Programmable
Array Logic

Handbook

PROGRAMMABLE
PROGRAMMABLE
ROGRAM
OGRAMMABLE LO
GRAMMABLE LOG
GRAMMABLE LOGI
RAMMABLE LOGIC
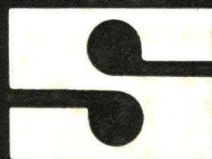
ADVANCED MICRO DEVICES

Advanced Micro Devices

Programmable Array Logic Handbook

Prepared by the Product Planning
and Applications Staff at Advanced
Micro Devices, Inc.

Brad Kitson, Editor

Contributors:
Warren Miller, Kevin Ow-Wing,
Jenny Yee, Phillip Sheu, Nick Zwick,
Mark Young, Jeff Kitson, Bill Sievers,
Mitch Richman

Preface

Programmable Array Logic (PAL) devices are fuse programmable logic building blocks capable of implementing complex, high performance functions which combine the architectural flexibility of a custom design with the instant availability, multiple sourcing and low cost of standard off-the-shelf products.

Early uses of PALs were predominantly as simple SSI and MSI replacement functions where standard TTL catalog items resulted in inefficient multiple package solutions. PALs provided a denser, faster, lower power and lower cost implementation. As designers learned to exploit the freedom of structuring their own components for a specific application, more innovative and efficient uses of fuse programmable logic began to emerge. Today a single PAL package is frequently employed to create functions that would require hundreds of conventional TTL gates.

Advanced Micro Devices is the world's largest merchant supplier of TTL compatible Bipolar LSI Logic and memory products. This has been achieved by implementing innovative, high performance LSI functions with advanced process technologies, such as IMOX^{TM} oxide isolation and ultra-reliable platinum-silicide fuse structures, and supporting them with dedicated high volume manufacturing facilities. These same capabilities have now been applied to fuse programmable logic devices. The result is a line of PAL components offering industry leading performance, programming yields, quality guarantees and functional flexibility.

The AmPAL 22V10, introduced in this book, represents a new generation of flexible architecture, fuse programmable logic products. Other, even more advanced, devices are in development based on the greater density and improved performance characteristics of new bipolar technologies. These will insure that programmable logic devices will continue to grow in importance as primary building blocks for advanced high performance systems.

This handbook is intended as an introduction to fuse programmable logic devices as well as a resource manual for experienced designers. If you require additional information on any of the products described in this book or our future plans in this area, please call your local Advanced Micro Devices Sales Office.

David A. Laws
Managing Director
Programmable Logic Products

"The VAX-11/730's circuit design is based on the use of
PALs, which have helped reduce board area for the
CPU by a factor of four and halve component costs, as
compared with equivalent performance MSI."

David A. Carlson and
Robert P. Morin
Digital Equipment Corporation
Electronics/October 6, 1982

"As time went on, however, it became clear that West
had made the right choice; PALs really were the chip
of the future."

Tracy Kidder
Referring to the Data General 32-bit
Eclipse MV8000 (Eagle) in
The Soul of a New Machine

Table of Contents

*See individual data sheets, Section 2.

Index to Product Specifications

Advanced PALs

Features of PALs

- User customizable, high performance logic building blocks
- Custom logic patterns may be generated in minutes with PROM type programmers
- Easy to use software design aids available (PALASM)
- Improves performance and reduces board area and cost of existing TTL SSI/MSI designs
- Aids creation of new system architectures through interactive design techniques
- Security fuse prevents copying of logic by competitors
- Slim 20 and 24-pin DIP packages

Advantages of AMD PALs

- IMOX oxide isolated technology insures industry's fastest (12ns typ) "A" versions and fastest half-power (24ns typ) "L" versions
- Platinum-silicide fuses and added test words insure programming yields > 98%
- Reliability assured through more than 40 billion fuse hours of life testing with no failures
- Full AC and DC parameter testing at the factory through on-board testing circuitry
- Preload feature permits full logical verification at the device level
- Power-up reset simplifies state machine design
- Industry leading quality guarantees

AMD PAL Speed/Power Families

Family	t_{pd} ns (Max)	$t_S^{(1)}$ ns (Max)	$t_{co}^{(1)}$ ns (Max)	$I_{cc}^{(2)}$ mA (Max)	I_{OL} mA (Min)
High Speed, "A"	25	20	15	155	24
Standard	35	30	25	155	24
Half Power, "L"	35	30	25	80	24

(1) Sequential functions.
(2) Combinatorial functions.

AMD PAL FUNCTIONS

Part Number	Array Inputs	Logic	OE	Outputs	Package Pins
16R8	Eight Dedicated Eight Feedback	Eight 8-Wide AND-OR	Dedicated	Registered Inverting	20
16R6	Eight Dedicated Six Feedback Two Bidirectional	Six 8-Wide AND-OR	Dedicated	Registered Inverting	20
		Two 7-Wide AND-OR-INVERT	Programmable	Bidirectional	
16R4	Eight Dedicated Four Feedback Four Bidirectional	Four 8-Wide AND-OR	Dedicated	Registered Inverting	20
		Four 7-Wide AND-OR-INVERT	Programmable	Bidirectional	
16L8	Ten Dedicated Six Bidirectional	Eight 7-Wide AND-OR-INVERT	Programmable	Six Bidirectional Two Dedicated	20
16H8	Ten Dedicated Six Bidirectional	Eight 7-Wide AND-OR	Programmable	Six Bidirectional Two Dedicated	20
16LD8	Ten Dedicated Six Bidirectional	Eight 8-Wide AND-OR-INVERT	–	Dedicated	20
16HD8	Ten Dedicated Six Bidirectional	Eight 8-Wide AND-OR	–	Dedicated	20
22V10	Twelve Dedicated Ten Bidirectional/ Feedback	Ten 12 (Average)-Wide AND-OR/ AND-OR-INVERT	Programmable	Ten Bidirectional/Registered Programmable Polarity	24

04140A-PLP

Introduction to Programmable Array Logic

Features of Programmable Array Logic
Advantages of AMD Programmable Array Logic
A Comparison Between Programmable Logic and Other Logic Alternatives
An Introduction to Programmable Logic Architecture
PALs Aid High Performance 32-Bit CPU Design

Features
of Programmable
Array Logic

Flexibility

Design Optimization
Programmable logic removes constraints placed on the
designer by the available selection of fixed-function TTL
SSI/MSI parts. If a desired function does not exist, the
designer may need to use a large number of packages
to generate it. With PALs, the designer can create a
customized part for a specific application.

Faster Design Cycle
Programmable logic offers a way to reduce design
cycle time. In a typical top down design, rather than
determining the specific logic required for a function
immediately, the designer can simply specify that a
programmable logic device will be used. This allows the
architecture and interface between logic blocks to be
defined before the specific details of each logic block are
specified. The individual logic blocks can then be
designed with a minimum number of iterations.

Simple Prototyping and Debugging
Programmable logic greatly reduces the costs and time
consuming effort associated with system design changes.
Any changes because of logic errors or revisions in
product specification may be easily implemented by
reprogramming the device instead of rewiring or relaying
out a board or making a new mask for a gate array.

High Performance

Optimized Design
System performance can be increased through the use
of programmable logic. The designer has the freedom
to optimize an architecture by tailoring programmable
devices to implement it precisely. Thus a design may be
implemented in the most efficient manner, frequently
increasing performance.

Reduced Delay
When a logic function is implemented in multiple SSI/MSI
packages, the total delay incurred includes the time
required for several on and off chip buffers. When the
same function is implemented in a single programmable
logic element, the delay per logic gate is reduced.

Low Cost

Reduces Board Space

PAL devices available today can provide logic complexity equivalent to 300 TTL gates. Implementing a design in programmable logic can therefore significantly reduce the board space or the number of boards necessary to implement a given function. This results in lower system cost, or alternatively, the ability to provide more function in the same enclosure.
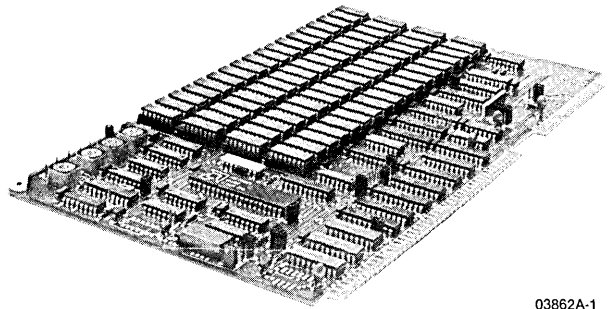
03862A-1

PAL Control Logic Fits One Megabyte of RAM onto a Single MULTIBUS Board (Am971024B)

Reduces Inventory Cost

Programmable logic can be used to replace more than 90% of standard TTL parts. This allows the user to reduce his inventory from hundreds of different TTL devices to just a few programmable logic device types. This simplifies inventory requirements as well as easing purchasing procedures.

Reliability

Reduced Parts Count

Compared to standard TTL SSI/MSI, programmable logic reduces the number of packages necessary to implement a given function. In some cases, an entire PC board can be eliminated. This results in increased reliability.

Reduced Interconnections

The least reliable portions of a digital system are the connections between integrated circuit devices. Reducing the number of packages reduces the number of external connections and therefore improves the reliability.

Support

PALASM

Programmable logic designs may be executed through an easy to use software design tool called PALASM. The user inputs the desired logic equations and PALASM automatically generates the fuse programming information. The input file, called a PAL DESIGN SPECIFICATION, provides excellent documentation on each design. The output may be downloaded to a wide variety of low cost logic programmers. Logic simulation capabilities are provided in PALASM to help the designer verify the logic design. The output of the simulator can also be used to test a programmed device.

Design Security

By programming a special "security fuse", the user can disable the fuse verify logic circuitry. This prevents unauthorized duplication of the device, while not interfering with the part's logic functionality. This makes programmable logic ideal for any application where design security is essential.

Advantages of AMD Programmable Array Logic

Improved Performance

IMOX

The AMD PAL family is manufactured using Advanced Micro Devices' IMOX advanced oxide isolation process. IMOX, which has been in production for many years on high performance bipolar LSI devices such as the Am2900 family, insures the best speed/power performance PALs available in the industry.

Higher Speed—"A" Versions

The use of IMOX technology insures high yields and therefore predictable availability of the high speed "A" versions of PALs. Worst case input to output delay of 25ns is specified with typical performance in the 10ns to 12ns region. New generations of IMOX will provide even faster guarantees.

Half Power—"L" Version

A new family of half-power PALs, designated "L" versions, provides standard 35ns maximum delays and full 24mA drive capability at half the standard power. Half power PALs will directly replace standard and early (− 2) low power functions while enhancing system reliability and performance.

In addition, AMD standard and high speed "A" combinatorial PALs, are specified at more than 20% lower power dissipation than other manufacturer's devices.

20-PIN, COMBINATORIAL PAL MAXIMUM SPECS COMMERCIAL DEVICES
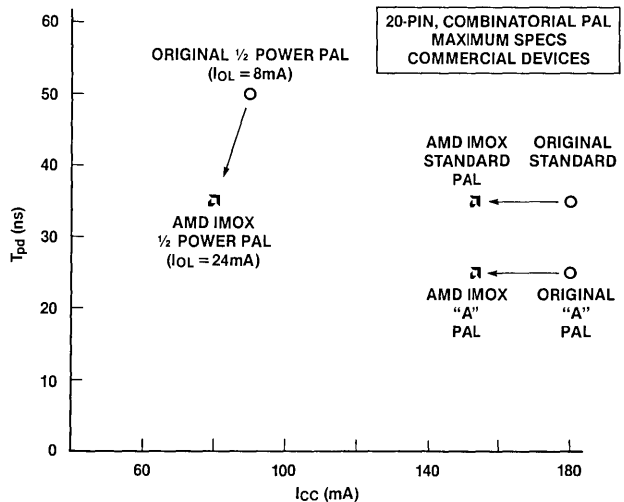
03862A-2

Flexibility

Enhanced Line (20-pin)
In addition to the popular high volume 20-pin PAL devices, Advanced Micro Devices offers three additional functions.

AmPAL16H8 is an active HIGH version of the AmPAL16L8. These parts together provide the capability of implementing logic with either active HIGH or active LOW outputs. Switching logic equations from one polarity to another can achieve a significant reduction in product term usage. The AmPAL16H8 can functionally replace all other active HIGH 20-pin PALs.

AmPAL16LD8 is an active LOW device implemented with dedicated outputs to increase the number of logical product terms to 8 per output.

AmPAL16HD8 is an active HIGH version of the AmPAL16LD8. These two functions give the capability of creating a wider range of functions in a single PAL.

Enhanced Line (24-pin)
The AmPAL22V10 is a 24-pin device which will allow the user to program the architecture. Each of the 10 outputs may be registered or combinatorial, active HIGH or active LOW. Variable product term distribution will permit between 8 and 16 logical product terms per output for a total of 120. This device provides a new standard of flexibility in PAL functions.

Power-up Reset
The registered devices in the AMD PAL family are designed to reset during system power-up. All registers will be set to zero, setting all the outputs to ones. This feature is especially valuable in simplifying state machine initialization.

Full Test Capability

Special Test Circuitry
All AMD PALs include special test circuitry to allow thorough AC and DC testing of unprogrammed units prior to shipment. The test circuitry is used to insure good programming yield and to verify that devices will meet all parametric and switching specifications after programming.

Programmability Testing
Programming circuitry testing includes tests to assure unique addressing of all fuses. The ability of circuitry in the programming path, to handle the large currents and voltages necessary to blow fuses, is also checked.

Special probing pads, high threshold voltage circuitry and test fuses are employed in programmability testing. This testing coupled with the platinum silicide fuse structure gives industry leading programming yields ($>$98%) for all AMD PALs.

DC Functional Testing

Special test circuitry, enabled by means of high voltage signals, checks functionality and DC parameters under conditions that simulate post programming operation. All circuitry and levels that can be involved in operation after programming are checked under worst case conditions.

For example, all input buffers are tested for functionality by switching them through a special path to a single output. All product term AND gates are switched and sensed for uniqueness and functionality.

AC Testing

Similar special test circuitry permits AC switching delays through worst case paths to be measured. This provides a means to guarantee AC specifications under worst case power supply and loading conditions.

PRELOAD for Logic Verification

AMD PALs provide the capability of loading the output registers of a PAL to any desired value during testing. PRELOAD is the only way to allow full logical verification of programmed registered PALs and thus guarantee correct logical functionality. Without PRELOAD, many device failures cannot be discovered until the device is tested as a part of the finished system.

Reliability

High Programming Yield

The proven platinum silicide fuse structure used for many years in AMD PROMs is also applied to PALs. This insures that AMD PALs consistently achieve better than 98% programming yields.

High Reliability

This same fuse technology has demonstrated an excellent reliability history. Zero fuse failures have been generated in over 40 billion fuse hours of life testing.

Design Aid Software

PALASM

The AMD PAL family is supported by an upgraded version of the PAL Logic Equation Assembler, PALASM. Known as AMPALASM20, this design aid software provides error checking and recovery features and the JEDEC Programmable Logic Data Transfer Format output capability. Advanced Micro Devices provides AMPALASM20 on an 8 inch CP/M floppy disk for the AMD System 8 and 29, and other popular computer systems. Advanced Micro Devices is committed to providing the continuing support necessary for programmable logic as new, faster and more complex devices become available.

A Comparison Between Programmable Logic and Other Logic Alternatives

Today's logic designer can choose from a wide variety of implementation alternatives. These can be classified into three basic categories: dedicated general purpose devices (hereafter called standard products), fuse programmable logic and custom integrated circuits (Figure 1).

Standard product architectures are defined by the IC manufacturer for a wide market and cannot be altered by the user. Examples of standard products are fixed instruction set MOS microprocessors, microprogrammable LSI building blocks, and TTL and CMOS SSI/MSI devices. Custom logic on the other hand, is defined by the user for his application. Programmable logic devices fit between standard products and custom logic. The IC manufacturer defines an architecture that a user can program in his facility by blowing appropriate fuses to fit his specific application. PALs, PLAs and PROMs are examples of programmable logic.

```
                  ┌─────────────────┐
                  │     DIGITAL     │
                  │  LOGIC FAMILIES │
                  └─────────────────┘
           ┌──────────────┼──────────────────────┐
   ┌───────────────┐ ┌──────────────┐      ┌──────────────┐
   │   STANDARD    │ │ PROGRAMMABLE │      │    CUSTOM    │
   │   PRODUCTS    │ │    LOGIC     │      │    LOGIC     │
   └───────────────┘ └──────────────┘      └──────────────┘
  54/74 TTL SSI/MSI        PAL          ┌─────────┴─────────┐
     2900 TTL LSI          FPLA    ┌──────────┐      ┌──────────┐
     iAPX 86 µP            PROM    │  SEMI-   │      │   FULL   │
                                   │  CUSTOM  │      │  CUSTOM  │
                                   └──────────┘      └──────────┘
                                      GATE         STANDARD CELLS
                                     ARRAYS      HANDCRAFTED DESIGNS
                                                        03862A-3
```
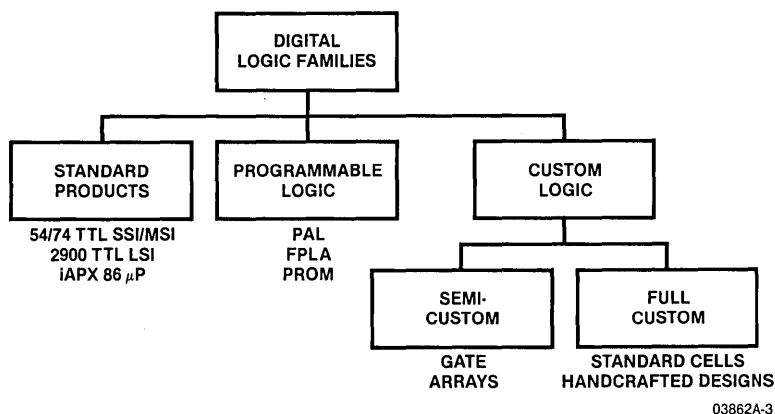
Figure 1. Basic Categories of Digital Logic

Each of these three design alternatives offers distinct advantages and disadvantages in terms of cost, availability and architectural flexibility. Many system designs today, such as the controller board in Figure 2, incorporate all three of the design approaches to some degree. However, in order to evaluate which type is best suited for a particular function, this review will consider each approach on a stand-alone basis.

DEDICATED GENERAL PURPOSE DEVICES— STANDARD PRODUCTS

There are five main advantages of standard products. They require little IC engineering expertise by the user, provide lowest cost for an individual device, usually have the best application support, provide the maximum logic density per device and are available off-the-shelf with no development lead time.

Development engineering effort at the IC level is minimal compared to the custom or programmable alternatives. The responsibility for design, test, and debugging is borne by the integrated circuit manufacturer. Because the integrated circuit manufacturer is doing this on a large scale, the process is very efficient. The engineering time and investment saved by the standard product user can be utilized to do design work that is more directly profitable and in his realm of expertise.

Standard products achieve a cost reduction on an individual device basis because they are high volume products. This volume results in lower manufacturing cost and thus lower price per unit. The increased competition encouraged by alternate sourcing products also results in lower cost.

The design support available for standard products is far greater than that for custom or programmable devices. Application software (assemblers, simulators), hardware (emulators) and literature (manuals, books, application notes) make them easier to design with. Since standard products reach a much larger market, the engineering effort necessary to provide this support can be spread over a large number of units, reducing the cost. When a custom or programmable logic device is used, this support must be developed by the engineer doing the design.
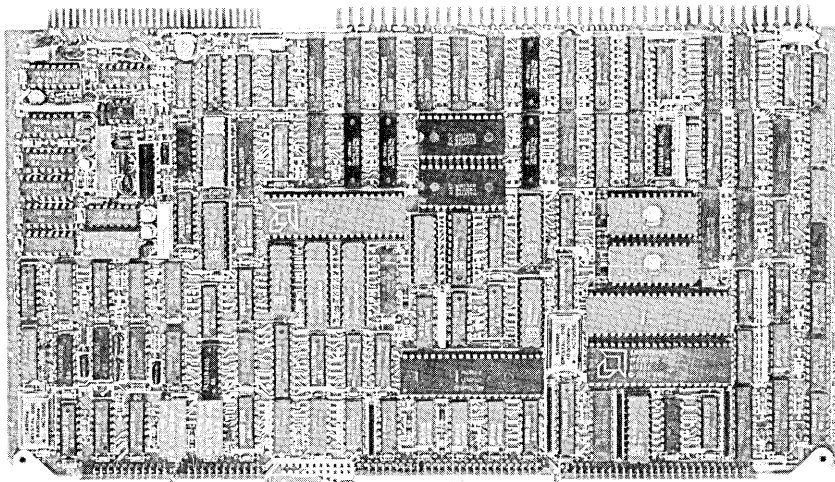
03862A-4

Figure 2. Using PALs for state machines, a gate array for error correction, and standard products including a DMA controller chip, and an 8085 microprocessor, Data Systems Design squeezes hard-disk, floppy-disk, and mag tape controllers onto one board. (7215 Controller Board) Photo Courtesy of Data Systems Design, Inc.

Standard products are optimized for high volume production. The density of logic functions is therefore generally much greater than on custom logic (when implemented with gate arrays) or programmable logic devices. A fixed instruction set microprocessor or microprogrammable building block duplicated with gate arrays or programmable logic devices would take several packages compared to the single dedicated device.

The three potential disadvantages of standard products are non-optimality, higher system cost and lack of unique feature advantages. A standard product, by the very nature of its generality, is not ideal for anyone. It includes too much functionality for some applications and not enough for others. The architecture is seldom ideal for a particular application. Standard products also offer a limited performance selection. IC manufacturers pick a specific performance level aiming at as large a market as possible.

Due to the general purpose nature of standard products, it is difficult to achieve the lowest package count solution. Additional components are required to tailor the function to fit a specific need. Even though individual devices may be lower in price, more of them must be used, raising the cost for the total system when considering the additional PC boards, testing, power supplies, fans, etc.

Another disadvantage of standard products is the lack of competitive features and advantages. Anyone can buy them so it is difficult to differentiate one system supplier's hardware from another.

CUSTOM LOGIC DEVICES—GATE ARRAYS

Custom logic, predominantly in the form of gate arrays today, offers the system designer important advantages over standard products. Compared to SSI/MSI implementations, reduced package count is of paramount importance. Standard LSI products provide the same benefit but force the designer to use a specific architecture. Custom logic allows the designer to implement his own architecture exactly. This freedom to develop innovative solutions to an applications problem can add a significant competitive advantage to a product.

The four main disadvantages of gate arrays are increased engineering effort, higher cost per individual device, lack of high level support tools and lower density compared to standard LSI products. Engineering effort for a gate array can significantly increase the cost of a system design. Not only must the system be designed, but the custom devices themselves must be designed, debugged and put into production. Both design tasks, chips as well as system, take similar amounts of engineering resources, possibly doubling the design effort and investment. Because of the lack of a competitive market (minimum second sourcing), custom logic devices can end up being substantially more expensive. Only

if the complete system solution can be optimized will the total cost be reduced. Another factor to be considered is the chance of design problems with a custom device. If extra iterations are necessary, or even worse a bug is discovered after a product has been released, correcting the problem can take several months or even years. These potential costs are difficult to estimate and have virtually no limit.

The third disadvantage of custom logic is the lack of high level support. Semiconductor manufacturers cannot provide significant support in the form of software, development systems, application notes, or books for a custom logic design because each device is different. The designer must document the design fully and provide enough support for the system engineer to utilize the device correctly.

Finally, a key disadvantage of gate arrays is the reduced density and therefore higher silicon cost compared to a dedicated general purpose device. They are designed by repeating a common loosely packed structure, leaving wide channels for the metal interconnect. For a given set of design rules a gate array will typically require two to five times the silicon area for the same gate count.

PROGRAMMABLE LOGIC DEVICES

Programmable logic combines the advantages of the flexible architecture of a custom design with the off-the-shelf availability and reduced investment—engineering time and device cost—of a standard product.

Programmable logic has the fastest design cycle time of any form of custom logic. Instead of months, or years, as with semicustom or full custom designs, a programmable logic element can be defined by programming the fuses on a blank device. This process takes only seconds. This fast turnaround time allows a revolutionary interactive approach to system design. The engineer can try out a new architectural approach and evaluate it very quickly. If it does not work, a new idea can be defined, programmed and ready to evaluate in hours. The speed with which a new design approach can be explored and evaluated creates a design environment that enhances innovation.

Programmable logic devices share the same economics of high volume production as standard products and other user customizable integrated circuits such as PROMs, EPROMs and EEPROMs. As the manufacturer produces identical blank elements by the millions of units per year, low costs can be achieved. This volume market attracts multiple vendors and encourages price competition, as well as provides alternate parallel construction source security. The cost advantages of a standard product are retained with programmable logic devices, but as parts are customized, system designs may be differentiated from the competition. In fact, truly innovative designs are even patentable, further protecting a design from the competition.

The engineering effort and time needed to design, test, debug and put into production a programmable logic device is larger than the effort necessary for a standard product, but substantially less than for a custom element (Figure 3). Software tools are provided to reduce this overhead considerably. These permit designs to be specified in terms of Boolean equations. The input specification format, for the software, serves as a "data sheet" for the particular application and generates the essential documentation information. Simulation and test vector generation programs also exist to reduce the engineering effort associated with debugging and testing, both in prototyping and production environments.

Programmable logic devices available today and in the near future provide the functional equivalent of up to 250 to 500 gates. While this is an order of magnitude better than typical

SSI/MSI designs, it is less than that of advanced gate array products. However when all costs are amortized, programmable logic can still provide the lowest price form of custom logic available to most system manufacturers.

This threshold is increasing rapidly as advanced process technologies improve the effective logic complexity of programmable devices.

SUMMARY

Programmable logic combines the strengths of the dedicated general purpose and custom logic design approaches. It provides interactive design via customizability and immediate turn-around time. This revolutionary design approach results in innovative, low cost designs, maximizing the competitive advantage of a product.
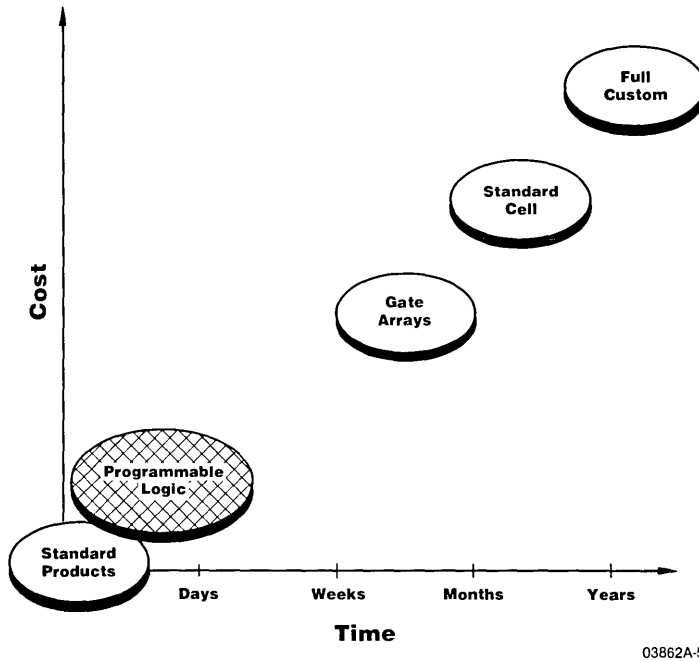
03862A-5

Figure 3. Relative Development Time vs Cost for Alternative Logic Implementations

An Introduction to Programmable Logic Architecture

Programmable array logic (PAL) devices have many features in common with programmable read-only memory (PROM) and programmable logic array (PLA) devices. All three share the same basic internal AND-OR structure, but vary in allocation of logic features and amount of programmability. Figure 1 shows the basic AND-OR structure of programmable devices. It consists of two levels; the first is the AND array which accepts inputs, performs the desired AND functions on the inputs and then outputs these functions to the second level, the OR array. The OR array combines various AND functions together producing the desired (AND-OR) outputs. This structure makes programmable devices ideal for implementing logic in Boolean sum-of-products form which is easily generated using logic design techniques such as Karnaugh maps.

INPUTS

AND
ARRAY

OR
ARRAY

OUTPUTS   03862A-6
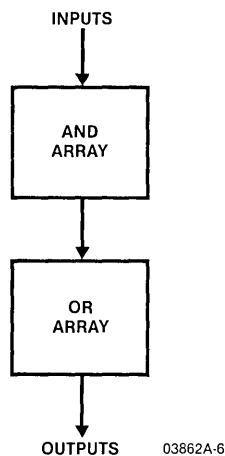
Figure 1. Basic Programmable Logic Array Architecture

Figures 2 and 3 depict the rules for understanding the notation commonly used in logic diagrams to describe programmable logic devices. Figure 2 shows the technique for describing an AND array. All array inputs (true and complement of each device input) are shown connecting to a single input AND-gate. In reality, each array input is an input to the AND-gate. Thus an N-input device will have AND-gates with 2N inputs. For example, the AmPAL16L8 has sixteen inputs and therefore each of its sixty-four AND-gates has thirty-two in-puts! In a programmable AND array each row and column in-tersection, as shown in Figure 2, represents a fusible input connection to the AND-gate. Thus, to create an AND func-tion, the fuses associated with undesired inputs must be blown. Figure 3 shows the technique for describing an OR ar-ray. All of the rules for the OR array are the same as for the AND, except that an OR function is being implemented in-stead of an AND.
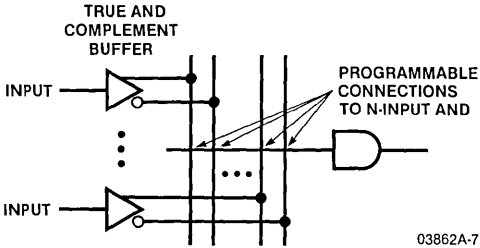
Figure 2a. Programmable AND Array
Logic Diagram Notation
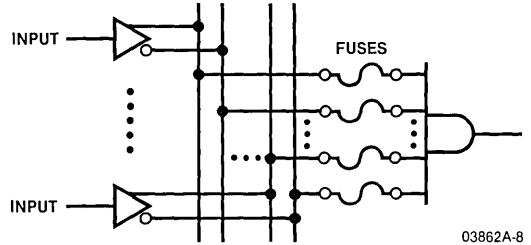
Figure 2b. Programmable AND Array
Logic Equivalent
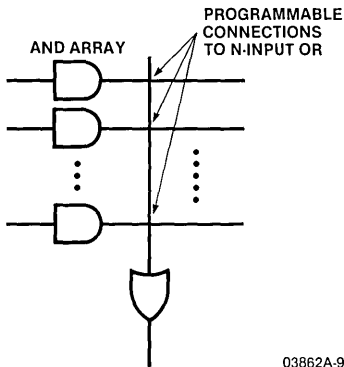
Figure 3a. Programmable OR Array
Logic Diagram Notation
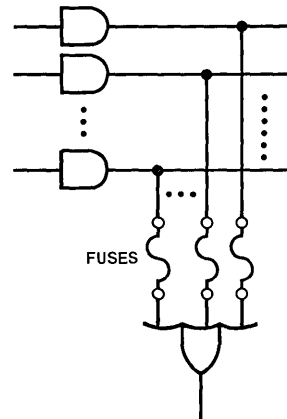
Figure 3b. Programmable OR Array
Logic Equivalent

PROM ARCHITECTURE DESCRIPTION

Figure 4 shows the basic architecture of a PROM using the notation of Figures 2 and 3. The PROM shown has three inputs, eight memory locations (AND-gates), and four outputs. The important feature of the PROM architecture, as far as programmable logic is concerned, is that the inputs are fully decoded by a fixed AND array which drives a programmable OR array. This means that every combination of inputs is represented by a separate AND-gate. Since there are 2^n combinations possible from n inputs, there are 2^n AND-gates in a PROM. For example, the PROM of Figure 4 has three inputs and because 2^3 is eight, there are eight AND-gates in Figure 4.

By programming the OR array for a given output, as desired, the PROM can implement any logic function limited only by the number of inputs available. A separate, independent logic function can be implemented for each device output.

The limitation of PROMs in performing logic functions is their inability to provide the number of inputs and outputs that logic functions need. PROMs have a fixed number of inputs and a fixed number of outputs. For example, a $1K \times 8$ PROM has ten inputs, to fully decode 1K locations (remember that's 1024 fixed AND-gates!), and eight outputs (some PROMs have only four outputs). Unfortunately, logic functions don't come with fixed numbers of inputs and outputs. This means that a logic function requiring a total number of inputs and outputs that is less than a device offers may not fit because it requires an allocation of inputs and outputs that doesn't fit the fixed PROM architecture. A function requiring eleven inputs and five outputs would not fit into the previously mentioned $1K \times 8$ PROM, despite requiring fewer total inputs and outputs than the device offers.

Typical logic functions can easily have up to sixteen inputs which would require a PROM with 64K locations. Few designs could utilize 64K AND-gates. Typical output functions don't always come in four or eight bits. Data path functions tend to be wider than the path itself because functions such as parity bits, ripple carrys, and serial inputs and outputs are usually required in addition to the data inputs and outputs. Thus four or eight bit data path functions would not be well served by PROMs.

Control path functions, such as state machines, can quickly use up both inputs and outputs. Using a PROM with a register on the outputs as a state machine requires both logical inputs and state feedback inputs, while also requiring state feedback and control outputs (see Figure 5). Note that the feedback inputs and outputs are tied together using up an input and output pin for each bit of state information. Thus, when a large number of states are required, few precious input and output pins are left over.

INDICATES PROGRAMMABLE CONNECTION

INDICATES FIXED CONNECTION

$O_3 \quad O_2 \quad O_1 \quad O_0$

03862A-11
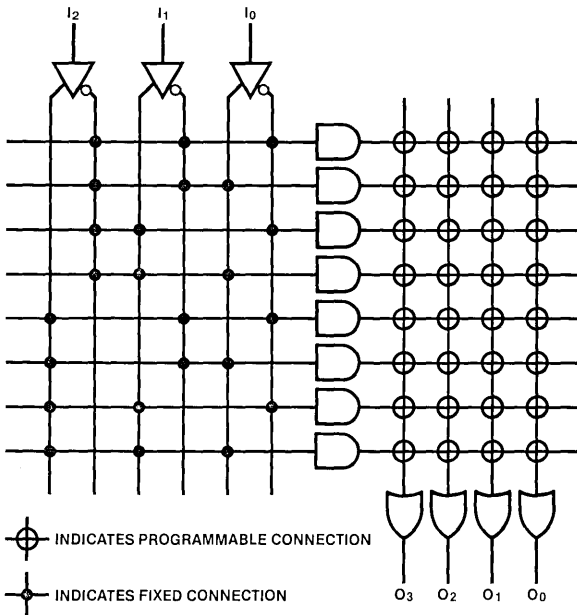
Figure 4. PROM Array Structure

03862A-12
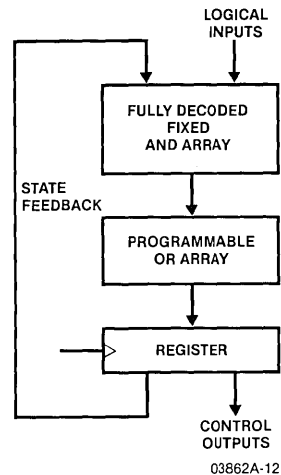
Figure 5. Registered PROM State Machine

PAL ARCHITECTURE DESCRIPTION

The array architecture of a PAL is shown in Figure 6. The basic PAL structure is exactly the opposite of a PROM; the AND array is programmable and the OR array is fixed. This immediately removes the restriction that for n inputs there are 2^n AND-gates. There are six inputs to the PAL array of Figure 6, but only eight AND-gates. Thus one of the key inefficiencies of a PROM is removed, allowing PALs to have as many inputs as needed. The fixed OR array of a PAL dedicates which OR-gate a particular AND-gate will input to. In Figure 6, two AND-gates are dedicated to each OR-gate in the array. This is the only limitation of PAL devices: the number of AND-gates required by an equation may not exceed the number provided.

PAL devices contain many additional architectural features which make them ideal for implementing logic functions. These features include programmable I/O pins, outputs with registers that internally feedback to the AND array, and active HIGH or active LOW outputs. Programmable I/O pins allow the PAL device to be tailored to fit the required allocation of inputs and outputs. Thus PALs effectively remove the limitation of inputs and outputs. This allows PALs to implement far more different and complex logic functions than a PROM (even one with more pins). Registered outputs with internal feedback give PALs the capability to implement state machines efficiently. Device inputs need not be sacrificed as feedback inputs as in the PROM. PALs also provide active HIGH or active LOW capability.

INDICATES PROGRAMMABLE CONNECTION

INDICATES FIXED CONNECTION
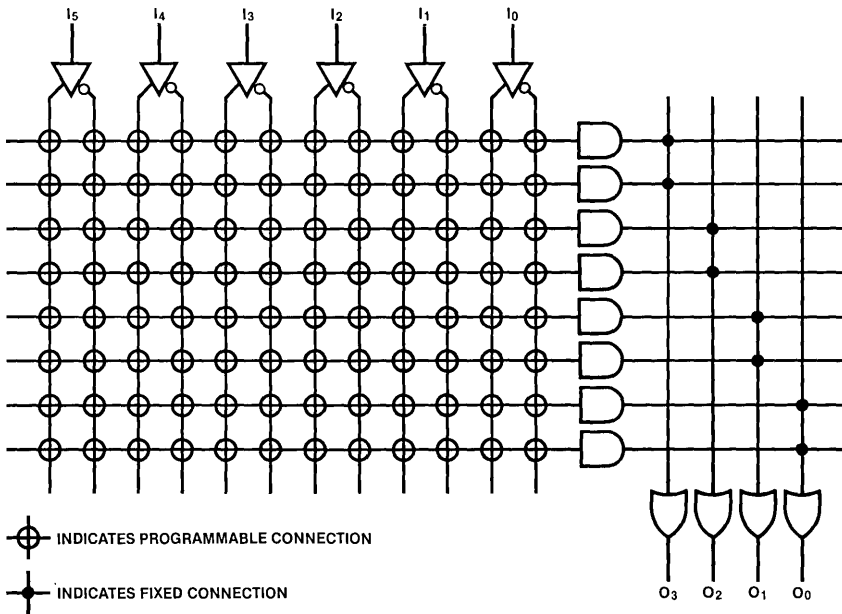
03862A-13

Figure 6. PAL Array Architecture

Logic diagrams for the bidirectional output structures of the PAL devices are shown in Figure 7. One feature of the PAL bidirectional output is the ability to program the output enable as a function of an AND-gate in the array. The output buffer may be programmed in one of three ways: as a dedicated output, a dedicated input, or a dynamically controllable input/output.

When programmed as a dedicated output, the output buffer is always enabled and the logic function is fed-back to the AND array. The feedback path allows more complex logic functions to be implemented by using two or more levels of AND-OR gating.

When programmed as a dedicated input, the AND-OR gate associated with that pin is unused, but an extra input has been created. This ability to trade-off outputs for inputs is one of the big advantages of PALs over other programmable logic devices. The designer isn't limited to a fixed number of input and output pins. The ratio may be programmed to fit the intended application.

Finally, when programmed as a dynamically controllable input/output buffer (i.e., enabled/disabled by a logical combination of one or more inputs) this pin may be utilized as an input, as well as retaining the full logical capability of the AND-OR gate. This is especially useful in control applications (microprocessor handshaking protocols) and bus oriented data operations (data steering and data storage/manipulation). A serial input/output pin is a common example. When left shifting the pin may be a serial input, but when right shifting the pin would be a serial output. This mode provides maximum utilization of the PAL architectural resources.

A logic diagram for one of the registered outputs of a PAL device is shown in Figure 8. The most important features of this structure are the feedback path and the dedicated output enable. This output enable is common to all registers on the chip. The output register is fed-back into the array internally instead of from the output pin as in the combinatorial part. This configuration is more useful because state information is available at all times instead of only when the output is enabled, simplifying state machine design.

The availability of a common, dedicated output enable makes registered PAL devices ideal for bus oriented systems. The registered PAL can be programmed to provide data storage, operation, or steering functions, the result of which is placed on a data bus by enabling the output buffer. Since all PAL outputs have 24mA current sinking capability, they can drive most on-board buses and many backplane buses.
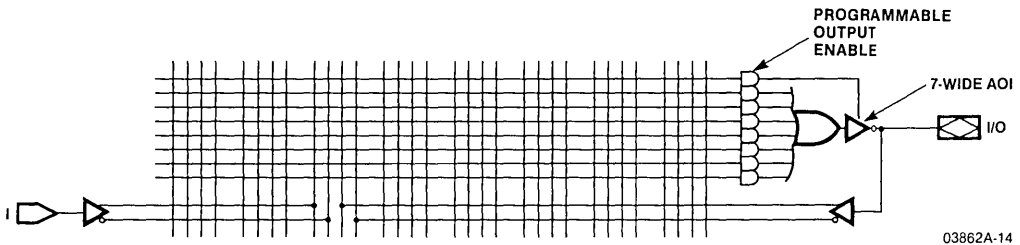
Figure 7a. Active LOW Bidirectional Output
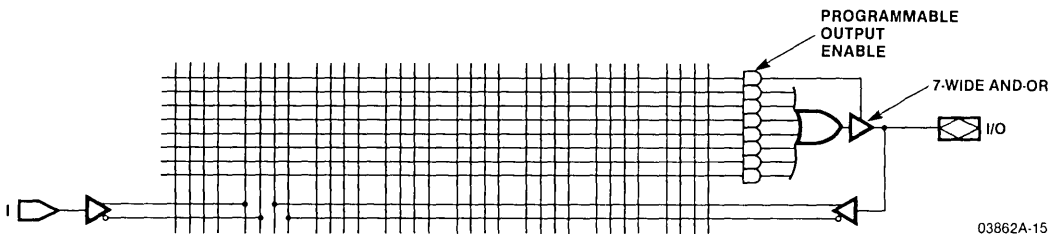
Figure 7b. Active HIGH Bidirectional Output
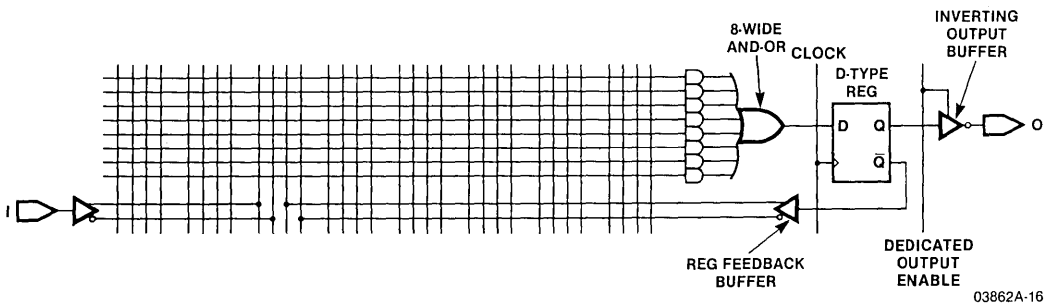
Figure 8. Registered Output

Figure 9 shows the active LOW and active HIGH versions of PAL dedicated outputs. The outputs are always enabled. The AND-gate previously used for this function provides an extra logical AND term in this structure. This brings the total number of AND-gates per output to eight. The feedback path from output to input is still provided, allowing for implementation of multi-level logic. The extra AND-gate makes these outputs ideal for non-bus oriented logic replacement, especially complex control signal generation, encoding and decoding.

AMD 20-PIN PAL ARCHITECTURE

The AMD 20-pin PAL family is based on an array of over 2000 platinum-silicide fuses. These provide the logical equivalent of sixty-four, 32-input programmable AND-gates. The array outputs feed eight, 8-input fixed OR-gates plus associated I/O and feedback circuitry. Each device type has a unique organization of these I/O components, optimized for specific functional applications.

As any logical function can be expressed in an AND-OR, sum-of-products form, these basic elements can be programmed to satisfy a wide variety of complex custom logic requirements. Where a system architecture has been created around this PAL structure, single 20-pin packages have been used to perform functions that would each require over 300 equivalent TTL gates.

A typical member of the AMD 20-pin PAL family, the AmPAL16R4, is shown in Figure 10. This device has 16 available inputs to the fuse programmable array. Eight of these are dedicated inputs (pin numbers 1 through 9), four are feedback paths from the $\overline{Q}$ outputs of the on-board registers and four are via the bidirectional input/output ports (pin numbers 12, 13, 18 and 19). It contains four 8-wide AND-OR structures with inverting registered outputs, each AND-gate having 32 inputs. As half of the inputs are true and the other half complementary, only sixteen of them have effective logical value. A common three-state output enable line serves all four registered outputs. Four more 7-wide AND-OR-INVERT structures have combinatorial outputs with three-state output enables that are programmable through the fuse array.
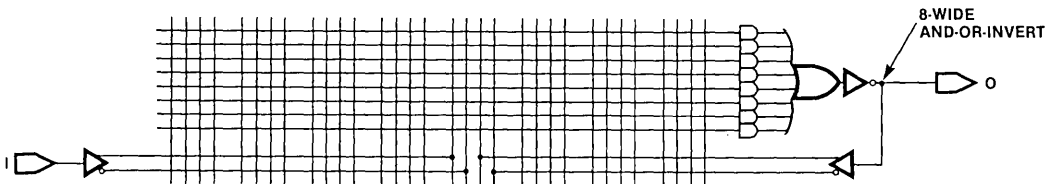
03862A-17

Figure 9a. Active LOW Dedicated Output
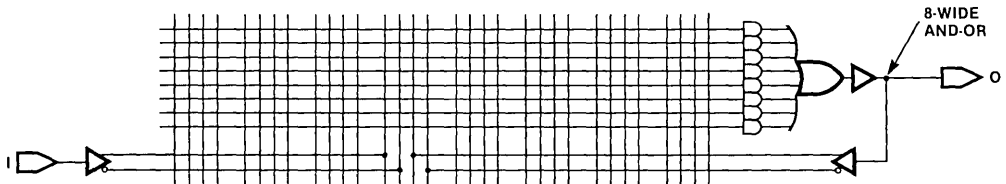
03862A-18

Figure 9b. Active HIGH Dedicated Output
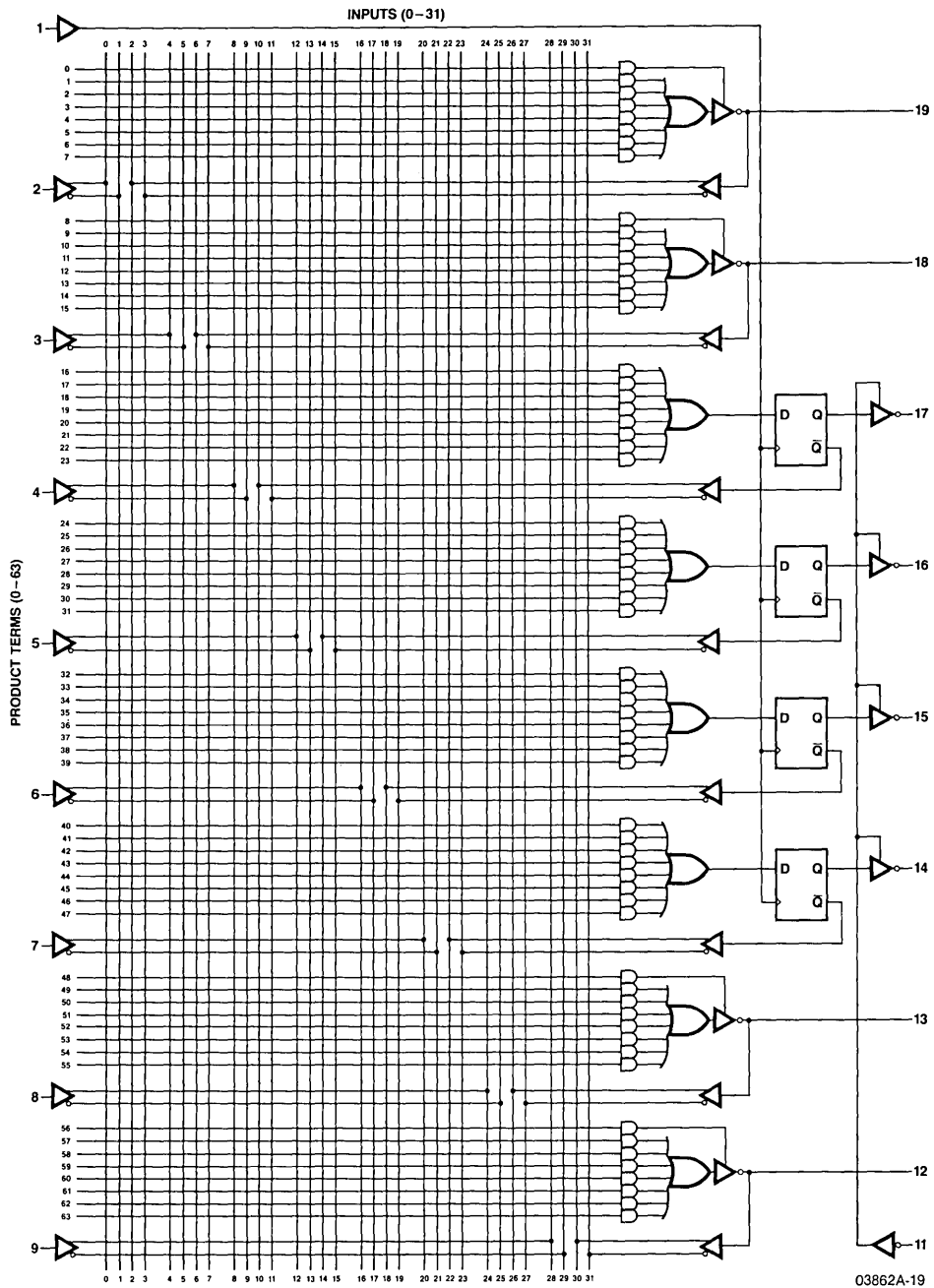
Figure 10. Logic Diagram of AmPAL16R4

- 16 Array Inputs
  - 8 Dedicated
  - 4 Registered Feedback
  - 4 Bidirectional I/O
- 4 8-Wide AND-OR Structures
  - Registered, Inverting Outputs with Common, Dedicated Output Enable
- 4 7-Wide AND-OR-INVERT Structures
  - Combinatorial Outputs with Programmable Output Enables

03862A-19

AMD 24-PIN PAL PROGRAMMABLE LOGIC STRUCTURE

A new 24-pin device, the AmPAL22V10, allows programming of the logical function of each output separately to allow the user to select the preferred output structure. The basic output structure, or "macrocell", is shown in Figure 11 along with diagrams of the different output configurations. The different configurations are bidirectional/active LOW, bidirectional/active HIGH, registered/active LOW, and registered/active HIGH. Thus the AmPAL22V10 can be architecturally optimized, as well as input and output optimized (as in the 20-pin family), to fit the particular logic function precisely.

S_1	S_0	OUTPUT CONFIGURATION
0	0	REGISTERED/ACTIVE LOW
0	1	REGISTERED/ACTIVE HIGH
1	0	COMBINATORIAL/ACTIVE LOW
1	1	COMBINATORIAL/ACTIVE HIGH

0 = UNBLOWN FUSE   1 = BLOWN FUSE

03862A-20
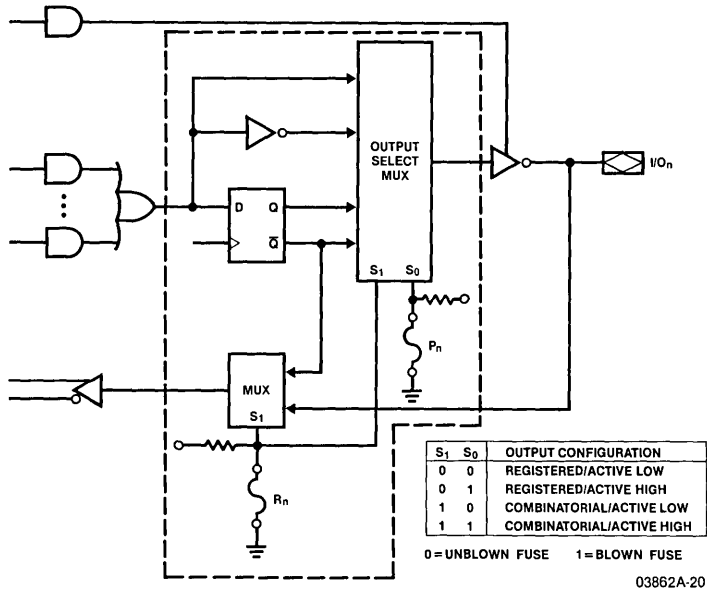
AmPAL22V10 Output Logic Macrocell

$S_0 = 0$
$S_1 = 0$

03862A-21
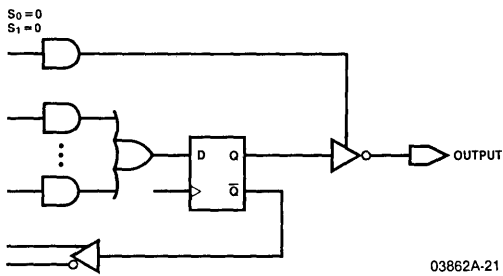
Registered/Active LOW Configuration

$S_0 = 0$
$S_1 = 1$

03862A-22
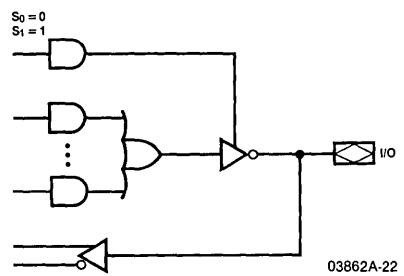
Combinatorial/Active LOW Configuration

$S_0 = 1$
$S_1 = 0$

03862A-23
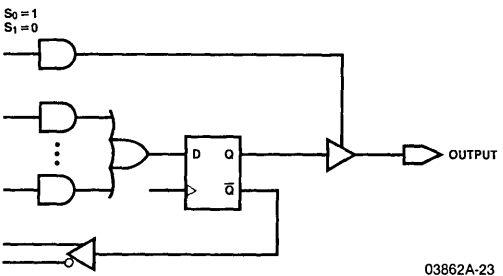
Registered/Active HIGH Configuration

$S_0 = 1$
$S_1 = 1$

03862A-24
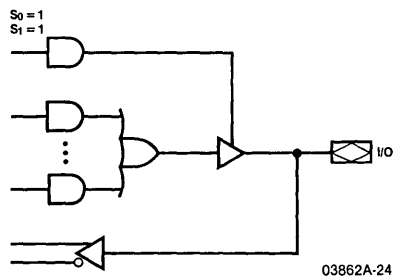
Combinatorial/Active HIGH Configuration

Figure 11

PLA ARCHITECTURE DESCRIPTION

The array architecture of a PLA is shown in Figure 12. The PLA allows both the AND array and the OR array to be programmed. This gives the PLA additional logic capability over both PROMs and PALs. PLAs can be designed to incorporate the same advantages over PROMs as do PALs. PLA devices can include the same logic features which reduce limitations of too few inputs, the allocation of inputs vs outputs, registered feedback, or output polarity, although few commercially available devices actually implement them. The programmable OR array allows AND-gates to be tied to OR-gates, as desired, by programming. Logic functions are limited by the total number of AND-gates allocated to all outputs instead of by the AND-gates allocated to a particular OR-gate (as in a PAL). Thus if a logic function requires a large number of AND-gates, they may be allocated to the particular OR-gate requiring them. Additionally, AND-gates may be connected (shared) to more than one OR-gate. This allows more efficient utilization of AND-gates in a PLA than in a PAL.

The disadvantages of PLAs are not quite so obvious. PLAs are inherently slower than PALs or PROMs because a given signal must pass through two programmable arrays. This can make a PLA unsuitable for many high performance applications. In practice the user can seldom take advantage of allocating a large number of AND-gates to a particular OR-gate. The number of AND-gates required for a particular equation is related to the number of inputs to the equation. PLA devices have a limited number of inputs, thus the number of AND-gates required by an equation is limited. Creation of equations using a large number of AND-gates can become very difficult. Logic design techniques such as Karnaugh maps cannot handle much more than five or six inputs and computer aid for this task is not generally available. Another problem is that commercially available PLAs have fewer AND-gates than comparable PALs because of the added silicon real estate required to provide the programmable OR array. If a designer creates an equation using most of his available AND-gates, only a few may remain for the other OR-gates.

To take advantage of potential AND-gate efficiency with respect to sharing is not easy. For example, in data path applications such as a barrel shifter, individual equations are dependent upon the data line of which they are in the path (i.e., the equation for output O_0 is dependent on D_0 and Q_1 is dependent on D_1, etc.). This makes sharing of AND-gates impossible. In other words, data path equations are ideally suited to the architecture of PALs. Since the critical path of most systems is the data path, and PALs are faster than PLAs, they are better suited for these applications.

