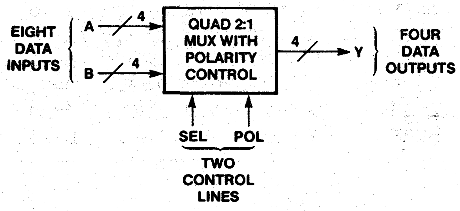
THE PLE10P4 ALSO FEATURES THREE-STATE OUTPUTS WITH TWO ACTIVE LOW ENABLE PINS (/E1 AND /E2). THE FUNCTION IS SUMMARIZED BELOW:

SEL	POL	A1-A4	B1-B4	Y1-Y4
L	H	A	X	A
L	L	A	X	/A
H	H	X	B	B
H	L	X	B	/B

QUAD 2:1 MULTIPLEXER
WITH POLARITY CONTROL

PLE10P4





TB01190M

```
PLE5P8                                PLE CIRCUIT DESIGN SPECIFICATION
P5006                                                 ULRIK MUELLER 04/29/84
HEXADECIMAL TO SEVEN SEGMENT DECODER
MMI SANTA CLARA, CALIFORNIA
.ADD A B C D LT
.DAT /OA /OB /OC /OD /OE /OF /OG /DP

OA =     B*   /D      ; SEGMENT A
   +     B* C
   + /A*   /C*/D
   +  A*    C*/D
   + /A*       D
   +    /B*/C* D
   +             LT   ; IF LT=H MAKE BLANK TEST ON SEGMENT A

OB =       /C*/D      ; SEGMENT B
   +  A* B*   /D
   + /A*/B*   /D
   +  A*/B*    D
   + /A*   /C
   +             LT   ; IF LT=H MAKE BLANK TEST ON SEGMENT B

OC =       /C* D      ; SEGMENT C
   +  A*/B
   +        C*/D
   +  A*      /D
   +    /B*   /D
   +             LT   ; IF LT=H MAKE BLANK TEST ON SEGMENT C

OD = /A*/B*/C         ; SEGMENT D
   +    /B*    D
   +  A*/B* C
   +  A* B*/C
   + /A* B* C
   + /A* B*   /D
   +             LT   ; IF LT=H MAKE BLANK TEST ON SEGMENT D

OE = /A*   /C         ; SEGMENT E
   +        C* D
   + /A* B
   +  A* B*    D
   +             LT   ; IF LT=H MAKE BLANK TEST ON SEGMENT E

OF = /A*/B            ; SEGMENT F
   +    /B* C*/D
   +       /C* D
   +     B*    D
   + /A* B* C
   +             LT   ; IF LT=H MAKE BLANK TEST ON SEGMENT F

OG =     B*/C         ; SEGMENT G
   + /A* B
   +       /C* D
   +  A*       D
   +    /B* C*/D
   +             LT   ; IF LT=H MAKE BLANK TEST ON SEGMENT G

DP = LT               ; TURNS DP ON ONLY WHEN LT=H
```

TB01200M

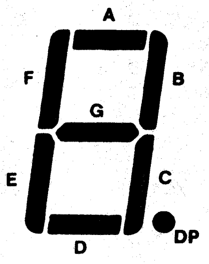
DESCRIPTION

THIS EXAMPLE ILLUSTRATES THE USE OF A PLE5P8 AS A HEXADECIMAL TO SEVEN SEGMENT DECODER. THE DEVICE DECODES A 4-BIT BINARY INPUT (D,C,B,A) INTO THE SEVEN SEGMENT OUTPUTS NEEDED TO DRIVE AN LED DISPLAY. NOTE THAT THIS DESIGN IS AN IMPROVEMENT FROM THE 74LS47 SINCE ALL 16 HEXADECIMAL DIGITS (0-F) CAN BE DISPLAYED. A LAMP TEST IS PROVIDED TO ILLUMINATE ALL SEVEN SEGMENTS AND THE DECIMAL POINT (IF DP IS CONNECTED) BY BRINGING LAMP TEST HIGH (LT=HIGH) REGARDLESS OF THE OTHER BINARY INPUTS. THREE-STATE OUTPUTS ARE ALSO PROVIDED WITH ONE ACTIVE LOW ENABLE PIN (/E).

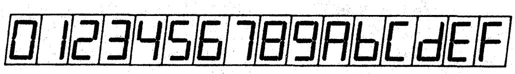
INPUT DIGIT	LT	INPUT D C B A	SEGMENT ON	OUTPUT DISPLAY
0	L	L L L L	ABCDEF	0
1	L	L L L H	BC	1
2	L	L L H L	ABDEG	2
3	L	L L H H	ABCDG	3
4	L	L H L L	BCDFG	4
5	L	L H L H	ACDFG	5
6	L	L H H L	ACDEFG	6
7	L	L H H H	ABC	7
8	L	H L L L	ABCDEFG	8
9	L	H L L H	ABCFG	9
A	L	H L H L	ABCEFG	A
B	L	H L H H	CDEFG	b
C	L	H H L L	ADEF	C
D	L	H H L H	BCDEG	d
E	L	H H H L	ADEFG	E
F	L	H H H H	AEFG	F
X	H	X X X X	ABCDEFG	8 *

* BLANK TEST OF DISPLAY

SEGMENT IDENTIFICATION

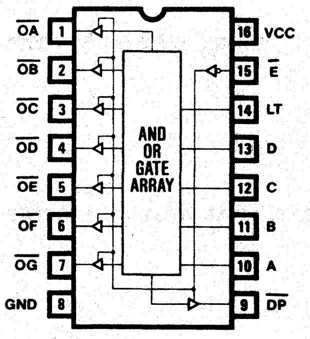


CHARACTER SET



HEXADECIMAL TO SEVEN-SEGMENT DECODER

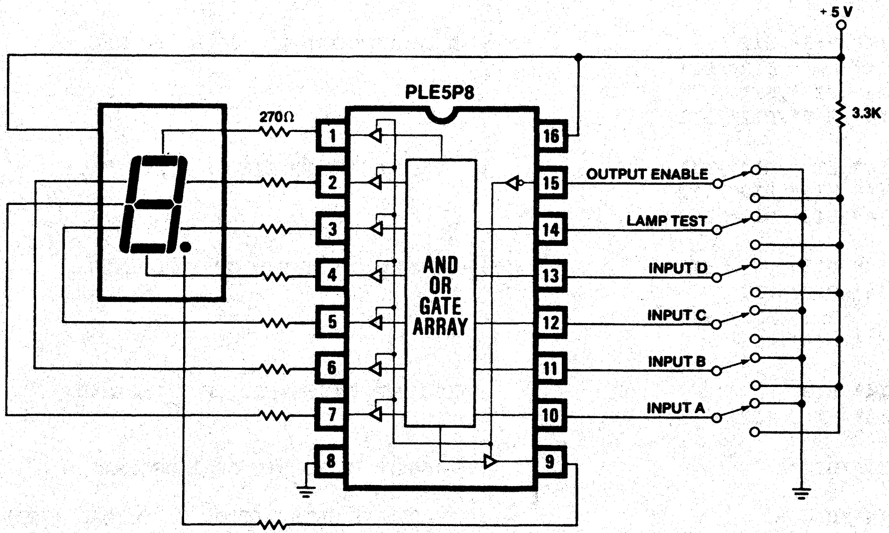
PLE5P8



TB01210M

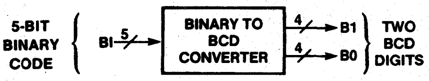
HEXADECIMAL TO SEVEN-SEGMENT DECODER (cont'd)

HEXADECIMAL TO SEVEN-SEGMENT DECODER



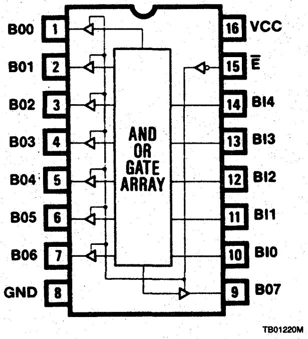
TC00010M

```
PLE5P8                                 PLE CIRCUIT DESIGN SPECIFICATION
P5007                                              VINCENT COLI 02/03/82
5-BIT BINARY TO BCD CONVERTER
MMI SANTA CLARA, CALIFORNIA
.ADD BI0 BI1 BI2 BI3 BI4
.DAT B00 B01 B02 B03 B10 B11 B12 B13


B00 =  BI0                                ; CONVERT FIRST BIT OF 0 DECIMAL (LSB)

B01 = /BI4*/BI3*      BI1                 ; CONVERT SECOND BIT OF 0 DECIMAL
    + /BI4* BI3* BI2*/BI1
    +  BI4* BI3*/BI2* BI1
    +  BI4*/BI3*/BI2*/BI1
    +      /BI3* BI2* BI1

B02 = /BI4*/BI3* BI2                      ; CONVERT THIRD BIT OF 0 DECIMAL
    + /BI4*      BI2* BI1
    +  BI4* BI3*/BI2
    +  BI4*/BI3*/BI2*/BI1

B03 = /BI4* BI3*/BI2*/BI1                 ; CONVERT FOURTH BIT OF 0 DECIMAL
    +  BI4* BI3* BI2*/BI1
    +  BI4*/BI3*/BI2* BI1

B10 = /BI4* BI3*      BI1                 ; CONVERT FIRST BIT OF 1 DECIMAL
    + /BI4* BI3* BI2
    +       BI3* BI2* BI1
    +  BI4*/BI3*/BI2

B11 =  BI4* BI3                           ; CONVERT SECOND BIT OF 1 DECIMAL
    +  BI4*      BI2

B12 =  BI4*/BI4                           ; CONVERT THIRD BIT OF 1 DECIMAL

B13 =  BI4*/BI4                           ; CONVERT FOURTH BIT OF 1 DECIMAL (MSB)
```



5-BIT BINARY
TO BCD
CONVERTER

PLE5P8



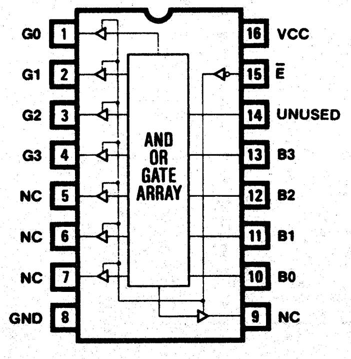
```
FUNCTION TABLE

BI4 BI3 BI2 BI1 BI0 B13 B12 B11 B10 B03 B02 B01 B00

;ADDRESS         ----DATA----
;BINARY          BCD 1   BCD 0       DESCRIPTION
;43 210          3210    3210      (DECIMAL VALUE)
-----------------------------------------------
 LL LLL          LLLL    LLLL                0
 LL LLH          LLLL    LLLH                1
 LL LHH          LLLL    LLHH                3
 LL LHL          LLLL    LLHL                2
 LL HHL          LLLL    LHHL                6
 LL HHH          LLLL    LHHH                7
 LL HLH          LLLL    LHLH                5
 LL HLL          LLLL    LHLL                4
;
 LH LLL          LLLL    HLLL                8
 LH LLH          LLLL    HLLH                9
 LH LHH          LLLH    LLLH              1 1
 LH LHL          LLLH    LLLL              1 0
 LH HHL          LLLH    LHLL              1 4
 LH HHH          LLLH    LHLH              1 5
 LH HLH          LLLH    LLHH              1 3
 LH HLL          LLLH    LLHL              1 2
;
 HL LLL          LLLH    LHHL              1 6
 HL LLH          LLLH    LHHH              1 7
 HL LHH          LLLH    HLLH              1 9
 HL LHL          LLLH    HLLL              1 8
 HL HHL          LLHL    LLHL              2 2
 HL HHH          LLHL    LLHH              2 3
 HL HLH          LLHL    LLLH              2 1
 HL HLL          LLHL    LLLL              2 0
;
 HH LLL          LLHL    LHLL              2 4
 HH LLH          LLHL    LHLH              2 5
 HH LHH          LLHL    LHHH              2 7
 HH LHL          LLHL    LHHL              2 6
 HH HHL          LLHH    LLLL              3 0
 HH HHH          LLHH    LLLH              3 1
 HH HLH          LLHL    HLLH              2 9
 HH HLL          LLHL    HLLL              2 8
-----------------------------------------------

DESCRIPTION

 THIS 5-BIT BINARY TO 2-DIGIT BCD CONVERTER IS IMPLEMENTED IN A PLE5P8
 LOGIC CIRCUIT.  THE DEVICE ACCEPTS A 5-BIT BINARY INPUT (BI) AND CONVERTS
 THIS INTO TWO 4-BIT BINARY CODED DECIMAL (BCD) OUTPUTS (B1 AND B0).

THREE-STATE OUTPUTS ARE ALSO PROVIDED WITH ONE ACTIVE LOW ENABLE PIN (/E).
```

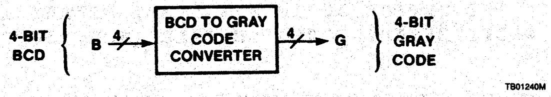
TB01230M

```
PLE5P8                         PLE CIRCUIT DESIGN SPECIFICATION
P5008                                         VINCENT COLI 10/16/81
4-BIT BCD TO GRAY CODE CONVERTER
MMI SANTA CLARA, CALIFORNIA
.ADD B0 B1 B2 B3
.DAT G0 G1 G2 G3


G0 = B0 :+: B1                           ; CONVERT G0 (LSB)

G1 = B1 :+: B2                           ; CONVERT G1

G2 = B2 :+: B3                           ; CONVERT G2

G3 = B3                                  ; CONVERT G3 (MSB)


DESCRIPTION

THIS PLE5P8 WILL CONVERT A 4-BIT BCD INPUT (B3-B0) INTO A 4-BIT
GRAY CODE REPRESENTATION (G3-G0) FOR OUTPUT.
```

4-BIT BCD TO GRAY CODE CONVERTER

PLE5P8

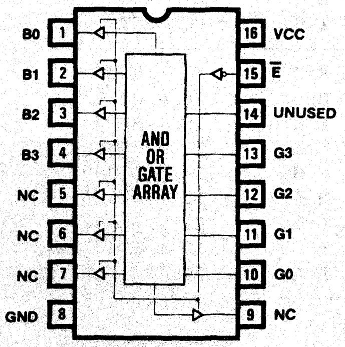


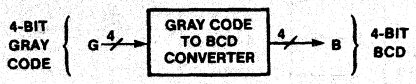


```
PLE5P8                        PLE CIRCUIT DESIGN SPECIFICATION
P5009                                      VINCENT COLI 03/16/84
4-BIT GRAY CODE TO BCD CONVERTER
MMI SANTA CLARA, CALIFORNIA
.ADD G0 G1 G2 G3
.DAT B0 B1 B2 B3


B0 = G0 :+: G1 :+: G2 :+: G3            ; CONVERT B0 (LSB)

B1 = G1 :+: G2 :+: G3                   ; CONVERT B1

B2 = G2 :+: G3                          ; CONVERT B2

B3 = G3                                 ; CONVERT B3 (MSB)


DESCRIPTION

THIS PLE5P8 WILL CONVERT A 4-BIT GRAY CODE INPUT (G3-G0) INTO A 4-BIT
BINARY REPRESENTATION (B3-B0) FOR OUTPUT.
```

4-BIT GRAY CODE TO BCD CONVERTER

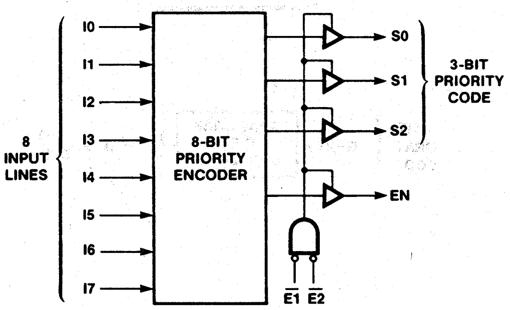
PLE5P8





TB01250M

```
PLE8P4                          PLE CIRCUIT DESIGN SPECIFICATION
P5010                                FRANK LEE/ULRIK MUELLER 05/14/84
8-BIT PRIORITY ENCODER
MMI SANTA CLARA, CALIFORNIA
.ADD I0 I1 I2 I3 I4 I5 I6 I7
.DAT S0 S1 S2 EN


S0 =  I7                                ; I7-I0 = 1XXXXXXX
   + /I6* I5                            ; I7-I0 = X01XXXXX
   + /I6*/I4* I3                        ; I7-I0 = X0X01XXX
   + /I6*/I4*/I2* I1                    ; I7-I0 = X0X0X01X

S1 =  I7                                ; I7-I0 = 1XXXXXXX
   +  I6                                ; I7-I0 = X1XXXXXX
   + /I5*/I4* I3                        ; I7-I0 = XX001XXX
   + /I5*/I4* I2                        ; I7-I0 = XX00X1XX

S2 =  I7                                ; I7-I0 = 1XXXXXXX
   +  I6                                ; I7-I0 = X1XXXXXX
   +  I5                                ; I7-I0 = XX1XXXXX
   +  I4                                ; I7-I0 = XXX1XXXX

EN = /I0*/I1*/I2*/I3*/I4*/I5*/I6*/I7    ; ALL LOWS ENABLE NEXT PRIORITY ENCODER


FUNCTION TABLE

I7 I6 I5 I4 I3 I2 I1 I0 EN S2 S1 S0

; --INPUT LINES--     -OUTPUTS-
; I I I I I I I I     E   S S S
; 7 6 5 4 3 2 1 0     N   2 1 0      COMMENT
-----------------------------------------------------------------
  H X X X X X X X     L   H H H      I7 = HIGH
  L H X X X X X X     L   H H L      I6 = HIGH
  L L H X X X X X     L   H L H      I5 = HIGH
  L L L H X X X X     L   H L L      I4 = HIGH
  L L L L H X X X     L   L H H      I3 = HIGH
  L L L L L H X X     L   L H L      I2 = HIGH
  L L L L L L H X     L   L L H      I1 = HIGH
  L L L L L L L H     L   L L L      I0 = HIGH
  L L L L L L L L     H   L L L      I7-I0 = LOW THEN CARRY OUT
-----------------------------------------------------------------
```



TB01260M

DESCRIPTION

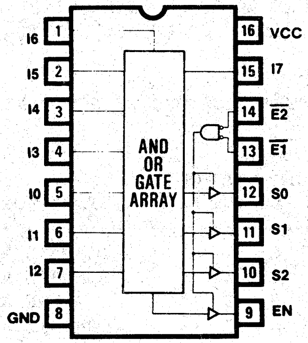
THIS 8-BIT PRIORITY ENCODER SCANS FOR THE FIRST HIGH INPUT LINE (I7-I0) FROM I7 (WHICH HAS THE HIGHEST PRIORITY) TO I0 (WHICH HAS THE LOWEST PRIORITY). IT WILL GENERATE A BINARY ENCODED OUTPUT (S2-S0) WHICH WILL POINT TO THE HIGHEST PRIORITY INPUT WHICH IS AT A HIGH STATE.

IF NO INPUT LINES ARE HIGH (I7-I0=LOW), THEN THE BINARY ENCODED OUTPUTS WILL BE ZERO (S2-S0=LOW) AND THE ENABLE OUTPUT WILL BE HIGH (EN=HIGH) INDICATING A CARRY OUT TO THE NEXT PRIORITY ENCODER. THE OUTPUT ENABLE WILL BE LOW (EN=LOW) IF ANY OF THE INPUT LINES ARE HIGH.

THE PLE8P4 ALSO HAS THREE-STATE OUTPUTS WITH TWO ACTIVE-LOW OUTPUT ENABLE CONTROL PINS (/E1 AND /E2).

8-BIT PRIORITY ENCODER

PLE8P4

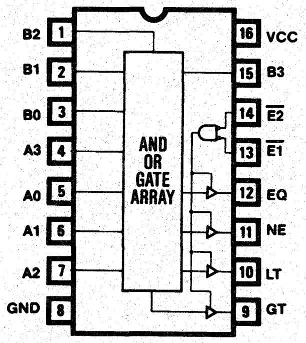


TB01270M

```
PLE8P4                                 PLE CIRCUIT DESIGN SPECIFICATION
P5011                                              ULRIK MUELLER 04/01/83
4-BIT MAGNITUDE COMPARATOR
MMI SANTA CLARA, CALIFORNIA
.ADD A0 A1 A2 A3 B0 B1 B2 B3
.DAT EQ NE LT GT

EQ =  A3:*:B3 *  A2:*:B2 *  A1:*:B1 *  A0:*:B0 ; A = B

NE =  A3:+:B3 +  A2:+:B2 +  A1:+:B1 +  A0:+:B0 ; A NOT = B

LT = /A3 * B3                                  ; A3 LT B3
   +  A3:*:B3 * /A2 * B2                       ; A2 LT B2
   +  A3:*:B3 *  A2:*:B2 * /A1 * B1            ; A1 LT B1
   +  A3:*:B3 *  A2:*:B2 *  A1:*:B1 * /A0 * B0 ; A0 LT B0

GT =  A3 */B3                                  ; A3 GT B3
   +  A3:*:B3 *  A2 */B2                       ; A2 GT B2
   +  A3:*:B3 *  A2:*:B2 *  A1 */B1            ; A1 GT B1
   +  A3:*:B3 *  A2:*:B2 *  A1:*:B1 *  A0 */B0 ; A0 GT B0

DESCRIPTION

THIS PLE8P4 COMPARES TWO 4-BIT NUMBERS (A3-A0 AND B3-B0) TO ESTABLISH IF THEY
ARE EQUAL (A = B  THEN  EQ=H), NOT EQUAL (A NOT = B  THEN NE=H), LESS THAN (A
LT B  THEN  LT=H), OR GREATER THAN (A GT B  THEN  GT=H) AND REPORTS THE
COMPARISON STATUS ON THE OUTPUTS (EQ, NE, LT, GT) AS ILLUSTRATED IN THE
OPERATIONS TABLE BELOW.

THE PLE8P4 ALSO FEATURES THREE-STATE OUTPUTS WITH TWO ACTIVE-LOW OUTPUT ENABLE
CONTROL PINS (/E1 AND /E2).

        INPUT NUMBERS    COMPARISON  STATUS
        A3-A0    B3-B0     EQ  NE  LT  GT          OPERATION
      ------------------------------------------------------------------
          A   =   B        H   L   L   L           COMPARE A EQUAL TO B
          A NOT = B        L   H   X   X           COMPARE A NOT EQUAL TO B
          A  LT   B        L   H   H   L           COMPARE A LESS THAN B
          A  GT   B        L   H   L   H           COMPARE A GREATER THAN B
      ------------------------------------------------------------------
```

4-BIT MAGNITUDE COMPARATOR

PLE8P4



TB01280M

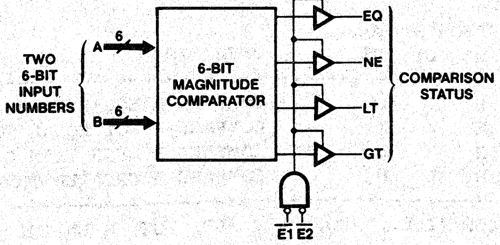
```
PLE12P4                                      PLE CIRCUIT DESIGN SPECIFICATION
P5012                                                   VINCENT COLI 10/16/83
6-BIT MAGNITUDE COMPARATOR
MMI SANTA CLARA, CALIFORNIA
.ADD A0 A1 A2 A3 A4 A5 B0 B1 B2 B3 B4 B5
.DAT EQ NE LT GT

EQ =  A5:*:B5 *  A4:*:B4 *  A3:*:B3 *  A2:*:B2 *  A1:*:B1 *  A0:*:B0 ; A = B

NE =  A5:+:B5 +  A4:+:B4 +  A3:+:B3 +  A2:+:B2 +  A1:+:B1 +  A0:+:B0 ; A NOT= B

LT = /A5 * B5                                                         ; A5 LT B5
   +  A5:*:B5 * /A4 * B4                                              ; A4 LT B4
   +  A5:*:B5 *  A4:*:B4 * /A3 * B3                                   ; A3 LT B3
   +  A5:*:B5 *  A4:*:B4 *  A3:*:B3 * /A2 * B2                        ; A2 LT B2
   +  A5:*:B5 *  A4:*:B4 *  A3:*:B3 *  A2:*:B2 * /A1 * B1             ; A1 LT B1
   +  A5:*:B5 *  A4:*:B4 *  A3:*:B3 *  A2:*:B2 *  A1:*:B1 * /A0 * B0 ; A0 LT B0

GT =  A5 */B5                                                         ; A5 GT B5
   +  A5:*:B5 *  A4 */B4                                              ; A4 GT B4
   +  A5:*:B5 *  A4:*:B4 *  A3 */B3                                   ; A3 GT B3
   +  A5:*:B5 *  A4:*:B4 *  A3:*:B3 *  A2 */B2                        ; A2 GT B2
   +  A5:*:B5 *  A4:*:B4 *  A3:*:B3 *  A2:*:B2 *  A1 */B1             ; A1 GT B1
   +  A5:*:B5 *  A4:*:B4 *  A3:*:B3 *  A2:*:B2 *  A1:*:B1 *  A0 */B0 ; A0 GT B0
```

DESCRIPTION

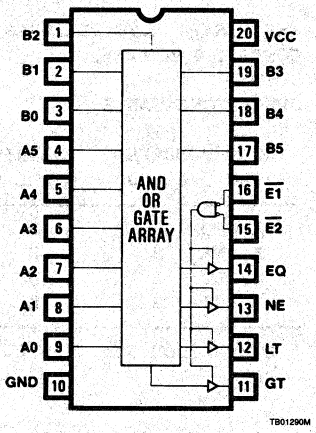
THIS PLE12P4 COMPARES TWO 6-BIT NUMBERS (A5-A0 AND B5-B0) TO ESTABLISH IF THEY ARE EQUAL (A = B THEN EQ=H), NOT EQUAL (A NOT = B THEN NE=H), LESS THAN (A LT B THEN LT=H), OR GREATER THAN (A GT B THEN GT=H) AND REPORTS THE COMPARISON STATUS ON THE OUTPUTS (EQ, NE, LT, GT) AS ILLUSTRATED IN THE OPERATIONS TABLE BELOW.

THE PLE12P4 ALSO FEATURES THREE-STATE OUTPUTS WITH TWO ACTIVE-LOW OUTPUT ENABLE CONTROL PINS (/E1 AND /E2).

INPUT NUMBERS			COMPARISON STATUS				
A5-A0		B5-B0	EQ	NE	LT	GT	OPERATION
A	=	B	H	L	L	L	COMPARE A EQUAL TO B
A	NOT =	B	L	H	X	X	COMPARE A NOT EQUAL TO B
A	LT	B	L	H	H	L	COMPARE A LESS THAN B
A	GT	B	L	H	L	H	COMPARE A GREATER THAN B



6-BIT MAGNITUDE COMPARATOR
PLE12P4



```
PLE9P4                                   PLE CIRCUIT DESIGN SPECIFICATION
P5011A                                             COLI/MUELLER 09/09/84
4-BIT MAGNITUDE COMPARATOR WITH POLARITY CONTROL
MMI SANTA CLARA, CALIFORNIA
.ADD A0 A1 A2 A3 B0 B1 B2 B3 POL
.DAT EQ NE LT GT

EQ =  A3:*:B3* POL *  A2:*:B2* POL *  A1:*:B1* POL *  A0:*:B0* POL  ; A  EQ B
   +  A3:+:B3*/POL *  A2:+:B2*/POL *  A1:+:B1*/POL *  A0:+:B0*/POL  ; A /EQ B

NE =  A3:+:B3* POL +  A2:+:B2* POL +  A1:+:B1* POL +  A0:+:B0* POL  ; A  NE B
   +  A3:*:B3*/POL +  A2:*:B2*/POL +  A1:*:B1*/POL +  A0:*:B0*/POL  ; A /NE B

LT = /A3 * B3* POL                                                  ; A3  LT B3
   +  A3:*:B3* POL * /A2 * B2* POL                                  ; A2  LT B2
   +  A3:*:B3* POL *  A2:*:B2* POL * /A1 * B1* POL                  ; A1  LT B1
   +  A3:*:B3* POL *  A2:*:B2* POL *  A1:*:B1* POL * /A0 * B0* POL  ; A0  LT B0
   +  A3 */B3*/POL                                                  ; A3 /LT B3
   +  A3:*:B3*/POL *  A2 */B2*/POL                                  ; A2 /LT B2
   +  A3:*:B3*/POL *  A2:*:B2*/POL *  A1 */B1*/POL                  ; A1 /LT B1
   +  A3:*:B3*/POL *  A2:*:B2*/POL *  A1:*:B1*/POL *  A0 */B0*/POL  ; A0 /LT B0
   +  A3:*:B3*/POL +  A2:*:B2*/POL +  A1:*:B1*/POL +  A0:*:B0*/POL  ; A  /LT B

GT =  A3 */B3* POL                                                  ; A3  GT B3
   +  A3:*:B3* POL *  A2 */B2* POL                                  ; A2  GT B2
   +  A3:*:B3* POL *  A2:*:B2* POL *  A1 */B1* POL                  ; A1  GT B1
   +  A3:*:B3* POL *  A2:*:B2* POL *  A1:*:B1* POL *  A0 */B0* POL  ; A0  GT B0
   + /A3 * B3*/POL                                                  ; A3 /GT B3
   +  A3:*:B3*/POL * /A2 * B2*/POL                                  ; A2 /GT B2
   +  A3:*:B3*/POL *  A2:*:B2*/POL * /A1 * B1*/POL                  ; A1 /GT B1
   +  A3:*:B3*/POL *  A2:*:B2*/POL *  A1:*:B1*/POL * /A0 * B0*/POL  ; A0 /GT B0
   +  A3:*:B3*/POL +  A2:*:B2*/POL +  A1:*:B1*/POL +  A0:*:B0*/POL  ; A  /GT B
```

DESCRIPTION

THIS PLE9P4 COMPARES TWO 4-BIT NUMBERS (A3-A0 AND B3-B0) TO ESTABLISH IF THEY ARE EQUAL (A EQ B), NOT EQUAL (A NE B), LESS THAN (A LT B), OR GREATER THAN (A GT B). THE COMPARISON STATUS IS REPORTED WITH ACTIVE-HIGH POLARITY (EQ, NE, LT, GT) WHEN THE POLARITY CONTROL INPUT IS TRUE (POL=H) AND WITH ACTIVE-LOW POLARITY (/EQ, /NE, /LT, /GT) WHEN THE POLARITY CONTROL INPUT IS FALSE (POL=L).

THE PLE8P4 ALSO FEATURES THREE-STATE OUTPUTS WITH ONE ACTIVE-LOW OUTPUT ENABLE CONTROL PIN (/E).

OPERATIONS TABLE:

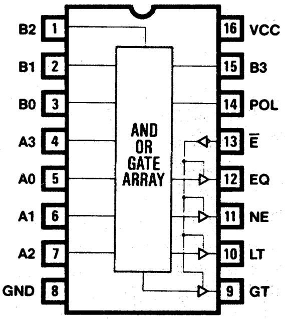
INPUT NUMBERS			POLARITY	COMPARISON STATUS				
A3-A0		B3-B0	POL *	EQ	NE	LT	GT	OPERATION
A	EQ	B	H	H	L	L	L	COMPARE A EQUAL TO B
A	NE	B	H	L	H	X	X	COMPARE A NOT EQUAL TO B
A	LT	B	H	L	H	H	L	COMPARE A LESS THAN B
A	GT	B	H	L	H	L	H	COMPARE A GREATER THAN B

* COMPARISON STATUS WILL BE ACTIVE-LOW (I.E., /EQ, /NE, /LT, /GT) WHEN POL=L.

TB01300M

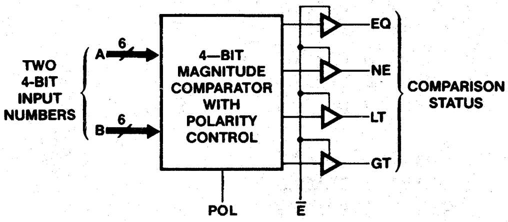
4-Bit Magnitude Comparator with Polarity Control

PLE9P4



CD00900M

Block Diagram



BD00060M

```
PLE11P8                                    PLE CIRCUIT DESIGN SPECIFICATION
P5013                                               VINCENT COLI 06/12/84
8-BIT BARREL SHIFTER
MMI SANTA CLARA, CALIFORNIA
.ADD D0 D1 D2 D3 D4 D5 D6 D7 S0 S1 S2
.DAT O0 O1 O2 O3 O4 O5 O6 O7


O0 = /S0*/S1*/S2* D0       ; SHIFT 0 PLACES
   +  S0*/S1*/S2* D1       ; SHIFT 1 PLACES
   + /S0* S1*/S2* D2       ; SHIFT 2 PLACES
   +  S0* S1*/S2* D3       ; SHIFT 3 PLACES
   + /S0*/S1* S2* D4       ; SHIFT 4 PLACES
   +  S0*/S1* S2* D5       ; SHIFT 5 PLACES
   + /S0* S1* S2* D6       ; SHIFT 6 PLACES
   +  S0* S1* S2* D7       ; SHIFT 7 PLACES

O1 = /S0*/S1*/S2* D1       ; SHIFT 0 PLACES
   +  S0*/S1*/S2* D2       ; SHIFT 1 PLACES
   + /S0* S1*/S2* D3       ; SHIFT 2 PLACES
   +  S0* S1*/S2* D4       ; SHIFT 3 PLACES
   + /S0*/S1* S2* D5       ; SHIFT 4 PLACES
   +  S0*/S1* S2* D6       ; SHIFT 5 PLACES
   + /S0* S1* S2* D7       ; SHIFT 6 PLACES
   +  S0* S1* S2* D0       ; SHIFT 7 PLACES

O2 = /S0*/S1*/S2* D2       ; SHIFT 0 PLACES
   +  S0*/S1*/S2* D3       ; SHIFT 1 PLACES
   + /S0* S1*/S2* D4       ; SHIFT 2 PLACES
   +  S0* S1*/S2* D5       ; SHIFT 3 PLACES
   + /S0*/S1* S2* D6       ; SHIFT 4 PLACES
   +  S0*/S1* S2* D7       ; SHIFT 5 PLACES
   + /S0* S1* S2* D0       ; SHIFT 6 PLACES
   +  S0* S1* S2* D1       ; SHIFT 7 PLACES

O3 = /S0*/S1*/S2* D3       ; SHIFT 0 PLACES
   +  S0*/S1*/S2* D4       ; SHIFT 1 PLACES
   + /S0* S1*/S2* D5       ; SHIFT 2 PLACES
   +  S0* S1*/S2* D6       ; SHIFT 3 PLACES
   + /S0*/S1* S2* D7       ; SHIFT 4 PLACES
   +  S0*/S1* S2* D0       ; SHIFT 5 PLACES
   + /S0* S1* S2* D1       ; SHIFT 6 PLACES
   +  S0* S1* S2* D2       ; SHIFT 7 PLACES

O4 = /S0*/S1*/S2* D4       ; SHIFT 0 PLACES
   +  S0*/S1*/S2* D5       ; SHIFT 1 PLACES
   + /S0* S1*/S2* D6       ; SHIFT 2 PLACES
   +  S0* S1*/S2* D7       ; SHIFT 3 PLACES
   + /S0*/S1* S2* D0       ; SHIFT 4 PLACES
   +  S0*/S1* S2* D1       ; SHIFT 5 PLACES
   + /S0* S1* S2* D2       ; SHIFT 6 PLACES
   +  S0* S1* S2* D3       ; SHIFT 7 PLACES
```

TB01310M

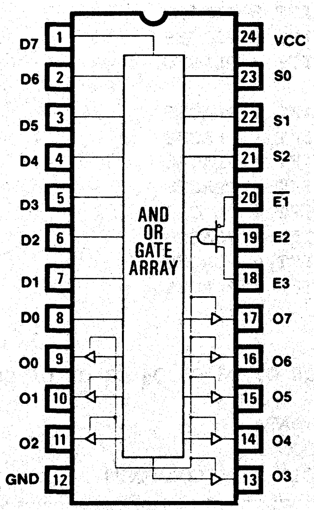
DESCRIPTION

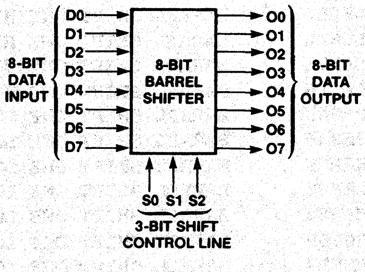
THE 8-BIT BARREL SHIFTER, IMPLEMENTED IN A PLE11P8, ROTATES EIGHT BITS OF DATA (D7-D0) A NUMBER OF LOCATIONS INTO THE OUTPUTS (O7-O0) AS SPECIFIED BY THE 3-BIT BINARY ENCODED SHIFT CONTROL LINE (S2-S0). THE THREE-STATE OUTPUTS ARE IN A HIGH-Z STATE WHEN ANY ONE OF THE TWO OUTPUT ENABLE PINS (/E1 OR /E1) ARE HIGH.

A POSSIBLE UPGRADE VERSION OF THIS DESIGN IMPLEMENTED IN A PLE12P8 COULD INCLUDE A DIRECTION CONTROL LINE. THIS CONTROL LINE PERMITS THE 8-BIT BARREL SHIFTER TO ROTATE DATA IN EITHER DIRECTION (LEFT OR RIGHT).

8-BIT BARREL SHIFTER

PLE11P8





TB01320M

```
O5 = /S0*/S1*/S2* D5      ; SHIFT 0 PLACES
   +  S0*/S1*/S2* D6      ; SHIFT 1 PLACES
   + /S0* S1*/S2* D7      ; SHIFT 2 PLACES
   +  S0* S1*/S2* D0      ; SHIFT 3 PLACES
   + /S0*/S1* S2* D1      ; SHIFT 4 PLACES
   +  S0*/S1* S2* D2      ; SHIFT 5 PLACES
   + /S0* S1* S2* D3      ; SHIFT 6 PLACES
   +  S0* S1* S2* D4      ; SHIFT 7 PLACES

O6 = /S0*/S1*/S2* D6      ; SHIFT 0 PLACES
   +  S0*/S1*/S2* D7      ; SHIFT 1 PLACES
   + /S0* S1*/S2* D0      ; SHIFT 2 PLACES
   +  S0* S1*/S2* D1      ; SHIFT 3 PLACES
   + /S0*/S1* S2* D2      ; SHIFT 4 PLACES
   +  S0*/S1* S2* D3      ; SHIFT 5 PLACES
   + /S0* S1* S2* D4      ; SHIFT 6 PLACES
   +  S0* S1* S2* D5      ; SHIFT 7 PLACES

O7 = /S0*/S1*/S2* D7      ; SHIFT 0 PLACES
   +  S0*/S1*/S2* D0      ; SHIFT 1 PLACES
   + /S0* S1*/S2* D1      ; SHIFT 2 PLACES
   +  S0* S1*/S2* D2      ; SHIFT 3 PLACES
   + /S0*/S1* S2* D3      ; SHIFT 4 PLACES
   +  S0*/S1* S2* D4      ; SHIFT 5 PLACES
   + /S0* S1* S2* D5      ; SHIFT 6 PLACES
   +  S0* S1* S2* D6      ; SHIFT 7 PLACES
FUNCTION TABLE

S2 S1 S0 D7 D6 D5 D4 D3 D2 D1 D0 O7 O6 O5 O4 O3 O2 O1 O0

;SHIFT    INPUT DATA    OUTPUT DATA
; SSS      DDDDDDDD      OOOOOOOO
; 210      76543210      76543210      COMMENTS
-----------------------------------------------------------------
  LLL      HLLLLLLL      HLLLLLLL      BARREL SHIFT ONE HIGH 0 PLACES
  LLH      HLLLLLLL      LHLLLLLL      BARREL SHIFT ONE HIGH 1 PLACES
  LHL      HLLLLLLL      LLHLLLLL      BARREL SHIFT ONE HIGH 2 PLACES
  LHH      HLLLLLLL      LLLHLLLL      BARREL SHIFT ONE HIGH 3 PLACES
  HLL      HLLLLLLL      LLLLHLLL      BARREL SHIFT ONE HIGH 4 PLACES
  HLH      HLLLLLLL      LLLLLHLL      BARREL SHIFT ONE HIGH 5 PLACES
  HHL      HLLLLLLL      LLLLLLHL      BARREL SHIFT ONE HIGH 6 PLACES
  HHH      HLLLLLLL      LLLLLLLH      BARREL SHIFT ONE HIGH 7 PLACES
  LLL      LHHHHHHH      LHHHHHHH      BARREL SHIFT ONE LOW  0 PLACES
  LLH      LHHHHHHH      HLHHHHHH      BARREL SHIFT ONE LOW  1 PLACES
  LHL      LHHHHHHH      HHLHHHHH      BARREL SHIFT ONE LOW  2 PLACES
  LHH      LHHHHHHH      HHHLHHHH      BARREL SHIFT ONE LOW  3 PLACES
  HLL      LHHHHHHH      HHHHLHHH      BARREL SHIFT ONE LOW  4 PLACES
  HLH      LHHHHHHH      HHHHHLHH      BARREL SHIFT ONE LOW  5 PLACES
  HHL      LHHHHHHH      HHHHHHLH      BARREL SHIFT ONE LOW  6 PLACES
  HHH      LHHHHHHH      HHHHHHHL      BARREL SHIFT ONE LOW  7 PLACES
-----------------------------------------------------------------
```

TB01330M

```
PLE11P4                                PLE CIRCUIT DESIGN SPECIFICATION
P5014                                                CHRIS JAY 05/30/84
4-BIT RIGHT SHIFTER WITH PROGRAMMABLE OUTPUT POLARITY
MMI LTD., FARNBOROUGH, U.K.
.ADD S0 S1 INV D0 D1 D2 D3 D4 D5 D6 /EN
.DAT O0 O1 O2 O3


O0 =  D0*/S0*/S1*/INV* EN       ; SELECT INPUT  D0
   + /D0*/S0*/S1* INV* EN       ; SELECT INPUT /D0
   +  D1* S0*/S1*/INV* EN       ; SELECT INPUT  D1
   + /D1* S0*/S1* INV* EN       ; SELECT INPUT /D1
   +  D2*/S0* S1*/INV* EN       ; SELECT INPUT  D2
   + /D2*/S0* S1* INV* EN       ; SELECT INPUT /D2
   +  D3* S0* S1*/INV* EN       ; SELECT INPUT  D3
   + /D3* S0* S1* INV* EN       ; SELECT INPUT /D3

O1 =  D1*/S0*/S1*/INV* EN       ; SELECT INPUT  D1
   + /D1*/S0*/S1* INV* EN       ; SELECT INPUT /D1
   +  D2* S0*/S1*/INV* EN       ; SELECT INPUT  D2
   + /D2* S0*/S1* INV* EN       ; SELECT INPUT /D2
   +  D3*/S0* S1*/INV* EN       ; SELECT INPUT  D3
   + /D3*/S0* S1* INV* EN       ; SELECT INPUT /D3
   +  D4* S0* S1*/INV* EN       ; SELECT INPUT  D4
   + /D4* S0* S1* INV* EN       ; SELECT INPUT /D4

O2 =  D2*/S0*/S1*/INV* EN       ; SELECT INPUT  D2
   + /D2*/S0*/S1* INV* EN       ; SELECT INPUT /D2
   +  D3* S0*/S1*/INV* EN       ; SELECT INPUT  D3
   + /D3* S0*/S1* INV* EN       ; SELECT INPUT /D3
   +  D4*/S0* S1*/INV* EN       ; SELECT INPUT  D4
   + /D4*/S0* S1* INV* EN       ; SELECT INPUT /D4
   +  D5* S0* S1*/INV* EN       ; SELECT INPUT  D5
   + /D5* S0* S1* INV* EN       ; SELECT INPUT /D5

O3 =  D3*/S0*/S1*/INV* EN       ; SELECT INPUT  D3
   + /D3*/S0*/S1* INV* EN       ; SELECT INPUT /D3
   +  D4* S0*/S1*/INV* EN       ; SELECT INPUT  D4
   + /D4* S0*/S1* INV* EN       ; SELECT INPUT /D4
   +  D5*/S0* S1*/INV* EN       ; SELECT INPUT  D5
   + /D5*/S0* S1* INV* EN       ; SELECT INPUT /D5
   +  D6* S0* S1*/INV* EN       ; SELECT INPUT  D6
   + /D6* S0* S1* INV* EN       ; SELECT INPUT /D6
```

TB01340M

```
FUNCTION TABLE

/EN S1 S0 INV D6 D5 D4 D3 D2 D1 D0 O3 O2 O1 O0

;-CONTROL-
;/       I    -INPUT  DATA-    OUTPUTS
;E  S S  N    D D D D D D D    O O O O
;N  1 0  V    6 5 4 3 2 1 0    3 2 1 0       COMMENTS
--------------------------------------------------------------------------
 H  X X  X    X X X X X X X    L L L L      TEST ENABLE, OUTPUTS GO LOW
 L  L L  L    L L L H H H H    H H H H      SHIFT COUNT = 0, TRUE POLARITY
 L  L H  L    L L L H H H H    L H H H      SHIFT COUNT = 1, TRUE POLARITY
 L  H L  L    L L L H H H H    L L H H      SHIFT COUNT = 2, TRUE POLARITY
 L  H H  L    L L L H H H H    L L L H      SHIFT COUNT = 3, TRUE POLARITY
 L  L L  H    L L L H H H H    L L L L      SHIFT COUNT = 0, COMP POLARITY
 L  L H  H    L L L H H H H    H L L L      SHIFT COUNT = 1, COMP POLARITY
 L  H L  H    L L L H H H H    H H L L      SHIFT COUNT = 2, COMP POLARITY
 L  H H  H    L L L H H H H    H H H L      SHIFT COUNT = 3, COMP POLARITY
--------------------------------------------------------------------------
```

TB01350M

DESCRIPTION

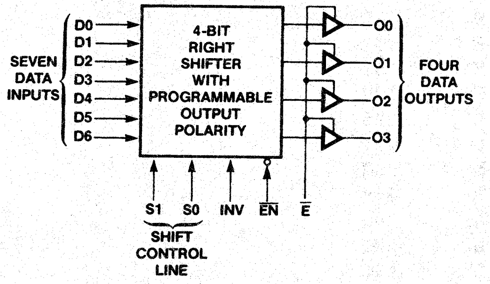
THIS PLE11P4 IMPLEMENTS A 4-BIT RIGHT SHIFTER WITH PROGRAMMABLE OUTPUT POLARITY. THE SHIFTER CAN RIGHT SHIFT SEVEN BITS OF DATA, FOUR BITS AT A TIME. THE SEVEN DATA INPUTS (D6-D0) ARE SHIFTED 0, 1, 2, OR 3 LOCATIONS AS DETERMINED BY THE 2-BIT SHIFT CONTROL LINE (S1-S0). THE SHIFTED DATA IS THEN DIRECTED TO THE FOUR OUTPUTS (O3-O0).