CONCLUSION

The three programmable logic architectures are represented by the PROM, PAL, and PLA devices. Although very similar in basic array architecture, they differ significantly in their ability to implement logic functions and in their applications. Each device type implements an AND-OR two-level logic array which allows implementation of logic equations in sum-of-products form. The PROM is the most limited of the three device types. While it is able to implement any logic function dependent upon its inputs, it has very few inputs to work with. The PROM also has a fixed number of inputs and outputs and does not provide any architectural features to enhance logic design capability. The PAL, on the other hand, provides significant capability to implement logic functions. The programmable AND array allows equations with many inputs. Architectural features such as programmable I/O, internal registered feedback, and choice of output polarity allow optimization of pin allocation and logic equations. The PLA provides the most flexible architecture of the three for implementation of logic equations by utilizing a programmable AND array and a programmable OR array. However, the added flexibility of the PLA can seldom be effectively utilized. Further, the inherent loss in speed performance when using a PLA is increasingly unacceptable in high performance designs.

INDICATES PROGRAMMABLE CONNECTION
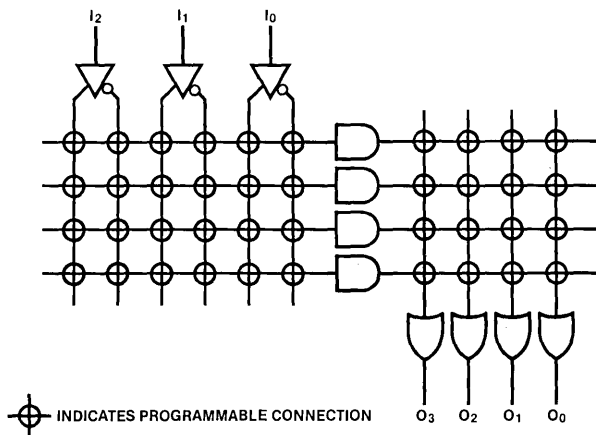
03862A-25

Figure 12. PLA Array Structure

PALs Aid High Performance 32-Bit CPU Design

Bradford S. Kitson and B. Joshua Rosen*

The Computervision Analytic Processing Unit (APU) was designed as a very high speed 32-bit super minicomputer intended for engineering applications. PALs were used extensively in the design. Instead of just replacing TTL SSI/MSI, this design utilized PALs as customizable logic building blocks, allowing powerful logic functions to be implemented in a minimum of space. The result is a machine which is twice as fast as competitive designs without an increase in board space. This paper describes several of the APU's "second generation" PAL based applications and illustrates the type of design techniques necessary to rise above the "first generation" TTL SSI/MSI replacement philosophy.

APPLICATION OF PALs IN THE COMPUTERVISION APU

The APU processor board set is divided into 4 modules, the Parser/Sequencer, which contains an instruction processor which fetches and decodes instructions in parallel with the execution unit; the Control Processor, which performs address and integer computations; the Floating Point Pipe, which performs both scalar and vector floating point operations; and the Cache/Address Translation Unit, which contains a 256 slot area page table entry cache and a 16K byte memory cache.

Approximately 25% of the chips in the APU board set are PALs. PALs were chosen for three reasons: flexibility, performance, and cost.

The APU is the first implementation of Computervision's new CPU architecture. As such, many aspects of the design were subject to change as the architecture evolved. The use of PALs permitted the designers of the machine to rapidly modify the hardware to fit the needs of this evolving architecture. In addition, new features and performance enhancements could easily be implemented with minimal impact on the development schedule.

The ability to generate a very large number of essentially custom ICs (there are over 200 different PAL codes used in the APU) resulted in a significant reduction in the processor's size while greatly enhancing its overall performance. The net result is apparent when one considers that although the APU and the Digital Equipment VAX-11/750, a gate array based machine, consume exactly the same amount of board space, the APU is more than twice as fast.

In fact, the Fortran performance of the APU is substantially faster than that of the VAX-11/780, a machine which consumes 5.2 times as much board space as the APU.

APU FLOATING POINT PIPE

The APU floating point pipe (FPP) was designed as a very high speed arithmetic extension to the APU execution engine. Unlike other comparable machines, the floating point arithmetic unit of the APU is an integral part of the internal architecture and not an optional add on. As a result, the FPP is used not only to accelerate scalar and vector floating point arithmetic, but also to perform byte, word, double word and quad word string operations. In addition, the FPP is also used to enhance the performance of important non-floating point instructions such as Procedure Call and Return.

The FPP board uses a total of 79 PALs for both control and data path applications. The remainder of this paper focuses on two particular subunits of the APU FPP: the multiplier and the barrel shifter.

3-TO-2 COUNTER

The multiplier section is the heart of the FPP. A block diagram of the multiplier appears in Figure 1. Double precision floating point multiplication requires the calculation of a 56-bit × 56-bit product. Unfortunately, 56 × 56 parallel multipliers do not exist on silicon. The best cost/performance solution is to use a number of smaller multipliers to build an intermediate sized parallel multiplier and then produce a large product (56 × 56) in multiple cycles.

The partial product generator logic of the FPP utilizes seven Am25S558 8 × 8 multiplier slices to implement an 8 × 56-bit multiplication array. Each multiplier chip produces a 16-bit product. In general, the most significant eight bits of each partial product generator must be added to the least significant eight bits of the next higher slice to generate the full 64-bit partial product. Exceptions are the most significant and least significant eight bits. For a graphic representation see Figure 2.

This technique also requires the ability to accumulate partial products with the partial products from previous cycles. Thus each cycle must be accompanied by two additions; the partial product summation and intermediate product accumulation.
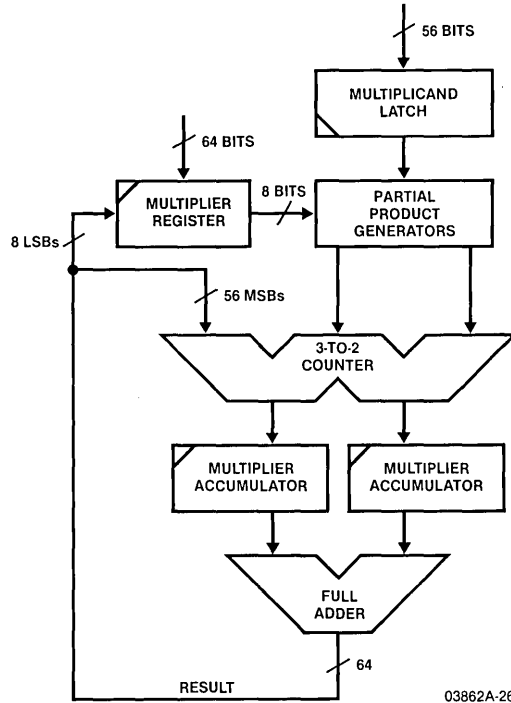
Figure 1. APU Multiplier (U.S. Patent Pending, Computervision Corp.)
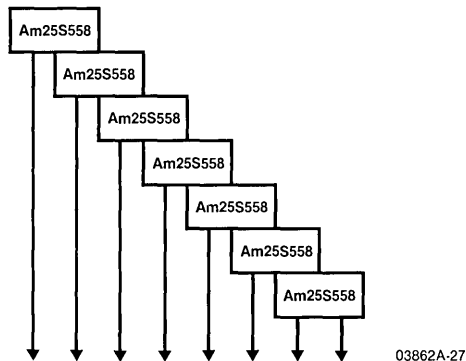
Figure 2. 64-Bit Partial Product Generation

The most straightforward way to accomplish this task is to follow the multipliers with two levels of lookahead adders, usually 74S181s. This technique results in a nanocycle time which is approximately 3 times longer than the partial product generation time of the 8 × 8 multipliers. This is clearly unacceptable. This scheme can be modified, however, by adding registers between each level of logic (see Figure 3). By pipelining the multiplier in this fashion, the nanocycle time can be reduced to something near the propagation delay time of the multiplier chips plus the clock to output time of the multiplier register plus the set-up time of the intermediate result register. The disadvantages of this scheme are increased pipe latency, caused by the two extra levels of pipelining, and a high part count. Still another technique involves replacing one level of the pipe and one level of

lookahead adders with carry save adders between the partial product generators and the pipeline registers. Carry save adders are used to implement a technique called 3-to-2 counting. As can be seen from Figure 4 any combination of 3 equally weighted bits can be recoded into a 2-bit field.

Thus it is possible to reduce the three operands generated by the multiplication process (the high and low partial products and the 64-bit intermediate product) into only two operands which may then be summed together in a single lookahead ALU. 3-to-2 recoding requires no carry propagate logic and is therefore very fast. Due to the speed of 3-to-2 counters, only one level of pipelining is required, which results in both a reduced parts count and a reduced pipe latency. The technique is ideally suited for implementation in PALs.
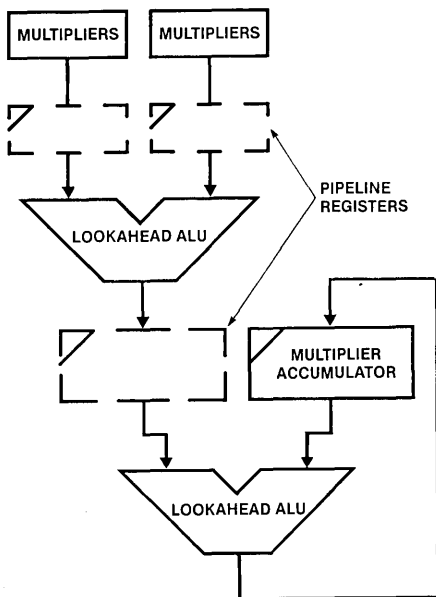
03862A-28

Figure 3. Two-Level Pipelined Multiplier Approach

INPUTS	OUTPUTS
0 0 0	0 0
0 0 1	0 1
0 1 0	0 1
0 1 1	1 0
1 0 0	0 1
1 0 1	1 0
1 1 0	1 0
1 1 1	1 1

03862A-29

Figure 4. 3-to-2 Counting

In the APU floating point engine, 16 AmPAL16R6s, programmed as triple 3-to-2 counters, are used to reduce the three multiplication operands to two intermediate results (see Figure 5 for a logic diagram). The registered outputs of the PALs are connected to the input buses of the Mantissa ALU, which is also used for floating point addition and subtraction. The Mantissa ALU then calculates the next intermediate product in parallel with the partial products calculations occurring in the 8 × 8 multipliers. This intermediate product and the new partial products are then recoded by the 3-to-2 counter PALs to form the next pair of intermediate results. This process continues until the complete 56 × 56 product is generated. Thus without adding pipe latency, the APU is able to accumulate partial products at a rate of 8 × 56 bits every 112ns, which happily coincides with the basic nanocycle time of the machine.
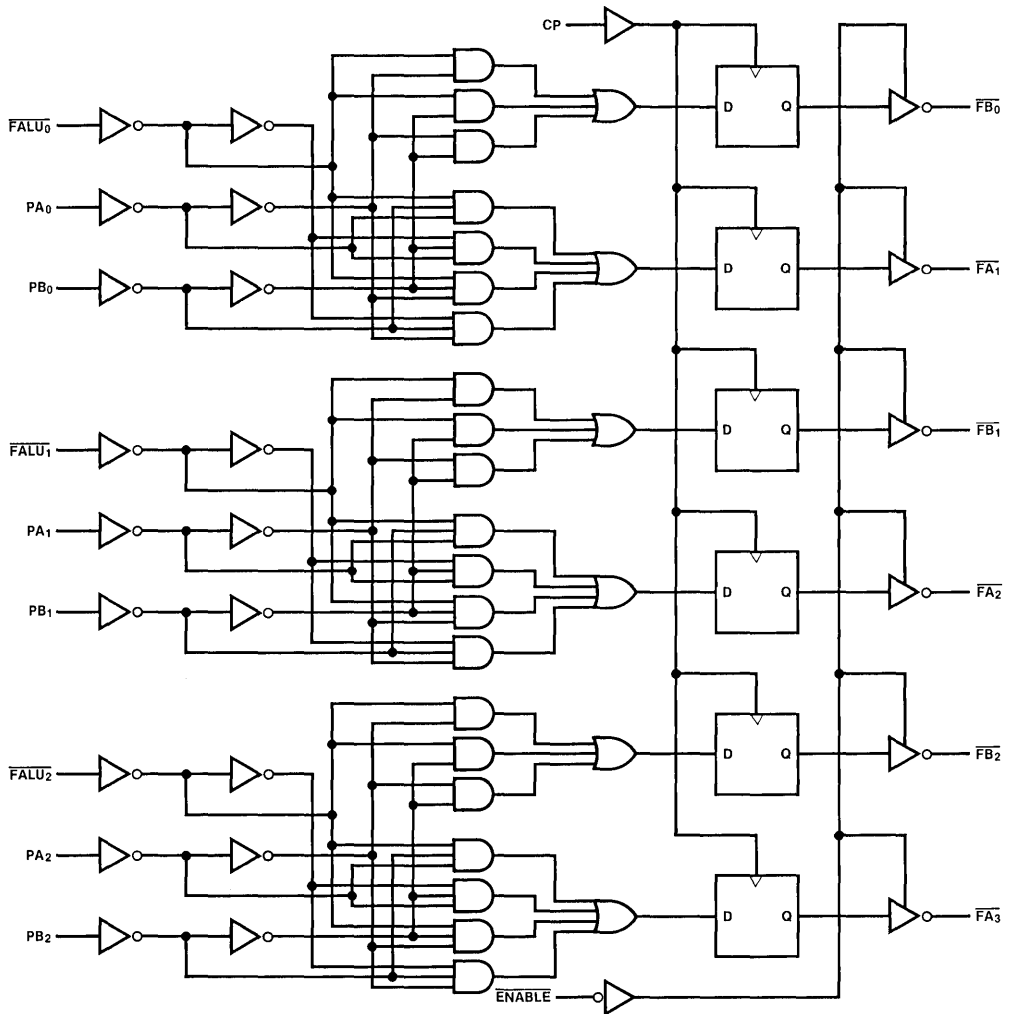
03862A-30

Figure 5. AmPAL16R6 3-to-2 Counter

BARREL SHIFTER

The APU Barrel Shifter, commonly referred to as the "Rosen Shifter", can perform left shift, right shift and rotate operations of from 0 to 63 bits in a single microcycle. The barrel shifter is used mainly for floating point prescale and normalize operations. A block diagram of the barrel shifter and its associated control logic is shown in Figure 6. The word rotater, nibble rotater and bit shift and mask logic comprise the three stages necessary to implement the barrel shifter. The prescale, leading zero detect and mask control logic comprise the logic required to control it.

The prescale logic converts the signed difference produced by the exponent arithmetic units as a result of the comparison of the two operand exponents, into an absolute shift distance which is then used to right shift (prescale) the Mantissa of the smaller operand of a floating point add or subtract operation. The leading zero detect logic determines the left shift distance required to produce a left justified (normalized) result. The mask control logic is used to convert rotated data to shifted data by masking off the appropriate leading or trailing bits to implement right or left shifts. These three sections are implemented in PALs, but will not be discussed in detail.
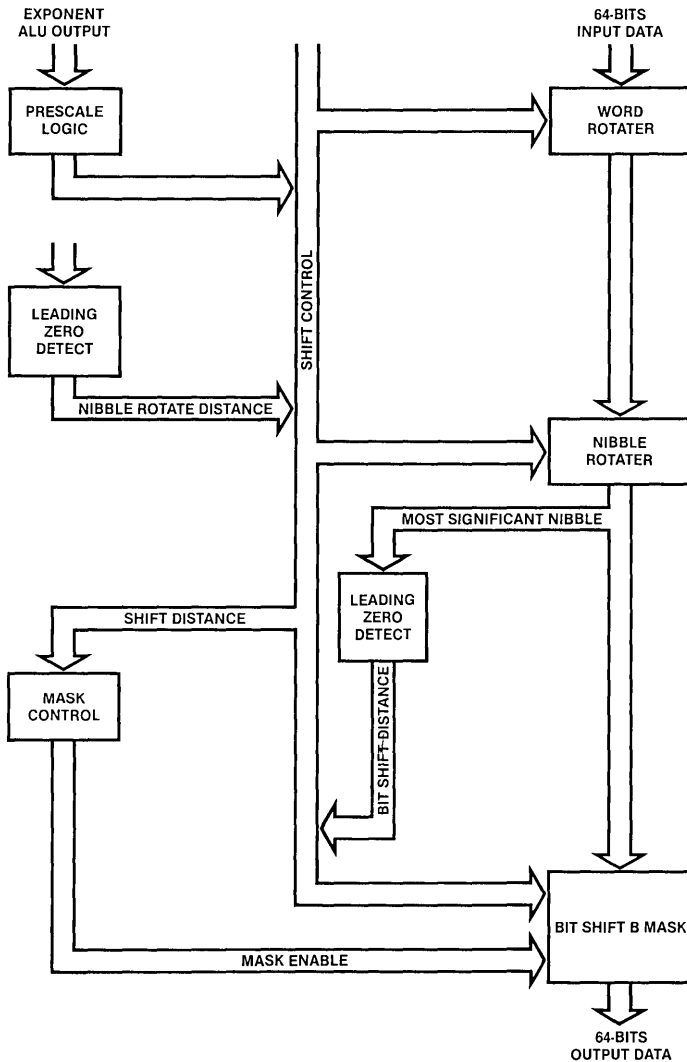
03862A-31

Figure 6. 64-Bit "Rosen Shifter"

To implement the three level 64-bit barrel shifter of Figure 6 in MSI requires the use of Am25S10 4-bit shifters (see Figure 7). The first level is the word rotater which performs a circular rotate of 0, 16, 32 or 48 bits. Although implementation is simple, the MSI solution requires 16 packages. The second level is the nibble rotater which is essentially identical to the word rotater but is wired to rotate 0, 4, 8 or 12 bits. The final stage of the barrel shifter requires not only bit rotate but also leading and trailing bit masking and sticky bit computation (i.e., the logical OR of the masked out bits). An MSI solution would require not only the 16 packages of Am25S10s, used in each of the preceding levels, but also 16 packages of AND gates, for masking, plus still another 16 packages of ANDs for the sticky bit computation. The control logic for performing the mask operation would probably require as much logic as the entire shift path. The more practical solution has usually been to build separate left and right shifters, 48 packages apiece, and not to implement a sticky bit at all.

The PAL implementation of a 64-bit rotater and shifter, with sticky bit computation, requires considerably fewer packages than a unidirectional MSI shifter.

The word rotater consists of 8 identical PALs programmed as two four-bit rotaters per package. The logic diagram of the word rotater is shown in Figure 8. The nibble shifter requires four Am25S10s and 8 PALs programmed as 6-bit wide 4 place shifters. The logic diagram of a nibble shifter is shown in Figure 9. The bit shift and mask logic requires sixteen PALs in the data path and two PALs in the control path. The logic diagram for a shift and mask PAL appears in Figure 10, but some explanation is required to understand the innovative design technique used to implement it.
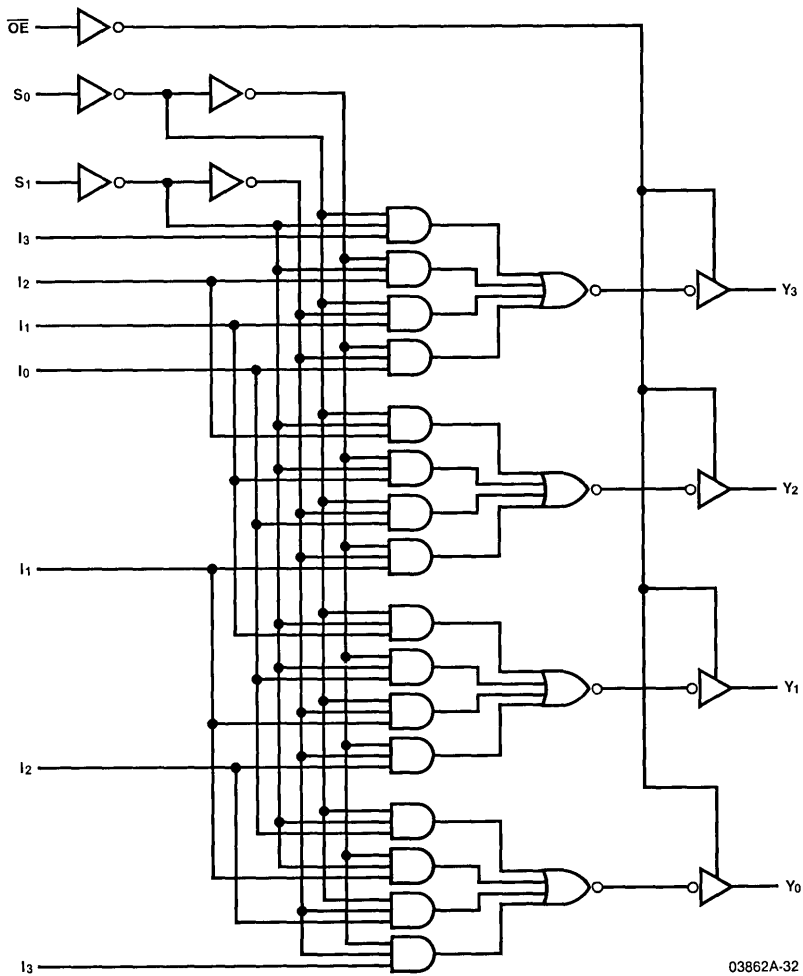
Figure 7. Am25S10 4-Bit Shifter
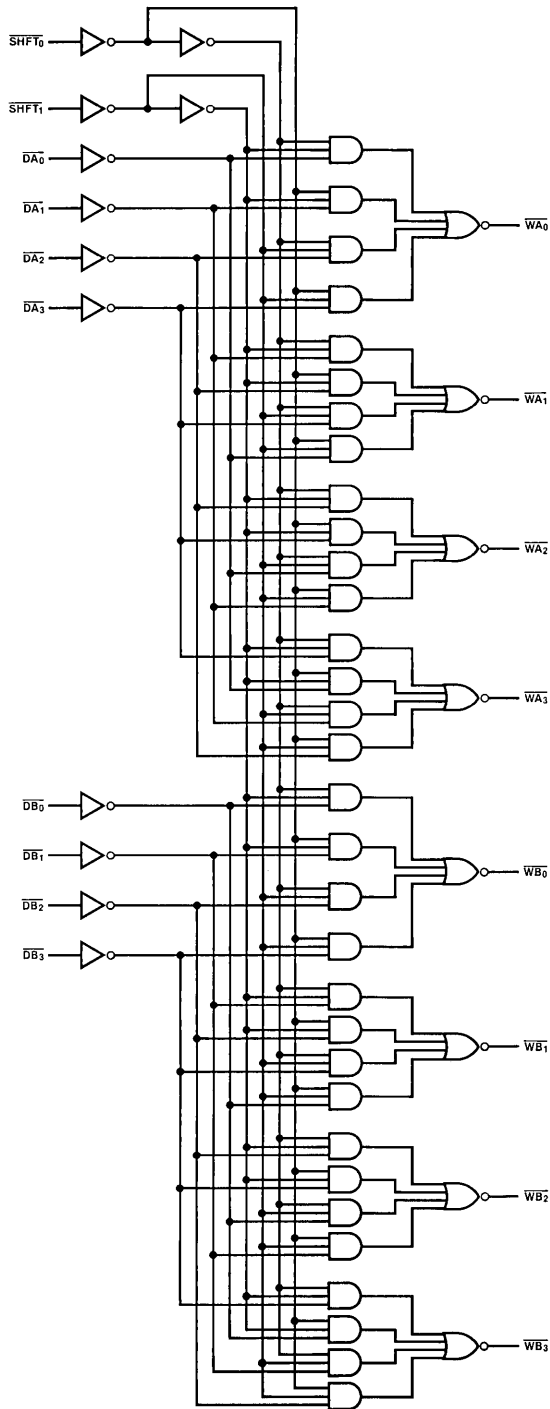
03862A-32

Figure 8. PAL Word Rotater
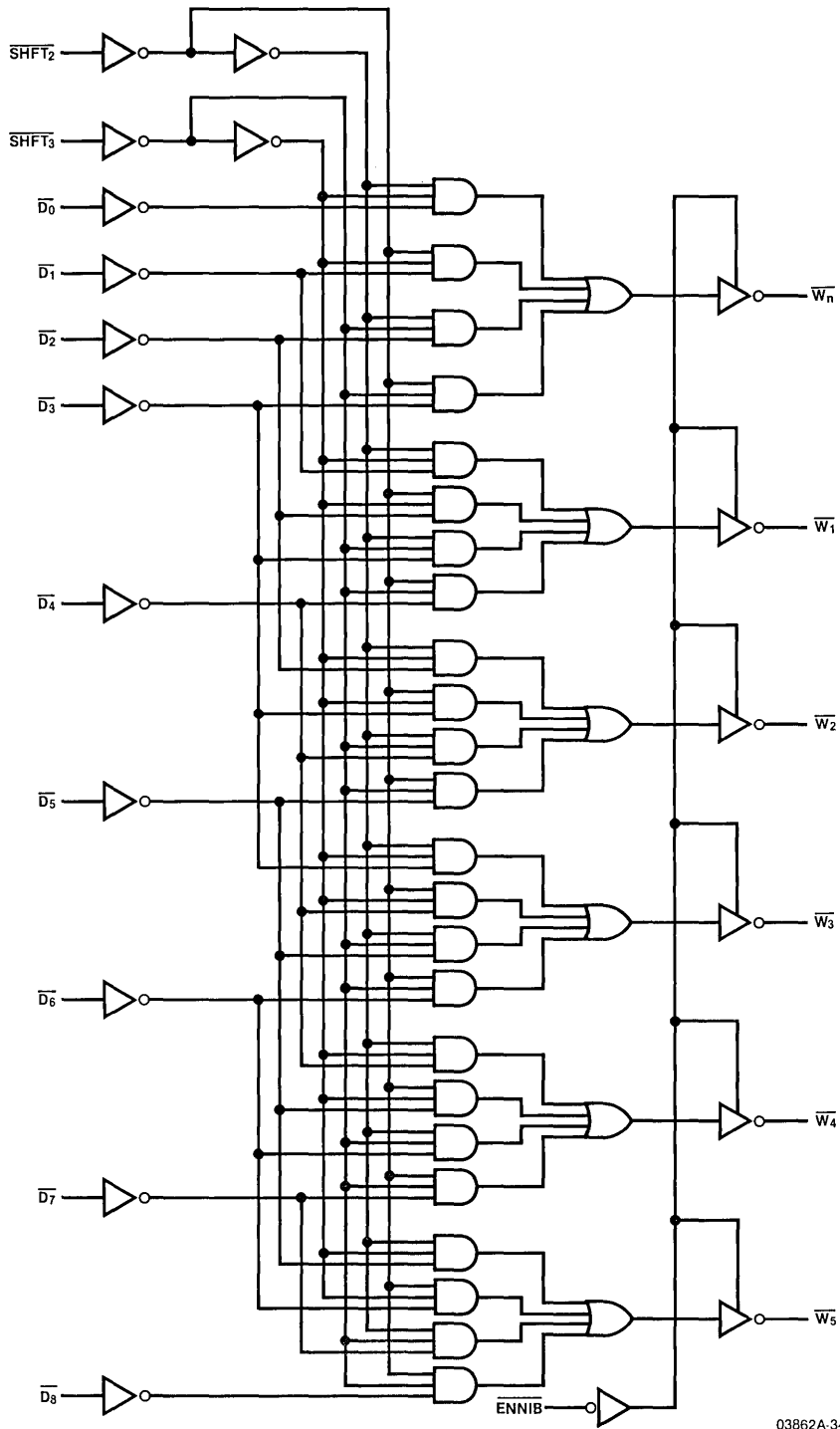
03862A-33

Figure 9. PAL Nibble Shifter
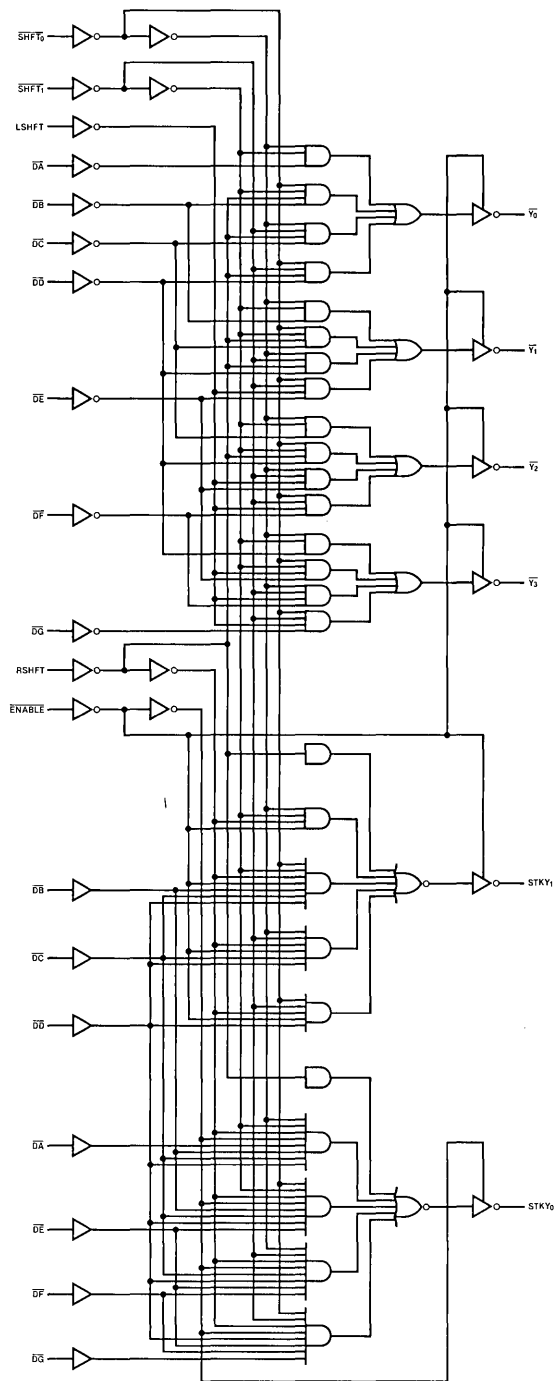
03862A-34

Figure 10. PAL Bit Shift and Mask Slice

03862A-35

The technique used to implement the masking function required for shifting is called "nearest neighbor masking", (U.S. patent pending, Computervision Corp.). Each of the shift and mask PALs has an enable input from one of the mask control PALs. In addition, each shift and mask PAL is also connected to the enable inputs from its left and right hand neighbors (see Figure 11). The mask control PAL input determines if all four bits from the slice should be masked off. The enables from the adjacent slice determine if a PAL is at a shift boundary. Should only one of the neighboring slices be disabled then the shift and mask PAL will mask off from 0 to 3 of the bits adjacent to the disabled slice, depending on the bit rotation distance. In this way, the 64-bit masking operation can be implemented with only 16 control lines as opposed to at least 64 for an SSI/MSI solution.

In addition to performing the final shift and mask operation, the bit shifter PALs also compute the logical OR of the masked out bits at each slice position. These outputs are then logically ORed together to generate a sticky bit. The extra hardware required is less than two SSI packages.

The entire PAL Barrel Shifter requires 38 devices to implement 64-bit rotation, left shifting, right shifting and sticky bit accumulation. An MSI based left/right shifter, without sticky bit computation, would require a minimum of 96 parts. In addition, the logic required for implementing the prescale and normalize operations, not discussed, is also significantly reduced through the use of PALs.

CONCLUSION

The full potential of PALs has begun to be realized with the availability of high speed devices. PALs now allow the designer to architect the device to fit the application instead of architecting the application to fit the device.

The 3-to-2 counter design example shown illustrated the implementation of a unique architecture in PALs, resulting in significant part count reduction and throughput increase over a comparable TTL solution. The barrel shifter design example shown illustrated the innovative implementation of an application not previously feasible in TTL.
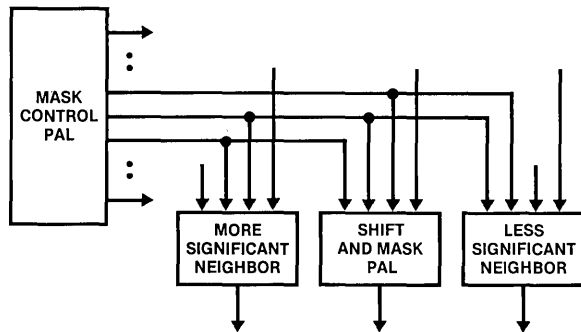
03862A-36

Figure 11. PAL "Nearest Neighbor" Interconnect

Product Specifications

AMD 20-Pin PAL Family
AMD Half Power PAL Family
AmPAL22V10 Advanced Information
AmPL64S16 Advanced Information
Am27S12A/13A, Am27S12/13 2048-Bit Generic Series Bipolar PROM
Am27S18A/19A, Am27S18/19 256-Bit Generic Series Bipolar PROM
Am27S20A/21A, Am27S20/21 1024-Bit Generic Series Bipolar PROM

AMD 20-Pin PAL* Family

20-Pin IMOX™ Programmable Array Logic Elements
Advanced Micro Devices

DISTINCTIVE CHARACTERISTICS

- **Fast**
  - High speed "A" versions
    (t_{pd} = 25ns, t_s = 20ns, t_{co} = 15ns, max)
  - Standard speed versions
    (t_{pd} = 35ns, t_s = 30ns, t_{co} = 25ns, max)
- **Flexible**
  - User programmability allows customized designs
  - Eases design updates in prototype or product
- **Low Cost**
  - Reduces board space/chip count
  - Reduces design time
  - Reduces inventory cost
- **Reliable**
  - Proven Platinum-Silicide fuse technology
  - Fully AC and DC tested
  - Preload of output registers allows full logical testing

FUNCTIONAL DESCRIPTION

AMD PALs are high speed electrically programmable array logic elements. They utilize the familiar sum-of-products (AND-OR) structure allowing users to program custom logic functions to fit most applications precisely.

Initially the AND gates are connected, via fuses, to both the true and complement of every input. By selective programming of fuses the AND gates may be "connected" to only the true input (by blowing the complement fuse), to only the complement input (by blowing the true fuse), or to neither type of input (by blowing both fuses) establishing a logical "don't care." When both the true and complement fuses are left intact a logical false results on the output of the AND gate. An AND gate with all fuses blown will assume the logical true state. The outputs of the AND gates are connected to fixed OR gates. The only limitations imposed are the number of inputs to the AND gates (up to 16) and the number of AND gates per OR (up to 8).

The part types in the AMD PAL family are differentiated by the allocation of registered (with internal feedback) and combinatorial (bi-directional and dedicated) outputs. All combinatorial AMD PALs are available in both active HIGH (AND-OR) and active LOW (AND-OR-INVERT) versions.

AMD PAL FAMILY CHARACTERISTICS

All members of the AMD PAL family have common electrical characteristics and programming procedures. All parts in this family are produced with a fusible link at each input to the AND gate array. Connections may be selectively removed by applying appropriate voltages to the circuit.

All parts are fabricated with AMD's fast programming, highly reliable Platinum-Silicide fuse technology. Utilizing an easily implemented programming algorithm, these products can be rapidly programmed to any customized pattern. Extra test words are pre-programmed during manufacturing to insure extremely high field programming yields (>98%), and provide extra test paths to achieve excellent parametric correlation.

Platinum-Silicide was selected as the fuse link material to achieve a well controlled melt rate resulting in large non-conductive gaps that ensure very stable, long term reliability. Extensive operating testing has proven that this low-field, large-gap technology offers the best reliability for fusible link programmable logic.

The AMD PAL family is manufactured using Advanced Micro Devices' selective oxidation process, IMOX. This advanced process permits an increase in density and a decrease in internal capacitance resulting in the fastest possible programmable logic devices.

The AMD PAL family also incorporates the unique capability of preloading the output registers during testing to any desired value. Preload is invaluable when testing the logical functionality of a programmed AMD PAL.

AMD PAL FAMILY TABLE

Part Number	Array Inputs	Logic	OE	Outputs	t_{pd} (MAX) STD	A	t_s (MAX) STD	A	t_{co} (MAX) STD	A	
AmPAL16R8	(8) Dedicated (8) Feedback	(8) 8-Wide AND-OR	Dedicated	Registered Inverting	–	–	30	20	25	15	ns
AmPAL16R6	(8) Dedicated (6) Feedback (2) Bidirectional	(6) 8-Wide AND-OR	Dedicated	Registered Inverting	35	25	30	20	25	15	ns
		(2) 7-Wide AND-OR-INVERT	Programmable	Bidirectional							
AmPAL16R4	(8) Dedicated (4) Feedback (4) Bidirectional	(4) 8-Wide AND-OR	Dedicated	Registered Inverting	35	25	30	20	25	15	ns
		(4) 7-Wide AND-OR-INVERT	Programmable	Bidirectional							
AmPAL16L8	(10) Dedicated (6) Bidirectional	(8) 7-Wide AND-OR-INVERT	Programmable	(6) Bidirectional (2) Dedicated	35	25	–	–	–	–	ns
AmPAL16H8	(10) Dedicated (6) Bidirectional	(8) 7-Wide AND-OR	Programmable	(6) Bidirectional (2) Dedicated	35	25	–	–	–	–	ns
AmPAL16LD8	(10) Dedicated (6) Bidirectional	(8) 8-Wide AND-OR-INVERT	–	Dedicated	35	25	–	–	–	–	ns
AmPAL16HD8	(10) Dedicated (6) Bidirectional	(8) 8-Wide AND-OR	–	Dedicated	35	25	–	–	–	–	ns

*PAL is a registered trademark of Monolithic Memories, Inc.
IMOX is a trademark of Advanced Micro Devices, Inc.

MAXIMUM RATINGS (Above which the useful life may be impaired)

Storage Temperature	−65 to +150°C
Temperature (Ambient) Under Bias	−55 to +125°C
Supply Voltage to Ground Potential (Pin 20 to Pin 10) Continuous	−0.5 to +7V
DC Voltage Applied to Outputs (Except During Programming)	−0.5V to +V_{CC} max
DC Voltage Applied to Outputs During Programming	21V
Output Current Into Outputs During Programming (Max Duration of 1 sec)	200mA
DC Input Voltage	−0.5 to +5.5V
DC Input Current	−30 to +5mA

OPERATING RANGE

Parameters	Description	Commercial Min	Commercial Max	Military Min	Military Max	Units
V_{CC}	Supply Voltage	4.75	5.25	4.50	5.50	V
T_A	Operating Free Air Temperature	0	75	−55		°C
T_C	Operating Case Temperature				125	°C

ELECTRICAL CHARACTERISTICS OVER OPERATING RANGE (Unless Otherwise Noted)

Parameters	Description	Test Conditions			Min	Typ (Note 1)	Max	Units
V_{OH}	Output HIGH Voltage	V_{CC} = MIN, V_{IN} = V_{IH} or V_{IL}	I_{OH} = −3.2mA	COM'L	2.4	3.5		Volts
			I_{OH} = −2mA	MIL				
V_{OL}	Output LOW Voltage	V_{CC} = MIN, V_{IN} = V_{IH} or V_{IL}	I_{OL} = 24mA	COM'L			0.50	Volts
			I_{OL} = 12mA	MIL				
V_{IH} (Note 2)	Input HIGH Level	Guaranteed input logical HIGH voltage for all inputs			2.0			Volts
V_{IL} (Note 2)	Input LOW Level	Guaranteed input logical LOW voltage for all inputs					0.8	Volts
I_{IL}	Input LOW Current	V_{CC} = MAX, V_{IN} = 0.40V				−20	−250	μA
I_{IH}	Input HIGH Current	V_{CC} = MAX, V_{IN} = 2.7V					25	μA
I_I	Input HIGH Current	V_{CC} = MAX, V_{IN} = 5.5V					1.0	mA
I_{SC}	Output Short Circuit Current	V_{CC} = MAX, V_{OUT} = 0.5V (Note 3)			−30	−60	−90	mA
I_{CC}	Power Supply Current	All inputs = GND, V_{CC} = MAX	16L8, 16H8, 16HD8, 16LD8 16L8A, 16H8A, 16HD8A, 16LD8A			110	155	mA
			16R8, 16R6, 16R4 16R8A, 16R6A, 16R4A			120	180	
V_I	Input Clamp Voltage	V_{CC} = MIN, I_{IN} = −18mA				−0.9	−1.2	Volts
I_{OZH}	Output Leakage Current (Note 4)	V_{CC} = MAX, V_{IL} = 0.8V V_{IH} = 2.0V	V_O = 2.7V				100	μA
I_{OZL}			V_O = 0.4V				−100	
C_{IN}	Input Capacitance	V_{IN} = 2.0V @f = 1MHz (Note 5)				6		pF
C_{OUT}	Output Capacitance	V_{OUT} = 2.0V @f = 1MHz (Note 5)				9		

Notes: 1. Typical limits are at V_{CC} = 5.0V and T_A = 25°C.
2. These are absolute values with respect to device ground and all overshoots due to system or tester noise are included.
3. Not more than one output should be tested at a time. Duration of the short circuit should not be more than one second. V_{OUT} = 0.5V has been chosen to avoid test problems caused by tester ground degradation.
4. I/O pin leakage is the worst case of I_{OZX} or I_{IX} (where X = H or L).
5. These parameters are not 100% tested, but are periodically sampled.

SWITCHING CHARACTERISTICS OVER OPERATING RANGE (Unless otherwise noted)
HIGH SPEED

Parameters	Description	Test Conditions	Typ (Note 1)	COM'L Min	COM'L Max	MIL Min	MIL Max	Units
t_{PD}	Input or Feedback to Non-Registered Output 16L8A, 16R6A, 16R4A, 16LD8A, 16H8A, 16HD8A		12		25		30	ns
t_{EA}	Input to Output Enable 16L8A, 16R6A, 16R4A, 16H8A		12		25		30	ns
t_{ER}	Input to Output Disable 16L8A, 16R6A, 16R4A, 16H8A	COM'L $R_1 = 200$ $R_2 = 390$	12		25		30	ns
t_{PZX}	Pin 11 to Output Enable 16R8A, 16R6A, 16R4A		8		20		25	ns
t_{PXZ}	Pin 11 to Output Disable 16R8A, 16R6A, 16R4A		8		20		25	ns
t_{CO}	Clock to Output 16R8A, 16R6A, 16R4A		8		15		20	ns
t_S	Input or Feedback Setup Time 16R8A, 16R6A, 16R4A	MIL $R_1 = 390$ $R_2 = 750$	10	20		25		ns
t_H	Hold Time 16R8A, 16R6A, 16R4A		−10	0		0		ns
t_P	Clock Period			35		45		ns
t_W	Clock Width			15		20		ns
f_{MAX}	Maximum Frequency				28.5		22	MHz

Notes: 1. Typical limits are at $V_{CC} = 5.0V$ and $T_A = 25°C$.
2. t_{PD} is tested with switch S_1 closed and $C_L = 50pF$.
3. For three-state outputs, output enable times are tested with $C_L = 50pF$ to the 1.5V level; S_1 is open for high impedance to HIGH tests and closed for high impedance to LOW tests. Output disable times are tested with $C_L = 5pF$. HIGH to high impedance tests are made to an output voltage of $V_{OH} - 0.5V$ with S_1 open; LOW to high impedance tests are made to the $V_{OL} + 0.5V$ level with S_1 closed.

SWITCHING CHARACTERISTICS OVER OPERATING RANGE (Unless otherwise noted)
STANDARD SPEED

Parameters	Description	Test Conditions	Typ (Note 1)	COM'L Min	COM'L Max	MIL Min	MIL Max	Units
t_{PD}	Input or Feedback to Non-Registered Output 16L8, 16R6, 16R4, 16LD8, 16H8, 16HD8		17		35		40	ns
t_{EA}	Input to Output Enable 16L8, 16R6, 16R4, 16H8		17		35		40	ns
t_{ER}	Input to Output Disable 16L8, 16R6, 16R4, 16H8	COM'L $R_1 = 200$ $R_2 = 390$	17		35		40	ns
t_{PZX}	Pin 11 to Output Enable 16R8, 16R6, 16R4		12		25		25	ns
t_{PXZ}	Pin 11 to Output Disable 16R8, 16R6, 16R4		12		25		25	ns
t_{CO}	Clock to Output 16R8, 16R6, 16R4		12		25		25	ns
t_S	Input or Feedback Setup Time 16R8, 16R6, 16R4	MIL $R_1 = 390$ $R_2 = 750$	15	30		35		ns
t_H	Hold Time 16R8, 16R6, 16R4		−10	0		0		ns
t_P	Clock Period			55		60		ns
t_W	Clock Width			20		25		ns
f_{MAX}	Maximum Frequency				18		16.5	MHz

Notes: 1. Typical limits are at $V_{CC} = 5.0V$ and $T_A = 25°C$.
2. t_{PD} is tested with switch S_1 closed and $C_L = 50pF$.
3. For three-state outputs, output enable times are tested with $C_L = 50pF$ to the 1.5V level; S_1 is open for high impedance to HIGH tests and closed for high impedance to LOW tests. Output disable times are tested with $C_L = 5pF$. HIGH to high impedance tests are made to an output voltage of $V_{OH} - 0.5V$ with S_1 open; LOW to high impedance tests are made to the $V_{OL} + 0.5V$ level with S_1 closed.

AC TEST LOAD
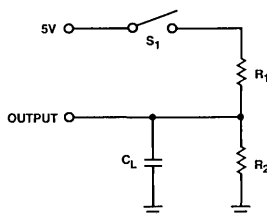
BPM-370

SWITCHING WAVEFORMS
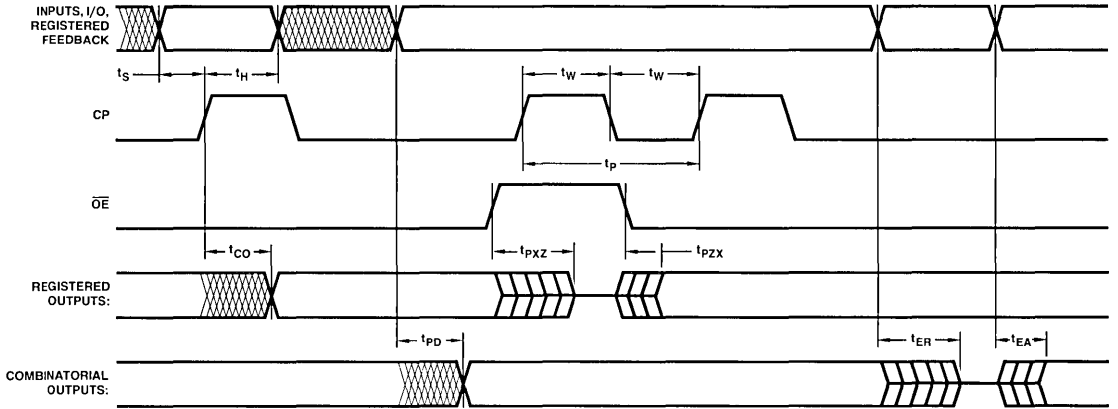
BPM-371

KEY TO TIMING DIAGRAM

WAVEFORM	INPUTS	OUTPUTS	WAVEFORM	INPUTS	OUTPUTS
	MUST BE STEADY	WILL BE STEADY		DON'T CARE; ANY CHANGE PERMITTED	CHANGING; STATE UNKNOWN
				DOES NOT APPLY	CENTER LINE IS HIGH IMPEDANCE "OFF" STATE

LOGIC DIAGRAM AmPAL16R8/AmPAL16R8A

INPUTS (0–31)

PRODUCT TERMS (0–63)
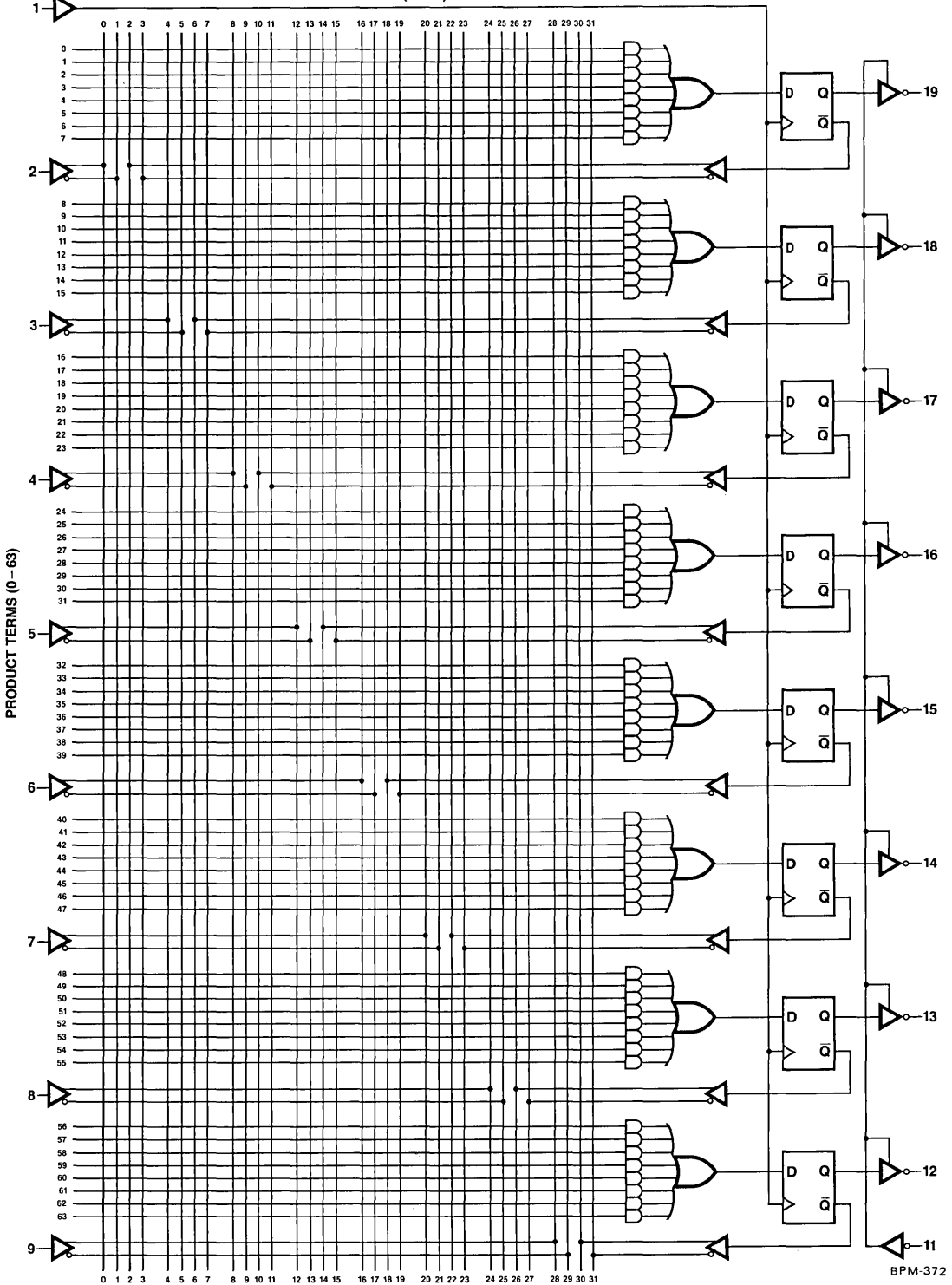
BPM-372

LOGIC DIAGRAM AmPAL16R6/AmPAL16R6A

INPUTS (0–31)
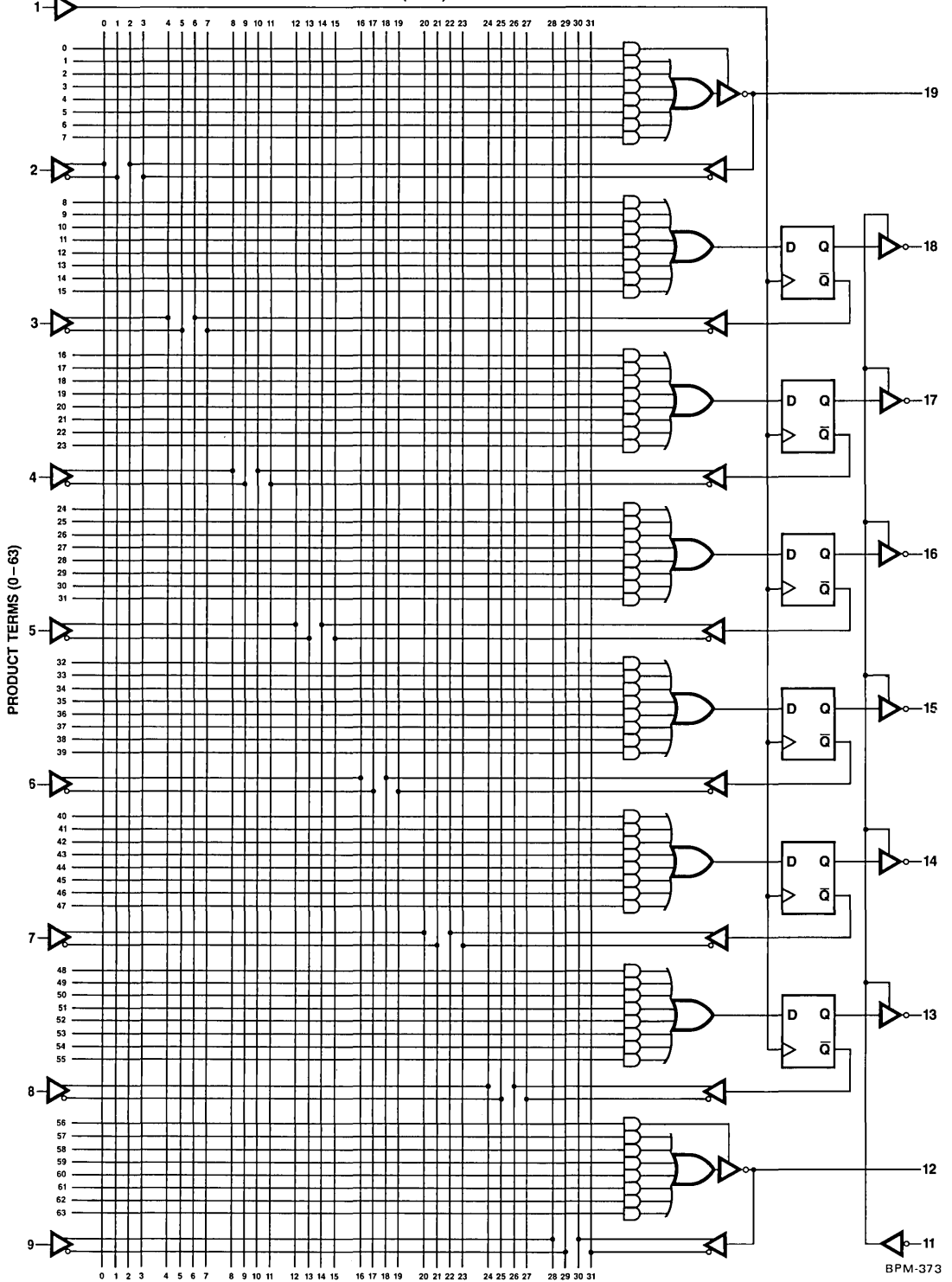
BPM-373

LOGIC DIAGRAM AmPAL16R4/AmPAL16R4A

INPUTS (0–31)

PRODUCT TERMS (0–63)