THE OUTPUT DATA IS NONINVERTED (O=D) WHEN INV=L AND INVERTED (O=/D) WHEN INV=H. THE OUTPUTS ARE FORCED LOW (O=L) WHEN /EN=H REGARDLESS OF OTHER INPUTS. THE PLE11P4 ALSO FEATURES THREE-STATE OUTPUTS WITH ONE ACTIVE LOW OUTPUT ENABLE (/E).

A POSSIBLE UPGRADE VERSION OF THIS DESIGN IMPLEMENTED IN A PLE12P4 COULD INCLUDE A DIRECTION CONTROL LINE. THIS CONTROL LINE PERMITS THE 4-BIT RIGHT SHIFTER TO SHIFT DATA IN EITHER DIRECTION (LEFT OR RIGHT).

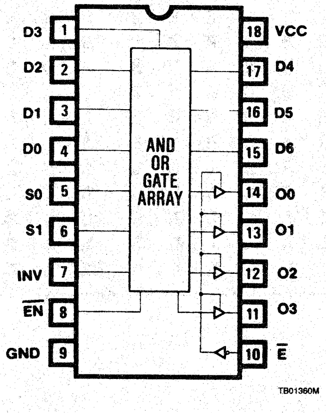
OPERATIONS TABLE:

/EN	INV	S1-S0	D6-D0	O3-O0	OPERATION
H	X	X	X	L	DISABLE OUTPUTS LOW
L	L	N	D	SHIFT(D)	SHIFT NONINVERTED DATA "N" PLACES
L	H	N	D	SHIFT(/D)	SHIFT INVERTED DATA "N" PLACES

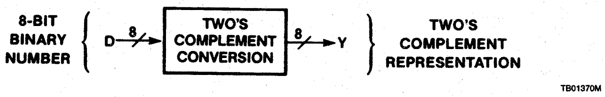


4-BIT RIGHT SHIFTER WITH PROGRAMMABLE OUTPUT POLARITY

PLE11P4



```
PLE8P8                                    PLE CIRCUIT DESIGN SPECIFICATION
P5015                                                MIKE VOGEL 11/28/83
8-BIT TWO'S COMPLEMENT CONVERSION
MMI BREA, CALIFORNIA
.ADD D0 D1 D2 D3 D4 D5 D6 D7
.DAT Y0 Y1 Y2 Y3 Y4 Y5 Y6 Y7


Y0 = D0                                        ; CONVERT 1ST BIT (LSB)

Y1 = D1 :+: D0                                 ; CONVERT 2ND BIT

Y2 = D2 :+: D0 + D1                            ; CONVERT 3RD BIT

Y3 = D3 :+: D0 + D1 + D2                       ; CONVERT 4TH BIT

Y4 = D4 :+: D0 + D1 + D2 + D3                  ; CONVERT 5TH BIT

Y5 = D5 :+: D0 + D1 + D2 + D3 + D4             ; CONVERT 6TH BIT

Y6 = D6 :+: D0 + D1 + D2 + D3 + D4 + D5        ; CONVERT 7TH BIT

Y7 = D7 :+: D0 + D1 + D2 + D3 + D4 + D5 + D6   ; CONVERT 8TH BIT (MSB)

FUNCTION TABLE

D7 D6 D5 D4 D3 D2 D1 D0  Y7 Y6 Y5 Y4 Y3 Y2 Y1 Y0   ;DECIMAL
-------------------------------------------------------------
 L  L  L  L  L  L  L  L   L  L  L  L  L  L  L  L         0
 L  L  L  L  L  L  L  H   H  H  H  H  H  H  H  H         1
 L  L  L  L  L  L  H  H   H  H  H  H  H  H  L  H         3
 L  L  L  L  L  H  H  H   H  H  H  H  H  L  L  H         7
 L  L  L  L  H  H  H  H   H  H  H  H  L  L  L  H        15
 L  L  L  H  H  H  H  H   H  H  H  L  L  L  L  H        31
 L  L  H  H  H  H  H  H   H  H  L  L  L  L  L  H        63
 L  H  H  H  H  H  H  H   H  L  L  L  L  L  L  H       127
 H  H  H  H  H  H  H  H   L  L  L  L  L  L  L  L       255
 H  H  H  H  H  H  H  L   L  L  L  L  L  L  H  L       254
 H  H  H  H  H  H  L  L   L  L  L  L  L  H  L  L       252
 H  H  H  H  H  L  L  L   L  L  L  L  H  L  L  L       248
 H  H  H  H  L  L  L  L   L  L  L  H  L  L  L  L       240
 H  H  H  L  L  L  L  L   L  L  H  L  L  L  L  L       224
 H  H  L  L  L  L  L  L   L  H  L  L  L  L  L  L       192
 H  L  L  L  L  L  L  L   H  L  L  L  L  L  L  L       128
-------------------------------------------------------------
```

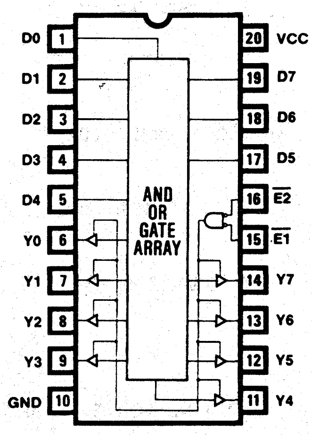


DESCRIPTION

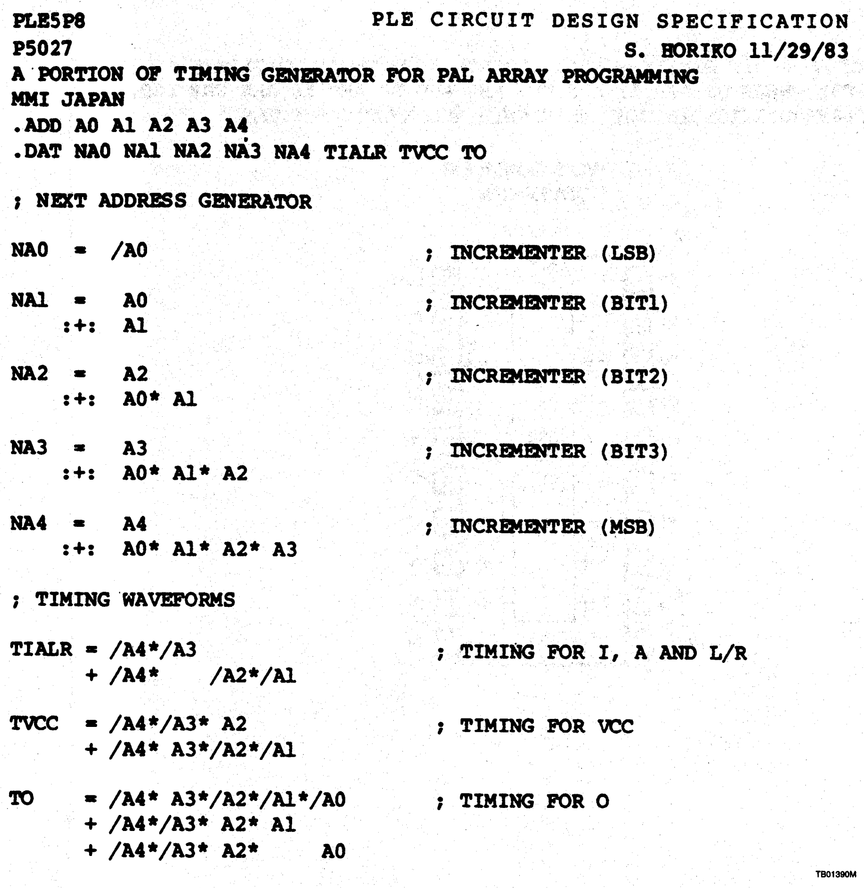
THIS PLE8P8 CONVERTS AN 8-BIT BINARY NUMBER (D7-D0) INTO TWO'S COMPLEMENT REPRESENTATION (Y7-Y0) WHERE D7 AND Y7 ARE THE MSB AND D0 AND Y0 ARE THE LSB. TWO'S COMPLEMENT REPRESENTATION IS USED IN SIGNED ARITHMETIC SYSTEMS.

8-BIT TWO'S COMPLEMENT CONVERSION

PLE8P8



TB01380M



```
PLE5P8                        PLE CIRCUIT DESIGN SPECIFICATION
P5027                                       S. HORIKO 11/29/83
A PORTION OF TIMING GENERATOR FOR PAL ARRAY PROGRAMMING
MMI JAPAN
.ADD A0 A1 A2 A3 A4
.DAT NA0 NA1 NA2 NA3 NA4 TIALR TVCC TO

; NEXT ADDRESS GENERATOR

NA0  =  /A0                    ; INCREMENTER (LSB)

NA1  =   A0                    ; INCREMENTER (BIT1)
    :+:  A1

NA2  =   A2                    ; INCREMENTER (BIT2)
    :+:  A0* A1

NA3  =   A3                    ; INCREMENTER (BIT3)
    :+:  A0* A1* A2

NA4  =   A4                    ; INCREMENTER (MSB)
    :+:  A0* A1* A2* A3

; TIMING WAVEFORMS

TIALR = /A4*/A3                 ; TIMING FOR I, A AND L/R
      + /A4*    /A2*/A1

TVCC  = /A4*/A3* A2             ; TIMING FOR VCC
      + /A4* A3*/A2*/A1

TO    = /A4* A3*/A2*/A1*/A0     ; TIMING FOR O
      + /A4*/A3* A2* A1
      + /A4*/A3* A2*     A0
```

TB01390M

```
FUNCTION TABLE

A4 A3 A2 A1 A0 NA4 NA3 NA2 NA1 NA0 TIALR TVCC TO

;          NNNNN
;AAAAA     AAAAA   TIMING WAVEFORMS
;43210     43210   TIALR  TVCC   TO     ; ## ;    COMMENTS
----------------------------------------------------------------------
 LLLLL     LLLLH     H     L     L      ; 01 ;    ASSERT TIALR
 LLLLH     LLLHL     H     L     L      ; 02 ;
 LLLHL     LLLHH     H     L     L      ; 03 ;
 LLLHH     LLHLL     H     L     L      ; 04 ;
 LLHLL     LLHLH     H     H     L      ; 05 ;    ASSERT TVCC
 LLHLH     LLHHL     H     H     H      ; 06 ;    ASSERT TO
 LLHHL     LLHHH     H     H     H      ; 07 ;
 LLHHH     LHLLL     H     H     H      ; 08 ;
 LHLLL     LHLLH     H     H     H      ; 09 ;
 LHLLH     LHLHL     H     H     L      ; 10 ;    CLEAR TO
 LHLHL     LHLHH     L     L     L      ; 11 ;    CLEAR TIALR & TVCC
 LHLHH     LHHLL     L     L     L      ; 12 ;
 LHHLL     LHHLH     L     L     L      ; 13 ;
 LHHLH     LHHHL     L     L     L      ; 14 ;
 LHHHL     LHHHH     L     L     L      ; 15 ;
 LHHHH     HLLLL     L     L     L      ; 16 ;
 HLLLL     HLLLH     L     L     L      ; 17 ;
 HLLLH     HLLHL     L     L     L      ; 18 ;
 HLLHL     HLLHH     L     L     L      ; 19 ;
 HLLHH     HLHLL     L     L     L      ; 20 ;
 HLHLL     HLHLH     L     L     L      ; 21 ;
 HLHLH     HLHHL     L     L     L      ; 22 ;
 HLHHL     HLHHH     L     L     L      ; 23 ;
 HLHHH     HHLLL     L     L     L      ; 24 ;
 HHLLL     HHLLH     L     L     L      ; 25 ;
 HHLLH     HHLHL     L     L     L      ; 26 ;
 HHLHL     HHLHH     L     L     L      ; 27 ;
 HHLHH     HHHLL     L     L     L      ; 28 ;
 HHHLL     HHHLH     L     L     L      ; 29 ;
 HHHLH     HHHHL     L     L     L      ; 30 ;
 HHHHL     HHHHH     L     L     L      ; 31 ;
 HHHHH     LLLLL     L     L     L      ; 32 ;
----------------------------------------------------------------------
```

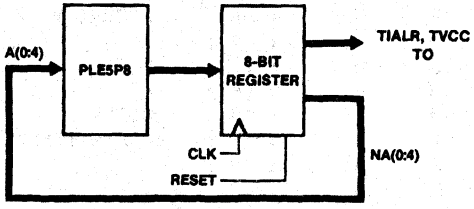
TB01400M

DESCRIPTION

THIS LOGIC SPECIFICATION IS A TIMING SIGNAL GENERATOR TO BE USED FOR ARRAY PROGRAMMING OF PAL DEVICES. A PLE5P8 FOLLOWED BY AN 8-BIT REGISTER ARE USED TO IMPLEMENT THIS FUNCTION.

THE PLE CONTAINS BOTH 5-BIT NEXT ADDRESS AND 3-BIT WAVEFORMS. TIALR OUTPUT IS A TIMING WAVEFORM FOR I, A, AND L/R SIGNALS, AND TVCC AND TO OUTPUTS ARE USED FOR VCC AND O SIGNALS, RESPECTIVELY.

THE SCHEMATIC IS AS FOLLOWS:

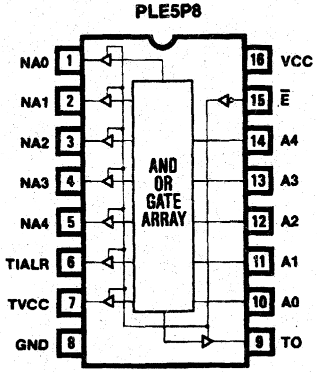


APPLYING 200KHz CLOCK SIGNAL TO THE CLK INPUT OF THE REGISTER GENERATES THE FOLLOWING TIMINGS:

1. I, A, AND L/R WIDTH : 50 usec
2. tD2 : 20 usec
3. tD : 5 usec
4. tVCCP : 30 usec
5. Tp : 20 usec

BECAUSE THE TIMING PATTERNS ARE STORED IN THE PROM, WE CAN EASILY CALIBRATE THE RELATIONS AND THE PERIOD AMONG THOSE SIGNALS TO MAKE AN OPTIMUM CONDITION.

A PORTION OF A TIMING GENERATOR FOR PAL LOGIC CIRCUIT ARRAY PROGRAMMING



TB01410M

```
PLE5P8                       PLE CIRCUIT DESIGN SPECIFICATION
P5028                                            S. HORIKO 11/29/83
TIMING GENERATOR FOR PAL DEVICE SECURITY FUSE PROGRAMMING
MMI JAPAN
.ADD A0 A1 A2 A3 A4
.DAT NA0 NA1 NA2 NA3 NA4 TVCC TP01 TP11

; NEXT ADDRESS GENERATOR
;  ( THE COUNTER LOCKS UP AT COUNT-22 )

NA0  = /A4*        /A1*/A0       ; INCREMENTER (LSB)
     + /A4*         A1*/A0       ; INCREMENTER (LSB)
     +  A4*/A3*/A2*    /A0       ; INCREMENTER (LSB)
     +  A4*/A3* A2*/A1           ; INCREMENTER (LSB)

NA1  = /A4*        /A1* A0       ; INCREMENTER (BIT1)
     + /A4*         A1*/A0       ; INCREMENTER (BIT1)
     +  A4*/A3*/A2*/A1* A0       ; INCREMENTER (BIT1)
     +  A4*/A3*/A2* A1*/A0       ; INCREMENTER (BIT1)

NA2  = /A4*     A2*/A1           ; INCREMENTER (BIT2)
     + /A4*     A2*    /A0       ; INCREMENTER (BIT2)
     + /A4*    /A2* A1* A0       ; INCREMENTER (BIT2)
     +  A4*/A3* A2*/A1           ; INCREMENTER (BIT2)
     +  A4*/A3*/A2* A1* A0       ; INCREMENTER (BIT2)

NA3  = /A4* A3*/A2               ; INCREMENTER (BIT3)
     + /A4* A3*    /A1           ; INCREMENTER (BIT3)
     + /A4* A3*        /A0       ; INCREMENTER (BIT3)
     + /A4*/A3* A2* A1* A0       ; INCREMENTER (BIT3)

NA4  = /A4* A3* A2* A1* A0       ; INCREMENTER (MSB)
     +  A4*/A3*/A2               ; INCREMENTER (MSB)
     +  A4*/A3*    /A1           ; INCREMENTER (MSB)

; TIMING WAVEFORMS

TVCC = /A4                       ; TIMING FOR VCC
     +  A4*/A3*/A2*/A1
     +  A4*/A3*/A2*    /A0

TP01 = /A4*/A3* A2               ; TIMING FOR PIN 01
     + /A4*/A3*     A1
     + /A4*/A3*         A0
     + /A4* A3*/A2*/A1*/A0

TP11 = /A4* A3* A2               ; TIMING FOR PIN 11
     + /A4* A3*     A1
     +  A4*/A3*/A2*/A1
```

TB01420M

```
FUNCTION TABLE

A4 A3 A2 A1 A0 NA4 NA3 NA2 NA1 NA0 TVCC TP01 TP11

;          NNNNN
;AAAAA     AAAAA    TIMING WAVEFORMS
;43210     43210    TVCCP  TP01 TP11      ; ## ;     COMMENTS
--------------------------------------------------------------------------------
 LLLLL     LLLLH      H      L    L       ; 01 ;     ASSERT TVCC, START HERE
 LLLLH     LLLHL      H      H    L       ; 02 ;     ASSERT TP01
 LLLHL     LLLHH      H      H    L       ; 03 ;
 LLLHH     LLHLL      H      H    L       ; 04 ;
 LLHLL     LLHLH      H      H    L       ; 05 ;
 LLHLH     LLHHL      H      H    L       ; 06 ;
 LLHHL     LLHHH      H      H    L       ; 07 ;
 LLHHH     LHLLL      H      H    L       ; 08 ;
 LHLLL     LHLLH      H      H    L       ; 09 ;     CLEAR TP01
 LHLLH     LHLHL      H      L    L       ; 10 ;     ASSERT TP11
 LHLHL     LHLHH      H      L    H       ; 11 ;
 LHLHH     LHHLL      H      L    H       ; 12 ;
 LHHLL     LHHLH      H      L    H       ; 13 ;
 LHHLH     LHHHL      H      L    H       ; 14 ;
 LHHHL     LHHHH      H      L    H       ; 15 ;
 LHHHH     HLLLL      H      L    H       ; 16 ;
 HLLLL     HLLLH      H      L    H       ; 17 ;
 HLLLH     HLLHL      H      L    H       ; 18 ;
 HLLHL     HLLHH      H      L    L       ; 19 ;     CLEAR TP11
 HLLHH     HLHLL      L      L    L       ; 20 ;     CLEAR TVCC
 HLHLL     HLHLH      L      L    L       ; 21 ;
 HLHLH     HLHLH      L      L    L       ; 22 ;     LOOP HERE UNTIL RESET
--------------------------------------------------------------------------------
```

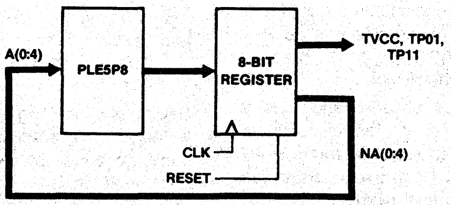
TB01430M

DESCRIPTION

THIS LOGIC SPECIFICATION IS A TIMING SIGNAL GENERATOR TO BE USED FOR SECURITY FUSE PROGRAMMING OF PAL DEVICES. A PLE5P8 FOLLOWED BY AN 8-BIT REGISTER ARE USED TO IMPLEMENT THIS FUNCTION.

THE PLE LOGIC CIRCUIT CONTAINS TWO FUNCTIONS IN THE SINGLE CHIP. THE FIRST FUNCTION IS A UNIQUE COUNTER USED FOR NEXT ADDRESS GENERATION. THE COUNTER INCREMENTS UP TO COUNT-21 AND THEN LOCKS UP THE INCREMENTAL OPERATION AT COUNT-22. THE SECOND FUNCTION IS A TIMING GENERATOR USED FOR DEFINING TIMING RELATIONSHIP AMOUNG VCC, P01, AND P11 SIGNALS.

THE SCHEMATIC IS AS FOLLOWS:



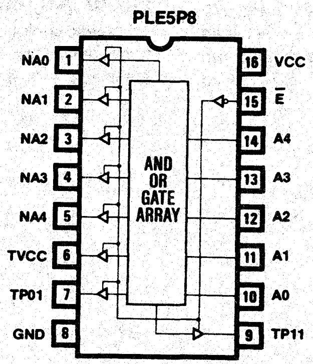
THIS LOGIC OUTPUTS A SEQUENCE OF TIMING PATTERNS DURING THE INCREMENTAL OPERATION AND THEN HOLDS ALL OUTPUTS LOW UNTIL A RESET SIGNAL FOR THE 8-BIT REGISTER IS APPLIED.

APPLYING 200 KHz CLOCK SIGNAL TO THE CLK INPUT OF THE REGISTER, THE FOLLOWING TIMINGS ARE GENERATED:

1. VCC WIDTH : 95 usec
2. TPP : 40 usec
3. tD : 5 usec

BY APPLYING THIS DESIGN METHOD, WE CAN EASILY GENERATE A SEQUENCE OF UNIQUELY DEFINED PATTERNS EACH TIME THE RESET PULSE IS APPLIED.

TIMING GENERATOR FOR PAL SECURITY FUSE PROGRAMMING



TB01440M

Fast Arithmetic Look-up

In performing arithmetic operations like trigonometric functions, multiplications and division, in order to reduce the delay, look-up tables are often used.

Sine Look-up

For trigonometric functions like sine function, it is very time-consuming to generate the function using the polynomial which represents the function. PLE devices can provide a very good alternative for sine look-up. An example is to use a 2Kx8 PLE device to do a sine look-up of an 11-bit input to 8-bit sine outputs.

Since sine function has the following property: sin (x) = sin (π – x) = –sin (π + x) = –sin (2π – x) = sin (2π + x), what is needed is just the sine function for $0 < x < \pi/2$; the rest can be easily calculated using the above relations. In order to fully utilize the dynamic range, the inputs of the sine look-up PLE device should be normalized to $(\pi/2)/(2^n) = \pi[2^{n\ +\ 1}]$ where n is the number of address lines to the device.

Since n is fixed for the PLE device chosen, and π is a constant, for the look-up table $\pi/[2^{n\ +\ 1}]$is a constant. Therefore, if the sine function of a given x is to be found, x will first be multiplied by the constant $[2^{n\ +\ 1}]/\pi$ and sent to the address of the PLE device to get the final result.

Cos (x) is related to sine function as sin ($\pi/2$ – x). Thus the cosine function can also be found in the same manner by using $\pi/2$ – x instead of just x. Other functions like tangent, secant etc., can also be found as a function of sine.

To increase the dynamic range of outputs, we can just use another PLE device to generate the less-significant bits of the sine function.

If a larger dynamic range is needed for the inputs, the result may be approximated using the Taylor series:

$$f (X) = f (X0) + f' (X0) (X - X0) + 1/2f'' (X0) (X - X0)^2 + ...$$

Where f' and f'' are the first and second derivations of f. Since X0 by itself represents a resolution of 2^{-n}, and X is X0 concatenated with the rest of the bits, X – X0 must lie between 0 and $1/2^{-n}$. For f (X) = sin (X),

f (X0) = sin (X0)

f' (X0) = cos (X0)

and f'' (X0) = –sin (X0)

So f'' (X0) is between –1 and 0 for X0 lies between 0 and $\pi/2$ and $X - X0 < 2^{-n}$. Therefore, the last term will be between '$1/2^n$ and 0, and as long as we do not want to expand the dynamic range of X beyond 2n-bits, it should be sufficient to approximate sin (X) in the first two terms:

sin (X) + sin (X0) + cos (X0) (X – X0)

Since X – X0 is represented by only the bits after the more significant n-bits, and cos (X0) = sin ($\pi/2$ – X0), the implementation will be very simple.

Division

Division will normally be much slower than multiplication. There are several ways to perform division. Bit-by-bit division restoring and nonrestoring algorithms are generally very slow. Another way is to use several bits at a time division which is faster than the previous methods. A third way is to multiply the dividend by the inverse of the divisor. The inverse of the divisor can be found by getting an approximation followed by iterations.

The approximation is again given by the Taylor series:

$$f (X) = f (X0) + f' (X0) (X - X0) + 1/2f'' (X0) (X - X0)^2 + ...$$

and f (X0) = 1/X0

f' (X0) = $-1/X0^2$

f'' (X0) = $2/X0^3$

Say X0 is 8-bits long and the first approximation of the inverse is found using a 256x8 PLE device. The first approximation can be obtained by subtracting $(X - X0)/(X0^2)$. Since the first approximation is limited by an error of approximately $(X - X0)^2/X0^2$, and if X0 at least 1, the error is limited by approximately $(X - X0)^2$. Since X0 has an 8-bit resolution, X – X0 is represented by the rest of the bits. The resolution of the second approximation will be about 16 bits. The third approximation is similarly deduced and has a resolution of about 32 bits, and the fourth has a resolution of about 64 bits.

The inverse thus obtained is then multiplied by the dividend to give the quotient.

Scaling

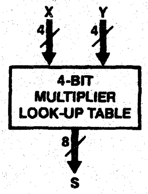
In arithmetic operations, scaling is sometimes needed. Scaling normally involves multiplication or division by a constant. If this constant can be expressed in 2^n where n is an integer, then scaling is simply shifting. Scaling with other constants may need a multiplier. A multiplier is more expensive and has a higher pin count than using a PLE device because the constant that the operand is to be scaled by is not required as an input as in the case of a multiplier. This will tremendously reduce the overhead for data scaling.

Other Applications

Arithmetic look-up are also very useful for arithmetic operations where conventional binary integral arithmetic is not applicable — like residue arithmetic, and distributed arithmetic.

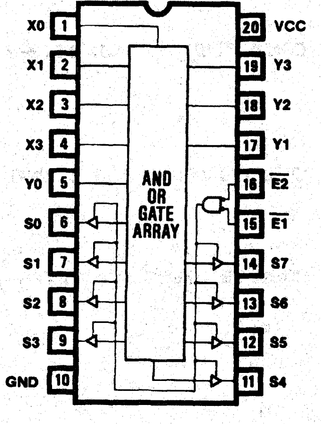
```
PLE8P8                              PLE CIRCUIT DESIGN SPECIFICATION
P5018                                          VINCENT COLI 12/08/82
4-BIT MULTIPLIER LOOK-UP TABLE
MMI SANTA CLARA, CALIFORNIA
.ADD X0 X1 X2 X3 Y0 Y1 Y2 Y3
.DAT S0 S1 S2 S3 S4 S5 S6 S7


S7,S6,S5,S4,S3,S2,S1,S0 = X3,X2,X1,X0 .*. Y3,Y2,Y1,Y0      ; S = X * Y

FUNCTION TABLE

X3 X2 X1 X0 Y3 Y2 Y1 Y0 S7 S6 S5 S4 S3 S2 S1 S0

;-OPERANDS-     PRODUCTS
;XXXX  YYYY     SSSSSSSS     COMMENTS
;3210  3210     76543210
------------------------------------------
 LLLL  LLLL     LLLLLLLL      0 *  0  =   0
 LLLH  HHHH     LLLLHHHH      1 * 15  =  15
 HHHH  LLLH     LLLLHHHH     15 *  1  =  15
 HHHH  HHHH     HHHLLLLH     15 * 15  = 225
------------------------------------------

DESCRIPTION

THIS PLE8P8 PERFORMS 4-BIT LOOK-UP TABLE MULTIPLICATION.  THE DEVICE
ACCEPTS TWO 4-BIT OPERANDS (X3-X0 AND Y3-Y0) TO PRODUCE THE 8-BIT
PRODUCT (S7-S0).  THE PLE8P8 ALSO HAS THREE-STATE OUTPUTS WITH TWO
ACTIVE-LOW OUTPUT ENABLE CONTROL PINS (/E1 AND /E2).
```

```
             X3 X2 X1 X0 }  TWO 4-BIT
      X      Y3 Y2 Y1 Y0 }  OPERANDS
-------------------------
S7 S6 S5 S4 S3 S2 S1 S0
         8-BIT PRODUCT
```



4-BIT MULTIPLIER LOOK-UP TABLE

PLE8P8



TB01450M

```
PLE5P8                                        PLE CIRCUIT DESIGN SPECIFICATION
P5023                                                 PETER ZECHERLE 03/06/84
ARC TANGENT LOOK-UP TABLE
MMI GMBH MUNICH
.ADD A0 A1 A2 A3 A4
.DAT F0 F1 F2 F3 F4 F5 F6 F7


F0 =      A1*    /A3*/A4      ; COMPUTE DIGIT FOR 2EXP-7 (0.00078125) (LSB)
   +          A2*/A3
   +  A0*/A1*     A3*/A4
   + /A0* A1*        /A4
   + /A0*/A1*    /A3* A4
   +  A0*     A2
   +      A1* A2

F1 =     /A1*     A3*/A4      ; COMPUTE DIGIT FOR 2EXP-6 (0.015625)
   +  A0*        /A3* A4
   +      A1*    /A3* A4
   +          A2*/A3* A4
   +  A0* A1*    /A3
   +  A0*     A2*/A3
   + /A0* A1* A2*    /A4
   +  A0*    /A2* A3*/A4

F2 =  A0*        /A3*/A4      ; COMPUTE DIGIT FOR 2EXP-5 (0.03125)
   +      A1*/A2*    /A4
   +              A3* A4
   + /A0*     A2* A3*/A4
   +     /A1* A2* A3*/A4

F3 =      A1*/A2*    /A4      ; COMPUTE DIGIT FOR 2EXP-4 (0.0625)
   +     /A1* A2*    /A4
   + /A0*         A3*/A4
   +     /A1*     A3*/A4

F4 =     /A1*     A3*/A4      ; COMPUTE DIGIT FOR 2EXP-3 (0.125)
   +  A0* A1*/A2*    /A4
   +      A1* A2*/A3*/A4
   + /A0*         A3*/A4

F5 =  A0*/A1*        /A4      ; COMPUTE DIGIT FOR 2EXP-2 (0.25)
   +          A2*/A3*/A4
   +         /A2* A3*/A4
   + /A0*         A3*/A4

F6 =  A0*/A1*/A2*/A3          ; COMPUTE DIGIT FOR 2EXP-1 (0.5)
   +  A0* A1* A2* A3
   +                  A4

F7 =      A1                  ; COMPUTE DIGIT FOR 2EXP0 (1) (MSB)
   +          A2
   +              A3
   +                  A4
```

TB01460M

FUNCTION TABLE

```
;----ANGLE----     --------F = ARCTAN(A)--------
;   INTEGER        INTEGER       FRACTIONS                      ---F = ARCTAN(A)---
A4 A3 A2 A1 A0        F7     F6 F5 F4 F3 F2 F1 F0      ;ANGLE    LOOK-UP   CALCULATED
-----------------------------------------------------------------------------------
L  L  L  L  L         L      L  L  L  L  L  L  L          0       0.0000     0.0000
L  L  L  L  H         L      H  H  L  L  H  L  L          1       0.7813     0.7854
L  L  L  H  L         H      L  L  L  H  H  L  H          2       1.1016     1.1071
L  L  H  L  L         H      L  H  L  H  L  L  H          4       1.3203     1.3258
L  L  H  L  H         H      L  H  L  H  H  H  H          5       1.3672     1.3734
L  H  L  L  L         H      L  H  H  H  L  H  L          8       1.4531     1.4464
H  L  L  L  L         H      H  L  L  L  L  L  H         16       1.5078     1.5084
H  H  H  H  H         H      H  L  L  L  H  L  H         31       1.5391     1.5385
-----------------------------------------------------------------------------------
```

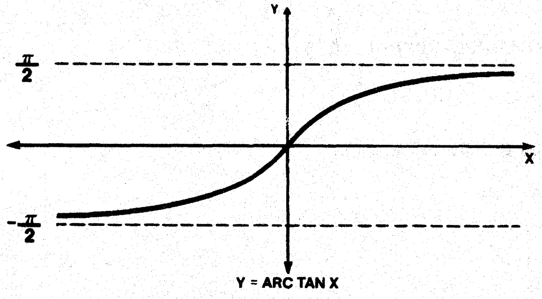
DESCRIPTION

THIS APPLICATION ILLUSTRATES THE CALCULATION OF THE ARC TANGENT FUNCTION USING A PLE5P8 AS A LOOK-UP TABLE. OTHER TRIGONOMETRIC FUNCTIONS (SUCH AS SINE, COSINE, COTANGENT, SECANT, COSECANT AND THEIR ARC INVERSE EQUIVALENT FUNCTIONS) OR HYPERBOLIC FUNCTIONS CAN ALSO BE CONSTRUCTED USING PLE DEVICES AS LOOK-UP TABLES.

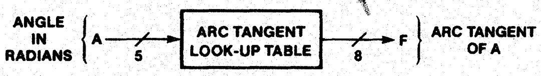
```
F = ARCTAN(A)      WHERE F = ARC TANGENT OF A
                         A = ANGLE IN RADIANS
```

EXAMPLE: FOR A = 5, F = ARCTAN(5) = 1.3672

A PLE DEVICE WITH MORE INPUTS, SUCH AS THE PLE11P8, SHOULD BE USED TO CONSTRUCT A LOOK-UP TABLE WHEN ADDITIONAL ACCURACY IS REQUIRED.

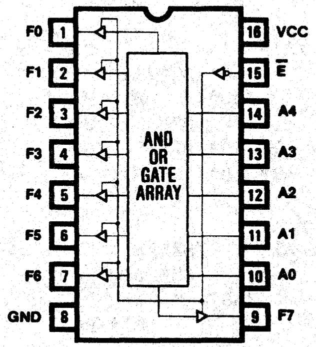


Y = ARC TAN X



ARC TANGENT LOOK-UP TABLE

PLE5P8



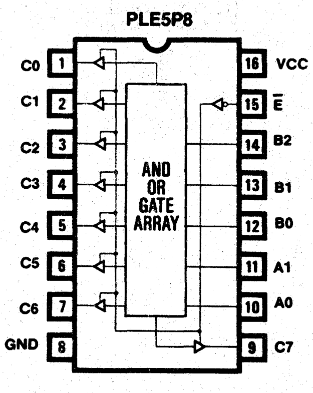
TB01470M

```
PLE5P8                               PLE CIRCUIT DESIGN SPECIFICATION
P5024                                         WILLY VOLDAN 06/02/84
HYPOTENUSE OF A RIGHT TRIANGLE LOOK-UP TABLE
MMI GMBH MUNICH
.ADD A0 A1 B0 B1 B2
.DAT C0 C1 C2 C3 C4 C5 C6 C7

C0 =      A0*/B2* B1          ; COMPUTE DIGIT FOR 2EXP-5 (0.03125) (LSB)
   + /A1* A0* B2*/B1
   +  A1*    /B2*     B0
   +  A1*    /B2* B1
   +  A1*/A0* B2*/B1*/B0
   +  A1* A0*     B1* B0
   +      A0*/B2*     B0

C1 =      A0*     B1*/B0      ; COMPUTE DIGIT FOR 2EXP-4 (0.0625)
   + /A1* A0* B2
   +  A1*/A0*/B2*     B0
   +  A1*/A0* B2*    /B0
   +      A0* B2*     B0
   +  A1* A0*     B1

C2 =      A0*/B2*     B0      ; COMPUTE DIGIT FOR 2EXP-3 (0.125)
   + /A1* A0*/B2* B1
   +  A1*/A0* B2*/B1
   +  A1* A0* B2* B1
   +  A1*    /B2*/B1* B0

C3 = /A1* A0*/B2*/B1* B0      ; COMPUTE DIGIT FOR 2EXP-2 (0.25)
   +  A1*/A0*     B1*/B0
   +  A1*/A0* B2
   +  A1*     B2*     B0

C4 =  A1*/A0*/B2* B1          ; COMPUTE DIGIT FOR 2EXP-1 (0.5)
   +  A1* A0* B2*/B1* B0
   +  A1* A0*     B1*/B0

C5 = /A1*             B0      ; COMPUTE DIGIT FOR 2EXP0 (1)
   +      A0*/B2*/B1
   +     /A0*     B1* B0
   +          B2*     B0
   +  A1* A0*/B2*    /B0
   +  A1* A0*    /B1

C6 = /A1*         B1          ; COMPUTE DIGIT FOR 2EXP1 (2)
   +  A1*    /B2*/B1
   +          B2* B1
   +     /A0*     B1
   +              B1*/B0

C7 =          B2              ; COMPUTE DIGIT FOR 2EXP2 (4) (MSB)
   +  A1* A0*     B1* B0
```

TB01480M

```
FUNCTION TABLE

;-LENGTH OF SIDES-    LENGTH OF THE HYPOTENUSE
;SIDE A    SIDE B     INTEGER      FRACTION          SIDES   LENGTH OF HYPOTENUSE
 A1 A0    B2 B1 B0    C7 C6 C5  C4 C3 C2 C1 C0       ;A B    LOOK-UP    CALCULATED
-------------------------------------------------------------------------------------
 L  L     L  L  L     L  L  L   L  L  L  L  L         0 0     0.00         0.00
 L  L     L  L  H     L  L  H   L  L  L  L  L         0 1     1.00         1.00
 L  L     L  H  L     L  H  L   L  L  L  L  L         0 2     2.00         2.00
 L  L     H  L  L     H  L  L   L  L  L  L  L         0 4     4.00         4.00
 L  H     L  L  L     L  L  H   L  L  L  L  L         1 0     1.00         1.00
 H  L     L  L  L     L  H  L   L  L  L  L  L         2 0     1.00         1.00
 H  L     L  H  L     L  H  L   H  H  L  L  H         2 2     2.78         2.83
 H  L     H  L  L     H  L  L   L  H  H  H  H         2 4     4.47         4.47
 H  H     L  H  L     L  H  H   H  L  L  H  H         3 2     3.59         3.61
 H  H     H  H  H     H  H  H   L  H  H  H  H         3 7     7.47         7.62
-------------------------------------------------------------------------------------
```

HYPOTENUSE OF A RIGHT TRIANGLE LOOK-UP TABLE



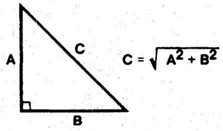
DESCRIPTION

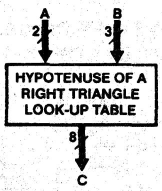
THE GENERATION OF COMPLEX ARITHMETIC FUNCTIONS SUCH AS THE PYTHAGOREAN THEOREM IS GENERALLY VERY DIFFICULT TO IMPLEMENT DIRECTLY IN HARDWARE. HOWEVER, IMPLEMENTING THE FUNCTION AS A LOOK-UP TABLE USING A PLE GREATLY SIMPLIFIES THE PROBLEM.

THIS EXAMPLE ILLUSTRATES HOW TO IMPLEMENT A LOOK-UP TABLE IN A PLE5P8 WHICH CALCULATES THE LENGTH OF THE HYPOTENUSE OF A RIGHT TRIANGLE AS A FUNCTION OF THE LENGTH OF THE TWO REMAINING SIDES OF THE TRIANGLE. THE THEOREM OF PATHAGOREAN STATES THAT THE LENGTH OF THE HYPOTENUSE OF A RIGHT TRIANGLE IS EQUAL TO THE SQUARE ROOT OF THE SUM OF THE SQUARE OF THE OTHER TWO SIDES OR C = SQRT(A**2 + B**2). THE INPUTS, "A" AND "B", CORRESPOND TO THE SIDES ADJACENT TO THE RIGHT ANGLE (I.E. 90 DEGREE ANGLE), WHILE THE OUTPUT, "C", CORRESPONDS TO THE SIDE OPPOSITE TO THE RIGHT ANGLE WHICH IS CALLED THE HYPOTENUSE.

```
C = SQRT(A**2 + B**2)     WHERE C = LENGTH OF SIDE C (THE HYPOTENUSE)
                                A = LENGTH OF SIDE A
                                B = LENGTH OF SIDE B
```

EXAMPLE: FOR A = 2 AND B = 4, C = SQRT(2**2 + 4**2) = 4.47





TB01500M

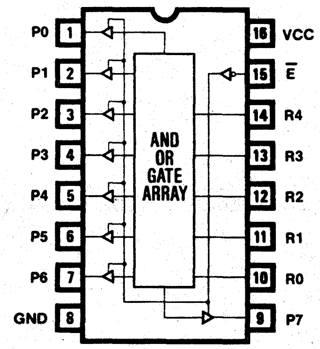
```
PLE5P8                             PLE CIRCUIT DESIGN SPECIFICATION
P5025                                           PETER WITTFOTH 06/02/84
PERIMETER OF A CIRCLE LOOK-UP TABLE
MMI GMBH MUNICH
.ADD R0 R1 R2 R3 R4
.DAT P0 P1 P2 P3 P4 P5 P6 P7


P0 =     /R1* R2*/R3*/R4      ; COMPUTE DIGIT FOR 2EXP0 (1) (LSB)
   + /R0*/R1* R2*     /R4
   +      R1* R2*      R4
   +      R1*/R2*/R3
   +  R0*/R1*/R2* R3
   + /R0* R1*/R2
   +      R1*/R2*     /R4
   +     /R1*/R2*      R4

P1 =  R0*    /R2*/R3          ; COMPUTE DIGIT FOR 2EXP1 (2)
   + /R0* R1* R2*/R3
   + /R0*    /R2* R3
   +  R0*     R2* R3
   + /R0*     R2*/R3* R4
   +  R0*/R1* R2*     /R4
   +     /R1* R2* R3*/R4
   +  R0* R1*     R3* R4

P2 =  R0*/R1*    /R3*/R4      ; COMPUTE DIGIT FOR 2EXP2 (4)
   +  R0* R1*/R2*/R3* R4
   + /R0* R1* R2* R3* R4
   + /R0* R1*/R2*     /R4
   +      R1* R2*/R3*/R4
   +  R0* R1*     R3*/R4
   + /R0*/R1*/R2*      R4
   +     /R1* R2*/R3* R4
   +  R0*/R1*     R3* R4

P3 = /R0* R1*/R2*     /R4     ; COMPUTE DIGIT FOR 2EXP3 (8)
   +  R0*/R1*/R2* R3
   +  R0*/R1*/R2*      R4
   +  R0*    /R2* R3* R4
   +  R0*     R2*/R3*/R4
   + /R0*     R2* R3*/R4
   +      R1* R2* R3*/R4
   + /R0*     R2*/R3* R4
   +      R1* R2*/R3* R4
   + /R0* R1* R2*      R4
   + /R0*/R1* R2*/R3

P4 =  R0* R1*/R2*/R3          ; COMPUTE DIGIT FOR 2EXP4 (16)
   +      R1*/R2*/R3* R4
   + /R0*/R1* R2*/R3
   +  R0*/R1* R2*     /R4
   +     /R1*/R2* R3
   + /R0* R1*     R3*/R4
   +      R1* R2* R3*/R4
   +     /R1*     R3* R4
   + /R0*     R2* R3* R4
```

TB01510M

```
P5 =      R1* R2*/R3*/R4       ; COMPUTE DIGIT FOR 2EXP5 (32)
   +     /R1*/R2* R3*/R4
   +         /R2*/R3* R4
   +      R1*/R2*      R4
   +     /R1* R2* R3* R4
   + /R0* R1*/R2* R3
   + /R0*/R1* R2*      R4
   + /R0*     R2* R3* R4

P6 =  R0* R1*     R3*/R4       ; COMPUTE DIGIT FOR 2EXP6 (64)
   + /R0*/R1*    /R3* R4
   +          R2* R3*/R4
   +         /R2*/R3* R4
   +  R0* R1* R2* R3

P7 =  R0*     R2*      R4      ; COMPUTE DIGIT FOR 2EXP7 (128) (MSB)
   +      R1* R2*      R4
   +              R3* R4

FUNCTION TABLE

;---RADIUS----    -------PERIMETER-------
;   INTEGER       MSB     INTEGER     LSB               PERIMETER OF A CIRCLE
R4 R3 R2 R1 R0    P7 P6 P5 P4 P3 P2 P1 P0    ;RADIUS      LOOK-UP  CALCULATED
------------------------------------------------------------------------------
L  L  L  L  L     L  L  L  L  L  L  L  L         0           0       0.0
L  L  L  L  H     L  L  L  L  L  H  H  L         1           6       6.3
L  L  L  H  L     L  L  L  L  H  H  L  H         2          13      12.6
L  L  L  H  H     L  L  L  H  L  L  H  H         3          19      18.8
L  L  H  L  L     L  L  L  H  H  L  L  H         4          25      25.1
L  H  L  L  L     L  L  H  H  L  L  H  L         8          50      50.3
H  L  L  L  L     L  H  H  L  L  H  L  H        16         101     100.5
H  H  H  H  H     H  H  L  L  L  L  H  H        31         195     194.8
------------------------------------------------------------------------------
```

PERIMETER OF A CIRCLE
LOOK-UP TABLE

PLE5P8



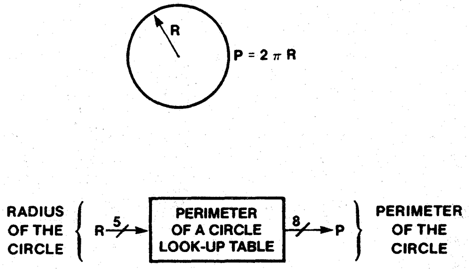
TB01520M

DESCRIPTION

THIS EXAMPLE ILLUSTRATES HOW TO IMPLEMENT A LOOK-UP TABLE IN A PLE5P8 FOR THE PERIMETER OF A CIRCLE AS A FUNCTION OF THE RADIUS. THE INPUT PINS (R4-R0), WHICH REPRESENT THE RADIUS OF A CIRCLE, ARE MULTIPLIED BY 2 TIMES PI IN ORDER TO CALCULATE THE PERIMETER OF A CIRCLE (P7-P0). THIS LOOK-UP TABLE IS VALID FOR RADII BETWEEN 0 AND 31. A PLE8P8 SHOULD BE USED INSTEAD IF A LARGER RADIUS RANGE (BETWEEN 0 AND 81) IS REQUIRED.

```
P = 2*PI*R     WHERE P  = PERIMETER OF THE CIRCLE
                     PI = 3.1415
                     R  = RADIUS OF THE CIRCLE (BETWEEN 0 AND 31)
```

EXAMPLE: FOR R = 3, P = 2*PI*3 = 19



TB01530M

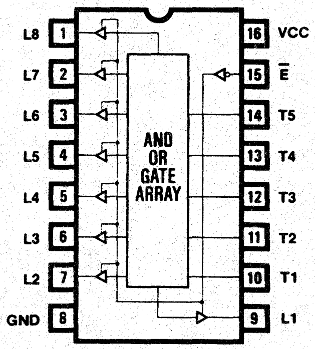
```
PLE5P8                                       PLE CIRCUIT DESIGN SPECIFICATION
P5026                                                   WILLY VOLDAN 06/03/84
PERIOD OF OSCILLATION FOR A MATHEMATICAL PENDULUM LOOK-UP TABLE
MMI GMBH MUNICH
.ADD L0 L1 L2 L3 L4
.DAT T0 T1 T2 T3 T4 T5 T6 T7

T0 = /L4*    /L2*/L1* L0          ; COMPUTE DIGIT FOR 2EXP-5 (0.03125) (LSB)
   +     /L3*/L2* L1*/L0
   +      L3*/L2*     L0
   + /L4*     L2*/L1*/L0
   +  L4* L3*         L0
   +  L4*/L3*     L1
   +  L4*/L3* L2*    /L0

T1 =         /L2* L1*/L0          ; COMPUTE DIGIT FOR 2EXP-4 (0.0625)
   + /L4*/L3* L2*     L0
   + /L4* L3*/L2* L1
   + /L4*     L2*/L1*/L0
   +  L4*/L3*/L2* L1
   +     /L3* L2*/L1
   +  L4* L3*    /L1* L0
   +  L4* L3*     L1*/L0

T2 = /L4*/L3*     L1*/L0          ; COMPUTE DIGIT FOR 2EXP-3 (0.125)
   + /L4* L3*/L2*     L0
   +  L4*/L3*/L2*/L1*/L0
   +  L4*/L3*     L1* L0
   +  L4* L3*/L2* L1*/L0
   +  L4* L3* L2*/L1
   +  L4*     L2*     L0
   + /L4*/L3*    /L1* L0
   + /L4*/L3* L2*    /L0

T3 = /L4* L3*     L1* L0          ; COMPUTE DIGIT FOR 2EXP-2 (0.25)
   +  L4*/L3*     L1
   +  L4* L3* L2*/L1
   +     /L3*/L2*/L1* L0
   +     /L3* L2* L1* L0
   +      L3*/L2* L1* L0
   +      L3* L2*/L1* L0
   + /L4*     L2*/L1* L0

T4 = /L4*/L3*     L1*/L0          ; COMPUTE DIGIT FOR 2EXP-1 (0.5)
   +     /L3* L2* L1
   + /L4* L3* L2*/L1
   +  L4*/L3* L2
   +  L4*     L2* L1
   +          L2* L1*/L0
```

TB01540M

```
T5 = /L4*    /L2*    /L0         ; COMPUTE DIGIT FOR 2EXP0 (1)
   + /L4*    /L2*/L1
   +       L3*       /L0
   +       L3*/L2
   +       L3*    /L1
   +   L4* L3

T6 = /L4*    /L2* L1* L0         ; COMPUTE DIGIT FOR 2EXP1 (2)
   + /L4* L3*        /L0
   + /L4*/L3* L2
   + /L4*/L3*    /L1

T7 =       L3* L2* L1* L0        ; COMPUTE DIGIT FOR 2EXP2 (4) (MSB)
   +   L4


FUNCTION TABLE

;--AMPLITUDE--     --PERIOD OF OSCILLATION--
;   INTEGER          INTEGER       FRACTION                  PERIOD OF OSCILLATION
L4 L3 L2 L1 L0     T7 T6 T5 T4    T3 T2 T1 T0   ;AMPLITUDE    LOOK-UP  CALCULATED
-------------------------------------------------------------------------------
L  L  L  L  L      L  L  H  L     L  L  L  L         1        2.0000     2.0050
L  L  L  L  H      L  L  H  L     H  H  L  H         2        2.8125     2.8356
L  L  L  H  L      L  L  H  H     L  H  H  H         3        3.4375     3.4728
L  L  H  L  L      L  H  L  L     L  H  H  H         5        4.4375     4.4834
L  H  L  L  L      L  H  H  L     L  L  L  L         9        6.0000     6.0151
H  L  L  L  L      H  L  L  L     L  H  L  L        17        8.2500     8.2670
H  H  H  H  H      H  L  H  H     L  H  L  H        32       11.3125    11.3423
-------------------------------------------------------------------------------
```

PERIOD OF OSCILLATION
FOR A MATHEMATICAL PENDULUM
LOOK-UP TABLE

PLE5P8



TB01550M

DESCRIPTION

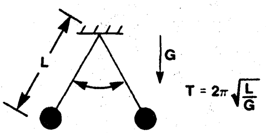
THIS PLE5P8 IS USED TO IMPLEMENT A LOOK-UP TABLE FOR THE PERIOD OF OSCILLATION OF A MATHEMATICAL PENDULUM. THE PERIOD OF OSCILLATION FOR MATHEMATICAL PENDULUM (T) IS DEPENDENT UPON ITS AMPLITUDE OF SWING (L) AND THE ACCELERATION DUE TO GRAVITY (G). THE PERIOD OF OSCILLATION IS CALCULATED USING THE FOLLOWING EQUATION:

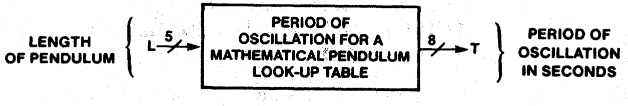
```
T = 2*PI*SQRT(L/G)      WHERE T  = PERIOD OF OSCILLATION IN SECONDS
                              PI = 3.14
                              L  = AMPLITUDE OF SWING IN METERS
                              G  = ACCELERATION DUE TO GRAVITY IN M/S/S
                                   (9.81 M/S/S)
```

EXAMPLE: FOR L = 5, T = 2*PI*SQRT(5/G) = 4.4375

A PLE DEVICE WITH 5 INPUTS CAN BE USED TO CALCULATE THE PERIOD OF OSCILLATION FOR AMPLITUDES UP TO L = 32 METERS. PLE DEVICE WITH MORE INPUTS SHOULD BE USED TO CALCULATE LARGER PERIODS OF OSCILLATION.

THIS EXAMPLE DEMONSTRATES HOW EASY IT IS TO CONSTRUCT LOOK-UP TABLES FOR COMPLEX ARITHMETIC FUNCTIONS USING PLE DEVICES




LENGTH OF PENDULUM
L
5
PERIOD OF OSCILLATION FOR A MATHEMATICAL PENDULUM LOOK-UP TABLE
8
T
PERIOD OF OSCILLATION IN SECONDS

TB01560M

```
PLE12P8                                   PLE CIRCUIT DESIGN SPECIFICATION
P5017                                                    FRANK LEE 10/14/83
ARITHMETIC LOGIC UNIT
MMI SANTA CLARA, CALIFORNIA
.ADD A3 A2 A1 A0 B3 B2 B1 B0 CIN I2 I1 I0
.DAT C3 C2 C1 C0 Z V C

;*************************************************
;*  THIS DESIGN IS NOT YET SUPPORTED BY PLEASM   *
;*************************************************

C,C3,C2,C1,C0 = /S2*/S1* S0*/A3,/A2,/A1,/A0         ;B - A - 1 + CIN
                    .+. B3, B2, B1, B0.+. CIN
              + /S2* S1*/S0* A3, A2, A1, A0          ;A - B - 1 + CIN
                    .+./B3,/B2,/B1,/B0.+. CIN
              + /S2* S1* S0* A3, A2, A1, A0          ;A + B + CIN
                    .+. B3, B2, B1, B0.+. CIN
              +  S2*/S1*/S0*/A3,/A2,/A1,/A0          ;A XOR B
                    :*: B3, B2, B1, B0
              +  S2*/S1* S0* A3, A2, A1, A0          ;A + B
                    +  S2*/S1* S0* B3, B2, B1, B0
              +  S2* S1*/S0* A3, A2, A1, A0          ;A * B
                    * B3, B2, B1, B0
              +  S2* S1* S0                          ;PRESET

V             =  C:+: C3                             ;OVERFLOW

Z             = /C3*/C2*/C1*/C0                      ;ZERO
```

DESCRIPTION

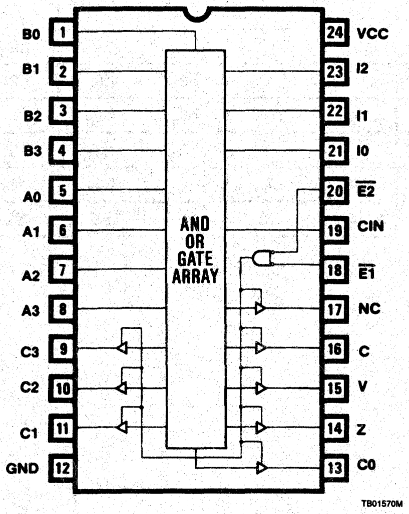
THIS ALU CAN PERFORM 8 FUNCTIONS ON TWO 4-BIT OPERANDS A (A3-A0) AND B (B3-B0) WITH CARRYIN (CIN) AND GIVES A 4-BIT RESULT C (C3-C0) WITH CARRYOUT (C). IT WILL ALSO GIVE STATUS AS OVERFLOW (V) AND ZERO (Z).

THE FUNCTION IS DETERMINED BY A 3-BIT FUNCTION SELECT CODE (S2-S0):

MODE	S2	S1	S0	FUNCTION
0	0	0	0	CLEAR
1	0	0	1	B - A - 1 + CIN
2	0	1	0	A - B - 1 + CIN
3	0	1	1	A + B + CIN
4	1	0	0	A XOR B
5	1	0	1	A + B
6	1	1	0	A * B
7	1	1	1	PRESET

ARITHMETIC LOGIC UNIT

PLE12P8



TB01570M

Wallace Tree Compression

In performing arithmetic calculations, it may happen that more than two numbers are to be added together. Adding two numbers can be achieved by using a simple adder. If there are more than two numbers to be summed, several levels of adders may be needed. This often causes too much delay.

An alternative is to use Wallace Tree Compression. Suppose there are m numbers each of n-bits wide. Summation over these numbers will range from 0 to m x ($2^n - 1$) which will take $\log_2[m(2^n - 1) + 1]$ bits (rounded UP to the nearest integer). For example, if there are five 2-bit numbers, i.e., m = 5, and n = 2, the sum will be bounded by 5 x ($2^2 - 1$) = 15 which will need a total of 4 bits.

One Wallace Tree Compression by itself will not be very useful. But consider if five 8-bit integers are added together. This technique enables vertical compression of these numbers in four groups. This type of vertical compression also eliminates the need of carry propagation. The five numbers are represented by:

A = (a7, a6, a5, a4, a3, a2, a1, a0)
B = (b7, b6, b5, b4, b3, b2, b1, b0)
C = (c7, c6, c5, c4, c3, c2, c1, c0)
D = (d7, d6, d5, d4, d3, d2, d1, d0)
E = (e7, e6, e5, e4, e3, e2, e1, e0)

where the 7th bits are the most significant; the calculation is as follows:

The groups are assigned as follows:

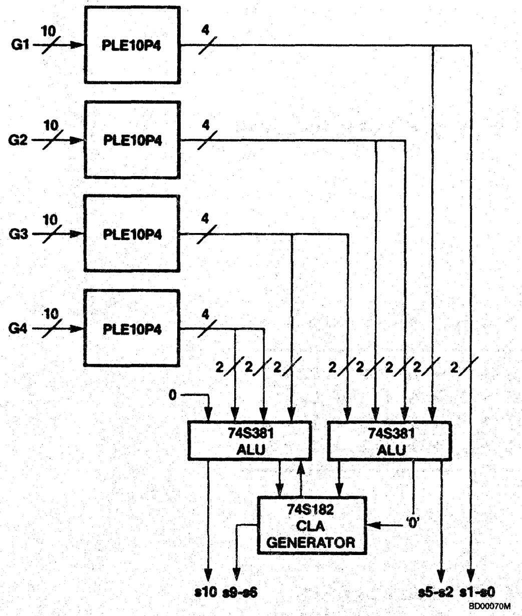
G1:(a0, a1, b0, b1,c0, c1, d0, d1, e0, e1)
G2:(a2, a3, b2, b3, c2, c3, d2, d3, e2, e3)
G3:(a4, a5, b4, b5, c4, c5, d4, d5, e4, e5)
G4:(a6, a7, b6, b7, c6, c7, d6, d7, e6, e7)

The above groups of bits can be compressed to:

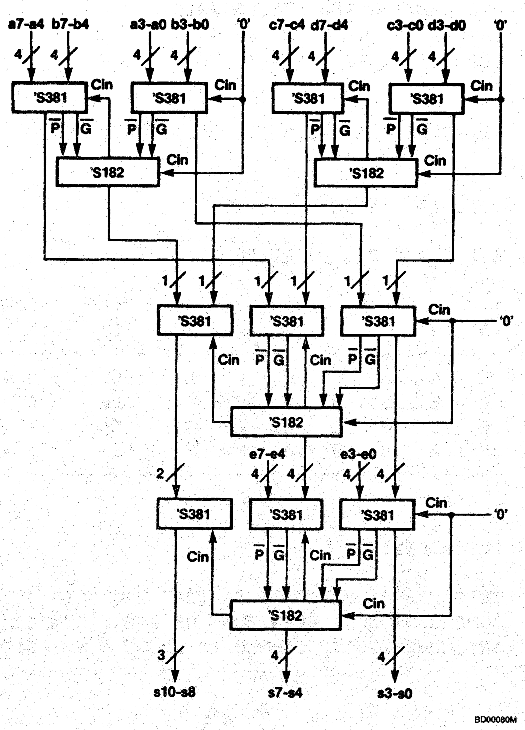
H1:($h1_3$, $h1_2$, $h1_1$, $h1_0$)
H2:($h2_3$, $h2_2$, $h2_1$, $h2_0$)
H3:($h3_3$, $h3_2$, $h3_1$, $h3_0$)
H4:($h4_3$, $h4_2$, $h4_1$, $h4_0$)

			G4		G3		G2		G1		
			a7	a6	a5	a4	a3	a2	a1	a0	= A
			b7	b6	b5	b4	b3	b2	b1	b0	= B
			c7	c6	c5	c4	c3	c2	c1	c0	= C
			d7	d6	d5	d4	d3	d2	d1	d0	= D
	+)		e7	e6	e5	e4	e3	e2	e1	e0	= E
							$h1_3$	$h1_2$	$h1_1$	$h1_0$	= H1
					$h2_3$	$h2_2$	$h2_1$	$h2_0$			= H2
			$h3_3$	$h3_2$	$h3_1$	$h3_0$					= H3
+)	$h4_3$	$h4_2$	$h4_1$	$h4_0$							= H4
			$h3_3$	$h3_2$	$h3_1$	$h3_0$	$h1_3$	$h1_2$	$h1_1$	$h1_0$	
+)	$h4_3$	$h4_2$	$h4_1$	$h4_0$	$h2_3$	$h2_2$	$h2_1$	$h2_0$			
S10	S9	S8	S7	S6	S5	S4	S3	S2	S1	S0	= result

S1 and S0 are just $h1_1$ and $h1_0$. S10-S2 can be obtained through addition of other bits. The hardware implementation is as follows:



It needs four PLE10P4 devices, two 74S381 ALUs and one 74S182. An alternative is using ten 74S381 ALUs and four 74S182 Carry Lookahead Generators.

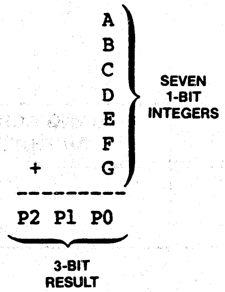


A comparison between the two architectures gives the following data:

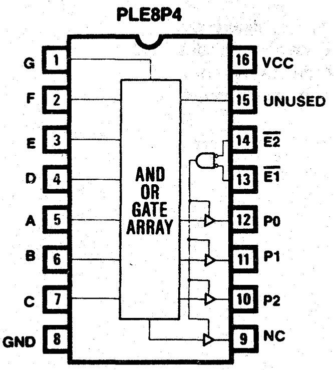
	USING WALLACE TREE COMPRESSION	USING CONVENTIONAL ARITHMETIC LOGIC
Delay (ns)	79	115
Number of components	7	14
Total number pins on the parts	128	264

Since Wallace tree compression can be of any configuration, there is no predefined part available. A PLE device provides an excellent solution. The designer may define his own configuration as long as it can be put in a commercially available PLE device.

```
PLE8P4                               PLE CIRCUIT DESIGN SPECIFICATION
P5019                                            VINCENT COLI 04/06/83
SEVEN 1-BIT INTEGER ROW PARTIAL PRODUCTS ADDER
MMI SANTA CLARA, CALIFORNIA
.ADD A  B  C  D  E  F  G
.DAT P0 P1 P2


P2,P1,P0 = A .+. B .+. C .+. D .+. E .+. F .+. G  ; P = A+B+C+D+E+F+G


FUNCTION TABLE

A B C D E F G P2 P1 P0

;                                    PPP    COMMENTS
;A   B   C   D   E   F   G           210    A + B + C + D + E + F + G = P
---------------------------------------------------------------------------
 L   L   L   L   L   L   L           LLL    0 + 0 + 0 + 0 + 0 + 0 + 0 = 0
 L   H   L   H   L   H   L           LHH    0 + 1 + 0 + 1 + 0 + 1 + 0 = 3
 H   L   H   L   H   L   H           HLL    1 + 0 + 1 + 0 + 1 + 0 + 1 = 4
 H   H   H   H   H   H   H           HHH    1 + 1 + 1 + 1 + 1 + 1 + 1 = 7
---------------------------------------------------------------------------


DESCRIPTION

THIS PLE8P4 PERFORMS PARTIAL PRODUCTS REDUCTION FOR WALLACE TREE
COMPRESSION.  SEVEN ROWS OF 1-BIT NUMBERS (A, B, C, D, E, F, AND G)
ARE NUMERICALLY SUMMED TO PRODUCE A 3-BIT RESULT (P2-P0).
```



SEVEN 1-BIT INTEGER ROW PARTIAL PRODUCTS ADDER



TB01580M

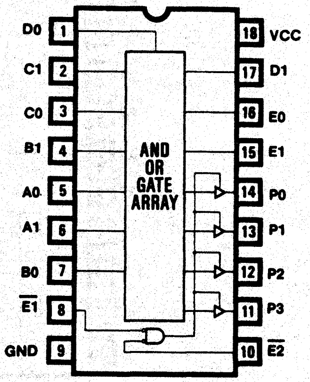
```
PLE10P4                              PLE CIRCUIT DESIGN SPECIFICATION
P5020                                           VINCENT COLI 08/22/83
FIVE 2-BIT INTEGER ROW PARTIAL PRODUCTS ADDER
MMI SANTA CLARA, CALIFORNIA
.ADD A0 A1 B0 B1 C0 C1 D0 D1 E0 E1
.DAT P0 P1 P2 P3


P3,P2,P1,P0 = A1,A0 .+. B1,B0 .+. C1,C0 .+. D1,D0 .+. E1,E0  ; P = A+B+C+D+E


FUNCTION TABLE

A1 A0 B1 B0 C1 C0 D1 D0 E1 E0 P3 P2 P1 P0

;AA   BB   CC   DD   EE     PPPP     COMMENTS
;10   10   10   10   10     3210     A + B + C + D + E = P
-----------------------------------------------------------
 LL   LL   LL   LL   LL     LLLL     0 + 0 + 0 + 0 + 0 =  0
 LH   LH   LH   LH   LH     LHLH     1 + 1 + 1 + 1 + 1 =  5
 HL   HL   HL   HL   HL     HLHL     2 + 2 + 2 + 2 + 2 = 10
 HH   HH   HH   HH   HH     HHHH     3 + 3 + 3 + 3 + 3 = 15
-----------------------------------------------------------


DESCRIPTION

THIS PLE10P4 PERFORMS PARTIAL PRODUCTS REDUCTION FOR WALLACE TREE
COMPRESSION.  FIVE ROWS OF 2-BIT NUMBERS (A1-A0, B1-B0, C1-C0,
D1-D0, AND E1-E0) ARE NUMERICALLY SUMMED TO PRODUCE A 4-BIT RESULT
(P3-P0).
```

$$\left.\begin{array}{rcc} & A1 & A0\\ & B1 & B0\\ & C1 & C0\\ & D1 & D0\\ + & E1 & E0\end{array}\right\}\text{FIVE 2-BIT INTEGERS}$$

$$\begin{array}{c}\text{------------}\\ \underbrace{P3\ P2\ P1\ P0}_{\text{4-BIT RESULT}}\end{array}$$

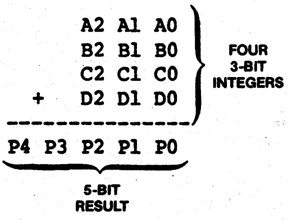
FIVE 2-BIT INTEGER ROW PARTIAL PRODUCTS ADDER

PLE10P4



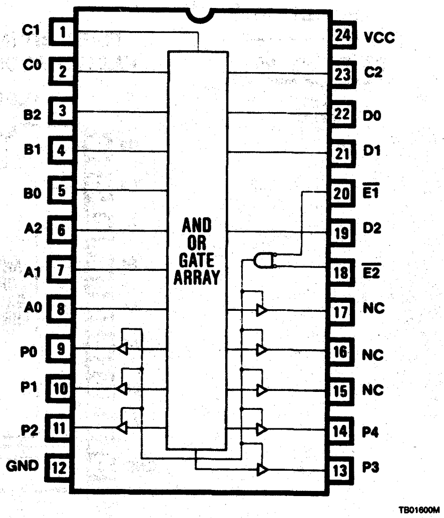
TB01590M

```
PLE12P8                                  PLE CIRCUIT DESIGN SPECIFICATION
P5021                                              VINCENT COLI 02/10/83
FOUR 3-BIT INTEGER ROW PARTIAL PRODUCTS ADDER
MMI SANTA CLARA, CALIFORNIA
.ADD A0 A1 A2 B0 B1 B2 C0 C1 C2 D0 D1 D2
.DAT P0 P1 P2 P3 P4

P4,P3,P2,P1,P0 = A2,A1,A0 .+. B2,B1,B0 .+. C2,C1,C0 .+. D2,D1,D0  ; P = A+B+C+D

FUNCTION TABLE

A2 A1 A0 B2 B1 B0 C2 C1 C0 D2 D1 D0 P4 P3 P2 P1 P0

;AAA   BBB   CCC   DDD   PPPPP    COMMENTS
;210   210   210   210   43210    A + B + C + D =  P
------------------------------------------------------
 LLL   LLL   LLL   LLL   LLLLL    0 + 0 + 0 + 0 =  0
 LLH   LLH   LLH   LLH   LLHLL    1 + 1 + 1 + 1 =  4
 LHL   LHL   LHL   LHL   LHLLL    2 + 2 + 2 + 2 =  8
 LHH   LHH   LHH   LHH   LHHLL    3 + 3 + 3 + 3 = 12
 HLL   HLL   HLL   HLL   HLLLL    4 + 4 + 4 + 4 = 16
 HHH   HHH   HHH   HHH   HHHLL    7 + 7 + 7 + 7 = 28
------------------------------------------------------

DESCRIPTION

THIS PLE12P8 PERFORMS PARTIAL PRODUCTS REDUCTION FOR WALLACE TREE
COMPRESSION.  FOUR ROWS OF 3-BIT NUMBERS (A2-A0, B2-B0, C2-C0, AND
D2-D0) ARE NUMERICALLY SUMMED TO PRODUCE A 5-BIT RESULT (P4-P0).
```



FOUR 3-BIT INTEGER ROW PARTIAL PRODUCTS ADDER

PLE12P8

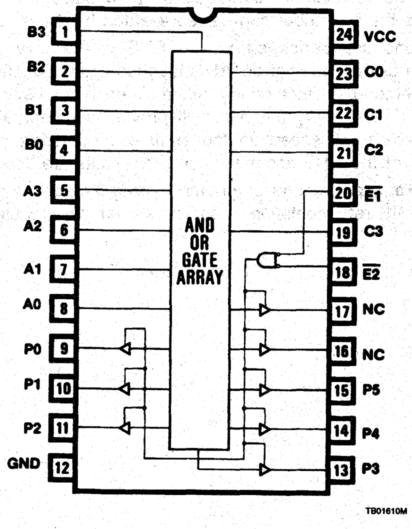


```
PLE12P8                                    PLE CIRCUIT DESIGN SPECIFICATION
P5022                                                 VINCENT COLI 08/10/83
THREE 4-BIT INTEGER ROW PARTIAL PRODUCTS ADDER
MMI SANTA CLARA, CALIFORNIA
.ADD A0 A1 A2 A3 B0 B1 B2 B3 C0 C1 C2 C3
.DAT P0 P1 P2 P3 P4 P5


P5,P4,P3,P2,P1,P0 = A3,A2,A1,A0 .+. B3,B2,B1,B0 .+. C3,C2,C1,C0  ; P = A+B+C


FUNCTION TABLE

A3 A2 A1 A0 B3 B2 B1 B0 C3 C2 C1 C0 P5 P4 P3 P2 P1 P0

;AAAA    BBBB    CCCC    PPPPPP       COMMENTS
;3210    3210    3210    543210        A  +  B  +  C  =  P
----------------------------------------------------------------
 LLLL    LLLL    LLLL    LLLLLL        0  +  0  +  0  =  0
 LLLH    LLLH    LLLH    LLLLHH        1  +  1  +  1  =  3
 LLHL    LLHL    LLHL    LLLHHL        2  +  2  +  2  =  6
 LHLL    LHLL    LHLL    LLHHLL        4  +  4  +  4  = 12
 HLLL    HLLL    HLLL    LHHLLL        8  +  8  +  8  = 24
 HHHH    HHHH    HHHH    HLHHLH       15  + 15  + 15  = 45
----------------------------------------------------------------


DESCRIPTION

THIS PLE12P8 PERFORMS PARTIAL PRODUCTS REDUCTION FOR WALLACE TREE
COMPRESSION.  THREE ROWS OF 4-BIT NUMBERS (A3-A0, B3-B0, AND C3-C0)
ARE NUMERICALLY SUMMED TO PRODUCE A 6-BIT RESULT (P5-P0).
```

```
       A3 A2 A1 A0      THREE
       B3 B2 B1 B0      4-BIT
  +    C3 C2 C1 C0      INTEGERS
  --------------------
  P5 P4 P3 P2 P1 P0

        6-BIT
        RESULT
```

THREE 4-BIT INTEGER ROW PARTIAL PRODUCTS ADDER

PLE12P8



Residue Arithmetic using PLE Devices

Conventional binary arithmetic can be replaced by another kind of computational methodology known as the Residue Number System. The use of this system allows integer arithmetic to be performed by arrays of PLE devices. The idea of PLE devices as arithmetic elements is simply to store pre-computed values of the arithmetic operation in the PLE memory cells and to use the input variables to the arithmetic as addresses to the PLE devices. Since we are computing the results of the arithmetic operations, the same PLE device organization may be used for many different functions. As an example, a 256x8-bit PLE device can be used as a 4 x 4-bit binary multiplier, or a 4 + 4-bit binary adder with the output multiplied by any 3-bit constant. It is this flexibility which holds so much appeal for the use of PLE devices as computational elements.

Introduction

Arithmetic operations often involve carry propagation. This propagation causes too much delay for high-speed arithmetic. The Residue Number System (RNS) provides the required separation property needed for high-speed arithmetic. Each digit of the RNS representation is coded into a certain number of bits. In performing the basic operations of addition, subtraction, and multiplication, no information is required to be passed between the digits. Therefore, the number of bits required for representing each digit can be partitioned so that commercially available PLE devices can be used to implement the arithmetic.

Basics of the Residue Number System

In this section, the elements of performing arithmetic using the RNS are introduced. The mechanism of coding numbers, the method of performing arithmetic using the RNS, and finally conversion between binary and RNS are presented.

Coding of Residue Numbers

In principle, the coding of Residue Numbers is extremely simple. A residue digit is the remainder when the number to be coded is divided by another number (a modulus). As an example, the residue of 15 divided by a modulus 7 which yields 1 as the remainder can be represented by$|15|_7 = 1$.

If operations are performed on an RNS where only one modulus is used, it will not be advantageous against a simple binary scheme at all since no information is encoded. Only the encoding of the binary numbers will provide the separation property which will speed up the arithmetic operation. The advantage of the RNS accrues when more digits are used.

Another example of encoding a number using 3 moduli to give a 3-digit RNS representation is as follows: let the moduli be m1 = 3, m2 = 4, m3 = 5. The residues of X = 25 will be shown as xi where i = 1, 2, 3. Thus,

$$X1 = |25|_{m1} = |25|_3 = 1$$
$$X2 = |25|_{m2} = |25|_4 = 1$$
$$X3 = |25|_{m3} = |25| 5 = 0$$

In the RNS using the moduli 3, 4, 5, the number 25 is represented as (1, 1, 0).

The number of unique representations for a set of moduli is the Least Common Multiple (LCM) of the moduli. The most efficient set of moduli is one in which all moduli are pairwise relatively prime.

Tables 1 illustrates an example of a set of moduli (3, 4) which can represent 12 integers. Note that the representations of 0 and 12 are the same, since the representation repeats itself after 12 integers.

X	(3) x1	(4) x2
0	0	0
1	1	1
2	2	2
3	0	3
4	1	0
5	2	1
6	0	2
7	1	3
8	2	0
9	0	1
10	1	2
11	2	3
12	0	0

Table 1. Representation of 0 to 12 in RNS Using Moduli 3 and 4. The Representation Repeats Itself After 12 Integers

In table 2, (4, 6) is the set of moduli uses. Since 4 and 6 are not relatively prime, the number of integers that can be represented is not the product of 4 and 6, but instead is the LCM of 4 and 6 which is 12. The representation again repeats itself once every 12 integers.

X	(3) x1	(4) x2
0	0	0
1	1	1
2	2	2
3	3	3
4	0	4
5	1	5
6	2	0
7	3	1
8	0	2
9	1	3
10	2	4
11	3	5
12	0	0
13	1	1
14	2	2
15	3	3
16	0	4
17	1	5
18	2	0
19	3	1
20	0	2
21	1	3
22	2	4
23	3	5
24	0	0

Table 2. Representation of 0x24 for Moduli 4 and 6. Since 4 and 6 are Not Relatively Prime, and their LCM is Only 12, the Representation Again Repeats Itself Every 12 Integers

Negative numbers are formed in the same way negative numbers are formed in binary (two's complement) system. To form the two's complement of a number in binary, we subtract the number 2^B where B is the number of bits of the representation. In RNS, we subtract the RNS number from mi to form the negative. Table 1 can be rewritten as in table 3 for encoding of negative numbers.

X	(3) x1	(4) x2
0	0	0
1	1	1
2	2	2
3	0	3
4	1	0
5	2	1
6	0	2
5	1	3
4	2	0
3	0	1
2	1	2
1	2	3

Table 3. Representation of –6 to 5 in RNS using Moduli 3 and 4

Arithmetic Using the RNS

For two RNS numbers, X and Y, the result of the addition of the two numbers, Z, in RNS is given by:

$|x_i + y_i|_{m_i} = z_i$ for all of the RNS digits.

The same result is found of subtraction and multiplication. This means that arithmetic can be carried out between the same digits of the two numbers, X and Y, without interaction between adjacent digits. The arithmetic is therefore "carry-free". As an example, let us consider the following computation:

Z = (863 x 3942) + (–862 x 3942) = 3942

We only need sufficient dynamic range to represent the result; intermediate overflows can be ignored. Let us choose the following moduli for the RNS representation:

m1 = 7, m2 = 9, m3 = 11, m4 = 13

M = 9009

The above set can represent numbers in the range –4505 to 4504, and so this number range is sufficient for the calculation of this example. The computation is shown in table 4.

X	(7) x1	(9) x2	(11) x3	(13) x4	
3942	1	0	4	3	
863	2	8	5	5	
862	1	7	4	4	
–862	6	2	7	9	
863 x 3942	2	0	9	2	
–862 x 3942	6	0	6	1	
Z	1	0	4	3	= 3942

Table 4. Calculating Z = 863 x 3942 + (–862 x 3942) = 3942

Division and Scaling

Division of residue numbers is more complicated than addition, subtraction, or multiplication. If the dividend is exactly divisible by the divisor, the operation is easier. In this case, a division by a number is the same as a multiplication by the inverse of that number. The multiplication inverse of an integer X in modulo arithmetic can be found by finding the vector (d1,...,dn) which satisfies the following:

$|X.d|_{mi} = 1$

For example, 95 divided by 5 in moduli 2, 7 and 9 can be done by first finding the vectors representing 95 and the inverse of 5.

$|95|_2 = 1$

$|95|_7 = 4$

$|95|_9 = 5$

So, for the multiplicative inverse of 5, we have:

$|1/5|_2 = 1 \qquad |5 \times 1|_2 = 1$

$|1/5|_7 = 3$ since $|5 \times 3|_7 = 1$

$|1/5|_9 = 2 \qquad |5 \times 2|_9 = 1$

Therefore,

$|95/5|_2 = |\,|95|_2\,|_2 \times |1/5|_2\,|_2 = |1 \times 1|_2 = 1$

$|95/5|_7 = |\,|95|_7 \times |1/5|_7\,|_7 = |4 \times 3|_7 = 5$

$|95/5|_9 = |\,|95|_9 \times |1/5|_9\,|_9 = |5 \times 2|_9 = 1$

and the answer is 19.

The operation becomes more complicated when the dividend is not exactly divisible by the divisor or one of the moduli of the miltiplicative inverse does not exist, say, if the residue of the divisor for that modulus is 0. In this case, we need to obtain the remainder and then subtract the remainder from the dividend and then perform the division. The problem in finding the remainder seems to be the same as performing the division itself. However, this type of division can be done in a process called scaling, which will not be discussed in detail in this paper.

In spite of the improvements made in implementing scaling algorithms, scaling still represents a major effort in any calculation. It is advisable to use RNS only on systems where many arithmetic operations can be performed for each scaling operation.

A System Using an RNS

An RNS is very useful in systems which have predefined operations and dynamic ranges. Moreover, it can only operate on integers, or at most, block floating-point numbers. Since the RNS involves conversions between integers and their RNS representations, and conversions by themselves are already time-consuming, the problem to be solved in the RNS system should be operation intensive.

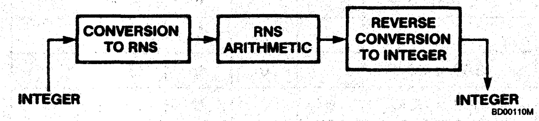


Figure 1. Architecture of an RNS

Conversion to RNS Representation

The conversion of an integer to RNS can be viewed as a mapping process. PLE devices provide a natural implementation for mapping. For example, if an 8-bit integer is used to represent numbers ranging from 0 to 255, and the following moduli are arbitrarily chosen for conversion to RNS — 2, 11 and 15 (which can represent 330 integers), 8 bits of address are needed for the integer input and 9 outputs (1 for modulus 2, 4 for modulus 11 and 4 for modulus 15). In reality, only 8 outputs are needed because that bit of residue for modulus 2 is not required, since the least significant bit of the integer is also the residue of itself in modulus. In fact, a PLE8P8 will be sufficient.

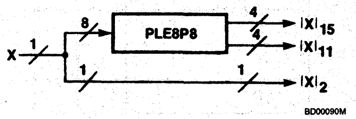


Figure 2. Mapping an 8-Bit Integer, X, to Its Residues on Moduli 2, 11 and 15

Another example is a 14-bit integer which is to be converted to RNS. A 14-bit address needs 16K address spaces for the mapping. 16K is too deep for a PLE device. An alternative is to use 4K-deep PLE devices. PLE12P4 and PLE12P8 devices and a selector (e.g., a PLE5P8 to control the PLE devices (See Figure 3)). The PLE5P8 device will decode two of the address bits and will selectively enable one of the four sets of PLE devices as the mapping set, thus deepening the effective address to 16K.

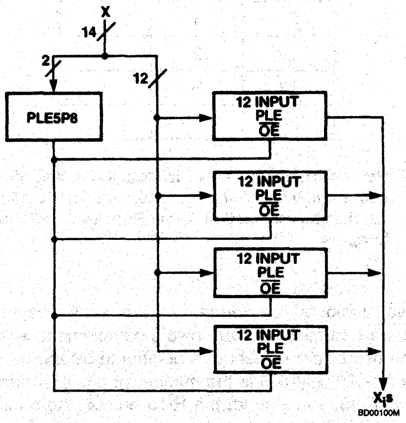


Figure 3. Mapping a 14-Bit Integer, X, to Its Residues by Selectively Enabling the Outputs of One of the Four Sets of 12-Input PLE Devices

This method of expansion is not effective with bigger integers. If the integer is N-bit and the PLE address space available is M-bit, then 2^{N-M} sets of PLE devices will be needed. Besides, as the dynamic range increases, the width of the outputs will also increase about proportionally. An alternative method is to use two or more levels of PLE devices to generate the residues. The first level generates the remainders from the more significant bits of the integer and the products of some of the moduli. These remainders are in turns concatenated with the rest of the bits to become the inputs to the second level PLE devices.

For example, for a 16-bit integer 43689, and let us use (2, 11, 13, 15, 23) as the set of moduli. We may choose 23, 30 and 143 as the moduli for the first level. The first level consists of PLE12P4s and PLE12P8s which generate the remainders of the most significant 12 bits of 43689 which is 2730. We know that $|2730|_{23}$ will be at most 22 and can therefore be represented by a 5-bit number; $|2730|_{15}$ will be at most 14 and can be represented by another 4-bit number; and $|2730|_{143}$ will be at most 142 and can be represented by a 6-bit number. The 5-bit number represented by $|2730|_{23}$ will be concatenated with the least significant 4 bits of the integer and gives a 9-bit number which can perform another division by 23 to give the final $|43689|_{23}$; the 4-bit number represented by $|2730|_{15}$ will be concatenated with the least significant 4 bits of the integer and gives an 8-bit number which can perform another division by 15 to give the final $|43689|_{15}$; the 6-bit number represented by $|2730|_{143}$ will be concatenated with the least significant 4 bits of the integer and gives a 10-bit number which can perform another division by 11 and 13 to give the final $|43689|_{11}$ and $|43689|_{13}$. As in the first example, $|43689|_{2}$ is just the least significant bit of the integer.

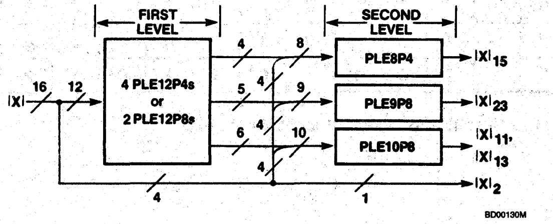


Figure 4. Mapping a 16-Bit Integer X to Residues in Modulo 2, 11, 13, 15, and 23 Using Two-Level Mapping. The First Level Gives Remainders from the More Significant Twelve Bits, While the Second Level Finds the Final Residues

In some circumstances, although an N-bit integer only has a dynamic range of 2^N, the intermediate calculations may overflow. It is sometimes necessary to add some other moduli to boost up the dynamic range for the intermediate calculations.

Arithmetic Operations In RNS

The arithmetic operations of the RNS is different from regular arithmetic in that even simple addition must be performed in modulo arithmetic. Simple ALU may not be able to handle this arithmetic. Again, PLE devices are proven to be most useful. A PLE8P4 device can perform addition, subtraction, or multiplication on two 4-bit residue numbers and give a 4-bit modulo result.

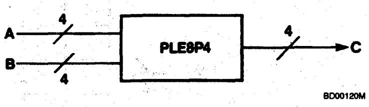


Figure 5. Calculating C = A + B, A - B, B - A x B Using PLE8P4

If the modulus is large, say greater than 64, the combined number of bits for two residues will be greater than the number of address bits for the largest of the commercially available PLE device. Of course, more than one device can be used to deepen the effective address space. In this case, for every additional bit of a modulus, two more bits of address will be needed — one for each operand. In other words, for each additional bit of a modulus the address space of operation will be quadrupled. It is not very effective when the modulus grows too large. Fortunately, for both addition and multiplication, there are more efficient procedures.

Large Modulus Addition

Table 5 shows the contents required for the addition operations in modulus 11. There is a lot of redundancy in the table which can be compressed by reducing what should be eight bits of inputs to five bits. What we need is just another level of mapping. There are a total of 121 combinations for a number of modulus 11 operating on another operand of the same modulus. In reality, only numbers ranging from 0 to 10 can be represented in modulus 11. The sum ranges from 0 to 20 (not in modulus 11). This range can be represented by a new set of submoduli (3, 7) which is five bits wide. In fact, any new set of submoduli which has a dynamic range of at least twenty-one can be used. The operands in modulus 11 will be converted to their representations in submoduli 3 and 7. The addition is done in the submoduli and the result is reconverted back to modulus 11 RNS (see Table 6).

x_{11} 4

y_{11} 4

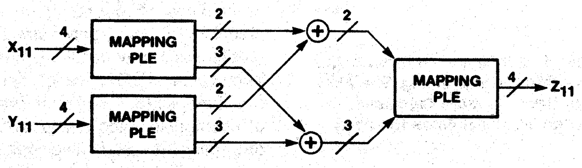
	0	1	2	3	4	5	6	7	8	9	10
0	0	1	2	3	4	5	6	7	8	9	10
1	1	2	3	4	5	6	7	8	9	10	0
2	2	3	4	5	6	7	8	9	10	0	1
3	3	4	5	6	7	8	9	10	0	1	2
4	4	5	6	7	8	9	10	0	1	2	3
5	5	6	7	8	9	10	0	1	2	3	4
6	6	7	8	9	10	0	1	2	3	4	5
7	7	8	9	10	0	1	2	3	4	5	6
8	8	9	10	0	1	2	3	4	5	6	7
9	9	10	0	1	2	3	4	5	6	7	8
10	10	0	1	2	3	4	5	6	7	8	9

TB01620M

Table 5. Addition Table in Modulo 11 Arithmetic

x + y	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
$\|x + y\|_{11}$	0	1	2	3	4	5	6	7	8	9	10	0	1	2	3	4	5	6	7	8	9
$\|x + y\|_{7}$	0	1	2	3	4	5	6	0	1	2	3	4	5	6	0	1	2	3	4	5	6
$\|x + y\|_{3}$	0	1	2	0	1	2	0	1	2	0	1	2	0	1	2	0	1	2	0	1	2

Table 6. Conversion Table Between Modulo 11 Arithmetic and Modulo 3 and 7 Arithmetic

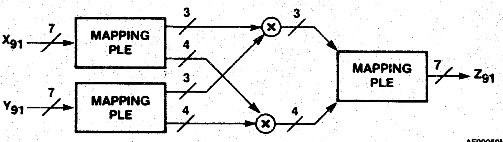


AF00030M

Figure 6. Calculating Addition of Two Numbers in Modulo 11 Using Submoduli Operations

Large Modulus Multiplication

The solution to this problem in multiplication is similar. For example, if two RNS digits in modulus 91 is to be multiplied, (7, 13) may be chosen as a set of submoduli. The representation of an RNS digit in modulus 91 needs 7-bits. These 7-bits are first mapped to two RNS digits — in modulo 7 which needs 3-bits; and in modulo 13 which needs 4-bits. The representations of the two operands in the two moduli can then be multiplied and give the result in modulo 7 and modulo 13. The result is then converted back to modulo 91. Unfortunately, this scheme can be used only when the modulus can be expressed as a product to two integers which are relatively prime. But, in this case, the RNS digit may simply be represented as the residue of the two smaller integers instead of using them as submoduli.