BPM-374

2-7

LOGIC DIAGRAM AmPAL16L8/AmPAL16L8A
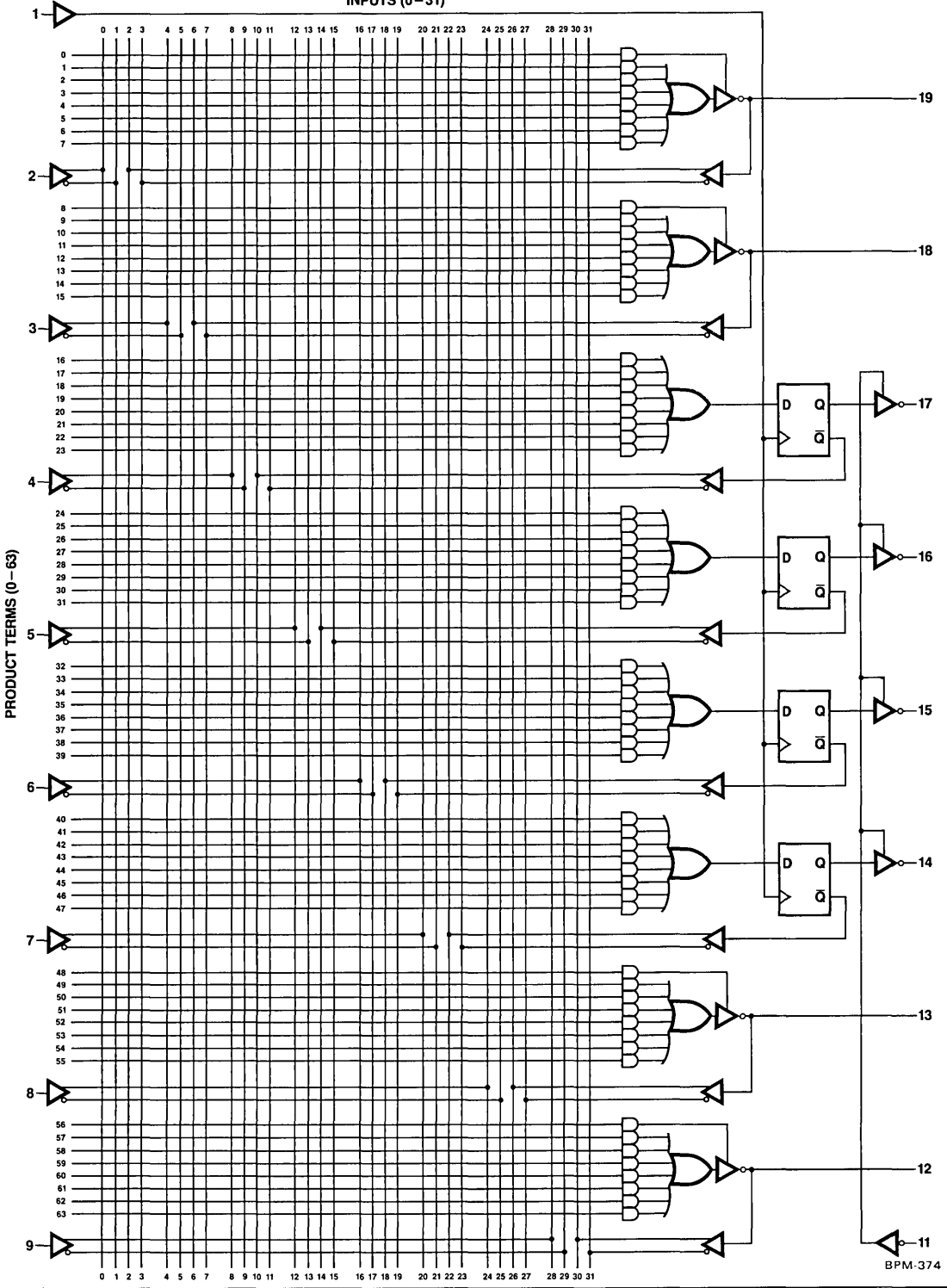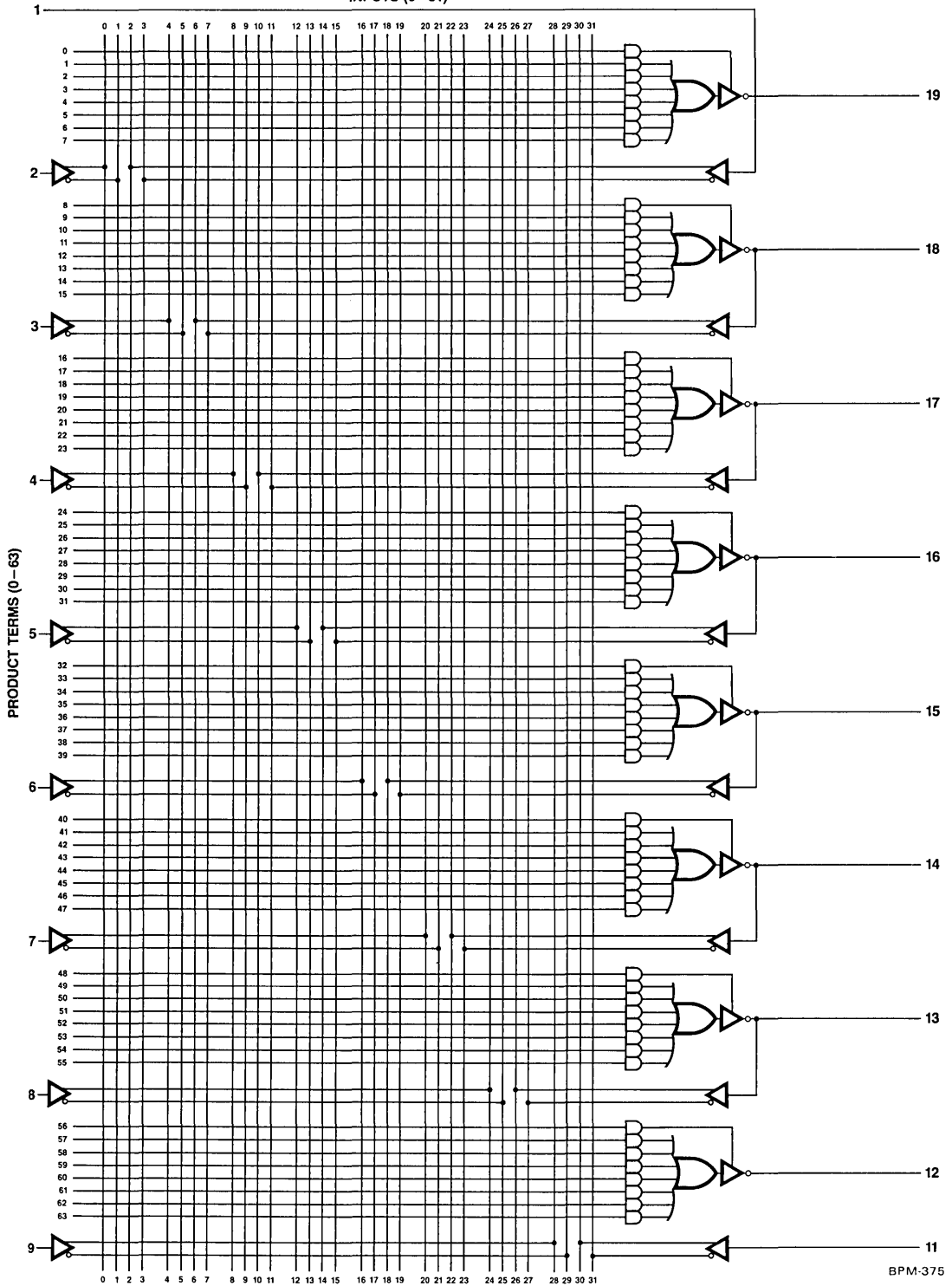
INPUTS (0-31)

BPM-375

LOGIC DIAGRAM AmPAL16LD8/AmPAL16LD8A

INPUTS (0–31)
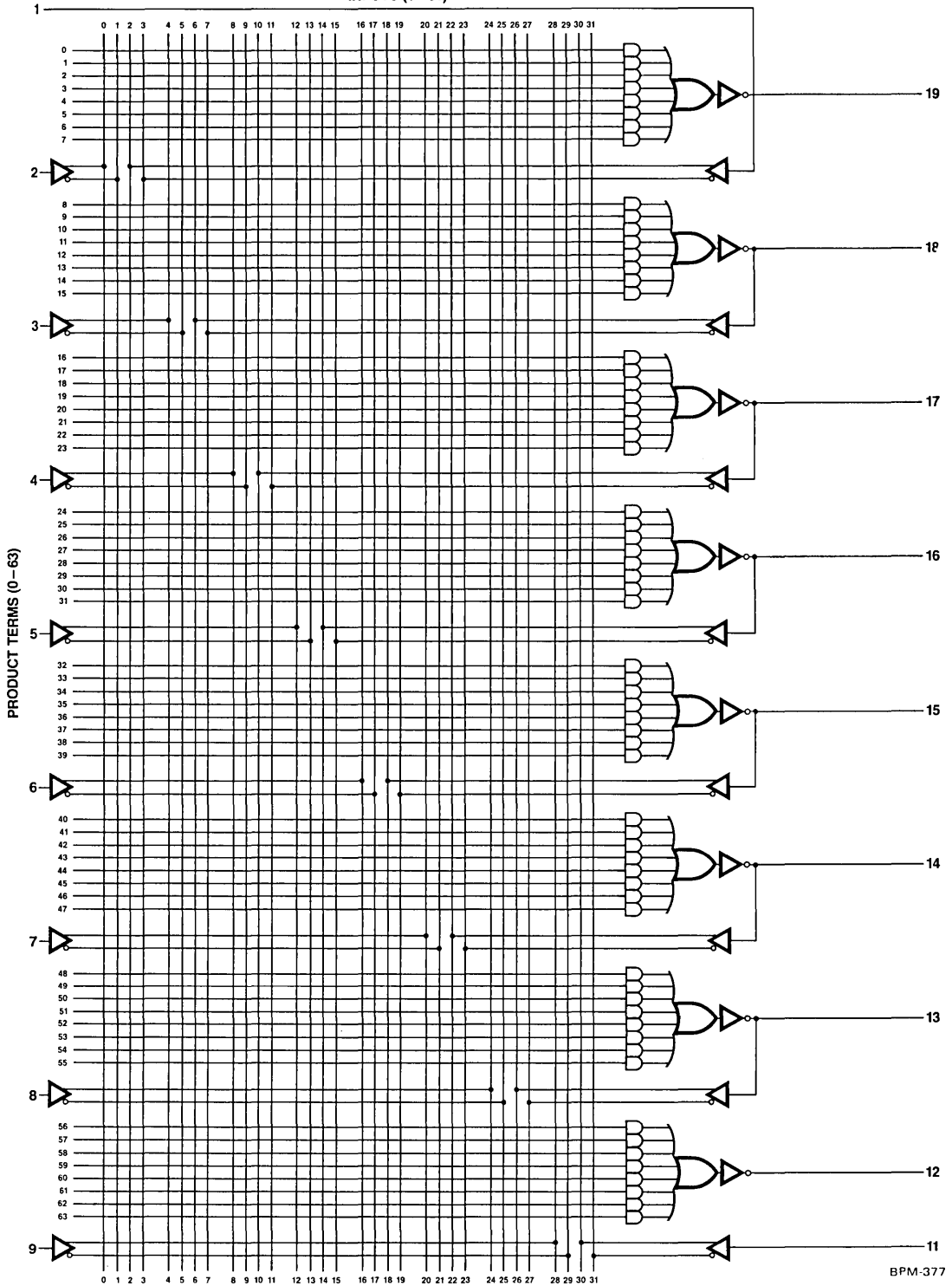
LOGIC DIAGRAM AmPAL16HD8/AmPAL16HD8A

INPUTS (0–31)
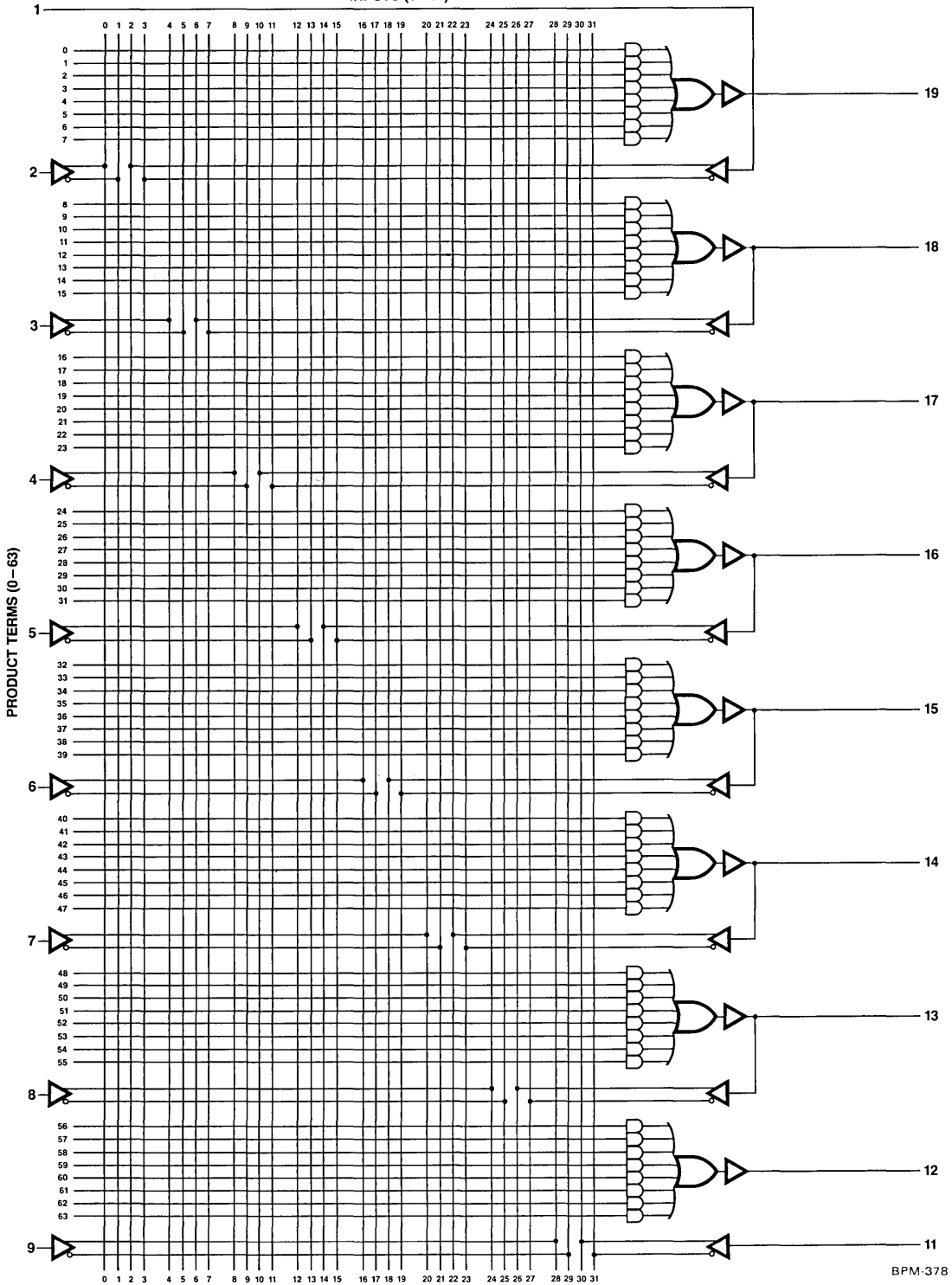
PRODUCT TERMS (0–63)

PROGRAMMING

Each AMD PAL fuse is programmed with a simple sequence of voltages applied to two control pins (1 and 11) and a programming voltage pulse applied to the output under programming. Addressing of the 2048 element fuse array is accomplished with normal TTL levels on eight input pins (five select the input line number and three select the product term number). V_{CC} is maintained at a normal level throughout the programming and verify cycle — no extra high levels are required.

The necessary sequence levels for programming any fuse is shown in the Programming Waveforms. The address of each fuse in terms of Input Line Number and Product Term Line Number is defined by the Fuse Address Tables 1 and 2. Current, voltage and timing requirements for each pin are specified in the Programming Parameter Table below.

The 16L8, 16R8, 16R6, 16R4, 16H8, 16LD8 and 16HD8 use identical programming conditions and sequences.

After all programming has been completed, the entire array should be reverified at V_{CCL} and again at V_{CCH}. Reverification can be accomplished by reading all eight outputs in parallel rather than one at a time. The array fuse verification cycle checks that the correct array fuses have been blown and can be sensed by the outputs.

AMD PALs have been designed with many internal test features that are used to assure high programming yield and correct logical operation for a correctly programmed part.

An additional fuse is provided on each AMD PAL circuit to prevent unauthorized copying of AMD PAL fuse patterns when design security is desired. Blowing the security fuse blocks entry to the fuse pattern verify mode.

To blow the security fuse:
1. Power up part to V_{CCP}
2. Raise Pin 5 to V_{HH}.
3. Pulse Pin 11 from ground to V_{OP} for a 50μsec duration.
4. Perform a normal end-of-programming verify cycle at V_{CCL} and V_{CCH}. All fuse locations should be sensed as blown if the security fuse has been successfully blown.

Note that parts with the security fuse blown may not be returned as programming rejects.

AMD PALs normally have high programming yields (>98%). Programming yield losses are frequently due to poor socket contact, equipment out of calibration or improperly used.

PROGRAMMING PARAMETERS $T_A = 25°C$

Parameters	Description		Min	Typ	Max	Units
V_{HH}	Control Pin Extra High Level	Pin 1 @ 10−40mA	10	11	12	Volts
		Pin 11 @ 10−40mA	10	11	12	
V_{OP}	Program Voltage Pins 12−19 @ 15−200mA		18	20	22	Volts
V_{IHP}	Input High Level During Programming and Verify		2.4	5	5.5	Volts
V_{ILP}	Input Low Level During Programming and Verify		0.0	0.3	0.5	Volts
V_{CCP}	V_{CC} During Programming @ $I_{CC} = 50-200mA$		5	5.2	5.5	Volts
V_{CCL}	V_{CC} During First Pass Verification @ $I_{CC} = 50-200mA$		4.1	4.3	4.5	Volts
V_{CCH}	V_{CC} During Second Pass Verification @ $I_{CC} = 50-200mA$		5.4	5.7	6.0	Volts
V_{Blown}	Successful Blown Fuse Sense Level @ Output	16L8, 16R8, 16R6, 16R4, 16LD8 16L8A, 16R8A, 16R6A, 16R4A, 16LD8A		0.3	0.5	Volts
		16H8, 16HD8, 16H8A, 16HD8A	2.4	3		
dV_{OP}/dt	Rate of Output Voltage Change		20		250	V/μsec
dV_{11}/dt	Rate of Fusing Enable Voltage Change (Pin 11 Rising Edge)		100		1000	V/μsec
t_P	Fusing Time First Attempt		40	50	100	μsec
	Subsequent Attempts		4	5	10	msec
t_D	Delays Between Various Level Changes		100	200	1000	ns
t_V	Period During which Output is Sensed for V_{Blown} Level				500	ns
V_{ONP}	Pull-Up Voltage On Outputs Not Being Programmed		$V_{CCP} - 0.3$	V_{CCP}	$V_{CCP} + 0.3$	Volts
R	Pull-Up Resistor On Outputs Not Being Programmed		1.9	2	2.1	KΩ

AMD PAL PROGRAMMING EQUIPMENT INFORMATION

Source and Location	Data I/O 10525 Willows Rd., N.E. Redmond, WA 98052	Kontron Electronics, Inc. 630 Price Ave. Redwood City, CA 94063	Stag Microsystems 528-5 Weddel Dr. Sunnyvale, CA 94086	Structured Design, Inc. 1700 Wyatt Dr. #3 Santa Clara, CA 95054	Digilec, Inc. 7335 E. Acoma Dr. Dept-103 Scottsdale, AZ 85260
Programmer Model(s)	Model-100, 29, 19 or 17	Model-MPP-80S or EPP80	Model-PPX	SD1000	TBA
			Model ZL-30		
AMD PAL Personality Module	Logicpak 950-1942-001	MOD-33	PPM2200	On Board	–
			On Board ZL-30		
Socket Adapter	715-1947-003	SA37	Am202S	On Board	–
			On Board ZL-30		

The machines noted above have been qualified by AMD to insure high programming yields. Check with the factory to determine the current status of vendors noted TBA or other available models.

PROGRAMMING WAVEFORMS

PINS 2-9 — INPUT LINE NUMBER AND PRODUCT TERM LINE NUMBER ADDRESSES — V_{IHP} / V_{ILP}

PIN 1 — V_{HH} / V_{IHP} / V_{ILP}

FUSING TIME

PROGRAMMED OUTPUT — Z — Z — H — L — V_{OP} / V_{OH} / V_{OL}

PIN 11 — V_{HH} / V_{ILP}

t_D t_D t_D t_P t_D t_D t_D t_D t_V t_D CYCLE ENDS

PROGRAMMING — VERIFY

BPM 379

TABLE 1. INPUT ADDRESSING

Input Line Number	Input Line Number Address Pin States				
	9	8	7	6	5
0	L	L	L	L	L
1	L	L	L	L	H
2	L	L	L	H	L
3	L	L	L	H	H
4	L	L	H	L	L
5	L	L	H	L	H
6	L	L	H	H	L
7	L	L	H	H	H
8	L	H	L	L	L
9	L	H	L	L	H
10	L	H	L	H	L
11	L	H	L	H	H
12	L	H	H	L	L
13	L	H	H	L	H
14	L	H	H	H	L
15	L	H	H	H	H
16	H	L	L	L	L
17	H	L	L	L	H
18	H	L	L	H	L
19	H	L	L	H	H
20	H	L	H	L	L
21	H	L	H	L	H
22	H	L	H	H	L
23	H	L	H	H	H
24	H	H	L	L	L
25	H	H	L	L	H
26	H	H	L	H	L
27	H	H	L	H	H
28	H	H	H	L	L
29	H	H	H	L	H
30	H	H	H	H	L
31	H	H	H	H	H

$L = V_{ILP}$
$H = V_{IHP}$

SIMPLIFIED PROGRAMMING DIAGRAM

V_{CCP} V_{ONP}

PINS 2-9

PIN 1

PIN 11

PAL

O_0 O_1 O_7 R   R   R

V_{HH} V_{HH} V_{OP}

BPM-380

TABLE 2. PRODUCT TERM ADDRESSING

Product Term Line Number								Product Term Select Address Pin		
								4	3	2
0	8	16	24	32	40	48	56	L	L	L
1	9	17	25	33	41	49	57	L	L	H
2	10	18	26	34	42	50	58	L	H	L
3	11	19	27	35	43	51	59	L	H	H
4	12	20	28	36	44	52	60	H	L	L
5	13	21	29	37	45	53	61	H	L	H
6	14	22	30	38	46	54	62	H	H	L
7	15	23	31	39	47	55	63	H	H	H
Pin 19	Pin 18	Pin 17	Pin 16	Pin 15	Pin 14	Pin 13	Pin 12			
Programming Access and Verify Pin										

$L = V_{ILP}$
$H = V_{IHP}$

PRELOAD OF REGISTERED OUTPUTS

AMD PAL registered outputs are designed with extra circuitry to allow loading each register asynchronously to either a HIGH or LOW state. This feature simplifies testing since any initial state for the registers can be set to optimize test sequencing.

The pin levels and timing necessary to perform the PRELOAD function are detailed below:

Level forced on registered output pin during PRELOAD cycle	Output state at the output pin after cycle
V_{HH}	HIGH
0V to V_{CCH} or OPEN	LOW

BPM-381

ORDERING INFORMATION

AmPAL   XXXXX   X   X   X   X

Basic Device
16R8
16R6
16R4
16L8
16H8
16LD8
16HD8

Speed Selection
= Standard
A = High Speed

Screening
C = 883C Basic Flow
B = 883B (Burn-in)

Temperature Range
C = Commercial
M = Military

Package
P = Molded DIP (Commercial Only)
D = Hermetic DIP
L = Chip-Pak™*

*Chip-Paks are rated at maximum case temperature only.

PHYSICAL DIMENSIONS

P-20-1
Molded DIP
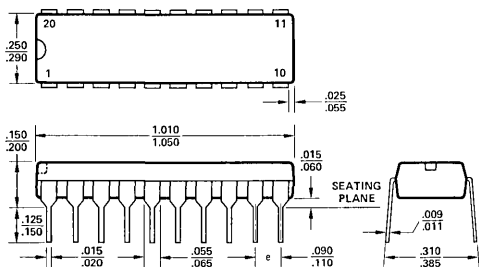
D-20-1
Hermetic DIP
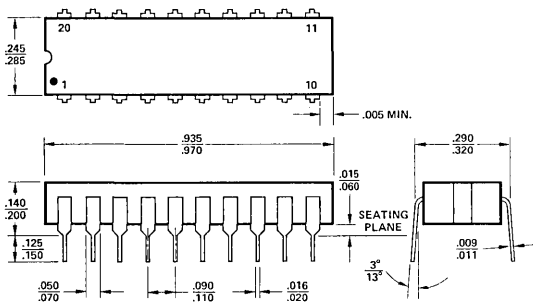
Chip-Pak
L-20-1
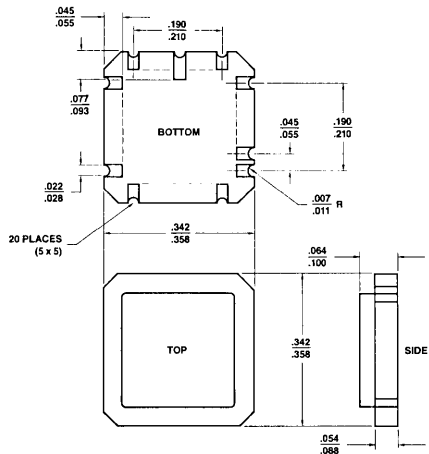
Top View
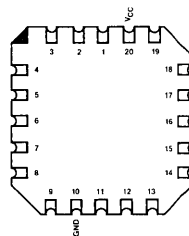
AMD Half Power PAL* Family

Half Power 20-Pin IMOX™ Programmable Array Logic Elements
Advanced Micro Devices

DISTINCTIVE CHARACTERISTICS

- **Low Power Dissipation**
  - 1/2 the power of standard PALs (I_{CC} = 60mA typ) Reduces power supply requirements and improves system reliability
- **High Performance**
  - Meets all standard power PAL AC specs (t_{pd} = 35ns max, t_S = 30ns max, t_{co} = 25ns max)
- **High Output Drive**
  - I_{OL} = 24mA Same as standard power devices
  - Drives buses directly (no special driver chips needed)
- **Plug-in Replacement for Standard PALs**
  - Meets ALL standard PAL specs at 1/2 the power
- **Excellent Programming Yield** (typ 98%)
  - Platinum Silicide fuses ensure high programming yield, fast programming and unsurpassed reliability
- **Improved Testability**
  - Preload feature permits full logical verification
- **Superior Quality**
  - Full AC and DC testing done at the factory utilizing special designed-in test features

GENERAL DESCRIPTION

These low power devices represent a breakthrough in PAL technology by meeting all the specifications of the standard power parts with only half the current requirements. They can directly replace the standard power PALs with no loss of performance, and they, like the standard devices, incorporate the PRELOAD feature essential for full logic verification during testing.

AMD Half Power PALs are high-speed Programmable Array Logic elements which utilize the familiar sum-of-products (AND-OR) logic structure which allows users to program custom logic functions to precisely fit their applications. They are a replacement for Low Power Schottky SSI/MSI logic circuits, reduce the chip count by more than 5 to 1 and greatly simplify prototyping and board layout.

Seven different device types are available, including both registered and combinatorial devices. In addition, the combinatorial devices are offered in both active HIGH (AND-OR) and active LOW (AND-OR-INVERT) versions. The low power dissipation means reduced power supply requirements and improved reliability, while the use of AMD's IMOX process permits extremely high operating speeds.

AMD HALF POWER PAL FAMILY TABLE

Part Number	Array Inputs	Logic	OE	Outputs	t_{pd} (Max) ns	t_S (Max) ns	t_{co} (Max) ns	I_{CC} (Max) mA
AmPAL16R8L	Eight Dedicated Eight Feedback	Eight 8-Wide AND-OR	Dedicated	Registered Inverting	–	30	25	90
AmPAL16R6L	Eight Dedicated Six Feedback	Six 8-Wide AND-OR	Dedicated	Registered Inverting	35	30	25	90
	Two Bidirectional	Two 7-Wide AND-OR-INVERT	Programmable	Bidirectional				
AmPAL16R4L	Eight Dedicated Four Feedback	Four 8-Wide AND-OR	Dedicated	Registered Inverting	35	30	25	90
	Four Bidirectional	Four 7-Wide AND-OR-INVERT	Programmable	Bidirectional				
AmPAL16L8L	Ten Dedicated Six Bidirectional	Eight 7-Wide AND-OR-INVERT	Programmable	Six Bidirectional Two Dedicated	35	–	–	80
AmPAL16H8L	Ten Dedicated Six Bidirectional	Eight 7-Wide AND-OR	Programmable	Six Bidirectional Two Dedicated	35	–	–	80
AmPAL16LD8L	Ten Dedicated Six Bidirectional	Eight 8-Wide AND-OR-INVERT	–	Dedicated	35	–	–	80
AmPAL16HD8L	Ten Dedicated Six Bidirectional	Eight 8-Wide AND-OR	–	Dedicated	35	–	–	80

*PAL is a registered trademark of Monolithic Memories, Inc.
IMOX is a trademark of Advanced Micro Devices, Inc.

AMD PAL FAMILY CHARACTERISTICS

All members of the AMD PAL family have common electrical characteristics and programming procedures. All parts are produced with a fusible link at each input to the AND gate array, and connections may be selectively removed by applying appropriate voltages to the circuit.

Initially the AND gates are connected, via fuses, to both the true and complement of each input. By selective programming of fuses the AND gates may be "connected" to only the true input (by blowing the complement fuse), to only the complement input (by blowing the true fuse), or to neither type of input (by blowing both fuses) establishing a logical "don't care." When both the true and complement fuses are left intact a logical false results on the output of the AND gate, while all fuses blown results in a logical true state. The outputs of the AND gates are connected to fixed OR gates. The only limitations imposed are the number of inputs to the AND gates (up to 16) and the number of AND gates per OR (up to 8).

All parts are fabricated with AMD's fast programming, highly reliable Platinum-Silicide Fuse technology. Utilizing an easily implemented programming algorithm, these products can be rapidly programmed to any customized pattern. Extra test words are pre-programmed during manufacturing to insure extremely high field programming yields (>98%), and provide extra test paths to achieve excellent parametric correlation.

POWER-UP RESET

The registered devices in the AMD PAL family have been designed to reset during system power-up. Following power-up, all registers will be initialized to zero, setting all the outputs to a logic 1. This feature provides extra flexibility to the designer and is especially valuable in simplifying state machine initialization.

PRELOAD

AMD Low Power PALs are designed with unique PRELOAD circuitry that provides an easy method of testing registered devices for logical functionality. PRELOAD allows any arbitrary state value to be loaded into the PAL's output registers.

A typical functional test sequence would be to verify all possible state transitions for the device being tested. This requires the ability to set the state registers into an arbitrary "present state" value and to set the device inputs to any arbitrary "present input" value. Once this is done, the state machine is clocked into a new state, or "next state." The next state is then checked to validate the transition from the present state. In this way any state transition can be checked.

Without PRELOAD, it is difficult and in some cases impossible to load an arbitrary present state value. This can lead to logic verification sequences that are either incomplete or excessively long. Long test sequences result when the feedback from the state register "interferes" with the inputs, forcing the machine to go through many transitions before it can reach an arbitrary state value. Therefore the test sequence will be mostly state initialization and not actual testing. The test sequence becomes excessively long when a state must be reentered many times to test a wide variety of input combinations.

In addition, complete logic verification may become impossible when states that need to be tested can not be entered with normal state transitions. For example, even though necessary, the state entered when machine powers up can not be tested, because it can not be entered from the main sequence. Similarly, "forbidden" or don't care states that are not normaly entered need to be tested to ensure that they return to the main sequence.

PRELOAD eliminates these problems by providing the capability to go directly to any desired arbitrary state. Thus test sequences may be greatly shortened, and all possible states can be tested, greatly reducing test time and development costs, and guaranteeing proper in-system operation.

INPUT CIRCUITRY
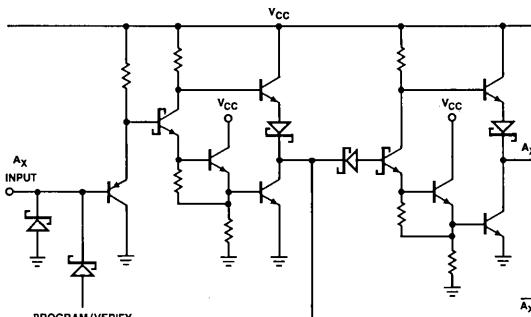
04114A-1

OUTPUT CIRCUITRY
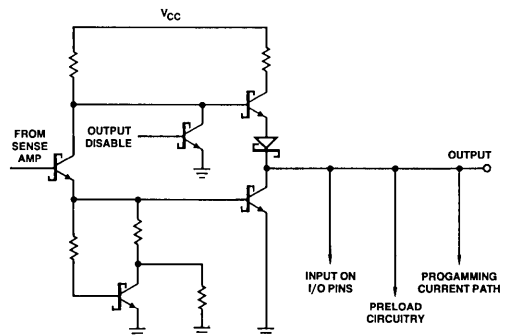
04114A-2

AmPAL16R8L
BLOCK DIAGRAM
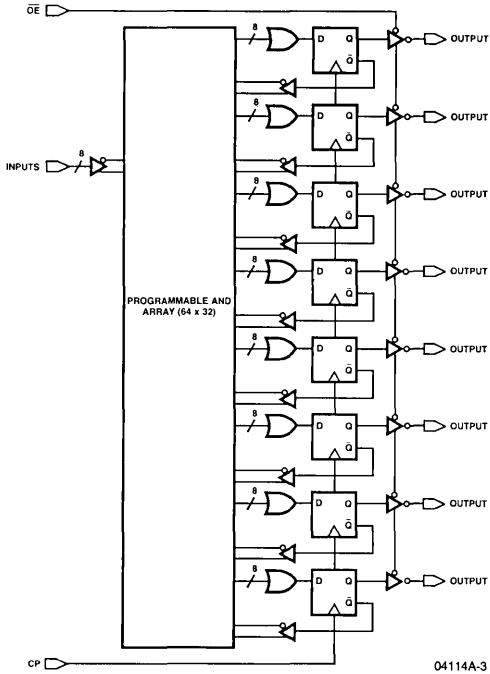
04114A-3

AmPAL16R6L
BLOCK DIAGRAM
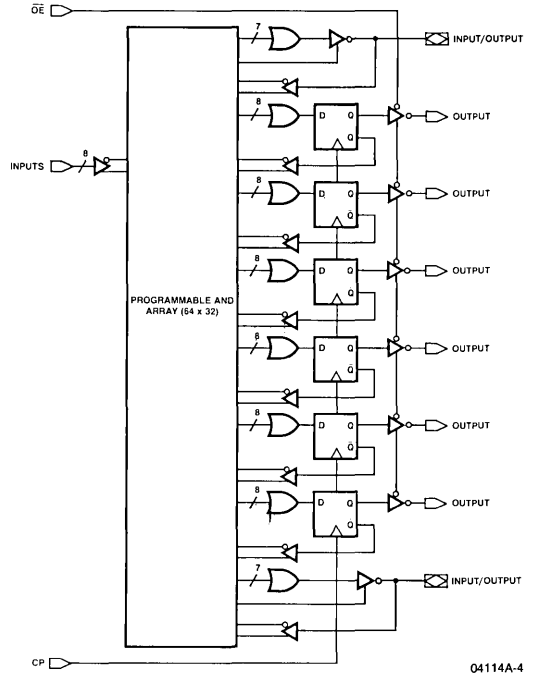
04114A-4

AmPAL16R4L
BLOCK DIAGRAM
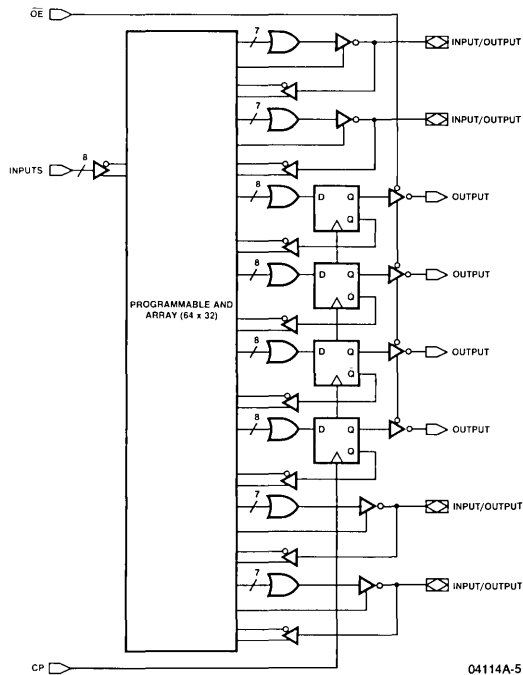
04114A-5

AmPAL16L8L
BLOCK DIAGRAM
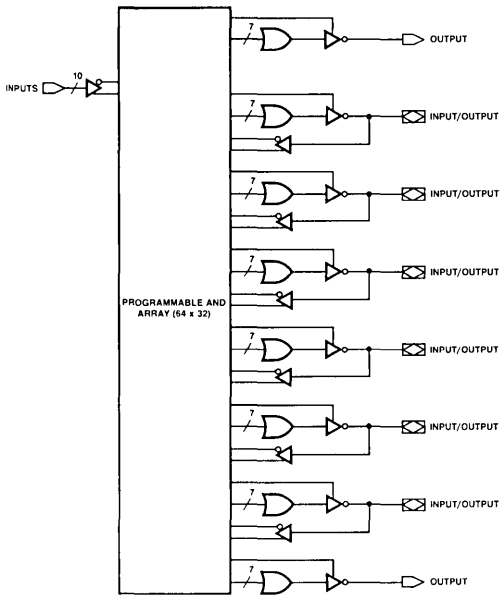
04114A-6

AmPAL16H8L
BLOCK DIAGRAM
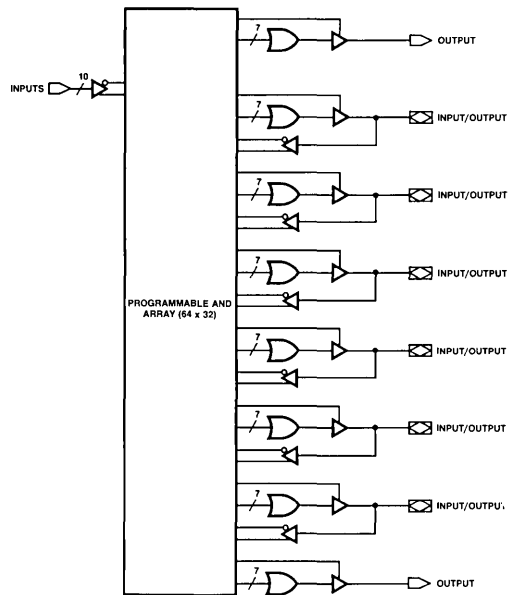
04114A-7

AmPAL16LD8L
BLOCK DIAGRAM
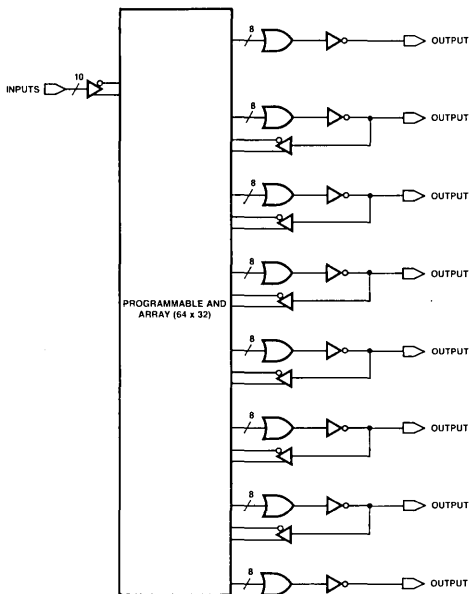
04114A-8

AmPAL16HD8L
BLOCK DIAGRAM
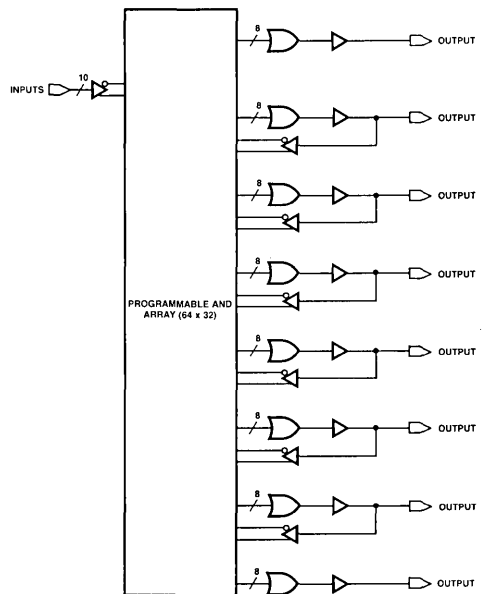
04114A-9

MAXIMUM RATINGS (Above which the useful life may be impaired)

Storage Temperature	-65 to $+150°C$
Supply Voltage to Ground Potential (Pin 20 to Pin 10) Continuous	-0.5 to $+7V$
DC Voltage Applied to Outputs (Except During Programming)	$-0.5V$ to $+V_{CC}$ max
DC Voltage Applied to Outputs During Programming	21V
Output Current Into Outputs During Programming (Max Duration of 1 sec)	200mA
DC Input Voltage	-0.5 to $+5.5V$
DC Input Current	-30 to $+5mA$

OPERATING RANGE

Parameters	Description	Commercial		Military		Units
		Min	Max	Min	Max	
V_{CC}	Supply Voltage	4.75	5.25	4.50	5.50	V
T_A	Operating Free Air Temperature	0	75	-55		°C
T_C	Operating Case Temperature				125	°C

ELECTRICAL CHARACTERISTICS OVER OPERATING RANGE (Unless Otherwise Noted)

Parameters	Description	Test Conditions			Min	Typ (Note 1)	Max	Units
V_{OH}	Output HIGH Voltage	V_{CC} = MIN, V_{IN} = V_{IH} or V_{IL}	I_{OH} = -3.2mA	COM'L	2.4	3.5		Volts
			I_{OH} = -2mA	MIL				
V_{OL}	Output LOW Voltage	V_{CC} = MIN, V_{IN} = V_{IH} or V_{IL}	I_{OL} = 24mA	COM'L			0.50	Volts
			I_{OL} = 12mA	MIL				
V_{IH} (Note 2)	Input HIGH Level	Guaranteed input logical HIGH voltage for all inputs			2.0			Volts
V_{IL} (Note 2)	Input LOW Level	Guaranteed input logical LOW voltage for all inputs					0.8	Volts
I_{IL}	Input LOW Current	V_{CC} = MAX, V_{IN} = 0.40V				-20	-250	μA
I_{IH}	Input HIGH Current	V_{CC} = MAX, V_{IN} = 2.7V					25	μA
I_I	Input HIGH Current	V_{CC} = MAX, V_{IN} = 5.5V					1.0	mA
I_{SC}	Output Short Circuit Current	V_{CC} = MAX, V_{OUT} = 0.5V (Note 3)			-30	-60	-90	mA
I_{CC}	Power Supply Current	All inputs = GND, V_{CC} = MAX	16L8L, 16H8L, 16HD8L, 16LD8L			55	80	mA
			16R8L, 16R6L, 16R4L			60	90	
V_I	Input Clamp Voltage	V_{CC} = MIN, I_{IN} = -18mA				-0.9	-1.2	Volts
I_{OZH}	Output Leakage Current (Note 4)	V_{CC} = MAX, V_{IL} = 0.8V V_{IH} = 2.0V	V_O = 2.7V				100	μA
I_{OZL}			V_O = 0.4V				-100	
C_{IN}	Input Capacitance	V_{IN} = 2.0V @f = 1MHz (Note 5)				6		pF
C_{OUT}	Output Capacitance	V_{OUT} = 2.0V @f = 1MHz (Note 5)				9		

Notes: 1. Typical limits are at V_{CC} = 5.0V and T_A = 25°C.
2. These are absolute values with respect to device ground and all overshoots due to system or tester noise are included.
3. Not more than one output should be tested at a time. Duration of the short circuit should not be more than one second. V_{OUT} = 0.5V has been chosen to avoid test problems caused by tester ground degradation.
4. I/O pin leakage is the worst case of I_{OZX} or I_{IX} (where x = H or L).
5. These parameters are not 100% tested, but are periodically sampled.

SWITCHING CHARACTERISTICS OVER OPERATING RANGE (Unless otherwise noted)

Parameters	Description	Test Conditions	Typ (Note 1)	COM'L Min	COM'L Max	MIL Min	MIL Max	Units
t_{PD}	Input or Feedback to Non-Registered Output 16L8L, 16R6L, 16R4L, 16LD8L, 16H8L, 16HD8L	COM'L $R_1 = 200$ $R_2 = 390$	−20		35		40	ns
t_{EA}	Input to Output Enable 16L8L, 16R6L, 16R4L, 16H8L		−20		35		40	ns
t_{ER}	Input to Output Disable 16L8L, 16R6L, 16R4L, 16H8L		−20		35		40	ns
t_{PZX}	Pin 11 to Output Enable 16R8L, 16R6L, 16R4L		−15		25		25	ns
t_{PXZ}	Pin 11 to Output Disable 16R8L, 16R6L, 16R4L		−15		25		25	ns
t_{CO}	Clock to Output 16R8L, 16R6L, 16R4L		−10		25		25	ns
t_S	Input or Feedback Setup Time 16R8L, 16R6L, 16R4L	MIL $R_1 = 390$ $R_2 = 750$	−20	30		35		ns
t_H	Hold Time 16R8L, 16R6L, 16R4L		−10	0		0		ns
t_P	Clock Period ($t_S + t_{CO}$)			55		60		ns
t_W	Clock Width			25		25		ns
f_{MAX}	Maximum Frequency			18		16.5		MHz

Notes: 1. Typical limits are at $V_{CC} = 5.0V$ and $T_A = 25°C$.
     2. t_{PD} is tested with switch S_1 closed and $C_L = 50pF$.
     3. For three-state outputs, output enable times are tested with $C_L = 50pF$ to the 1.5V level; S_1 is open for high impedance to HIGH tests and closed for high impedance to LOW tests. Output disable times are tested with $C_L = 5pF$. HIGH to high impedance tests are made to an output voltage of $V_{OH} − 0.5V$ with S_1 open; low-to-high impedance tests are made to the $V_{OL} + 0.5V$ level with S_1 closed.

SWITCHING WAVEFORMS

04114A-10

AC TEST LOAD
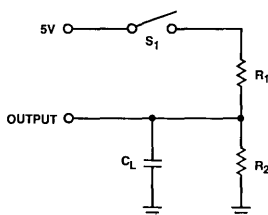
04114A-11

KEY TO TIMING DIAGRAM

WAVEFORM	INPUTS	OUTPUTS
———	MUST BE STEADY	WILL BE STEADY
▨▨▨	DON'T CARE; ANY CHANGE PERMITTED	CHANGING; STATE UNKNOWN
⫸⫸⫷	DOES NOT APPLY	CENTER LINE IS HIGH IMPEDANCE "OFF" STATE

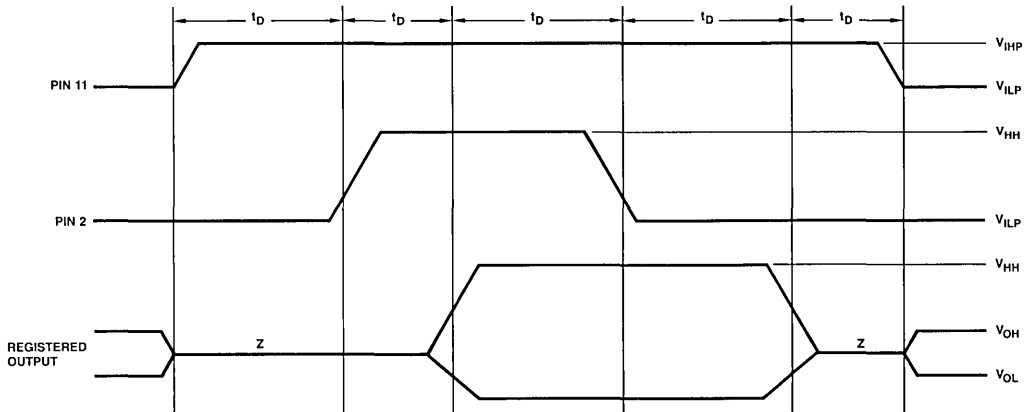## PRELOAD OF REGISTERED OUTPUTS

AMD PAL registered outputs are designed with extra circuitry to allow loading each register asynchronously to either a HIGH or LOW state. This feature simplifies testing since any initial state for the registers can be set to optimize test sequencing.
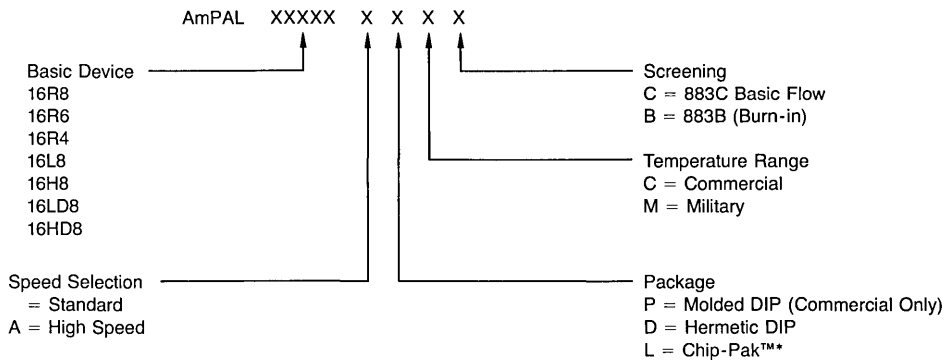
The pin levels and timing necessary to perform the PRELOAD function are detailed below:
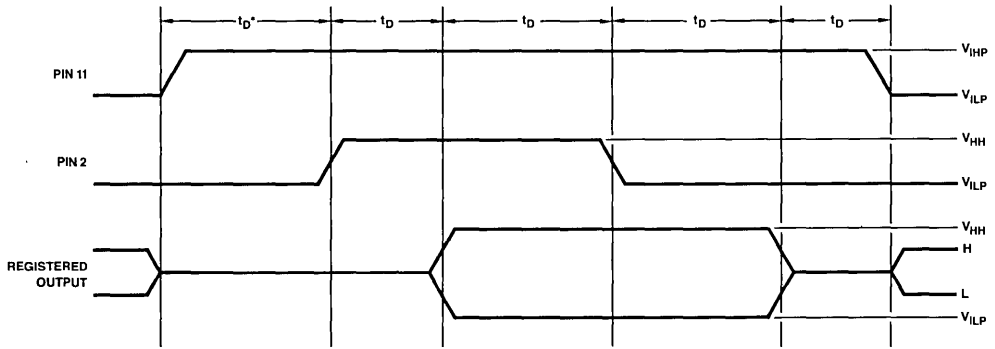
04114A-12

$*t_D$ = 200ns typically. See Programming Parameters.