AF00050M

Figure 7. Calculating Multiplication of Two Numbers in Modulo 91 Using Submoduli Operations

Suppose another modulus 101 is used. 101 is a prime number and RNS in modulus 101 ranges from 0 to 100. The real dynamic range of the product of two numbers in modulus 101 is 0 to 10000, which is already too large for a PLE address space. For this modulus, we may use three 4K-deep PLE devices to deepen the address space. For a modulus like 1001, it may not be too efficient to use this scheme. Instead, since:

$$xy = [(x + y)^2 - (x - y)^2]/4$$

$$\text{or} \quad = [x + y)^2]/4 - [(x - y)^2]/4.$$

we may do x + y and x – y first and then do the squaring of the sum and the difference scaled by a factor of 4. Since the final product of two integers must be an integer, the squaring and scaling may be performed in one operation with the fractional part discarded. The way to obtain x + y and x – y is the same as what was discussed earlier in the "Large Modulo Addition" session.

In any event, operations on residues of large moduli are slower and involve more hardware and are not recommended.

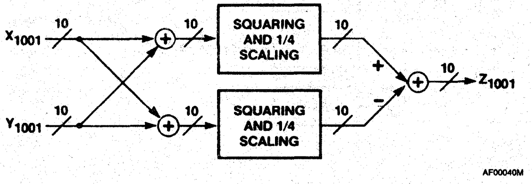


Figure 8. Performing Modulo 1001 Multiplication

The Reverse Conversion

The reverse mapping from RNS to integer is not as straightforward as the other way. For an RNS system which has a total of twelve bits for all the residues, we can still use 12-input PLE devices to convert. We may also use several sets of 12-input PLE devices to reverse map the RNS if the integer is not much longer. But for very long integers, we may need to use the general algorithm for the reverse map:

1. Find M = m1 x m2 x ...xm_{n-1} (where n is the number of moduli)
2. Find ti = M/mi
3. Find X = |x1t1 + x2t2 + ... + xn – 1tn – 1| M

In hardware implementation, ti's are all known beforehand. We can map xi's to get the xiti's.Then we may perform Wallace Tree Compression (see the session on this subject in this handbook for more information) on the xiti's to give two-level operands which add to the final sum and divide it by M to get X. Again, PLE devices provide the best solution for Wallace Tree Compression.

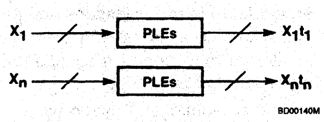


Figure 9a. Reverse Mapping to Get X_it_i

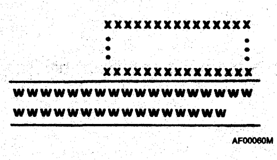


Figure 9b. Modulo M Wallace Tree Compression to Reduce the Number of Levels for Summation to 2 Followed by an Addition and Division to get $X = |x_1t_1 + ... + x_nt_n| M$

Conclusion

Memory elements provide excellent solutions to mapping functions — for control purposes, for arithmetic operations and general logic replacements. This paper investigates the possibility of using PLE devices as arithmetic units. In fact, for logic like residue number arithmetic, there is no better solution than to use these devices.

Acknowledgement

Portions of this article were extracted from *"Integer Arithmetic Using PROMs"* by Dr. G.A. Jullien of the University of Windsor, Canada.

Distributed Arithmetic Using PLE Devices

In digital signal processing, sum-of-product type of operations are often necessary. These operations take the form of:

$$y = \sum_{i=0}^{M} a_i x_i \quad \text{where } a_i\text{'s are some constants}$$

If real multiplications are to be performed on every product term, it will need a total of M multiplications and M-1 additions. Multiplication operations normally take much longer than simple addition. An alternative to calculate equations of the above form is by using distributed arithmetic.

Suppose there is an N-bit integer X given by:

$$x = [x(N-1), x(N\text{-}2), \ldots, x(1), x(0)]$$

or equivalently:

$$x = \underset{j=0}{\overset{N-1}{Z}} x(j)\, 2^j$$

where x(N − 1) is the most significant bit. The equation:

$$y = \sum_{i=0}^{M} a_i x_i$$

can be expressed as:

$$\begin{aligned} y &= \sum_{i=1}^{M} a_j \left(\sum_{i=0}^{N-1} x_i(j)\, 2^j \right) \\ &= \sum_{j=0}^{N-1} 2^j \left(\sum_{i=1}^{M} a_i x_i(j) \right) \end{aligned}$$

Now, let:

$$H(j) = \sum_{i=1}^{M} a_i x_i(j)$$

Since H (j) is independent of i and since a_1's are all constants, we precompute for every x(j) = [$x_1(j)$, $x_2(j)$,..., $x_M(j)$] the values [$x_1(j)$, $x_2(j)$,..., $x_M(j)$] the values of H (j). Then x(j) can be used as the address of PLE devices whose outputs are the precomputed result H(j).

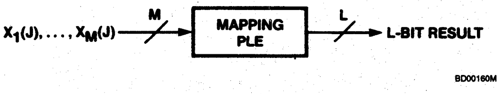


Figure 1. Mapping the j^{th} Bit from Each of the x_i's to An L-bit Result

If there are M bits of data and the result is L-bit wide, and if M is very large, say 20, and L is 8, then we need 20 bits of address lines if we want to use only PLE mapping. Since 20 bits of address translate to 1M words, and there is no available 1M-deep PLE device on the market, it is not realistic to use PLE mapping. Instead, H (j) can be partitioned as follows:

$$\begin{aligned} H(j) &= \sum_{i=1}^{20} a_i x_i(j) \qquad \text{for } M = 20 \\ &= \sum_{1=1}^{10} a_i x_i(j) + \sum_{i=11}^{20} a_i x_i(j) \end{aligned}$$

the 20-bit address can be separated to two 10-bit addresses and each of them is individually mapped. The two outputs will then be added together to give H (J). An implementation of this mapping is shown in figure 2.

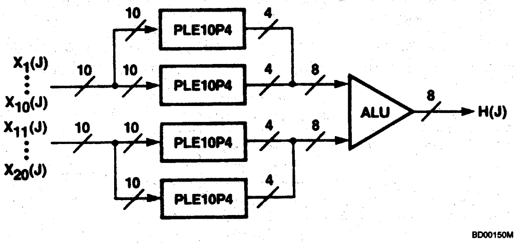


Figure 2. Mapping the j^{th} Bit of Each of x_i's to an L-bit Result When There Are Too Many x's (20 in This Case)

There is another alternative for implementing a sum-of-product operation: by using a multiplier accumulator (MAC).

The main constraint on distributed arithmetic is that one set of the multiplicands must be fixed, i.e. ai's in this case, for the sum-of-product mapping while a MAC will allow flexibility.

There are normally some constraints on the width for the data bus from which the operands are loaded. If all the operands are new, it will need M cycles to load in the operands anyway, distributed arithmetic offers no advantages over MAC since distributed arithmetic needs to wait for all the operands to be loaded in before any operation can start while MAC can perform a multiplication and an addition every cycle. M cycles will be needed anyway for the complete operation using a MAC while distributed arithmetic may take even longer.

On the other hand, for operations like convolutions where one set of operands are fixed and only one new variable operand is needed for every result, distributed arithmetic will be a better solution since it can give a result in every clock-cycle while a MAC will need M-cycles (because recalculations of all the product terms are necessary). An implementation for convolution is shown in Figure 3.

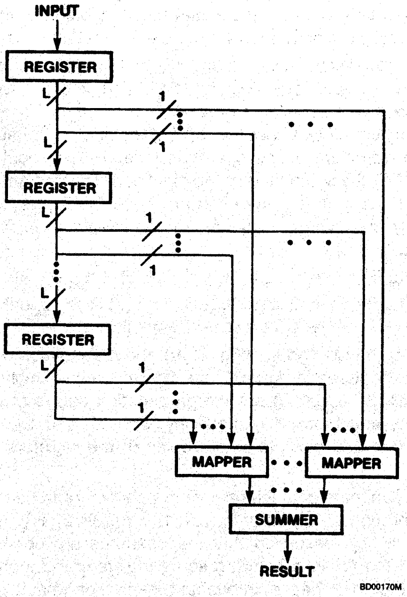


Figure 3. An Implementation of a Distributed Arithmetic System for Convolution

There is another way to implement distributed arithmetic through bit-serialization:

From H (j), the sum-of-product of y can be obtained as:

$$y = \sum_{j=0}^{N-1} 2^j J(j)$$

To implement this equation, consider that the least significant bit of the result is to be used only for rounding purposes only. Only the more significant bits will be retained. The computation can be performed in the following way:

1) For j = 0,
$y_0 = 2^0 H(0) = H(0)$

2) For j = 1 to N − 1
$y_j = H(j) + 1/2 H(j-1)$

Note that the second term of the last equation means that the previous result (y_{j-1}) is shifted right one-bit; the last bit of y_{j-1} is truncated.

The implementation of such a system is shown in Figure 3. The system consists of a shift register, a mapper (PLE circuits, or PLE circuits with adders), an accumulator, and an ALU.

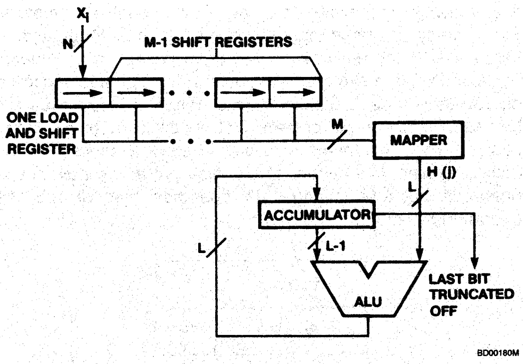


Figure 4. A Bit-Serialization Implementation for a Distributed Arithmetic System

The operations are as follows:

1) Load x_i onto the load and shift register at clock 0.
2) Load H (0) onto accumulator and shift all registers at clock 1.
3) From clock k (between clock 2 to clock N − 1), the content of the accumulator will be replaced by the sum of H (k-1) and the more significant N − 1 bits of the current accumulator value.
4) For clock N, the following are performed:
   a) Repeat step 3. At the end of the operation, the accumulator contains the value of the result (scaled by the number of shifts)

b) x_{i+1} is loaded onto the load and shift register. The shifting frequency is equal to N times the basic rate.

Due to the fact that there are a number of shift operations necessary for each data load, this method is recommended for the following conditions:

1) This design is under cost, power dissipation, and board space constraints.
2) This design is for high M-to-N-ratio array multiplications.

Registered PLE Devices in Pipelined Arithmetic

PLE devices are useful as logic elements, and registered PLEs are excellent media for pipelined arithmetic. Monolithic Memories supplies a number of registered PLE devices which provide effective solutions to pipelined systems.

A data processing system may have fall-through architecture. Since many of these operations may take a long time, it happens that the devices are not often tied up in operations. For example, in a system as in figure 1, the operations can be divided into three functional blocks. When the operands are loaded in, block 1 will operate first, followed by block 2 and then by block 3. When the data is in block 2, block 1 is not doing anything. We cannot at this time put in the next set of operands because changes in operands may disturb the operation in block 2.

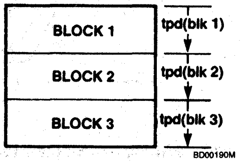


Figure 1. An Example of the Fall-Through Approach to Arithmetic Operation

A solution to this is by registering the operands and signal paths when the operations is switched to block 2; and by registering the operands and signal paths again when the operations is carried out in block 3. The result is stated in figure 2. This architecture is called the pipelined structure. It makes the loading of the second set of operands possible even before the first result is out, thus increasing the throughput.

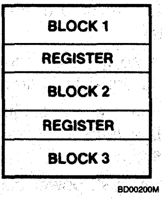


Figure 2. Pipelined Arithmetic Operation

The introduction of the registers for the pipeline increases the operation time of every block due to the addition of the setup times and the clock to output delays. The result is as follows:

1) Overall delay. The architecture in Figure 2 will need at least an additional 2 setup time and 2 clock to output delays of a register. In real, it will be more, because the minimum clock period will be determined by the sum of (i) the maximum of the operation times of individual blocks, and (ii) the setup time of the pipelined registers and (iii) the clock to output delay of the pipelined registers. Symbolically, the overall delays for the architectures in Figures 1 and 2 are:

$$t_{pd}(\text{Fig.1}) = t_{pd}(\text{blk } 1) + t_{pd}(\text{blk } 2) + t_{pd}(\text{blk } 3)$$

$$t_{pd}(\text{Fig.2}) = 2x\{\max[t_{pd}(\text{blk } 1), t_{pd}(\text{blk } 2), t_{pd}(\text{blk } 3)] + t_{su} + t_{clk}\} + t_{pd}(\text{blk } 3)$$

Where t_{pd}(Fig.1) and t_{pd} (Fig. 2) are the propagation delays of the architectures in figure 1 and figure 2 respectively; t_{pd}(blk 1) t_{pd}(blk 2), t_{pd}(blk 3) are the propagation delays of block 1, block 2, and block 3 respectively; and t_{su} and t_{clk} are the setup time and clock-to-output delay of the registers respectively.

2) Throughputs of clock rate. The architecture in figure 1 has a throughput period of (t_{pd}(blk 1) + t_{pd}(blk 2) + t_{pd}(blk 3) + t_{su} + t_{clk}), assuming that the operands are coming from and the result is going to some registers; the architecture in Figure 2 has a throughput period of (max[t_{pd}(blk 1), t_{pd}(blk 2), t_{pd}(blk 3)] + t_{su} + t_{clk}) which is faster.

PLE devices are useful as logic elements, and registered PLE devices are excellent media for pipelined arithmetic. Monolithic Memories supplies a number of registered PLE devices which provide effective solutions to pipelined systems.

Applications for pipeline arithmetic include array and digital signal processing.

Table of Contents

Article Reprints

Testing Your PAL Devices

Manouchehr Vafai

Introduction

The advantage of Programmable Array Logic (PAL®) circuits as a basic building block of digital system is now well established.

PAL circuits are a unified group of devices which combine programmable flexibility with high speed and extensive selection of interface options.

The architecture of PAL circuits consists of programmable-AND-OR gate arrays, output-registers and I/O feedback as shown in Figure 1.

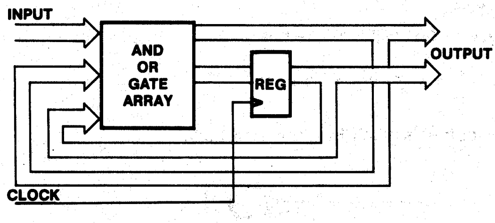


Figure 1. PAL Circuit Architecture

The increased system speed, reduced chip count and availability of a CAD tool called PALASM™ software should leave no doubt for design engineers that they have made a right choice in choosing PAL circuits.

The HAL circuit family is the masked program version of a PAL circuit. HAL® circuits will provide the users a cost-effective solution for large quantities and is unique in that it is a gate array with a programmable prototype.

The following steps are required when designing with PAL circuits.

- Familiarity with Demorgan's law.
- Familiarity with the Karnaugh maps.
- Ability to express logic equation in Sum-of-Product form.
- Ability to write simple seed vector for function table.
- Familiarity with different PAL circuits.

Programming PAL Circuits

PAL circuits will be programmed using PALASM software.

PALASM software is the CAD tool developed by Monolithic Memories to facilitate the process of programming. PALASM software is a Fortran IV program which assembles and simulates PAL circuits design specifications. It generates PAL circuit fuse patterns in formats compatible with PAL circuits or PROM programmers.

Besides generating PAL circuit fuse pattern in different programming formats, PALASM software does the following:

- Assembles PAL circuit design specification and reports error messages.
- Simulates the Function Table.
- Tests each product term for Stuck at zero (SA0) and stuck at one (SA1) faults.

The purpose of writing vectors is to prove that a device is capable of performing it's function before it is put in a system. PALASM software will exercise the vectors and will report any discrepancy. Writing vectors will raise confidence that a device will function properly at least in the design level. The simulator also transfers the function table vectors to a set of universal test vectors which may be used for functional testing after the device is programmed.

When a new system is transferred to production, the system designer hands over the responsibility for the system to the test engineering department, who now determines how and what test should be performed to ensure proper operation of the system. At this point the system designer transmits the necessary information for understanding the system operation. Unfortunately, much information is lost at this point. Test engineers usually have a hard time understanding how the system works with insufficient information. It is the design engineer who best knows the operation of his PAL circuit design, and it is the design engineer who can quickly specify a few seed vectors to give the test a starting point in solving the future problem.

Design for Testability

In short the only way to control a digital circuit is to apply a known value to it's input. Fault simulation has been the best technique of yielding a quantitive measure of test effectiveness. Fault simulation will test stuck-at-0 (SA0) and stuck-at-1 (SA1) of input and output lines. By generating test vectors that will test for each product term for (SA0) and (SA1) faults, then by observing the corresponding output and comparing it with the fault-free output, one can conclude whether a fault can be detected or not.

Consider the following example:

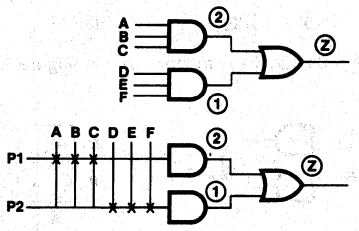


Figure 2. Logic Diagram and It's PAL Circuit Implementation

A B C D E F Z
1 1 1 0 X X 1 (vector-1)

The (vector-1) selects a product term P1. Under a fault-free condition, the output (2) will be high (we can observe this); however, under a fault condition the output will be low. In other words, one can conclude that either product term (P1) is (SA0) or outputs Z (Figure 2.) are (SA0).

Now consider vector 2.

A B C D E F G
0 0 0 0 0 0 0 (vector-2)

As it can be seen that both of the product terms are low, if the observed output is high, one can conclude that either product terms or outputs are Stuck-at-one.

Fault simulation grading is used by Monolithic Memories to evaluate candidates design for transfering from a PAL circuit to a HAL circuit.

In designing with PAL circuits, four different cases should be considered.

1. A purely combinational circuit where output is function of input.
2. A purely combinational circuit where output is function of input and feedback from output.
3. A purely sequential logic where output is function of input and feedback from output.
4. A combinational-sequential logic where output is function of input, feedback from combinational output and feedback from sequential output.

In cases 1 and 2 we can define a structured way of writing function table. Cases 3 and 4, on the other hand, because of dependency of the device on the previous state of the device, impose a relatively more sophisticated scheme of testing strategy.

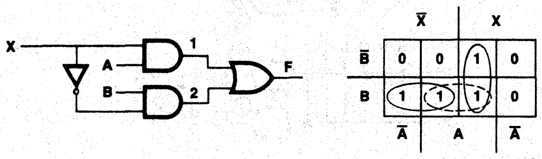
In the following examples the various techniques which might be helpful in testability of PALs, will be discussed.

Example 1: Glitch-free and Testable

Suppose we want to implement (EQ-1) using any of the combinational PAL.

$$F = X{*}A + \bar{X}{*}B \qquad \text{(eq-1)}$$

The K-MAP and logic diagram are shown in (Figure 3.)



	$\bar{X}$		X	
$\bar{B}$	0	0	1	0
B	1	1	1	0
	$\bar{A}$	A		$\bar{A}$

Figure 3. Logic Diagram and It's K-Map

The above logic is testable because we have full control over each node for (SA0) and (SA1) test.

The implementation using PAL circuits is as follows:

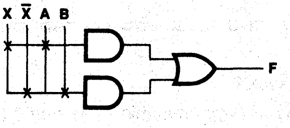


Figure 4. PAL Circuit Implementation of the Logic

Ideally the output should always be high if both inputs are high. The circuit is not glitch-free, the output might momentarily drop to low if we change the state of X, due to propagation delay between X and $\bar{X}$.

The problem will be solved by including a redundant (AB) term to (eq-2).

The equation will look like this.

$$F = XA + \bar{X}B + AB \qquad \text{(eq-2)}$$

The output is glitch-free, but untestable!

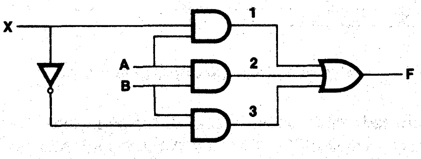


Figure 5. A Glitch-free Circuit

Node (2) is not Observable for (SA0). One can not force node (2) to one and keep node (1) and (3) in the low state. So the redundant product term is untestable.

This circuit can be made testable by the addition of control signal (Y) as follows:

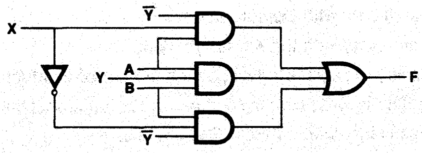


Figure 6. Glitch-free and Testable Logic

Now the logic is glitch-free and testable.

Example 2: Untestable Logic — A Simple Example

The logic F := $\bar{F}$ is untestable

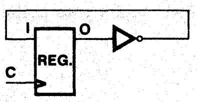


Figure 7. Implementation of F := $\bar{F}$

The initial state of the oscillator is unknown; this system can be made testable as follows:

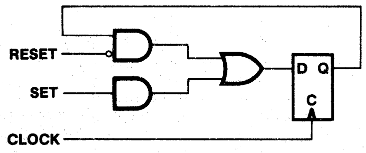


Figure 8. Implementation of F := $\overline{F}$ with SET and RESET

It has been done by addition of two control signals (RESET and SET) and one extra product term.

Illegal States

Upon power-up the initial state of output registers are unknown; this might force the device into one of the "illegal states".

The design engineer should be worried about the illegal state at design time. For example let's look at modulo-6 state machine

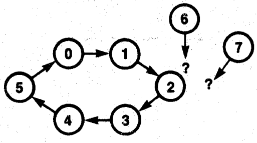


Figure 9. State Transition Diagram for a Modulo-6 Counter

The design engineer might ignore the other possible state (6, 7) but his ignorance might be costly at test time. If upon power-up the machine starts at either of (6) or (7) state, there is no way to control the state-machine. The best solution is to force both of the illegal states into one of the known states.

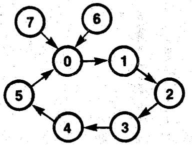


Figure 10. A Modulo 5 Counter with No Illegal State

Example 3: Design "pitfall" Case One

Consider the implementation of the following example

$$Q1 := I1 * Q1$$

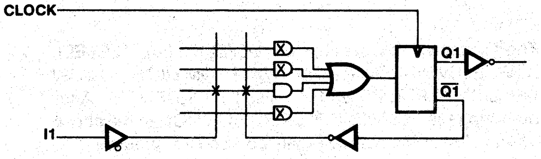


Figure 11. Implementation of Q1 := I1 * Q1 Using a PAL

If Q1 falls to zero, it will stay there forever. The logic needs a control signal for output reset.

Example 4: Design "Pitfall" Case Two

Consider the implementation of the following equation:

$$\overline{Q0} = \overline{A}*\overline{Q1}*Q0 + A*Q1*\overline{Q0} + \overline{Q0}$$

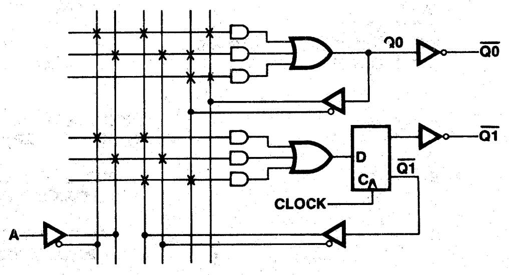


Figure 12. Implementation of $\overline{Q0} = \overline{A}*\overline{Q1}*Q0 + A*Q1*\overline{Q0} + \overline{Q0}$

If $\overline{Q0}$ goes to one it will stay there forever, the logic needs a control signal to clear it's output.

Hard Array Logic (HAL) Devices

The HAL device is the Hard Array version of a PAL device.

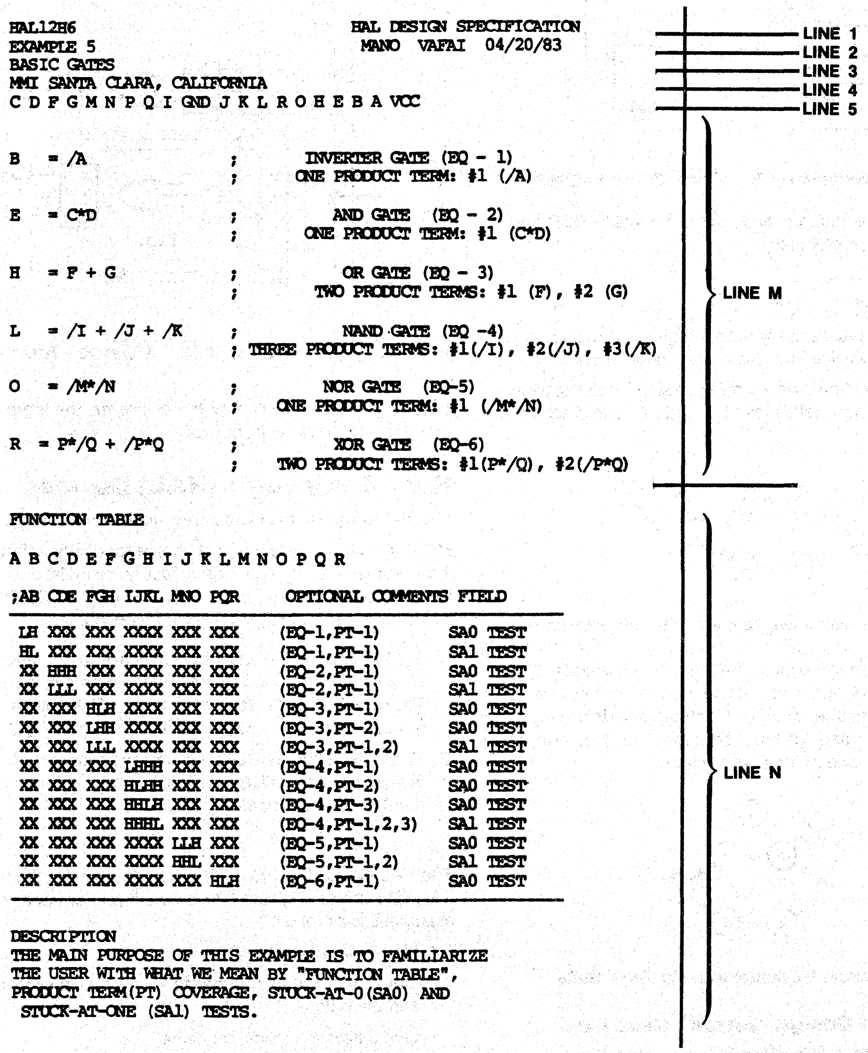
HAL logic circuits are the best choice for designs that are firm and volumes are large enough to justify the initial cost. Besides having Boolean equation in PAL DESIGN SPECIFICATION format the user should provide the following.

1. A FUNCTION TABLE which gives enough information about the operation of the device. Normally this FUNCTION TABLE shall test a minimum of 50% "Stuck at fault" grading using PALASM or TEGAS fault grading test.
2. The FUNCTION TABLE shall be constructed such that the device may be initilized to a known state within a specified number of steps (or clocks).

The HAL CIRCUIT SPECIFICATION is the input file used with PALASM software for the HAL's. The input format as shown in example 5 is as follow:

- Line 1 HAL circuit part number
- Line 2 user's part number followed by originator's name and the date
- Line 3 device application name
- Line 4 user's company name, city, state
- Line 5 pin list which is a sequence of symbolic names separated by one or more spaces. All pins including VCC and GND must be named
- Line M the logic equation which are used to generate metal masks from the provided equations

PAL®, and HAL® are Registered Trademarks of Monolithic Memories



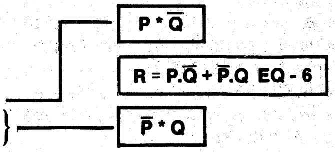
```
HAL12H6                               HAL DESIGN SPECIFICATION
EXAMPLE 5                              MANO  VAFAI  04/20/83
BASIC GATES
MMI SANTA CLARA, CALIFORNIA
C D F G M N P Q I GND J K L R O H E B A VCC

B   = /A               ;      INVERTER GATE (EQ - 1)
                       ;    ONE PRODUCT TERM: #1 (/A)

E   = C*D              ;        AND GATE  (EQ - 2)
                       ;     ONE PRODUCT TERM: #1 (C*D)

H   = F + G            ;         OR GATE (EQ - 3)
                       ;     TWO PRODUCT TERMS: #1 (F), #2 (G)

L   = /I + /J + /K     ;         NAND GATE (EQ -4)
                       ; THREE PRODUCT TERMS: #1(/I), #2(/J), #3(/K)

O   = /M*/N            ;        NOR GATE  (EQ-5)
                       ;   ONE PRODUCT TERM: #1 (/M*/N)

R  = P*/Q + /P*Q       ;         XOR GATE  (EQ-6)
                       ;   TWO PRODUCT TERMS: #1(P*/Q), #2(/P*Q)

FUNCTION TABLE

A B C D E F G H I J K L M N O P Q R

;AB CDE FGH IJKL MNO PQR      OPTIONAL COMMENTS FIELD
--------------------------------------------------------------
 LH XXX XXX XXXX XXX XXX     (EQ-1,PT-1)          SA0 TEST
 HL XXX XXX XXXX XXX XXX     (EQ-1,PT-1)          SA1 TEST
 XX HHH XXX XXXX XXX XXX     (EQ-2,PT-1)          SA0 TEST
 XX LLL XXX XXXX XXX XXX     (EQ-2,PT-1)          SA1 TEST
 XX XXX HLH XXXX XXX XXX     (EQ-3,PT-1)          SA0 TEST
 XX XXX LHH XXXX XXX XXX     (EQ-3,PT-2)          SA0 TEST
 XX XXX LLL XXXX XXX XXX     (EQ-3,PT-1,2)        SA1 TEST
 XX XXX XXX LHHH XXX XXX     (EQ-4,PT-1)          SA0 TEST
 XX XXX XXX HLHH XXX XXX     (EQ-4,PT-2)          SA0 TEST
 XX XXX XXX HHLH XXX XXX     (EQ-4,PT-3)          SA0 TEST
 XX XXX XXX HHHL XXX XXX     (EQ-4,PT-1,2,3)      SA1 TEST
 XX XXX XXX XXXX LLH XXX     (EQ-5,PT-1)          SA0 TEST
 XX XXX XXX XXXX HHL XXX     (EQ-5,PT-1,2)        SA1 TEST
 XX XXX XXX XXXX XXX HLH     (EQ-6,PT-1)          SA0 TEST
--------------------------------------------------------------

DESCRIPTION
THE MAIN PURPOSE OF THIS EXAMPLE IS TO FAMILIARIZE
THE USER WITH WHAT WE MEAN BY "FUNCTION TABLE",
PRODUCT TERM(PT) COVERAGE, STUCK-AT-0(SA0) AND
 STUCK-AT-ONE (SA1) TESTS.
```

EXAMPLE 5

- Line N the function table which begins with the key word "FUNCTION TABLE." It's followed by a pin list which may be in a different order and polarity from the pin list in line 5. VCC and GND cannot be listed. The pin list is followed by dashed line; e.g.;_ _ _ _ which in turn is followed by a list of vectors, one vector per line. One state must be specified for each pin name and optionally separated by spaces. A vector is a sequence of states listed in the same order as the pin list and followed by an optional comment.

The allowable states are H (HIGH LEVEL), L (LOW LEVEL), X (IRRELEVENT), C (TRANSITION FROM HIGH TO LOW OR CLOCK PULSE) and Z (HIGH IMPEDENCE). After preparing the PAL DESIGN SPECIFICATION in the above format, PALASM software can be used to simulate and perform fault testing.

```
BASIC GATES

   1 XXXXXXXXXXXXXXXXXH01
   2 XXXXXXXXXXXXXXXXXL11
   3 11XXXXXXXXXXXXXXHXX1
   4 00XXXXXXXXXXXXXXLXX1
   5 XX10XXXXXXXXXXXHXXX1
   6 XX01XXXXXXXXXXXHXXX1
   7 XX00XXXXXXXXXXXLXXX1
   8 XXXXXXXX0X11HXXXXXX1
   9 XXXXXXXX1X01HXXXXXX1
  10 XXXXXXXX1X10HXXXXXX1
  11 XXXXXXXX1X11LXXXXXX1
  12 XXXX00XXXXXXXXHXXXX1
  13 XXXX11XXXXXXXXLXXXX1
  14 XXXXXX10XXXXXHXXXXX1

PASS SIMULATION
 PRODUCT:    1 OF EQUATION.  6 UNTESTED(SA1)FAULT
 PRODUCT:    2 OF EQUATION.  6 UNTESTED(SA1)FAULT
 PRODUCT:    2 OF EQUATION.  6 UNTESTED(SA0)FAULT

NUMBER OF STUCK AT ONE  (SA1)  FAULTS ARE =  8

NUMBER OF STUCK AT ZERO (SA0) FAULTS ARE =  9

PRODUCT  TERM   COVERAGE                   = 85%
```



FAULT-TESTING

The following information is reported to the user

- — Total number of SA1 Faults. (8 in example 5)
- — Total number of detected SA0 faults. (9 in example 5)
- — $\dfrac{\text{SA1 faults + SA0 faults}}{\text{2 * total number of product terms}}$ * 100 % $\left(\dfrac{8+9}{2*10}\right.$ *100% = 85% ex-5)
- — One vector may detect more than one SA0 OR SA1 FAULTS (vector # 11 in example 5)
- — The user is reported with a message which tells him the product term for which it was not tested. (PRODUCT TERM 1 & 2-EQ 6, in example 5)

The following vectors can be added to the function table in example 5 in order to achieve 100% fault coverage.

```
 AB CDE FGH IJKL MNO PQR            COMMENTS (EXAMPLE 5)
 XX XXX XXX XXXX XXX LHH            (EQ-6,PT2)      SA0 TEST
 XX XXX XXX XXXX XXX HHL            (EQ-6,PT-1,2)  SA1 TEST
```

PALASM™ software has tested the above function table for example 5, the result is as follows:

```
 BASIC GATES
 1 XXXXXXXXXXXXXXXXXH01
 2 XXXXXXXXXXXXXXXXXL11
 3 11XXXXXXXXXXXXXXHXX1
 4 00XXXXXXXXXXXXXXLXX1
 5 XX10XXXXXXXXXXXHXXX1
 6 XX01XXXXXXXXXXXHXXX1
 7 XX00XXXXXXXXXXXLXXX1
 8 XXXXXXXX0X11HXXXXXX1
 9 XXXXXXXX1X01HXXXXXX1
10 XXXXXXXX1X10HXXXXXX1
11 XXXXXXXX1X11LXXXXXX1
12 XXXX00XXXXXXXXHXXXX1
13 XXXX11XXXXXXXXLXXXX1
14 XXXXXX10XXXXXHXXXXX1
15 XXXXXX01XXXXXHXXXXX1
16 XXXXXX11XXXXXLXXXXX1
PASS SIMULATION
NUMBER OF STUCK AT ONE  (SA1)  FAULTS ARE = 10
NUMBER OF STUCK AT ZERO (SA0) FAULTS ARE = 10
PRODUCT  TERM   COVERAGE                 =100%
```

PALASM™ is a trademark of Monolithic Memories.

PAL®20RA10 Design for Testability

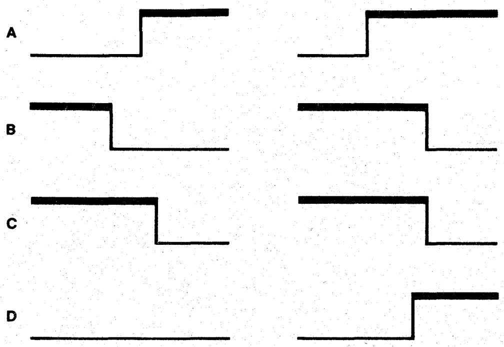
Edwin Young

This article is written to help customers of the PAL20RA1 recognize some fundamental design-for-testability issues which may arise due to the part's unique architecture. Customers should understand that these issues represent design criteria which Monolithic Memories will use to accept PAL20RA10 patterns for test generation/fault grading and for estimating the resource cost to test engineering if accepted. This article does not address the BUSINESS REQUIREMENTS such as the need for acceptable test vectors and the acceptability of a particular pattern for processing as a HAL® device.

The designer who wishes to use a 20RA10 in his/her design must bear in mind that although the part has preloadability, certain designs could diminish the effectiveness of this feature. The following rules are presented to help establish Test Engineering acceptance standards for the 20RA10. Additional general guidelines applicable are available in the PAL Handbook article reprint "Testing Your PAL Devices" by M. Vafai.

Avoid False Latching Situations

The equation D = (A*B) + (C+D) and its variants are susceptible to latching hazards since ATE may have considerable input skew. Of course, from a testing viewpoint, such implementations should be avoided. But if they must be implemented, care must be exercised in developing the function table so as to account for the possibility of latching. The designer must adopt and stick to some guideline such as "no more than one input undergoes a change in logic value per vector" when specifying the function table.

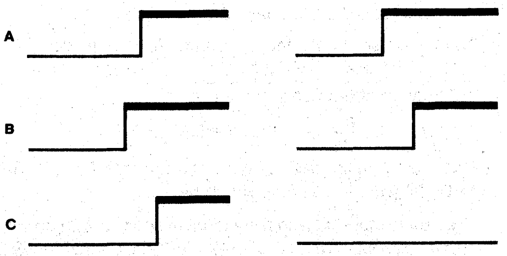


Assume A, B and C are primary inputs while D is a fed-back output. The waveforms to the left show two possible outcomes for output D depending on the skew of inputs A and B, which is a function of tester calibration.

The latch problem described is not unique to the 20RA10 but is clearly applicable to any PAL with asynchronous outputs with feedback (e.g., 16R4). The designer should realize, however, that false latching may occur on a 20RA10 even if all outputs are registered. Consider the equation set D:= C and D.CLKF = A*B for a simple registered 20RA10 output. The resulting waveforms would look similar to those of the previous asynchronous example. The important distinction here is that a 20RA10 has programmable asynchronous clocks rather than a single 'master clock' pin which can cause difficulties in testing.

Allow Data to Setup Prior to Clocking

The previous two pitfalls were examples of flaky latching due to glitches during testing. Consider the equation set C := B and C.CLKF = A for a registered output. The following example shows a definite positive latching...but of flaky (skewed) data.



Assume B is a primary input. Then the timing for situations at far left and left may be with A as feedback and as primary input respectively.

This example illustrates another aspect of the programmable asynchronous clock feature of the 20RA20: Clock pulses can have critical minimal or no delay relative to data setup time. Note that all other registered PAL devices have dedicated common clock pins to which delayed pulses are applied by the ATE to allow sufficient data setup time and ATE input skew.

Avoid Unreachable States

The 20RA10 may be preloaded to any state desired for testing purposes. Unfortunately, the desired state may not exist long enough for the simulator or ATE to use it. With all other preloadable PALs, any arbitrary state may be preloaded into the registers on a given test vector and the state will persist into the next vector providing the required conditions to detect some fault/s. This means all stuck-at-type faults *possibly* detectable can be detected. With the 20RA10, the preloaded state may feed back to assert state dependent resets or presets on one or more

PAL® and HAL® are registered trademarks of Monolithic Memories.

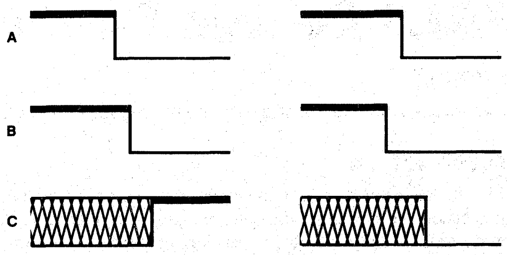
registers. Consequently, the desired state may last only a few nanoseconds after the preload vector is complete before changing to some new state. Since the desired state is not stable going into the vector immediately following the preload vector, the faults expected to be detected become non-detectable.

Another problem arises whenever output control logic is a function of state. In this case, assume the desired state for detecting faults is preload and is stable in the next vector. If this state provides the conditions necessary to detect faults and also disables the outputs, then the faults will be effectivley masked from detection.

Caution on Individual Register Bypass Mode

The 20RA10 allows the designer to permanently and independently bypass any register. Those registers not permanently bypassed may be bypassed under program control by setting both SET and RESET nodes to logic high. In this 'bypass mode', the register's D node is multiplexed to the output rather than its Q node. There is generally no test problem in going into bypass mode. The pitfall is in returning to 'register mode' operation, which only the 20RA10 can do. Consider the equation set C.RSTF = A and C.SETF = B of a simple registered output and the following possible waveforms. A race condition will occur to see whether set or reset operation prevails in going from bypass to register mode. There are two methods by which to get known states for testing purposes:

1) Clock a known value into the register on the next vector or;
2) Set RESET to logic low on one vector and then SET to low on the next or vice versa.



An indeterminate state on the output can occur if both primary inputs A and B go to logic low on the same vector.

PAL® Design Function and Test Vectors

E. Young

Introduction

This article was written to help customers understand the purpose of seed vectors and provide some general guidelines as to what elements are important in developing them. It is assumed that the reader has read the "PALASM™ Manual" and the "PAL® Handbook" article reprint *Testing Your PAL Devices.*

In general, PAL®/HAL® devices are required to provide a function table or "Seed Vectors" to Monolithic Memories in order to ensure that parts shipped have a high degree of reliability for the application intended. Ideally, these vectors should accomplish three objectives:

1) Initialize the PAL device preferably in the same way as in the actual system;
2) Exercise the customer's functions thoroughly, emulating actual system operation as closely as possible;
3) Provide a high degree of fault coverage.

Initialization

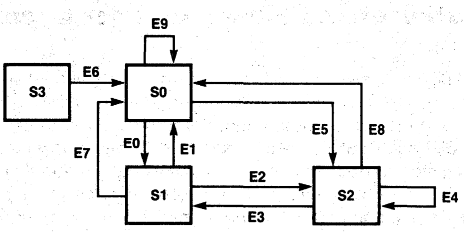
Seed vectors which initialize the PAL logic circuit consist of one or more vectors placed at the very beginning which will bring both combinatorial and registered outputs to a stable and known logic state (1 or 0). This is necessary in the system also so that its operation upon power-up is predictable. Furthermore, care should be taken to ensure that the initialization state is a legal state of the state machine for which the PAL device is intended.

Exercise Functions

The essential functions for which the PAL device was originally designed must be exercised fully. This will assure that the tested parts work the way they were intended to. In addition to essential "designed-for" functions, it is prudent to include general test exercises such as verifying that outputs don't change in the absence of clock pulses and checking to see that inputs in the "don't care" (X) state don't produce adverse responses. General test exercises help to reinforce the validity of a design and can uncover overlooked design errors. After a set of exercises has been decided upon, the next step is to write them in a format suitable for simulation purposes.

The designer may have originally defined the functions in terms of equations, state diagrams, truth tables, etc. Truth tables are readily reformatted to PALASM1 syntax "Function Tables" and exercises with the simulation option (code=S). State diagrams can be converted by expressing each state and input edge in binary vector format and sequencing them according to the diagram's flow. The customer should become thoroughly familiar with the syntax of the function table description (see the PALASM Manual for a detailed treatment of syntax) before attempting to translate truth tables, etc.

The following simple example demonstrates how exercising seed vectors might be derived from a designer's state diagram:



Assume the following state and edge definitions accompany the diagram:

	STATE 0 = LL		EDGE 0 = LL
	STATE 1 = LH		EDGE 1 = LH
	STATE 2 = HL		EDGE 2 = LL
(ILLEGAL)	STATE 3 = HH		EDGE 3 = LH
			EDGE 4 = LL
			EDGE 5 = HH
		(INITIALIZING)	EDGE 6 = HL
		(INITIALIZING)	EDGE 7 = HL
		(INITIALIZING)	EDGE 8 = HL
		(INITIALIZING)	EDGE 9 = HL

From the above information, it is possible to create the truth table for the diagram and then the function table representation:

EDGE	PRESENT STATE	NEXT STATE
AB	CD	CD
HL	XX	LL
HH	LL	HL
LL	HL	HL
LH	HL	LH
LH	LH	LL
LL	LL	LH
LL	LH	HL

TWX: 910-338-2376
2175 Mission College Blvd. Santa Clara, CA 95054-1592 Tel: (408) 970-9700 TWX: 910-338-2374
Monolithic Memories MMI

FUNCTION TABLE REPRESENTATION		
FUNCTION TABLE		
AB	CD	
HL	LL	INITIALIZE DEVICE
HH	HL	TEST EDGE 5
LL	HL	TEST EDGE 4
LH	LH	TEST EDGE 3
LH	LL	TEST EDGE 1
LL	LH	TEST EDGE 0
LL	HL	TEST EDGE 2

Fault Coverage

Another criterion for seed vector completeness is "fault coverage". Fault coverage is an empirical method and is more quantitative than functional exercising — indeed, no knowledge of the circuit's intended function is necessary or assumed (although it could help) while developing fault coverage vectors.

Fault coverage, being an empirical approach to determining a logic circuit's reliability, uses the concept of "failure models" to grade the effectiveness of a given set of test vectors. This is called "fault grading". In fault grading a set of vectors, a fault coverage value is calculated that is simply the ratio of detected faults to total faults expressed as a percentage.

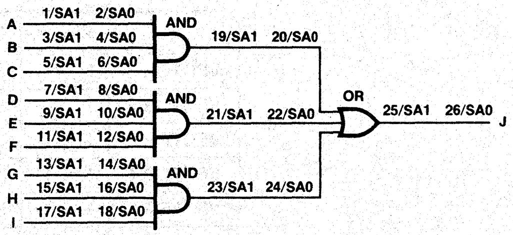
Test vectors may be graded against one or more failure models. Some well-known models include single stuck-at-1/stuck-at-0, pattern sensitivities, shorts and opens and multiple stuck-at models. Selection of a failure model (or models) for fault grading fundamentally depends on the model's empirical effectiveness for screening bad parts and will be affected by a number of factors including circuit technology and fault simulator capabilities.

The most common and primary fault coverage failure model considered by "TGEN" at Monolithic Memories is the classic single stuck-at-1/stuck-at-0 failure model. "TGEN" automatically appends test vectors which test for the following additional failure models where applicable: 1) Adjacencies, 2) Clock, 3) Tri-state.

"TGEN" has a specified minimum value of fault coverage for PAL and HAL devices based on the single stuck-at failure model. The minimum values are determined by current "TGEN" policy (see your FAE) and reflect the economic trade-off between acceptable levels of reliability and the cost of test generation for maximum coverage. PAL and HAL devices for which the specified minimum values cannot be attained will require the customer's written waiver for low coverage prior to production release of the pattern.

The fault coverage percentage determined by "TGEN" is different from the percentage determined by selecting the fault testing option (code=F) of PALASM1 software. In PALASM1 software fault coverage is based on product term coverage (PTC). PTC is still the ratio of detected to total faults except that "detected" and "total" fault sums refer to stuck-at faults on product term outputs only. PTC ignores stuck-at faults which occur anywhere else. A more accurate procedure is to calculate the coverage based on all the circuit nodes where a stuck-at condition may occur. When every node (fault site or wire) is considered, the coverage calculated correlates to the design's testability better and will generally be a much lower value than PTC. "TGEN" goes one step further in conservatism by calculating fault coverage on a "collapsed" fault basis. Fault collapsing simply divides all the stuck-at faults into groups such that, within a group, if one fault is detected, then all the others in the group are detected too. The advantage of collapsing it that only one representative fault in a group needs to be selected for test generation and if it is detected, then the other "equivalent" faults are detected by definition. This saves time and effort on test generation for equivalent faults. Calculations on a collapsed fault basis treat each group as one fault.

The following simplified example demonstrates the difference in fault coverage calculations using a collapsed fault list:



Note: This example is a simplified one for illustrative purposes only and does not show the effects of faults normally associated with input or output buffers. Also, some partial collapsing has already been done (i.e., input faults of "OR" gate are collapsed into output faults of "AND" gates).

Assume the above circuit is to be realized as a PAL or HAL device. Suppose some seed vectors are provided also, as shown here:

	A	B	C	D	E	F	G	H	I
VECTOR 1	H	H	H	L	L	L	L	L	L
VECTOR 2	L	L	L	H	H	H	L	L	L
VECTOR 3	L	L	L	L	L	L	H	H	H
VECTOR 4	L	L	L	L	L	L	L	L	L

The seed vectors on the previous page yield various values for fault coverages corresponding to the method of calculation as shown in the table below.

	PTC	EVERY NODE	COLLAPSED	
SUBSET OF TOTAL FAULTS CONSIDERED FOR CALCULATIONS THAT ARE DETECTED BY EACH VECTOR (SHOWN AT FAR RIGHT)	20	2,4,6,20,26	2	Vector 1
	22	8,10,12,22	8	Vector 2
	24	14,16,18,24	14	Vector 3
	19,21,23	19,21,23,25	19	Vector 4
TOTAL FAULTS CONSIDERED FOR CALCULATIONS	19,20, 21,22, 23,24	1,2,3,4,5,6,7,8,9,10,11, 12,13,14,15,16,17,18, 19,20,21,22,23,24,25,26	1,2,3,5,7, 8,9,11,13, 14,15,17,19	
PERCENT COVERAGE	6/6 = 100%	17/26 = 65%	4/13 = 31%	

As can be seen from the above example, given the same seed vectors, PALASM1 software would show 100% coverage whereas "TGEN" would show 31% coverage. Notice that the poor coverage by "TGEN" is due to none of the input nodes being tested for stuck-at-1. In most instances, a better set of test vectors can improve the coverage significantly. For the above example, the reader can verify that the following slight modification of the seed vectors would yield 100% coverage for all calculation methods:

	A	B	C	D	E	F	G	H	I
VECTOR 1	H	H	H	L	L	L	L	L	L
VECTOR 2	L	L	L	H	H	H	L	L	L
VECTOR 3	L	L	L	L	L	L	H	H	H
VECTOR 4	L	H	H	L	H	H	L	H	H
VECTOR 5	H	L	H	H	L	H	H	L	H
VECTOR 6	H	H	L	H	H	L	H	H	L

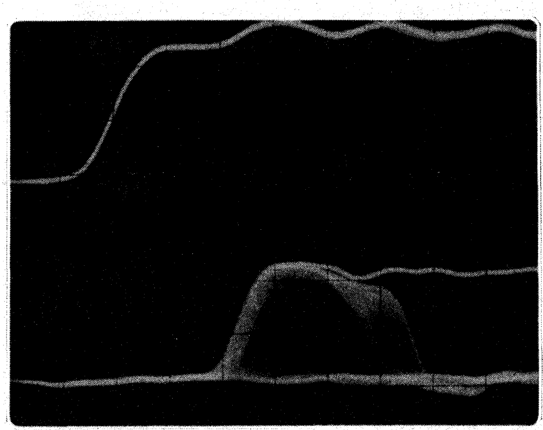
Testability

The previous sections described some essentials for comprehensive seed vector set. Variations on how fault coverage is calculated was covered also. However, no matter how it is calculated, fault coverage is only as good as the testability of the circuit permits. Using the stuck-at failure model, the customer must consider both absolute and practical fault coverages achievable for his PAL/HAL logic circuit design. Certain testability factors, such as redundancy, number of test points (outputs) or reconvergence, affect absolute (i.e., theoretical maximum) coverage. Other factors, including preloadable state machines, the amount of feedback and overall controllability, will affect practical coverage since many faults may be potentially detectable but uneconomical to detect due to excessive vectors or difficult to reach states. As testability is improved, absolute and practical fault coverage will usually increase.

METASTABILITY

A Study of the anomalous behavior of synchronizer circuits

Danesh M. Tavana



INTRODUCTION

This article will summarize the results of the studies performed on synchronizer circuits. The information presented may be used by system designers to gain insight into the anomalous behavior of edge-triggered flip-flops. Understanding flip-flop behavior and applying some simple design practices can result in an increased reliability of any system.

METASTABILITY

In the digital world a bit represents the fundamental unit of measure. The output state of any digital device is either "HIGH" (a voltage level above VIH) or "LOW" (a voltage level below VIL) as shown in figure 2. Under the proper operating conditions the register in figure 1 outputs a HIGH or a LOW on the rising edge of the clock within a nominal delay called the "clock to out" delay. If the setup and hold times are violated the register has a small probability of entering a third region of operation called the "metastable" state. Metastable is a Greek word meaning "in between" and it is a state between HIGH and LOW. Even though most synchronizers snap out of metastability in a short period of time, theoretically this state can persist indefinitely. Some of the registers built from older technologies had metastable states which lasted as long as a few microseconds. When the output of a device goes into metastability the clock to out delay will be grossly affected. This may alter the system's worst case propagation delay and potentially lead to a system crash!

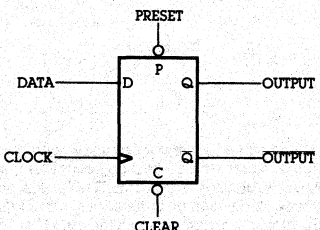


Figure 1

SYNCHRONIZERS

The design of a synchronous digital system is based on the assumption that the maximum propagation delay of a flip-flop and any other gates are known. A digital system is free of hazardous race conditions and timing anomalies if the maximum propagation delay in the system does not exceed the clock's period. In systems where an asynchronous input is interfaced with a clocked device such as a flip-flop, the maximum specified propagation delay of this device may no longer be valid if certain electrical parameters are violated. Computer peripherals, an operator's keyboard, or two independently clocked subsystems are instances where there is a possibility of interfacing an asynchronous input which will violate the synchronizer's electrical parameters.

A popular device typically used in synchronized systems is the edge-triggered register shown in figure 1. The edge-triggered register will properly synchronize the incoming data to the system's clock as long as its operating conditions are satisfied. Table 1 summarizes these specifications for Monolithic Memories Inc.'s (MMI) 74LS374 register. It is difficult to guarantee setup and hold time requirements when the data is asynchronously interfaced to a register. The violation of setup or hold time in a register has a probability of initiating a misbehavior termed "Metastability."

SYMBOL	PARAMETER	COMMERCIAL			UNIT
		MIN.	TYP.	MAX.	
V_{CC}	Supply Voltage	4.5	5	5.5	V
T_A	Operating free air temp.	0		75	°C
t_w	Width of clock	15			ns
t_{su}	Setup time	20			ns
t_h	Hold time	0			ns

Table 1

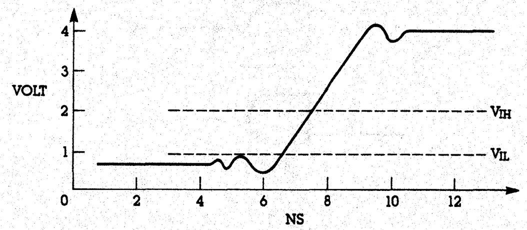


Figure 2

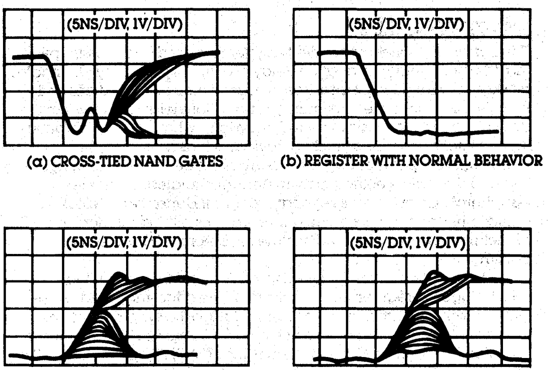
The diagrams in figure 3 illustrate some examples of waveforms in the metastable condition. From the waveforms it is evident that the outputs are distorted under metastable conditions. Figure 3d shows the output of a typical 74LS374 register manufactured by Monolithic Memories. Monolithic Memories family of bipolar devices exhibit superior metastable hardened performance due to their high speed bipolar technology and advance Schottky TTL circuit design techniques. Most of these devices typically snap out of metastability in a flashing 15 nanoseconds.

WHY THE SYNCHRONIZER FAILS

Before attempting to explain how the synchronizer's internal circuity fails let's take a look at an interesting problem.

PROBLEM: In the SR type latch shown in figure 4 what happens if the set (S) and the reset (R) inputs are simultaneously raised from a LOW voltage level to a HIGH level?

ANSWER: The outputs will be in a stable state of HIGH prior to the RS transition and will quickly oscillate to a final steady state of either HIGH or LOW (see figure 3a). To demonstrate this result the reader is encouraged to do this excercise either mentally or to actually build the circuit and view the output on the oscilloscope.



(a) CROSS-TIED NAND GATES

(b) REGISTER WITH NORMAL BEHAVIOR

(c) MONOLITHIC MEMORIES INC'S PAL16R4A

(d) MONOLITHIC MEMORIES INC'S 74LS374

Figure 3

Clock driven master-slave flip-flops contain the same type of cross tied RS latch within their internal circuitry. The NAND gate equivalent of the master-slave D type flip-flop is shown in figure 5. The gates circled in this figure can potentially behave similar to the above problem. If the clock and data are triggered within a specific window of one another the output may have an oscillatory behavior before settling down.

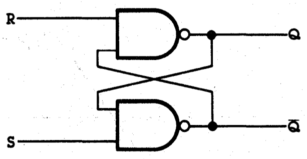
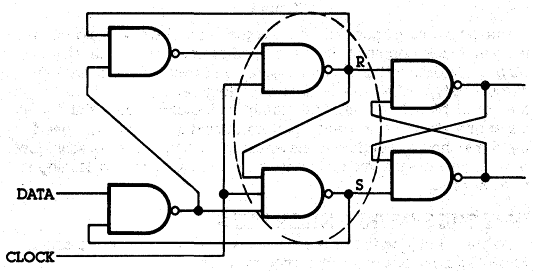


Figure 4



Cross tied RS latch structure is seen in the master-slave edge triggered flip-flop.

Figure 5

METASTABLE DETECTOR

This section will show how to characterize the behavior of an edge triggered flip-flop with an asynchronous data interface. If the setup and hold times of the flip-flop are satisfied the output behaves properly (figure 6a). One of the four possible events below can take place if the flip-flop goes metastable:

1) The output starts to make a transition but snaps back to its original state (figure 6b).

2) The output makes a complete transition but the maximum propagation delay of the device is exceeded (figure 6c).

3) The output starts oscillating and retains its present state (figure 6d).

4) The output oscillates to a new state (figure 6e).

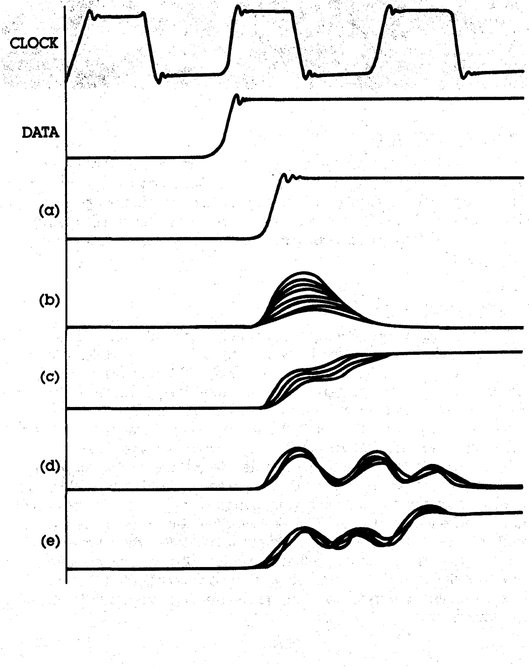
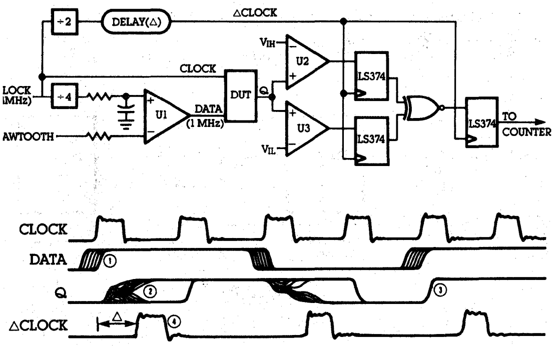


Figure 6

The circuit shown in figure 7 is used to obtain experimental results of a metastable device. The circuit can detect and count the number of events of metastability. The device under test (DUT) is forced into metastability by repeatedly sweeping the edges of the data past the rising edges of the clock. The modulation of the data is possible by using a comparator device (U1) along with an external sawtooth waveform. Thousands of transitions are created within the setup and hold time window of the DUT. Sweeping the data edges past the low to high clock transitions simulates an asynchonous input and increases the probability of getting a metastable failure on the output (Q) of the DUT.



① JITTER BAND SIMULATES AN ASYNCHRONOUS BEHAVIOR

② METASTABLE OUTPUT

③ PROPER OUTPUT WAVEFORM

④ ΔCLOCK SAMPLES THE OUTPUT (Q) AFTER A DELAY Δ.

Figure 7

If the output of the device goes into metastability it will be detected by the comparator pair (U2) and (U3). The comparators will have complementary outputs if the output (Q) of DUT is anywhere between VIH and VIL. The outputs of the comparators are latched by a delayed version of the clock (ΔClock). The EXCLUSIVE-NOR gate followed by the register signal the event of metastability to an external counter.

The variable delay (Δ) between the two clocks will sample the output at various locations on the time axis. As this delay is varied the event of metastability is sampled and counted at these locations by our circuit. Therefore the output of our circuit measures the rate of metastability versus time delay. The real behavior of a metastable output can thus be effectively characterized with this scheme, that is, we can determine the length of time a metastable condition will persist and the density distribution of the metastable event.

Three 74374 devices and four PAL devices are used in this experiment. The plots of metastable failure versus time are shown in figures 8a,b. The next section will discuss in detail the characteristics of these plots.

EXPERIMENTAL RESULTS

Various graphs of metastability failure rate versus delay time are illustrated in figure 8. We can conclude from these graphs that the rate of metastability failure decreases as the sample clock (ΔCLOCK) moves farther and farther away from the DUT clock. The pictures shown in figure 9 have captured repeated events of metastability on the oscilloscope.

Let's take a closer look at one of the graphs to examine the behavior of the device. The PAL16R4A-4 device exhibits one count per second if the delay (Δ) is 60 nanoseconds. As the delay (Δ) is decreased, the rate increases exponentially until the delay equals 32 ns at which point the rate flattens out and remains fixed. The 32 ns forms the knee of our graph and will be referred to as Δo. The rate will remain constant if the delay (Δ) is decreased past the knee of our graph. Further reduction in the delay will place the sampling clock's rising edge prior to data transitions and thus the error rate vanishes to zero. The time at which the rate goes to zero is marked with an (X) on the graphs. By using this time (X), and another location on the graph such as the time where only one error per second occurs, we can associate an approximate range of metastability for different devices. This range of metastability is referred to as the "mean time to snap out of metastability". From the graph it is evident that the mean time to snap out of metastability for the PAL16R4A-4 logic circuit is the difference between 60 ns and 25 ns which is 35 ns.

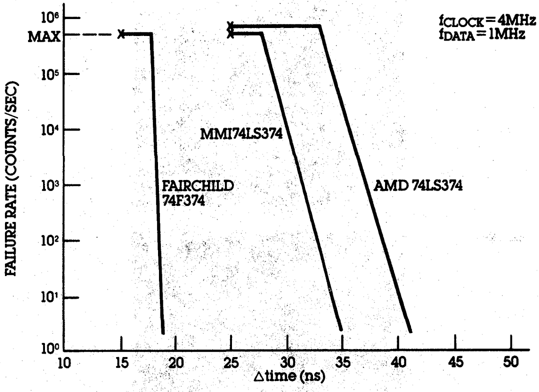


Figure 8a

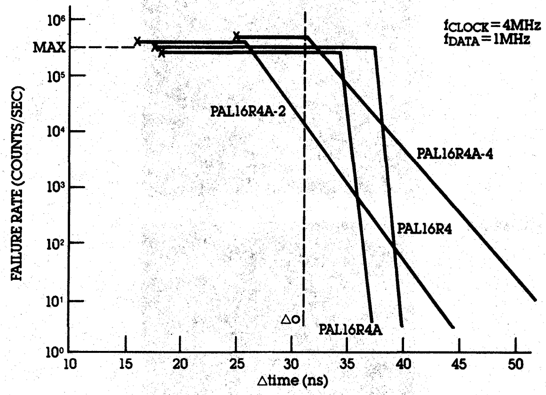


Figure 8b

All of the graphs illustrated can be quantified by an equation of the form:

$$\log \text{FAILURE} = \log \text{MAX} - b(\Delta - \Delta o)$$

Since a natural logarithm is a constant multiple of base 10 logarithm we can rewrite the above equation as:

$$a \cdot \ln \text{FAILURE} = a \cdot \ln \text{MAX} - b(\Delta - \Delta o)$$

In the above equation the MAX value is representative of the maximum metastability failure rate in our device. This MAX value is closely related to the frequency at which a metastable condition may occur in our device. The frequency at which metastability occurs is simply a constant multiple of the product of CLOCK and DATA frequency.

$$\text{MAX} = \text{K1} \cdot f_{\text{CLOCK}} \cdot f_{\text{DATA}}$$

Substituting this in our original equation we get:

$$a \cdot \ln \text{FAILURE} = a \cdot \ln(\text{K1} \cdot f_{\text{CLOCK}} \cdot f_{\text{DATA}}) - b(\Delta - \Delta o)$$

$$\ln \text{FAILURE} = \ln(\text{K1} \cdot f_{\text{CLOCK}} \cdot f_{\text{DATA}}) - b/a(\Delta - \Delta o)$$

$$\text{FAILURE} = (\text{K1} \cdot f_{\text{CLOCK}} \cdot f_{\text{DATA}})\, e^{-k2(\Delta - \Delta o)}$$

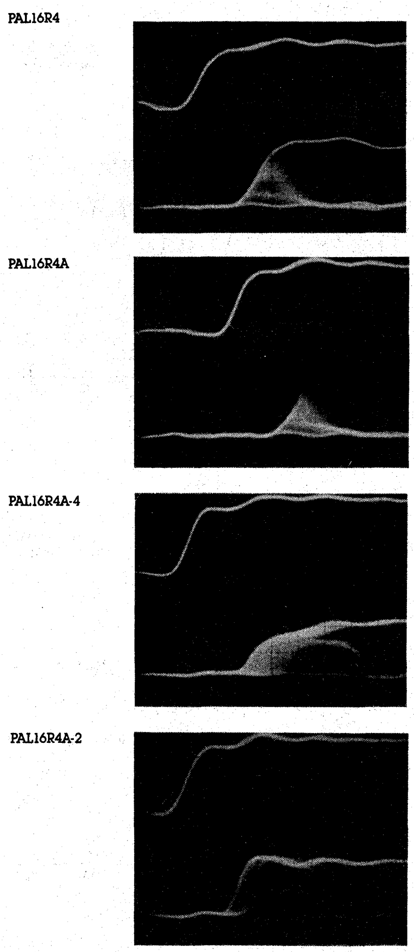


Figure 9 (2v/DIV, 5ns/DIV)

Table 2 gives the three important parameters which can be used by system designers to fully characterize the metastable behavior of the mentioned devices. These parameters can be obtained for different devices by duplicating this experiment. An example is given below to show how the information on table 2 may help the designer in the design of asynchronous systems.

MANUFACTURER	DEVICE	K_1 (Sec)	K_2 (ns^{-2})	Δo (ns)
MMI	PAL16R4	1×10^{-7}	4.3	37
	PAL16R4A	1×10^{-7}	4.3	34.5
	PAL16R4A-2	1×10^{-7}	.64	25
	PAL16R4A-4	1×10^{-7}	.5	31
	74LS374	2×10^{-7}	1.8	27.5
AMD	74LS374	2×10^{-7}	2.0	34.5
FAIRCHILD	74F374	2×10^{-7}	11.5	17.5

Table 2

EXAMPLE

For the hardware implementation in figure 10 determine the maximum clock frequency to give a typical error rate of one failure per year. We must choose the minimum period to give an error rate of less than one failure per year. From this result we can determine the maximum clock frequency. The time Δ in the equation below will determine the distance between clock edges. We must determine Δ from the equation by numerical extrapolation. The system clock's period can be represented as (Δ + Tcc + setup), or plugging in the numbers it is Δ+75.

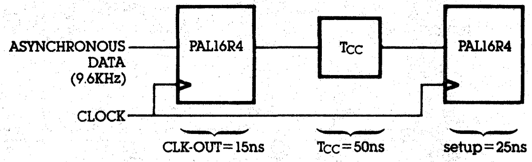


Figure 10

$$\text{FAILURE} = (\text{K1} \cdot f_{\text{CLOCK}} \cdot f_{\text{DATA}})\, e^{-\text{K2}(\Delta - \Delta o)}$$

and plugging in the appropriate values we have:

$$3.2\text{EE} - 8 = [(1\text{EE} - 7)\,(1/(\Delta + 75\text{ns}))\,(9600)]\, e^{-[(4.3)(\Delta - 37)]}$$

Solving for Δ, we see that it is approximately 43 nanoseconds. The system period is thus seen to be the sum of 43ns and 75ns or 118ns. The maximum clock frequency is the inverse of the period or approximately 8 MHz.

CONCLUSION

Synchronization of two independent pulse trains is possible through the use of edge triggered registers. The electrical characteristics of the flip-flop are affected when the setup and hold times of the device are violated. This misbehavior is termed "metastability" and its probability of occurrence can be derived for a given system. The factors which affect this probability and the length of time which a metastable condition persists are influenced by the technology of the device as well as by the circuit design techniques.

An important fact which needs to be stressed is that even if a register's output goes metastable, the system may not necessarily fail if the register snaps out in time to satisfy the system's worst case timing requirement. The following design practices are suggested when using synchronizers:

Try to minimize the number of locations where asynchronous signals enter your system.

Clocking the asynchronous inputs through two pipelined registers can greatly reduce the error rate.

Use a single clock within your local system environment. For multiple system clocks, derive all the clock signals from a single source to assure synchronization between different devices within the system.

When analyzing the worst case timing of your system, add the time to snap out of metastability to any register in an asynchronous data path.

A single PAL® with registers can be your best choice for state machine analysis of asynchronous events. As the registers have virtually identical setup times, the simultaneous observation of a metastable event by different register states are likely to be the same. Contrasted to a distributed system of observing register states with different setup times, the PAL system of register states with identical setup times is a superior synchronizer.

Avoid edge sensitive devices on the output paths of the registers which have asynchronous inputs. The glitch created when the synchronizer goes metastable is enough to trigger the edge sensitive device. The use of level sensitive devices is generally a better design practice.

PAL devices can be effective synchronizers where various registering schemes are easily implemented.

CP-141
Conference Proceedings

High-Speed Bipolar PROMs Find New Applications As Programmable Logic Elements*

Vincent J. Coli and Frank Lee

Classic applications for bipolar PROMs include instruction storage for microprogram control store and software for microprocessor programs. However, due to a new design methodology and state-of-the-art performance, PROMs are finding increasing numbers of applications as Programmable Logic Element (or PLE) devices. This paper will cover the architecture, applications, and software support for PLE devices.

* This paper is a slightly modified version of the paper by the same name which appeared in the Conference Proceedings of the 9th West Coast Computer Faire, pages 40-47; April 1984.

High-Speed Bipolar PROMs Find New Applications As Programmable Logic Elements

Vincent J. Coli and Frank Lee

Classic applications for bipolar PROMs include instruction storage for microprogram control store and software for microprocessor programs. However, due to a new design methodology and state-of-the-art performance, PROMs are finding increasing numbers of applications as Programmable Logic Element (or PLE) devices. This paper will cover the architecture, applications, and software support for PLE devices.

Fuse-Programmable Logic Families

A typical combinatorial Boolean equation can be written in sum-of-product form, which consists of several AND gates summed at an OR gate. In general, a set of combinatorial Boolean equations with n inputs (I0, I1, . . . , In–1) and m outputs (O0, O1, . . . , Om–1) can be generated through one level of AND gates followed by one level of OR gates. Custom logic functions can be defined using programmable logic.

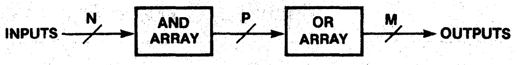


Figure 1. Structure of Programmable Logic Devices

Fuse-programmable devices normally consist of two levels of logic — AND-array and OR-array — as suggested above. There are three basic types of fuse-programmable devices — PROM (Programmable Read Only Memory), PLA (Programmable Logic Array), and PAL® (Programmable Array Logic) devices. Which arrays are fuse-programmable distinguish these three types of devices.

PLAs offer the greatest flexibility since both the AND and OR arrays are programmable. This flexibility comes with the cost of lower performance, higher power dissipation, and generally higher price.

A PAL device has only the AND-array programmable; the OR-array is fixed. Each output has an OR gate associated with it which sums a fixed number of product terms (AND combinations). Statistically there is only a limited number of product terms in any equation. So the flexibility of a PLA is normally not needed. This is a compromise between flexibility and cost and performance.

The OR-array is programmable in a PROM, but the fixed AND-array consists of all combinations of literals for each of the input variables. For example, there are 32 product terms available in a PROM with 5 inputs a,b,c,d,e (corresponding to words 0 through 31 in the PROM memory):

/a*/b*/c*/d*/e	(Word 0)
/a*/b*/c*/d* e	(Word 1)
/a*/b*/c* d*/e	(Word 2)
.	.
.	.
.	.
a* b* c*/d* e	(Word 29)
a* b* c* d*/e	(Word 30)
a* b* c* d* e	(Word 31)

where '*' represents the Boolean AND operator and '/' represents the Boolean NOT or inverter operator. The fuses in the OR-array are programmed to select the desired AND combinations.

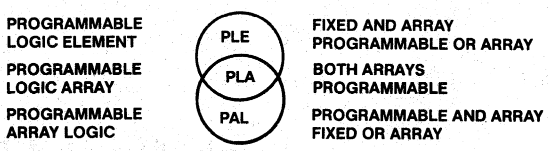

PROGRAMMABLE LOGIC ELEMENT
PROGRAMMABLE LOGIC ARRAY
PROGRAMMABLE ARRAY LOGIC
PLE
PLA
PAL
FIXED AND ARRAY PROGRAMMABLE OR ARRAY
BOTH ARRAYS PROGRAMMABLE
PROGRAMMABLE AND ARRAY FIXED OR ARRAY

Figure 2. Structural Difference Between PLE (PROM), PLA and PAL Devices. Note that the PAL and PLE Logic Circuits Complement Each Other. The PAL Device has Many Input Terms While the PLE Device is Rich in Product Terms

The existence of all combinations of literals for all inputs makes it possible to define functions which cannot be implemented in a PLA or a PAL device. For example, a 5-input Exclusive-OR (XOR) function can be implemented using sixteen product terms. This may exceed the number of product terms available in a PAL device and will consume too many product terms in a PLA, but can be constructed quite efficiently in a PROM. It is important to realize that any combination of inputs can be decoded in a PROM as long as sufficient input pins are provided since a PROM provides 2^n product terms (where n is the number of inputs). Another way of looking at this is that PROMs store the logic transfer function in a memory. The fixed AND-array (or AND-plane) consists fo the row and column decoders while the fuses in the OR-array (or OR-plane) are the bits in the memory. In a memory, a fuse blown versus a fuse intact distinguishes a HIGH from a LOW.

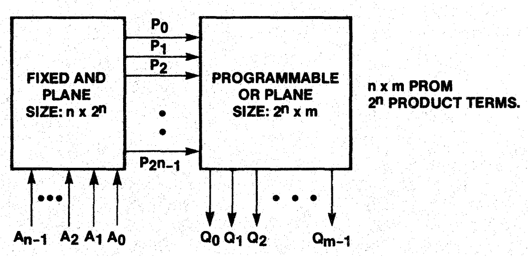


Figure 3. Block Diagram of a PROM Viewed as PLE Device. Notice that the PLE Provides Many (2^n, Where n is the Number of Inputs) Product Terms. A By-Product of this is Programmable Output Polarity: Either Active-High or Active-Low Output Polarities are Available

Due to this special characteristic of abundant product terms, PROMs are also often used as logic devices. In this paper, PROMs are referred to as PLE (Programmable Logic Element) devices.

Advantages of PLE Devices

PLE devices provide a cost-effective solution for many applications. Here are just some of the advantages of PLE devices:

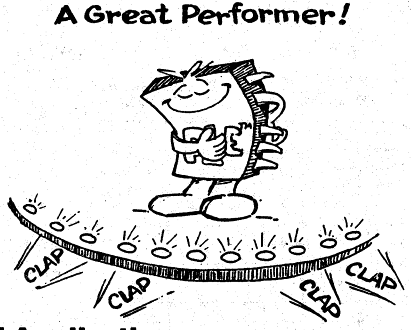
1) Customizable Logic — The designeer is limited to standard functions if SSI/MSI devices are used. The designer can create his own logic chips using PLE devices.

2) Design Flexibility — Modification of design is possible even without redesigning the PC board. For example, the address space of a microprocessor-based system can be reconfigured by merely programming a new device if the decoding is implemented in a PLE device. This feature comes in handy if you want to upgrade a system which originally used 64-K Dynamic RAMs to now use state-of-the-art 256-K Dynamic RAMs.

3) Reduce Errors — Errors are sometimes unavoidable and oftentimes quite expensive. Programmable devices make it easier and less expensive to correct errors.

4) Reduction of Printed Circuit Board Space—PLE devices save PC board space since several SSI/MSI functions can be integrated into a single package.

5) Fast Turnaround Time — With existing commercial programmers and development software support, a prototype of the custom tailored PLE device will be ready in just a few minutes.



PLE Applications

PLE applications include random logic replacement, decoder/encoders, code converters, custom ALUs, error detection and correction, look-up tables (both trigonometric and arithmetic), data scaling, compression arithmetic like Wallace Tree adders, distributed arithmetic, and residue arithmetic.

Several levels of random logic chips can be replaced by one PLE logic circuit. As discussed earlier, PLE devices can implement logic in sum of products form.

Despite the existence of dedicated encoders and decoders, many of these functions are application dependent. A standard 3-to-8 decoder/demultiplexer (74S138) can be used in decoding applications. But the decoding scheme may require several 3-to-8 decoder/demultiplexers and additional SSI OR-gates. On the other hand, a PLE device can be customized to perform the required decoding function with no additional gates. Simple decoders, such as those used for decoding memory chip selects from address lines, can be implemented in a PLE device with five to ten inputs. More complex decoding may require eight to twelve inputs.

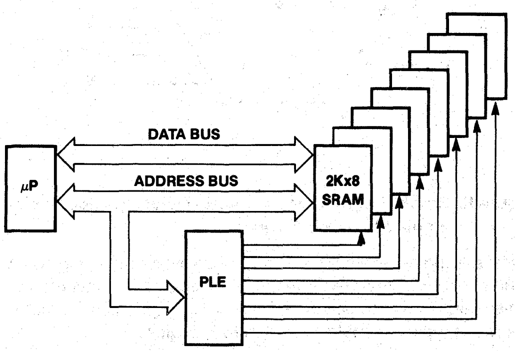


Figure 4. PLE Address Decoding Application. The PLE Device Selects One of Eight 2Kx8 Static RAMs by Decoding Several Microprocessor Address Lines

PLE devices also offer a very flexible solution for code conversion applications. Translations of codes such as from ASCII to EBCIDIC, Binary to BCD (Binary Coded Decimal), or BCD to Gray code can be implemented in PLEs. The 74S184 Binary-to-BCD Converter is actually a 32x8 PROM.

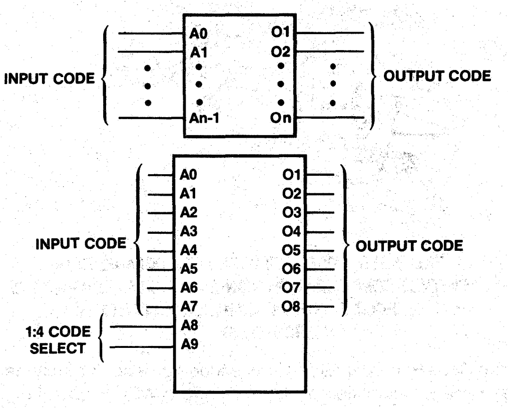


Figure 5. Two Examples of PLE Code Converters. The Second Example Illustrates How to Use Two Inputs as Code Select Lines so that Four Converters can be Provided in One PLE Device

Standard ALUs (such as the 74S181) may not provide a very specailized function which a particular system requires, such as BCD arithmetic. In this case a PLE device is again a good alternative. Although the PLE device may be slower than a dedicated ALU, the presence of this specialized function is critical. For example, a 4-bit ALU can be constructed in a PLE device with twelve inputs (A3-A0, B3-B0, I2-I0, Cin) and eight outputs (F3-F0,/G,/P, Cout, A = B). Any eight functions can be implemented.

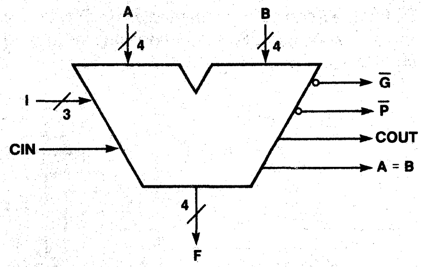
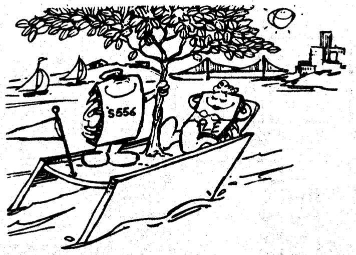


Figure 6. Block Diagram for a 4-Bit ALU which can be Implemented in a PLE Device

Data scaling is another PLE device application. A dedicated multiplier is not required if the scaling factor is a constant; the prescaled result can be stored in a PLE device. Fixed-bit multipliers are typically implemented in PLE devices.

Column compression technique (also called Wallace Tree Compression) is used when expanding an array of several smaller parallel multipliers to perform large wordlength multiplication. These smaller multipliers will generate partial products (intermediate results) which must be numerically summed according to bit significance in order to calculate the final wordlength multiplication. Many levels of 2-input bus adders can be used to add these partial products, but the carry propagation delays may be too long. However, partial product adders implemented in PLE devices can do compression of many levels without passing carries. Thus, the summmation will be much faster.



". . . THE 'S556, TOGETHER WITH PLEs ORGANIZED IN A WALLACE-TREE CONFIGURATION, CAN SAIL RIGHT ALONG AT THE RATE OF FOUR 56 X 56 MULTIPLICATIONS EVERY MICROSECOND . . ."

Group Code Recorder (GCR) is an encoding/decoding scheme used for error detection on tape. During a WRITE operation, each 8-bit word is divided into two 4-bit nibbles. Both nibbles are then encoded into 5-bit codes before being recorded onto tape. Both 5-bit codes are decoded back to the original 4-bit nibbles and then combined during a READ operation. PLE circuits are exceptionally useful in mapping the 4-bit data to the 5-bit code and back.

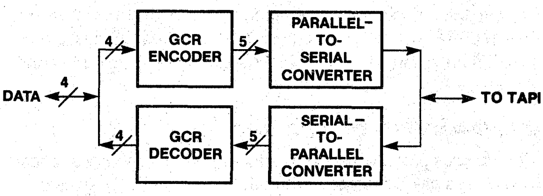
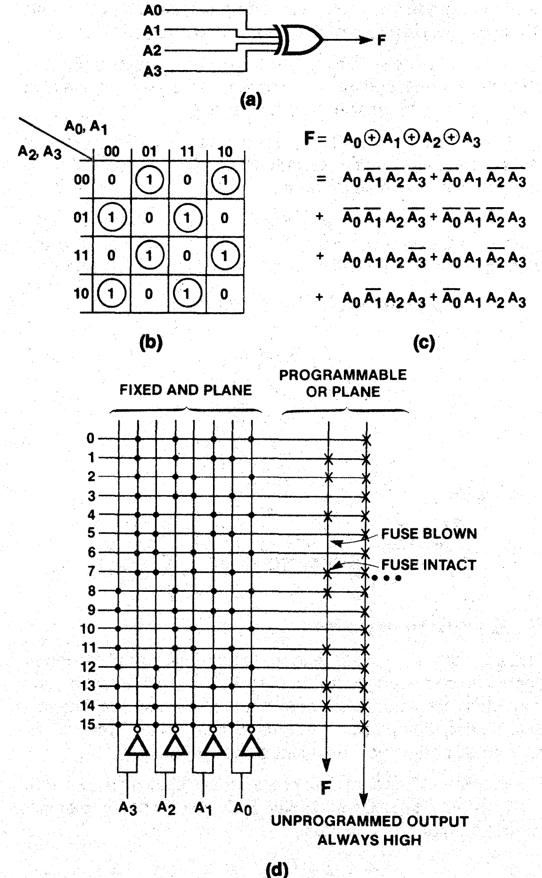


Figure 7. GCR Encoder/Decoder Block Diagram

Exclusive-OR gates, being half adders, are very prevalent in Error Detection and Correction (EDC) schemes. Many SSI chips are required to implement this function while PLA and PAL devices may not provide sufficient product terms. PLE devices are again an ideal solution.

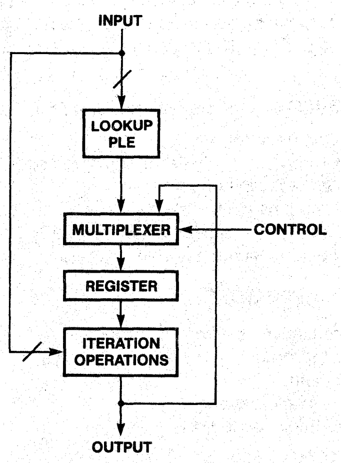


A2, A3 \ A0, A1	00	01	11	10
00	0	1	0	1
01	1	0	1	0
11	0	1	0	1
10	1	0	1	0

$$\begin{aligned} F = {} & A_0 \oplus A_1 \oplus A_2 \oplus A_3 \\ = {} & A_0 \overline{A_1}\,\overline{A_2}\,\overline{A_3} + \overline{A_0} A_1 \overline{A_2}\,\overline{A_3} \\ + {} & \overline{A_0}\,\overline{A_1} A_2 \overline{A_3} + \overline{A_0}\,\overline{A_1}\,\overline{A_2} A_3 \\ + {} & A_0 A_1 A_2 \overline{A_3} + A_0 A_1 \overline{A_2} A_3 \\ + {} & A_0 \overline{A_1} A_2 A_3 + \overline{A_0} A_1 A_2 A_3 \end{aligned}$$

Figure 8. Exclusive-OR Gates can be Implemented in PLE Very Efficiently. A 4-Input XOR Gate (a) Maps into a Checkerboard Pattern in a Karnaugh Map (b) and Requires Eight Product Terms (c). The PLE Implementation is Shown in (d). An 8-input XOR Gate Requires Sixteen Product Terms

In many applications, the speed of the converging series used to generate the trigonometric functions is too slow and the accuracy obtained by direct table look-up requires too much hardware. A good compromise between speed and hardware is to store an approximation to the function in a PLE device. Then use this approximation as a starting point for an iterative algorithm (such as Newton-Raphson) to obtain additional accuracy. High-Speed division, multiplication, and square-root calculations can be performed in a similar manner.



ˑigure 9. PLE Look-Up Tables and Iteration Loops can be Used to Generate Very Accurate Trigonometric and Arithmetic Functions. An Approximation to the Function is Stored and Additional Accuracy is Gained Using Iteration Operations

)istributed arithmetic is used for performing convolution opertions without using multiplier/accumulators. If the coefficients re constant, a look-up table for convolution can be stored in a 'LE device, thus replacing the multiplier.

ɩesidue arithmetic (also called Carry-Independant arithmetic) ; a technique used to perform very fast integer arithmetic. High peed is achieved by using numbers in residue representation ɔ that the sequential delay of carries on digits of higher ignificance is eliminated. A Residue Numbering System (RNS) ; determined using an optimum moduli when designing the ystem. Conversion to and from residue representation are asic mapping functions which can be conveniently done in a LE device. Also, since operations in residue arithmetic are erformed using modulo addition and multiplication without arries, these operations can also be done using PLE devices. In eneral, residue arithmetic should only be used for integer ·ithmetic which requires intensive operations.