Level forced on registered output pin during PRELOAD cycle	Output state at the output pin after cycle
V_{HH}	HIGH
0V to V_{CCH} or OPEN	LOW

POWER-UP RESET

The registered devices in the AMD PAL family have been designed to reset during system power-up. Due to the asynchronous operation of the power-up reset and the wide range of ways V_{CC} can rise to its steady state, three conditions are required to insure a valid power-up reset. These conditions are:

1. The V_{CC} rise must be monotonic.
2. Following reset, the clock input must not be driven from low to high until all applicable input and feedback setup times are met.
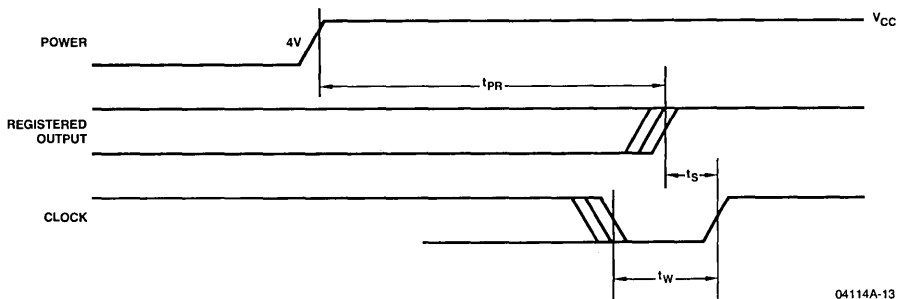
04114A-13

Parameters	Description	Min	Typ	Max	Units
t_{pr}	Power-Up Reset Time		600	1000	ns
t_s	Input or Feedback Setup Time	See Switching Characteristics			
t_w	Clock Width				

LOGIC DIAGRAM AmPAL16R8L

INPUTS (0–31)

PRODUCT TERMS (0–63)
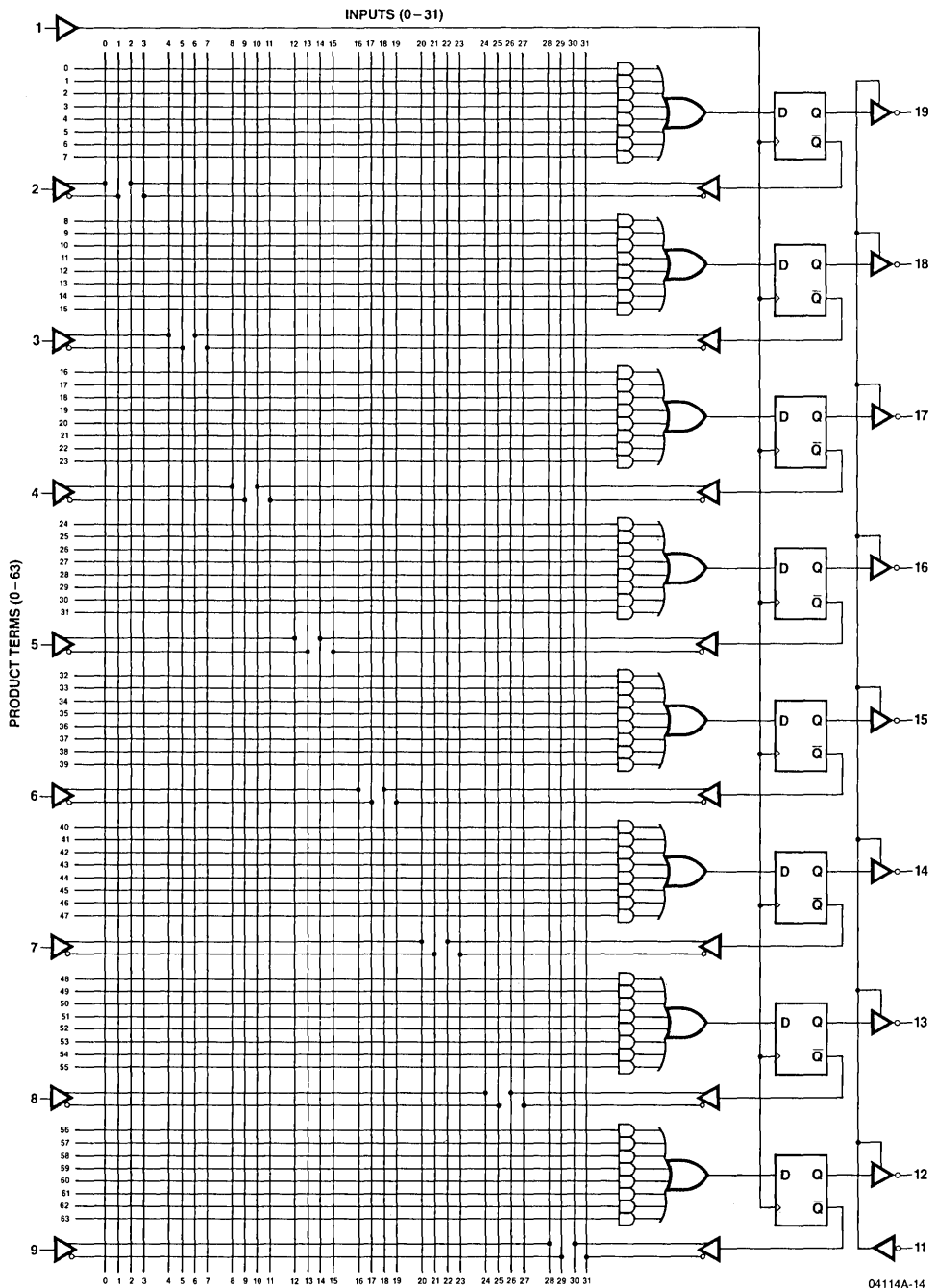
04114A-14

16 Fuse Array Inputs
- 8 dedicated
- 8 internal registered feedback

8, 8-Wide AND-OR Structures
- Registered, inverting outputs
- Common dedicated output enable

2-24

LOGIC DIAGRAM AmPAL16R6L

INPUTS (0 – 31)

PRODUCT TERMS (0 – 63)

04114A-15
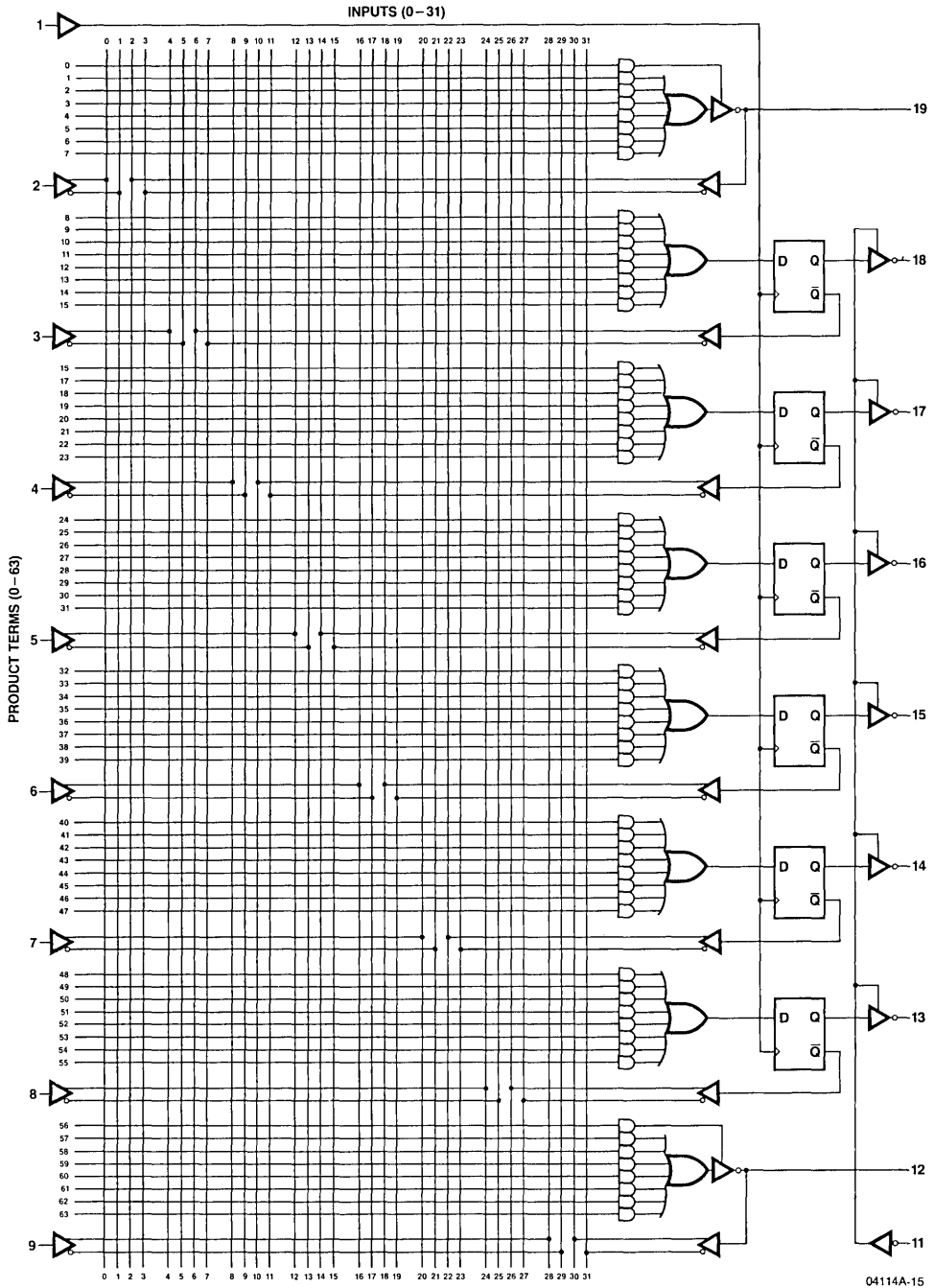
16 Array Inputs
- 8 dedicated
- 6 registered feedback
- 2 bidirectional I/O

6, 8-Wide AND-OR Structures
- Registered, inverting outputs with common dedicated output enable

2, 7-Wide AND-OR-INVERT Structures
- Combinatorial outputs with programmable output enables

LOGIC DIAGRAM AmPAL16R4L

04114A-16

16 Array Inputs
- 8 dedicated
- 4 registered feedback
- 4 bidirectional I/O

4, 8-Wide AND-OR Structures
- Registered, inverting outputs with common, dedicated output enable

4, 7-Wide AND-OR-INVERT Structures
- Combinatorial outputs with programmable output enables

LOGIC DIAGRAM AmPAL16L8L

INPUTS (0 – 31)

PRODUCT TERMS (0 – 63)

04114A-17

16 Array Inputs
- 10 dedicated
- 6 bidirectional I/O

8, 7-Wide AND-OR-INVERT Structures
- Combinatorial outputs
- Programmable output enable for each output

LOGIC DIAGRAM AmPAL16LD8L

INPUTS (0–31)

PRODUCT TERMS (0–63)
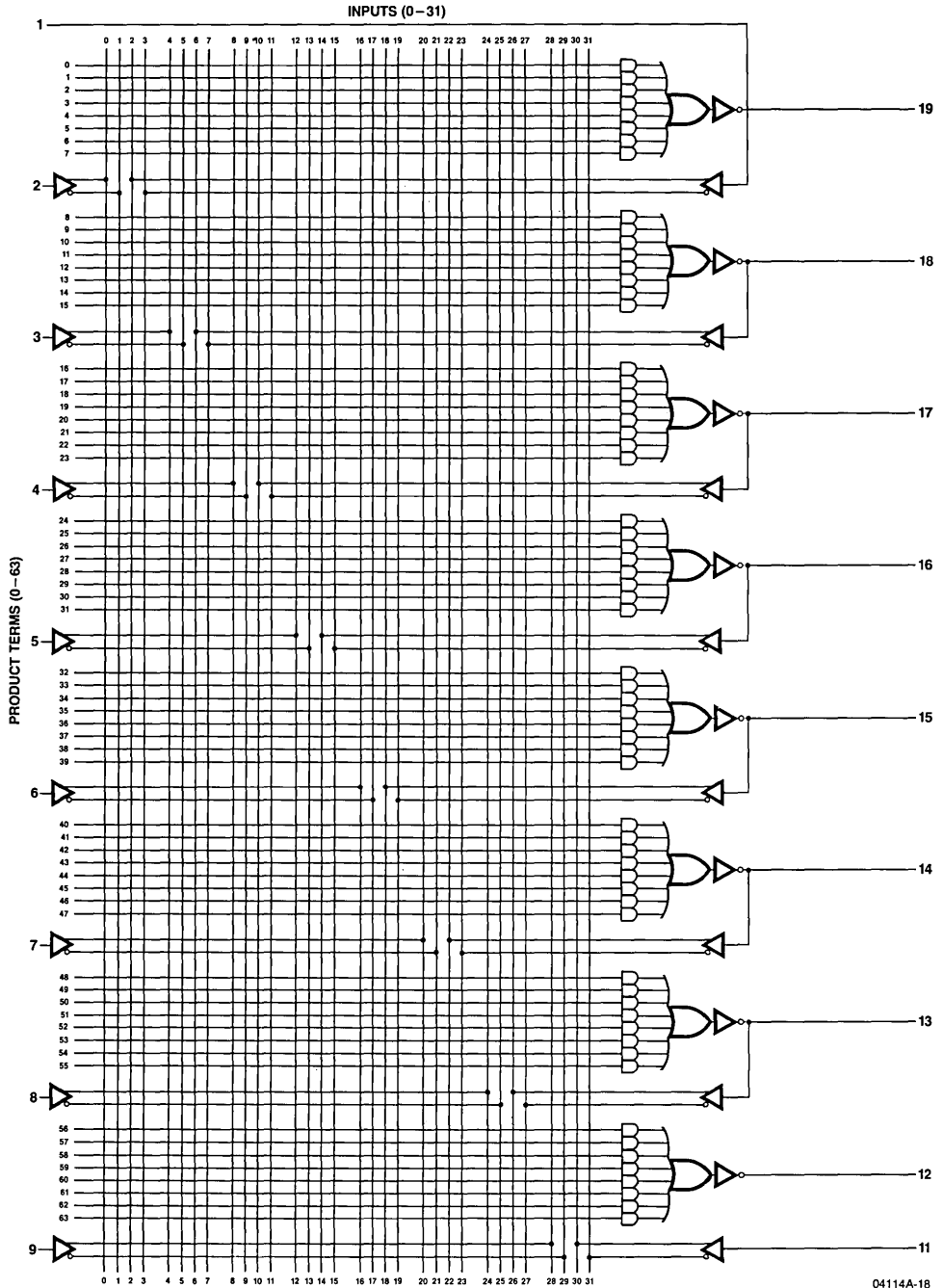
04114A-18

16 Array Inputs   **8, 8-Wide AND-OR-INVERT Structures**
- 10 dedicated   — Dedicated combinatorial outputs
- 6 feedback

2-28

LOGIC DIAGRAM AmPAL16H8L

INPUTS (0–31)

PRODUCT TERMS (0–63)
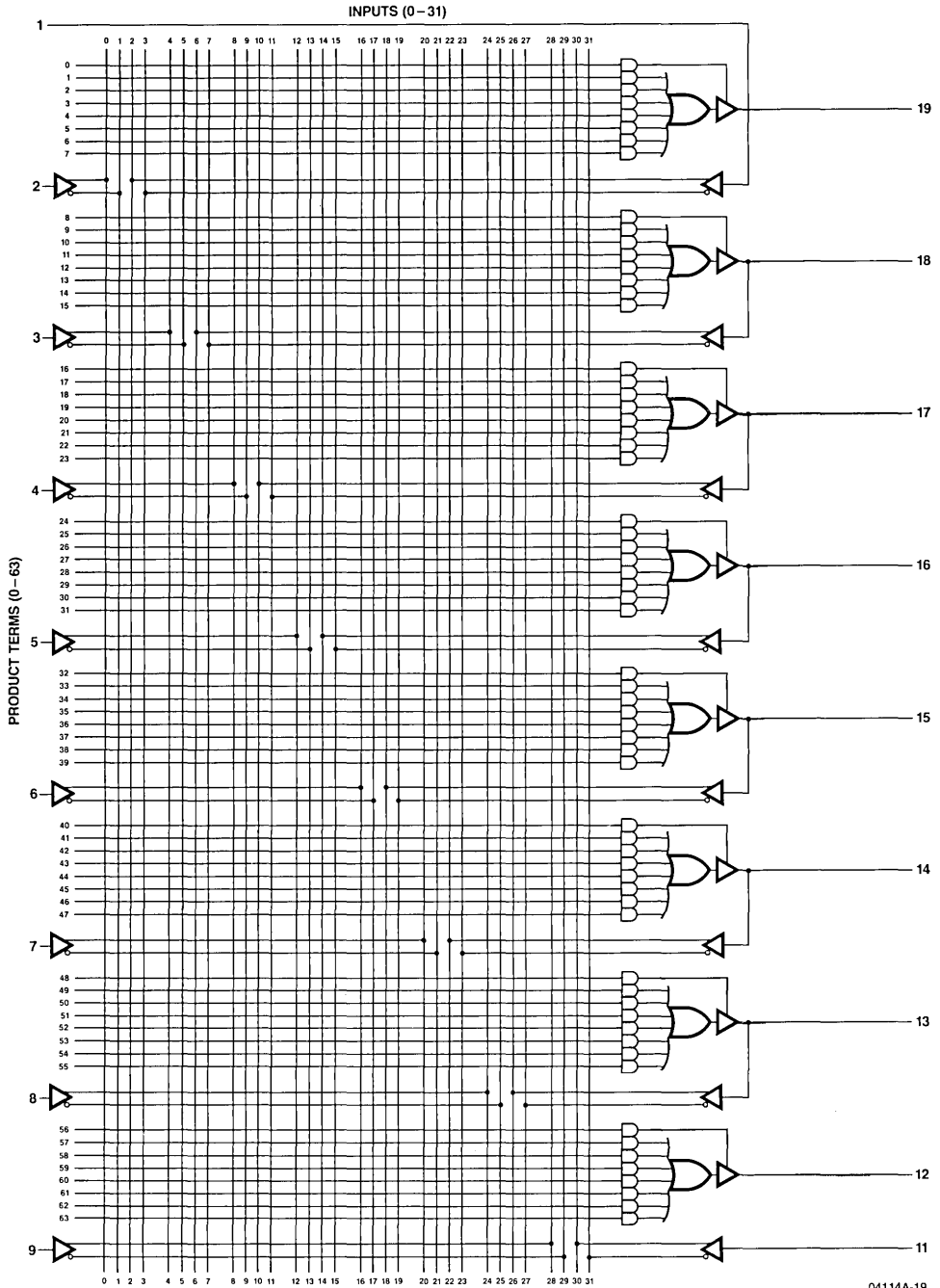
04114A-19

16 Array Inputs
- 10 dedicated
- 6 bidirectional I/O

8, 7-Wide AND-OR Structures
- Combinatorial outputs
- Programmable output enable for each output

LOGIC DIAGRAM AmPAL16HD8L

INPUTS (0–31)
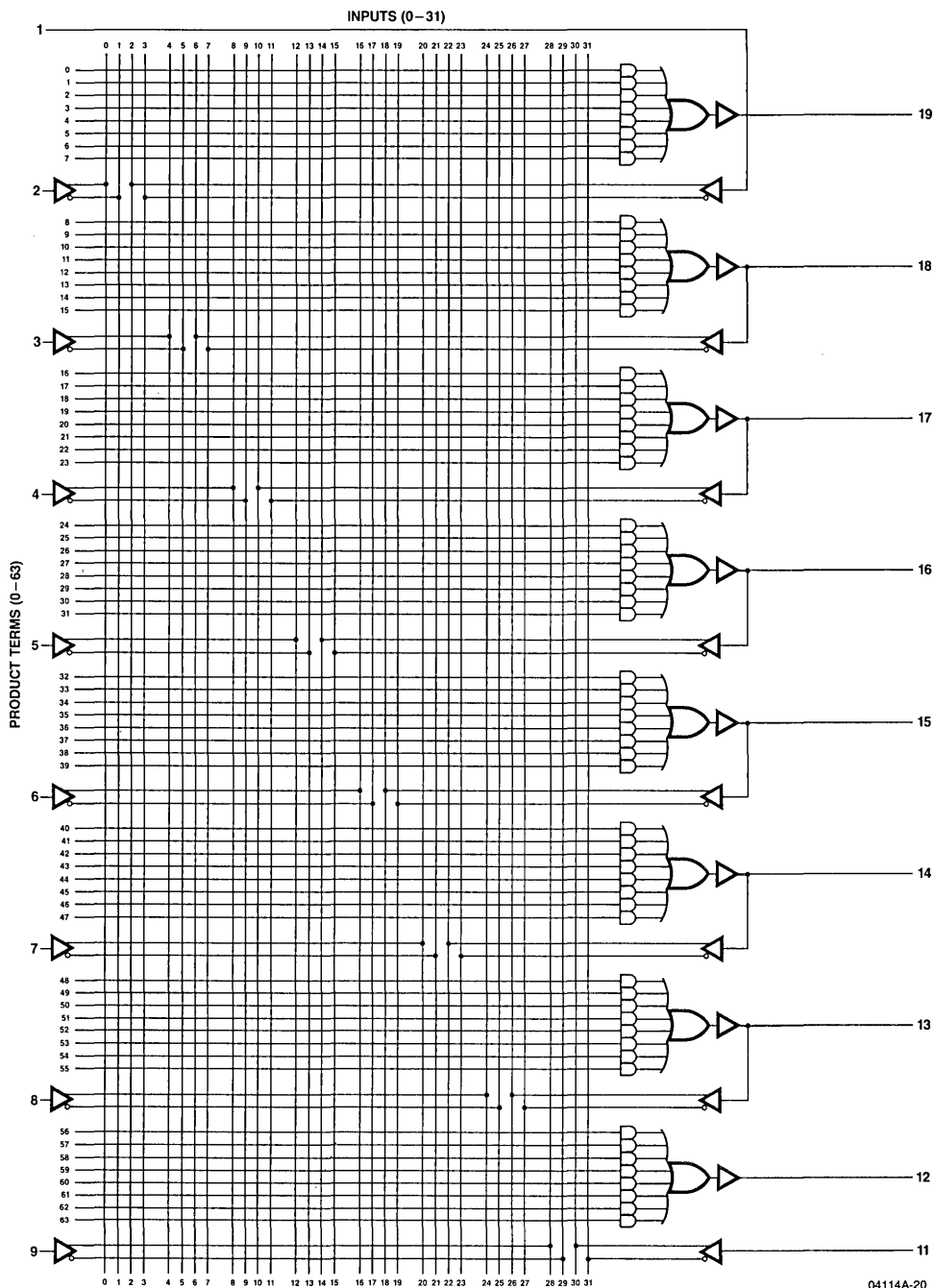
16 Array Inputs
- 10 dedicated
- 6 feedback

8, 8-Wide AND-OR Structures
- Dedicated combinatorial outputs

04114A-20

PROGRAMMING

Each AMD PAL fuse is programmed with a simple sequence of voltages applied to two control pins (1 and 11) and a programming voltage pulse applied to the selected output, one output at a time. Addressing of the 2048 element fuse array is accomplished with normal TTL levels on eight input pins (five select the input line number and three select the product term number). V_{CC} is maintained at a normal level throughout the programming and verify cycle — no extra high levels are required.

The necessary sequence of levels for programming any fuse is shown in the Programming Waveforms. The address of each fuse in terms of Input Line Number and Product Term Line Number is defined by the Fuse Address Tables 1 and 2. Current, voltage and timing requirements for each pin are specified in the Programming Parameter Table below.

The 16L8L, 16R8L, 16R6L, 16R4L, 16H8L, 16LD8L and 16HD8L use identical programming conditions and sequences.

After all programming has been completed, the entire array should be reverified at V_{CCL} and again at V_{CCH}. Reverification can be accomplished by reading all eight outputs in parallel rather than one at a time. The array fuse verification cycle checks that the correct array fuses have been blown and can be sensed by the outputs.

AMD PALs have been designed with many internal test features that are used to assure high programming yield and correct logical operation for a correctly programmed part.

An additional fuse is provided on each AMD PAL circuit to prevent unauthorized copying of AMD PAL fuse patterns when design security is desired. Blowing the security fuse blocks entry to the fuse pattern verify mode.

To blow the security fuse:
1. Power up part to V_{CCP}
2. Raise Pin 5 to V_{HH}.
3. Pulse Pin 11 from ground to V_{OP} for a 50μsec duration.
4. Perform a normal end-of-programming verify cycle at V_{CCL} and V_{CCH}. All fuse locations should be sensed as blown if the security fuse has been successfully blown.

Note that parts with the security fuse blown may not be returned as programming rejects.

AMD PALs normally have high programming yields (>98%). Programming yield losses are frequently due to poor socket contact, equipment out of calibration or improperly used.

PROGRAMMING PARAMETERS ($T_A = 25°C$)

Parameters	Description		Min	Typ	Max	Units
V_{HH}	Control Pin Extra High Level	Pin 1 @ 10−40mA	10	11	12	Volts
		Pin 11 @ 10−40mA	10	11	12	
V_{OP}	Program Voltage Pins 12−19 @ 15−200mA		18	20	22	Volts
V_{IHP}	Input High Level During Programming and Verify		2.4	5	5.5	Volts
V_{ILP}	Input Low Level During Programming and Verify		0.0	0.3	0.5	Volts
V_{CCP}	V_{CC} During Programming @ I_{CC} = 50−200mA		5	5.2	5.5	Volts
V_{CCL}	V_{CC} During First Pass Verification @ I_{CC} = 50−200mA		4.1	4.3	4.5	Volts
V_{CCH}	V_{CC} During Second Pass Verification @ I_{CC} = 50−200mA		5.4	5.7	6.0	Volts
V_{Blown}	Successful Blown Fuse Sense Level @ Output	16L8L, 16R8L, 16R6L, 16R4L, 16LD8L		0.3	0.5	Volts
		16H8L, 16HD8L	2.4	3		
dV_{OP}/dt	Rate of Output Voltage Change		20		250	V/μsec
dV_{11}/dt	Rate of Fusing Enable Voltage Change (Pin 11 Rising Edge)		100		1000	V/μsec
t_P	Fusing Time First Attempt		40	50	100	μsec
	Subsequent Attempts		4	5	10	msec
t_D	Delays Between Various Level Changes		100	200	1000	ns
t_V	Period During which Output is Sensed for V_{Blown} Level				500	ns
V_{ONP}	Pull-Up Voltage On Outputs Not Being Programmed		$V_{CCP} - 0.3$	V_{CCP}	$V_{CCP} + 0.3$	Volts
R	Pull-Up Resistor On Outputs Not Being Programmed		1.9	2	2.1	KΩ

AMD PAL PROGRAMMING EQUIPMENT INFORMATION

Source and Location	Data I/O 10525 Willows Rd., N.E. Redmond, WA 98052	Kontron Electronics, Inc. 630 Price Ave. Redwood City, CA 94063	Stag Microsystems 528-5 Weddel Dr. Sunnyvale, CA 94086	Structured Design, Inc. 1700 Wyatt Dr. #3 Santa Clara, CA 95054	Digilec, Inc. 7335 E. Acoma Dr. Dept-103 Scottsdale, AZ 85260
Programmer Model(s)	Model-100, 29, 19 or 17	Model-MPP-80S or EPP80	Model-PPX	SD1000	TBA
			Model ZL-30		
AMD PAL Personality Module	Logicpak 950-1942-001	MOD-33	PPM2200	On Board	−
			On Board ZL-30		
Socket Adapter	715-1947-003	SA37	Am202S	On Board	−
			On Board ZL-30		

The machines noted above have been qualified by AMD to insure high programming yields. Check with the factory to determine the current status of vendors noted TBA or other available models.

PROGRAMMING WAVEFORMS

04114A-21

TABLE 1. INPUT ADDRESSING

Input Line Number	Input Line Number Address Pin States				
	9	8	7	6	5
0	L	L	L	L	L
1	L	L	L	L	H
2	L	L	L	H	L
3	L	L	L	H	H
4	L	L	H	L	L
5	L	L	H	L	H
6	L	L	H	H	L
7	L	L	H	H	H
8	L	H	L	L	L
9	L	H	L	L	H
10	L	H	L	H	L
11	L	H	L	H	H
12	L	H	H	L	L
13	L	H	H	L	H
14	L	H	H	H	L
15	L	H	H	H	H
16	H	L	L	L	L
17	H	L	L	L	H
18	H	L	L	H	L
19	H	L	L	H	H
20	H	L	H	L	L
21	H	L	H	L	H
22	H	L	H	H	L
23	H	L	H	H	H
24	H	H	L	L	L
25	H	H	L	L	H
26	H	H	L	H	L
27	H	H	L	H	H
28	H	H	H	L	L
29	H	H	H	L	H
30	H	H	H	H	L
31	H	H	H	H	H

$L = V_{ILP}$
$H = V_{IHP}$

SIMPLIFIED PROGRAMMING DIAGRAM

04114A-22

TABLE 2. PRODUCT TERM ADDRESSING

Product Term Line Number								Product Term Select Address Pin		
								4	3	2
0	8	16	24	32	40	48	56	L	L	L
1	9	17	25	33	41	49	57	L	L	H
2	10	18	26	34	42	50	58	L	H	L
3	11	19	27	35	43	51	59	L	H	H
4	12	20	28	36	44	52	60	H	L	L
5	13	21	29	37	45	53	61	H	L	H
6	14	22	30	38	46	54	62	H	H	L
7	15	23	31	39	47	55	63	H	H	H
Pin 19	Pin 18	Pin 17	Pin 16	Pin 15	Pin 14	Pin 13	Pin 12			
Programming Access and Verify Pin										

$L = V_{ILP}$
$H = V_{IHP}$

ORDERING INFORMATION

AmPAL   16HD8   L   D   M   B

Basic Device
16R8
16R6
16R4
16L8
16H8
16LD8
16HD8

Screening
Blank = Standard Process Flow
B = Burned-In

Temperature Range
C = Commercial
M = Military

Speed/Power Selection
L = Standard Speed/Low Power

Package
P = Molded DIP (Commercial Only)
D = Hermetic DIP
L = Chip-Pak™*

*Chip-Paks are rated at maximum case temperature only.

PHYSICAL DIMENSIONS

P-20-1
Molded DIP

D-20-1
Hermetic DIP

Chip-Pak
L-20-1

Top View

Pads are located on bottom of package, and are shown for reference only.

Chip-Pak is a trademark of Advanced Micro Devices, Inc.

AmPAL* 22V10

24-Pin IMOX™ Programmable Array Logic
Advanced Micro Devices

DISTINCTIVE CHARACTERISTICS

- Second generation PAL architecture
- Increased logic power — up to 22 inputs and 10 outputs
- Increased product terms — average 12 per output
- Variable product term distribution improves ease of use
- Each output USER PROGRAMMABLE for registered or combinatorial operation
- Individually USER PROGRAMMABLE output polarity
- Extra terms provide logical synchronous PRESET and asynchronous RESET capability
- Comes in standard and high speed versions — 15ns typical propagation delay
- TTL level PRELOAD for improved testability
- Packaged in SLIMDIP (300 mil) 24 pin package
- Platinum Silicide fuses ensure high programming yield, fast programming and unsurpassed reliability
- Full AC and DC testing done at the factory utilizing special designed-in test features

GENERAL DESCRIPTION

The AmPAL22V10 is a second generation Programmable Array Logic device. It utilizes the familiar sum-of-products (AND-OR) logic structure, allowing users to program custom logic functions. The AmPAL22V10 is an extension of the PAL concept. First generation devices were largely limited to TTL replacement applications. The AmPAL22V10 permits the development of custom LSI functions of 300 to 500 equivalent gate complexity.

The AmPAL22V10 contains up to 22 inputs and 10 outputs. It incorporates the unique capability of defining and programming the architecture of each output on an individual basis. Each output is user programmable for either registered or combinatorial operation. This allows the designer to optimize the device design, by having only as many registers as needed. In addition each output has user programmable output polarity, further simplifying design and contributing to precise applications requirements.

Increased logic power has been built into the AmPAL22V10 by increasing the number of product terms from 8 per output to an average of 12 per output. Further innovation can be seen in the introduction of variable product term distribution. This technique allocates from 8 to 16 logical product terms to each output (please refer to block diagram for distribution details.) This variable allocation of terms allows far more complex functions to be implemented than in previous devices.

System operation has been enhanced by the addition of a synchronous PRESET and an asynchronous RESET product term. These terms are common to all outputs.

The AmPAL22V10 also incorporates power-up reset and the unique capability to PRELOAD the output registers to any desired state during testing. PRELOAD is essential to permit full logical verification during test.
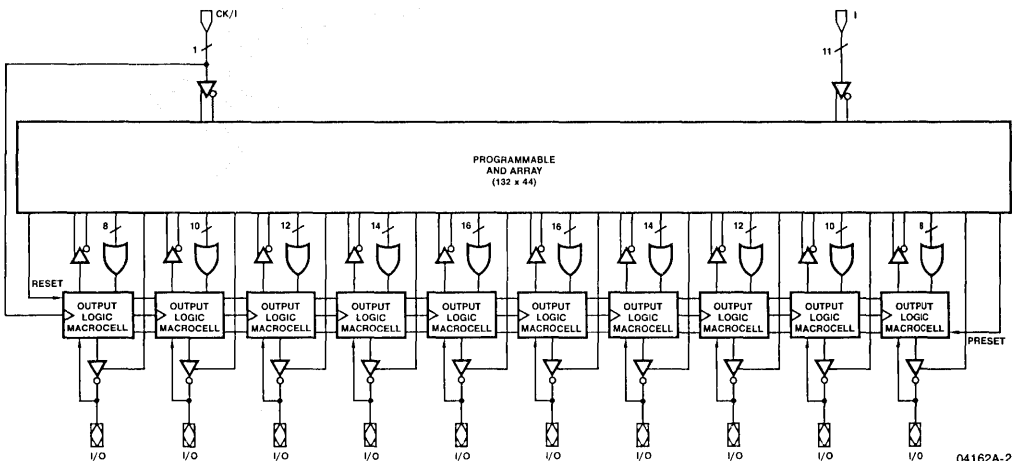
CONNECTION DIAGRAM — Top View

```
CK/I  [ 1        24 ]  Vcc
   I  [ 2        23 ]  I/O
   I  [ 3        22 ]  I/O
   I  [ 4        21 ]  I/O
   I  [ 5        20 ]  I/O
   I  [ 6  AmPAL22V10  19 ]  I/O
   I  [ 7        18 ]  I/O
   I  [ 8        17 ]  I/O
   I  [ 9        16 ]  I/O
   I  [ 10       15 ]  I/O
   I  [ 11       14 ]  I/O
 GND  [ 12       13 ]  I
```

04162A-1

BLOCK DIAGRAM

04162A-2

FUNCTIONAL DESCRIPTION

The AmPAL22V10 is a second generation Programmable Array Logic device. It contains a programmable fuse array organized in the familiar sum-of-products (AND-OR) structure.

The block diagram below shows the basic architecture of the AmPAL22V10. There are up to 22 inputs and 10 outputs available. The inputs are connected to a programmable AND array which contains 120 logical product terms. Initially the AND gates are connected, via fuses, to both the true and complement of every input. By selective programming of fuses the AND gates may be "connected" to only the true input (by blowing the complement fuse), to only the complement input (by blowing the true fuse), or to neither type of input (by blowing both fuses) establishing a logical "don't care". When both the true and complement fuses are left intact a logical false results on the output of the AND gate. An AND gate with all fuses blown will assume the logical true state. The outputs of the AND gates are connected to fixed OR gates. There are an average of 12 product terms per OR gate (output) and as the block diagram shows, variable product term distribution has been implemented. This technique allocates different quantities of logical product terms to different outputs, allowing more complex logical functions to be performed than were previously possible. Up to 16 logical terms can be evaluated in one output in a single clock cycle (no feedback necessary).

A dramatic innovation in logic design is the implementation on the AmPAL22V10 of variable output architecture. This allows the user to program on an output by output basis the function of the outputs. As shown in the output logic Macrocell diagram below, each output cell contains two additional fuses (R_n and P_n). R_n controls whether the output will be registered or combinatorial. P_n controls the output polarity (active HIGH or active LOW). Depending on the states of these 2 fuses, an individual output will operate in one of four modes (see logic diagrams below): Registered active LOW; Registered active HIGH; Combinatorial active LOW; Combinatorial active HIGH. (Note that the feedback path also changes with the output mode.) This innovation gives the designer more flexibility and enables him to optimize the device for precise application requirements. It also allows for better device utilization — you only program as many registers as are needed.

To improve in-system functionality the AmPAL22V10 has additional PRESET and RESET product terms. These terms are connected to all registered outputs. When the synchronous PRESET product term is asserted (HIGH) the output registers will be loaded with a HIGH on the next LOW-to-HIGH clock transition. When the asynchronous RESET product term is asserted (HIGH) the output registers will be immediately loaded with a LOW (independent of the clock). These functions are particularly useful for applications such as system power-on and reset.

BLOCK DIAGRAM
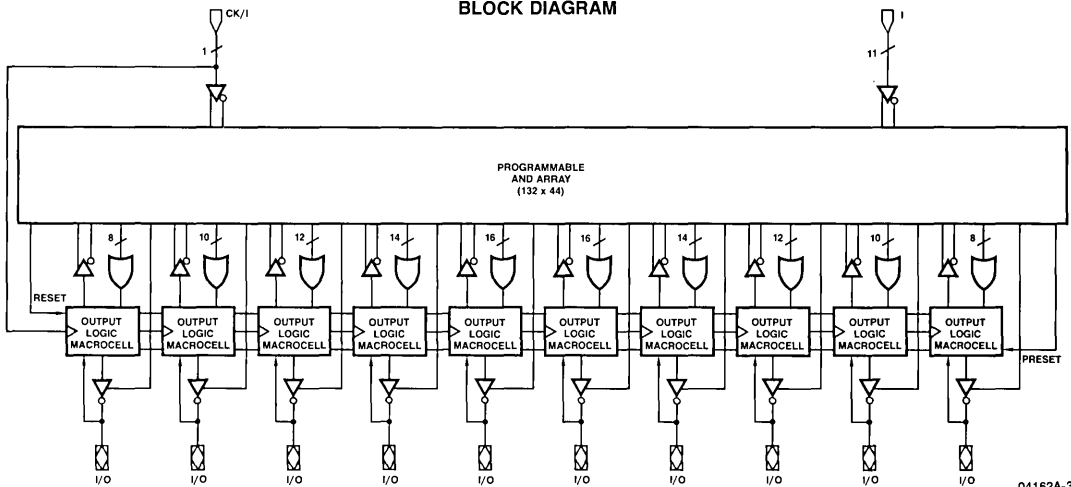
04162A-3

OUTPUT LOGIC MACROCELL DIAGRAM
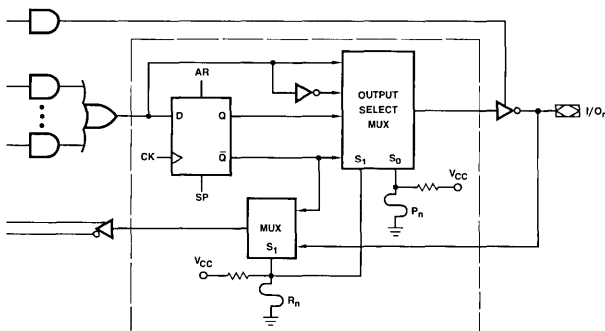
S_1	S_0	Output Configuration
0	0	Registered/Active Low
0	1	Registered/Active High
1	0	Combinatorial/Active Low
1	1	Combinatorial/Active High

0 = Unblown Fuse
1 = Blown Fuse

04162A-4

In order to simplify testing problems, the AmPAL22V10 is designed with unique PRELOAD circuitry that provides an easy method of testing registered devices for logical functionality. PRELOAD allows any arbitrary state value to be loaded into the PAL's output registers.

A typical functional test sequence would be to verify all possible state transitions for the device being tested. To verify these transitions requires the ability to set the state registers into an arbitrary "present state" value and to set the device inputs to any arbitrary "present input" value. Once this is done, the state machine is then clocked into a new state, or "next state". The next state is then checked to validate the transition from the present state. In this way any state transition can be checked.

Without PRELOAD, it is difficult and in some cases impossible to load an arbitrary present state value. This can lead to logic verification sequences that are either incomplete or excessively long. Long test sequences result when the feedback from the state register "interferes" with the inputs, forcing the state machine to go through many state transitions before it can reach an arbitrary state value. Therefore the test sequence will be mostly state initialization and not actual testing. The test sequence becomes excessively long when a state must be re-entered many times to test a wide variety of input combinations.

In addition, complete logic verification may become impossible when states that need to be tested can not be entered with normal state transitions. For example, the state which the machine powers up into cannot be tested because it cannot be entered from the main sequence. Similarly, "forbidden" or don't care states that are not normally entered need to be tested to ensure that they return to the main sequence.

PRELOAD eliminates these problems by providing the capability to go directly to any desired arbitrary state. Thus test sequences may be greatly shortened, and all possible states can be tested, greatly reducing test time and development costs, and guaranteeing proper in-system operation.

The AmPAL22V10 is manufactured using Advanced Micro Devices' IMOX™ oxide isolation process. This advanced process permits an increase in density and a decrease in internal capacitance resulting in the fastest possible programmable logic devices.

The AmPAL22V10 is fabricated with AMD's fast programming, highly reliable Platinum-Silicide Fuse technology. Utilizing an easily implemented programming algorithm, these products can be rapidly programmed to any customized pattern. Extra test words are preprogrammed during manufacturing to ensure extremely high field programming yields ($>98\%$), and provide extra test paths to achieve excellent parametric correlation.

Platinum-Silicide was selected as the fuse link material to achieve a well controlled melt rate resulting in large nonconductive gaps that ensure very stable, long term reliability. Extensive operating testing has proven that this low-field, large-gap technology offers the best reliability for fusible link programmable logic.

REGISTERED/ACTIVE LOW

$S_0 = 0$
$S_1 = 0$
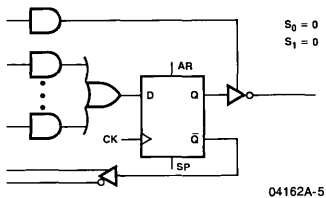
04162A-5

COMBINATORIAL/ACTIVE LOW

$S_0 = 0$
$S_1 = 1$

04162A-6

REGISTERED/ACTIVE HIGH

$S_0 = 1$
$S_1 = 0$

04162A-7

COMBINATORIAL/ACTIVE HIGH

$S_0 = 1$
$S_1 = 1$
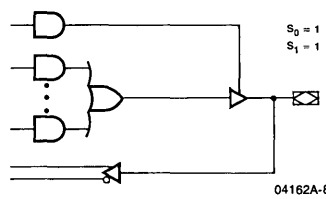
04162A-8

KEY TO LOGIC DIAGRAM

TRUE AND COMPLEMENT BUFFER

PROGRAMMABLE CONNECTIONS TO N-INPUT AND
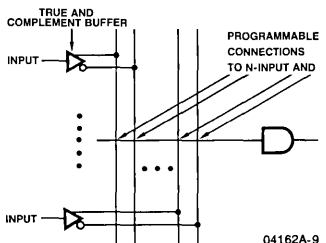
INPUT

INPUT

04162A-9

Programmable AND Array
PAL Logic Diagram Representation

INPUT

N FUSES
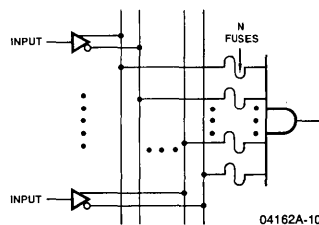
INPUT

04162A-10

Programmable AND Array
Logic Equivalent

LOGIC DIAGRAM AmPAL22V10

INPUT LINES
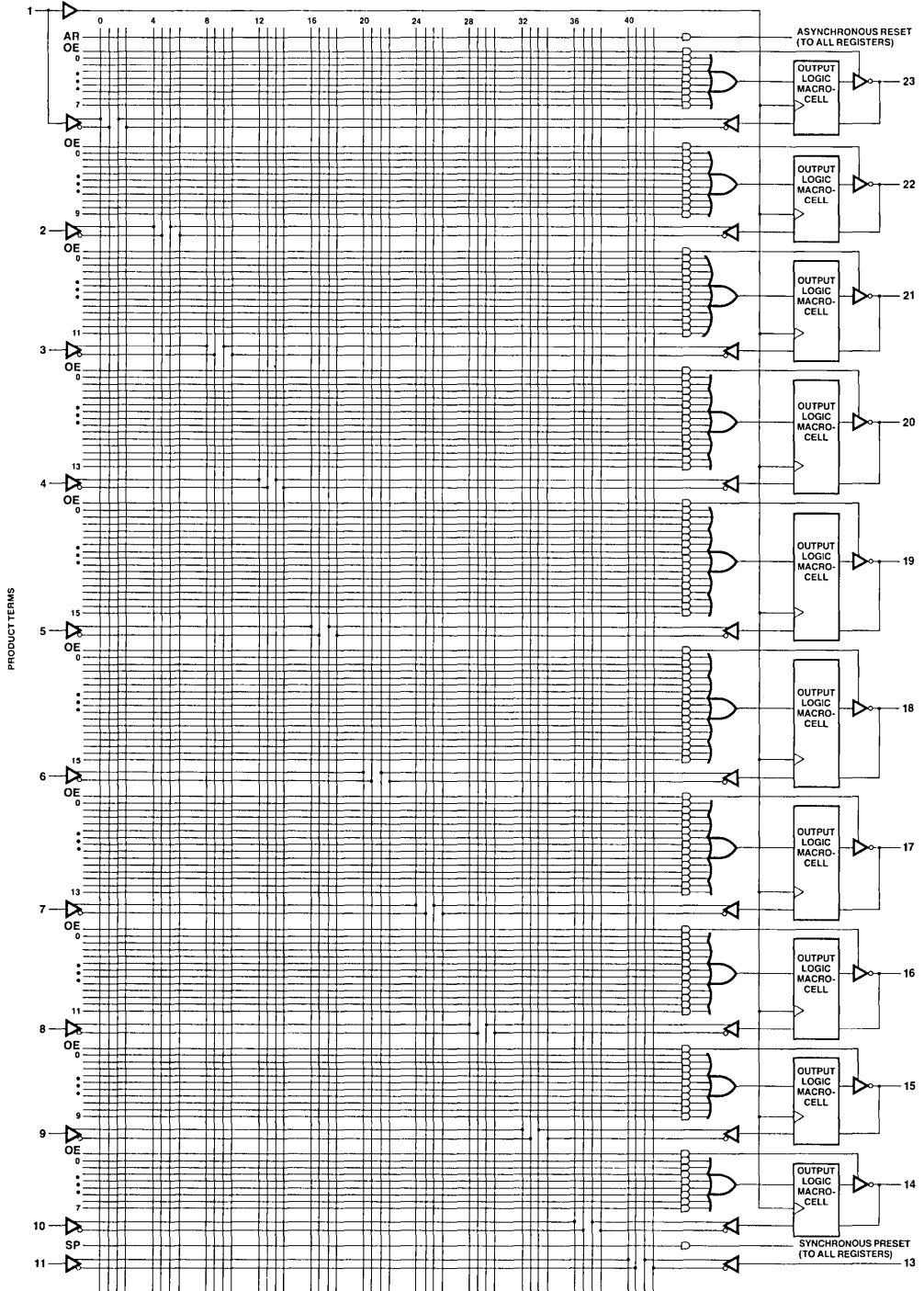
04162A-11

MAXIMUM RATINGS (Above which the useful life may be impaired)

Storage Temperature	-65 to $+150°C$
Supply Voltage to Ground Potential (Pin 24 to Pin 12) Continuous	-0.5 to $+7V$
DC Voltage Applied to Outputs (Except During Programming)	$-0.5V$ to $+V_{CC}$ Max
DC Voltage Applied to Outputs During Programming	$21V$
Output Current Into Outputs During Programming (Max Duration of 1 sec)	$200mA$
DC Input Voltage	-0.5 to $+5.5V$
DC Input Current	-30 to $+5mA$
Ambient Temperature with Power Applied	$+125°C$

OPERATING RANGE

Parameters	Description	Commercial		Military		Units
		Min	Max	Min	Max	
V_{CC}	Supply Voltage	4.75	5.25	4.50	5.50	V
T_A	Operating Free Air Temperature	0	75	-55		°C
T_C	Operating Case Temperature				125	°C

ELECTRICAL CHARACTERISTICS OVER OPERATING RANGE (Unless Otherwise Noted)
Advanced Information

Parameters	Description	Test Conditions			Min	Typ (Note 1)	Max	Units
V_{OH}	Output HIGH Voltage	V_{CC} = MIN, $V_{IN} = V_{IH}$ or V_{IL}	$I_{OH} = -3.2mA$	COM'L	2.4	3.5		Volts
			$I_{OH} = -2mA$	MIL				
V_{OL}	Output LOW Voltage	V_{CC} = MIN, $V_{IN} = V_{IH}$ or V_{IL}	$I_{OL} = 24mA$	COM'L			0.50	Volts
			$I_{OL} = 12mA$	MIL				
V_{IH} (Note 2)	Input HIGH Level	Guaranteed input logical HIGH voltage for all inputs			2.0			Volts
V_{IL} (Note 2)	Input LOW Level	Guaranteed input logical LOW voltage for all inputs					0.8	Volts
I_{IL}	Input LOW Current	V_{CC} = MAX, V_{IN} = 0.40V				-20	-250	μA
I_{IH}	Input HIGH Current	V_{CC} = MAX, V_{IN} = 2.7V					25	μA
I_I	Input HIGH Current	V_{CC} = MAX, V_{IN} = 5.5V					1.0	mA
I_{SC}	Output Short Circuit Current	V_{CC} = MAX, V_{OUT} = 0.5V (Note 3)			-30	-60	-90	mA
I_{CC}	Power Supply Current	All Inputs = GND, V_{CC} = MAX				120	180	mA
V_I	Input Clamp Voltage	V_{CC} = MIN, $I_{IN} = -18mA$				-0.9	-1.2	Volts
I_{OZH}	Output Leakage Current (Note 4)	V_{CC} = MAX, V_{IL} = 0.8V V_{IH} = 2.0V	V_O = 2.7V				100	μA
I_{OZL}			V_O = 0.4V				-100	
C_{IN}	Input Capacitance	V_{IN} = 2.0V @f = 1MHz (Note 5)				6		pF
C_{OUT}	Output Capacitance	V_{OUT} = 2.0V @f = 1MHz (Note 5)				9		

Notes: 1. Typical limits are at V_{CC} = 5.0V and T_A = 25°C.
2. These are absolute values with respect to device ground and all overshoots due to system or tester noise are included.
3. Not more than one output should be tested at a time. Duration of the short circuit should not be more than one second. V_{OUT} = 0.5V has been chosen to avoid test problems caused by tester ground degradation.
4. I/O pin leakage is the worst case of I_{OZX} or I_{IX} (where X = H or L).
5. These parameters are not 100% tested, but are periodically sampled.

SWITCHING CHARACTERISTICS OVER OPERATING RANGE (Unless otherwise noted)
Advanced Information

Parameters	Description	Test Conditions	Typ (Note 1)	COM'L "A" Min	COM'L "A" Max	COM'L "Std" Min	COM'L "Std" Max	MIL "A" Min	MIL "A" Max	MIL "Std" Min	MIL "Std" Max	Units
t_{PD}	Input or Feedback to Non-Registered Output		18									ns
t_{EA}	Input to Output Enable	COM'L $R_1 = 200$ $R_2 = 390$	18									ns
t_{ER}	Input to Output Disable		18									ns
t_{CO}	Clock to Output		10									ns
t_S	Input or Feedback Setup Time		18									ns
t_H	Hold Time	MIL $R_1 = 390$ $R_2 = 750$										ns
t_P	Clock Period											ns
t_W	Clock Width											ns
f_{MAX}	Maximum Frequency											MHz

Notes: 1. Typical limits are at $V_{CC} = 5.0V$ and $T_A = 25°C$.
2. t_{PD} is tested with switch S_1 closed and $C_L = 50pF$.
3. For three-state outputs, output enable times are tested with $C_L = 50pF$ to the 1.5V level; S_1 is open for high impedance to HIGH tests and closed for high impedance to LOW tests. Output disable times are tested with $C_L = 5pF$. HIGH to high impedance tests are made to an output voltage of $V_{OH} - 0.5V$ with S_1 open; LOW to high impedance tests are made to the $V_{OL} + 0.5V$ level with S_1 closed.

SWITCHING WAVEFORMS

04162A-12

KEY TO TIMING DIAGRAM

AC TEST LOAD

04162A-13

2-40

INPUT AND OUTPUT CIRCUITRY

INPUT CIRCUITRY
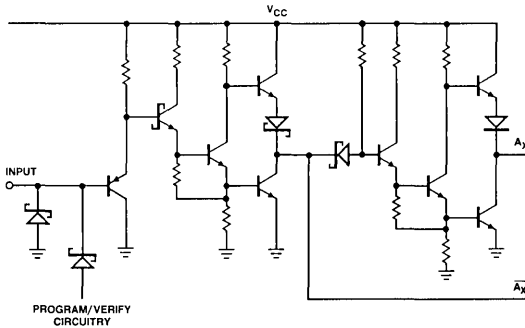
04162A-14

OUTPUT CIRCUITRY
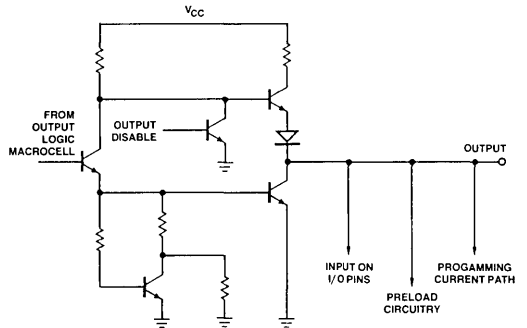
04162A-15

POWER-UP RESET

The registered devices in the AMD PAL family have been designed with the capability to reset during system power-up. Following power-up, all registers will be zero, setting the outputs to all ones. This feature provides extra flexibility to the designer and is especially valuable in simplifying state machine initialization. A timing diagram and parameter table are shown below. Due to the asynchronous operation of the power-up reset and the wide range of ways V_{CC} can rise to its steady state, two conditions are required to insure a valid power-up reset. These conditions are:

1. The V_{CC} rise must be monotonic.
2. Following reset, the clock input must not be driven from low to high until all applicable input and feedback setup times are met.

Following power-up, all registers are set to zero, setting all outputs to one. This provides extra flexibility to the designer and is especially valuable in simplifying state machine initialization. A timing diagram and parameter table are shown below.

04114A-13

Parameters	Description	Min	Typ	Max	Units
t_{pr}	Power-Up Reset Time		600	1000	ns
t_s	Input or Feedback Setup Time		See Switching Characteristics		
t_w	Clock Width				

2-41
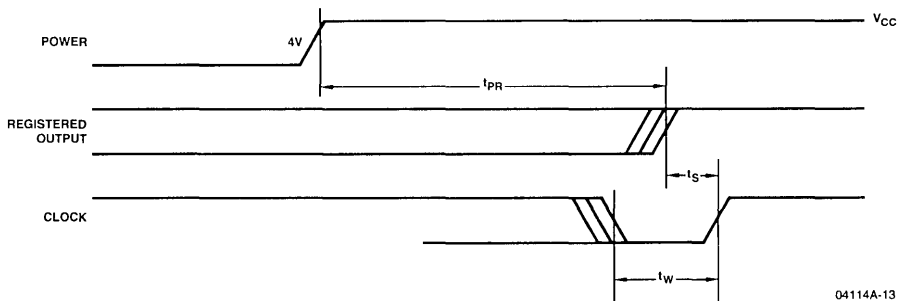
PROGRAMMING AND VERIFICATION

The AmPAL22V10 is programmed and verified using AMD's standard programmable logic programming algorithm. The fuse to be programmed is selected by input line number (array column), product term (array row), and by output (one at a time). The fuse is then programmed and verified by applying a simple sequence of voltages to two control pins (1 and 13).

Input line numbers (0 – 43) are addressed using a full decode scheme via TTL levels on pins 6 – 11 where 6 is the LSB and 11 is the MSB. Even numbered input lines represent the true version of a signal and odd numbered lines represent the complement. Input line addressing is shown in Table 1. Note that input lines 44 – 62 are reserved for further expansion and input line 63 is utilized for selecting the fuses used for programming output polarity and whether the output is registered or combinatorial.

Product terms are addressed using a one-of-24 addressing scheme on pins 2 – 5 where pin 2 is the LSB and 5 is the MSB. Product term addressing is shown in Table 2. Logical product terms (0 – 15) are selected via TTL levels on the four addressing pins. Note that outputs with fewer than 16 product terms will decode blank space for decoding values greater than the number of product terms on that output. Architectural product terms are selected by placing a zener voltage level (V_{HH}) on the MSB (pin 5) and using pins 2 – 4 for an additional eight decoding states (only 5 are used). The specific decoding of architectural features is best shown in Table 2.

Fuse selection by output must be done one output at a time (following control pin 1 going to V_{HH}) as shown in the programming timing diagram (Figure 1).

Once fuses have been selected, the simple programming and verification sequence may be completed as shown in Figure 1. AC and DC requirements for programming are shown in the programming parameter table.

SECURITY FUSE PROGRAMMING

A single fuse is provided on each AmPAL22V10 part to prevent unauthorized copying of PAL fuse patterns. Once blown, the circuitry enabling fuse verification and registered output preload is permanently disabled.

Programming of the security fuse is the same as shown before. Verification of a blown security fuse is accomplished by verifying the whole fuse array as if every fuse is blown.

PROGRAMMING YIELD

AMD PALs have been designed to insure extremely high programming yields (>98%). To help insure that a part was correctly programmed, once programming is completed, the entire fuse array should be reverified at both low and high V_{CC}. Reverification can be accomplished by reading all ten outputs in parallel rather than one at a time. This verification cycle checks that the array fuses have been blown and can be sensed by the outputs under varying conditions.

AMD PALs contain many internal test features, including circuitry and extra fuses which allow AMD to test the ability of each part to perform programming before shipping, to assure high programming yields and correct logical operation for a correctly programmed part. Programming yield losses are most likely due to poor programming socket contact, programming equipment out of calibration, or improper usage of said equipment.

PROGRAMMING PARAMETERS ($T_A = 25°C$)

Parameters	Description		Min	Typ	Max	Units
V_{HH}	Control Pin Extra High Level	Pin 1 @ 10–40mA	10	11	12	Volts
		Pin 13 @ 10–40mA	10	11	12	
V_{OP}	Program Voltage Pins 14–23 @ 15–200mA		18	20	22	Volts
V_{IHP}	Input High Level During Programming and Verify		2.4	5	5.5	Volts
V_{ILP}	Input Low Level During Programming and Verify		0.0	0.3	0.5	Volts
V_{CCP}	V_{CC} During Programming @ $I_{CC} = 50–200mA$		5	5.2	5.5	Volts
V_{CCL}	V_{CC} During First Pass Verification @ $I_{CC} = 50–200mA$		4.1	4.3	4.5	Volts
V_{CCH}	V_{CC} During Second Pass Verification @ $I_{CC} = 50–200mA$		5.4	5.7	6.0	Volts
V_{Blown}	Successful Blown Fuse Sense Level @ Output			0.3	0.5	Volts
dV_{OP}/dt	Rate of Output Voltage Change		20		250	V/μsec
dV_{13}/dt	Rate of Fusing Enable Voltage Change (Pin 13 Rising Edge)		100		1000	V/μsec
t_P	Fusing Time First Attempt		40	50	100	μsec
	Subsequent Attempts		4	5	10	msec
t_D	Delays Between Various Level Changes		100	200	1000	ns
t_V	Period During which Output is Sensed for V_{Blown} Level				500	ns
V_{ONP}	Pull-Up Voltage On Outputs Not Being Programmed		$V_{CCP} - 0.3$	V_{CCP}	$V_{CCP} + 0.3$	Volts
R	Pull-Up Resistor On Outputs Not Being Programmed		1.9	2	2.1	kΩ

AmPAL22V10 PROGRAMMING EQUIPMENT INFORMATION

Source and Location	Data I/O 10525 Willows Rd., N.E. Redmond, WA 98052	Kontron Electronics, Inc. 630 Price Ave. Redwood City, CA 94063	Stag Microsystems 528-5 Weddel Dr. Sunnyvale, CA 94086	Structured Design, Inc. 1700 Wyatt Dr. #3 Santa Clara, CA 95054	Digilec, Inc. 7335 E. Acoma Dr. Dept-103 Scottsdale, AZ 85260
Programmer Model(s)	Model-100, 29, 19 or 17	TBA	Model-PPX	SD 1000	TBA
			Model ZL-30		
AMD PAL Personality Module	Logicpak 950-1942-001		PPM2200	On Board	–
			On Board ZL-30		
Socket Adapter	715-1947-003	TBA	Am203S	On Board	–
			On Board ZL-30		

The machines noted above have been qualified by AMD to insure high programming yields. Check with the factory to determine the current status of vendors noted TBA or other available models.

PROGRAMMING WAVEFORMS

PIN 5

PINS 2-11   INPUT LINE NUMBER AND COLUMN LINE NUMBER ADDRESSES V_{HH} H   L

PIN 1 V_{HH} V_{IHP}

FUSING TIME

V_{OP}

OUTPUT ASSOCIATED WITH PRODUCT TERM TO BE PROGRAMMED   Z   Z   H   L

PIN 13 V_{HH} V_{HH}

V_{ILP}

t_D t_D t_P t_D t_D t_D t_D t_V t_D

INPUT LINE NUMBER AND COLUMN NUMBER SELECTED   FUSING ENABLED   PROGRAMMING VOLTAGE REMOVED   OUTPUTS ENABLED   REGISTERS CLOCKED   VERIFY   CYCLE ENDS

PROGRAMMING VOLTAGE APPLIED   FUSE BLOWN = LOW

OUTPUTS DISABLED NORMAL INPUTS DISABLED, PROGRAMMING DECODE ENABLED

04162A-16

TABLE 1. INPUT ADDRESSING

Input Line Number	Input Line Number Address Pin States						Input Line Number	Input Line Number Address Pin States					
	11	10	9	8	7	6		11	10	9	8	7	6
0	L	L	L	L	L	L	32	H	L	L	L	L	L
1	L	L	L	L	L	H	33	H	L	L	L	L	H
2	L	L	L	L	H	L	34	H	L	L	L	H	L
3	L	L	L	L	H	H	35	H	L	L	L	H	H
4	L	L	L	H	L	L	36	H	L	L	H	L	L
5	L	L	L	H	L	H	37	H	L	L	H	L	H
6	L	L	L	H	H	L	38	H	L	L	H	H	L
7	L	L	L	H	H	H	39	H	L	L	H	H	H
8	L	L	H	L	L	L	40	H	L	H	L	L	L
9	L	L	H	L	L	H	41	H	L	H	L	L	H
10	L	L	H	L	H	L	42	H	L	H	L	H	L
11	L	L	H	L	H	H	43	H	L	H	L	H	H
12	L	L	H	H	L	L							
13	L	L	H	H	L	H							
14	L	L	H	H	H	L							
15	L	L	H	H	H	H							
16	L	H	L	L	L	L							
17	L	H	L	L	L	H							
18	L	H	L	L	H	L							
19	L	H	L	L	H	H							
20	L	H	L	H	L	L				Reserved			
21	L	H	L	H	L	H							
22	L	H	L	H	H	L							
23	L	H	L	H	H	H							
24	L	H	H	L	L	L							
25	L	H	H	L	L	H							
26	L	H	H	L	H	L							
27	L	H	H	L	H	H							
28	L	H	H	H	L	L							
29	L	H	H	H	L	H							
30	L	H	H	H	H	L							
31	L	H	H	H	H	H	63*	H	H	H	H	H	H

*Architecture row.

TABLE 2. COLUMN NUMBER ADDRESSING

Column Number										Column Number Select Address Pin States				
										5	4	3	2	
0	0	0	0	0	0	0	0	0	0	L	L	L	L	Logical PT's
1	1	1	1	1	1	1	1	1	1	L	L	L	H	
2	2	2	2	2	2	2	2	2	2	L	L	H	L	
3	3	3	3	3	3	3	3	3	3	L	L	H	H	
4	4	4	4	4	4	4	4	4	4	L	H	L	L	
5	5	5	5	5	5	5	5	5	5	L	H	L	H	
6	6	6	6	6	6	6	6	6	6	L	H	H	L	
7	7	7	7	7	7	7	7	7	7	L	H	H	H	
–	8	8	8	8	8	8	8	8	–	H	L	L	L	
–	9	9	9	9	9	9	9	9	–	H	L	L	H	
–	–	10	10	10	10	10	10	–	–	H	L	H	L	
–	–	11	11	11	11	11	11	–	–	H	L	H	H	
–	–	–	12	12	12	12	–	–	–	H	H	L	L	
–	–	–	13	13	13	13	–	–	–	H	H	L	H	
–	–	–	–	14	14	–	–	–	–	H	H	H	L	
–	–	–	–	15	15	–	–	–	–	H	H	H	H	
OE	OE	OE	OE	OE	OE	OE	OE	OE	OE	HH	L	L	L	Output Enable
P	P	P	P	P	P	P	P	P	P	HH	L	L	H	Output Polarity
R	R	R	R	R	R	R	R	R	R	HH	L	H	L	Register/Non-Register Output
–	AR*	–	–	–	–	–	–	SP**	–	HH	L	H	H	*Asynchronous Reset
–	SF	–	–	–	–	–	–	–	–	HH	H	L	H	**Synchronous Preset
Pin 23	Pin 22	Pin 21	Pin 20	Pin 19	Pin 18	Pin 17	Pin 16	Pin 15	Pin 14	L = V_ILP H = V_IHP HH = V_HH				Security Fuse (Special Verify Required)
Programming Access and Verify Pin														

ORDERING INFORMATION

AmPAL22V10   D   M   B

Screening
* = Standard Process Flow
B = Burned-In

Package
D = Hermetic DIP
L = Chip-Pak
P = Molded DIP (Commercial Only — to Be Announced)

Temperature Range
C = Commercial
M = Military

*A blank in this position of the ordering code indicates the part has been screened to standard process flow.

PHYSICAL DIMENSIONS
Dual In-Line

D-24-1AA
Hermetic DIP

P-24-1AA
Molded DIP

.245
.285

.045
.065

24   13
1   12

.010

SEATING
PLANE

1.230
1.285

.140
.220

.290
.320

.120
.150

.009
.011

.090   .016
.110   .020

.015
.060

3°
13°
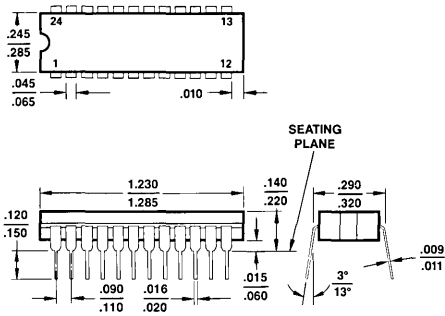
TO BE
ANNOUNCED

AmPL64S16

Fuse Microprogrammable Controllers
Advanced Micro Devices

DISTINCTIVE CHARACTERISTICS

- Implements complex state machines
- 64 words of 32-bit wide microprogram memory
- 16 control outputs
- 7 conditional test inputs
- Cascadable to any width
- High level instruction set
  - conditional
  - conditional looping
  - conditional subroutine call
  - multiway branch
- SSR™ diagnostics on chip

FUNCTIONAL DESCRIPTION

The AmPL64S16 is a single-chip Fuse Microprogrammable Controller (FMC) which allows complex control sequences to be implemented by programming the desired series of microinstructions. Because the AmPL64S16 FMC is micropro-grammable, designers can easily design complex state machines without the need for Karnaugh Map or Boolean Equation design techniques. State machines may be constructed directly from high-level microinstruction control flow definitions. Jumps, loops and subroutine calls, which can be conditionally executed based on the test inputs, provide the designer with powerful control flow primitives.

A microprogram address sequencer is the heart of the FMC. It provides the microprogram address to an internal 64 word by 32-bit PROM. Instructions such as loop, subroutine call, or branch can be executed conditionally based on the values on the test inputs. The 32-bit microinstruction pipeline register holds the sequencer instruction access time.

BLOCK DIAGRAM

04179A-1

04179A-PLP
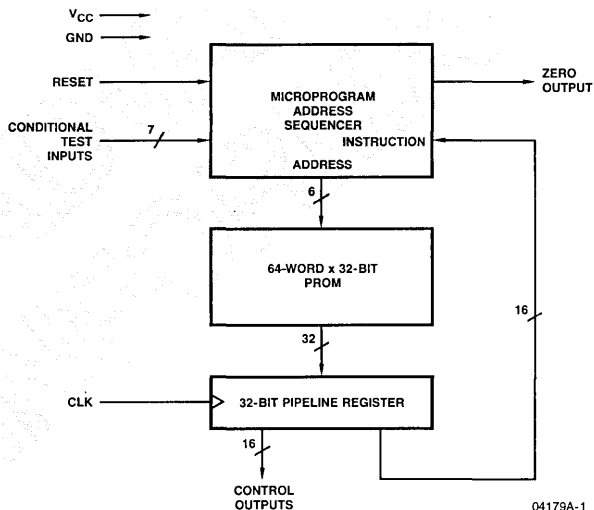
Am27S12A • Am27S13A
Am27S12 • Am27S13

2048-Bit Generic Series Bipolar PROM
(512 x 4 bits with ultra fast access time)

"A" VERSION ADVANCED INFORMATION

DISTINCTIVE CHARACTERISTICS

- High Speed – 30ns max commercial range access time
- Excellent performance over full MIL and commercial ranges
- Highly reliable, ultra-fast programming Platinum-Silicide fuses
- High programming yield
- Low current PNP inputs
- High current open collector and three-state outputs
- Fast chip select
- Access time tested with N^2 patterns
- Pin for pin replacements for industry standard products
- Common Generic PROM series electrical characteristics and simple programming procedures

GENERIC SERIES CHARACTERISTICS

The Am27S12A/12 and Am27S13A/13 are members of an Advanced PROM series incorporating common electrical characteristics and programming procedures. All parts in this series are produced with a fusible link at each memory location storing a logic LOW and can be selectively programmed to a logic HIGH by applying appropriate voltages to the circuit.

All parts are fabricated with AMD's fast programming highly reliable Platinum-Silicide Fuse technology. Utilizing easily implemented programming (and common programming personality card sets) these products can be rapidly programmed to any customized pattern. Extra test words are pre-programmed during manufacturing to insure extremely high field programming yields, and produce excellent parametric correlation.
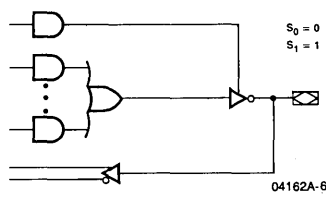
Platinum-Silicide was selected as the fuse link material to achieve a well controlled melt rate resulting in large nonconductive gaps that ensure very stable long term reliability. Extensive operating testing has proven that this low-field, large-gap technology offers the best reliability for fusible link PROMs.
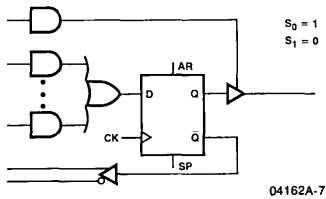
Common design features include active loading of all critical AC paths regulated by a built-in temperature and voltage compensated bias network to provide excellent parametric performance over MIL supply and temperature ranges. Selective feedback techniques have been employed to minimize delays through all critical paths producing the fastest speeds possible from Schottky processed PROMs.

FUNCTIONAL DESCRIPTION

The Am27S12A/12 and Am27S13A/13 are high speed electrically programmable Schottky read only memories. Organized in the industry standard 512 x 4 configuration, they are available in both open collector Am27S12A/12 and three-state Am27S13A/13 output versions. After programming, stored information is read on outputs O_0-O_3 by applying unique binary addresses to A_0-A_8 and holding the chip select input, $\overline{CS}$, at a logic LOW. If the chip select input goes to a logic HIGH, O_0-O_3 go to the off or high impedance state.

BLOCK DIAGRAM

03208A-1

CONNECTION DIAGRAMS – Top Views

DIP

Chip-Pak™
L-20-1

Note: Pin 1 is marked for orientation.

03208A-2

03208A-3

03208A-BPM

MAXIMUM RATINGS (Above which the useful life may be impaired)

Storage Temperature	−65 to +150°C
Temperature (Ambient) Under Bias	−55 to +125°C
Supply Voltage to Ground Potential (Pin 16 to Pin 8) Continuous	−0.5 to +7.0V
DC Voltage Applied to Outputs (Except During Programming)	−0.5V to +V_{CC} max
DC Voltage Applied to Outputs During Programming	21V
Output Current into Outputs During Programming (Max Duration of 1 sec)	250mA
DC Input Voltage	−0.5 to +5.5V
DC Input Current	−30 to +5mA

OPERATING RANGE

Range	V_{CC}	Temperature
COM'L	4.75 to 5.25V	$T_A = 0$ to +75°C
MIL	4.5 to 5.5V	$T_C = -55$ to +125°C

LOGIC SYMBOL


```
5 ───── A₀
6 ───── A₁
7 ───── A₂
4 ───── A₃
3 ───── A₄      512 x 4
2 ───── A₅       PROM
1 ───── A₆
15 ──── A₇
14 ──── A₈
13 ───○ CS    O₀   O₁   O₂   O₃

              12   11   10   9
```

V_{CC} = Pin 16
GND = Pin 8

03208A-4

ELECTRICAL CHARACTERISTICS OVER OPERATING RANGE (Unless Otherwise Noted)

Parameters	Description	Test Conditions		Min	Typ (Note 1)	Max	Units
V_{OH} (Note 2)	Output HIGH Voltage	V_{CC} = MIN, I_{OH} = −2.0mA $V_{IN} = V_{IH}$ or V_{IL}		2.4			Volts
V_{OL}	Output LOW Voltage	V_{CC} = MIN, I_{OL} = 16mA $V_{IN} = V_{IH}$ or V_{IL}				0.45	Volts
V_{IH}	Input HIGH Level	Guaranteed input logical HIGH voltage for all inputs (Note 3)		2.0			Volts
V_{IL}	Input LOW Level	Guaranteed input logical LOW voltage for all inputs (Note 3)				0.8	Volts
I_{IL}	Input LOW Current	V_{CC} = MAX, V_{IN} = 0.45V			−0.010	−0.250	mA
I_{IH}	Input HIGH Current	V_{CC} = MAX, V_{IN} = 2.7V				25	μA
I_{SC} (Note 2)	Output Short Circuit Current	V_{CC} = MAX, V_{OUT} = 0.0V (Note 4)		−20	−40	−90	mA
I_{CC}	Power Supply Current	All inputs = GND V_{CC} = MAX			100	130	mA
V_I	Input Clamp Voltage	V_{CC} = MIN, I_{IN} = −18mA				−1.2	Volts
I_{CEX}	Output Leakage Current	V_{CC} = MAX $V_{\overline{CS}}$ = 2.4V (Note 2)	V_O = 4.5V			40	μA
			V_O = 2.4V			40	
			V_O = 0.4V			−40	
C_{IN}	Input Capacitance	V_{IN} = 2.0V @ f = 1MHz (Note 5)			4		pF
C_{OUT}	Output Capacitance	V_{OUT} = 2.0V @ f = 1MHz (Note 5)			8		

Notes: 1. Typical limits are at V_{CC} = 5.0V and T_A = 25°C.
     2. This applies to three-state devices only.
     3. These are absolute voltages with respect to device ground pin and include all overshoots due to system and/or tester noise. Do not attempt to test these values without suitable equipment.
     4. Not more than one output should be shorted at a time. Duration of the short circuit should not be more than one second.
     5. These parameters are not 100% tested, but are periodically sampled.

Am27S18A • Am27S19A
Am27S18 • Am27S19

256-Bit Generic Series Bipolar PROM
(32 x 8 bits with ultra fast access time)
"A" VERSION ADVANCED INFORMATION

DISTINCTIVE CHARACTERISTICS

- High Speed – 25ns max commercial range access time
- Excellent performance over full MIL and commercial ranges
- Highly reliable, ultra-fast programming Platinum-Silicide fuses
- High programming yield
- Low current PNP inputs
- High current open collector and three-state outputs
- Fast chip select
- Access time tested with N^2 patterns
- Pin for pin replacements for industry standard products
- Common Generic PROM series electrical characteristics and simple programming procedures

FUNCTIONAL DESCRIPTION

The Am27S18A/18 and Am27S19A/19 are high speed electrically programmable Schottky read only memories. Organized in the industry standard 32 x 8 configuration, they are available in both open collector Am27S18A/18 and three-state Am27S19A/19 output versions. After programming, stored information is read on outputs O_0-O_7 by applying unique binary addresses to A_0-A_4 and holding the chip select input, $\overline{CS}$, at a logic LOW. If the chip select input goes to a logic HIGH, O_0-O_7 go to the off or high impedance state.

GENERIC SERIES CHARACTERISTICS

This Am27S18A/18 and Am27S19A/19 are members of an Advanced PROM series incorporating common electrical characteristics and programming procedures. All parts in this series are produced with a fusible link at each memory location storing a logic LOW and can be selectively programmed to a logic HIGH by applying appropriate voltages to the circuit.

All parts are fabricated with AMD's fast programming highly reliable Platinum-Silicide Fuse technology. Utilizing easily implemented programming (and common programming personality card sets) these products can be rapidly programmed to any customized pattern. Extra test words are pre-programmed during manufacturing to insure extremely high field programming yields, and produce excellent parametric correlation.

Platinum-Silicide was selected as the fuse link material to achieve a well controlled melt rate resulting in large non-conductive gaps that ensure very stable long-term reliability. Extensive operating testing has proven that this low-field, large-gap technology offers the best reliability for fusible link PROMs.

Common design features include active loading of all critical AC paths regulated by a built-in temperature and voltage compensated bias network to provide excellent parametric performance over MIL supply and temperature ranges. Selective feedback techniques have been employed to minimize delays through all critical paths producing the fastest speeds possible from Schottky processed PROMs.

BLOCK DIAGRAM

03209A-1

CONNECTION DIAGRAMS
Top Views

DIP

03209A-2

Chip-Pak™
L-20-1

03209A-3

Note: Pin 1 is marked for orientation.

03209A-BPM

MAXIMUM RATINGS (Above which the useful life may be impaired)

Storage Temperature	-65 to $+150°C$
Temperature (Ambient) Under Bias	-55 to $+125°C$
Supply Voltage to Ground Potential (Pin 16 to Pin 8) Continuous	-0.5 to $+7.0V$
DC Voltage Applied to Outputs (Except During Programming)	$-0.5V$ to $+V_{CC}$ max
DC Voltage Applied to Outputs During Programming	21V
Output Current into Outputs During Programming (Max Duration of 1 sec)	250mA
DC Input Voltage	-0.5 to $+5.5V$
DC Input Current	-30 to $+5mA$

OPERATING RANGE

Range	V_{CC}	Temperature
COM'L	4.75 to 5.25V	$T_A = 0$ to $+75°C$
MIL	4.5 to 5.5V	$T_C = -55$ to $+125°C$

LOGIC SYMBOL

V_{CC} = Pin 16
GND = Pin 8

03209A-4

ELECTRICAL CHARACTERISTICS OVER OPERATING RANGE (Unless Otherwise Noted)

Parameters	Description	Test Conditions		Min	Typ (Note 1)	Max	Units
V_{OH} (Note 2)	Output HIGH Voltage	V_{CC} = MIN, $I_{OH} = -2.0mA$ $V_{IN} = V_{IH}$ or V_{IL}		2.4			Volts
V_{OL}	Output LOW Voltage	V_{CC} = MIN, $I_{OL} = 16mA$ $V_{IN} = V_{IH}$ or V_{IL}				0.45	Volts
V_{IH}	Input HIGH Level	Guaranteed input logical HIGH voltage for all inputs (Note 3)		2.0			Volts
V_{IL}	Input LOW Level	Guaranteed input logical LOW voltage for all inputs (Note 3)				0.8	Volts
I_{IL}	Input LOW Current	V_{CC} = MAX, $V_{IN} = 0.45V$			-0.010	-0.250	mA
I_{IH}	Input HIGH Current	V_{CC} = MAX, $V_{IN} = 2.7V$				25	μA
I_{SC} (Note 2)	Output Short Circuit Current	V_{CC} = MAX, $V_{OUT} = 0.0V$ (Note 4)		-20	-40	-90	mA
I_{CC}	Power Supply Current	All inputs = GND, V_{CC} = MAX			90	115	mA
V_I	Input Clamp Voltage	V_{CC} = MIN, $I_{IN} = -18mA$				-1.2	Volts
I_{CEX}	Output Leakage Current	V_{CC} = MAX $V_{\overline{CS}} = 2.4V$	$V_O = 4.5V$			40	μA
			(Note 2) $V_O = 2.4V$			40	
			$V_O = 0.4V$			-40	
C_{IN}	Input Capacitance	$V_{IN} = 2.0V$ @ f = 1MHz (Note 5)			4		pF
C_{OUT}	Output Capacitance	$V_{OUT} = 2.0V$ @ f = 1MHz (Note 5)			8		

Notes: 1. Typical limits are at $V_{CC} = 5.0V$ and $T_A = 25°C$.
2. This applies to three-state devices only.
3. These are absolute voltages with respect to device ground pin and include all overshoots due to system and/or tester noise.
   Do not attempt to test these values without suitable equipment.
4. Not more than one output should be shorted at a time. Duration of the short circuit should not be more than one second.
5. These parameters are not 100% tested, but are periodically sampled.

Am27S20A • Am27S21A
Am27S20 • Am27S21
1024-Bit Generic Series Bipolar PROM
(256 x 4 bits with ultra fast access time)
"A" VERSION ADVANCED INFORMATION

DISTINCTIVE CHARACTERISTICS

- High Speed — 30ns max commercial range access time
- Excellent performance over full MIL and commercial ranges
- Highly reliable, ultra-fast programming Platinum-Silicide fuses
- High programming yield
- Low current PNP inputs
- High current open collector and three-state outputs
- Fast chip select
- Access time tested with N^2 patterns
- Pin for pin replacements for industry standard products
- Common Generic PROM series electrical characteristics and simple programming procedures

GENERIC SERIES CHARACTERISTICS

The Am27S20A/20 and Am27S21A/21 are members of an Advanced PROM series incorporating common electrical characteristics and programming procedures. All parts in this series are produced with a fusible link at each memory location storing a logic LOW and can be selectively programmed to a logic HIGH by applying appropriate voltages to the circuit.

All parts are fabricated with AMD's fast programming highly reliable Platinum-Silicide Fuse technology. Utilizing easily implemented programming (and common programming personality card sets) these products can be rapidly programmed to any customized pattern. Extra test words are pre-programmed during manufacturing to insure extremely high field programming yields, and produce excellent parametric correlation.

Platinum-Silicide was selected as the fuse link material to achieve a well controlled melt rate resulting in large non-conductive gaps that ensure very stable long term reliability. Extensive operating testing has proven that this low-field, large-gap technology offers the best reliability for fusible link PROMs.

Common design features include active loading of all critical AC paths regulated by a built-in temperature and voltage compensated bias network to provide excellent parametric performance over MIL supply and temperature ranges. Selective feedback techniques have been employed to minimize delays through all critical paths producing the fastest speeds possible from Schottky processed PROMs.

FUNCTIONAL DESCRIPTION

The Am27S20A/20 and Am27S21A/21 are high speed electrically programmable Schottky read only memories. Organized in the industry standard 256 x 4 configuration, they are available in both open collector Am27S20A/20 and three-state Am27S21A/21 output versions. After programming, stored information is read on outputs O_0-O_3 by applying unique binary addresses to A_0-A_7 and holding chip select inputs, $\overline{CS}_1$ and $\overline{CS}_2$, at a logic LOW. If either chip select input goes to a logic HIGH, O_0-O_3 go to the OFF or high impedance state.

BLOCK DIAGRAM
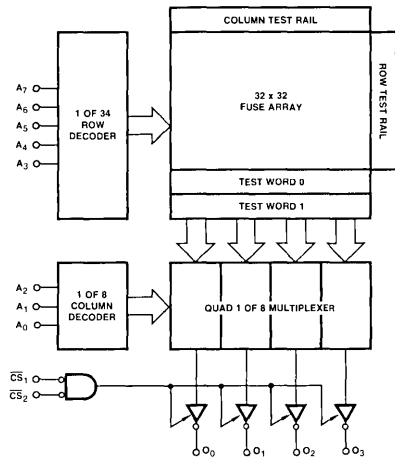
03206A-1

CONNECTION DIAGRAMS – Top Views

DIP
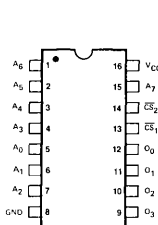
03206A-2

Chip-Pak™
L-20-1
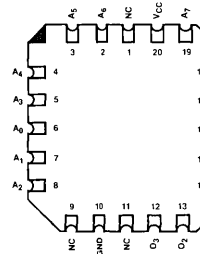
03206A-3

Note: Pin 1 is marked for orientation.

MAXIMUM RATINGS (Above which the useful life may be impaired)

Storage Temperature	-65 to $+150°C$
Temperature (Ambient) Under Bias	-55 to $+125°C$
Supply Voltage to Ground Potential (Pin 16 to Pin 8) Continuous	-0.5 to $+7.0V$
DC Voltage Applied to Outputs (Except During Programming)	$-0.5V$ to $+V_{CC}$ max
DC Voltage Applied to Outputs During Programming	21V
Output Current into Outputs During Programming (Max Duration of 1 sec)	250mA
DC Input Voltage	-0.5 to $+5.5V$
DC Input Current	-30 to $+5mA$

OPERATING RANGE

Range	V_{CC}	Temperature
COM'L	4.75 to 5.25V	$T_A = 0$ to $+75°C$
MIL	4.5 to 5.5V	$T_C = -55$ to $+125°C$

LOGIC SYMBOL

$V_{CC} =$ Pin 16
GND $=$ Pin 8

03206A-4

ELECTRICAL CHARACTERISTICS OVER OPERATING RANGE (Unless Otherwise Noted)

Parameters	Description	Test Conditions			Min	Typ (Note 1)	Max	Units
V_{OH} (Note 2)	Output HIGH Voltage	$V_{CC} =$ MIN, $I_{OH} = -2.0mA$ $V_{IN} = V_{IH}$ or V_{IL}			2.4			Volts
V_{OL}	Output LOW Voltage	$V_{CC} =$ MIN, $I_{OL} = 16mA$ $V_{IN} = V_{IH}$ or V_{IL}					0.45	Volts
V_{IH}	Input HIGH Level	Guaranteed input logical HIGH voltage for all inputs (Note 3)			2.0			Volts
V_{IL}	Input LOW Level	Guaranteed input logical LOW voltage for all inputs (Note 3)					0.8	Volts
I_{IL}	Input LOW Current	$V_{CC} =$ MAX, $V_{IN} = 0.45V$				-0.010	-0.250	mA
I_{IH}	Input HIGH Current	$V_{CC} =$ MAX, $V_{IN} = 2.7V$					25	μA
I_{SC} (Note 2)	Output Short Circuit Current	$V_{CC} =$ MAX, $V_{OUT} = 0.0V$ (Note 4)			-20	-40	-90	mA
I_{CC}	Power Supply Current	All inputs = GND $V_{CC} =$ MAX				100	130	mA
V_I	Input Clamp Voltage	$V_{CC} =$ MIN, $I_{IN} = -18mA$					-1.2	Volts
I_{CEX}	Output Leakage Current	$V_{CC} =$ MAX $V_{\overline{CS}_1} = 2.4V$	$V_O = 4.5V$				40	μA
			$V_O = 2.4V$ (Note 2)				40	
			$V_O = 0.4V$				-40	
C_{IN}	Input Capacitance	$V_{IN} = 2.0V$ @ f = 1MHz (Note 5)				4		pF
C_{OUT}	Output Capacitance	$V_{OUT} = 2.0V$ @ f = 1MHz (Note 5)				8		

Notes: 1. Typical limits are at $V_{CC} = 5.0V$ and $T_A = 25°C$.
2. This applies to three-state devices only.
3. These are absolute voltages with respect to device ground pin and include all overshoots due to system and/or tester noise. Do not attempt to test these values without suitable equipment.
4. Not more than one output should be shorted at a time. Duration of the short circuit should not be more than one second.
5. These parameters are not 100% tested, but are periodically sampled.

How to Design with PALs

How to Design with PALs

Since any Boolean function can be expressed in the sum-of-products form, any logic function can be implemented in a PAL as long as the number of inputs, outputs and product terms required to perform the function do not exceed what is available in the device. There are many ways of deriving this sum-of-products form, one common way is by the use of Karnaugh maps (K-Maps). When implementing functions in PALs with active HIGH outputs (AND-OR), the usual method of grouping the "1"s will produce the desired equations (see Figure 1a). However, when using PALs with active LOW outputs (AND-OR-INVERT), the sum-of-products equations are obtained by grouping the "0"s (Figure 1b). Grouping the "0"s instead of the "1"s has the effect of inverting the equations. This is a convenient technique for generating inverted logic for use with active LOW PALs.

When using PALs, the term "product term" is often used. This is simply another way of describing an AND function. Referring again to Figure 1a, since there are 3 logical AND terms in the function equation, we would say that it would require 3 product terms to perform that function. The phrase "product term" is often abbreviated as "PT".

A demonstration of an early methodology used to relate logic equations to the internal structure of PALs is shown in the design example. This is a technique which relies on the hand generation of fuse maps. As you might imagine, it is slow and extremely cumbersome, but is included for the interested reader. Instead, to aid in the development of PAL designs, a software tool called PALASM has been developed. This tool greatly speeds the development of PAL designs. AMD offers a version of this called AMPALASM20, which supports all AMD 20-pin PALs, and offers improved ease of use and error detection. AMPALASM20 has been used in most of the examples in this section. Its use is described more fully in Section 4 of this handbook.

$$f = C \cdot \overline{D} + A \cdot \overline{D} + B \cdot C$$

03862A-37

(a)

$$f' = \overline{A} \cdot \overline{C} + \overline{C} \cdot D + \overline{B} \cdot D$$

03862A-38

(b)
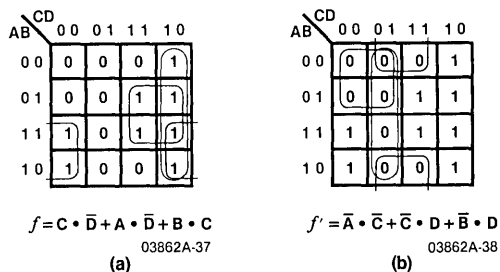
Figure 1. Using K-Maps for Obtaining the
Sum-of-Products Form of PAL Logic

INTRODUCTION TO FUSE MAPS AND DESIGN EXAMPLES

Prior to the availability of PALASM as a tool to aid in the development of PAL designs, all designs were generated by hand using a technique which depended upon fuse maps. With this technique the designer was required to actually identify the individual fuses which needed to be blown to implement the desired function. While this technique is no longer used, it is still useful, as an understanding of it provides a firm foundation in physical reality of how a PAL implements a function.

As a demonstration of this methodology, we will use the arbitrary circuit shown in Figure 2 as an example for selecting and coding a PAL. This example shows the technique of using a PAL once the logic diagram has been developed. Again, this technique is no longer used, as it is much easier to simply generate the equations directly as input to PALASM.
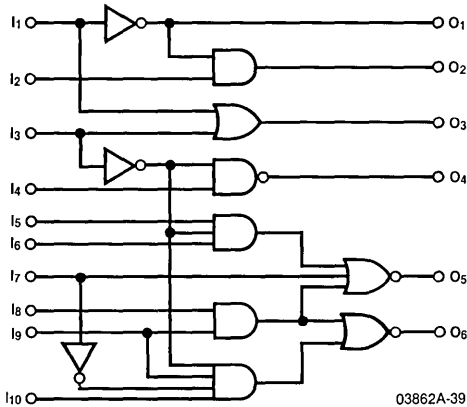
Figure 2. Design Example, Logic Diagram

The first step is to generate the Boolean equations for this function. These are derived directly from the logic diagram. (It should be noted that good design practice is to use meaningful pin names rather than I_1 or O_6. This will help in providing good design documentation.)

The following symbols will be used for all logic equations:

$$* = \text{AND} \quad + = \text{OR} \quad / = \text{NOT (invert)}$$

Boolean equations for Figure 2 are:

$$O1 = /I1$$

$$O2 = /I1*I2$$

$$O3 = I1 + I3$$

$$O4 = /(/I3*I4)$$

$$O5 = /(/I3*I5*I6 + I7 + I8*I9)$$

$$O6 = /(I8*I9 + /I3*/I7*I9*I10)$$

The next step is to select the particular PAL we want to use for this function. Since no registers are required, we should select from the combinatorial devices (AmPAL16L8, AmPAL16H8, AmPAL16LD8 or AmPAL16HD8). Since 3 outputs have AND-OR functions and 3 outputs have AND-OR-INVERT functions, we could still select from either active HIGH or active LOW (H or L) parts, but since the more complex functions are AND-OR-INVERT, the active LOW (L) series is most likely. Finally, we see that no output enable is required, thus we could use either the AmPAL16L8 or the AmPAL16LD8. For the purposes of this example we will select the AmPAL16L8.

Now, since we have selected an AmPAL16L8 (which has inverting outputs) we need to apply DeMorgan's theorem to convert these equations from the active HIGH to the active LOW output. DeMorgan's theorem can be used to convert any logic equation in any form into the AND-OR structure used in PALs. Applying DeMorgan's theorem gives the active LOW form of the equation:

$$/O1 = I1$$

$$/O2 = I1 + /I2$$

$$/O3 = /I1*/I3$$

$$/O4 = /I3*I4$$

$$/O5 = /I3*I5*I6 + I7 + I8*I9$$

$$/O6 = I8*I9 + /I3*/I7*I9*I10$$

We can now determine which fuses need to be programmed for the PAL to perform this function. Figure 3 shows the conventions which are used when coding fuse maps.

Figure 4 shows the logic diagram of the AmPAL16L8. We will assign outputs O_1–O_6 to pins 14–19, and inputs I_1–I_{10} to pins 2–9, 11 and 13.

O_1 is assigned to pin 19. To make this output the inverse of I_1, leave input line 0 connected (not blown) to product term 1 and blow all the remaining fuses on that product term. This is indicated by the X at the intersection of input line 0 and product term 1 in Figure 4.

Since the other inputs to the OR gate are unused, they are forced to zero by leaving all the fuses intact on product terms

2–7. As shown in Figure 3, unused product terms (those with all fuses intact) are indicated by Xs in the small AND gates at the NOR gate inputs.

The final consideration for O_1 is the output enable. By referring back to either the logic diagram for the circuit (Figure 2) or the Boolean equation, we see that there is no output enable function for O_1. In other words, O_1 should be enabled all the time. Referring to Figure 4, we see that product term 0 controls the output enable function for O_1. To have the output always enabled, we blow all the fuses in product term (PT) 0. A product term with all fuses blown is always HIGH, so this will leave the three-state gate always on, and the output will always be enabled.
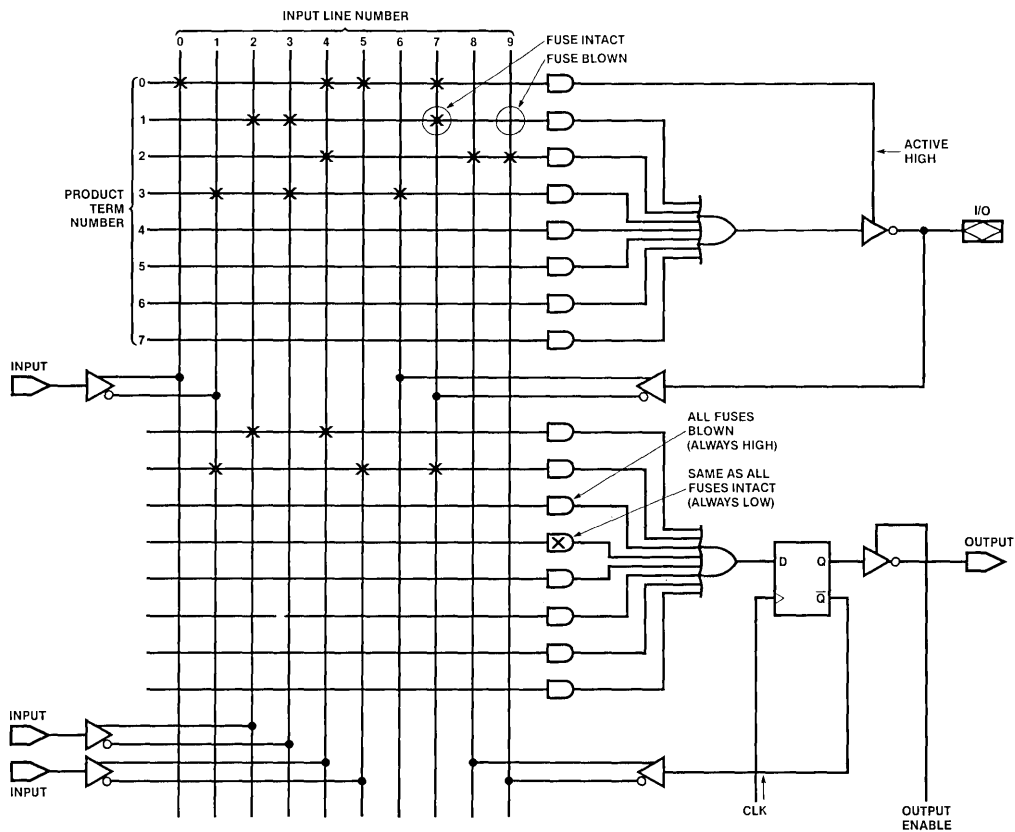
03862A-40

Symbols

(L)	Low
(H)	High
(N)	Negative
(P)	Positive
(0)	Zero
(1)	One
✳	Fuse intact
┼	Fuse blown

Figure 3. Coding Conventions
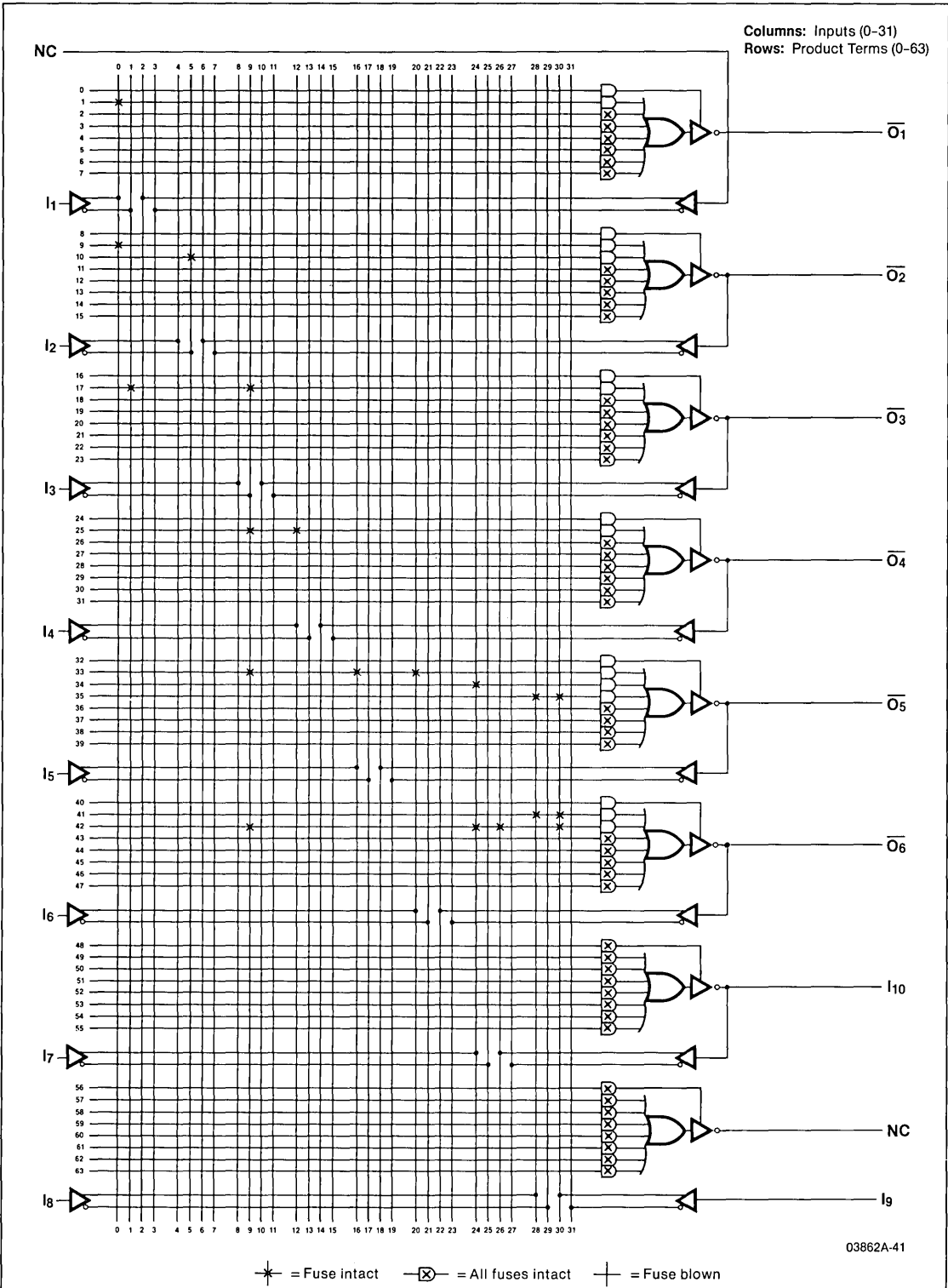
Figure 4. Logic Diagram for Function of Figure 1 Using the AmPAL16L8

The next output, O_2, is the AND function of I_1 and I_2. Again, since we are using an "L" device, we must apply DeMorgan's theorem and use the inverted form. Since we want to have $O_2 = I_1 + \overline{I_2}$, we first leave input line 0 (I_1) connected to PT 9 and blow the rest of the fuses on that product term. Then since we have an OR ($+$) function, we go to the next product term, and leave input line 5 ($\overline{I_2}$) connected to PT 10.

Since PT 11–15 will be unused, we indicate this as we did in O_1, by putting a X in the AND gate at the input to the NOR gate. Also, since we want O_2 to be always enabled, we leave PT 8 blank, indicating that all of the fuses in that product term should be programmed.

Output O_3 is the AND of $\overline{I_1}$ and $\overline{I_3}$. To implement this, we leave input line 1 connected to PT 17. Since we want an AND function, we also leave input line 9 ($\overline{I_3}$) connected to PT 17. These connections are indicated by Xs. We then blow the rest of the fuses in PT 17. Since the rest of the product terms are unused, we place an X in the AND gates for PT 18–23. Again, we leave PT 16 blank, which will leave O_3 always enabled.

Output O_4 is very similar to O_3. In order to generate this AND function, we leave input line 9 ($\overline{I_3}$) and input line 12 (I_4) connected to PT 25, and program the remainder of the fuses in PT

25. We again indicate that the rest of the product terms are unused, and the output is always enabled.

Output O_5 is generated by ANDing $\overline{I_3}$, I_5, and I_6 on PT 33, connecting I_7 to PT 34, ANDing I_8 and I_9 on PT 35, and leaving PT 36–39 unused.

Output O_6 is generated by ANDing I_8 and I_9 on PT 41 and ANDing $\overline{I_3}$, $\overline{I_7}$, I_9 and I_{10} on PT 42. Product terms 43–47 are left unused.

Since pins 12 and 13 are not being used as outputs, Xs are put in the AND gates for all of those product terms.

The completed fuse map is shown in Figure 4.

As you can see, any function can be put into the sum-of-products form and then used to generate a fuse map. However, it can be very time consuming to generate these maps by hand. Therefore, AMD has developed a software tool called AMPALASM20 which will automatically generate the fuse map from the Boolean equations. This software tool is described in Section 4, but for comparison purposes, Figures 5 and 6 show abbreviated input and output data for AMPALASM20.

```
PAL16L8                          PAL DESIGN SPECIFICATION
PAT001                           MITCH RICHMAN  4/7/83
DESIGN EXAMPLE
ADVANCED MICRO DEVICES
NC   I1  I2  I3  I4  I5  I6  I7  I8  GND
I9  NC  I10 O6  O5  O4  O3  O2  O1  VCC
;
;
/O1 = I1

/O2 = I1             +
      /I2

/O3 = /I1*/I3

/O4 = /I3*I4

/O5 = /I3*I5*I6      +
      I7             +
      I8*I9

/O6 = I8*I9          +
      /I3*/I7*I9*I10
```

03862A-42

Figure 5. Abbreviated AMPALASM20 Input

```
DESIGN EXAMPLE

                 11 1111 1111 2222 2222 2233
      0123 4567 8901 2345 6789 0123 4567 8901

    0 ---- ---- ---- ---- ---- ---- ---- ----
    1 X--- ---- ---- --.- ---- ---- ---- ---- I1
    2 XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
    3 XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
    4 XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
    5 XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
    6 XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
    7 XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

    8 ---- ---- ---- ---- ---- ---- ---- ----
    9 X--- ---- ---- ---- ---- ---- ---- ---- I1
   10 ---- -X-- ---- ---- ---- ---- ---- ---- /I2
   11 XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
   12 XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
   13 XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
   14 XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
   15 XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

   16 ---- ---- ---- ---- ---- ---- ---- ----
   17 -X-- ---- -X-- ---- ---- ---- ---- ---- /I1*/I3
   18 XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
   19 XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
   20 XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
   21 XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
   22 XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
   23 XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

   24 ---- ---- ---- ---- ---- ---- ---- ----
   25 ---- ---- -X-- X--- ---- ---- ---- ---- /I3*I4
   26 XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
   27 XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
   28 XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
   29 XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
   30 XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
   31 XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

   32 ---- ---- ---- ---- ---- ---- ---- ----
   33 ---- ---- -X-- ---- X--- X--- ---- ---- /I3*I5*I6
   34 ---- ---- ---- ---- ---- ---- X--- ---- I7
   35 ---- ---- ---- ---- ---- ---- ---- X-X- I8*I9
   36 XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
   37 XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
   38 XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
   39 XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

   40 ---- ---- ---- ---- ---- ---- ---- ----
   41 ---- ---- ---- ---- ---- ---- ---- X-X- I8*I9
   42 ---- ---- -X-- ---- ---- ---- -XX- --X- /I3*/I7*I9*I10
   43 XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
   44 XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
   45 XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
   46 XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
   47 XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

   48 XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
   49 XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
   50 XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
   51 XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
   52 XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
   53 XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
   54 XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
   55 XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

   56 XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
   57 XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
   58 XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
   59 XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
   60 XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
   61 XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
   62 XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
   63 XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX


LEGEND:  X : FUSE NOT BLOWN (L,N,0)   - : FUSE BLOWN   (H,P,1)

NUMBER OF FUSES BLOWN =  493
```

03862A-43

Figure 6. AMPALASM20 Output Fuse Map

Complex Functions

Most complex functions can be built up from smaller functional building blocks that are easy to understand and design. This section will show some techniques for designing several common functional blocks that can be tailored for specific applications and used to build up the desired PAL designs. Each block includes a PAL DESIGN SPECIFICATION, a compiled AMPALASM20 output, and a fuse map of the desired function.

Control Functions

An important feature of PALs with registers is the availability of registered outputs as inputs to the programmable array. This feedback is often used to configure PALs into state sequencers. Feedback also gives an easy way to create a device with a "clock enable". The purpose of disabling the clock is to prevent the contents of the output registers from changing (retain the same state). This result is easily obtained by feeding back the contents of the registers and having these values be clocked back in. In this way, the clock is not actually disabled, but the desired effect is achieved.

Another feature of PALs is programmable I/O pins. One product term controls a three-state driver whose output is fed-back as an array input as well as connected to a pin. Thus, if the three-state driver is disabled (high impedance state), the pin can be used as an input. This makes these pins perfect for use as bidirectional lines for such purposes as shifting data serially.

Several common "special" functions, such as clearing and setting, can be incorporated into PALs with ease. The clear function is performed by disabling all of the AND-gates, causing the output of the register to go LOW. On the other hand, the set function is performed by forcing one product term HIGH, causing the output of the register to go HIGH. Loading the registers with data is another common function and, for each bit, can be accomplished by selecting one AND-gate to pass the input data to the registers while disabling the remaining AND-gates. In general, a control or logic function is performed by selecting the necessary AND-gates and deselecting all others unused by the function.

EXCLUSIVE-OR

An Exclusive-OR (XOR) is often used as a selective inverter or digital comparator. The XOR (2-input) performs the following function: if either of the inputs is HIGH, but not both, then the output is HIGH. The function table and logic equation of the 2-input XOR appearing in Figure 7 shows this graphically. The logic equation can be derived through the use of a Karnaugh map if desired. The logic diagram and logic symbol for the 2-input XOR appear in Figure 8.

XORs of more than two inputs are often used for Modulo-2 arithmetic and odd parity generation. Intuitively, the multiple input XOR is HIGH when an odd number of inputs is HIGH. Figure 9 shows the Karnaugh map for a 4-input XOR. Notice that due to the diagonal pattern of one's in the map, there is no way to combine terms. Also note that exactly half of the squares in the map are ones. Since an n-input function results in 2^n squares in a Karnaugh map, an n-input XOR will require $2^{(n-1)}$ product terms. For example, the 4-input XOR of Figure 7 requires eight product terms (see Figure 10).

INPUTS A B	OUTPUT Y
0 0	0
0 1	1
1 0	1
1 1	0

$$Y = A \cdot \overline{B} + \overline{A} \cdot B$$

03862A-44

Figure 7. XOR Function Table and Logic Equation
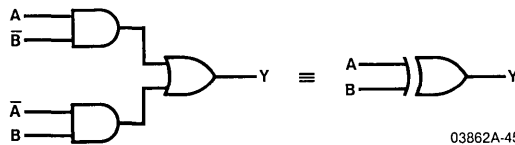
(a) XOR Logic and (b) Symbolic Representation

Figure 8. XOR Logic Diagram and Logic Symbol
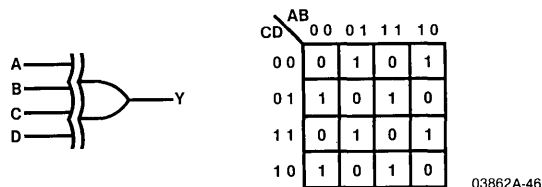
Figure 9. Karnaugh Map and Logic Symbol for a 4-Input XOR

INPUTS						OUTPUT
S_1	S_0	D_3	D_2	D_1	D_0	Y
0	0	X	X	X	0	0
0	0	X	X	X	1	1
0	1	X	X	0	X	0
0	1	X	X	1	X	1
1	0	X	0	X	X	0
1	0	X	1	X	X	1
1	1	0	X	X	X	0
1	1	1	X	X	X	1

Figure 11. Function Table for 1-of-4 Multiplexer
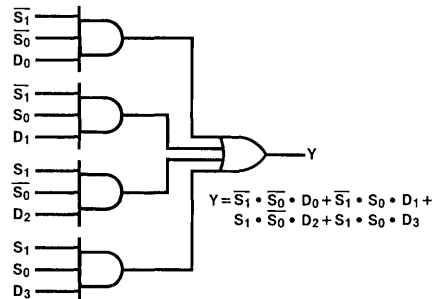
$$Y = \overline{S_1} \cdot \overline{S_0} \cdot D_0 + \overline{S_1} \cdot S_0 \cdot D_1 + S_1 \cdot \overline{S_0} \cdot D_2 + S_1 \cdot S_0 \cdot D_3$$

$$Y = \overline{A} \cdot \overline{B} \cdot \overline{C} \cdot D + \overline{A} \cdot \overline{B} \cdot C \cdot \overline{D} + \overline{A} \cdot B \cdot \overline{C} \cdot \overline{D} + A \cdot \overline{B} \cdot \overline{C} \cdot \overline{D} +$$
$$\overline{A} \cdot B \cdot C \cdot D + A \cdot \overline{B} \cdot C \cdot D + A \cdot B \cdot \overline{C} \cdot D + A \cdot B \cdot C \cdot \overline{D}$$

Figure 10. 4-Input XOR Logic

Figure 12. Multiplexer Logic

THE MULTIPLEXER

The multiplexer (also called a data selector) is used to selectively route data from several inputs to one output. A simple 4-to-1 multiplexer has four data input lines and two control lines that select which one of the four data inputs is to be passed to the output. The function table for this device is shown in Figure 11. The logic equation can be derived directly from the function table (see Figure 12).

Each AND-gate has a data input and a two-input combination of select inputs. Given one of four possible combinations of select bits, only the AND-gate corresponding to this combination is enabled, allowing the desired input data to pass to the output. This can be easily expanded to accommodate more data simply by adding more select lines and data inputs. For every n select lines there can be 2^n data inputs, each of which requires one product term.

The Design Specification and Logic Diagram for this Exclusive-OR and multiplexer based on an AmPAL16H8 are shown in Tables 1(a) and 1(b).

Table 1(a). Design Specification for XOR and MUX Function

```
PAL16H8                                  PAL DESIGN SPECIFICATION
PAT02O                                   JENNY YEE 10/22/82
XOR AND MUX FUNCTION
ADVANCED MICRO DEVICES
XA  XB  XC  DO  D1  D2  D3 S1 SO GND
NC NC NC NC NC NC NC  MUXY  XORY VCC


;
;XOR AND MUX OUTPUT SIGNALS
;

  XORY    = /XA*/XB* XC     +
            /XA* XB*/XC     +
            XA*/XB*/XC      +
            XA* XB* XC

  MUXY    = /S1*/SO*DO      +
            /S1* SO*D1      +
            S1*/SO*D2       +
            S1* SO*D3


FUNCTION TABLE

S1 SO  XA  XB  XC D3 D2 D1 DO           XORY  MUXY
--------------------------------------------------------------------
;ACTIVATE XOR OUTPUT
;
X  X   L   L   L  X  X   X  X           L     X
X  X   L   L   H  X  X   X  X           H     X
X  X   L   H   L  X  X   X  X           H     X
X  X   L   H   H  X  X   X  X           L     X
X  X   H   L   L  X  X   X  X           H     X
X  X   H   L   H  X  X   X  X           L     X
X  X   H   H   H  X  X   X  X           H     X
;
;ACTIVATE MUX OUTPUT
;
L  L   X   X   X  X  X   X  L           X     L
L  L   X   X   X  X  X   X  H           X     H
L  H   X   X   X  X  X   L  X           X     L
L  H   X   X   X  X  X   H  X           X     H
H  L   X   X   X  X  L   X  X           X     L
H  L   X   X   X  X  H   X  X           X     H
H  H   X   X   X  L  X   X  X           X     L
H  H   X   X   X  H  X   X  X           X     H
--------------------------------------------------------------------

DESCRIPTION:  THIS IS A SIMPLE EXAMPLE EXECUTING THE FUNCTIONS OF
THE EXCLUSIVE-OR AND THE MULTIPLEXER                   03862A-50
```

Note: See Section 4 for description of the **PAL DESIGN SPECIFICATION** format.

Table 1(a). Design Specification for XOR and MUX Function (Continued)

```
PAL16H8                                      PAL DESIGN SPECIFICATION
PAT020                                       JENNY YEE 10/22/82
XOR AND MUX FUNCTION
ADVANCED MICRO DEVICES
*D9725
*F0*
L0000 1111 1111 1111 1111 1111 1111 1111 1111 *
L0032 1010 0111 1111 1111 1111 1111 1111 1111 *
L0064 0110 1011 1111 1111 1111 1111 1111 1111 *
L0096 1001 1011 1111 1111 1111 1111 1111 1111 *
L0128 0101 0111 1111 1111 1111 1111 1111 1111 *
L0256 1111 1111 1111 1111 1111 1111 1111 1111 *
L0288 1111 1111 0111 1111 1111 1111 1011 1011 *
L0320 1111 1111 1111 0111 1111 1111 1011 0111 *
L0352 1111 1111 1111 1111 0111 1111 0111 1011 *
L0384 1111 1111 1111 1111 1111 0111 0111 0111 *
C26D2*
V0001 000XXXXXX0XXXXXXXXXL1 *
V0002 001XXXXXX0XXXXXXXXXH1 *
V0003 010XXXXXX0XXXXXXXXXH1 *
V0004 011XXXXXX0XXXXXXXXXL1 *
V0005 100XXXXXX0XXXXXXXXXH1 *
V0006 101XXXXXX0XXXXXXXXXL1 *
V0007 111XXXXXX0XXXXXXXXXH1 *
V0008 XXX0XXX000XXXXXXXXXLX1 *
V0009 XXX1XXX000XXXXXXXXXHX1 *
V0010 XXXX0XX010XXXXXXXXXLX1 *
V0011 XXXX1XX010XXXXXXXXXHX1 *
V0012 XXXXX0X100XXXXXXXXXLX1 *
V0013 XXXXX1X100XXXXXXXXXHX1 *
V0014 XXXXXX0110XXXXXXXXXLX1 *
V0015 XXXXXX1110XXXXXXXXXHX1 *
0221
```

Table 1(b). Logic Diagram for XOR and Multiplexer Using AmPAL16H8

Columns: Inputs (0–31)
Rows: Product Terms (0–63)

XA

XORY

XB

MUXY

XC

NC

D₀

NC

D₁

NC

D₂

NC

D₃

NC

S₁

NC

S₀

NC

03862A-52

—✱— = Fuse intact   —⊠— = All fuses intact   —┼— = Fuse blown

3-11
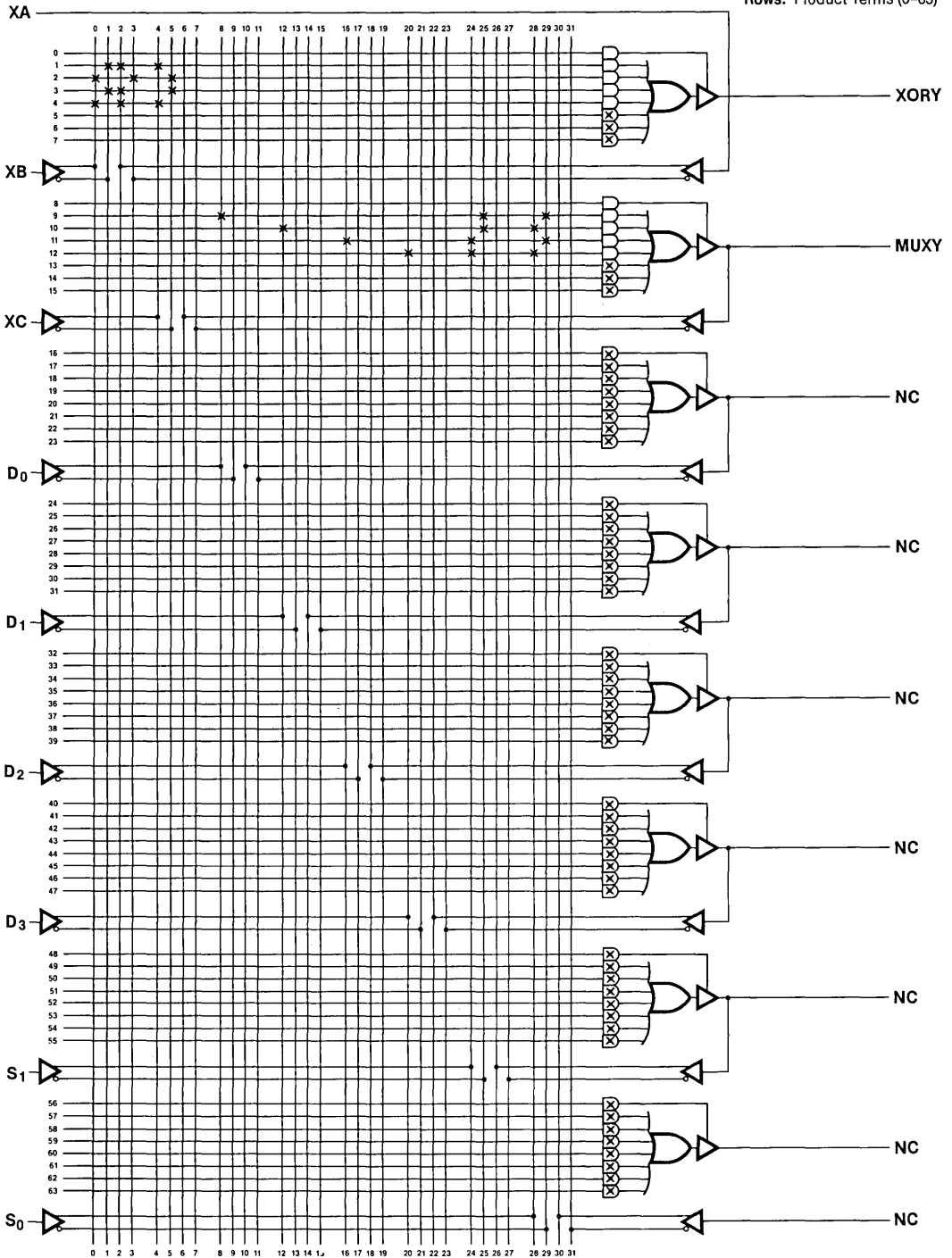
DECODING/CHIP SELECT

Decoding is one of the most common logic functions to be performed in a design. Essentially all random logic control signals are generated by decoding. Two examples are output enabling and chip selection. The decoding function can be described as being true when the desired set of inputs is true. This, of course, is simply the AND function. In its most general form, an n-input decoder can have 2^n decoded outputs implemented by 2^n AND-gates. Each AND-gate has a different combination of inputs allowing only one output to be true at any given time. If Karnaugh maps were used to describe an n-input decoder, there would be 2^n maps with only a single one in each one. Figure 13 shows the truth table, logic equations and logic diagram for a 3-to-8 decoder imple-

mented with an AmPAL16H8. The design specification and logic diagram for this decoder are shown in Tables 2(a) and 2(b).