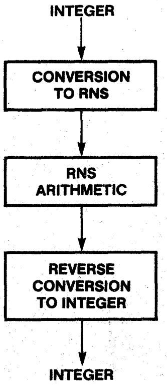


Figure 10. Architecture of a System Based on RNS. An Integer Number is Converted to RNS Representation Using PLE Devices, Then the RNS Arithmetic is Performed Using Some Other PLE Devices, and Finally the RNS Result is Converted Back to Integer Representation Again Using PLE Devices

Restrictions

The basic restrictions for using PLE devices to replace SSI/MSI parts are:

1) Since a memory element has a product term for every combination of literals of all the input terms, static hazard is normally unavoidable. For example, there are 5 inputs available in a 32 x 8 PROM. In order to generate a function like:

$$f = a^* b^* c^* d$$

The actual implementation inside the PROM will be:

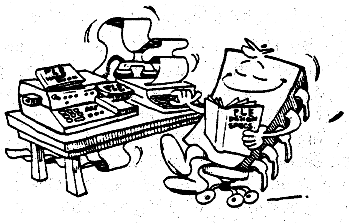
$$f = a^* b^* c^* d^*/e + a^* b^* c^* d^* e$$

If a = b = c = d = HIGH, according to the first equation, we shall expect f to remain HIGH independent of e changing. In the actual PROM implementaton, there will be no hazard if e stays either HIGH or LOW. But if e changes, depending on whether e or /e will occur first, there exists the possibility that both product terms in the second equation will be LOW momentarily, which may cause a static logic hazard (HIGH to LOW to HIGH) for f. This hazard is commonly called a "glitch". Static hazards are not a problem for many applications, like those offered in this paper, but extreme care must be taken if the output of a PLE device is used to strobe another device.

```
ADDRESS e    d    c    b    a    f
-------------------------------------
  00    0    0    0    0    0    0
  01    0    0    0    0    1    0
  02    0    0    0    1    0    0
                  .
                  .
                  .
  0C    0    1    1    0    0    0
  0D    0    1    1    0    1    0
  0E    0    1    1    1    0    0
  0F    0    1    1    1    1    1<-
  10    1    0    0    0    0    0  |
  11    1    0    0    0    1    0  |
  12    1    0    0    1    0    0  |
                  .                 |
                  .                 |
                  .                 |
  1D    1    1    1    0    1    0  |
  1E    1    1    1    1    0    0  |
  1F    1    1    1    1    1    1<-
-------------------------------------
```

Figure 11. This Truth Table Graphically Illustrates the Possible Glitch (HIGH to LOW to HIGH Hazard) for the Function f = a* b* c* d Implemented in a 32x8 PROM. Address 0F and 1F Contain a 1 while All Other Locations Contain a 0 for Output f. If Address Input e Should Change, the PROM Decoders Could Momentarily Selct a Location Containing a 0

2) Although PROMs are available with registered outputs, internal feedback from the outputs and buried registers are not yet available in PROMs. External connections from some outputs to inputs must be made for applications which require feedback (such as in state machines). However, Registered PROMs without feedback are useful for pipelining (overlap instruction fetch and execution) in order to increase system throughput.



PLEASM Software Support

Monolithic Memories has developed a software tool to assist in designing and programming PROMs as PLE devices. This package, called "PLEASM" (PLE Assembler), is available for several computers including the VAX/VMS and IBM PC/DOS. PLEASM converts design equations (Boolean and arithmetic) into truth tables and formats compatible with PROM programmers. A simulator is also provided to test a design using a Function Table before actually programming the PLE device. The PLEASM operators are listed below and the PLEASM catalog of operations is given on the next page. A sample PLE Design Specification (source code for PLEASM software) with PLEASM outputs is given in Figure 12. PLEASM software may be requested through the Monolithic Memories IdeaLogic Exchange.

Operators (in hierarchy of evaluation)

;	Comment follows
.	Dot operator (pin list or arithmetic operator follows)
ADD	Address pins (Inputs)
DAT	Data pins (Outputs)
,	Delimiter, separates binary bits (MSB first)
=	Equality (combinatorial)

BOOLEAN OPERATORS

/	Complement, prefix to a pin name
*	AND (product)
+	OR (sum)
:+:	XOR (exclusive or)
:*:	XNOR (exclusive nor)

ARITHMETIC OPERATORS

.*.	Multiply (numeric multiplication)
.+.	Plus (numeric addition)

Monolithic Memories PLEASM version 1.2D © copyright 1984 Monolithic Memories

PLEASM — PLE Assembler — provides the following options:

C Catalog	— Prints the PLEASM catalog of operations
E Echo Input	— Prints the PLE design specifications
T Truth Table	— Prints the entire truth table
B Brief Table	— Prints only used addresses in the truth table
H Hex Table	— Prints the truth table in HEX form
S Simulate	— Exercises the function table in the logic equations
I Intel Hex	— Generates INTEL HEX programming format
A ASCII Hex	— Generates ASCII HEX programming format
Q Quit	— Exits PLEASM

```
PLE5P8                                     PLE DESIGN SPECIFICATION
P5000                                         VINCENT COLI 10/03/82
BASIC GATES
MMI SANTA CLARA, CALIFORNIA
.ADD I0 I1 I2 I3 I4
.DAT O1 O2 O3 O4 O5 O6 O7 O8

O1 =  I0                                  ; BUFFER

O2 = /I0                                  ; INVERTER

O3 =  I0  *  I1  *  I2  *  I3  *  I4      ; AND GATE

O4 =  I0  +  I1  +  I2  +  I3  +  I4      ; OR GATE

O5 = /I0  + /I1  + /I2  + /I3  + /I4      ; NAND GATE

O6 = /I0  * /I1  * /I2  * /I3  * /I4      ; NOR GATE

O7 =  I0 :+: I1 :+: I2 :+: I3 :+: I4      ; EXCLUSIVE OR GATE

O8 =  I0 :*: I1 :*: I2 :*: I3 :*: I4      ; EXCLUSIVE NOR GATE

FUNCTION TABLE

I0 I1 I2 I3 I4 O1 O2 O3 O4 O5 O6 O7 O8

;INPUT   -  -  OUTPUTS  FROM  BASIC  GATES  -  -
;01234   BUF  INV  AND  OR  NAND  NOR  XOR  XNOR   COMMENTS
---------------------------------------------------------------------
 LLLLL    L    H    L    L    H     H    L    L     ALL ZEROS
 HHHHH    H    L    H    H    L     L    H    H     ALL ONES
 HLHLH    H    L    L    H    H     L    H    H     ODD CHECKERBOARD
 LHLHL    L    H    L    H    H     L    L    L     EVEN CHECKERBOARD
---------------------------------------------------------------------

DESCRIPTION

THIS EXAMPLE ILLUSTRATES THE USE OF PLEs TO IMPLEMENT THE BASIC GATES:
BUFFER, INVERTER, AND GATE, OR GATE, NAND GATE, NOR GATE, EXCLUSIVE OR
GATE, AND EXCLUSIVE NOR GATE.

NOTE ALSO THAT THREE-STATE OUTPUTS ARE PROVIDED WITH ONE ACTIVE LOW
OUTPUT ENABLE CONTROL (/E).

PLEASM GENERATES THE PROM TRUTH TABLE FROM THE LOGIC EQUATIONS AND
SIMULATES THE FUNCTION TABLE IN THE LOGIC EQUATIONS.
```

Figure 12a. PLE Design Specification. This is the Source Code for PLEASM Software. PLEASM Software Generates the Truth Table and Programming Formats from the Equations. PLEASM Software Also Exercises the Function Table in the Equation and Reports Errors

```
BASIC GATES
.ADD I0 I1 I2 I3 I4
.DAT O1 O2 O3 O4 O5 O6 O7 O8

 ADD   A0 A1 A2 A3 A4  O1 O2 O3 O4 O5 O6 O7 O8
-----------------------------------------------
   0   L  L  L  L  L   L  H  L  L  H  H  L  L
   1   H  L  L  L  L   H  L  L  H  H  L  H  H
   2   L  H  L  L  L   L  H  L  H  H  L  H  H
   3   H  H  L  L  L   H  L  L  H  H  L  L  L
   4   L  L  H  L  L   L  H  L  H  H  L  H  H
   5   H  L  H  L  L   H  L  L  H  H  L  L  L
   6   L  H  H  L  L   L  H  L  H  H  L  L  L
   7   H  H  H  L  L   H  L  L  H  H  L  H  H
   8   L  L  L  H  L   L  H  L  H  H  L  H  H
   9   H  L  L  H  L   H  L  L  H  H  L  L  L
  10   L  H  L  H  L   L  H  L  H  H  L  L  L
  11   H  H  L  H  L   H  L  L  H  H  L  H  H
  12   L  L  H  H  L   L  H  L  H  H  L  L  L
  13   H  L  H  H  L   H  L  L  H  H  L  H  H
  14   L  H  H  H  L   L  H  L  H  H  L  H  H
  15   H  H  H  H  L   H  L  L  H  H  L  L  L
  16   L  L  L  L  H   L  H  L  H  H  L  H  H
  17   H  L  L  L  H   H  L  L  H  H  L  L  L
  18   L  H  L  L  H   L  H  L  H  H  L  L  L
  19   H  H  L  L  H   H  L  L  H  H  L  H  H
  20   L  L  H  L  H   L  H  L  H  H  L  L  L
  21   H  L  H  L  H   H  L  L  H  H  L  H  H
  22   L  H  H  L  H   L  H  L  H  H  L  H  H
  23   H  H  H  L  H   H  L  L  H  H  L  L  L
  24   L  L  L  H  H   L  H  L  H  H  L  L  L
  25   H  L  L  H  H   H  L  L  H  H  L  H  H
  26   L  H  L  H  H   L  H  L  H  H  L  H  H
  27   H  H  L  H  H   H  L  L  H  H  L  L  L
  28   L  L  H  H  H   L  H  L  H  H  L  H  H
  29   H  L  H  H  H   H  L  L  H  H  L  L  L
  30   L  H  H  H  H   L  H  L  H  H  L  L  L
  31   H  H  H  H  H   H  L  H  H  L  L  H  H
-----------------------------------------------

HEX CHECK SUM = 00F3C
```

igure 12b. Truth Table. PLEASM Software Generates This Truth Table which can be Used for Verifying Your Design

```
BASIC GATES
.ADD I0 I1 I2 I3 I4
.DAT O1 O2 O3 O4 O5 O6 O7 O8

 ADD      HEX ADDRESS      HEX DATA
------------------------------------
   0          000             32
   1          001             D9
   2          002             DA
   3          003             19
   4          004             DA
   5          005             19
   6          006             1A
   7          007             D9
   8          008             DA
   9          009             19
  10          00A             1A
  11          00B             D9
  12          00C             1A
  13          00D             D9
  14          00E             DA
  15          00F             19
  16          010             DA
  17          011             19
  18          012             1A
  19          013             D9
  20          014             1A
  21          015             D9
  22          016             DA
  23          017             19
  24          018             1A
  25          019             D9
  26          01A             DA
  27          01B             19
  28          01C             DA
  29          01D             19
  30          01E             1A
  31          01F             CD
------------------------------------

HEX CHECK SUM = 00F3C
```

Figure 12c. Hex Table. PLEASM Software Generates This Truth Table in Hexadecimal Form for Verification of Locations in the PLE

```
32 D9 DA 19 DA 19 1A D9 DA 19 1A D9 1A D9 DA 19 .
DA 19 1A D9 1A D9 DA 19 1A D9 DA 19 DA 19 1A CD .

00F3C
```

Figure 12d. ASCII Hex Programming Format. PLEASM Software Generates this ASCII Hex Programming Format with Hex Check Sum. Control Characters are Included so that the Information can be Down-Loaded Directly to a PROM Programmer

11

```
:1000000032D9DA19DA191AD9DA191AD91AD9DA1940
:10001000DA191AD91AD9DA191AD9DA19DA191ACD54
:00000001FF
```

Figure 12e. Intel Hex Programming Format. PLEASM Software Generates this Intel Hex Programming Format with a Hex Check Sum Following Every 16 Bytes of Data

PLE Family

Monolithic Memories carries a family of fast PROMs which can be used as Memory or PLE devices. Since the critical parameter for logic applications is speed, our series of fast PROMs have worst-case memory access times (or propagation delays) ranging from 15 ns for small PROMs to 40 ns for large PROMs. The Logic Symbols for four of the PLE devices are given in Figure 13 and a summary of the PLE family is given below:

PLE Selection Guide

PART NUMBER	INPUTS	OUTPUTS	PRODUCT TERMS	OUTPUT REGISTERS	t_{PD} (ns) MAX*
PLE5P8	5	8	32		25
PLE5P8A	5	8	32		15
PLE8P4	8	4	256		30
PLE8P8	8	8	256		28
PLE9P4	9	4	512		35
PLE9P8	9	8	512		30
PLE10P4	10	4	1024		35
PLE10P8	10	8	1024		35†
PLE11P4	11	4	2048		35
PLE11P8	11	8	2048		35
PLE12P4	12	4	4096		35
PLE12P8	12	8	4096		40
PLE9R8	9	8	512	8	15
PLE10R8	10	8	1024	8	15
PLE11RA8	11	8	2048	8	15
PLE11RS8	11	8	2048	8	15

*Clock to output time for registered outputs

† Preliminary data.

NOTE: Commercial limits specified.

Acknowledgements

Several of the designs discussed in this paper were proposed by our good friend and colleague Ulrik Mueller, who is now studying Computer Science in his native country, Denmark, and our Monolithic Memories Pal Zahir Ebrahim. Special thanks also go to Ranjit Padmanabhan for writing the PLEASM simulator.

Summary

There are many interesting applications for high-speed PROMs used as PLE devices. A software package called "PLEASM" software is available as a development tool.

References

1. "PAL Programmable Array Logic Handbook", 3rd edition, J. Birkner, V. Coli, Monolithic Memories, Inc.
2. "Systems Design Handbook", Monolithic Memories, Inc.
3. "Bipolar LSI 1984 Databook", 5th edition, Monolithic Memories, Inc.
4. "PROMs and PLEs: An Application Perspective", Z. Ebrahim, Monolithic Memories Application Note AN-126.
5. "An Introduction to Arithmetic for Digital Designers", S. Waser, M.J. Flynn, Holt, Rinehart & Winston, N.Y., 1982.

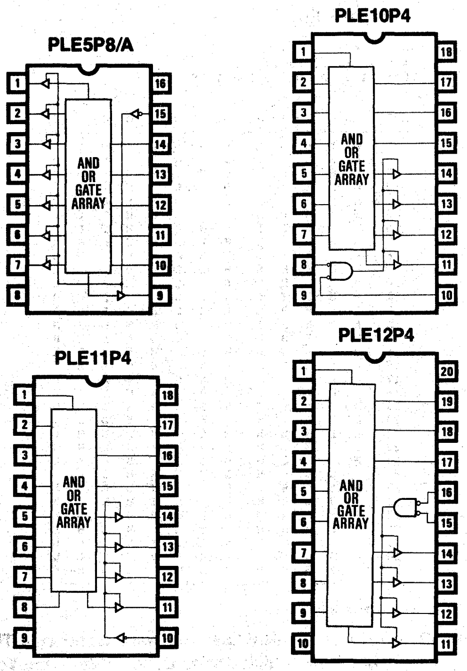


Figure 13. Four Sample PLE Logic Symbols

ABEL™, A Complete Design Tool For Programmable Logic

Michael J. Holley
DATA I/O
10525 Willows Rd. N.E.
Redmond, WA 98073-9746

As the use of PAL® and PLE devices (PROMs) increases, the need for high-level design tools becomes necessary. Designers need easier, faster, and more efficient ways to design with such programmable devices. With the more complex devices currently being introduced to the market, this need is even greater. Additionally, a designer should be able to specify logic designs in a way that makes sense in engineering terms; he or she should not have to learn a new way of thinking about designs.

ABEL™, a complete logic design tool for PAL devices, PLE devices and FPLA devices meets these requirements. ABEL™ incorporates a high-level design language and a set of software programs that process logic designs to give correct and efficient designs.

The ABEL™ design language offers structures familiar to designers: state diagrams, truth tables, and Boolean equations. The designer can choose any of these structures or combine them to describe a design. Macros and directives are also available to simplify complex designs.

The ABEL™ software programs process designs described with the high-level language. Processing includes syntax checking, automatic logic reduction, automatic design simulation, verification that a given design can be implemented in a chosen device, and automatic generation of design documentation.

To use ABEL™, the designer uses an editor to create a source file containing an ABEL™ design description. He then processes the source file with the ABEL™ software programs to produce a programmer load file. The programmer load file is used by logic and PLE programmers to program devices. Several programmer load file formats are supported by ABEL™ so that different programmers may be used.

The source file created by the designer must contain test vectors if simulation is to be performed. Test vectors describe the desired (expected) input-to-output function of the design in a truth table format. The ABEL™ simulator applies the inputs contained in the test vectors to the design and checks the obtained outputs against the expected outputs in the vectors. If the outputs obtained during simulation do not match those specified in the test vectors, an error is reported.

Following are two designs described in the ABEL™ design language. These designs would be processed to verify their correctness and to reduce the number of terms required to implement them. The first design is for a PAL device, the second for a PLE logic circuit.

\BEL™ is a trademark of DATA I/O.

6809 MEMORY ADDRESS DECODER

Address decoding is a typical application of programmable logic devices, and the following describes the ABEL™ implementation of such a design.

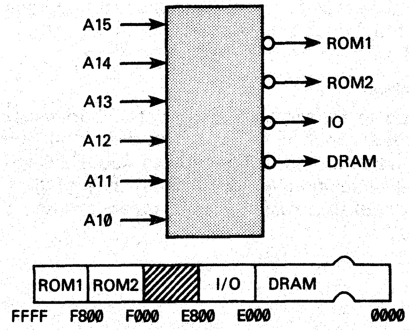


Figure 1. Block Diagram: 6809 Memory Address Decoder

Design Specification

Figure 1 shows a block diagram for the design and a continuous block of memory divided into sections containing dynamic RAM (*DRAM*), I/O (*IO*), and two sections of ROM (*ROM1* and *ROM2*). The purpose of this decoder is to monitor the six high-order bits (*A15-A10*) of a sixteen-bit address bus and select the correct section of memory based on the value of these address bits. To perform this function, a simple decoder with six inputs and four outputs is designed with a 14L4 PAL device.

Table 1 shows the address ranges associated with each section of memory. These address ranges can also be seen in figure 1.

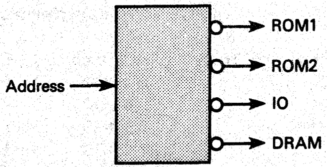


Figure 2. Simplified Block Diagram: 6809 Memory Address Decoder

Design Method

Figure 2 shows a simplified block diagram for the address decoder. The address decoder is implemented with simple

AL® is a registered trademark of Monolithic Memories.

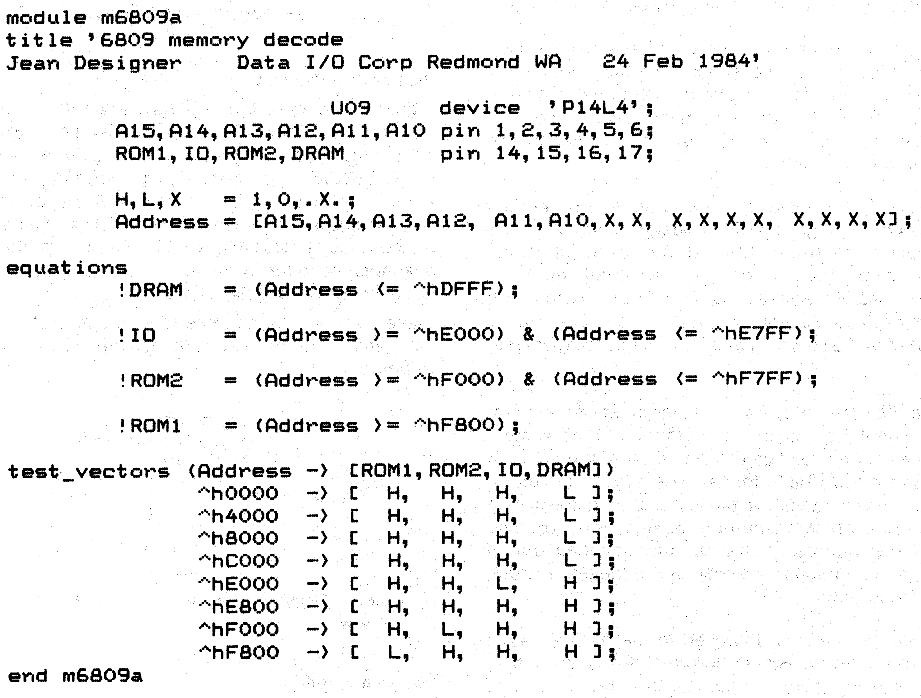
Boolean equations employing both relational and logical operators as shown in figure 3. A significant amount of simplification is achieved by grouping the address bits into a set named *Address.* The lower-order ten address bits that are not used for the address decode are given "don't care" values in the address set. In this way, the designer indicates that the address in the overall design (that beyond the decoder) contains sixteen bits, but that bits 0-9 do not affect the decode of that address. This is opposed to simply defining the set as, *Address = [A15,A14,A13,A12,A11,A10],* which ignores the existence of the lower-order bits. Specifying all 16 address lines as members of the address set also allows full 16-bit comparisons of the address value against the ranges shown in table 1.

Memory Section	Address Range (hex)
DRAM	0000–DFFF
I/O	E000–E7FF
ROM2	F000–F7FF
ROM1	F800–FFFF

Table 1. Address Ranges for 6809 Controller

Test Vectors

In this design, the test vectors are a straightforward listing of the values that must appear on the output lines for specific address values. The address values are specified in hexadecimal notation on the left side of the "->"" symbol. Input to a design always appear on the left side of the test vectors. The expected outputs are specified to the right of the "->" symbol. The designer chose in this case to use the symbols *H* and *L* instead of the binary values 1 and 0 to describe the outputs. The correspondence between the symbols and the binary values was defined in the constant declaration section of the source file, just above the section labeled *equations.*



```
module m6809a
title '6809 memory decode
Jean Designer    Data I/O Corp Redmond WA   24 Feb 1984'

                            U09      device  'P14L4';
          A15,A14,A13,A12,A11,A10 pin 1,2,3,4,5,6;
          ROM1,IO,ROM2,DRAM       pin 14,15,16,17;

          H,L,X    = 1,0,.X.;
          Address  = [A15,A14,A13,A12, A11,A10,X,X, X,X,X,X, X,X,X,X];

equations
          !DRAM    = (Address <= ^hDFFF);

          !IO      = (Address >= ^hE000) & (Address <= ^hE7FF);

          !ROM2    = (Address >= ^hF000) & (Address <= ^hF7FF);

          !ROM1    = (Address >= ^hF800);

test_vectors (Address -> [ROM1,ROM2,IO,DRAM])
                 ^h0000  -> [  H,   H,   H,    L ];
                 ^h4000  -> [  H,   H,   H,    L ];
                 ^h8000  -> [  H,   H,   H,    L ];
                 ^hC000  -> [  H,   H,   H,    L ];
                 ^hE000  -> [  H,   H,   L,    H ];
                 ^hE800  -> [  H,   H,   H,    H ];
                 ^hF000  -> [  H,   L,   H,    H ];
                 ^hF800  -> [  L,   H,   H,    H ];
end m6809a
```

Figure 3. Source File: 6809 Memory Address Decoder

Seven-Segment Display Decoder

This display decoder decodes a four-bit binary number to display the decimal equivalent on a seven-segment LED display. The design incorporates the ABEL™ truth table format and is implemented on a RA5P8 PLE.

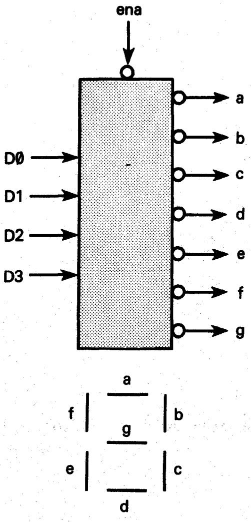


Figure 4. Block Diagram: Seven-Segment Display Decoder

Design Specification

Figure 4 shows a block diagram for the decoder design and a drawing of the display with each of the seven segments labeled to correspond to the decoder outputs. To light up any one of the segments, the corresponding line must be driven low. Four input lines *D0-D3* are decoded to drive the correct output lines. The outputs are named *a, b, c, d, e, f,* and *g* corresponding to the display segments. All outputs are active low. An enable, *ena*, is provided. When *ena* is low, the decoder is enabled; when *ena* is high, all outputs are driven to high impedance.

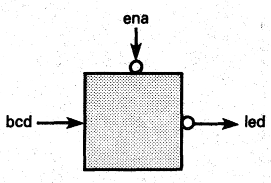


Figure 5. Simplified Block Diagram: Seven-Segment Display Decoder

Design Method

Figures 5 and 6 show the simplified block diagram and the source file for the ABEL™ implementation of the display decoder. The FLAG statement is used to make sure that the programmer load file is in the Motorola Exorciser format. The binary inputs and the decoded outputs are grouped into the sets *bcd* and *led* to simplify notation. The constants *ON* and *OFF* are declared so that the design can be described in terms of turning a segment on or off. To turn a segment on, the appropriate line must be driven low, thus *ON* is declared as 0 and *OFF* as 1.

The design is described in two sections: an equations section and a truth table section. The decoding function is described with a truth table that specifies the outputs required for each combination of inputs. The first line of the truth table (the truth table header) names the inputs and outputs. In this example, the inputs are contained in the set named *bcd* and the outputs are in *led*. The body of the truth table defines the input-to-output function. Because the design decodes a number to a seven-segment display, values for *bcd* are expressed as decimal numbers, and values for *led* are expressed with the constants *ON* and *OFF* that were defined in the declarations section of the source file. This makes the truth table easy to read and understand; the incoming value is a number and the outputs are on and off signals to the LED.

The input and output values could have just as easily been described in another form. Take for example the line in the truth table:

```
5 -> [ON,OFF,ON,ON,OFF,ON,ON]
```

This could have been written in the equivalent form:

```
[0,1,0,1] - > 36
```

In this second form, 5 was expressed as a set containing binary values, and the LED set was converted to decimal. (Remember that *ON*was defined as 0 and *OFF* was defined as 1.) Either of the two forms is valid, but the first is more appropriate for this design. The first form can be read as, "the number five turns on the first segment, turns off the second,..." whereas the second form cannot be so easily translated into terms meaningful for this design.

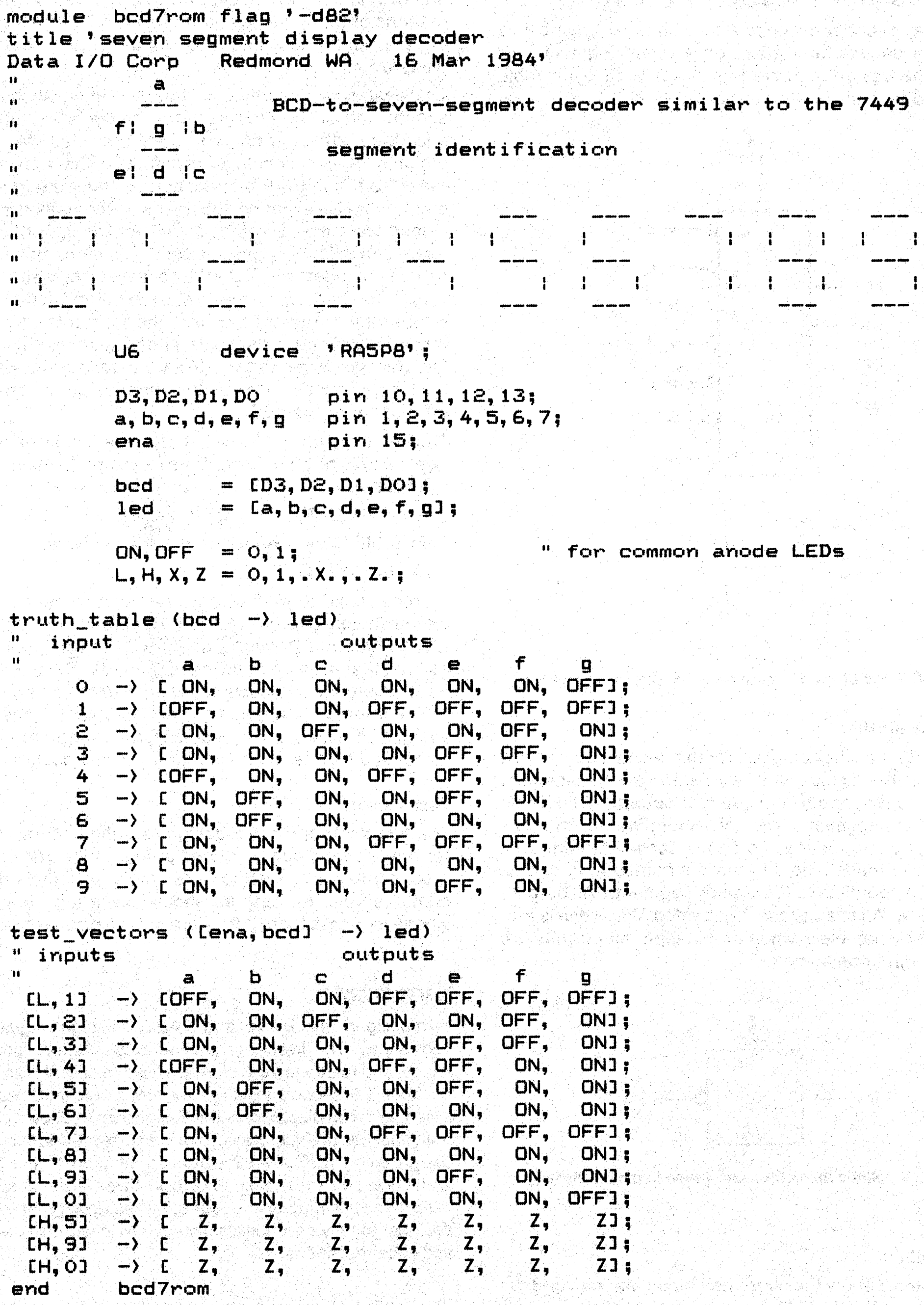
Test Vectors

The test vectors for this design test the decoder outputs for the ten valid combinations of input bits. The enable is also tested by setting *ena* high for the different combinations. All outputs should be at high impedance whenever *ena* is high. If they are not, an error has occurred.

Summary

Two designs described with the ABEL™ design language have been shown. The first design shows how Boolean equations with logical and relational operators are used to describe an address decoder. The second design shows how a truth table describes a seven-segment display decoder design for a PLE logic circuit. In both designs, test vectors were written to test the function of the design using ABEL™'s simulator. In addition to the Boolean equations and truth table shown in these examples, ABEL™ features a state diagram structure. The state diagram allows the designer to fully describe state machines in terms of their states and state transitions.

Regardless of the method used to describe logic, ABEL™'s automatic logic reduction and simulation ensure that the design uses as few terms as possible and that it operates as the designer intended. The end results are savings in time, devices, board space, and money.



```
module  bcd7rom flag '-d82'
title 'seven segment display decoder
Data I/O Corp   Redmond WA   16 Mar 1984'
"          a
"         ---       BCD-to-seven-segment decoder similar to the 7449
"       f| g |b
"         ---           segment identification
"       e| d |c
"         ---
"  ---         ---     ---           ---    ---    ---    ---    ---
" |   |   |       |       |  |   |  |      |          |  |   |  |   |
"              ---     ---    ---    ---    ---           ---    ---
" |   |   |   |           |      |      |  |   |      |  |   |      |
"  ---         ---     ---           ---    ---           ---    ---

        U6      device  'RA5P8';

        D3,D2,D1,D0     pin 10,11,12,13;
        a,b,c,d,e,f,g   pin 1,2,3,4,5,6,7;
        ena             pin 15;

        bcd     = [D3,D2,D1,D0];
        led     = [a,b,c,d,e,f,g];

        ON,OFF  = 0,1;                  " for common anode LEDs
        L,H,X,Z = 0,1,.X.,.Z.;

truth_table (bcd  -> led)
"  input                 outputs
"            a    b    c    d    e    f    g
     0  -> [ ON,  ON,  ON,  ON,  ON,  ON, OFF];
     1  -> [OFF,  ON,  ON, OFF, OFF, OFF, OFF];
     2  -> [ ON,  ON, OFF,  ON,  ON, OFF,  ON];
     3  -> [ ON,  ON,  ON,  ON, OFF, OFF,  ON];
     4  -> [OFF,  ON,  ON, OFF, OFF,  ON,  ON];
     5  -> [ ON, OFF,  ON,  ON, OFF,  ON,  ON];
     6  -> [ ON, OFF,  ON,  ON,  ON,  ON,  ON];
     7  -> [ ON,  ON,  ON, OFF, OFF, OFF, OFF];
     8  -> [ ON,  ON,  ON,  ON,  ON,  ON,  ON];
     9  -> [ ON,  ON,  ON,  ON, OFF,  ON,  ON];

test_vectors ([ena,bcd]  -> led)
" inputs                 outputs
"            a    b    c    d    e    f    g
 [L,1]  -> [OFF,  ON,  ON, OFF, OFF, OFF, OFF];
 [L,2]  -> [ ON,  ON, OFF,  ON,  ON, OFF,  ON];
 [L,3]  -> [ ON,  ON,  ON,  ON, OFF, OFF,  ON];
 [L,4]  -> [OFF,  ON,  ON, OFF, OFF,  ON,  ON];
 [L,5]  -> [ ON, OFF,  ON,  ON, OFF,  ON,  ON];
 [L,6]  -> [ ON, OFF,  ON,  ON,  ON,  ON,  ON];
 [L,7]  -> [ ON,  ON,  ON, OFF, OFF, OFF, OFF];
 [L,8]  -> [ ON,  ON,  ON,  ON,  ON,  ON,  ON];
 [L,9]  -> [ ON,  ON,  ON,  ON, OFF,  ON,  ON];
 [L,0]  -> [ ON,  ON,  ON,  ON,  ON,  ON, OFF];
 [H,5]  -> [  Z,    Z,    Z,    Z,    Z,    Z,    Z];
 [H,9]  -> [  Z,    Z,    Z,    Z,    Z,    Z,    Z];
 [H,0]  -> [  Z,    Z,    Z,    Z,    Z,    Z,    Z];
end     bcd7rom
```

Figure 6. Source File: Seven-Segment Display Decoder

CUPL™ The Universal Compiler For Programmable Logic

CUPL is the first software CAD tool designed especially for the support of all programmable logic devices (PLDs), including PALs and PROMs. It was developed specifically for YOU, the Hardware Design Engineer. Each feature of the CUPL language has been chosen to make using programmable logic easier and faster than conventional TTL logic design.

MAJOR FEATURES OF CUPL

1. UNIVERSAL

   a. PRODUCT SUPPORT: CUPL supports products from every manufacturer of programmable logic. With CUPL you are free to use not only PALS, but also other programmable logic devices.

   b. PALASM CONVERSIONS: CUPL has a PALASM to CUPL language translator which allows for an easy conversion from your previous PALASM designs to CUPL.

   c. LOGIC PROGRAMMER COMPATIBILITY: CUPL produces a standard JEDEC download file and is compatible with any logic programmer that uses JEDEC files.

2. HIGH LEVEL LANGUAGE

High Level Language means that the software has features that allow you to work in terms that are more like the way you think than like the final PLD programming pattern. Examples of these are:

   a. FLEXIBLE INPUT: CUPL gives the engineer complete freedom in entering logic descriptions for their design:

      - EQUATIONS
      - TRUTH TABLES
      - STATE MACHINE SYNTAX

   b. EXPRESSION SUBSTITUTION: This allows you to pick a name for an equation and then, rather than write the equation each time it is used, you need only use the name. CUPL will properly substitute the equation during the compile process.

c. SHORTHAND FEATURES: Instead of writing out fully expanded equations CUPL provides various shorthand capabilities such as:

- LIST NOTATION:
```
Rather than         [A7,A6,A5,A4,A3,A2,A1,A0]
CUPL only requires    [A7..0]
```

- BIT FIELDS: A group of bits may be assigned to a name, asin FIELD ADDR = [A7..0]
  Then ADDR may be used in other expressions

- RANGE FUNCTION:
```
Rather than      A15 &  !A14 #
                 A15 &   A14 &  !A13 #
                 A15 &   A14 &   A13 & !A12
CUPL only requires   ADDR:[8000..EFFF]
```

- THE DISTRIBUTIVEPROPERTY:
```
From Boolean Algebra, where   A & (B # C)
is replaced by               A & B # A & C
```

- DeMORGAN'S THEOREM:
```
From Boolean Algebra, where   !(A & B)
is replaced by               !A # !B
```

3. SELF DOCUMENTING

CUPL provides a template file which provides a standard "fill-in-the-blanks" documentation system that is uniform among all CUPL users. Also, CUPL allows for free form comments through out your work so there can be detailed explanations included in each part of the project.

4. ERROR CHECKING

CUPL includes a comprehensive error checking capability with detailed error messages designed to lead you to the source of the problem.

5. LOGIC REDUCTION

CUPL contains the fastest and most powerful minimizer offered for Programmable Logic equation reduction. The minimizer allows the choice of various levels of minimization ranging from just fitting into the target device to the absolute minimum.

6. SIMULATION

With CSIM, the CUPL Simulator, you can simulate your logic prior to programming an actual device. Not only can this save devices but it can help in debugging a system level problem.

7. TEST VECTOR GENERATION

Once the stimulus/response function table information has been entered into the simulator, CSIM will verify the associated test vectors and append them to the JEDEC file for downloading to the logic programmer. The programmer will verify not only the fuse map, but also the functionality of the PLD, giving you added confidence in the operation of your custom part.

8. EXPANDABILITY

CUPL is designed for growth so as new PALs and other devices are introduced you will be kept current with updated device libraries and product enhancements.

DESIGN EXAMPLE USING CUPL

In the following design example, a single PAL (or PROM) is used to replace four TTL packages on the interface card for an IBM-PC computer. The Prototype I/O Channel Interface Card, as supplied by IBM, uses four SSI packages to decode the ten bit I/O address and control the direction and enable for the bus buffer on the PCB. The PAL approach conserves real estate and also adds flexibility to decode not only the pre-assigned address, but the ability to change the board address to any location in the I/O map by merely replacing the programmable device.

1. CIRCUIT OPERATION

The inputs to the decoding logic are the expansion bus addresses A0 thru A9. The logic compares the address on the expansion bus and asserts the "IO_DECODE" signal when the correct address range of 3F0-3FF is seen. In addition, the "ENABLE" signal is also asserted if either the I/O READ or I/O WRITE signals are active during this time. The READ signal, which controls the direction of the data bus buffer, is asserted whenever I/O READ is active and AEN, the DMA Address enable signal is inactive. The AEN signal is negated when the microprocessor has control of the address bus and is generating an I/O cycle.

2. DESIGN METHOD

First, all device pins are assigned in the logic description file (see figure 1) using CUPL's pin declaration statements. Note the use of indexed variables for the address bus allows a simple assignment for pins 1 thru 8. The active polarity for input and output pins are made in these declarations, so the designer need only be concerned with the logic instead of voltage levels.

The address bus is assigned a name using the FIELD statement. This lets the designer then describe the desired address range with the single equation:

```
          range = ioadr:[300..31F] ;
```

instead of the difficult to understand

```
          range = a9 & a8 & !a7 & !a6 & !a5 ;
```

This range expression is then used in the output equation for IO_DECODE and ENABLE. Since ENABLE may be asserted whenever IOR or IOW are true, the intermediate variable IOREQ is created to define this condition. The resultant CUPL equation for ENABLE is simply

```
          enable = range & ioreq ;
```

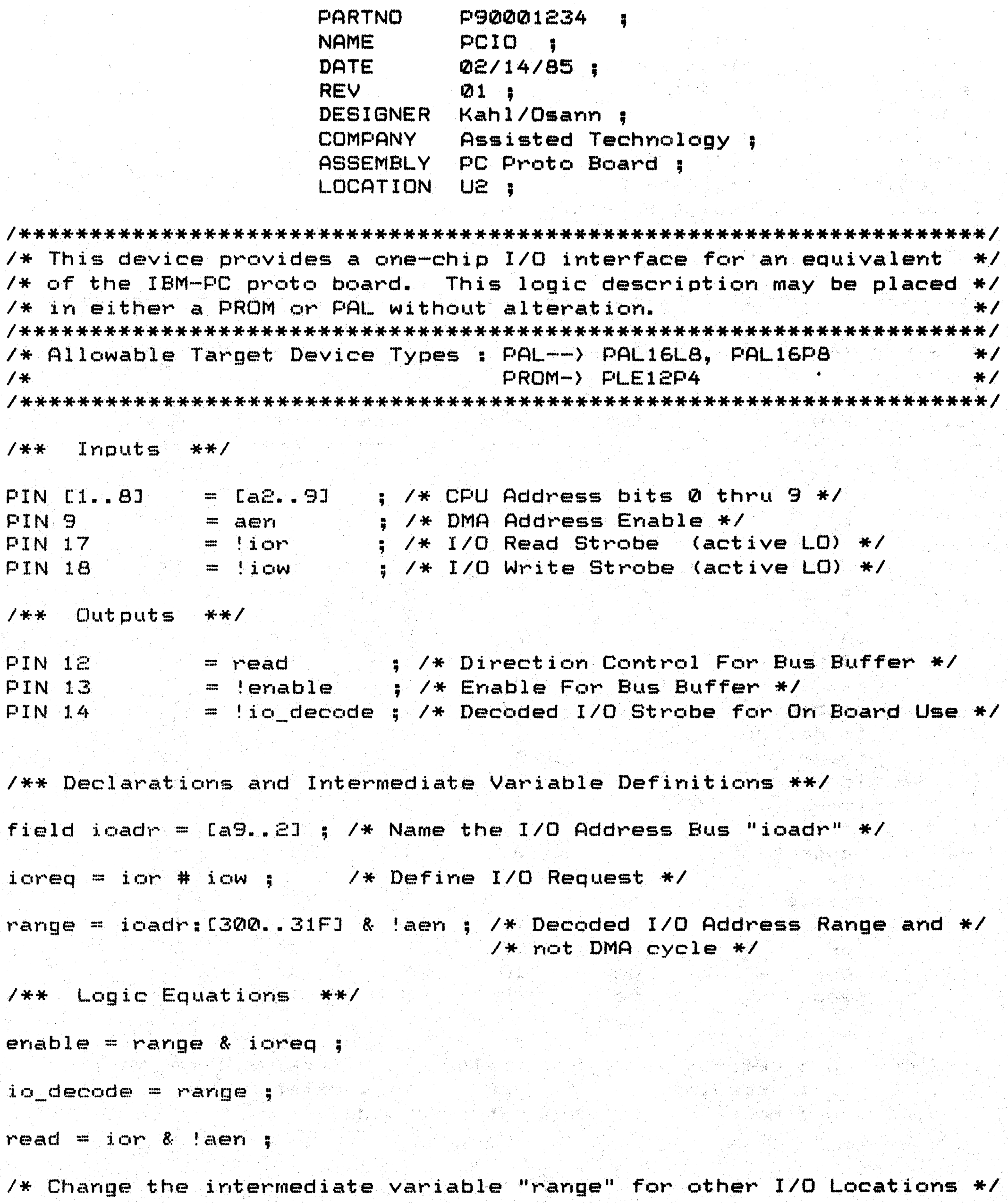
Finally, the READ signal is created using the active IOR and the inactive AEN signals as follows:

```
          read = ior & !aen ;
```

Note that for a device such as the PAL16L8 which has a fixed inverting buffer on all of its output pins, CUPL will automatically convert the logic equations when an output is desired to be active-level HI, as with the READ output above.