In most applications, three inputs (such as in a 3-to-8 decoder), are insufficient to perform the entire decode. For example, decoding the address and control for an I/O device requires anywhere from 8 to 16 address control and timing inputs. Besides the address lines, the control and timing signals which are commonly used as inputs for decoding are: READ/WRITE (R/W), MEMORY/IO (M/IO), BYTE/WORD (B/W) and ADDRESS STROBE (AS). Figure 14 shows typical input and output ports with required decoders.

INPUTS			OUTPUTS							
S_2	S_1	S_0	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
0	0	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	1	0
0	1	0	0	0	0	0	0	1	0	0
0	1	1	0	0	0	0	1	0	0	0
1	0	0	0	0	0	1	0	0	0	0
1	0	1	0	0	1	0	0	0	0	0
1	1	0	0	1	0	0	0	0	0	0
1	1	1	1	0	0	0	0	0	0	0

03862A-53

(a) Truth Table 3-to-8 Decoder

$$D_0 = \overline{S_2} \cdot \overline{S_1} \cdot \overline{S_0}$$
$$D_1 = \overline{S_2} \cdot \overline{S_1} \cdot S_0$$
$$D_2 = \overline{S_2} \cdot S_1 \cdot \overline{S_0}$$
$$D_3 = \overline{S_2} \cdot S_1 \cdot S_0$$
$$D_4 = S_2 \cdot \overline{S_1} \cdot \overline{S_0}$$
$$D_5 = S_2 \cdot \overline{S_1} \cdot S_0$$
$$D_6 = S_2 \cdot S_1 \cdot \overline{S_0}$$
$$D_7 = S_2 \cdot S_1 \cdot S_0$$

03862A-54

(b) Equations 3-to-8 Decoder
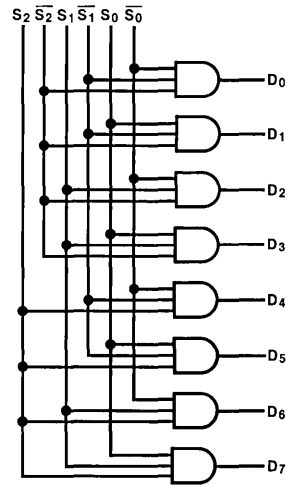
03862A-55

Figure 13

(c) Logic Diagram 3-to-8 Decoder
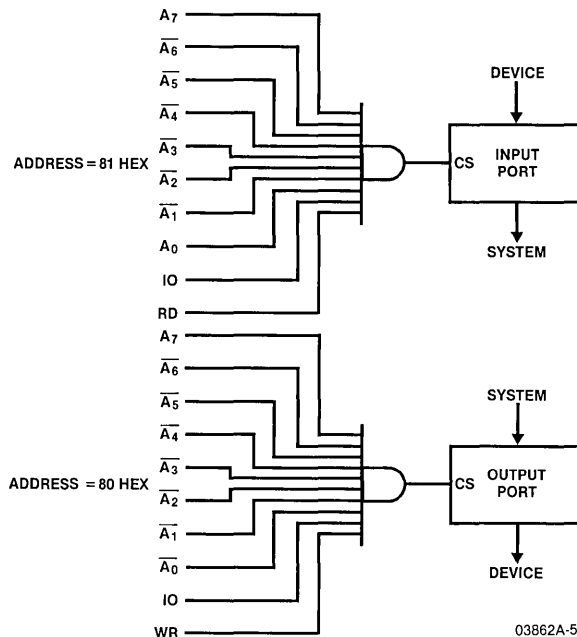
03862A-56

Figure 14. Typical Input and Output Ports

Table 2(a). Design Specification for 3-to-8 Decoder

```
PAL16H8                                    PAL DESIGN SPECIFICATION
PAT022                                     JENNY YEE   10/22/82
3 TO 8 DECODER
ADVANCED MICRO DEVICES
S2  S1  S0  NC  NC  NC  NC  NC  NC GND
NC  D0  D1  D2  D3  D4  D5  D6  D7 VCC
;
;DECODER OUTPUT SIGNALS
;
 D0        = /S2*/S1*/S0
 D1        = /S2*/S1* S0
 D2        = /S2* S1*/S0
 D3        = /S2* S1* S0
 D4        =  S2*/S1*/S0
 D5        =  S2*/S1* S0
 D6        =  S2* S1*/S0
 D7        =  S2* S1* S0

FUNCTION TABLE
S2  S1  S0            D7  D6  D5  D4  D3  D2  D1  D0
-----------------------------------------------------------------------
L   L   L             L   L   L   L   L   L   L   H
L   L   H             L   L   L   L   L   L   H   L
L   H   L             L   L   L   L   L   H   L   L
L   H   H             L   L   L   L   H   L   L   L
H   L   L             L   L   L   H   L   L   L   L
H   L   H             L   L   H   L   L   L   L   L
H   H   L             L   H   L   L   L   L   L   L
H   H   H             H   L   L   L   L   L   L   L
-----------------------------------------------------------------------
DESCRIPTION
THIS DEVICE IMPLEMENTS A 3 TO 8 DECODER. THE SAMPLE SHOWS THE
DESIGN OF THE DECODER USING  PAL.                        03862A-57
```

Table 2(a). Design Specification for 3-to-8 Decoder (Continued)

```
PAL16H8                                    PAL DESIGN SPECIFICATION
PAT022                                     JENNY YEE   10/22/82
3 TO 8 DECODER
ADVANCED MICRO DEVICES
*D9725
*F0*
L0000 1111 1111 1111 1111 1111 1111 1111 1111 *
L0032 0101 0111 1111 1111 1111 1111 1111 1111 *
L0256 1111 1111 1111 1111 1111 1111 1111 1111 *
L0288 0101 1011 1111 1111 1111 1111 1111 1111 *
L0512 1111 1111 1111 1111 1111 1111 1111 1111 *
L0544 1001 0111 1111 1111 1111 1111 1111 1111 *
L0768 1111 1111 1111 1111 1111 1111 1111 1111 *
L0800 1001 1011 1111 1111 1111 1111 1111 1111 *
L1024 1111 1111 1111 1111 1111 1111 1111 1111 *
L1056 0110 0111 1111 1111 1111 1111 1111 1111 *
L1280 1111 1111 1111 1111 1111 1111 1111 1111 *
L1312 0110 1011 1111 1111 1111 1111 1111 1111 *
L1536 1111 1111 1111 1111 1111 1111 1111 1111 *
L1568 1010 0111 1111 1111 1111 1111 1111 1111 *
L1792 1111 1111 1111 1111 1111 1111 1111 1111 *
L1824 1010 1011 1111 1111 1111 1111 1111 1111 *
C3EC4*
V0001 000XXXXXX0XHLLLLLLL1 *
V0002 001XXXXXX0XLHLLLLLL1 *
V0003 010XXXXXX0XLLHLLLLL1 *
V0004 011XXXXXX0XLLLHLLLL1 *
V0005 100XXXXXX0XLLLLHLLL1 *
V0006 101XXXXXX0XLLLLLHLL1 *
V0007 110XXXXXX0XLLLLLLHL1 *
V0008 111XXXXXX0XLLLLLLLH1 *
FC30                                               03862A-58
```

Table 2(b). Logic Diagram for 3-to-8 Decoder Using AmPAL16H8

Columns: Inputs (0–31)
Rows: Product Terms (0–63)

S_2

S_1

S_0

NC

NC

NC

NC

NC

NC

D_7

D_6

D_5

D_4

D_3

D_2

D_1

D_0

NC

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

—✶— = Fuse intact   —⊠— = All fuses intact   —┼— = Fuse blown
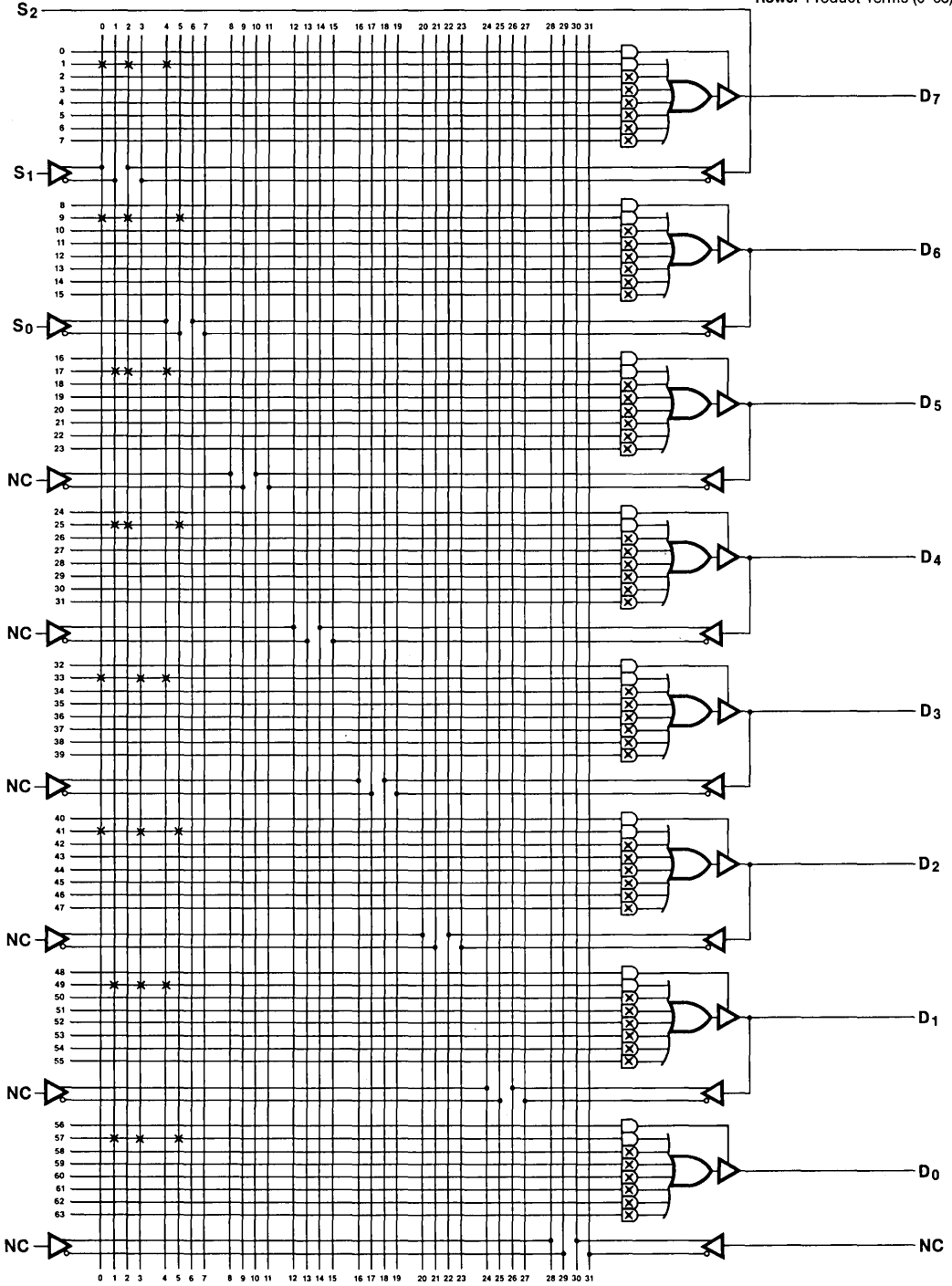
03862A-59

SHIFT REGISTERS

Shift registers are useful for multiplication, division and serial communications.

An easy way to construct a shift register is to think of it as being composed of a set of multiplexers with registered outputs. A typical shift register can load data, shift data to the right, shift data to the left, and "hold" (leave unchanged) the data. There would be two select lines that control the multiplexers and choose the desired function. If data is to be loaded into the shift register, this path is enabled and the data is passed through and clocked into the registers (see Figure 15a). Figures 15b and 15c illustrate how the shift functions are performed. When shifting right, bit i will clock in bit i + 1. This is accomplished by feeding back the output of register i + 1 to the multiplexer of bit i and selecting this path. Similarly, shifting left is done by clocking bit i − 1 into bit i,

using similar feedback techniques. "Holding" data is easily done by feeding back register i to itself and selecting the path, as shown in Figure 15d. Thus keeping data unchanged is accomplished by feedback, **not** by gating the clock (which is a poor design practice).

This information is now used to design a 4-bit shift register using an AmPAL16H8. The function has four data inputs, four outputs, and two select lines, a left serial input, a right serial input, an output enable, and a clock input. The serial inputs are bidirectional and are used to input new data while shifting. The function table for the shift register is shown in Figure 16 and the logic diagram in Figure 17. The Design Specification and Logic Diagram for these shifters are shown in Tables 3(a) and 3(b).
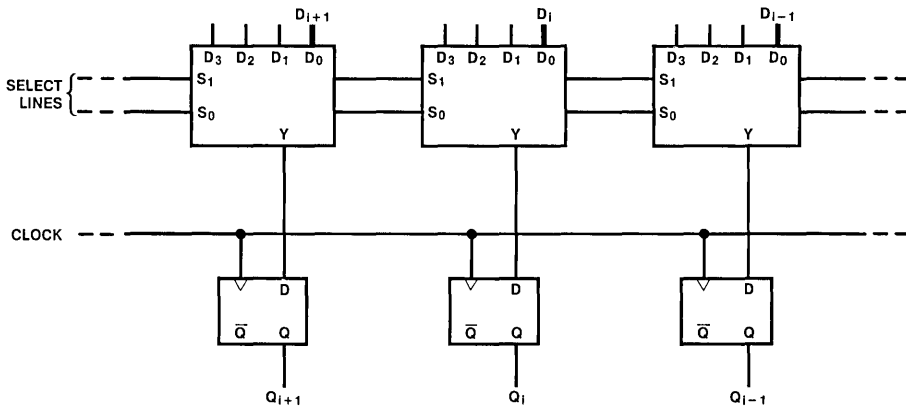
03862A-60

Figure 15a. Loading the Shifter
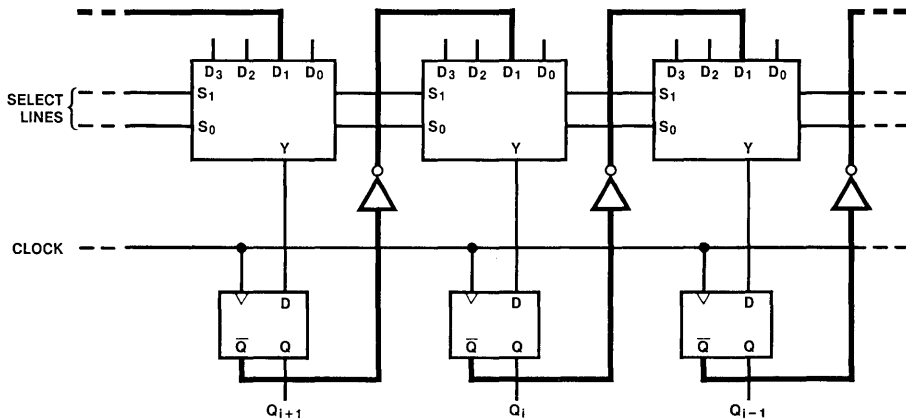
03862A-61

Figure 15b. Shifting Right

3-16
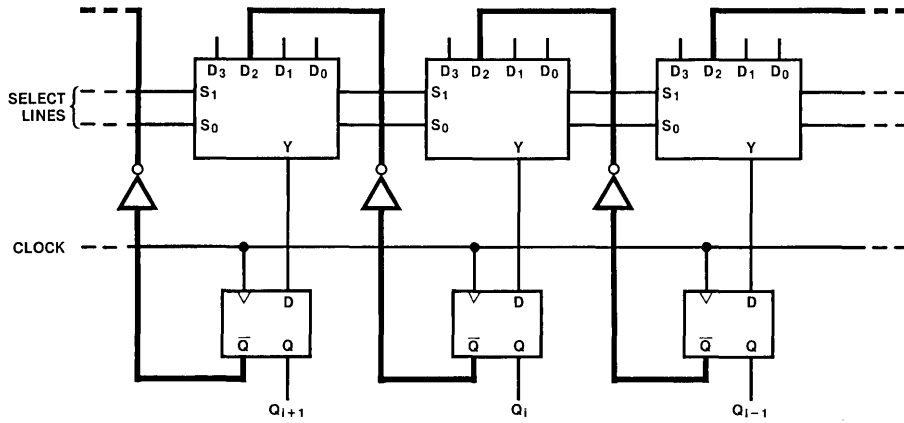
Figure 15c. Shifting Left
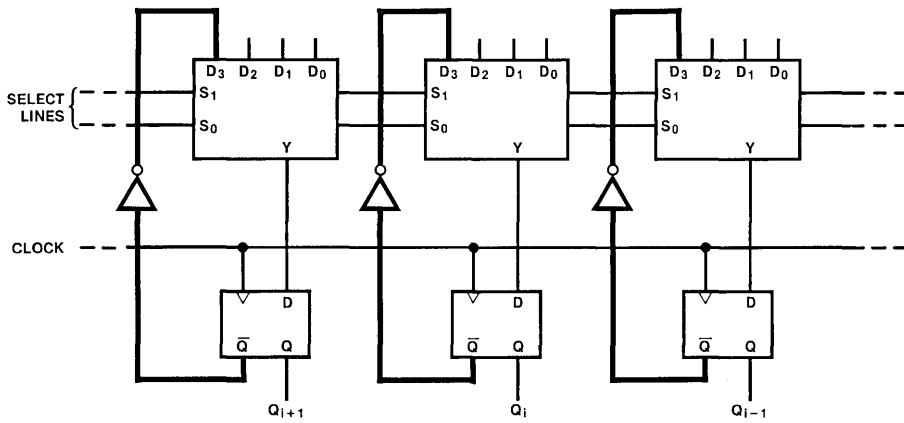
03862A-62

Figure 15d. "Holding" Data

03862A-63

INPUTS				OUTPUTS						
S_1	S_0	SERIAL		Q_3	Q_2	Q_1	Q_0	SERIAL	
		LEFT	RIGHT					RIGHT	LEFT
0	0	X	X	D_3	D_2	D_1	D_0	Z	Z
0	1	X	0	0	$Q3_0$	$Q2_0$	$Q1_0$	$Q0_0$	Z
0	1	X	1	1	$Q3_0$	$Q2_0$	$Q1_0$	$Q0_0$	Z
1	0	0	X	$Q2_0$	$Q1_0$	$Q0_0$	0	Z	$Q3_0$
1	0	1	X	$Q2_0$	$Q1_0$	$Q0_0$	1	Z	$Q3_0$
1	1	X	X	$Q3_0$	$Q2_0$	$Q1_0$	$Q0_0$	Z	Z

03862A-64

Figure 16. Shift Register Function Table
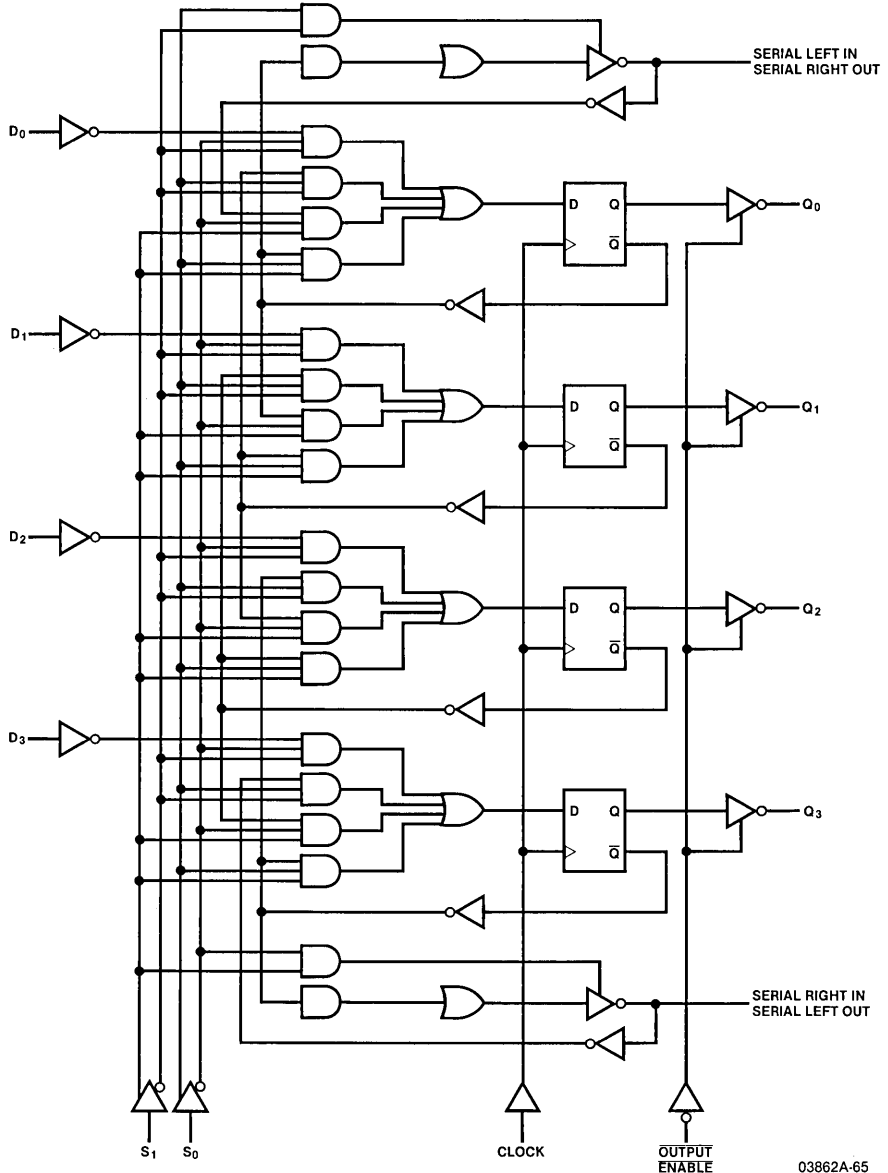
Figure 17. 4-Bit Bidirectional Shift Register

03862A-65

```
PAL16R6                                    PAL DESIGN SPECIFICATION
PAT023                                     JENNY YEE    10/22/82
SHIFT REGISTER
ADVANCED MICRO DEVICES
CK   S1  S0    D3    D2   D1   D0   NC  NC  GND
OE   SRISLO  NC  Q3  Q2   Q1   Q0 NC  SLISRO VCC
;
;SHIFT REGISTER OUTPUT SIGNALS
;
/Q3   := /S1*/S0*/D3           +
         /S1* S0*/SRISLO       +
          S1*/S0*/Q2           +
          S1* S0*/Q3

/Q2   := /S1*/S0*/D2           +
         /S1* S0*/Q3           +
          S1*/S0*/Q1           +
          S1* S0*/Q2

/Q1   := /S1*/S0*/D1           +
         /S1* S0*/Q2           +
          S1*/S0*/Q0           +
          S1* S0*/Q1

/Q0   := /S1*/S0*/D0           +
         /S1* S0*/Q1           +
          S1*/SLISRO*/S0       +
          S1* S0*/Q0

IF(/S1*S0) /SLISRO = /Q0


IF(S1*/S0) /SRISLO = /Q3                    03862A-66
```

Table 3(a). Design Specification for Shift Register (Continued)

```
FUNCTION TABLE
CK  S1 SO D3 D2 D1 DO OE       SRISLO  SLISRO    Q3   Q2  Q1  QO
----------------------------------------------------------------------
;
; LOAD AND SHIFT RIGHT
C   L  L  L  L  L  L  L         Z       Z         L    L   L   L
C   H  H  X  X  X  X  L         Z       Z         L    L   L   L
C   L  H  X  X  X  X  L         H       L         H    L   L   L
C   L  H  X  X  X  X  L         L       L         L    H   L   L
C   L  H  X  X  X  X  L         L       L         L    L   H   L
C   L  H  X  X  X  X  L         L       H         L    L   L   H
C   L  H  X  X  X  X  L         L       L         L    L   L   L
;
; LOAD AND SHIFT LEFT
C   L  L  H  H  H  H  L         Z       Z         H    H   H   H
C   H  H  X  X  X  X  L         Z       Z         H    H   H   H
C   H  L  X  X  X  X  L         H       L         H    H   H   L
C   H  L  X  X  X  X  L         H       H         H    H   L   H
C   H  L  X  X  X  X  L         H       H         H    L   H   H
C   H  L  X  X  X  X  L         L       H         L    H   H   H
C   H  L  X  X  X  X  L         H       H         H    H   H   H
;
; HOLD
C   H  H  X  X  X  X  L         Z       Z         H    H   H   H
----------------------------------------------------------------------
DESCRIPTION
THIS DEVICE IMPLEMENTS A SHIFT REGISTER.  THE LAYOUT PROVIDED IS
A DEMONSTRATION OF HOW THE SHIFT REGISTER MAY BE DESIGNED USING
A PAL.                                                    03862A-67
```

Table 3(a). Design Specification for Shift Register (Continued)

```
PAL16R6                                    PAL DESIGN SPECIFICATION
PAT023                                     JENNY YEE     10/22/82
SHIFT REGISTER
ADVANCED MICRO DEVICES
*D9724
*FO*
L0000 1011 0111 1111 1111 1111 1111 1111 1111 *
L0032 1111 1111 1110 1111 1111 1111 1111 1111 *
L0512 1011 1011 1111 1111 1111 1011 1111 1111 *
L0544 1011 0111 1111 1110 1111 1111 1111 1111 *
L0576 0110 1011 1111 1111 1111 1111 1111 1111 *
L0608 0111 0111 1110 1111 1111 1111 1111 1111 *
L0768 1011 1011 1111 1111 1011 1111 1111 1111 *
L0800 1011 0111 1111 1111 1110 1111 1111 1111 *
L0832 0111 1011 1110 1111 1111 1111 1111 1111 *
L0864 0111 0111 1111 1110 1111 1111 1111 1111 *
L1024 1011 1011 1111 1011 1111 1111 1111 1111 *
L1056 1011 0111 1111 1111 1111 1110 1111 1111 *
L1088 0111 1011 1111 1110 1111 1111 1111 1111 *
L1120 0111 0111 1111 1111 1110 1111 1111 1111 *
L1280 1011 1011 1011 1111 1111 1111 1111 1111 *
L1312 1011 0111 1111 1111 1111 1111 1111 1110 *
L1344 0111 1011 1111 1111 1110 1111 1111 1111 *
L1376 0111 0111 1111 1111 1111 1110 1111 1111 *
L1792 0111 1011 1111 1111 1111 1111 1111 1111 *
L1824 1111 1111 1111 1111 1111 1110 1111 1111 *
C49E9*
V0001 C000000XX00ZXLLLLXZ1 *
V0002 C11XXXXXX00ZXLLLLXZ1 *
V0003 C01XXXXXX001XHLLLXL1 *
V0004 C01XXXXXX000XLHLLXL1 *
V0005 C01XXXXXX000XLLHLXL1 *
V0006 C01XXXXXX000XLLLHXH1 *
V0007 C01XXXXXX000XLLLLXL1 *
V0008 C001111XX00ZXHHHHXZ1 *
V0009 C11XXXXXX00ZXHHHHXZ1 *
V0010 C10XXXXXX00HXHHHLXO1 *
V0011 C10XXXXXX00HXHHLHX11 *
V0012 C10XXXXXX00HXHLHHX11 *
V0013 C10XXXXXX00LXLHHHX11 *
V0014 C10XXXXXX00HXHHHHX11 *
V0015 C11XXXXXX00ZXHHHHXZ1 *
5060                                                  03862A-68
```

Table 3(b). Logic Diagram for Shift Register Using AmPAL16R6

Columns: Inputs (0–31)
Rows: Product Terms (0–63)
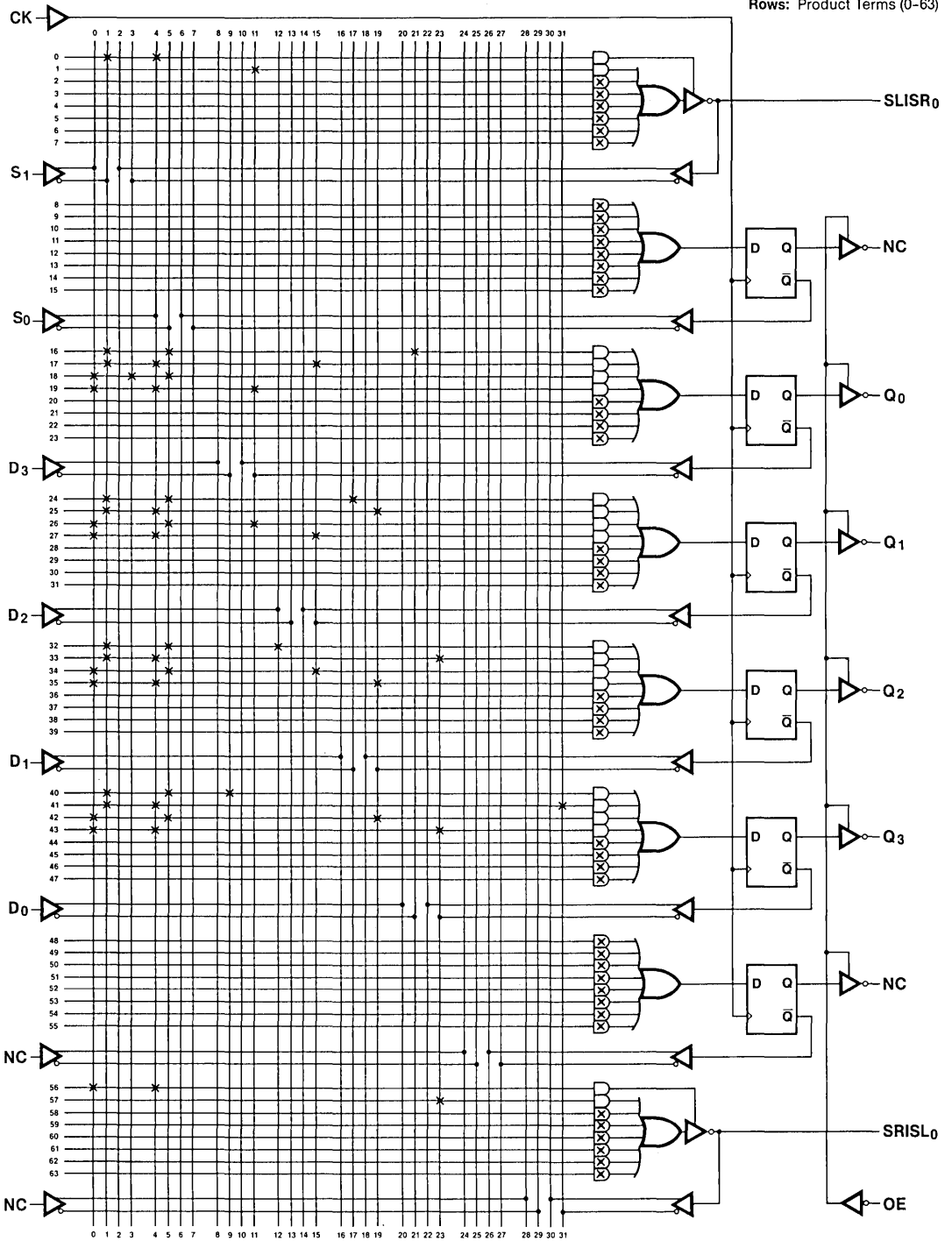
$\dashv\!\!\ast\!\!\vdash$ = Fuse intact $\dashv\!\!\boxtimes\!\!\vdash$ = All fuses intact $\dashv\!\!\vdash$ = Fuse blown

03862A-69

THE COUNTER

Counters are used for such purposes as state sequencing, delay timing, and event counting. The key to designing a counter is knowing when a bit should be toggled. For an up-counter, a bit is toggled whenever every bit of lesser significance is HIGH (see the counting sequence of Figure 18).

Conversely, for a down-counter, a bit is toggled whenever every bit of lesser significance is LOW. In both cases, the LSB is always toggled. By ANDing all bits of lesser significance along with the complement of the current data in the register, the problem of when this bit is to be toggled has been solved. However, this is not sufficient. In order to complete the design, it must be ensured that the bit remains unchanged under all other conditions. This can be accomplished by ORing the complements of the lesser significant bits together and then ANDing the result with the current data in the register (see Figure 19). The equation in Figure 19 can be changed into the sum-of-products form (Figure 20) for direct implementation in a PAL. Thus if a bit is to be toggled, the complement of the current data will be clocked in; if not, the data remains unchanged by clocking in the current data.

A 4-bit up-counter example illustrates this approach. Typical counter functions are loading data, counting, and "holding" data ($\overline{COUNTING}$). The function table is shown in Figure 21 and the logic diagram in Figure 22.

Expanding the number of bits in the counter is done by expanding the function table to incorporate the additional bits. Karnaugh maps, although not essential, can be used to find the required equations in sum-of-products form for PAL implementation. In general, besides any fixed overhead for control functions (CLEAR, LOAD, and HOLD in this example) bit n will require n additional product terms. Therefore, if this example 4-bit counter is to be expanded to five bits, the fifth bit will require five product terms plus three additional product terms for clearing, loading, and counting (see Figure 23). Notice that the original 4-bit block is unaffected by the addition of the fifth bit. This basic counter is easily expandable to perform more complex functions. The Design Specification and Logic Diagram for this counter, using an AmPAL16R8 device, are shown in Tables 4(a) and 4(b).

CURRENT STATE	NEXT STATE
0 0 0 0	0 0 0 1
0 0 0 1	0 0 1 0
0 0 1 0	0 0 1 1
0 0 1 1	0 1 0 0
0 1 0 0	0 1 0 1
0 1 0 1	0 1 1 0
0 1 1 0	0 1 1 1
0 1 1 1	1 0 0 0
1 0 0 0	1 0 0 1
1 0 0 1	1 0 1 0
1 0 1 0	1 0 1 1
1 0 1 1	1 1 0 0
1 1 0 0	1 1 0 1
1 1 0 1	1 1 1 0
1 1 1 0	1 1 1 1
1 1 1 1	0 0 0 0

03862A-70

Figure 18. Counting Sequence

$$\overline{TOGGLE} = Q_i \cdot (\overline{Q_0} + \overline{Q_1} + \overline{Q_2} + \cdots + \overline{Q_{i-1}})$$

03862A-71
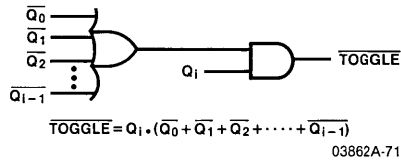
Figure 19. Logic for not Toggling Bit i

$$\overline{TOGGLE} = \overline{Q_0} \cdot Q_i + \overline{Q_1} \cdot Q_i + \overline{Q_2} \cdot Q_i + \cdots + \overline{Q_{i-1}} \cdot Q_i$$

03862A-72
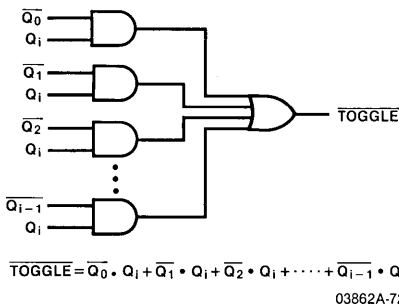
Figure 20. Equivalent Form of Figure 16

	INPUTS		OUTPUTS							
	S_1	S_0	$Q3_0$	$Q2_0$	$Q1_0$	$Q0_0$	Q_3	Q_2	Q_1	Q_0
CLEAR	0	0	X	X	X	X	0	0	0	0
LOAD	0	1	X	X	X	X	D_3	D_2	D_1	D_0
COUNT	1	0	0	0	0	0	0	0	0	1
	1	0	0	0	0	1	0	0	1	0
	1	0	0	0	1	0	0	0	1	1
	1	0	0	0	1	1	0	1	0	0
	1	0	0	1	0	0	0	1	0	1
	1	0	0	1	0	1	0	1	1	0
	1	0	0	1	1	0	0	1	1	1
	1	0	0	1	1	1	1	0	0	0
	1	0	1	0	0	0	1	0	0	1
	1	0	1	0	0	1	1	0	1	0
	1	0	1	0	1	0	1	0	1	1
	1	0	1	0	1	1	1	1	0	0
	1	0	1	1	0	0	1	1	0	1
	1	0	1	1	0	1	1	1	1	0
	1	0	1	1	1	0	1	1	1	1
	1	0	1	1	1	1	0	0	0	0
HOLD	1	1	X	X	X	X	$Q3_0$	$Q2_0$	$Q1_0$	$Q0_0$

03862A-73

Figure 21. Function Table for 4-Bit Up Counter
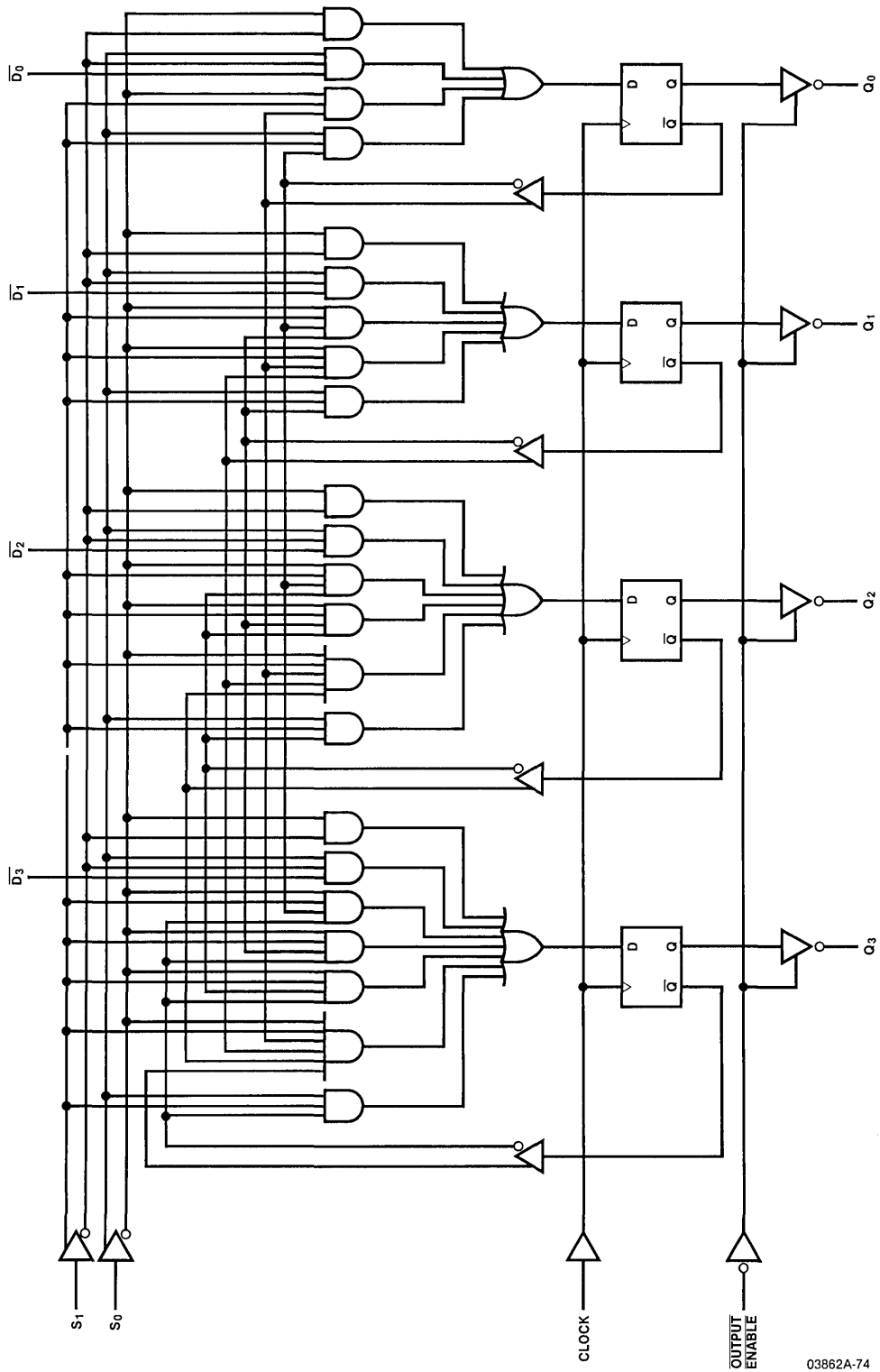
Figure 22. 4-Bit Up Counter

03862A-74
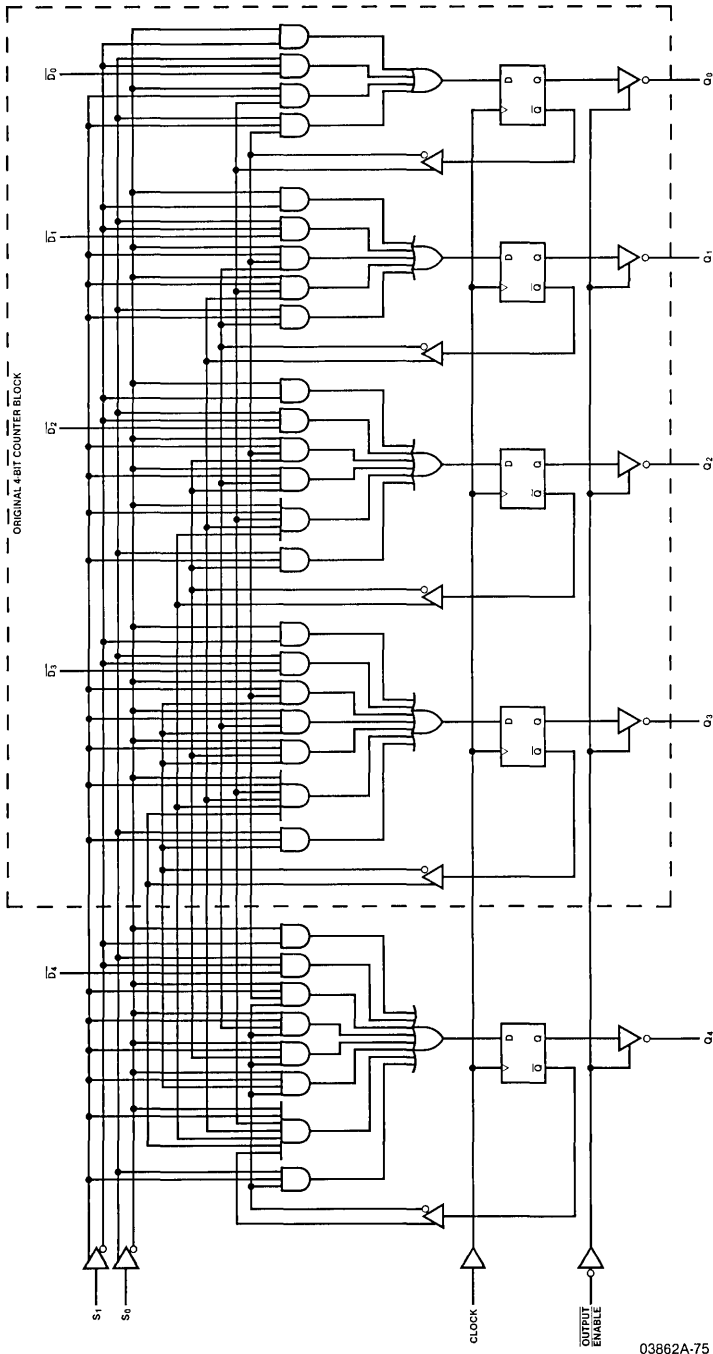
Figure 23. 5-Bit Up Counter

03862A-75

3-25

Table 4(a). Design Specification for 4-Bit Counter

```
PAL16R8                                 PAL DESIGN SPECIFICATION
PATO21                                  JENNY YEE  10/22/82
4-BIT COUNTER
ADVANCED MICRO DEVICES
CK   S1 SO D4  D3  D2  D1  DO NC GND
OE   NC NC NC  QO  Q1  Q2  Q3 NC VCC
;
;COUNTER OUTPUT SIGNALS
;
/QO      := /S1*/SO                  +
            /S1* SO*/DO              +
            S1*/SO* QO              +
            S1* SO*/QO

/Q1      := /S1*/SO                  +
            /S1* SO*/D1              +
            S1*/SO*/Q1*/QO          +
            S1*/SO* Q1* QO          +
            S1* SO*/Q1

/Q2      := /S1*/SO                  +
            /S1* SO*/D2              +
            S1*/SO*/Q2*/QO          +
            S1*/SO*/Q2*/Q1          +
            S1*/SO* Q2* Q1* QO      +
            S1* SO*/Q2

/Q3      := /S1*/SO                  +
            /S1* SO*/D3              +
            S1*/SO*/Q3*/QO          +
            S1*/SO*/Q3*/Q1          +
            S1*/SO*/Q3*/Q2          +
            S1*/SO* Q3* Q2* Q1* QO  +
            S1* SO*/Q3
```

03862A-76

Table 4(a). Design Specification for 4-Bit Counter (Continued)

```
FUNCTION TABLE

CK   S1 SO OE DO D1 D2 D3      Q3 Q2 Q1 Q0
-------------------------------------------------------------------
;
; CLEAR
;
C    L  L  L  X  X  X  X       L  L  L  L
;
; LOAD
;
C    L  H  L  H  H  H  H       H  H  H  H
C    L  H  L  L  L  L  L       L  L  L  L
;
; COUNT
;
C    H  L  L  X  X  X  X       L  L  L  H
C    H  L  L  X  X  X  X       L  L  H  L
C    H  L  L  X  X  X  X       L  L  H  ·H
C    H  L  L  X  X  X  X       L  H  L  L
C    H  L  L  X  X  X  X       L  H  L  H
C    H  L  L  X  X  X  X       L  H  H  L
C    H  L  L  X  X  X  X       L  H  H  H
C    H  L  L  X  X  X  X       H  L  L  L
C    H  L  L  X  X  X  X       H  L  L  H
C    H  L  L  X  X  X  X       H  L  H  L
C    H  L  L  X  X  X  X       H  L  H  H
C    H  L  L  X  X  X  X       H  H  L  L
C    H  L  L  X  X  X  X       H  H  L  H
C    H  L  L  X  X  X  X       H  H  H  L
C    H  L  L  X  X  X  X       H  H  H  H
C    H  L  L  X  X  X  X       L  L  L  L
;
; HOLD
;
C    H  H  L  X  X  X  X       L  L  L  L
-------------------------------------------------------------------
DESCRIPTION
THIS DEVICE IS AN UP-COUNTER. THE SAMPLE LAYOUT SHOWS THE
FORMAT OF DESIGNING THE COUNTER USING A PAL.          03862A-77
```

Table 4(a). Design Specification for 4-Bit Counter (Continued)

```
PAL16R8                                    PAL DESIGN SPECIFICATION
PAT021                                     JENNY YEE  10/22/82
4-BIT COUNTER
ADVANCED MICRO DEVICES
*D9724
*F0*
L0256 1011 1011 1111 1111 1111 1111 1111 1111 *
L0288 1011 0111 1111 1011 1111 1111 1111 1111 *
L0320 0111 1010 1111 1111 1110 1111 1111 1111 *
L0352 0111 1010 1111 1110 1111 1111 1111 1111 *
L0384 0111 1010 1110 1111 1111 1111 1111 1111 *
L0416 0111 1001 1101 1101 1101 1111 1111 1111 *
L0448 0111 0110 1111 1111 1111 1111 1111 1111 *
L0512 1011 1011 1111 1111 1111 1111 1111 1111 *
L0544 1011 0111 1111 1111 1011 1111 1111 1111 *
L0576 0111 1011 1110 1111 1110 1111 1111 1111 *
L0608 0111 1011 1110 1110 1111 1111 1111 1111 *
L0640 0111 1011 1101 1101 1101 1111 1111 1111 *
L0672 0111 0111 1110 1111 1111 1111 1111 1111 *
L0768 1011 1011 1111 1111 1111 1111 1111 1111 *
L0800 1011 0111 1111 1111 1111 1011 1111 1111 *
L0832 0111 1011 1111 1110 1110 1111 1111 1111 *
L0864 0111 1011 1111 1101 1101 1111 1111 1111 *
L0896 0111 0111 1111 1110 1111 1111 1111 1111 *
L1024 1011 1011 1111 1111 1111 1111 1111 1111 *
L1056 1011 0111 1111 1111 1111 1111 1011 1111 *
L1088 0111 1011 1111 1111 1101 1111 1111 1111 *
L1120 0111 0111 1111 1111 1110 1111 1111 1111 *
C4FAE*
V0001 C00XXXXXX00XXXLLLLX1 *
V0002 C01X1111X00XXXHHHHX1 *
V0003 C01X0000X00XXXLLLLX1 *
V0004 C10XXXXXX00XXXHLLLX1 *
V0005 C10XXXXXX00XXXLHLLX1 *
V0006 C10XXXXXX00XXXHHLLX1 *
V0007 C10XXXXXX00XXXLLHLX1 *
V0008 C10XXXXXX00XXXHLHLX1 *
V0009 C10XXXXXX00XXXLHHLX1 *
V0010 C10XXXXXX00XXXHHHLX1 *
V0011 C10XXXXXX00XXXLLLHX1 *
V0012 C10XXXXXX00XXXHLLHX1 *
V0013 C10XXXXXX00XXXLHLHX1 *
V0014 C10XXXXXX00XXXHHLHX1 *
V0015 C10XXXXXX00XXXLLHHX1 *
V0016 C10XXXXXX00XXXHLHHX1 *
V0017 C10XXXXXX00XXXLHHHX1 *
V0018 C10XXXXXX00XXXHHHHX1 *
V0019 C10XXXXXX00XXXLLLLX1 *
V0020 C11XXXXXX00XXXLLLLX1 *
887C
```

03862A-78

Table 4(b). Logic Diagram for 4-Bit Counter Using AmPAL16R8

Columns: Inputs (0–31)
Rows: Product Terms (0–63)
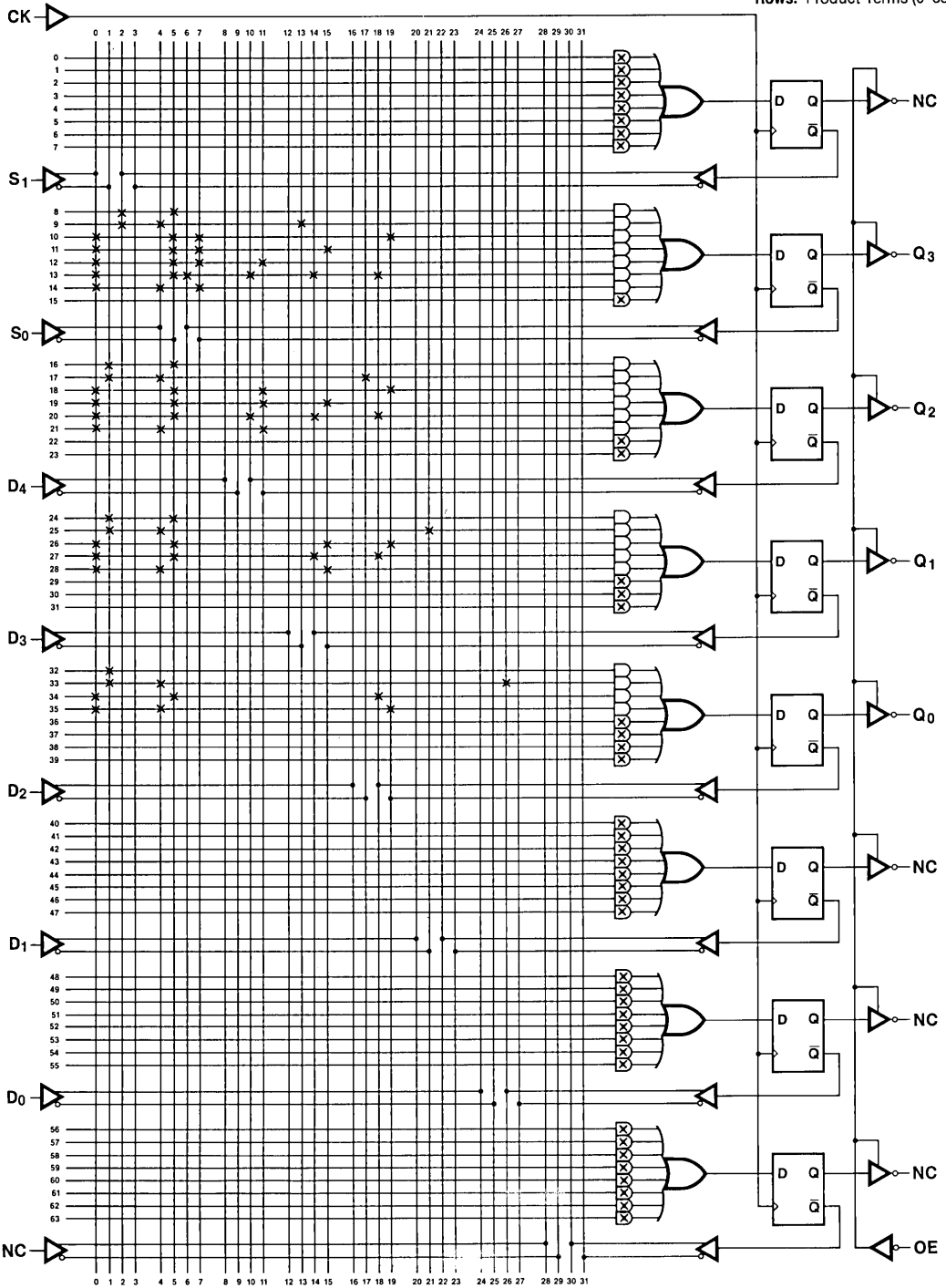
03862A-79

──✳── = Fuse intact   ─⊗─ = All fuses intact   ─┼─ = Fuse blown

Software Support for AMD PALs

Design Aid Software for Programmable Logic
PAL DESIGN SPECIFICATION

Design Aid Software for Programmable Logic

INTRODUCTION

The main function of programmable logic design aid software is to translate a custom logic design specification into a format which can be accepted by a programmer.

Programmable logic software is also an excellent tool for design simulation and documentation. Simulation aids in debugging an initial design and helps to assure that a device will operate as intended the first time instead of requiring multiple design iterations. Documentation capability is essential for someone other than the original designer to understand a custom programmable logic specification.

THIRD PARTY SOFTWARE

Many different programmable logic design aid software programs and software programs resident on programmable logic programmers are available or under development. Table 1 lists some current suppliers of these design tools. Contact the indicated companies for the status of their particular product.

PALASM

One particular software program, specifically for PALs, is PALASM (short for PAL ASSEMBLER).

PALASM is a computer aided design tool which has the capability of accepting an input file of Boolean equations and assembling the file into an output (fuse map) in a format compatible with PAL programmers. PALASM also allows the designer to input, in a tabular format, test vectors to perform simulation and debug of the Boolean equation input. The PAL design input file, called PAL DESIGN SPECIFICATION, when used in conjunction with PALASM documentation outputs, can provide as much documentation as is required to understand custom PAL design.