3. CUPL OUTPUT FILES

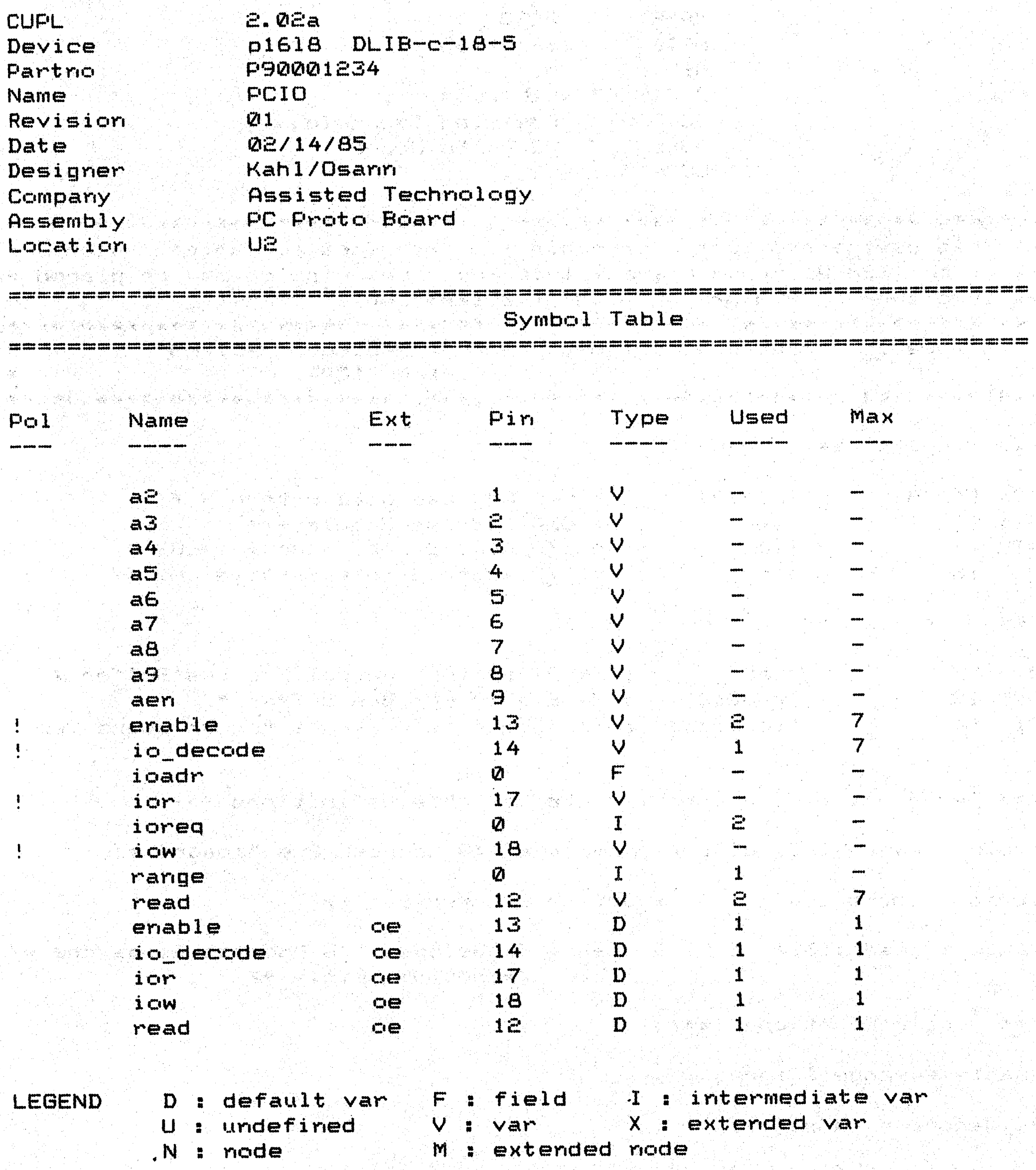
CUPL will create a standard JEDEC output file which is compatible with most logic programmers. A simple serial download link is all that is usually required to transfer the fuse information to the programmer. In addition, CUPL generates an extensive documentation file which assists the designer in analyzing his/her design. Figure 2 shows a small section of this file, illustrating such features as pin and variable names, product term utilization, and other information.



```
                        PARTNO    P9ØØØ1234  ;
                        NAME      PCIO  ;
                        DATE      Ø2/14/85 ;
                        REV       Ø1 ;
                        DESIGNER  Kahl/Osann ;
                        COMPANY   Assisted Technology ;
                        ASSEMBLY  PC Proto Board ;
                        LOCATION  U2 ;

/**********************************************************************/
/* This device provides a one-chip I/O interface for an equivalent   */
/* of the IBM-PC proto board.  This logic description may be placed */
/* in either a PROM or PAL without alteration.                       */
/**********************************************************************/
/* Allowable Target Device Types : PAL--> PAL16L8, PAL16P8           */
/*                                 PROM-> PLE12P4                    */
/**********************************************************************/

/**  Inputs  **/

PIN [1..8]     = [a2..9]    ; /* CPU Address bits Ø thru 9 */
PIN 9          = aen        ; /* DMA Address Enable */
PIN 17         = !ior       ; /* I/O Read Strobe  (active LO) */
PIN 18         = !iow       ; /* I/O Write Strobe (active LO) */

/**  Outputs  **/

PIN 12         = read        ; /* Direction Control For Bus Buffer */
PIN 13         = !enable     ; /* Enable For Bus Buffer */
PIN 14         = !io_decode ; /* Decoded I/O Strobe for On Board Use */


/** Declarations and Intermediate Variable Definitions **/

field ioadr = [a9..2] ; /* Name the I/O Address Bus "ioadr" */

ioreq = ior # iow ;     /* Define I/O Request */

range = ioadr:[3ØØ..31F] & !aen ; /* Decoded I/O Address Range and */
                                  /* not DMA cycle */

/**  Logic Equations  **/

enable = range & ioreq ;

io_decode = range ;

read = ior & !aen ;

/* Change the intermediate variable "range" for other I/O Locations */
```

Figure 1.

11



```
CUPL            2.02a
Device          p1618  DLIB-c-18-5
Partno          P90001234
Name            PCIO
Revision        01
Date            02/14/85
Designer        Kahl/Osann
Company         Assisted Technology
Assembly        PC Proto Board
Location        U2
==================================================================
                                 Symbol Table
==================================================================

Pol     Name            Ext     Pin     Type    Used    Max
---     ----            ---     ---     ----    ----    ---

        a2                      1       V       -       -
        a3                      2       V       -       -
        a4                      3       V       -       -
        a5                      4       V       -       -
        a6                      5       V       -       -
        a7                      6       V       -       -
        a8                      7       V       -       -
        a9                      8       V       -       -
        aen                     9       V       -       -
!       enable                  13      V       2       7
!       io_decode               14      V       1       7
        ioadr                   0       F       -       -
!       ior                     17      V       -       -
        ioreq                   0       I       2       -
!       iow                     18      V       -       -
        range                   0       I       1       -
        read                    12      V       2       7
        enable          oe      13      D       1       1
        io_decode       oe      14      D       1       1
        ior             oe      17      D       1       1
        iow             oe      18      D       1       1
        read            oe      12      D       1       1


LEGEND    D : default var    F : field      I : intermediate var
          U : undefined      V : var        X : extended var
          N : node           M : extended node
```

Figure 2.

CUPL-GTS
DRAW LOGIC SCHEMATICS FOR PAL DESIGNS!

In recent years, programs like CUPL and ABEL have become available to provide high level language support for PAL designs. These languages allow the designer to represent a PAL function in terms of high-level equations, truth tables or state machines. All of these logic description formats are non-graphical in nature and require a good working knowledge of the computer they run on.

Many hardware designers, however, are most comfortable with the traditional logic schematic and have historically had little reason to use a computer in the design process. Use of a high-level PAL design language presents most of us with a variety of simultaneous unknowns:

1. The computer and its operating system.

2. The full screen editor necessary to create the logic description file.

3. The logic compiler or assembler language syntax.

4. Boolean algebra theory.

5. PAL architectures.

Where this combination places an unnecessary burden on the designer, an alternative is now available.

CUPL-GTS is a powerful combination of hardware and software which turns an IBM-PC type computer into a programmable logic workstation which allows the user to draw logic schematics for the function of a PAL. A basic premise in creating CUPL-GTS was to provide a friendly environment where the user is isolated from the traditional keyboard as much as possible. To this end, virtually all functions can be actuated with one button by way of the mouse and a series of pop-up menus which ease the user's task. An area is provided at the top of the CUPL-GTS screen for prompting the user regarding the next operation in a command sequence. Highlighting of various elements on the screen is coordinated with these prompts to enhance their effectiveness. For the most part, the user need only utilize the conventional keyboard for defining symbolic names for wires, pins, objects, and files.

An on-screen HELP facility is provided to aid the user with CUPL-GTS commands. In addition to the basic set of object types which can be easily picked from a pop-up menu, the ability to call up macro-objects is also provided. These macro-objects have been previously drawn using CUPL-GTS and stored away on the disk under their own symbolic name.

After a logic schematic has been entered, the user may quickly check to see if the design fits in a specific PAL. This is done by selecting the "Translate to PLD" command from the main menu which automatically invokes the GTS translation programs. These programs run in an on-screen window which overlays the graphical information, providing feedback in the form of error messages displayed in this window. Following the automatic execution of these programs, the cursor is returned to the user who can then continue to work in the graphics environment without ever having fully left. In this way many errors can be quickly determined and remedied without ever having to let go of the mouse.

When the user wishes a hard copy version of a design, the print command from the main menu may be selected. This causes the GTS print program to execute in an on-screen window according to the printer configuration file (PRINTCAP) which is stored on the disk. The PRINTCAP file allows the user to configure the GTS print function for any dot matrix printer they might have.

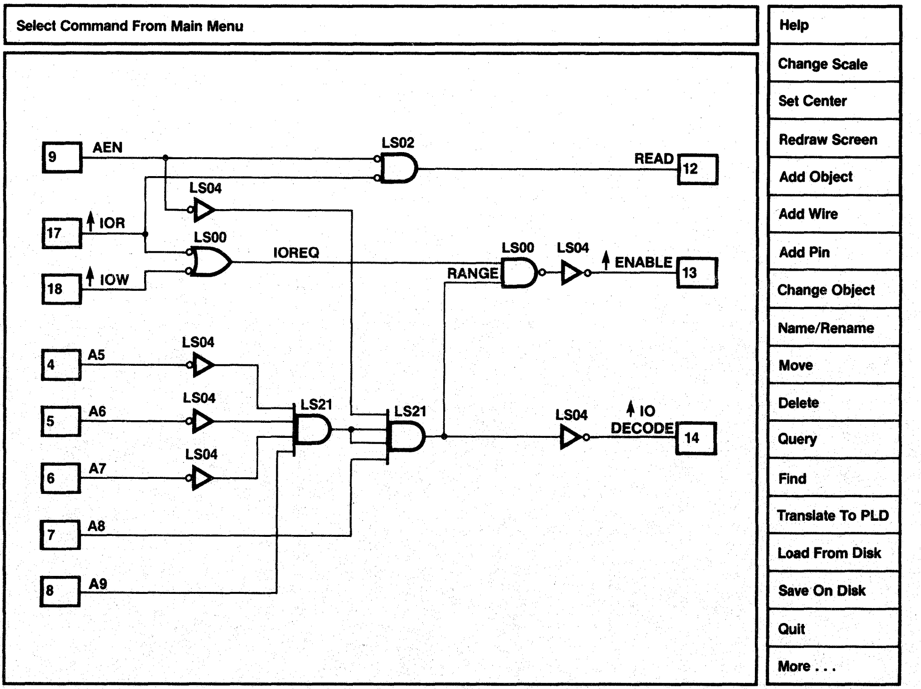
Often a logic description not fit in a particular PAL due to a logic capacity (product-term) limitation. When this occurs, the universal capability of CUPL-GTS will easily allow the user to try placing this same logic in a different PAL of similar architecture.

Since CUPL-GTS incorporates CUPL the high level language in its internal operation, it also benefits from CUPL's powerful "Quine Procedure" logic minimizer. This is especially advantageous for CUPL-GTS as logic descriptions showing many levels of gates can be very deceptive in their ability to consume the logic capacity of a PAL. The presence of the logic minimizer can eliminate unnecessary and redundant logical functions, and maximizes the probability that a design will fit in a target PAL.

Also included with CUPL-GTS is the CUPL simulator, CSIM, which allows the user to simulate a logic design prior to physically creating a programmed PAL. Not only can this save devices, but it can help significantly in debugging a system level problem.

CUPL-GTS is desinged for growth and expandability. As new programmable logic devices are introduced users will be kept current with updated device libraries and product enhancements.

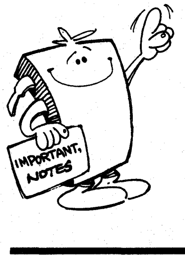
Most of us first use PAL devices to replace TTL in order to shrink a design and/or add functionality. The following example shows how the simple I/O decoder design previously discussed would appear on the CUPL-GTS screen prior to translation to a PAL16L8, PAL16P8 or PLE12P4.



Select Command From Main Menu

A CUPL-GTS Design Screen

Notes



Clock Frequency

Maximum clock frequency, f_{max}

The highest rate at which the clock input of a bistable circuit can be driven through its required sequence while maintaining stable transitions of logic level at the output with input conditions established that should cause changes of output logic level in accordance with the specification.

Current

High-level input current, I_{IH}

The current into* an input when a high-level voltage is applied to that input.

High-level output current, I_{OH}

The current into* an output with input conditions applied that according to the product specification will establish a high level at the output.

High-level output current, I_{CEX}

The high-level leakage current of an open collector output.

Low-level input current, I_{IL}

The current into* an input when a low-level voltage is applied to that input.

Low-level output current, I_{OL}

The current into* an output with input conditions applied that according to the product specification will establish a low level at the output.

Off-state (high-impedance-state) output current (of a three-state output), I_{OZ}

The current into* an output having three-state capability with input conditions applied that according to the product specification will establish the high-impedance state at the output.

Short-circuit output current, I_{OS}

The current into* an output when that output is short-circuited to ground (or other specified potential) with input conditions applied to establish the output logic level farthest from ground potential (or other specified potential).

Supply current, I_{CC}

The current into* the V_{CC} supply terminal of an integrated circuit.

*Current out of a terminal is given as a negative value.

Hold Time

Hold time t_h

The interval during which a signal is retained at a specified input terminal after an active transition occurs at another specified input terminal.

NOTES:
1. The hold time is the actual time between two events and may be insufficient to accomplish the intended result. A minimum value is specified that is the shortest interval for which correct operation of the logic element is guaranteed.
2. The hold time may have a negative value in which case the minimum limit defines the longest interval (between the release of data and the active transition) for which correct operation of the logic element is guaranteed.

Output Enable and Disable Time

Output enable time (of a three-state output) to high level, t_{PZH} (or low level, t_{PZL})

The propagation delay time between the specified reference points on the input and output voltage waveforms with the three-state output changing from a high-impedance (off) state to the defined high (or low) level.

Output enable time (of a three-state output) to high or low level, t_{PZX}

The propagation delay time between the specified reference points on the input and output voltage waveforms with the three-state output changing from a high-impedance (off) state to either of the defined active levels (high or low).

Output disable time (of a three-state output) from high level, t_{PHZ} (or low level, t_{PLZ})

The propagation delay time between the specified reference points on the input and output voltage waveforms with the three-state output changing from the defined high (or low) level to a high-impedance (off) state.

Output disable time (of a three-state output) from high or low level, t_{PXZ}

The propagation delay time between the specified reference points on the input and output voltage waveforms with the three-state output changing from either of the defined active levels (high or low) to a high-impedance (off) state.

t_{EA} is the output enable access time of memory devices.

t_{ER} is the output disable (enable recovery) time of memory devices.

Propagation Time

Propagation delay time, t_{PD}

The time between the specified reference points on the input and output voltage waveforms with the output changing from one defined level (high or low) to the other defined level.

Propagation delay time, low-to-high-level output, t_{PLH}

The time between the specified reference points on the input and output voltage waveforms with the output changing from the defined low level to the defined high level.

Propagation delay time, high-to-low-level output, t_{PHL}

The time between the specified reference points on the input and output voltage waveforms with the output changing from the defined high level to the defined low level.

t_{AA} is the address (to output) access time of memory devices.

Pulse Width

Pulse width, t_w

The time interval between specified reference points on the leading and trailing edges of the pulse waveform.

Setup Time

Setup time, t_{su}

The time interval between the application of a signal that is maintained at one specified input terminal and a consecutive active transition at another specified input terminal.

NOTES: 1. The setup time is the actual time between two events and may be insufficient to accomplish the setup. A minimum value is specified that is the shortest interval for which correct operation of the device is guaranteed.

2. The setup time may have a negative value in which case the minimum limit defines the longest interval (between the active transition and the application of the other signal) for which correct operation of the device is guaranteed.

Voltage

High-level input voltage, V_{IH}

An input voltage within the more positive (less negative) of the two ranges of values assumable by a binary variable.

NOTE: A minimum is specified that is the least positive value of high-level voltage for which operation of the logic element within specification limits is guaranteed.

High-level output voltage, V_{OH}

The voltage at an output terminal with input conditions applied that will establish a high level at the output. The actual input conditions needed are determined by the individual product specification.

Input clamp voltage, V_{IC}

An input voltage in a region of relatively low differential resistance that serves to limit the input voltage swing.

Low-level input voltage, V_{IL}

An input voltage level within the less positive (more negative) of the two ranges of values assumable by a binary variable.

NOTE: A maximum is specified that is the most positive value of low-level input voltage for which operation of the logic element within specification limits is guaranteed.

Low-level output voltage, V_{OL}

The voltage at an output terminal with input conditions applied that will establish a low level at the output. The actual input conditions needed are determined by the individual product specification.

Negative-going threshold voltage V_{T-}

The voltage level at an input that causes a transition as the input voltage falls from a level above the positive-going threshold voltage, V_{T+}.

Positive-going threshold voltage, V_{T+}

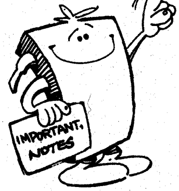
The voltage level at an input that causes a transition as the input voltage rises from a level below the negative-going threshold voltage, V_{T-}.

Truth Table Explanations

H	= high level (steady-state)
L	= low level (steady-state)
↑	= transition from low-to-high level
↓	= transition from high-to-low level
X	= don't care (any input, including transitions)
Z	= off (high-impedance) state of a three-state output
a..h	= the level of steady-state inputs at inputs A through H respectively
Q_0	= level of Q before the indicated steady-state input conditions were established
$\overline{Q}_0$	= complement of Q_0 or level of $\overline{Q}$ before the indicated steady-state input conditions were established
Q_n	= level of Q before the most recent active transition indicated by ↓ or ↑

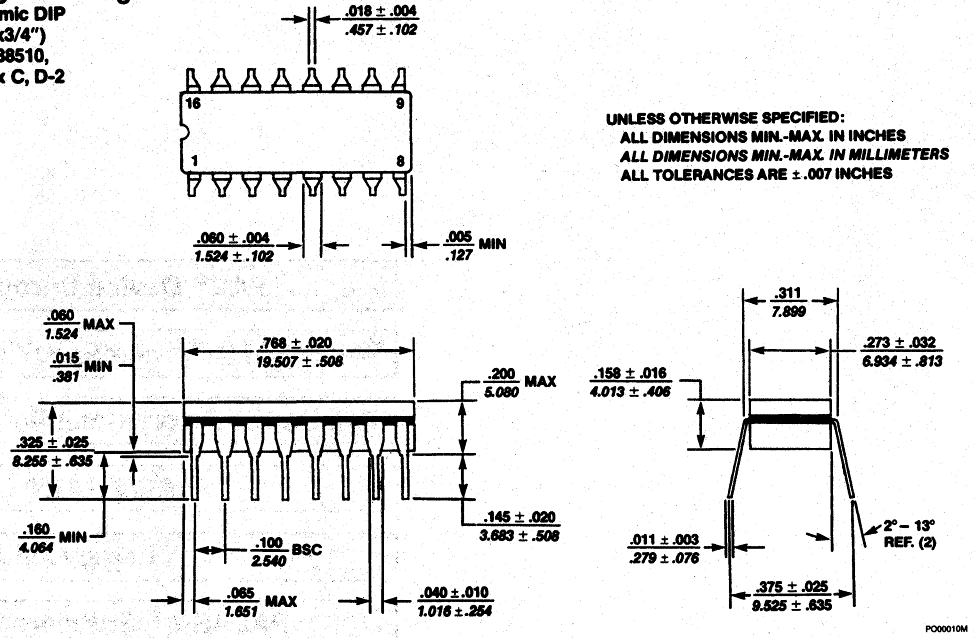
If, in the input columns, a row contains only the symbols H, L, and/or X, this means the indicated output is valid whenever the input configuration is achieved and regardless of the sequence in which it is achieved. The output persists as long as the input configuration is maintained.

If, in the input columns, a row contains H, L, and/or X together with ↑ and/or ↓, this means the output is valid whenever the input configuration is achieved and the indicated transition has occurred (the transition(s) must occur following the achievement of the steady-state levels). If the output is shown as a level (H, L, Q_0, or $\overline{Q}_0$), it persists as long as the steady-state input levels and the levels that terminate indicated transitions are maintained. Unless otherwise indicated, input transitions in the opposite direction to those shown have no effect at the output.


IMPORTANT NOTES

Package Drawings

16J Ceramic DIP
(5/16"x3/4")
Mil-M-38510,
Appendix C, D-2



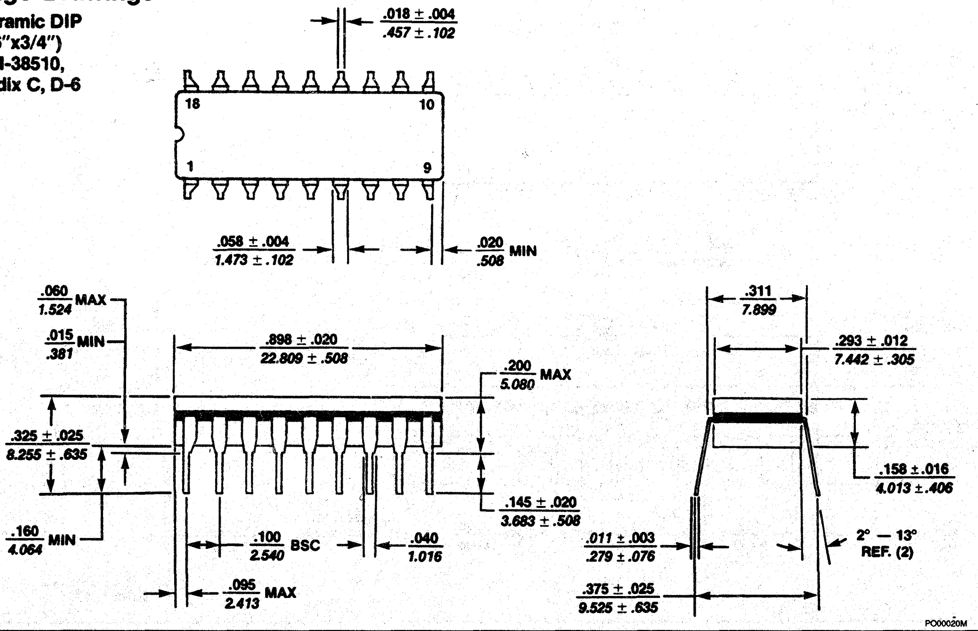
UNLESS OTHERWISE SPECIFIED:
ALL DIMENSIONS MIN.-MAX. IN INCHES
ALL DIMENSIONS MIN.-MAX. IN MILLIMETERS
ALL TOLERANCES ARE ± .007 INCHES

Notes:

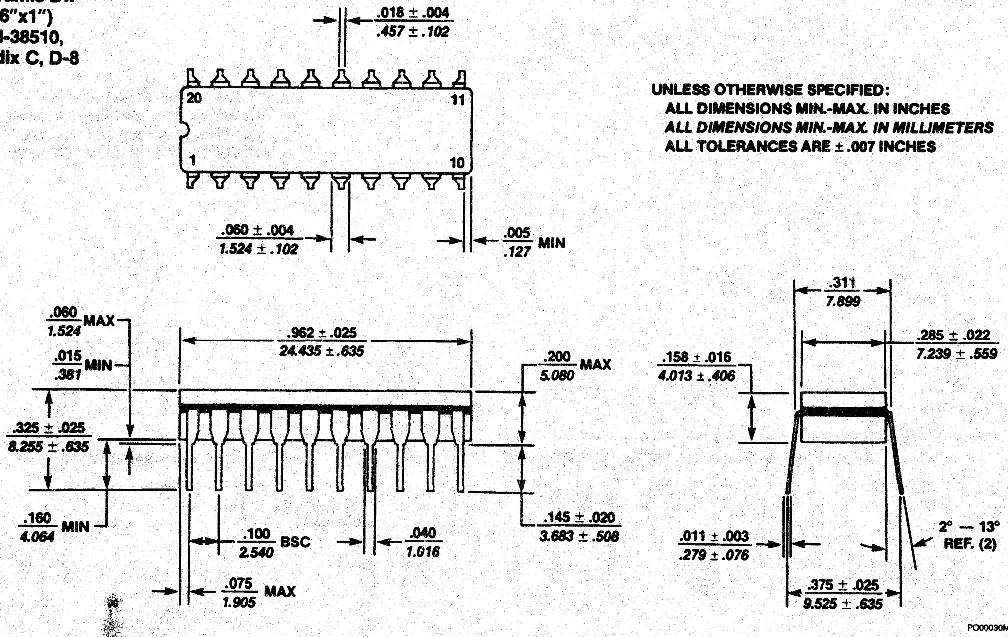
1. Specified body dimensions allow for differences between SSI, MSI and LSI packages.
2. Lead material tolerances are for tin plate finish only. Solder dip finish adds 2 – 10 mils thickness to all lead tip dimensions.

Package Drawings

18J Ceramic DIP (5/16"x3/4") Mil-M-38510, Appendix C, D-6



20J Ceramic DIP (5/16"x1") Mil-M-38510, Appendix C, D-8

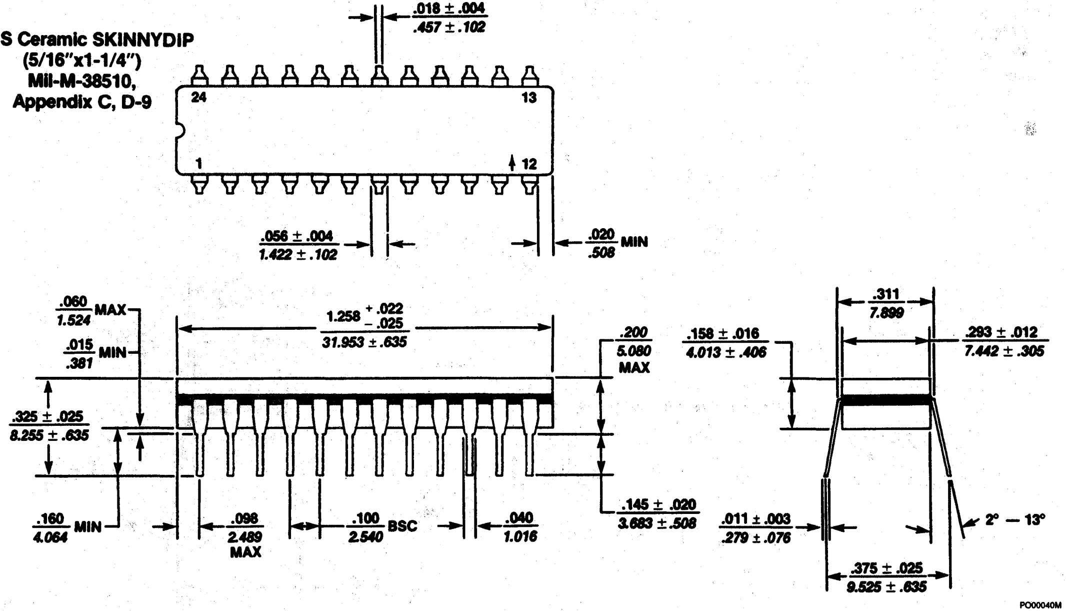


UNLESS OTHERWISE SPECIFIED:
ALL DIMENSIONS MIN.-MAX. IN INCHES
ALL DIMENSIONS MIN.-MAX. IN MILLIMETERS
ALL TOLERANCES ARE ± .007 INCHES

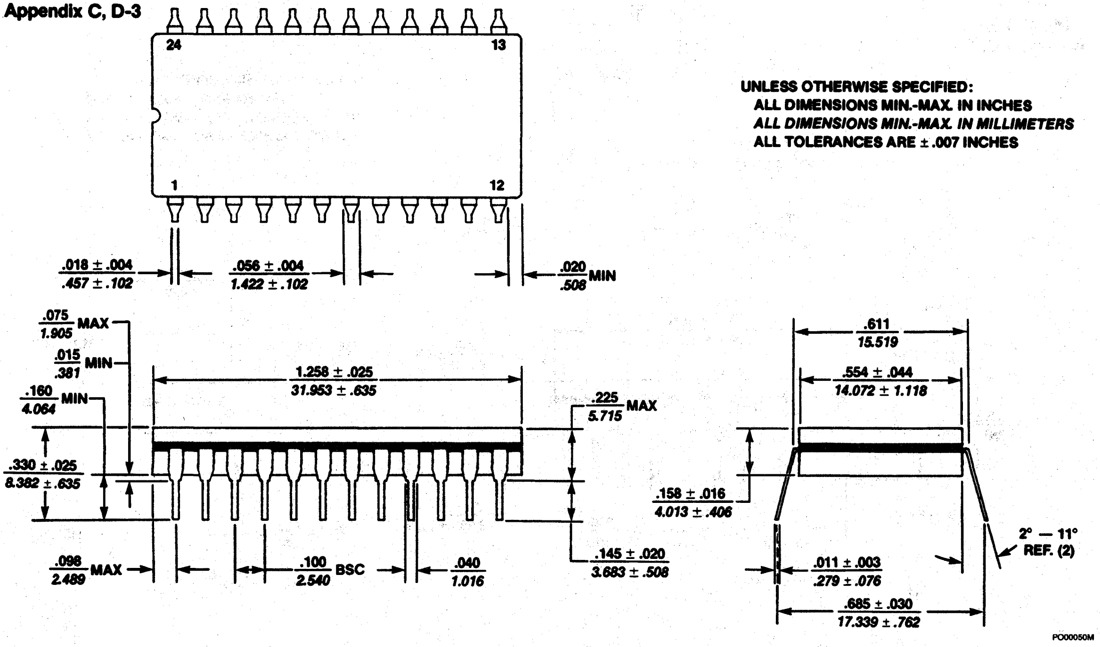
Notes:
1. Specified body dimensions allow for differences between SSI, MSI and LSI packages.
2. Lead material tolerances are for tin plate finish only. Solder dip finish adds 2 – 10 mils thickness to all lead tip dimensions.

Package Drawings

24JS Ceramic SKINNYDIP
(5/16"x1-1/4")
Mil-M-38510,
Appendix C, D-9



24J Ceramic DIP
Mil-M-38510,
Appendix C, D-3

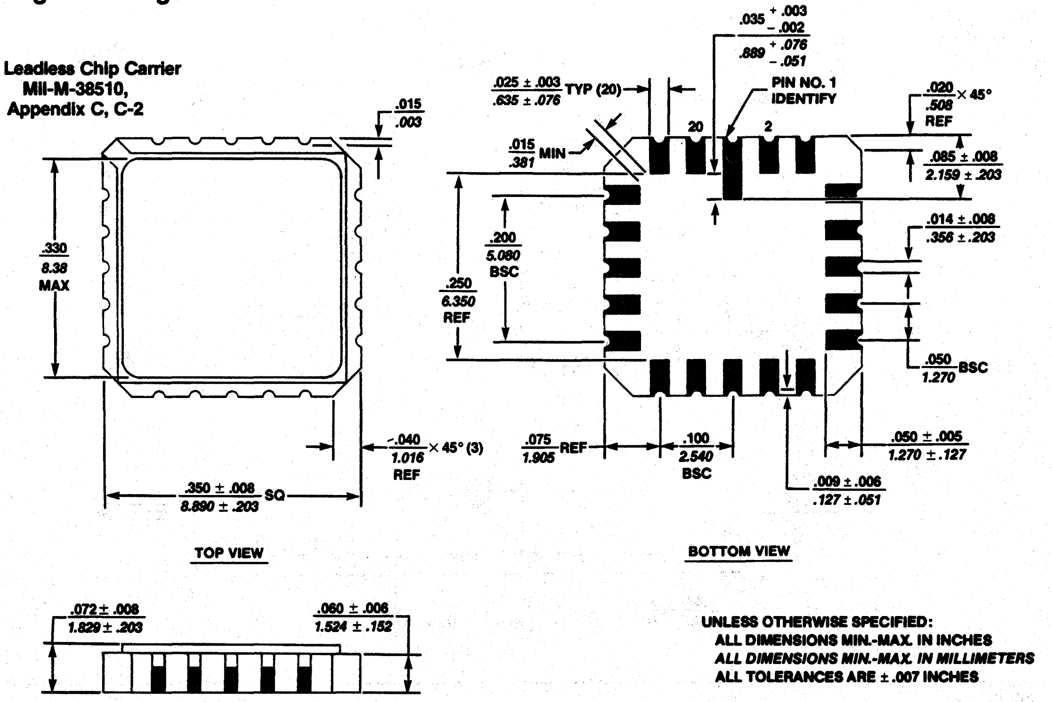


UNLESS OTHERWISE SPECIFIED:
ALL DIMENSIONS MIN.-MAX. IN INCHES
ALL DIMENSIONS MIN.-MAX. IN MILLIMETERS
ALL TOLERANCES ARE ± .007 INCHES

Notes:
1. Specified body dimensions allow for differences between MSI and LSI packages.
2. Lead material tolerances are for tin plate finish only. Solder dip finish adds 2 – 10 mils thickness to all lead tip dimensions.

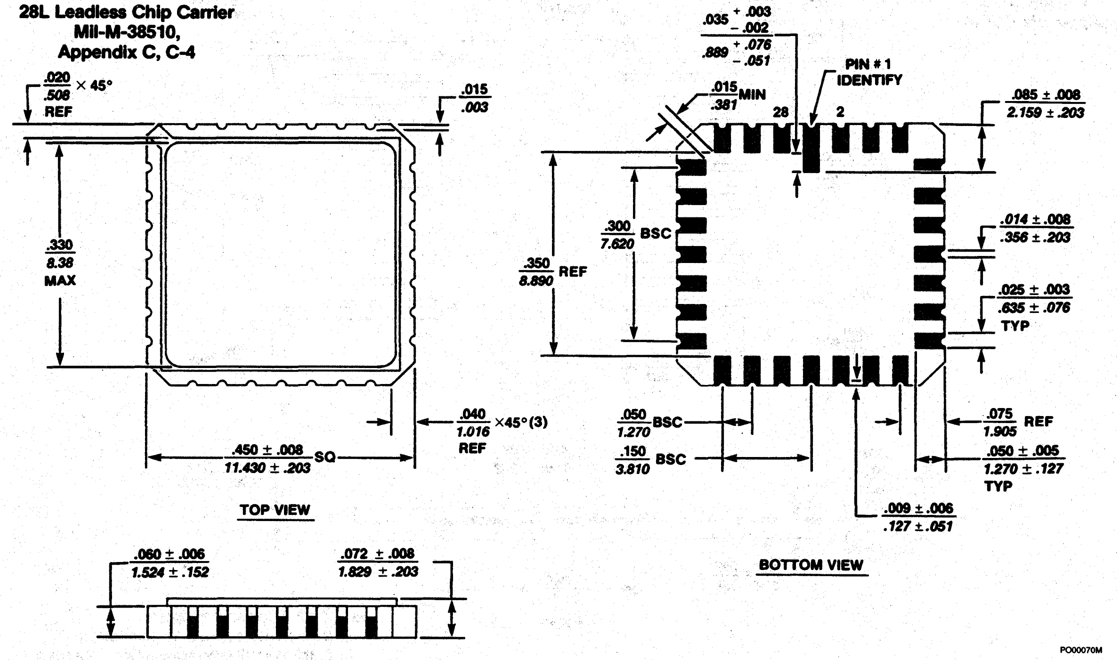
Package Drawings

20L Leadless Chip Carrier Mil-M-38510, Appendix C, C-2



UNLESS OTHERWISE SPECIFIED:
ALL DIMENSIONS MIN.-MAX. IN INCHES
ALL DIMENSIONS MIN.-MAX. IN MILLIMETERS
ALL TOLERANCES ARE ± .007 INCHES

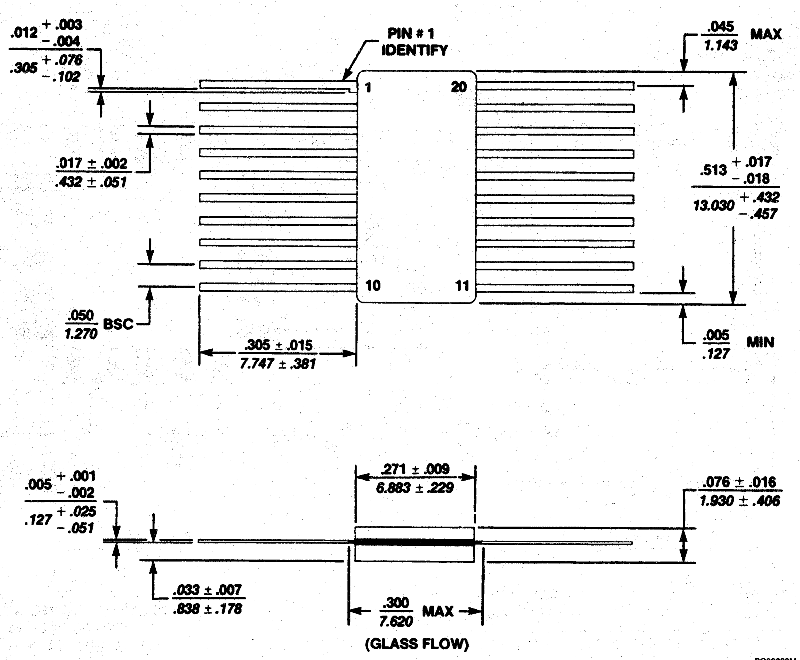
28L Leadless Chip Carrier Mil-M-38510, Appendix C, C-4



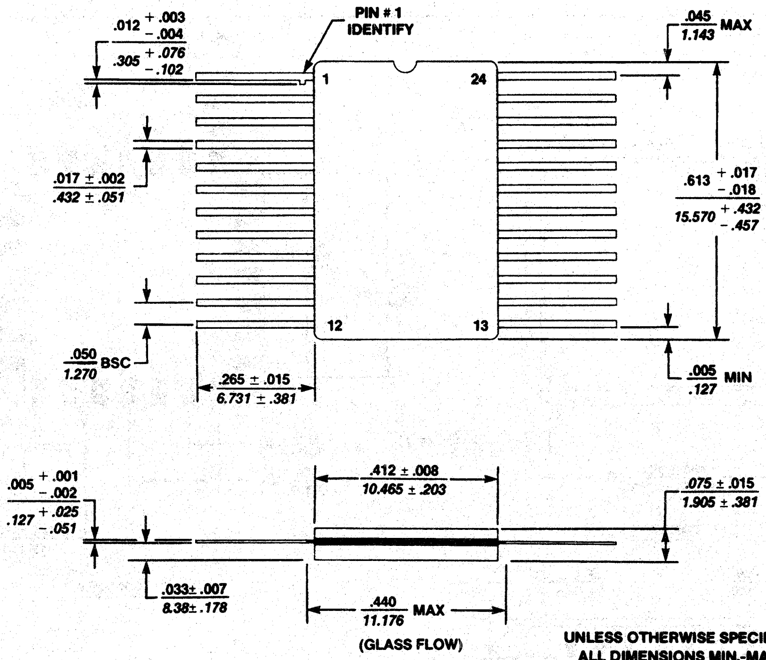
Notes:
1. Solder fillets on lid edges not shown.

Package Drawings

20W Cerpack
Mil-M-38510, Appendix C, F-9



24W Cerpack
Mil-M-38510, Appendix C, F-6

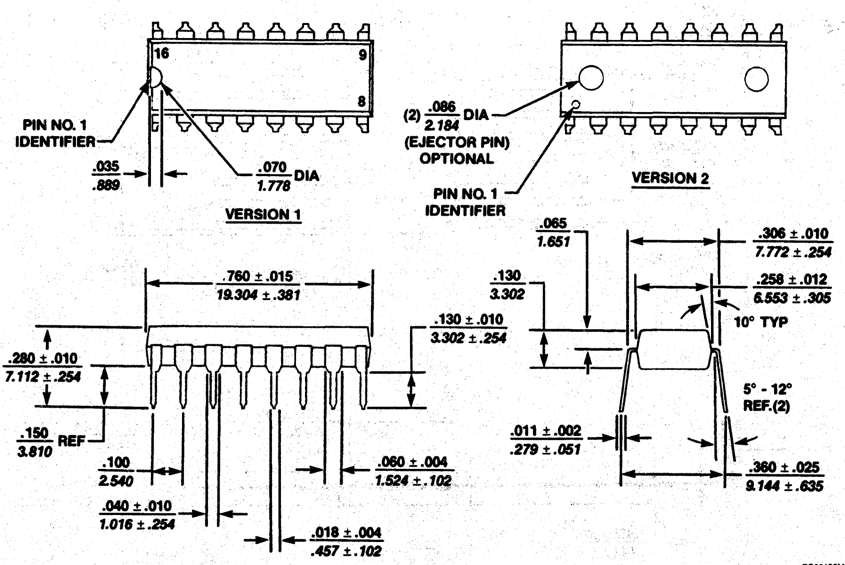


UNLESS OTHERWISE SPECIFIED:
ALL DIMENSIONS MIN.-MAX. IN INCHES
ALL DIMENSIONS MIN.-MAX. IN MILLIMETERS
ALL TOLERANCES ARE ± .007 INCHES

PO00090M

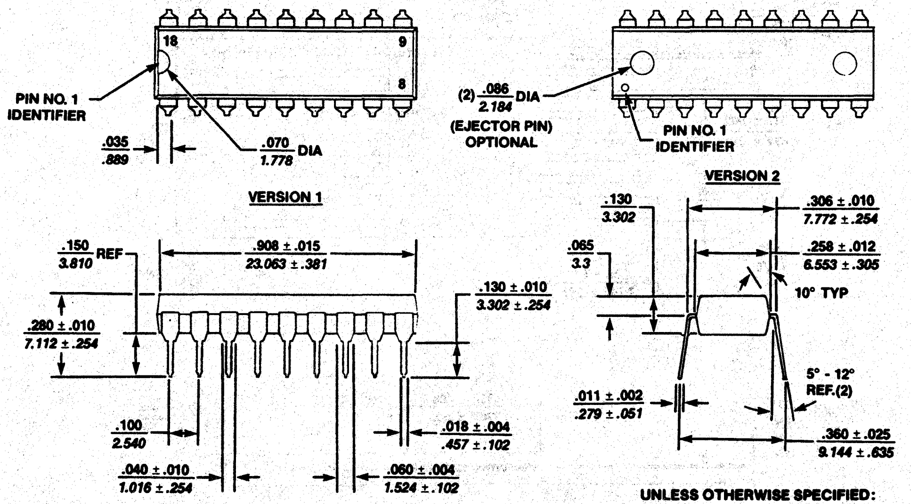
Package Drawings

16N Molded DIP
(1/4"x3/4")



PO00100M

18N Molded DIP
(1/4"x7/8")



UNLESS OTHERWISE SPECIFIED:
ALL DIMENSIONS MIN.-MAX. IN INCHES
ALL DIMENSIONS MIN.-MAX. IN MILLIMETERS
ALL TOLERANCES ARE ± .007 INCHES

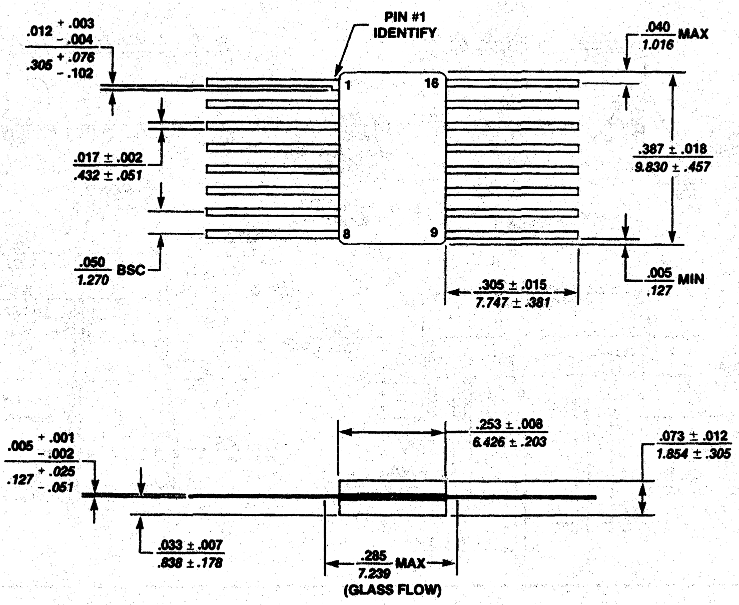
PO00110M

Notes:

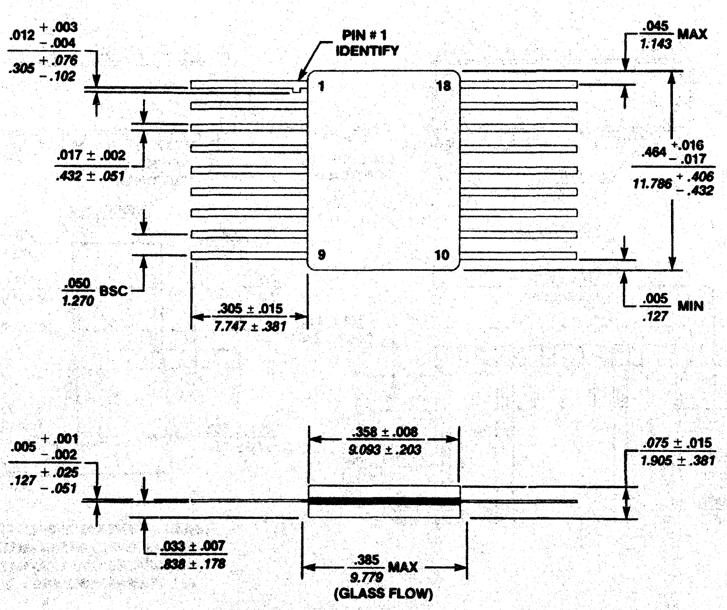
1. Lead material tolerances are for tin plate finish only. Solder dip finish adds 2 – 10 mils thickness to all lead tip dimensions.
2. Both Version 1 and Version 2 configurations are manufactured interchangeably.
3. Ejector pin marks on Version 1 are optional.

Package Drawings

16W Cerpack
Mil-M-38510,
Appendix C, F-5



18W Cerpack

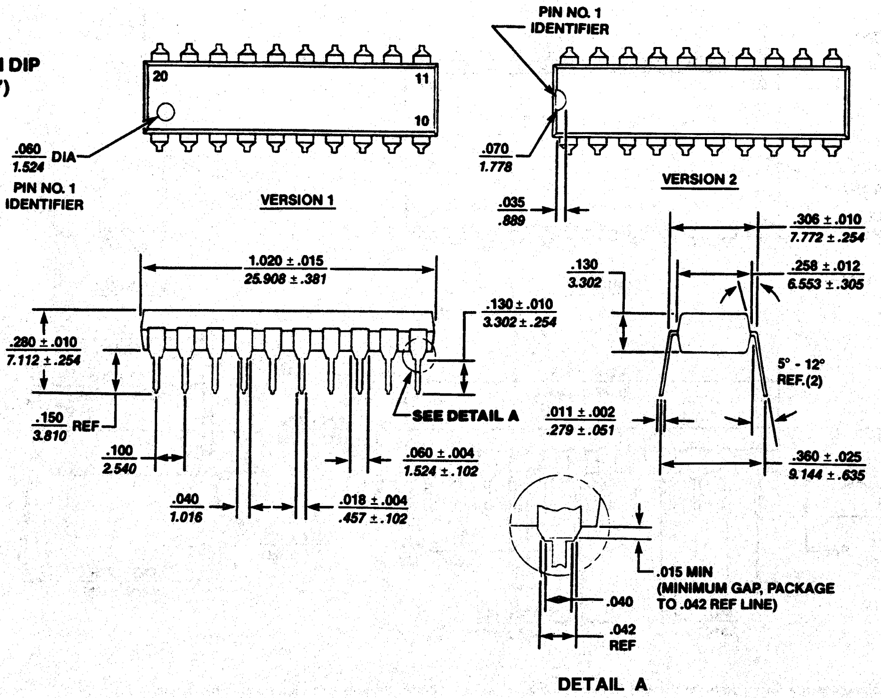


UNLESS OTHERWISE SPECIFIED:
ALL DIMENSIONS MIN.-MAX. IN INCHES
ALL DIMENSIONS MIN.-MAX. IN MILLIMETERS
ALL TOLERANCES ARE ± .007 INCHES

PO00130M

Package Drawings

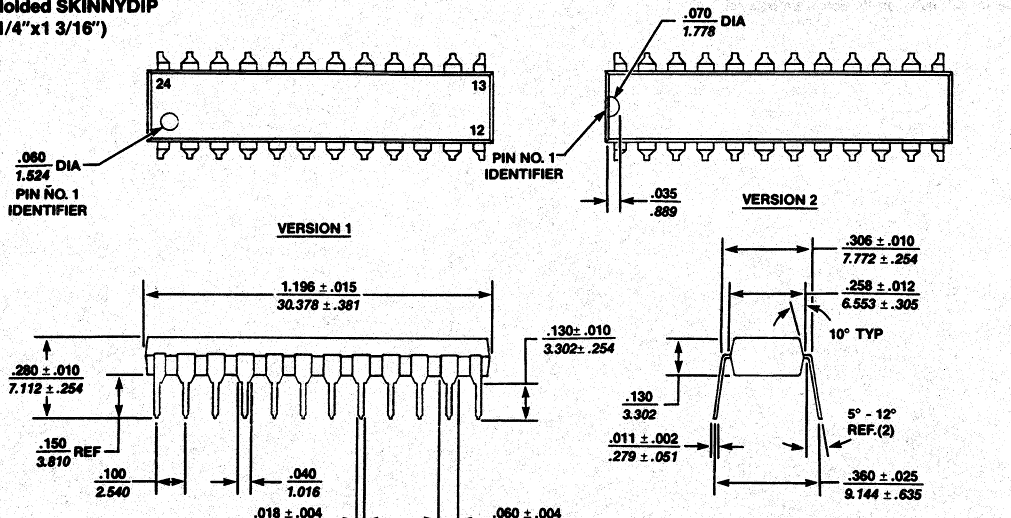
20N Molded DIP
(1/4"x1")



DETAIL A

PO00140M

24NS Molded SKINNYDIP
(1/4"x1 3/16")



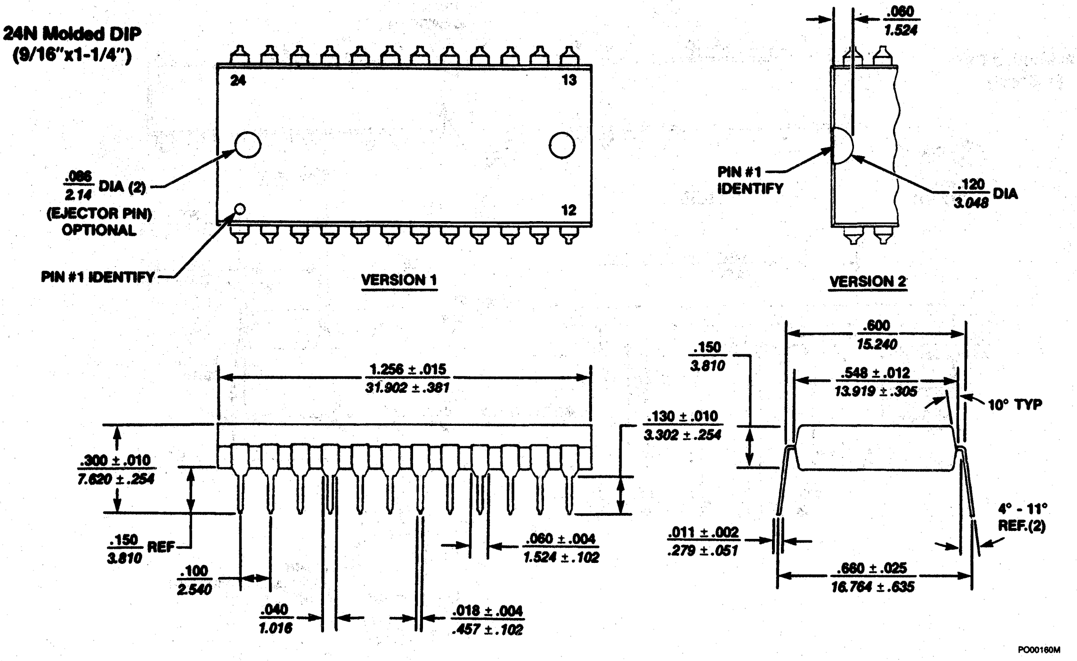
UNLESS OTHERWISE SPECIFIED:
ALL DIMENSIONS MIN.-MAX. IN INCHES
ALL DIMENSIONS MIN.-MAX. IN MILLIMETERS
ALL TOLERANCES ARE ± .007 INCHES

PO00150M

Notes:
1. Lead material tolerances are for tin plate finish only. Solder dip finish adds 2 – 10 mils thickness to all lead tip dimensions.
2. Both Version 1 and Version 2 configurations are manufactured interchangeably.
3. Ejector pin marks on Version 1 are optional.

Package Drawings

24N Molded DIP (9/16"x1-1/4")

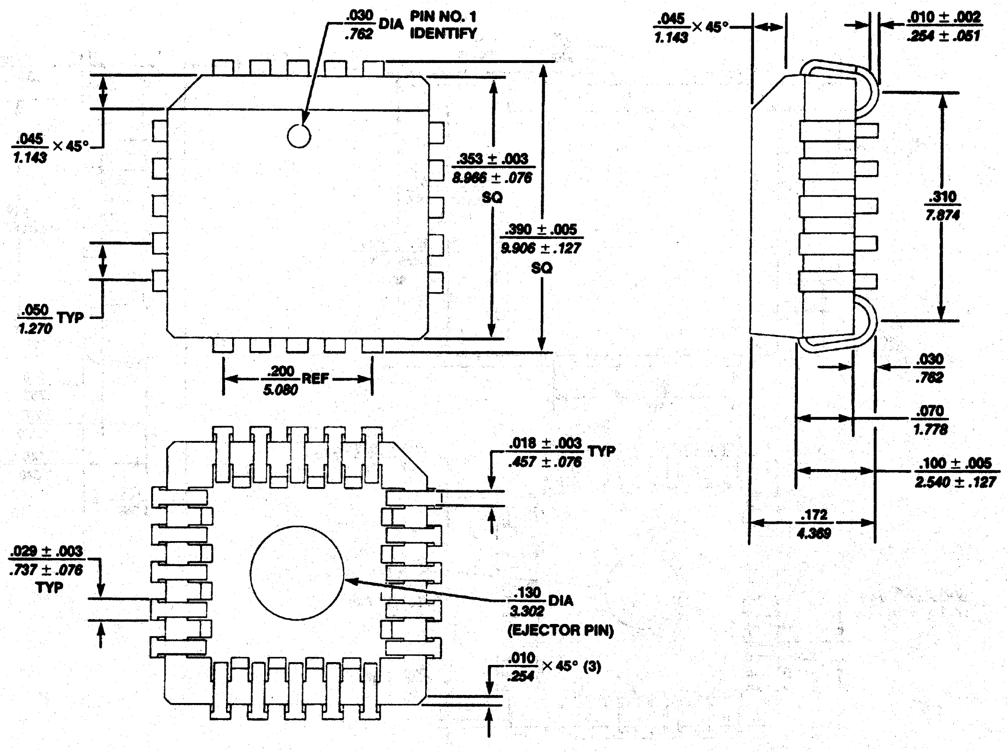


Notes:

1. Lead material tolerances are for tin plate finish only. Solder dip finish adds 2 – 10 mils thickness to all lead tip dimensions.
2. Both Version 1 and Version 2 configurations are manufactured interchangeably.
3. Ejector pin marks on Version 1 are optional.

Package Drawings

20NL Molded Chip Carrier
(.351"x.351")

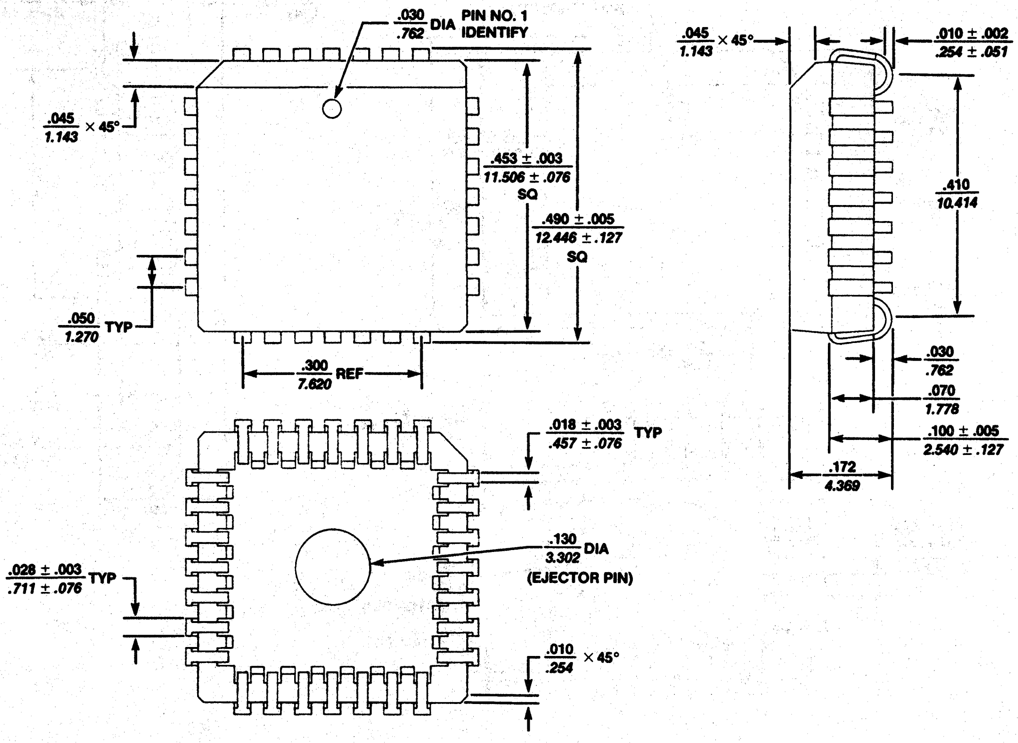


UNLESS OTHERWISE SPECIFIED:
ALL DIMENSIONS MIN.-MAX. IN INCHES
ALL DIMENSIONS MIN.-MAX. IN MILLIMETERS
ALL TOLERANCES ARE ± .007 INCHES

PO00170M

Package Drawings

28NL Molded Chip
(.451"x.451")

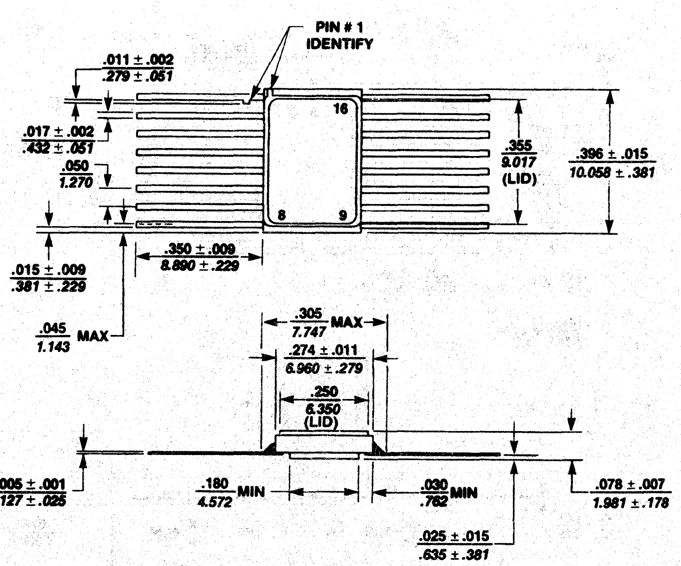


UNLESS OTHERWISE SPECIFIED:
ALL DIMENSIONS MIN.-MAX. IN INCHES
ALL DIMENSIONS MIN.-MAX. IN MILLIMETERS
ALL TOLERANCES ARE ±.007 INCHES

PO00180M

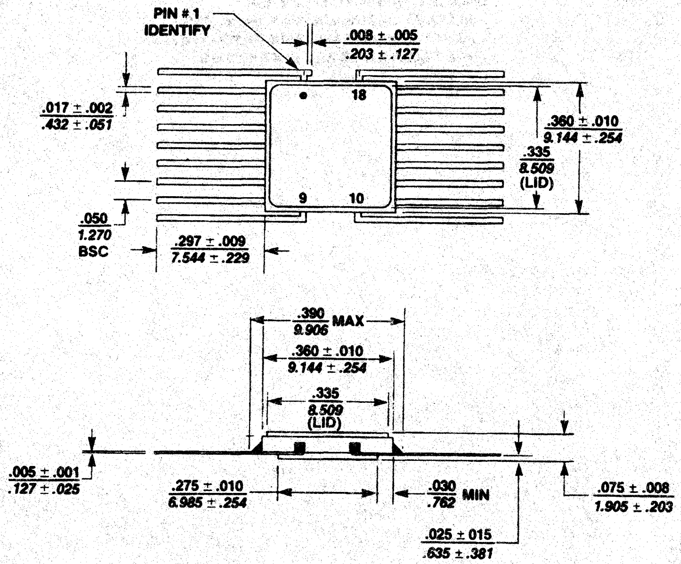
Package Drawings

16F-4/5 Flat Pack
(1/4"x3/8")



PO00190M

18F-2/3 Flat Pack
(3/8"x3/8")

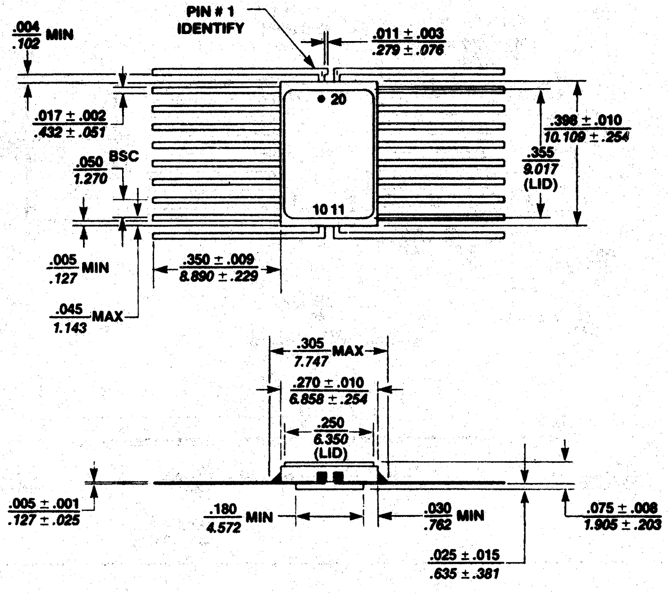


UNLESS OTHERWISE SPECIFIED:
ALL DIMENSIONS MIN.-MAX. IN INCHES
ALL DIMENSIONS MIN.-MAX. IN MILLIMETERS
ALL TOLERANCES ARE ±.007 INCHES

PO00200M

Package Drawings

20F-3 Flat Pack
(1/4"x3/8")



UNLESS OTHERWISE SPECIFIED:
ALL DIMENSIONS MIN.-MAX. IN INCHES
ALL DIMENSIONS MIN.-MAX. IN MILLIMETERS
ALL TOLERANCES ARE ±.007 INCHES

PO00210M

Monolithic Memories Area and Regional Sales Managers

Arizona	
Phoenix	
Ron Scarfo	(602) 971-7997
California	
San Jose	
George Anderl	(408) 249-7766
Mark Lunsford	(408) 249-7766
Lou Scalzo	(408) 249-7766
Santa Ana	
Bernie Brafman	(714) 543-8664
Mike Vogel	(714) 543-8664
Florida	
Longwood	
Jim McGrath	(305) 682-6708

Georgia	
Norcross	
Tom Lewis	(404) 447-0006
Illinois	
Naperville	
Dick Jones	(312) 961-9200
Massachusetts	
Framingham	
Jack Abbott	(617) 875-7373
Russ French	(617) 875-7373
Daniel Kinsella	(617) 875-7373
Bob Norling	(617) 875-7373
Minnesota	
Edina	
Alex Sherbanenko	(612) 829-7343

New Jersey	
Cherry Hill	
Bruce Berlet	(609) 751-1288
Scott Dunlop	(609) 751-1288
Ken Toney	(609) 751-1288
Ohio	
Dayton	
Mike Wier	(513) 439-0470
Oregon	
Medford	
John Charles	(503) 779-8945
Texas	
Dallas	
Ray Gouldsberry	(214) 690-3812
Dennis Prestel	(214) 690-3812
Bob Rainwater	(214) 690-3812

November 13, 1986

Monolithic Memories Representatives

U.S.A.	
Alabama	
Huntsville	
REP, Inc.	(205) 881-9270
Arizona	
Scottsdale	
Summit Sales	(602) 998-4850
California	
Canoga Park	
Bager Electronics	(818) 712-0011
Fountain Valley	
Bager Electronics	(714) 957-3367
San Diego	
Littlefield & Smith	(619) 455-0055
Santa Clara	
Criterion	(408) 988-6300
Colorado	
Wheatridge	
Waugaman Assoc.	(303) 423-1020
Connecticut	
Wallingford	
Comp Rep Associates	(203) 269-1145
Florida	
Deerfield Beach	
Sales Engineering Concepts Inc.	(305) 426-4601
Altamonte Springs	
Sales Engineering Concepts Inc.	(305) 682-4800
Georgia	
Tucker	
REP, Inc.	(404) 938-4358
Illinois	
Rolling Meadows	
Sumer	(312) 991-8500
Indiana	
Indianapolis	
DeVoe Co.	(317) 842-3245
Iowa	
Cedar Rapids	
S & O Sales	(319) 393-1845
Kansas	
Olathe	
Rush and West	(913) 764-2700
Maryland	
Baltimore	
Conroy Sales	(301) 296-2444
Massachusetts	
Westwood	
Comp Rep Associates	(617) 329-3454
Michigan	
Grosse Point Park	
Greiner Associates	(313) 499-0188
Minnesota	
Edina	
Mel Foster Tech Sales	(612) 941-9790
Missouri	
Ballwin	
Rush and West	(314) 394-7271
New Jersey	
Haddonfield	
Tritek Sales, Inc.	(609) 429-1551
Teaneck	
Technical Marketing Group	(201) 692-0200
New Mexico	
Albuquerque	
BFA Corporation	(505) 292-1212
New York	
East Rochester	
Tri-Tech Electronics, Incorporated	(716) 385-6500
Endwell	
Tri-Tech Electronics, Incorporated	(607) 754-1094
Fayetteville	
Tri-Tech Electronics, Incorporated	(315) 446-2881
Fishkill	
Tri-Tech Electronics, Incorporated	(914) 897-5611
Melville	
Technical Marketing Group	(516) 351-8833
North Carolina	
Charlotte	
REP, Inc.	(704) 563-5554
Raleigh	
REP, Inc.	(919) 851-3007
Ohio	
Cincinnati	
Makin Associates	(513) 871-2424
Columbus	
Makin Associates	(614) 481-8898
Solon	
Makin Associates	(216) 248-7370
Oklahoma	
Tulsa	
West Associates	(918) 665-3465
Oregon	
Portland	
Northwest Marketing	(503) 620-0441
Puerto Rico	
Caguas	
Comp Rep Associates	(809) 746-6550
Tennessee	
Jefferson City	
REP, Inc.	(615) 475-9012
Texas	
Austin	
West Associates	(512) 454-3681
Richardson	
West Associates	(214) 680-2800
Houston	
West Associates	(713) 621-5983
Utah	
Salt Lake City	
Waugaman Assoc.	(801) 261-0802
Washington	
Bellevue	
Northwest Marketing	(206) 455-5846
Wisconsin	
Brookfield	
Sumer	(414) 784-6641
CANADA	
British Columbia	
Vancouver	
Davetek Marketing	(604) 430-3680
Ontario	
Brampton	
Cantec	(416) 791-5922
Ottawa	
Cantec	(613) 725-3704
Quebec	
Dollard Des Ormeaux	
Cantec	(514) 683-6131

Monolithic Memories World-Wide Applications Support

U.S.A.

California

Canoga Park

Michael Sholklapper	(818) 710-0664

Granada Hills

Mark Kovalik	(818) 341-7257

San Jose

Shiri Kadambi	(408) 249-7766
Joe Kotas	(408) 249-7766
Masood Shakeel	(408) 249-7766

Santa Ana

Scott Thomas	(714) 543-8664
Bill McNamara	(714) 543-8664

Georgia

Norcross

Mark Reynolds	(404) 447-0006
Ian Scott	(404) 447-0006

Illinois

Naperville

Chris Belanger	(312) 961-9200

Massachusetts

Framingham

Gary Smith	(617) 875-7373
Bob Norling	(617) 875-7373
Mike Gridley	(617) 875-7373
Dan Kinsella	(617) 875-7373

New Jersey

Steven Traum	(609) 751-1288
Scott Dunlop	(609) 751-1288
Paul J. Koep	(609) 751-1288

Ohio

Dayton

Bill Hollon	(513) 866-8928

Texas

Barry Seidner	(214) 690-3812

EUROPE

England

Harry Hughes	44-252-517431
Steve Scard	44-252-517431
Simon Moran-Smith	44-252-517431

France

Jose Juntas	46-8734-62
Michel Rolland	46-8734-62
Marc Guyonnet	46-8734-62

Germany

Peter Reidle	49-89-984961
Willy Voldan	

JAPAN

Tokyo

Tervo Saitoh	81-3-207-3131
Hide Imatomi	81-3-207-3131
Sakyo Nagashima	81-3-207-3131
Mitsvaki Ukiya	81-3-207-3131

Singapore

Ian Meikle	65-225-7544

Monolithic Memories Franchised Distributors

U.S.A.

Alabama

Huntsville	
Marshall Electronics Group	(205) 881-9235
Arrow Electronics	(205) 837-6955
Kierulff Electronics	(205) 883-6070

Arizona

Phoenix	
Kierulff Electronics	(602) 437-0750
Tempe	
Anthem Electronics	(602) 966-6600
Arrow Electronics	(602) 968-4800
Marshall Electronics Group	(602) 968-6181

California

Agoura	
Image Electronics	(818) 707-0911
Canoga Park	
Marshall Electronics Group	(818) 509-0001
Chatsworth	
Anthem Electronics	(818) 700-1000
Arrow Electronics	(818) 701-7500
Marshall Electronics Group	(818) 407-4100
Kierulff Electronics	(213) 725-0325
Cypress	
Kierulff Electronics	(714) 220-6566
El Monte	
Marshall Electronics Group	(818) 442-7204
Irvine	
Marshall Electronics Group	(714) 458-5311
Anthem	(714) 768-4444
Milpitas	
Marshall Electronics Group	(408) 943-4600
Hayward	
Arrow Electronics	(415) 487-4600
Sacramento	
Arrow Electronics	(916) 925-7456
Anthem	(916) 922-6800
Rancho Cordova	
Marshall Electronics Group	(916) 635-9700
San Diego	
Anthem Electronics	(619) 453-4871
Arrow Electronics	(619) 565-4800
Kierulff Electronics	(619) 278-2112
Marshall Electronics Group	(619) 578-9600
San Jose	
Anthem Electronics	(408) 295-4200
Kierulff Electronics	(408) 971-2600
Sunnyvale	
Arrow Electronics	(408) 745-6600
Tustin	
Arrow Electronics	(714) 669-4524
Image Electronics	(714) 259-0900
Kierulff Electronics	(714) 731-5711

Colorado

Aurora	
Arrow Electronics	(303) 696-1111
Englewood	
Anthem Electronics	(303) 790-4500
Kierulff Electronics	(303) 790-4444
Thornton	
Marshall Electronics Group	(303) 451-8383

Connecticut

Meriden	
Lionex Corporation	(203) 237-2282
Wallingford	
Arrow Electronics	(203) 265-7741
Kierulff Electronics	(203) 265-1115
Marshall Electronics Group	(203) 265-3822

Florida

Clearwater	
Arrow Electronics	(813) 576-8995
Deerfield Beach	
Arrow Electronics	(305) 429-8200
Fort Lauderdale	
Kierulff Electronics	(305) 486-4004
Marshall Electronics Group	(305) 982-0661
Orlando	
Marshall Electronics Group	(305) 841-1878
Palm Bay	
Arrow Electronics	(305) 725-1480
St. Petersburg	
Kierulff Electronics	(813) 576-1966

Georgia

Norcross	
Arrow Electronics	(404) 449-8252
Kierulff Electronics	(404) 447-5252
Marshall Electronics Group	(404) 923-5750

Illinois

Itasca	
Kierulff Electronics	(312) 250-0500
Schaumburg	
Arrow Electronics	(312) 397-3440
Marshall Electronics Group	(312) 490-0155

Indiana

Indianapolis	
Arrow Electronics	(317) 243-9353
Marshall Electronics Group	(317) 297-0483

Kansas

Lenexa	
Arrow Electronics	(913) 541-9542
Marshall Electronics Group	(913) 492-3121

Maryland

Columbia	
Arrow Electronics	(301) 995-0003
Lionex	(301) 964-0040
Gaithersburg	
Marshall Electronics Group	(301) 840-9450
Kierulff Electronics	(301) 840-1155
Linthicum	
Kierulff Electronics	(301) 636-5800

Massachusetts

Billerica	
Kierulff Electronics	(617) 667-8331
Burlington	
Marshall Electronics Group	(617) 272-8200
Wilmington	
Lionex Corporation	(617) 657-5170
Woburn	
Arrow Electronics	(617) 933-8130

Michigan

Ann Arbor	
Arrow Electronics	(313) 971-8220
Grand Rapids	
Arrow Electronics	(616) 243-0912
Livonia	
Marshall Electronics Group	(313) 525-5850

Minnesota

Edina	
Arrow Electronics	(612) 830-1800
Kierulff Electronics	(612) 941-7500
Plymouth	
Marshall Electronics Group	(612) 559-2211

Missouri

St. Louis	
Arrow Electronics	(314) 567-6888
Kierulff Electronics	(314) 739-0855

New Hampshire

Manchester	
Arrow Electronics	(603) 668-6968

New Jersey

Clifton	
Vantage Electronics	(201) 777-4100
Fairfield	
Arrow Electronics	(201) 575-5300
Kierulff Electronics	(201) 575-6750
Lionex	(201) 227-7960
Marshall Electronics Group	(201) 882-0320

Mt. Laurel	
Kierulff Electronics	(609) 235-1618
Marshall Electronics Group	(609) 234-9100
Marlton	
Arrow Electronics	(609) 596-8000

New Mexico

Albuquerque	
Arrow Electronics	(505) 243-4566

New York

Buffalo	
Summit Distributors	(716) 884-3450
Johnson City	
Marshall Electronics Group	(607) 798-1611
Hauppauge	
Arrow Electronics	(516) 231-1000
Current Components	(516) 272-2600
Lionex	(516) 273-1660
Marshall Electronics Group	(516) 273-2424
Liverpool	
Arrow Electronics	(315) 652-1000
Melville	
Arrow Electronics	(516) 694-6800
Rochester	
Arrow Electronics	(716) 427-0300
Marshall Electronics Group	(716) 235-7620
Summit Distributors	(716) 334-8118

North Carolina

Raleigh	
Arrow Electronics	(919) 876-3132
Kierulff Electronics	(919) 872-8410
Marshall Electronics Group	(919) 878-9882
Quality Components	(919) 876-7767

Ohio

Centerville	
Arrow Electronics	(513) 435-5563
Cleveland	
Kierulff Electronics	(216) 587-6558
Dayton	
Marshall Electronics Group	(513) 236-8088
Kierulff Electronics	(513) 439-0045
Solon	
Arrow Electronics	(216) 248-3990
Marshall Electronics Group	(216) 248-1788

Oklahoma

Tulsa	
Arrow Electronics	(918) 665-7700
Kierulff Electronics	(918) 252-7537
Quality Components	(918) 664-8812

Oregon

Beaverton	
Almac Electronics	(503) 629-8090
Marshall Electronics Group	(503) 644-5050
Lake Oswego	
Anthem Electronics	(503) 684-2661
Tigard	
Arrow Electronics	(503) 684-1690

Pennsylvania

Horsham	
Lionex Corporation	(215) 443-5150
Monroeville	
Arrow Electronics	(412) 856-7000

Texas

Addison	
Quality Components	(214) 733-4300
Austin	
Arrow Electronics	(512) 835-4180
Kierulff Electronics	(512) 835-2090
Quality Components	(512) 835-0220
Marshall Electronics	(512) 837-1991
Carrollton	
Arrow Electronics	(214) 380-6464
Marshall Electronics Group	(214) 233-5200
Dallas	
Kierulff Electronics	(214) 343-2400
Houston	
Arrow Electronics	(713) 530-4700
Kierulff Electronics	(713) 530-7030
Marshall Electronics	(713) 895-9200
Sugarland	
Quality Components	(713) 240-2255

Utah

Salt Lake City	
Kierulff Electronics	(801) 973-6913
Arrow Electronics	(801) 972-0404
Anthem Electronics	(801) 973-8555

Washington

Bellevue	
Almac Electronics Corporation	(206) 643-9992
Arrow Electronics	(206) 643-4800
Bellingham	
RAE Electronics	(607) 291-8866
Redmond	
Anthem Electronics	(206) 881-0850
Kent	
Kierulff Electronics	(206) 575-4420

Wisconsin

Brookfield	
Arrow Electronics	(414) 792-0150
Waukesha	
Kierulff Electronics	(414) 784-8160

CANADA

Alberta

Calgary	
Zentronics Limited	(403) 272-1021

British Columbia

Richmond	
Zentronics Limited	(604) 273-5575
Vancouver	
RAE Electronics	(604) 291-8866

Manitoba

Winnipeg	
Zentronics Limited	(204) 694-1957

Ontario

Brampton	
Zentronics Limited	(416) 451-9600
Nepean	
Zentronics Limited	(613) 226-8840

Quebec

Montreal	
Arrow Electronics	(514) 735-5511
Prelco Electronics	(514) 389-8051
St. Laurent	
Zentronics Limited	(514) 737-9700

Monolithic Memories Overseas Representatives and Distributors

ARGENTINA

Electroquimica Delta
Timoteo Gordillo 72
1408 Buenos Aires
Argentina
Phone: 9-011-641-3193 or 641-0449
Telex: 21212 AR EDELTA

AUSTRALIA

R & D Electronics Pty Ltd.
4 Florence St.
Burwood, Vic. 3125
Phone: 61-3-288-8911
Telex: AA33288
Fax: 61-3-288-9168

R & D Electronics Pty Ltd.
P.O. Box 57
Crows Nest, NSW 2065
Phone: 61-2-439-5488
Telex: AA25468

AUSTRIA

Ing. Steiner
Hummelgasse 14
1130 Wien
Phone: 9-011-32-22-8274740
Telex: 135026

BELGIUM

D & D Electronics PVBA
7E Olympiadelaan 93
2020 Antwerp
Phone: 03-23-8277934
Telex: 73121

BRAZIL

Itautec Componentes S.A.
Largo de Arouche No. 24
9th Andar
01219 Sao Paulo
Phone: 222-9200
Telex: 13087
Fax: 2783043

DENMARK

C-88
Kokkedal Industripark 42A
DK-2980 Kokkedal-Denmark
Phone: 2-244888
Telex: 41198 CEIGTY DK

ENGLAND

Monolithic Memories Ltd.
Monolithic House
1 Queens Road
Farnborough
Hampshire
GU14 6DJ
Phone: (0252) 517431
Telex: 858051 MONO UK G
Fax: 44-252-521041

Analog Devices Ltd.
Central Avenue
East Molesey
Surrey KT8 OSN
Phone: 01-941-1066
Telex: 929962 ANALOGG

Analog Devices Ltd.
Executive House
South Road
Harlow
Essex CM202BX
Phone: 9-011-44-24-941-8611
Telex: 817169

Analog Devices Ltd.
5th Floor Hagley House
Hagley Road
Edgbaston
Birmingham B168QG
Phone: 021-455-9395
Telex: 339752 ANALOG G

Macro Marketing Ltd.
396 Bath Road
Slough
Berkshire
Phone: 9-011-44-6286-63011
Telex: 847083

Microlog Ltd.
The Cornerstone
The Broadway
Woking Surrey GU21 5EZ
Phone: (04862) 29551
Telex: 859219 ULOG G

Rapid Recall Ltd.
Rapid House
Denmark Street
High Wycombe
Bucks HP11 2ER
Phone 0494-26271
Telex: 837931 RAPIDG
Fax: 21680

FINLAND

Findip-Elektroniikka
Instrumentarium Elektroniikka
P.O. Box 64
SF-02631-ESPOO
Phone: 358-0-5281
Telex: 124426 HAVUL SF

FRANCE

Monolithic Memories France S.A.R.L.
8, Rue De L'Esterel
SILIC 463
94613 Rungis Cedex
France
Phone: 46-87-34-62
Telex: 45-60-57-25

Almex
Zone industrielle d'Antony
48 rue de Aubepine
B.P. 102
92164 Antony Cedex
Phone: 9-011-(33)-1-46-66-21-12
Telex: 250067F

Composants S.A.
Avenue Gustave Eiffel
B.P. 81
33605 Pessac Cedex
Phone: (33) 56-36-40-40
Telex: 550696F
FAX: (33) 56-07-64-41

Datadis S.A.
10-12 Rue Emile Landrin
92100 Boulogne
Phone: 9-011-(33)-1-46-05-60-00
Telex: 201905F

Dimel
Le Marino
Ave. Claude Farrere
B.P. 1153
83058 Toulon Cedex
Phone: (33) 94-41-49-63
Telex: 490093F
Fax: (33)1-69-07-05-59

Generim
Zone d'Activities de Courtaboeuf
Avenue de la Baltique
B.P. Box 88
91943 Les Ulis Cedex
Phone: (33) 1-69-07-78-78
Telex: 691700F
Fax: (33) 1-69-07-05-59

Generim
24 Avenue De La Houville Blanche
B.P. 1-38170 Seyssinet
Phone: 9-011-33-1-76-49-491449
Telex: 320000

Jermyn
Zone Silic 585
73/79 Rue de Solets
94663 Rungis Cedex
Phone: 9-011-(33)-45-60-04-00
Telex: 260967F
Fax: (33)1-45-60-05-46

GERMANY

Monolithic Memories, GmbH
Mauerkircherstrasse 4/11
8000 Munich 80
Phone: 0-89-984961
Telex: 524385 MONO D
Fax: 89-983162

Monolithic Memories Overseas Representatives and Distributors

Astronic GmbH
Winzerstrasse 47D
8000 Munich 80
Phone: 089-304011
Telex: 5216187

Dr. Dohrenberg Vertrießs & GmbH
Bayreuther Strasse 3
1000 Berlin 30
Phone: 0-30-2138043
Telex: 184860

Electronic 2000 Vertrießs GmbH
Neumarkter Strasse 57
8000 Munich 82
Phone: 089-434061
Telex: 522561

Nordelektronik GmbH KG
Harksheiderweg 238-240
2085 Quickborn
Phone: 04160-4031
Telex: 214299

Positron Bauelemente Vertriebs GmbH
Benzstrasse 1
Postfach 1140
7016 Gerlingen-Stuttgart
Phone: 07156-23051
Telex: 7245266

Mueller Wuppertal Vertrießs GmbH
Vereinsstr. 17
5600 Wuppertal
Phone: 0202-426016
Telex: 8591543

HONG KONG

CET Ltd.
10/F Hua Hsia Bldg.
64-66 Gloucester Rd.
Hong Kong
Phone: 852-5-200922
Telex: 85148 CET HX
Fax: 5-285764

INDIA

Micro Aids India
26/1 Venkata Krishna Road
RA Puram
Madras 600028
Phone: 75757
Telex: 416517 TRUEIN

Micro Aids International
501D Vandell Way
Campbell, CA 95008
Phone: (408) 866-7000
Telex: 499-3462

IRELAND

Micro Marketing Ltd.
Glenageary Office Park
Glenageary
County Dublin
Eire
Phone: 856325
Telex: 31584 (MMI EI)

ISRAEL

Aviv Electronics
12 Kehilat Venezia
Tel Aviv 69101
Phone: 9-011-972-3-494450
Telex: 33572 MAVIVIL
Fax: 3-494065

ITALY

Comprel S.P.A.
Viale Fulvio Testio 115
20092 Cinisello Balsamo (MI)
Milano, Italy
Phone: 2-6120641
Telex: 332484

JAPAN

Monolithic Memories Japan KK
Shinju Nomura Shoken Bldg SF
5-17-9 Shinjuku
Shinjuku-Ku
Tokyo 160
Phone: 81-3-207-3131
Telex: 232-3390 MMI KK J
Fax: 81-3-207-3130
81-3-207-3139

Ado Electronic Industrial Co., Ltd.
Bldg. Sasage 7F
4-6, Soto-Kanda, 2-chome
Chiyoda-Ku, Tokyo, Japan 101
Phone: 03-257-1025
FAX: 03-257-1579

Comtecs Co., Ltd.
2-19-7 Higashi-Gotanda
Shinagawa-Ku
Tokyo 141
Phone: (03) 441-7100
Fax: (03) 441-7185

Internix Inc.
Shinjuku Hamada Bldg.
7-4-7 Nishi-Shinjuku
Shinjuku-Ku
Tokyo 160
Phone: (03) 369-1101
Fax: (03) 366-8566

Nihon Denshikizai Co., Ltd.
Sanyo Bldg.
15-22 Hiroshiba-Cho
Suita-Shi
Osaka 564
Phone: (06) 385-6707
Fax: (06) 330-6814

Synderdyne Inc.
Ishibashi Bldg.
1-20-2 Dogenzaka
Shibuya-Ku
Tokyo 150
Phone: (03) 461-9311
Fax: (03) 461-9854

Tokiwa & Company
1-1-10 Omori
Ohta-Ku
Tokyo 143
Phone: (03) 766-6701
Fax: (03) 766-1300

Tokiwa & Company of Osaka
3-8-13 Emisu
Naniwa-Ku
Osaka 556
Phone: (06) 643-3521
Fax: (06) 631-6418

Tokyo Denshi Kagaku-Kizai Co., Ltd.
Dempa Bldg. 3F
2-4-4 Soto Kanda
Chiyoda-Ku
Tokyo 101
Phone: (03) 257-1361
Fax: (03) 255-1978

KOREA

Kortronics Enterprise
Rm 307, 9-Dong, B-Block
#604-l Guro-Dong, Guro-GU
Seoul
Phone: 635-1043
Telex: KORTRONK26759
Fax: 2-6750514

NETHERLANDS

Alcom Electronics B.V.
Postbus 358, 2900 AJ
Capelle A. D. Ijssel Holland
Esse Baan 1
Phone: 010-519533
Telex: 26160

NORWAY

Henaco A/S
P.O. Box 126 Kaldbakken
Trondheimsveien 436 Ammerud
Oslo 9
Phone: 02-162110
Telex: 76716 HENACO

PORTUGAL

Digicontrole
Apartado 2-Sabugo 2715
Pero Pinheiro
Phone: 35-1-292-3924
Telex: 62551 STUREP P

SINGAPORE

MMI Integrated Circuits Pte. Ltd.
19 Keppel Road 11-06
Jit Poh Building
Singapore 0208
Phone: 65-2257544
Telex: RS55650 MMI RS
Fax: 65-2246113

Dynamar Computer Systems (Pte) Ltd.
12 Lorong Bakar Batu #05-11
Kolam Ayer Industrial Park
Singapore 1334
Phone: 65-7476188
Telex: RS26283 DYNAMA
Fax: 65-747-2648

SOUTH AFRICA

Promilect Pty Ltd.
P.O. Box 1194
Randburg 2125
Phone: 9-011-27-11-886-2410
Telex: 424822

SOUTH AMERICA

Key Source
637 East Arques Avenue
Sunnyvale, CA 94086
Phone: 408-730-0607

SPAIN

Amitron S.A.
Avenida De Valladolid, 47A
28008 Madrid
Phone: (34) 1-247-93-13
Telex: 45550 AMIT-E
Fax: (34) 1-248-79-58

Sagitron
Castello 25-2-E
28001 Madrid
Phone: 1-402-6085
Telex: 43819
Fax: (34) 1-275-40-23

SWEDEN

Naxab
Box 4115
S 17104 Solna
Phone: 08-985140
Telex: 17912
Fax: 08-7645451

SWITZERLAND

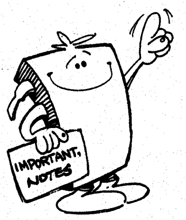
Industrade AG
Gemsenstrasse 2
CH 8021 Zurich
Phone: 01-3632230
Telex: 56788

TAIWAN

Sertek International, Inc.
3FL, #135 Chien Kuo N. Road, Sec 2
Taipei, Taiwan R.O.C. 10479
Phone: 886-2-501-0055
Telex: 23756 SERTEK
Telex: 13579MSCGP
Fax: 2-5012521

IMPORTANT,
NOTES

Notes



Monolithic MMI Memories

Corporate Sales Offices

Americas
Monolithic Memories, Inc.
2175 Mission College Blvd.
Santa Clara, CA 95054-1592
Phone (408) 970-9700
TWX (910) 338-2374
TWX (910) 338-2376
TWX (910) 338-2405
Fax (408) 988-4254

France
Monolithic Memories France S.A.R.L.
Silic 463
F 94613 Rungis Cedex
France
Phone 46-87-34-62
Telex 202146
Fax 1-45-60-57-25

Japan
Monolithic Memories Japan KK
5-17-9 Shinjuku-Ku
Shinjuku
Tokyo 160
Japan
Phone 81-3-207-3131
Telex 232-3390 MMIKKJ
Fax 81-3-207-3130

United Kingdom
Monolithic Memories, Ltd.
Monolithic House
1 Queens Road
Farnborough, Hants
England GU146DJ
Phone 0252-517431
Telex 858051 MONO UKG
Fax 44-252-521041

Singapore
Monolithic Memories Singapore Pte., Ltd.
19 Kepple Road 11-06
Jit Poh Building
Singapore 0208
Phone 65-2257544
Telex RS55650 MMI RS
Fax 2246113

Germany
Monolithic Memories, GmbH
Mauerkircherstr 4
D 8000 Munich 80
West Germany
Phone 89-984961
Telex 5-24385 MONO D
Fax 983162

Domestic Sales Offices

Monolithic Memories, Inc.
4040 Moorpark Avenue, No. 216
San Jose, CA 95117
(408) 249-7766

Monolithic Memories, Inc.
1800 East McFadden, No. 110
Santa Ana, CA 92705
(714) 543-8664

Monolithic Memories, Inc.
One Energy Center, No. 250
300 East Shuman Blvd.
Naperville, IL 60566
(312) 961-9200

Monolithic Memories, Inc.
12801 N. Central Exp., Suite No. 530, L.B. 35
Dallas, TX 75243
(214) 690-3812

Monolithic Memories, Inc.
40 Speen Street
Framingham, MA 01701
(617) 875-7373

Other Technical Support Offices

Canoga Park, CA	—	(818) 341-7257
Norcross, GA	—	(404) 447-0006
Cherry Hill, NJ	—	(609) 751-1288
Cincinatti, OH	—	(513) 866-8928

Technical Application Hotline:
800-222-9323

Applications for any integrated circuits contained in this publication are for illustration purposes only and Monolithic Memories makes no representation or warranty that such applications will be suitable for the use specified without further testing or modification. Devices sold by Monolithic Memories are covered by the warranty and patent indemnification provisions appearing in its Terms of Sale only. Monolithic Memories makes no warranty, express, statutory, implied, or by description regarding the information set forth herein or regarding the freedom of the described devices from patent infringement in any described application or otherwise and no licenses of any kind are granted or implied under any patent or other right. Monolithic Memories makes no warranty of merchantability or fitness for any purposes. Monolithic Memories reserves the right to discontinue production and change specifications and prices at any time and without notice. Monolithic Memories' products are intended for use in normal commercial applications. Applications requiring extended temperature range, unusual environmental requirements, or high reliability applications, such as military, medical life-support or life-sustaining equipment are specifically **NOT** recommended without written Monolithic Memories' acceptance and additional processing for such applications.

Printed in U.S.A.