Table 1. Third Party Design Aid Software Tools

CUPL (Software) IBM PC	Assisted Technology, Inc. Suite 150 2381 Zanker Road San Jose, CA 95131 (408) 942-8787
PALASM (Programmer Resident)	Data I/O Corporation 10525 Willows Road N.E./C-46 Redmond, WA 98052 (206) 881-6444
PALASM (Programmer Resident)	Digilec, Inc. 7335 East Acoma Drive STE. 103 Scottsdale, AZ 85260 (602) 991-7268
PALASM (Programmer Resident)	Stag Microsystems, Inc. 528-5 Weddell Drive Sunnyvale, CA 94086 (408) 745-1991
PALASM (Programmer Resident)	Structured Design 1700 Wyatt Drive Suite 3 Santa Clara, CA 95054 (408) 988-0725

03862A-80

AMPALASM20

Advanced Micro Devices offers a powerful, enhanced version of PALASM, called AMPALASM20 to support the AMD 20-pin PAL family. It is based on the original version of PALASM but adds extensive error checking features and the industry standard JEDEC output format PLDTF (Programmable Logic Data Transfer Format).

The new error checking functions insure that creation of a correct PAL DESIGN SPECIFICATION is an easily understood and straightforward process. Most important, AMPALASM20 does not allow the designer to create incorrect fuse patterns or try to program a device with inadequate logical capacity for the desired function. The JEDEC PLDTF output is designed to transfer data in a format which can program PALs from all manufacturers. This eliminates the need for the user to modify code to accommodate different supplier's devices.

AMPALASM20 is currently available on 8 inch floppy disk media in IBM 3740 format for the CP/M operating system. It is offered in both Fortran source code and object code versions. Advanced Micro Devices only supports object code written for 8080 based systems. Source code is provided for customers who wish to port AMPALASM20 to their particular system, but no support is furnished and no responsibility can be accepted for any use of source code information.

To use this version of AMPALASM20 the designer should have access to:

—A microcomputer with a CP/M operating system

—An 8 inch floppy disk drive (IBM 3740 format)

—A CRT terminal with keyboard

—System software which includes an editor

—A means of downloading the fuse programming data to a programmer

To perform automatic downloading of AMPALASM20 output to a PAL programmer, a link between the computer and the programmer is required. Most programmers accept input via an RS232 interface. This usually requires only an RS232 port, and RS232 cable from the computer and a software driver for the port, which are resident on most computers. Please refer to the programmer manufacturer's reference materials for details. The CP/M version of AMPALASM20 provides the necessary software driver.

Versions of AMPALASM20 for the IBM CMS operating system and VAX VMS operating system are under development. Check with your local Advanced Micro Devices sales office for price and availability.

A complete specification of the CP/M version AMPALASM20 and the PAL DESIGN SPECIFICATION input file appears later in Section 4.

PAL DESIGN
SPECIFICATION

The input to AMPALASM20 is called the PAL DESIGN SPECIFICATION. Since AMPALASM20 is a batch-oriented program, the PAL DESIGN SPECIFICATION is intended to be an input file created using the editing facility of the machine on which AMPALASM20 is executing. The PAL DESIGN SPECIFICATION is divided into five main sections: the heading, pin list, logic equations, function table, and description.

HEADING

The heading comprises the first four lines of the PAL DESIGN SPECIFICATION. Line one requires the left justified PAL part number starting in column one. Valid PAL part numbers supported by the present version of AMPALASM20 are the following:

```
AMPAL16L8
AMPAL16R8
AMPAL16R6
AMPAL16R4
AMPAL16H8
AMPAL16LD8
AMPAL16HD8
```

An invalid part number will generate a fatal (nonrecoverable) error. The rest of the heading is optional, the following recommendations are standard.

The recommended heading "PAL DESIGN SPECIFICATION" should be included after the part number (separated by at least one space).

Lines two through four are reserved for design documentation. The recommended format for these lines is:

Line 2: User internal part number, date
Line 3: Device application name
Line 4: Company name, address

An example heading is given below:

```
AMPAL16R4                                  PAL DESIGN SPECIFICATION
PATTERN NUMBER K2/044-C                       WARREN MILLER 4/1/82
DUAL BUS 4-BIT COUNTER WITH BIT SWAPPABLE OUTPUTS
ADVANCED MICRO DEVICES     901 THOMPSON PLACE    SUNNYVALE CA 94086
```

PIN LIST

The pin list must begin on line five. It is a sequence of symbolic names given to device pins 1 through 20 in order. Each pin name must be separated by one or more spaces. The following rules should be followed when specifying the pin list:

(1) The pin list must contain exactly 20 names or a fatal error will be reported.

(2) Pin names should not exceed 8 characters. If they do, only the last 8 characters will be used.

(3) Any printable characters may be used in a pin name except " = : * + ( );". A slash may be used as the first character in a pin name to indicate an active LOW input signal.

(4) Duplicated pin names are allowed (for example, NC for No Connection), but only if they are not used in any equations.

(5) GND and VCC may be usd on pins 10 and 20 respectively, but should not be used in any equations.

An example pin list is given below:

```
CLK   CARRYIN ADATAO ADATA1 ADATA2 ADATA3 SO S1     RESET  GND
/OE   BDATAO  BDATA1 QO     Q1     Q2     Q3 BDATA2 BDATA3 VCC
```

LOGIC EQUATIONS

Logic equations may begin on the first line after the pin list. These equations are the heart of the PAL DESIGN SPECIFICATION. They specify the logical operations of a device in a sum-of-products (AND-OR) Boolean format.

The terms used when writing logic equations are:

OPERATOR SYMBOLS (in hierarchy of evaluation):

* AND (product)
+ OR (sum)

EXPRESSION is a sequence of PIN NAMES (or their complements) separated by operators, where the PIN NAME is the symbolic input or output name taken from the pin list.

PRODUCT is a sequence of PIN NAMES (or their complements) separated by the AND operator, "*", where the PIN NAME is the symbolic input or output name taken from the pin list.

ADDITIONAL SYMBOLS:

; The remainder of the present line is a comment
/ Complement, prefix to a pin name
= Combinatorial equality
:= Sequential equality

Three forms of logic equations are possible:

PIN NAME = EXPRESSION

Combinatorial equality. The output specified by the PIN NAME is logically equal to the expression. If the device has an inverting output, the complement of the PIN NAME must be specified.

IF (PRODUCT) PIN NAME = EXPRESSION

Conditional combinatorial equality. When the product is logically true, the output specified by PIN NAME is equivalent to the expression. When the product is logically false, the output is placed in the high impedance (high-Z) state. If the device has an inverting output, the complement of the PIN NAME must be specified.

PIN NAME : = EXPRESSION

Sequential equality. On the low to high transition of the clock, the registered output specified by the PIN NAME is loaded with the logical value defined by the equation. If the device has an inverting output, the complement of the PIN NAME must be specified.

It is important to notice that AMPALASM20 equations are written with respect to the AND inputs of the internal AND-OR structure. The only way to enable (output = HIGH) an AND-gate is with all HIGHs on the inputs. Therefore to enable an AND-gate when active LOW inputs (pin names with preceding slashes) are specified, the complement of the inputs (active HIGH or true version, without the slash) are necessary. Also, notice that to enable an AND-gate with active HIGH inputs the active HIGH version of the inputs is necessary (no slash). Thus the polarity of the input signals is written the same independent of the polarities of the pin names. It is, then merely a question of AMPALASM20 automatically selecting the inverting (active LOW) on non-inverting (active HIGH) path of the input buffer to the AND-gate.

It is also important to notice that the AMPALASM20 equations are written with respect to the output of the OR-structure in the combinatorial (16L8, 16H8, 16LD8, 16HD8) devices and at the Q outputs of the registers for the registered (16R8, 16R6, 16R4) devices. On the active LOW parts (16L8, 16LD8, 16R8, 16R6, 16R4) the output signals on the output pins are inverted versions of the internal, AMPALASM20 specified, OR-structure outputs. A warning is generated by AMPALASM20 if this inversion between pin list pin names and AMPALASM20 output pin names is not observed.

An example with a pin list and output equations is given below. Note the inversion between pin names and left hand sides of the equations, the use of sequential equality on registered outputs, and combinatorial equality on the non-registered outputs. Furthermore, note that an input such as RESET could be made active LOW by just adding a preceding slash "/" to the pin name (this tells PALASM 20 to select the inverting path of the input buffer).

```
CLK   CARRYIN ADATA0 ADATA1 ADATA2 ADATA3 SO S1      RESET  GND
/OE   BDATA0  BDATA1 Q0     Q1     Q2     Q3 BDATA2  BDATA3 VCC

        /Q0 :=    RESET                          +
                  /S1*/S0*/ADATA0               +   ;LOAD A
                  /S1* S0*/BDATA0               +   ;LOAD B
                  S1* Q0* CARRYIN               +   ;COUNT
                  S1*/Q0*/CARRYIN

        /Q1 :=    RESET                          +
                  /S1*/S0*/ADATA1               +   ;LOAD A
                  /S1* S0*/BDATA1               +   ;LOAD B
                  S1* Q1* Q0* CARRYIN           +   ;COUNT
                  S1*/Q1*/Q0                    +
                  S1*/Q1*/CARRYIN

        /Q2 :=    RESET                          +
                  /S1*/S0*/ADATA2               +   ;LOAD A
                  /S1* S0*/BDATA2               +   ;LOAD B
                  S1* Q2* Q1* Q0* CARRYIN       +   ;COUNT
                  S1*/Q2*/Q1                    +
                  S1*/Q2*/Q0                    +
                  S1*/Q2*/CARRYIN

        /Q3 :=    RESET                          +
                  /S1*/S0*/ADATA3               +   ;LOAD A
                  /S1* S0*/BDATA3               +   ;LOAD B
                  S1* Q3* Q2* Q1* Q0* CARRYIN   +   ;COUNT
                  S1*/Q3*/Q2                    +
                  S1*/Q3*/Q1                    +
                  S1*/Q3*/Q0                    +
                  S1*/Q3*/CARRYIN

   IF( S1 ) /BDATA0 =  /S0*/Q0          +   ;ENABLE NORMAL
                        S0*/Q3              ;ENABLE SWAPPED

   IF( S1 ) /BDATA1 =  /S0*/Q1          +   ;ENABLE NORMAL
                        S0*/Q2              ;ENABLE SWAPPED

   IF( S1 ) /BDATA2 =  /S0*/Q2          +   ;ENABLE NORMAL
                        S0*/Q1              ;ENABLE SWAPPED

   IF( S1 ) /BDATA3 =  /S0*/Q3          +   ;ENABLE NORMAL
                        S0*/Q0              ;ENABLE SWAPPED
```

FUNCTION TABLE

The function table portion of the PAL DESIGN SPECIFICA-TION is similar to the familiar truth table found in most TTL data books. It defines, in a tabular format, how the device is to function. Additionally, the function table is used by the SIMULATE function to check a PAL DESIGN for correctness. SIMULATE will take the vector specified in the function table and "plug it into" the equation portion of the PAL DESIGN SPECIFICATION.

Any discrepancy between the computed values and the function table values will be flagged by the output of SIMULATE. Besides catching the most common errors (signal inversions and typing errors), it also provides a check on the more subtle logic errors. The more detailed the function table, the higher the confidence level that the device will function as desired.

For combinatorial PALs, by using the input values specified in a function table line, the outputs may be computed against the output values listed in the same line of the table. For registered PALs, the present state (before clock) of the registers may also be necessary to compute and check the next state (after clock) outputs. The function table defines the present state outputs to be the registered outputs of the previous vector (i.e., line) and next state outputs to be in the current vector (line) with the present inputs. Note that the first vector in a function table has no present state and therefore cannot be dependent upon one. Also note that the combinatorial outputs of registered PALs are defined as a function of the present inputs and next state outputs.

The function table format is given below:

```
FUNCTION TABLE:    (pin list)
-------------------------------
d d .......................d
...............................
....            ........
d d .......................d
-------------------------------
```

The beginning of the function table is defined by the text "FUNCTION TABLE:". This identifier must begin in column 1. Following the function table, on the next line, is a pin list which defines the order of function table entries. This pin list must use the same signal names as the pin list at the beginning of the equation section of AMPALASM20. However, either the true or complement of the signal may be specified. Additionally, the order of entries need not be the same between the two pin lists.

Following the pin list is a dashed line of any length, which specifies the beginning of the body of the function table. Each entry in the function table indicates the forced state (in the case of inputs) or the checked state (in the case of outputs). A full line in the function table represents a test vector. A state (e.g., L, H, C, X, Z) must be specified for each pin name, with separating spaces optional. Optional comments may follow the vector. An entire line may be commented if the first character of the line is a semi-colon ";". The definitions for inputs and outputs in the function table are:

For inputs:
  L indicates drive input to a logical LOW
  H indicates drive input to a logical HIGH
  C indicates drive input to a logical LOW, then to a logical HIGH (clock)
  X indicates an irrelevant input (treated as drive to a logical LOW)

For outputs:
  L indicates test output for a logical LOW
  H indicates test output for a logical HIGH
  Z indicates test output for a logical HIGH impedance
  X indicates don't test

A function table for the sample device in the equations section is shown on the following page. The sample device equations implement a 4-bit loadable up counter. The counter is loadable from both the ADATA and BDATA ports. The counter outputs are available on both the three-state Q output and the BDATA port. When enabled onto the BDATA port either the normal output (Q_3, Q_2, Q_1, Q_0) or a swapped output (Q_0, Q_1, Q_2, Q_3) is available.

Note that the function table pin list does not maintain the same order as the device pinout. This allows the table to be laid out in a logical manner and be easily read and understood. Also note that the pin list is defined with the same polarity as the device pinout. This makes the table more like a "black box" definition of how the device will work. In general, the complement version of the device pinout is more desirable in the list only when internal state variables or multilevel logic is defined in the device. This is because these functions are intermediate and have nothing to do with the "black box" definition, and also because the true version of internal points of a PAL is usually the complement of the pinout (it's easier to read).

The first test vector of the function table checks the output disable function. This is done by driving the OE signal HIGH (H) and checking the Q outputs for the disabled state (Z). The second vector tests the reset function. The RESET signal is driven LOW (L), OE is LOW (L) and the clock input is clocked (C). The resulting Q outputs should be LOW (L).

The next two function table entries load the A and B ports respectively. This is done by selecting the load operations (LL and LH select codes) and checking that the correct value is clocked into the register.

The next 16 function table entries check the increment and B outputs functions. The first 7 vectors check the normal B output, the last 9 check the swapped B output. It is important to realize that the next state value of registered outputs can depend on both the inputs and the previous state (from the previous line) of the register. For example, the first entry in the increment section uses the previous value of the register (LLLL) as an input. This is incremented to the new value (LLLH). The next vector then uses that register state (LLLH) for the starting value of the test.

```
FUNCTION TABLE:
CLK ADATA3 ADATA2 ADATA1 ADATA0 BDATA3 BDATA2 BDATA1 BDATA0
/RESET S1 S0 CARRYIN /OE Q3 Q2 Q1 Q0

;                                    C
;    A A A A   B B B B   /           A
;    D D D D   D D D D   R           R
;    A A A A   A A A A   E           R
;C   T T T T   T T T T   S           Y   /
;L   A A A A   A A A A   E   S S     I   O   Q Q Q Q
;K   3 2 1 0   3 2 1 0   T   1 0     N   E   3 2 1 0        COMMENTS
-----------------------------------------------------------------------
X   X X X X   X X X X   X   X X     X   H   Z Z Z Z   ;OUTPUT DISABLE
;
C   H H H H   H H H H   L   L L     L   L   L L L L   ;RESET
;
C   H H H H   L L L L   H   L L     L   L   H H H H   ;LOAD A
;
C   H H H H   L L L L   H   L H     L   L   L L L L   ;LOAD B
;
C   X X X X   L L L H   H   H L     H   L   L L L H   ;INCREMENT WITH
C   X X X X   L L H L   H   H L     H   L   L L H L   ;B NORMAL
C   X X X X   L L H H   H   H L     H   L   L L H H
C   X X X X   L H L L   H   H L     H   L   L H L L
C   X X X X   L H L H   H   H L     H   L   L H L H
C   X X X X   L H H L   H   H L     H   L   L H H L
C   X X X X   L H H H   H   H L     H   L   L H H H
C   X X X X   L L L H   H   H H     H   L   H L L L   ;INCREMENT WITH
C   X X X X   H L L H   H   H H     H   L   H L L H   ;B SWAPPED
C   X X X X   L H L H   H   H H     H   L   H L H L
C   X X X X   H H L H   H   H H     H   L   H L H H
C   X X X X   L L H H   H   H H     H   L   H H L L
C   X X X X   H L H H   H   H H     H   L   H H L H
C   X X X X   L H H H   H   H H     H   L   H H H L
C   X X X X   H H H H   H   H H     H   L   H H H H
C   X X X X   L L L L   H   H L     H   L   L L L L
;
C   X X X X   L L L L   H   H L     L   L   L L L L   ;DISABLE INCREMENT
-----------------------------------------------------------------------
```

PAL DESCRIPTION

The description section of the PAL DESIGN SPECIFICATION is an important documentation tool. In this section the designer can describe the operation of the device and its intended application. If this section is done correctly the PAL DESIGN SPECIFICATION becomes a data sheet for the newly created device, completely documenting the design. The format for the description section is simply the keyword DESCRIPTION: followed by the text describing the design. An example description is given below:

```
DESCRIPTION:  THIS PAL IMPLEMENTS A 4-BIT COUNTER FOR A
MULTIPLE BUS CPU.  THE COUNTER CAN BE LOADED FROM EITHER THE
ADATA INPUT BUS OR THE BDATA INPUT-OUTPUT BUS.  THE CONTENTS
OF THE COUNTER CAN BE OUTPUT TO THE THREE STATE Q BUS OR THE
BDATA INPUT-OUTPUT BUS.  ADDITIONALLY WHEN THE BDATA BUS IS
USED TO OUTPUT THE COUNTER CONTENTS EITHER A BIT-NORMAL OR A
BIT-REVERSED VERSION CAN BE SELECTED.

MODE DEFINITIONS ARE GIVEN BELOW.
                          S1    S0    CARRY-IN
LOAD A                    0     0       X
LOAD B                    0     1       X
HOLD,   OUTPUT B NORMAL   1     0       0
COUNT,  OUTPUT B NORMAL   1     0       1
HOLD,   OUTPUT B SWAPPED  1     1       0
COUNT,  OUTPUT B SWAPPED  1     1       1
```

EXAMPLE

The following pages contain the complete PAL DESIGN SPECIFICATION for the example used in the previous sections.

```
AMPAL16R4                                   PAL DESIGN SPECIFICATION
PATTERN NUMBER K2/044-C                      WARREN MILLER 4/1/82
DUAL BUS 4-BIT COUNTER WITH BIT SWAPPABLE OUTPUTS
ADVANCED MICRO DEVICES     901 THOMPSON PLACE    SUNNYVALE CA 94086
CLK   CARRYIN ADATA0 ADATA1 ADATA2 ADATA3 S0 S1     RESET   GND
/OE   BDATA0  BDATA1 Q0     Q1     Q2      Q3 BDATA2 BDATA3 VCC

        /Q0 :=      RESET                       +
                    /S1*/S0*/ADATA0             +  ;LOAD A
                    /S1* S0*/BDATA0             +  ;LOAD B
                     S1* Q0* CARRYIN            +  ;COUNT
                     S1*/Q0*/CARRYIN

        /Q1 :=      RESET                       +
                    /S1*/S0*/ADATA1             +  ;LOAD A
                    /S1* S0*/BDATA1             +  ;LOAD B
                     S1* Q1* Q0* CARRYIN        +  ;COUNT
                     S1*/Q1*/Q0                 +
                     S1*/Q1*/CARRYIN

        /Q2 :=      RESET                       +
                    /S1*/S0*/ADATA2             +  ;LOAD A
                    /S1* S0*/BDATA2             +  ;LOAD B
                     S1* Q2* Q1* Q0* CARRYIN    +  ;COUNT
                     S1*/Q2*/Q1                 +
                     S1*/Q2*/Q0                 +
                     S1*/Q2*/CARRYIN

         /Q3 :=     RESET                       +
                    /S1*/S0*/ADATA3             +  ;LOAD A
                    /S1* S0*/BDATA3             +  ;LOAD B
                     S1* Q3* Q2* Q1* Q0* CARRYIN +  ;COUNT
                     S1*/Q3*/Q2                 +
                     S1*/Q3*/Q1                 +
                     S1*/Q3*/Q0                 +
                     S1*/Q3*/CARRYIN

   IF( S1 ) /BDATA0 =  /S0*/Q0        +  ;ENABLE NORMAL
                        S0*/Q3           ;ENABLE SWAPPED

   IF( S1 ) /BDATA1 =  /S0*/Q1        +  ;ENABLE NORMAL
                        S0*/Q2           ;ENABLE SWAPPED

   IF( S1 ) /BDATA2 =  /S0*/Q2        +  ;ENABLE NORMAL
                        S0*/Q1           ;ENABLE SWAPPED

   IF( S1 ) /BDATA3 =  /S0*/Q3        +  ;ENABLE NORMAL
                        S0*/Q0           ;ENABLE SWAPPED
```

```
FUNCTION TABLE:
CLK ADATA3 ADATA2 ADATA1 ADATA0 BDATA3 BDATA2 BDATA1 BDATA0
/RESET S1 S0 CARRYIN /OE Q3 Q2 Q1 Q0

;                                 C
;    A A A A  B B B B  /          A
;    D D D D  D D D D  R          R
;    A A A A  A A A A  E          R
;C   T T T T  T T T T  S          Y  /
;L   A A A A  A A A A  E  S S  I  O  Q Q Q Q
;K   3 2 1 0  3 2 1 0  T  1 0  N  E  3 2 1 0      COMMENTS
--------------------------------------------------------------
X   X X X X  X X X X  X  X X  X  H  Z Z Z Z  ;OUTPUT DISABLE
;
C   H H H H  H H H H  L  L L  L  L  L L L L  ;RESET
;
C   H H H H  L L L L  H  L L  L  L  H H H H  ;LOAD A
;
C   H H H H  L L L L  H  L H  L  L  L L L L  ;LOAD B
;
C   X X X X  L L L H  H  H L  H  L  L L L H  ;INCREMENT WITH
C   X X X X  L L H L  H  H L  H  L  L L H L  ;B NORMAL
C   X X X X  L L H H  H  H L  H  L  L L H H
C   X X X X  L H L L  H  H L  H  L  L H L L
C   X X X X  L H L H  H  H L  H  L  L H L H
C   X X X X  L H H L  H  H L  H  L  L H H L
C   X X X X  L H H H  H  H L  H  L  L H H H
C   X X X X  L L L H  H  H H  H  L  H L L L  ;INCREMENT WITH
C   X X X X  H L L H  H  H H  H  L  H L L H  ;B SWAPPED
C   X X X X  L H L H  H  H H  H  L  H L H L
C   X X X X  H H L H  H  H H  H  L  H L H H
C   X X X X  L L H H  H  H H  H  L  H H L L
C   X X X X  H L H H  H  H H· H  L  H H L H
C   X X X X  L H H H  H  H H  H  L  H H H L
C   X X X X  H H H H  H  H H  H  L  H H H H
C   X X X X  L L L L  H  H L  H  L  L L L L
;
C   X X X X  L L L L  H  H L  L  L  L L L L  ;DISABLE INCREMENT
--------------------------------------------------------------
     DESCRIPTION:  THIS PAL IMPLEMENTS A 4-BIT COUNTER FOR A
     MULTIPLE BUS CPU.  THE COUNTER CAN BE LOADED FROM EITHER THE
     ADATA INPUT BUS OR THE BDATA INPUT-OUTPUT BUS.  THE CONTENTS
     OF THE COUNTER CAN BE OUTPUT TO THE THREE STATE Q BUS OR THE
     BDATA INPUT-OUTPUT BUS.  ADDITIONALLY WHEN THE BDATA BUS IS
     USED TO OUTPUT THE COUNTER CONTENTS EITHER A BIT-NORMAL OR A
     BIT-REVERSED VERSION CAN BE SELECTED.

     MODE DEFINITIONS ARE GIVEN BELOW.
                                   S1    S0    CARRY-IN
     LOAD A                        0     0        X
     LOAD B                        0     1        X
     HOLD,  OUTPUT B NORMAL        1     0        0
     COUNT, OUTPUT B NORMAL        1     0        1
     HOLD,  OUTPUT B SWAPPED       1     1        0
     COUNT, OUTPUT B SWAPPED       1     1        1
```

AMPALASM20 COMMANDS (CP/M Version)

AMPALASM20 contains several different commands which simplify the design process. These commands not only translate the PAL DESIGN SPECIFICATION into a format useable by a programmable logic device programmer, but also help in checking designs for correctness and assist in design documentation. The list of AMPALASM20 commands appears below. A detailed description of each command follows the list.

JEDEC: Creates a fuse map in the JEDEC standard PLDTF from the PAL DESIGN SPECIFICATION and stores it on disk.

SIMULATE: Simulates the Boolean equation input by comparing it with a user created table of test vectors in the PAL DESIGN SPECIFICATION, called the FUNCTION TABLE. Following simulation, a JEDEC standard PLDTF output with fuse map and test vectors is created.

ECHO: Prints a copy of the PAL DESIGN SPECIFICATION.

PLOT: Prints a graphic representation of the programmed PAL fuse map.

NEXT: Loads a new file from the disk to be assembled. This allows multiple files to be assembled in a single session.

CHANGE: Changes the output destination to either the CRT, disk or programmer.

QUIT: Quit AMPALASM20 and return to native operating system.

JEDEC STANDARD PROGRAMMABLE LOGIC DATA TRANSFER FORMAT "J"

The Joint Electron Devices Engineering Council (JEDEC) programmable logic data transfer format (PLDTF) is a universal transfer format for fuse and test information between hosts and intelligent device programmers. This format is an industry standard used by many commercial programmer manufacturers. It gives each fuse a unique decimal address (specified in a logic diagram schematic) and indicates the state of each fuse for the programmed part. The format consists of four main sections: the design specification identifier, fuse link information, structured functional test information, and the sumcheck. The output of this command is a fuse map ready to be downloaded to a PAL programmer.

Details of the various sections of the PLDTF are available in Appendix A for the interested user.

An example of the JEDEC transfer format is shown on the next page. All fields except the structured test vectors are shown. The structured vectors are only created from AMPALASM20 following the SIMULATE command (explained later).

```
ENTER PAL20 OPTION: J

AMPAL16R4                                       PAL DESIGN SPECIFICATION
PATTERN NUMBER K2/O44-C                          WARREN MILLER 4/1/82
DUAL BUS 4-BIT COUNTER WITH BIT SWAPPABLE OUTPUTS
ADVANCED MICRO DEVICES      901 THOMPSON PLACE  SUNNNYVALE CA 94086
*D9724
*F0*
L0000 1111 1111 1111 1111 1111 1111 0111 1111 *
L0032 1111 1111 1110 1111 1111 1011 1111 1111 *
L0064 1111 1111 1111 1111 1111 0110 1111 1111 *
L0256 1111 1111 1111 1111 1111 1111 0111 1111 *
L0288 1111 1111 1111 1110 1111 1011 1111 1111 *
L0320 1111 1111 1111 1111 1110 0111 1111 1111 *
L0512 1111 1111 1111 1111 1111 1111 1111 0111 *
L0544 1111 1111 1111 1111 1011 1011 1011 1111 *
L0576 1110 1111 1111 1111 1111 0111 1011 1111 *
L0608 0111 1111 1101 1101 1101 1101 0111 1111 *
L0640 1111 1111 1110 1110 1111 1111 0111 1111 *
L0672 1111 1111 1110 1111 1110 1111 0111 1111 *
L0704 1111 1111 1110 1111 1111 1110 0111 1111 *
L0736 1011 1111 1110 1111 1111 1111 0111 1111 *
L0768 1111 1111 1111 1111 1111 1111 1111 0111 *
L0800 1111 1111 1111 1011 1111 1011 1011 1111 *
L0832 1111 1110 1111 1111 1111 0111 1011 1111 *
L0864 0111 1111 1111 1101 1101 1101 0111 1111 *
L0896 1111 1111 1111 1110 1110 1111 0111 1111 *
L0928 1111 1111 1111 1110 1111 1110 0111 1111 *
L0960 1011 1111 1111 1110 1111 1111 0111 1111 *
L1024 1111 1111 1111 1111 1111 1111 1111 0111 *
L1056 1111 1111 1011 1111 1111 1011 1011 1111 *
L1088 1111 1111 1111 1111 1111 0111 1010 1111 *
L1120 0111 1111 1111 1111 1101 1101 0111 1111 *
L1152 1111 1111 1111 1111 1110 1110 0111 1111 *
L1184 1011 1111 1111 1111 1110 1111 0111 1111 *
L1280 1111 1111 1111 1111 1111 1111 1111 0111 *
L1312 1111 1011 1111 1111 1111 1011 1011 1111 *
L1344 1111 1111 1111 1111 1111 0111 1011 1110 *
L1376 0111 1111 1111 1111 1111 1101 0111 1111 *
L1408 1011 1111 1111 1111 1111 1110 0111 1111 *
L1536 1111 1111 1111 1111 1111 1111 0111 1111 *
L1568 1111 1111 1111 1111 1110 1011 1111 1111 *
L1600 1111 1111 1111 1110 1111 0111 1111 1111 *
L1792 1111 1111 1111 1111 1111 1111 0111 1111 *
L1824 1111 1111 1111 1111 1111 1010 1111 1111 *
L1856 1111 1111 1110 1111 1111 0111 1111 1111 *
C8C36*
89A3
```

SIMULATE "S"

The SIMULATE command uses the function table entries and the logic equations to emulate the operation of the specified device. Any difference between the expected value (computed value) and the actual value (function table value) is flagged as a fatal error. The output format follows the JEDEC Programmable Logic Data Transfer Format as described previously under the JEDEC output format command. This output may be used by intelligent device programmers to program and test programmable logic devices, verifying logical functionality. An example output of the SIMULATE command is given below:

```
ENTER PAL20 OPTION: S

AMPAL16R4                              PAL DESIGN SPECIFICATION
PATTERN NUMBER K2/O44-C                   WARREN MILLER 4/1/82
DUAL BUS 4-BIT COUNTER WITH BIT SWAPPABLE OUTPUTS
ADVANCED MICRO DEVICES     901 THOMPSON PLACE  SUNNNYVALE CA 94086
*D9724
*FO*
L0000 1111 1111 1111 1111 1111 1111 0111 1111 *
L0032 1111 1111 1110 1111 1111 1011 1111 1111 *
L0064 1111 1111 1111 1111 1111 0110 1111 1111 *
L0256 1111 1111 1111 1111 1111 1111 0111 1111 *
L0288 1111 1111 1111 1110 1111 1011 1111 1111 *
L0320 1111 1111 1111 1111 1110 0111 1111 1111 *
L0512 1111 1111 1111 1111 1111 1111 1111 0111 *
L0544 1111 1111 1111 1111 1011 1011 1011 1111 *
L0576 1110 1111 1111 1111 1111 0111 1011 1111 *
L0608 0111 1111 1101 1101 1101 1101 0111 1111 *
L0640 1111 1111 1110 1110 1111 1111 0111 1111 *
L0672 1111 1111 1110 1111 1110 1111 0111 1111 *
L0704 1111 1111 1110 1111 1111 1110 0111 1111 *
L0736 1011 1111 1110 1111 1111 1111 0111 1111 *
L0768 1111 1111 1111 1111 1111 1111 1111 0111 *
L0800 1111 1111 1111 1011 1111 1011 1011 1111 *
L0832 1111 1110 1111 1111 1111 0111 1011 1111 *
L0864 0111 1111 1111 1101 1101 1101 0111 1111 *
L0896 1111 1111 1111 1110 1110 1111 0111 1111 *
L0928 1111 1111 1111 1110 1111 1110 0111 1111 *
L0960 1011 1111 1111 1110 1111 1111 0111 1111 *
L1024 1111 1111 1111 1111 1111 1111 1111 0111 *
L1056 1111 1111 1011 1111 1111 1011 1011 1111 *
L1088 1111 1111 1111 1111 1111 0111 1010 1111 *
L1120 0111 1111 1111 1111 1101 1101 0111 1111 *
L1152 1111 1111 1111 1111 1110 1110 0111 1111 *
L1184 1011 1111 1111 1111 1110 1111 0111 1111 *
L1280 1111 1111 1111 1111 1111 1111 1111 0111 *
L1312 1111 1011 1111 1111 1111 1011 1011 1111 *
L1344 1111 1111 1111 1111 1111 0111 1011 1110 *
L1376 0111 1111 1111 1111 1111 1101 0111 1111 *
L1408 1011 1111 1111 1111 1111 1110 0111 1111 *
L1536 1111 1111 1111 1111 1111 1111 0111 1111 *
L1568 1111 1111 1111 1111 1110 1011 1111 1111 *
L1600 1111 1111 1111 1110 1111 0111 1111 1111 *
L1792 1111 1111 1111 1111 1111 1111 0111 1111 *
L1824 1111 1111 1111 1111 1111 1010 1111 1111 *
L1856 1111 1111 1110 1111 1111 0111 1111 1111 *
C8C36*
V0001 XXXXXXXXX01XXZZZZXX1 *
V0002 C011110010011LLLL111 *
V0003 C011110000000HHHH001 *
V0004 C011111000000LLLL001 *
V0005 C1XXXX01000HLHLLLLL1 *
V0006 C1XXXX01000LHLHLLLL1 *
V0007 C1XXXX01000HHHHLLLL1 *
V0008 C1XXXX01000LLLLHLHL1 *
V0009 C1XXXX01000HLHLHLHL1 *
V0010 C1XXXX01000LHLHHLHL1 *
V0011 C1XXXX01000HHHHHLHL1 *
V0012 C1XXXX11000HLLLLHLL1 *
V0013 C1XXXX11000HLHLLHLH1 *
V0014 C1XXXX11000HLLHLHHL1 *
V0015 C1XXXX11000HLHHLHHH1 *
V0016 C1XXXX11000HHLLHHLL1 *
V0017 C1XXXX11000HHHLHHLH1 *
V0018 C1XXXX11000HLHLHHHHL1 *
V0019 C1XXXX11000HHHHHHHH1 *
V0020 C1XXXX01000LLLLLLLL1 *
V0021 C0XXXX01000LLLLLLLL1 *
1B05
```

ECHO "E"

The ECHO command displays the PAL DESIGN SPECIFICATION input file on the console. This mode allows the file to be reviewed for correctness or spooled to a listing device as a hard copy. An example of the ECHO command is given below:

```
ENTER PAL20 OPTION: E

AMPAL16R4                                   PAL DESIGN SPECIFICATION
PATTERN NUMBER K2/044-C                      WARREN MILLER 4/1/82
DUAL BUS 4-BIT COUNTER WITH BIT SWAPPABLE OUTPUTS
ADVANCED MICRO DEVICES      901 THOMPSON PLACE    SUNNYVALE CA 94086
CLK   CARRYIN ADATA0 ADATA1 ADATA2 ADATA3 SO S1       RESET   GND
/OE   BDATA0  BDATA1 Q0     Q1     Q2      Q3 BDATA2 BDATA3 VCC

        /Q0 :=     RESET                        +
                   /S1*/SO*/ADATA0              +  ;LOAD A
                   /S1* SO*/BDATA0              +  ;LOAD B
                    S1* Q0* CARRYIN             +  ;COUNT
                    S1*/Q0*/CARRYIN

        /Q1 :=     RESET                        +
                   /S1*/SO*/ADATA1              +  ;LOAD A
                   /S1* SO*/BDATA1              +  ;LOAD B
                    S1* Q1* Q0* CARRYIN         +  ;COUNT
                    S1*/Q1*/Q0                  +
                    S1*/Q1*/CARRYIN

        /Q2 :=     RESET                        +
                   /S1*/SO*/ADATA2              +  ;LOAD A
                   /S1* SO*/BDATA2              +  ;LOAD B
                    S1* Q2* Q1* Q0* CARRYIN     +  ;COUNT
                    S1*/Q2*/Q1                  +
                    S1*/Q2*/Q0                  +
                    S1*/Q2*/CARRYIN

         /Q3 :=    RESET                        +
                   /S1*/SO*/ADATA3              +  ;LOAD A
                   /S1* SO*/BDATA3              +  ;LOAD B
                    S1* Q3* Q2* Q1* Q0* CARRYIN +  ;COUNT
                    S1*/Q3*/Q2                  +
                    S1*/Q3*/Q1                  +
                    S1*/Q3*/Q0                  +
                    S1*/Q3*/CARRYIN

    IF( S1 ) /BDATA0 =  /SO*/Q0          +  ;ENABLE NORMAL
                         SO*/Q3             ;ENABLE SWAPPED

    IF( S1 ) /BDATA1 =  /SO*/Q1          +  ;ENABLE NORMAL
                         SO*/Q2             ;ENABLE SWAPPED

    IF( S1 ) /BDATA2 =  /SO*/Q2          +  ;ENABLE NORMAL
                         SO*/Q1             ;ENABLE SWAPPED

    IF( S1 ) /BDATA3 =  /SO*/Q3          +  ;ENABLE NORMAL
                         SO*/Q0             ;ENABLE SWAPPED
```

```
FUNCTION TABLE:
CLK ADATA3 ADATA2 ADATA1 ADATAO BDATA3 BDATA2 BDATA1 BDATAO
/RESET S1 SO CARRYIN /OE Q3 Q2 Q1 QO

;                               C
;   A A A A  B B B B  /         A
;   D D D D  D D D D  R         R
;   A A A A  A A A A  E         R
;C  T T T T  T T T T  S         Y  /
;L  A A A A  A A A A  E  S S  I  O  Q Q Q Q
;K  3 2 1 0  3 2 1 0  T  1 0  N  E  3 2 1 0      COMMENTS
---------------------------------------------------------------
X  X X X X  X X X X  X  X X  X  H  Z Z Z Z  ;OUTPUT DISABLE
;
C  H H H H  H H H H  L  L L  L  L  L L L L  ;RESET
;
C  H H H H  L L L L  H  L L  L  L  H H H H  ;LOAD A
;
C  H H H H  L L L L  H  L H  L  L  L L L L  ;LOAD B
;
C  X X X X  L L L H  H  H L  H  L  L L L H  ;INCREMENT WITH
C  X X X X  L L H L  H  H L  H  L  L L H L  ;B NORMAL
C  X X X X  L L H H  H  H L  H  L  L L H H
C  X X X X  L H L L  H  H L  H  L  L H L L
C  X X X X  L H L H  H  H L  H  L  L H L H
C  X X X X  L H H L  H  H L  H  L  L H H L
C  X X X X  L H H H  H  H L  H  L  L H H H
C  X X X X  L L L H  H  H H  H  L  H L L L  ;INCREMENT WITH
C  X X X X  H L L H  H  H H  H  L  H L L H  ;B SWAPPED
C  X X X X  L H L H  H  H H  H  L  H L H L
C  X X X X  H H L H  H  H H  H  L  H L H H
C  X X X X  L L H H  H  H H  H  L  H H L L
C  X X X X  H L H H  H  H H  H  L  H H L H
C  X X X X  L H H H  H  H H  H  L  H H H L
C  X X X X  H H H H  H  H H  H  L  H H H H
C  X X X X  L L L L  H  H L  H  L  L L L L
;
C  X X X X  L L L L  H  H L  L  L  L L L L  ;DISABLE INCREMENT
---------------------------------------------------------------
```

DESCRIPTION: THIS PAL IMPLEMENTS A 4-BIT COUNTER FOR A
MULTIPLE BUS CPU. THE COUNTER CAN BE LOADED FROM EITHER THE
ADATA INPUT BUS OR THE BDATA INPUT-OUTPUT BUS. THE CONTENTS
OF THE COUNTER CAN BE OUTPUT TO THE THREE STATE Q BUS OR THE
BDATA INPUT-OUTPUT BUS. ADDITIONALLY WHEN THE BDATA BUS IS
USED TO OUTPUT THE COUNTER CONTENTS EITHER A BIT-NORMAL OR A
BIT-REVERSED VERSION CAN BE SELECTED.

MODE DEFINITIONS ARE GIVEN BELOW.

	S1	SO	CARRY-IN
LOAD A	0	0	X
LOAD B	0	1	X
HOLD, OUTPUT B NORMAL	1	0	0
COUNT, OUTPUT B NORMAL	1	0	1
HOLD, OUTPUT B SWAPPED	1	1	0
COUNT, OUTPUT B SWAPPED	1	1	1

PLOT "P"

The PLOT command displays a graphical representation of the location of blown and intact fuses defined in the PAL DESIGN SPECIFICATION. It is oriented similarly to the fuse map in the PAL logic diagram; inputs from top to bottom, product terms from left to right. Additionally, the equation is displayed to the right of each product term. A cross "X" indicates an intact link and a dash "−" indicates a blown link for the appropriate input line. An example plot is given below:

```
                        ENTER PAL20 OPTION: P


                DUAL BUS 4-BIT COUNTER WITH BIT SWAPPABLE OUTPUTS

                            11 1111 1111 2222 2222 2233
                  0123 4567 8901 2345 6789 0123 4567 8901

              0 ---- ---- ---- ---- ---- ---- X--- ---- S1
              1 ---- ---- ---X ---- ---- -X-- ---- ---- /S0*/Q3
              2 ---- ---- ---- ---- ---- X--X ---- ---- S0*/Q0
              3 XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
              4 XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
              5 XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
              6 XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
              7 XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

              8 ---- ---- ---- ---- ---- ---- X--- ---- S1
              9 ---- ---- ---- ---X ---- -X-- ---- ---- /S0*/Q2
             10 ---- ---- ---- ---- ---X X--- ---- ---- S0*/Q1
             11 XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
             12 XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
             13 XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
             14 XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
             15 XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

             16 ---- ---- ---- ---- ---- ---- X--- RESET
             17 ---- ---- ---- ---- -X-- -X-- -X-- ---- /S1*/S0*/ADATA3
             18 ---X ---- ---- ---- ---- X--- -X-- ---- /S1*S0*/BDATA3
             19 X--- --X- --X- --X- --X- X--- ---- ---- S1*Q3*Q2*Q1*Q0*CARRYIN
             20 ---- ---- ---X ---X ---- ---- X--- ---- S1*/Q3*/Q2
             21 ---- ---- ---X ---- ---X ---- X--- ---- S1*/Q3*/Q1
             22 ---- ---- ---X ---- ---- ---X X--- ---- S1*/Q3*/Q0
             23 -X-- ---- ---X ---- ---- ---- X--- ---- S1*/Q3*/CARRYIN

             24 ---- ---- ---- ---- ---- ---- ---- X--- RESET
             25 ---- ---- ---- -X-- ---- -X-- -X-- ---- /S1*/S0*/ADATA2
             26 ---- ---X ---- ---- ---- X--- -X-- ---- /S1*S0*/BDATA2
             27 X--- ---- ---- --X- --X- --X- X--- ---- S1*Q2*Q1*Q0*CARRYIN
             28 ---- ---- ---- ---X ---X ---- X--- ---- S1*/Q2*/Q1
             29 ---- ---- ---- ---X ---- ---X X--- ---- S1*/Q2*/Q0
             30 -X-- ---- ---- ---X ---- ---- X--- ---- S1*/Q2*/CARRYIN
             31 XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

             32 ---- ---- ---- ---- ---- ---- ---- X--- RESET
             33 ---- ---- -X-- ---- ---- -X-- -X-- ---- /S1*/S0*/ADATA1
             34 ---- ---- ---- ---- ---- X--- -X-X ---- /S1*S0*/BDATA1
             35 X--- ---- ---- ---- --X- --X- X--- ---- S1*Q1*Q0*CARRYIN
             36 ---- ---- ---- ---- ---X ---X X--- ---- S1*/Q1*/Q0
             37 -X-- ---- ---- ---- ---X ---- X--- ---- S1*/Q1*/CARRYIN
             38 XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
             39 XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
             40 ---- ---- ---- ---- ---- ---- ---- X--- RESET
             41 ---- -X-- ---- ---- ---- -X-- -X-- ---- /S1*/S0*/ADATA0
             42 ---- ---- ---- ---- ---- X--- -X-- ---X /S1*S0*/BDATA0
             43 X--- ---- ---- ---- ---- --X- X--- ---- S1*Q0*CARRYIN
             44 -X-- ---- ---- ---- ---- ---X X--- ---- S1*/Q0*/CARRYIN
             45 XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
             46 XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
             47 XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

             48 ---- ---- ---- ---- ---- ---- X--- ---- S1
             49 ---- ---- ---- ---- ---X -X-- ---- ---- /S0*/Q1
             50 ---- ---- ---- ---X ---- X--- ---- ---- S0*/Q2
             51 XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
             52 XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
             53 XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
             54 XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
             55 XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

             56 ---- ---- ---- ---- ---- ---- X--- ---- S1
             57 ---- ---- ---- ---- ---- -X-X ---- ---- /S0*/Q0
             58 ---- ---- ---X ---- ---- X--- ---- ---- S0*/Q3
             59 XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
             60 XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
             61 XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
             62 XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
             63 XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX


             LEGEND:  X : FUSE NOT BLOWN (L,N,0)   - : FUSE BLOWN   (H,P,1)

             NUMBER OF FUSES BLOWN = 1120
```

NEXT "N"

The NEXT command loads a new file from the disk to be assembled. This allows multiple files to be assembled in a single session.

Enter PALASM20 Option: N

Enter PAL DESIGN SPEC Input Filename: DMC.PAL

Output Destination:
  D = DISK
  C = CRT
  P = PROGRAMMER (PUN)

Enter Destination: C

ASSEMBLY SUCCESSFUL!

CHANGE "C"

The CHANGE command reassigns the output destination to either the CRT, disk or programmer.

Enter PALASM20 Option: C

Output Destination:
  D = DISK
  C = CRT
  P = PROGRAMMER (PUN)

Enter Destination: C

QUIT "Q"

The QUIT command transfers control back to the native operating system.

Enter PALASM20 Option: Q

ERROR MESSAGES

Introduction

AMPALASM20 may produce error messages when improper PAL description files are processed. Three types of error messages are possible: warnings, errors, or fatal errors. Examples of typical error messages are given below. Details of others are provided on AMPALASM20 HELP file.

Warning Messages

Warnings are used to notify the user that an oversight may have been made when the PAL description file was created. A warning will not prohibit creation of an output file, but the user should check the warnings before programming a device to insure that the output matches the intended device definition.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                                                       %
% ****WARNING****      VCC/GND IS NOT RECOMMENDED TO BE USED %
%                      IN THE EQUATIONS!                                %
%                                                                       %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

DESCRIPTION:

You do not have to use VCC or GND in the equation. Try to delete it.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                                                       %
% ****WARNING****      THE FUNCTION TABLE IS OMITTED            %
%                      PREVENTING SIMULATION!                           %
%                                                                       %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Error Messages

Errors are used to notify the user that an illegal operation has been requested during execution of AMPALASM20. For example if an illegal file is requested as input to or output from AMPALASM20, an error message will be displayed.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                                               %
%  ****ERROR****          INPUT READ ERROR IN LINE @            %
%                                                               %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

DESCRIPTION:

Physical read error.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                                               %
%  ****ERROR****          INPUT FILE DOES NOT EXIST, PLEASE     %
%                         REENTER!                              %
%                                                               %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

DESCRIPTION:

The program cannot find the input file specified; you have to reenter an existing input file name.

Fatal Error Messages

Fatal errors are used to notify the user that a nonrecoverable error exists in a PAL description file. These errors prohibit output file creation and must be corrected by the user prior to re-running AMPALASM20.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                                               %
%  ***FATAL ERROR***      THE PAL PART YOU SPECIFY CAN NOT BE   %
%                         RECOGNIZED BY AMPALASM 20!            %
%                                                               %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

DESCRIPTION:

This occurs because AMPALASM20 cannot recognize the PAL part you specify in the first line, note that the first line must begin with PAL, AMPAL, AmPAL, otherwise will activate this fatal error. Check your input file!

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                                               %
%  ***FATAL ERROR***      MISSING PIN NAME FOLLOWING THE SYMBOL %
%                         @ IN THE FOLLOWING LINES:             %
%                         -----                                 %
%                         -----                                 %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

DESCRIPTION:

A pin name is missed between the specified pin name and following operator.

Symbol Definition

@ is a place holder for the part number or pin name associated with the indicated error. In the displayed error message the actual number or name will be shown.

— is a place holder for the line of text associated with the indicated error. In the displayed error message the actual text will be shown.

APPENDIX A

JEDEC Standard Programmable Logic Data Transfer Format "J"

The Joint Electron Devices Engineering Council (JEDEC) programmable logic data transfer format (PLDTF) is a universal transfer format for fuse and test information between hosts and intelligent device programmers. This format is an industry standard used by many commercial programmer manufacturers. It gives each fuse a unique decimal address (specified in a logic diagram schematic) and indicates the state of each fuse for the programmed part. The format consists of four main sections: the design specification identifier, fuse link information, structured functional test information, and the sumcheck. The output of this command is a fuse map ready to be downloaded to a PAL programmer.

Design Specification Identifier

The DESIGN SPECIFICATION identifier is a heading used by the designer to document the device to be programmed. The user is free to specify any documenting text desired. The AMD recommended format is identical to the first four lines of the PAL DESIGN SPECIFICATION. The heading is begun with an ASCII "STX" (02 HEX) and is terminated with an ASCII asterisk "*" (2A HEX). Notice that this requires the PAL DESIGN SPECIFICATION heading to be asterisk free.

An optional device code can be specified, indicating to the device programmer the type of part to be programmed. This code is a variable length decimal number, preceded by an ASCII "D" (44 HEX) and terminated by an ASCII "*" (2A HEX).

Fuse Link Information

This section of the format defines the state of each fuse, and consists of three fields: fuse default state, link information, and checksum. The first field is optional and is used to indicate the fuse default state. If a fuse state isn't otherwise specified, the default state will be used. The default state is specified by an ASCII "F" (46 HEX) followed by an ASCII "0" (32 HEX) for a low resistance link or an ASCII "1" (33 HEX) for a high resistance link. The field is terminated with an ASCII "*" (2A HEX).

The second field, link information, identifies the state of each fuse individually. The field begins with an ASCII "L" (4C HEX) followed by an ASCII decimal fuse address of variable length, terminated by an ASCII "*" (2A HEX). This indicates the fuse address of the first fuse state. The individual fuses are specified by an ASCII "0" (32 HEX) or an ASCII "1" (33 HEX) indicating low resistance and high resistance respectively. The fuse address is incremented for each additional fuse state. Thus fuse states can be specified sequentially. Any number of fuse addresses may be specified by inserting additional "L" fields. If a link is specified 2 or more times, the last state replaces the preceding entries.

The third field is an optional checksum for the link information. This checksum is computed by performing a 16-bit addition of the 8-bit words constructed from the specified state of each fuse link in the device. The 8-bit words are constructed sequentially from the single bit fuse state definitions. The method of constructing these words is shown below.

```
                     MSB                              LSB

            **************************************
Word   0    *   *   *   *   *   *   *   *   *
            **************************************
Link No.    7   6   5   4   3   2   1   0


            **************************************
Word   1    *   *   *   *   *   *   *   *   *
            **************************************
Link No.    15  14  13  12  11  10  9   8


            **************************************
Word   2    *   *   *   *   *   *   *   *   *
            **************************************
Link No.    23  22  21  20  19  18  17  16


                              *

                              *

                              *


            ****************************
Word  137   *   *   *   *   *   *
            ****************************
Link No.    1100 1099 1098 1097 1096
            last
            link
```

The word encompassing the last link is constructed by setting zeros for all bit locations more significant than the last link. The 16-bit sum is expressed as 4 ASCII Hex characters. On ASCII "C" precedes the four Hex characters. The last character is followed by an ASCII "∗".

Structured Functional Test Information

The structured functional test information is an optional field which can define test vectors to be used by intelligent programmable logic device programmers to test the logical functionality of a programmed device. These vectors specify the driven state for inputs and the checked value for outputs.

The field is specified with an ASCII "V" followed by a variable length decimal test vector address. The address is terminated by an ASCII "W". Following the test vector address is a series of 20 characters specifying the driven or tested state for each pin.

The format for each character in the test vector output is given below:

H   Test output for a logical HIGH
L   Test output for a logical LOW

1   Drive input to a logical HIGH
0   Drive input to a logical LOW
X   Irrelevant. If an output do not test. If an input drive to a logical LOW as a default.
C   Drive input from logical LOW to logical HIGH. (clock pulse).
Z   Test output for high impedance.

The test vector is terminated with an ASCII asterisk "∗". Multiple test vectors are specified by incrementing the decimal vector address by 1 for each additional test vector.

Sumcheck

The sumcheck field provides redundancy to help in detecting errors in data transmission. This field is constructed by performing a binary addition of each character between the STX and ETX in the transmitted file. The resulting least significant 16 bits are the sumcheck. This number is represented as four Hex characters preceded by an ASCII asterisk "∗" in the final printout.

Examples of the JEDEC transfer format are shown previously in descriptions of the JEDEC and SIMULATE commands.

Section 5

Applications

Four-Bit Slice Registered Barrel Shifter

by Warren Miller
Advanced Micro Devices

In most data processing systems, some form of data shifting or rotation is necessary. This elementary function is used in such diverse applications as floating point arithmetic and string manipulation. In the typical computer, the shifter is located on the output of the ALU. This architecture allows single cycle add and shift, and mask and shift operations (see Figure 1 for a typical computer ALU architecture).

DESIGN REQUIREMENTS

A barrel shifter takes data input and cyclically rotates it by an arbitrary number of bit positions. This cyclic rotation means that data rotated off the most significant end of the shifter is brought back on the least significant end. The name barrel shifter is used because of the circular nature of the shift operation.

The storage register on the output of the shifter is used in this architecture to pipeline the data operation, increasing throughput. The three-state buffer on the register output is necessary to interface the ALU to the output data bus.
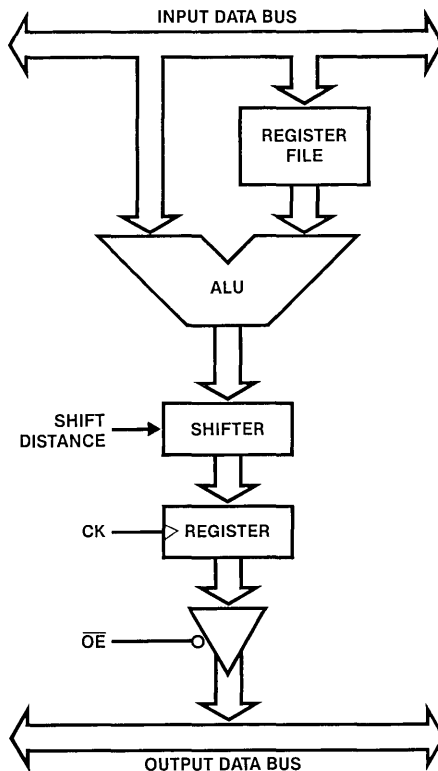
03862A-81

Figure 1. Typical ALU Architecture

DESIGN APPROACH

An 8-bit registered barrel shifter would take at least 8 data inputs, 8 registered data outputs, and 3 control lines to implement (see Figure 2). The AmPAL16R8 has 8 registered data outputs but only 8 inputs, not enough to implement the 8-bit registered barrel shifter in a single device. In cases like this, where the function to be implemented can't fit into a single PAL because of pin limitations, the problem is best approached by dividing it into smaller functions. These smaller functions should be chosen in such a way that each function requires a reduced number of pins, hence, easing the pin limitation problem. One possible approach would be to "slice" the 8-bit function into two 4-bit functions. Figure 3 shows the approach used to divide the part into two sections, each containing a smaller number of pins.

Notice that every data and control input participates in each output function. Thus the 4-bit partition must include all data and control inputs, giving a total of 11 array inputs and 4 outputs per device. If both devices are programmed identically, the 8-bit function can be implemented as shown in Figure 4. The data inputs are "offset" by 4 places to differentiate between the upper and lower nibbles.

A 4-bit slice of the 8-bit barrel shifter can be implemented in a AmPAL16R4. This part allows up to 12 inputs (one more than necessary) and 4 registered outputs. The extra input is used for a set function, loading the output register with all ones. The logic diagram and PALASM listing for the registered barrel shifter slice are shown following.

The PAL solution requires only two 20-pin packages. An MSI version of the 8-bit barrel shifter would require 4 Am25S10 four-bit shifters and an Am25S374 octal three state register. The savings in cost, board space and power are considerable, and the PAL solution is also faster.
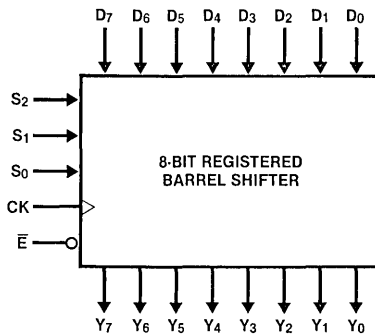
FUNCTION DEFINITION

CONTROL INPUTS			INPUT TO OUTPUT MAPPING							
S_2	S_1	S_0	Y_7	Y_6	Y_5	Y_4	Y_3	Y_2	Y_1	Y_0
0	0	0	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
0	0	1	D_6	D_5	D_4	D_3	D_2	D_1	D_0	D_7
0	1	0	D_5	D_4	D_3	D_2	D_1	D_0	D_7	D_6
0	1	1	D_4	D_3	D_2	D_1	D_0	D_7	D_6	D_5
1	0	0	D_3	D_2	D_1	D_0	D_7	D_6	D_5	D_4
1	0	1	D_2	D_1	D_0	D_7	D_6	D_5	D_4	D_3
1	1	0	D_1	D_0	D_7	D_6	D_5	D_4	D_3	D_2
1	1	1	D_0	D_7	D_6	D_5	D_4	D_3	D_2	D_1

03862A-82

Figure 2. 8-Bit Registered Barrel Shifter
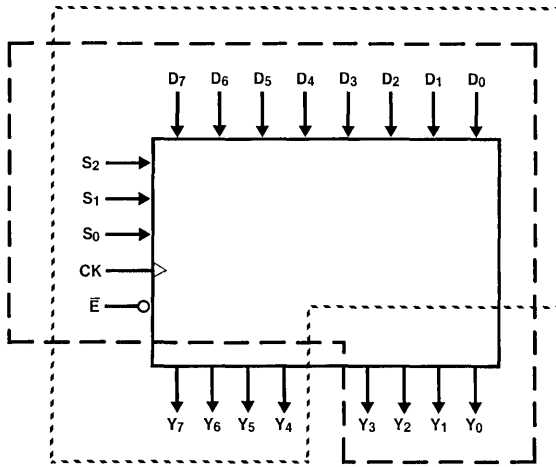
Figure 3. 4-Bit Slice Approach
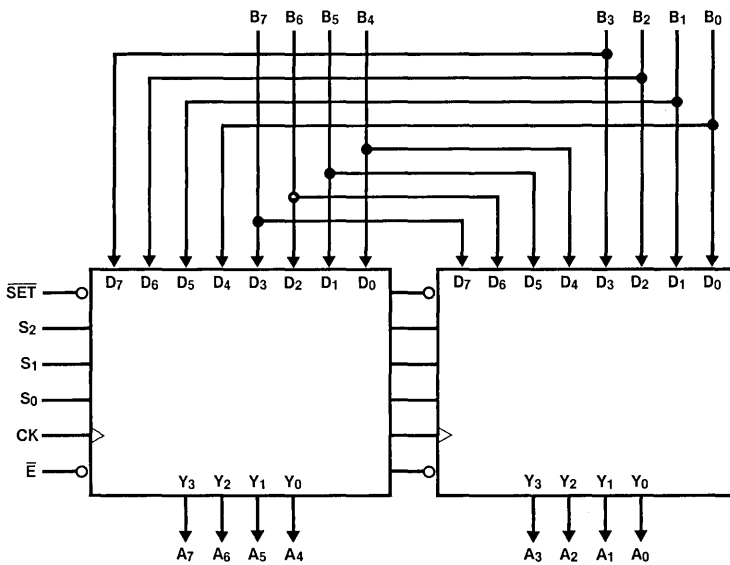
Figure 4. 8-Bit Barrel Shifter Implementation with Two 4-Bit Slices

```
PAL16R4                                     PAL DESIGN SPECIFICATION
PAT001                                      KEVIN M. OW-WING 6/22/82
4-BIT SLICE FOR AN 8-BIT BARREL SHIFTER
ADVANCED MICRO DEVICES
CK D7   D6 D5 D4 D3 D2 D1 DO GND
/E /SET SO Q0 Q1 Q2 Q3 S1 S2 VCC

/Q3 := /SET*/S2*/S1*/SO*/D3 +
       /SET*/S2*/S1* SO*/D2 +
       /SET*/S2* S1*/SO*/D1 +
       /SET*/S2* S1* SO*/D0 +
       /SET* S2*/S1*/SO*/D7 +
       /SET* S2*/S1* SO*/D6 +
       /SET* S2* S1*/SO*/D5 +
       /SET* S2* S1* SO*/D4

/Q2 := /SET*/S2*/S1*/SO*/D2 +
       /SET*/S2*/S1* SO*/D1 +
       /SET*/S2* S1*/SO*/D0 +
       /SET*/S2* S1* SO*/D7 +
       /SET* S2*/S1*/SO*/D6 +
       /SET* S2*/S1* SO*/D5 +
       /SET* S2* S1*/SO*/D4 +
       /SET* S2* S1* SO*/D3

/Q1 := /SET*/S2*/S1*/SO*/D1 +
       /SET*/S2*/S1* SO*/D0 +
       /SET*/S2* S1*/SO*/D7 +
       /SET*/S2* S1* SO*/D6 +
       /SET* S2*/S1*/SO*/D5 +
       /SET* S2*/S1* SO*/D4 +
       /SET* S2* S1*/SO*/D3 +
       /SET* S2* S1* SO*/D2

/Q0 := /SET*/S2*/S1*/SO*/D0 +
       /SET*/S2*/S1* SO*/D7 +
       /SET*/S2* S1*/SO*/D6 +
       /SET*/S2* S1* SO*/D5 +
       /SET* S2*/S1*/SO*/D4 +
       /SET* S2*/S1* SO*/D3 +
       /SET* S2* S1*/SO*/D2 +
       /SET* S2* S1* SO*/D1
```

FUNCTION TABLE

CK	/E	/SET	S2	S1	S0	D7	D6	D5	D4	D3	D2	D1	D0	Q3	Q2	Q1	Q0
;																	
; HIGH Z TEST																	
X	H	X	X	X	X	X	X	X	X	X	X	X	X	Z	Z	Z	Z
;																	
; SET OUTPUTS TEST																	
C	L	L	X	X	X	X	X	X	X	X	X	X	X	H	H	H	H
;																	
; PSEUDO RANDOM BARREL SHIFTER TEST SEQUENCE																	
C	L	H	L	L	L	L	L	L	L	L	L	L	H	L	L	L	H
C	L	H	L	L	H	L	L	L	L	L	L	H	L	L	H	L	L
C	L	H	L	H	L	H	L	L	L	L	L	L	L	L	L	H	L
C	L	H	L	H	H	L	L	H	L	L	L	L	L	L	L	L	H
C	L	H	H	L	L	L	H	L	L	L	L	L	L	L	H	L	L
C	L	H	H	L	H	L	L	L	H	L	L	L	L	L	L	H	L
C	L	H	H	H	L	L	L	H	L	L	L	L	L	H	L	L	L
C	L	H	H	H	H	L	L	L	L	L	L	H	L	L	L	L	H
C	L	H	L	L	H	L	L	L	L	L	H	L	L	H	L	L	L
C	L	H	H	H	H	L	L	L	L	H	L	L	L	L	H	L	L

DESCRIPTION: THIS PART IS A 4-BIT SLICE OF AN 8-BIT REGISTERED
BARREL SHIFTER. IT TAKES 8 DATA INPUTS AND CYCLICALLY ROTATES THE
DATA FROM 0 TO 7 PLACES UNDER CONTROL OF THE SELECT (S) INPUTS. A
SET INPUT CAN BE USED TO INITIALIZE THE OUTPUTS TO THE ALL ONES
STATE.

PAL16R4   PAL DESIGN SPECIFICATION
PAT001   KEVIN M. OW-WING 6/22/82
4-BIT SLICE FOR AN 8-BIT BARREL SHIFTER
ADVANCED MICRO DEVICES
*D9724
F0
L0512 1110 1110 1111 1111 1011 1111 1110 1101 *
L0544 1110 1110 1111 1111 1111 1011 1101 1101 *
L0576 1110 1101 1111 1111 1111 1111 1010 1101 *
L0608 1110 1101 1111 1111 1111 1111 1101 1001 *
L0640 1001 1110 1111 1111 1111 1111 1110 1101 *
L0672 1101 1010 1111 1111 1111 1111 1101 1101 *
L0704 1101 1101 1011 1111 1111 1111 1110 1101 *
L0736 1101 1101 1111 1011 1111 1111 1101 1101 *
L0768 1110 1110 1111 1111 1111 1011 1110 1101 *
L0800 1110 1110 1111 1111 1111 1111 1001 1101 *
L0832 1110 1101 1111 1111 1111 1111 1110 1001 *
L0864 1010 1101 1111 1111 1111 1111 1101 1101 *
L0896 1101 1010 1111 1111 1111 1111 1110 1101 *
L0928 1101 1110 1011 1111 1111 1111 1101 1101 *
L0960 1101 1101 1111 1011 1111 1111 1110 1101 *
L0992 1101 1101 1111 1111 1011 1111 1101 1101 *
L1024 1110 1110 1111 1111 1111 1111 1010 1101 *
L1056 1110 1110 1111 1111 1111 1111 1101 1001 *
L1088 1010 1101 1111 1111 1111 1111 1110 1101 *
L1120 1110 1001 1111 1111 1111 1111 1101 1101 *
L1152 1101 1110 1011 1111 1111 1111 1110 1101 *
L1184 1101 1110 1111 1011 1111 1111 1101 1101 *
L1216 1101 1101 1111 1111 1011 1111 1110 1101 *
L1248 1101 1101 1111 1111 1111 1011 1101 1101 *
L1280 1110 1110 1111 1111 1111 1111 1110 1001 *
L1312 1010 1110 1111 1111 1111 1111 1101 1101 *
L1344 1110 1001 1111 1111 1111 1111 1110 1101 *
L1376 1110 1101 1011 1111 1111 1111 1101 1101 *
L1408 1101 1110 1111 1011 1111 1111 1101 1101 *
L1440 1101 1110 1111 1111 1011 1111 1101 1101 *
L1472 1101 1101 1111 1111 1111 1011 1110 1101 *
L1504 1101 1101 1111 1111 1111 1111 1001 1101 *
C67E0*
V0001 XXXXXXXXX01XXZZZZXX1 *
V0002 CXXXXXXXX000XHHHHXX1 *
V0003 C000000010010HLLL001 *
V0004 C000000100011LLHL001 *
V0005 C100000000010LHLL101 *
V0006 C001000000011HLLL101 *
V0007 C010000000010LLHL011 *
V0008 C000100000011LHLL011 *
V0009 C001000000010LLLH111 *
V0010 C000000100011HLLL111 *
V0011 C000001000011LLLH001 *
V0012 C000010000011LLHL111 *
994E

LOGIC DIAGRAM FOR:
4-BIT SLICE REGISTERED BARREL SHIFTER USING AmPAL16R4

Columns: Inputs (0–31)
Rows: Product Terms (0–63)

CK

D_7 D_6 D_5 D_4 D_3 D_2 D_1 D_0

S_2 S_1 S_0

Y_3 Y_2 Y_1 Y_0

$\overline{SET}$ $\overline{E}$

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

—✳— = Fuse intact   —⊠— = All fuses intact   —┼— = Fuse blown

03862A-85

5-7

Dynamic Memory Control State Sequencer

by Brad Kitson
Advanced Micro Devices

An example of a control path application for an AMD PAL is in a memory system. Most large memory systems use MOS dynamic RAMs. Their high density allows packing a large memory size into a small board area. Dynamic RAM prices also make them very cost effective.

Dynamic RAMs require external logic for address multiplexing, timing generation and refresh control. This application note shows the use of an AmPAL16R8A and an Am2964B to provide the necessary external logic for a typical dynamic memory system. The PAL is used as a state sequencer for timing generation and the Am2964B provides specialized control circuitry and reduces timing skew between control signals. This implementation replaces about 20 SSI/MSI packages.

DESIGN REQUIREMENTS

A system block diagram is shown in Figure 1. The control bus provides most of the inputs to the PAL state sequencer. These include: Memory Request ($\overline{\text{MREQ}}$), READ/$\overline{\text{WRITE}}$ ($R\overline{W}$), RESET ($\overline{\text{RST}}$), Refresh Clock (RFCK), and Read-Modify-Write (RMW). Two upper address lines of the address bus serve as board selects (BS$_1$, BS$_0$), and one local signal, $\overline{\text{SLOW}}$/FAST Memory (FAST), allows use of either slow or fast memory. A READ/$\overline{\text{WRITE}}$ sequence is initialized by $\overline{\text{MREQ}}$ ANDed with the proper board select conditions and a refresh sequence is initialized by RFCK. If both sequences are requested at the same time, a refresh sequence is performed. $R\overline{W}$ when HIGH selects a READ operation and when LOW selects a WRITE operation. RMW when HIGH selects a Read-Modify-Write cycle.
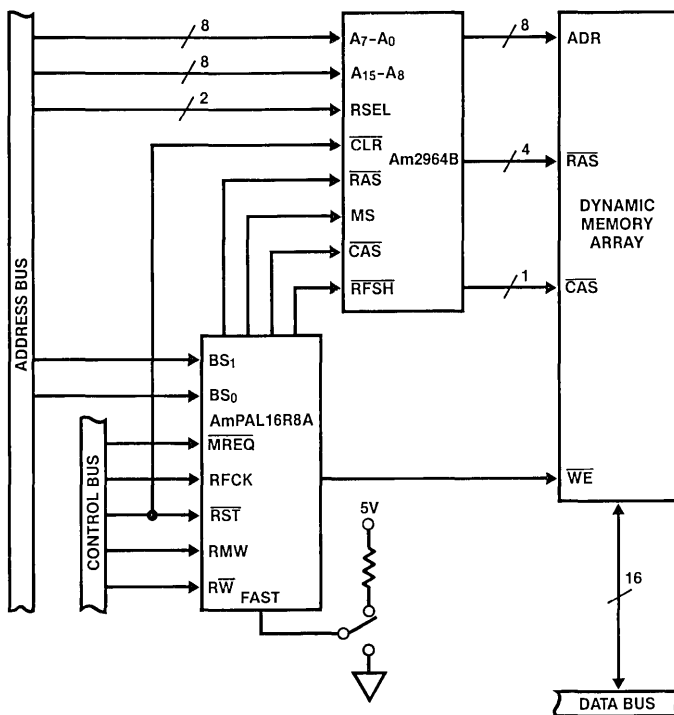
Figure 1. Dynamic Memory Controller

03862A-86

The outputs of the PAL provide the timing and control inputs to the Am2964B. These are: Row Address Strobe ($\overline{RAS}$), Address Multiplexer Select (MS), Column Address Strobe ($\overline{CAS}$), and Refresh ($\overline{RFSH}$). In addition, the PAL provides the Write Enable ($\overline{WE}$) to the Memory Array. Figure 2 shows the timing for fast READ/WRITE cycles. The memory cycle is initiated by $\overline{MREQ}$ going LOW. The PAL responds by bringing $\overline{RAS}$ LOW at t_0, followed by MS going LOW at t_1, and finally bringing $\overline{CAS}$ LOW at t_2. If RW is LOW, $\overline{WE}$ is also brought LOW at t_2. $\overline{WE}$ is held LOW until t_4. $\overline{RAS}$, MS and $\overline{CAS}$ are brought HIGH at t_5. The rising edge of any of these 3 signals may be used to latch output data during a Read operation. The state sequencer is then disabled for 3 states to allow for memory precharge.

By holding the FAST input LOW, an extended memory cycle is available to accommodate slower RAMs. The timing appears in Figure 3.
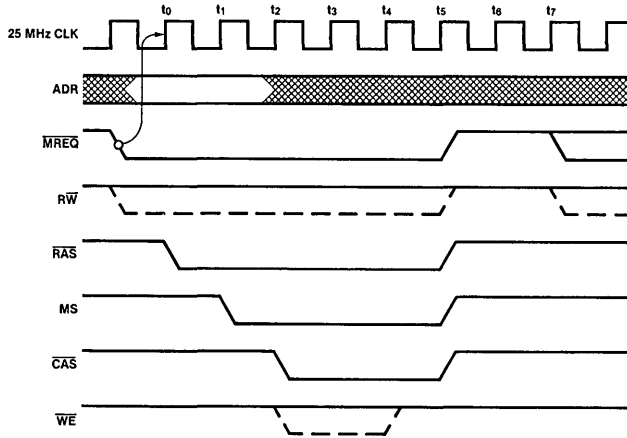
03862A-87

Figure 2. Fast READ/WRITE Cycle
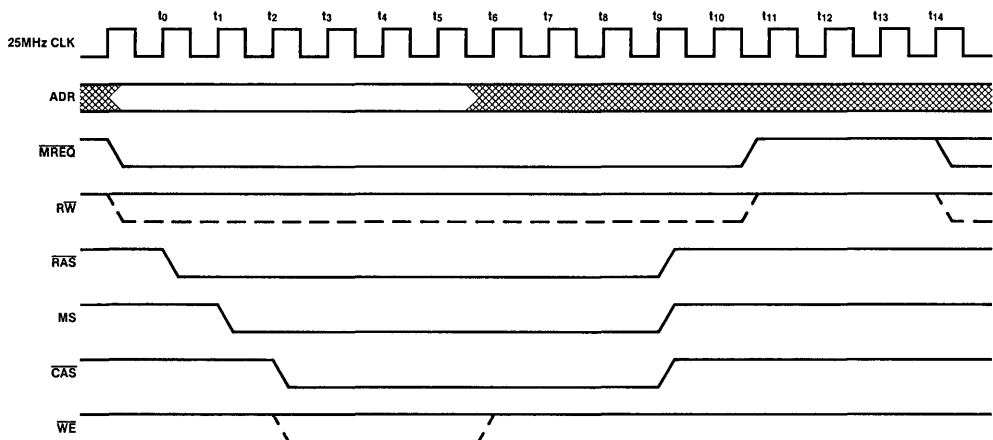
03862A-88

Figure 3. Extended Memory Cycle

$\overline{RAS}$-Only refresh cycle timing is shown in Figure 4. The refresh cycle is initiated when RFCK goes HIGH. The $\overline{RFSH}$ output goes LOW at t_0, followed by $\overline{RAS}$ at t_1. The Am2964B supplies the necessary refresh address. $\overline{RAS}$ is brought back HIGH at t_5 and precharge is then timed out. An extended refresh cycle for slower memory is available also. Burst refresh can be accomplished by leaving RFCK HIGH for as many refresh cycles as desired.

Read-Modify-Write cycle timing is activated by setting RMW HIGH. This is especially valuable in systems with Error Detection/Correction (EDC) capability. Data can be read, modified by the EDC circuitry (Am2960), and if necessary, written back to memory in a single memory cycle. Read-Modify-Write cycle timing is shown in Figure 5. Note that $\overline{WE}$ goes LOW at the end of the cycle.
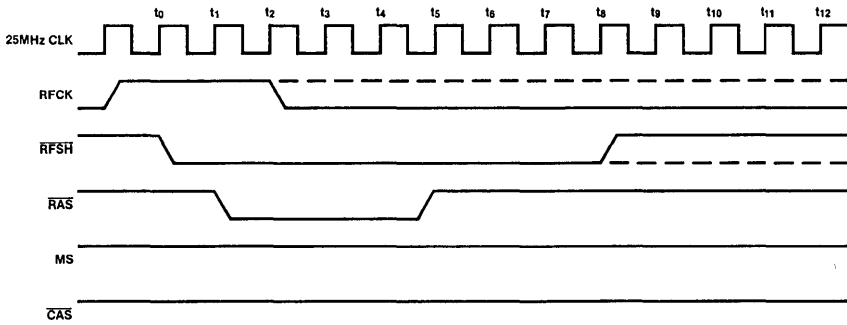
03862A-89

Figure 4. $\overline{RAS}$-Only Refresh Cycle
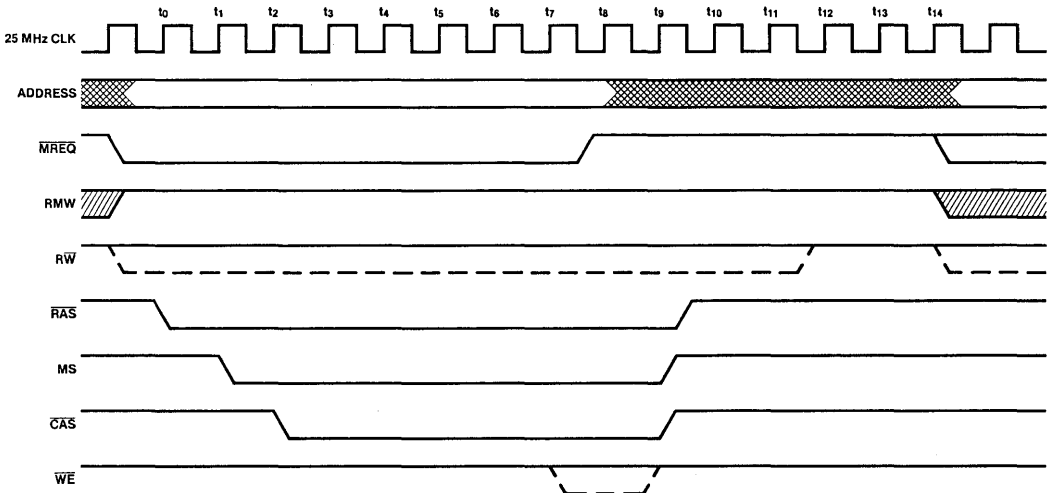
03862A-90

Figure 5. Read-Modify-Write Cycle

DESIGN APPROACH

The first step in the state sequencer design process is to define the timing waveforms for all of the functions desired. Figures 2, 3, 4 and 5 are the result. Next, characteristics of the resulting waveforms are examined. Initially, the sequencer is waiting on the $\overline{\text{MREQ}}$ or RFCK input. If $\overline{\text{MREQ}}$ goes LOW, the $\overline{\text{RAS}}$ to MS to $\overline{\text{CAS}}$ sequence is initiated. If RFCK goes HIGH, the $\overline{\text{RFSH}}$ to $\overline{\text{RAS}}$ sequence is initiated. Both sequences are equivalent to a simple "shift" function. Once the shift sequence is completed and the signals are asserted, they must stay asserted for a specific time depending on the selected function. To time the length that signals must stay asserted

requires a "counting" function. The precharge sequence at the end of all cycles also requires "counting". This partitions most of the design into two smaller functional blocks; a shifter and a counter. The remaining function select and control logic is partitioned into a "multiplexer-like" functional block. Figure 6 shows the PAL partitioned into functional blocks. By dividing the design into blocks, its implementation becomes simple.

The following pages show the easy to read PAL DESIGN SPECIFICATION and a logic diagram for the AmPAL16R8A dynamic memory state sequencer.
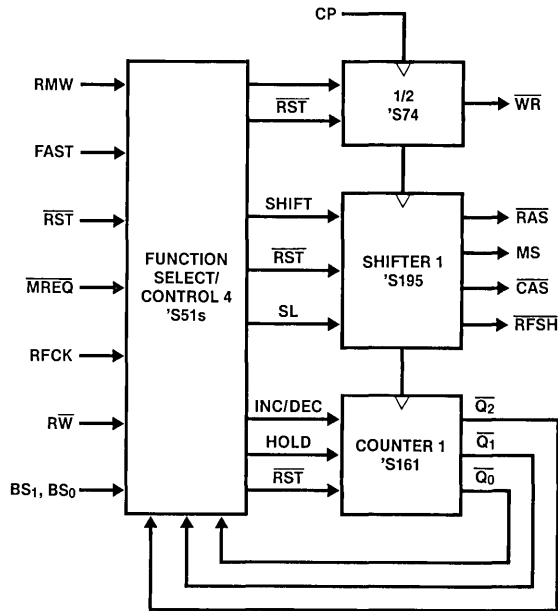
03862A-91

Figure 6. Partitioned Design/PAL Equivalent

```
PAL16R8                              PAL DESIGN SPECIFICATION
PAT002                               BRAD S. KITSON   2/10/82
DYNAMIC MEMORY CONTROL STATE SEQUENCER
ADVANCED MICRO DEVICES
CK RFCK /RST RW  /MREQ RMW FAST  BS1 BSO  GND
/E /QO  /Q1   /Q2 /RFSH /WE /CAS MS  /RAS VCC

QO  :=  /RST*  /MS*/QO          +
        /RST* RFSH*RAS*/QO       +
        /RST*/FAST*/QO*Q2        +
        /RST*/FAST*/QO*Q1        +
        /RST*/FAST* Q1*Q2        +
        /RST* FAST*/RMW*QO*/Q1 +
        /RST* FAST*/RMW*QO*/Q2

Q1  :=  /RST* RAS*/QO* Q1       +
        /RST* RAS* QO*/Q1       +
        /RST*/RAS* QO* Q1       +
        /RST*/RAS*/QO* Q2

Q2  :=  /RST*RAS*Q2             +
        /RST* QO*Q2             +
        /RST*RAS*QO*Q1

RFSH := /RST*RFCK*/Q2*/Q1*/QO*/RAS +
        /RST*RFSH*RAS            +
        /RST*RFSH*/FAST* Q1     +
        /RST*RFSH* Q2

WE  :=  /RST*/RW*/MS*/RFSH*/RMW*/QO*/Q2     +
        /RST*/RW*/MS*/RFSH*/RMW*/Q1*/Q2     +
        /RST*/RW*/MS*/RFSH* RMW*/QO* Q1*Q2 +
        /RST*/RW*/MS*/RFSH* RMW* QO*/Q1*Q2

CAS :=  /RST*/RFSH*/MS*/QO      +
        /RST*/RFSH*/MS*/Q1      +
        /RST*/RFSH*/MS*/Q2

/MS :=  /RST*/RFSH*RAS*/QO      +
        /RST*/RFSH*RAS*/Q1      +
        /RST*/RFSH*RAS*/Q2

RAS :=  /RST*/RFCK*/QO*/Q1*/Q2*MREQ*/BS1*/BSO +
        /RST*/RFSH*/QO*/Q1*/Q2*MREQ*/BS1*/BSO +
        /RST* RFSH*/QO*/Q1*/Q2 +
        /RST*RAS*/QO            +
        /RST*RAS*/Q1            +
        /RST*RAS*/Q2
```

FUNCTION TABLE

CK	/E	/RST	/MREQ	BS1	BSO	RFCK	RW	RMW	FAST	RAS	/MS	CAS	WE	RFSH	Q2	Q1	QO
;																	
;INITIALIZE																	
C	L	L	X	X	X	X	X	X	X	L	L	L	L	L	L	L	L
;																	
;FAST WRITE OPERATION																	
C	L	H	L	L	L	L	X	X	X	H	L	L	L	L	L	L	L
C	L	H	X	X	X	X	X	X	X	H	H	L	L	L	L	L	L
C	L	H	X	X	X	X	L	L	H	H	H	H	H	L	L	L	H
C	L	H	X	X	X	X	L	L	H	H	H	H	H	L	L	H	H
C	L	H	X	X	X	X	L	L	H	H	H	H	L	L	H	L	H
C	L	H	X	X	X	X	L	L	H	H	H	H	L	L	H	H	H
C	L	H	X	X	X	X	L	L	H	L	L	L	L	L	H	L	L
C	L	H	X	X	X	X	L	L	H	L	L	L	L	L	L	H	L
C	L	H	X	X	X	X	L	L	H	L	L	L	L	L	L	L	L
;																	
;RAS ONLY REFRESH CYCLE																	
C	L	H	H	X	X	H	X	X	X	L	L	L	L	H	L	L	L
C	L	H	X	X	X	H	X	X	X	H	L	L	L	H	L	L	L
C	L	H	X	X	X	X	X	X	H	H	L	L	L	H	L	L	H
C	L	H	X	X	X	X	X	X	H	H	L	L	L	H	L	H	H
C	L	H	X	X	X	X	X	X	H	H	L	L	L	H	H	L	H
C	L	H	X	X	X	X	X	X	H	H	L	L	L	H	H	H	H
C	L	H	X	X	X	X	X	X	H	L	L	L	L	H	H	L	L
C	L	H	X	X	X	X	X	X	H	L	L	L	L	H	L	H	L
C	L	H	X	X	X	X	X	X	H	L	L	L	L	L	L	L	L
;																	
;READ-MODIFY-WRITE OPERATION																	
C	L	H	L	L	L	L	X	X	X	H	L	L	L	L	L	L	L
C	L	H	X	X	X	X	X	X	X	H	H	L	L	L	L	L	L
C	L	H	X	X	X	X	L	H	L	H	H	H	L	L	L	L	H
C	L	H	X	X	X	X	L	H	L	H	H	H	L	L	L	H	L
C	L	H	X	X	X	X	L	H	L	H	H	H	L	L	L	H	H
C	L	H	X	X	X	X	L	H	L	H	H	H	L	L	H	L	L
C	L	H	X	X	X	X	L	H	L	H	H	H	L	L	H	L	H
C	L	H	X	X	X	X	L	H	L	H	H	H	H	L	H	H	L
C	L	H	X	X	X	X	L	H	L	H	H	H	H	L	H	H	H
C	L	H	X	X	X	X	L	H	L	L	L	L	L	L	H	L	H
C	L	H	X	X	X	X	L	H	L	L	L	L	L	L	H	L	L
C	L	H	X	X	X	X	L	H	L	L	L	L	L	L	L	H	H

DESCRIPTION
DYNAMIC MEMORY CONTROL STATE SEQUENCER FOR USE WITH THE AM2964B MEMORY
CONTROLLER. THE SEQUENCER PROVIDES /RAS,MS,/CAS, & REFRESH TIMING
GENERATION TO THE AM2964B AND /WE TO THE DRAMS. IT SUPPORTS BOTH FAST
(150NS) AND SLOW (300NS) READ/WRITE CYCLES, /RAS ONLY REFRESH, BURST
REFRESH, AND READ-MODIFY-WRITE FOR MEMORY BOARDS OF UP TO 256K.

```
PAL16R8                                    PAL DESIGN SPECIFICATION
PAT002                                     BRAD S. KITSON    2/10/82
DYNAMIC MEMORY CONTROL STATE SEQUENCER
ADVANCED MICRO DEVICES
*D9724
*F0*
L0000 1011 0111 1111 1011 1111 1101 1001 1001 *
L0032 1111 0111 1111 1011 1101 1101 1001 1001 *
L0064 1111 0111 1111 1111 1110 1101 1101 1101 *
L0096 1110 0111 1111 1111 1111 1111 1111 1101 *
L0128 1110 0111 1111 1111 1111 1111 1101 1111 *
L0160 1110 0111 1111 1111 1111 1101 1111 1111 *
L0256 1110 0111 1111 1111 1101 1111 1111 1101 *
L0288 1110 0111 1111 1111 1101 1111 1101 1111 *
L0320 1110 0111 1111 1111 1101 1101 1111 1111 *
L0512 1111 0110 1111 1111 1101 1111 1111 1101 *
L0544 1111 0110 1111 1111 1101 1111 1101 1111 *
L0576 1111 0110 1111 1111 1101 1101 1111 1111 *
L0768 1111 0110 1011 1111 1001 1101 1111 1101 *
L0800 1111 0110 1011 1111 1001 1101 1101 1111 *
L0832 1111 0110 1011 1111 0101 1110 1110 1101 *
L0864 1111 0110 1011 1111 0101 1110 1101 1110 *
L1024 0101 0111 1111 1111 1111 1101 1101 1101 *
L1056 1110 0111 1111 1111 1110 1111 1111 1111 *
L1088 1111 0111 1111 1111 1110 1011 1110 1111 *
L1120 1111 0111 1111 1111 1110 1110 1111 1111 *
L1280 1110 0111 1111 1111 1111 1110 1111 1111 *
L1312 1111 0111 1111 1111 1111 1110 1111 1110 *
L1344 1110 0111 1111 1111 1111 1111 1110 1110 *
L1536 1110 0111 1111 1111 1111 1111 1110 1101 *
L1568 1110 0111 1111 1111 1111 1111 1101 1110 *
L1600 1101 0111 1111 1111 1111 1111 1110 1110 *
L1632 1101 0111 1111 1111 1111 1110 1111 1101 *
L1792 1111 0110 1111 1111 1111 1111 1111 1101 *
L1824 1110 0111 1111 1111 1110 1111 1111 1101 *
L1856 1111 0111 1111 1111 1111 1010 1111 1101 *
L1888 1111 0111 1111 1111 1111 1011 1110 1101 *
L1920 1111 0111 1111 1111 1111 1010 1110 1111 *
L1952 1111 0111 1111 1111 1011 0111 1101 1110 *
L1984 1111 0111 1111 1111 1011 0101 1111 1110 *
C713B*
V0001 CX0XXXXXX00HHHHHHHH1 *
V0002 C01X0XX0000HHHHHHHL1 *
V0003 CX1XXXXXX00HHHHHHLL1 *
V0004 CX10X01XX00LHHHLLLL1 *
V0005 CX10X01XX00LLHHLLLL1 *
V0006 CX10X01XX00LHLHHLLL1 *
V0007 CX10X01XX00LLLHHLLL1 *
V0008 CX10X01XX00HHLHHHHH1 *
V0009 CX10X01XX00HLHHHHHH1 *
V0010 CX10X01XX00HHHHHHHH1 *
V0011 C11X1XXXX00HHHLHHHH1 *
V0012 C11XXXXXX00HHHLHHHL1 *
V0013 CX1XXX1XX00LHHLHHHL1 *
V0014 CX1XXX1XX00LLHLHHHL1 *
```
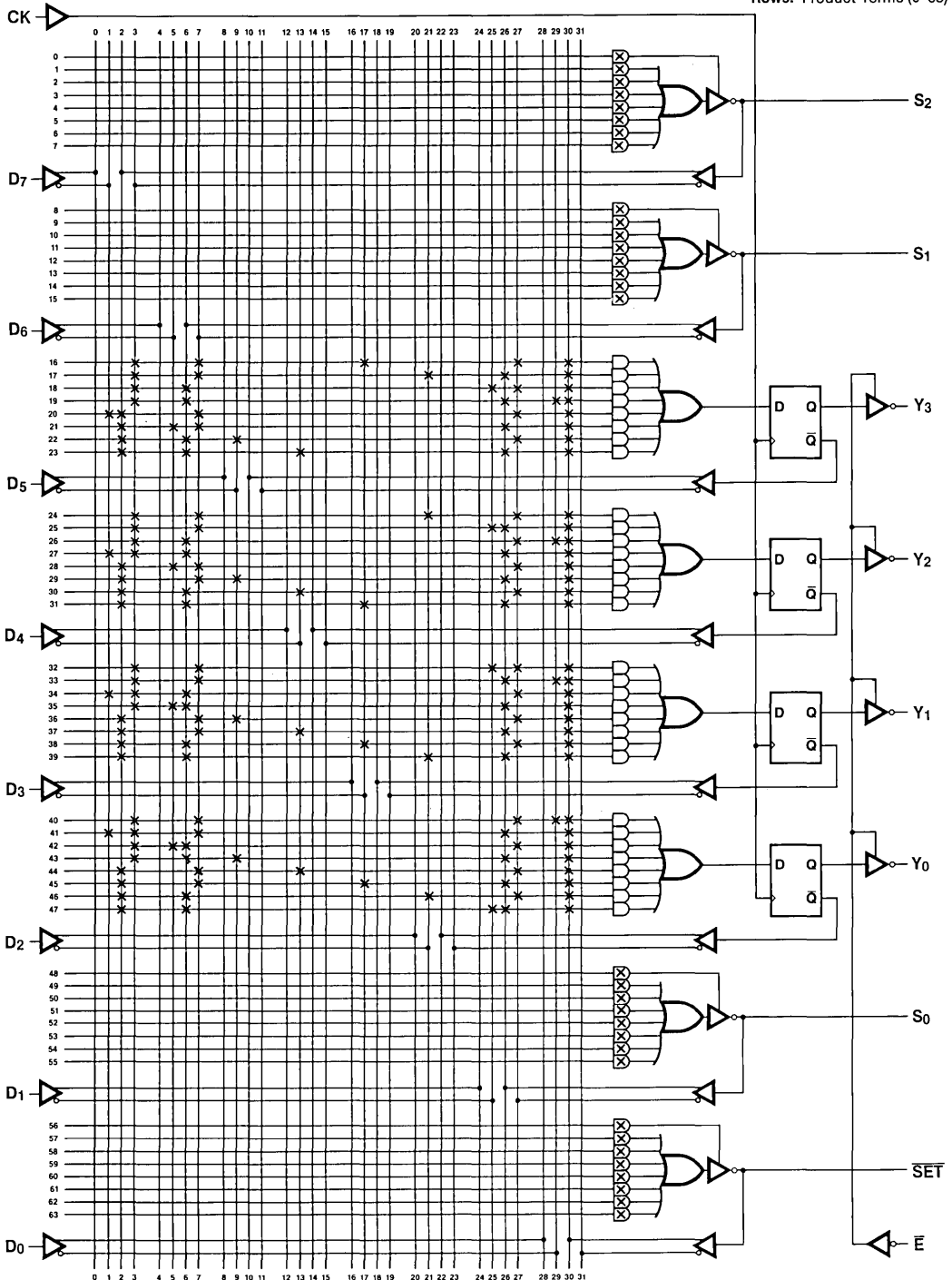
```
V0015  CX1XXX1XX00LHLLHHHL1  *
V0016  CX1XXX1XX00LLLLHHHL1  *
V0017  CX1XXX1XX00HHLLHHHH1  *
V0018  CX1XXX1XX00HLHLHHHH1  *
V0019  CX1XXX1XX00HHHHHHHH1  *
V0020  C01X0XX0000HHHHHHHL1  *
V0021  CX1XXXXXX00HHHHHHLL1  *
V0022  CX10X10XX00LHHHHLLL1  *
V0023  CX10X10XX00HLHHHLLL1  *
V0024  CX10X10XX00LLHHHLLL1  *
V0025  CX10X10XX00HHLHHLLL1  *
V0026  CX10X10XX00LHLHHLLL1  *
V0027  CX10X10XX00HLLHLLLL1  *
V0028  CX10X10XX00LLLHLLLL1  *
V0029  CX10X10XX00LHLHHHHH1  *
V0030  CX10X10XX00HHLHHHHH1  *
V0031  CX10X10XX00LLHHHHHH1  *
395F
```

LOGIC DIAGRAM FOR:
DYNAMIC MEMORY CONTROL STATE SEQUENCER USING AmPAL16R8A

Columns: Inputs (0–31)
Rows: Product Terms (0–63)

CK

RFCK

RST

RW

MREQ

RMW

FAST

BS₁

BS₀

RAS

MS

CAS

WE

RFSH

Q₂

Q₁

Q₀

E

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

✳ = Fuse intact   ⊠ = All fuses intact   ┼ = Fuse blown

03862A-92

5-17
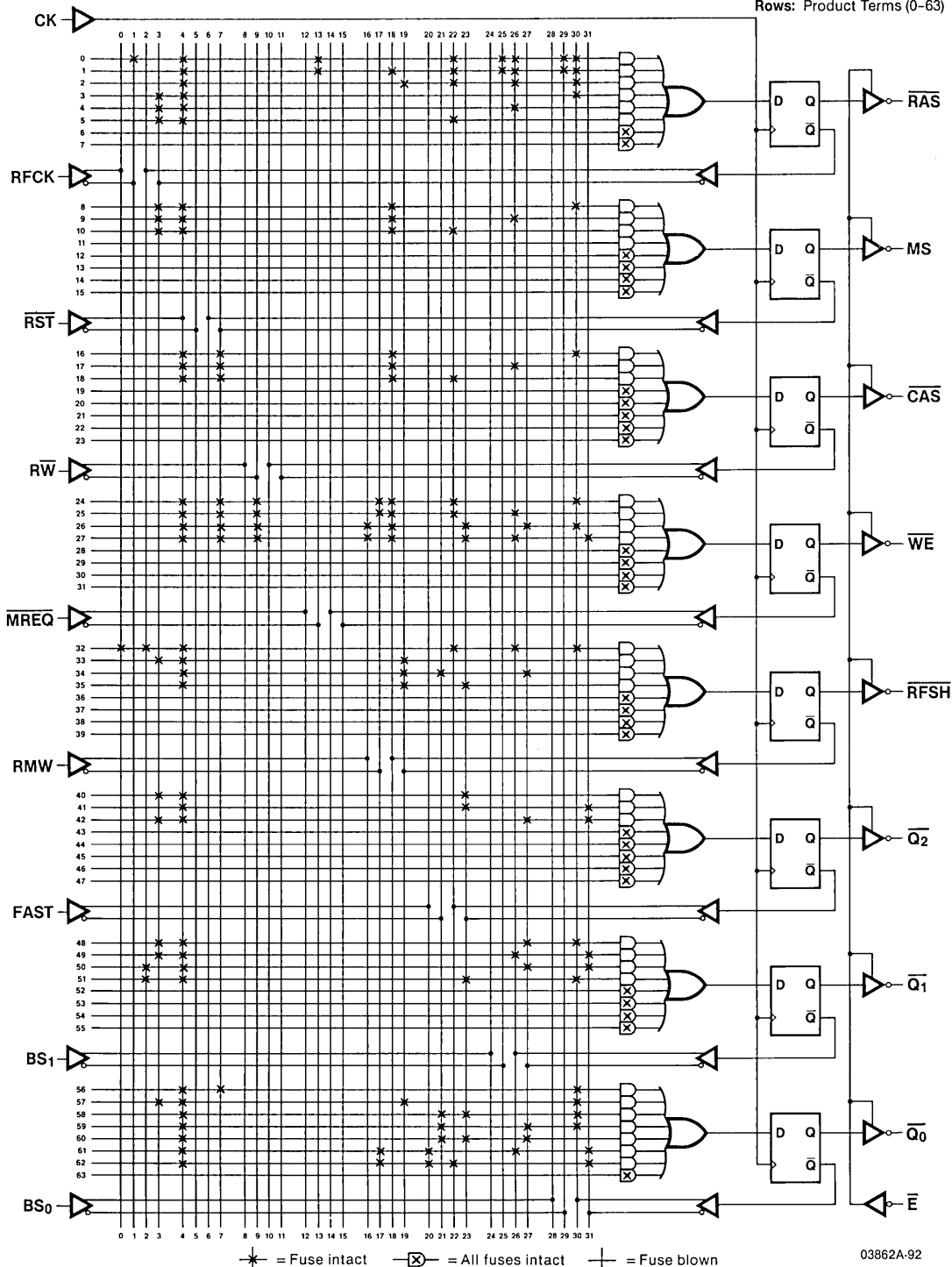
GCR (4B-5B) Encoder/Decoder

by Warren Miller
Advanced Micro Devices

One of the more common logic functions performed on serial data is the data encode/decode function. Usually it is desirable to map (encode) the logical bit stream to a physical bit stream, adjusting for the peculiarities of the particular transmission or storage media.

Noise, bandwidth, and reliability considerations may mean that a different data format would be desirable when data is sent along to or stored on a given media. For example, group-coded recording (GCR) formats take a given number of data bits and encode them with a larger number of bits. A 4B-5B GCR code would take 4 data bits and encode them into 16 states with 5 new bits. A particular 4B-5B code is shown in Table 1.

This mapping allows at most two zeros to occur in succession. Also note that data combinations with more than one zero at the beginning and end of the word are excluded. This is necessary to insure that when data words are serialized, no more than two zeros occur in succession at any point in the bit stream. Finally, the data combination 11111 is reserved as a synchronization mark. In tape systems, this results in increased bit density and eases clock synchronization.

Table 1. 4B-5B Code

4-Bit Data	5-Bit Data
0 0 0 0	1 1 0 0 1
0 0 0 1	1 1 0 1 1
0 0 1 0	1 0 0 1 0
0 0 1 1	1 0 0 1 1
0 1 0 0	1 1 1 0 1
0 1 0 1	1 0 1 0 1
0 1 1 0	1 0 1 1 0
0 1 1 1	1 0 1 1 1
1 0 0 0	1 1 0 1 0
1 0 0 1	0 1 0 0 1
1 0 1 0	0 1 0 1 0
1 0 1 1	0 1 0 1 1
1 1 0 0	1 1 1 1 0
1 1 0 1	0 1 1 0 1
1 1 1 0	0 1 1 1 0
1 1 1 1	0 1 1 1 1

03862A-93

DESIGN REQUIREMENTS

The system diagram in Figure 1 shows how the GCR Encoder/Decoder (GCR E/D) interfaces to a tape drive and tape controller. Parallel input data is given to the GCR E/D, converted to the 5-bit format, serialized, and written to the tape. On a read, the serial data from the tape is parallelized, converted back to the 4-bit format and output to the tape controller. Additionally, during a read, two status signals are developed. The first signal, $\overline{INV}$, indicates the presence of an invalid input, ie., too many zeros in succession. The second status signal, $\overline{H}$, indicates the detection of the synchronization mark (11111).

The operation modes for the GCR E/D are shown in the Data-Flow Diagrams of Figure 2. The control signal definition and operation functions are indicated for each operation mode. In particular, the data flow between each bit of the output register is indicated schematically.

The first mode of operation of the GCR E/D is the HOLD mode. When $\overline{ENABLE}$ is HIGH, all data operations on the output register are disabled, independent of the two mode controls, M_1 and M_0. The output data is simply fed-back to the register inputs. Thus the register content is retained after the clock transition.

When the $\overline{ENABLE}$ input is LOW, the operations indicated by the M_1 and M_0 mode bits are executed on the clock transition. When M_1 and M_0 are both LOW, the SERIAL SHIFT IN mode is selected. In this mode the output register is configured as a serial shift register. The serial input is consecutively shifted into the register until all 5 bits from the tape have been stored, MSB at Y_3 and LSB at SERIAL OUT.

The CONVERT SERIAL INPUT AND LOAD operation is selected when $\overline{ENABLE}$ is LOW, M_1 is HIGH and M_0 is LOW. After the 5 bits of data have been serialized by the SERIAL SHIFT IN instruction, the 5B code must be converted to a 4B code. This is accomplished by taking the outputs of the 5 register bits and converting them to 4 bits with combinatorial logic. On the clock transition, the result is loaded into the Y register. On the same clock transition that loads the converted data into the Y register, the serial input is loaded into the serial output register. Because the serial data is being read continuously, one data bit per clock transition, the conversion must be done without missing a serial data bit.

The CONVERT PARALLEL INPUT AND LOAD operation is selected when $\overline{ENABLE}$ is LOW, M_1 is HIGH and M_0 is HIGH. This mode takes the 4 input data bits and converts them to the 5 bit representation. The result is loaded into the output register on the clock transition. The LSB of the 5B representation is loaded into the Y_3 bit of the output register and the MSB is loaded into the serial output bit. This configuration, in conjunction with the next instruction, allows the serial data to be written to the tape drive one bit per clock transition.

The final operation, SERIAL SHIFT OUT, is selected when $\overline{ENABLE}$ is LOW, M_1 is LOW and M_0 is HIGH. After the CONVERT PARALLEL INPUT AND LOAD operation is executed, the SERIAL SHIFT OUT operation outputs the converted data to the tape drive. A series of one convert operation followed by 4 shift operations will transfer a sequence of 5-bits to the tape drive, one bit per clock cycle.
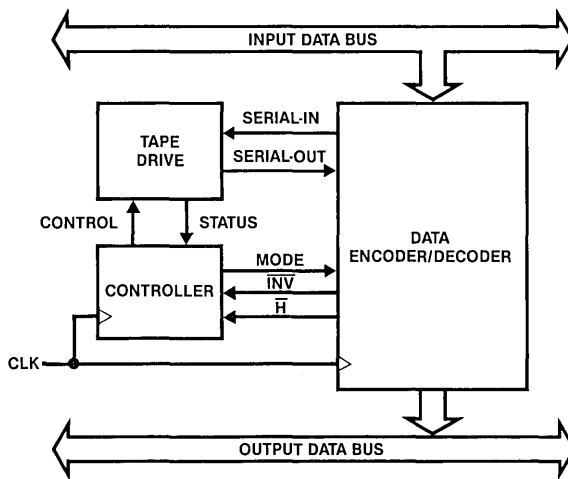
03862A-94

Figure 1. Typical Tape Storage System

ENABLE	M_1	M_0	OPERATION	DATA-FLOW DIAGRAM
1	X	X	HOLD	
0	0	0	SERIAL SHIFT IN	
0	1	0	CONVERT SERIAL INPUT AND LOAD	
0		1	CONVERT PARALLEL INPUT AND LOAD	
0	0	1	SERIAL SHIFT OUT	
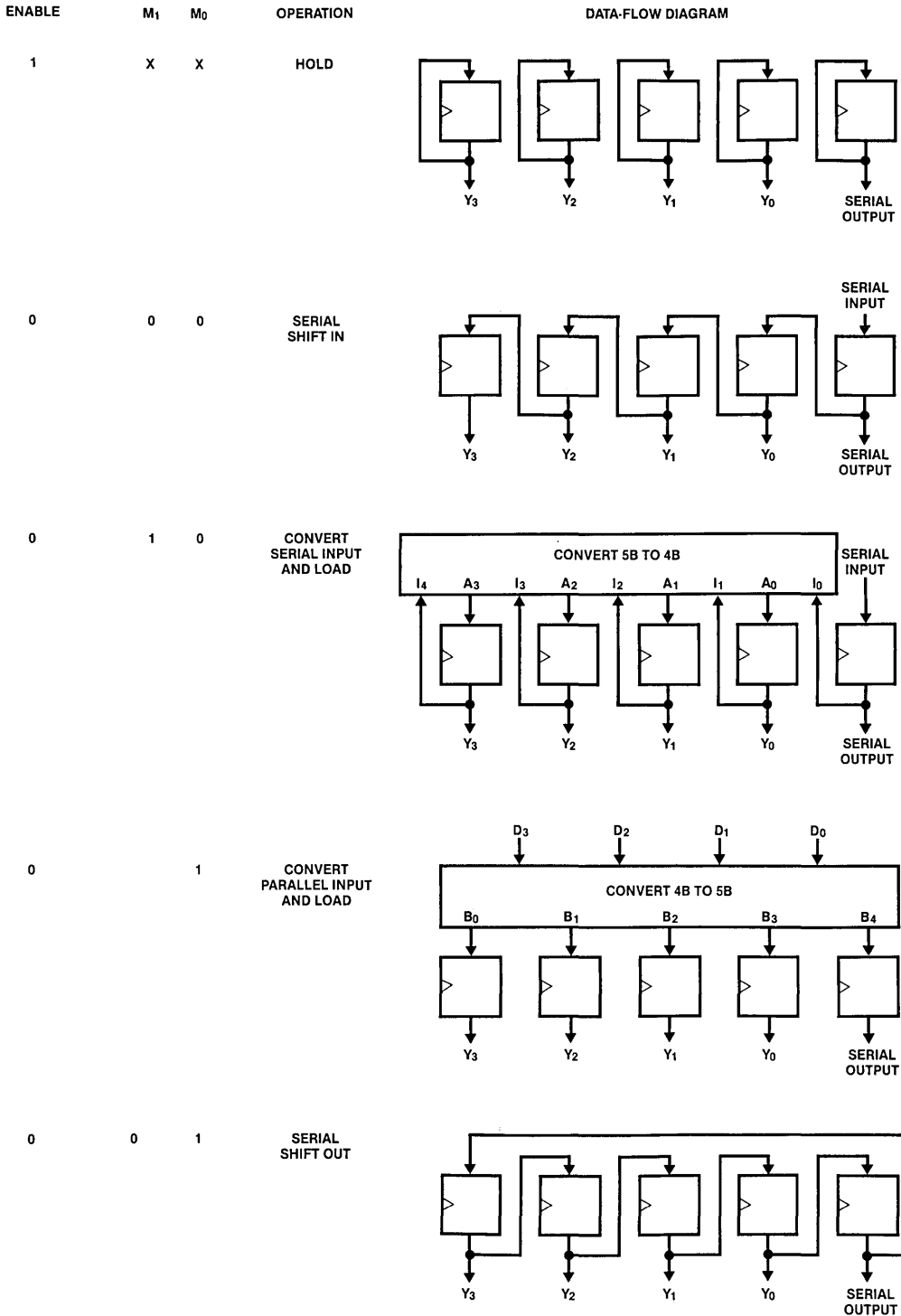
Figure 2. GCR E/D Mode Definitions

03862A-95

DESIGN APPROACH

The PAL implementation of the GCR Encoder/Decoder takes advantage of the multiplexer-like structure of the AND-OR array. Each valid combination of ENABLE, M_1 and M_0 selects a different set of AND terms. In some cases, only one term is selected (in data steering operations for example). In other cases, multiple AND terms are selected to implement a combinatorial logic function (the 5B to 4B conversion for example). This concept, using the control inputs to enable one or more AND terms, allows the direct implementation of the PAL design from the mode Data-Flow Diagrams (with a little Karnaugh map help). The K-Maps for the 5B to 4B conversion logic and the 4B to 5B conversion logic for the Y_3 output are shown in Figures 3 and 4. Given these maps and the flow diagrams in Figure 2, the Boolean equations can be constructed for the Y_3 output. The resulting equation, in PALASM format, is shown in Figure 5.

It is important to note that the equation in Figure 5 is written for the inverse of the Y_3 output ($\overline{Y_3}$). This is necessary if true data is desired on the output pin because of the inverting nature of the output buffer on the PAL. The inverted form of the equation is easily implemented by selecting the negative version of the data ($\overline{Y_3}$ in the hold operation for example) or by grouping zeros in a combinatorial logic function (see Figures 3 and 4). Notice that the multiplexer strategy works equally well for active LOW or active HIGH logic functions.

Once the transformation of the Data-Flow Diagrams and K-Maps to Boolean equations is understood, the interested reader should be able to construct K-Maps for the other Y outputs and, in conjunction with the Data-Flow Diagrams of Figure 2, write the PALASM equations for the resulting logic functions. This exercise will help the reader to fully appreciate the advantages of the Data-Flow Diagram/Multiplexer method of PAL design. Consult the full PALASM listing (Figure 9) for the complete solutions.

Once the data portion of the Encoder/Decoder is completed, only the two status outputs, $\overline{H}$ and $\overline{INV}$, need to be implemented. $\overline{H}$ indicates the synchronization mark (11111) has been detected and is simply an AND of Y_3 through S_{OUT}. $\overline{INV}$ indicates an invalid serial input was received.

The $\overline{INV}$ signal is registered and held until the clear $\overline{INV}$ flag input (CIF) is brought LOW, deactivating the flag. Only during a 5B-4B conversion operation ($M_1 =$ HIGH, $M_0 =$ LOW) is the $\overline{INV}$ flag activated. Figures 6 and 7 show the $\overline{INV}$ flag mode definitions and the intermediate INVALID logic equation respectively.

In this case, an active LOW output is desired so the active HIGH form of the $\overline{INV}$ signals is developed internally. Ones are grouped in the intermediate combinatorial logic function (INVALID) and the true version of the data is selected. The complete PALASM equation for $\overline{INV}$ is given in Figure 8.

$$\overline{A_3} = Y_3 * \overline{Y_2} + Y_3 * S_{OUT}$$ 03862A-96
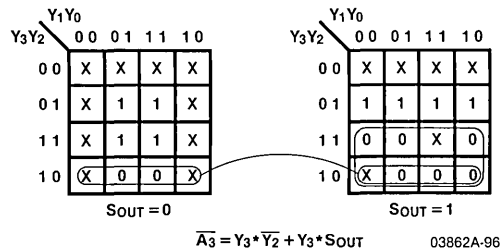
Figure 3. 5B to 4B Conversion K-Map for Y_3 Output

$$\overline{B_0} = D_3 * \overline{D_0} + D_1 * \overline{D_0}$$ 03862A-97
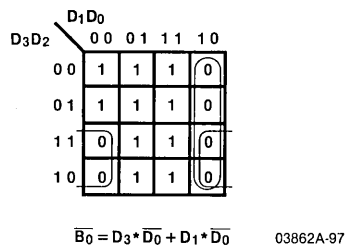
Figure 4. 4B to 5B Conversion K-Map for Y_3 Output

$\overline{Y_3} := \text{EN} * \overline{Y_3}$	+ ;HOLD
$\overline{\text{EN}} * \overline{M_1} * \overline{M_0} * \overline{Y_2}$	+ ;SERIAL SHIFT IN
$\overline{\text{EN}} * \overline{M_1} * M_0 * \overline{S_{OUT}}$	+ ;SERIAL SHIFT OUT
$\overline{\text{EN}} * M_1 * \overline{M_0} * Y_3 * S_{OUT}$	+ ;CONVERT SERIAL
$\overline{\text{EN}} * M_1 * \overline{M_0} * \overline{Y_3} * \overline{Y_2}$	+ ;INPUT AND LOAD
$\overline{\text{EN}} * M_1 * M_0 * D_3 * \overline{D_0}$	+ ;CONVERT PARALLEL
$\overline{\text{EN}} * M_1 * M_0 * D_1 * \overline{D_0}$	;INPUT AND LOAD

Figure 5. PALASM Equation for $\overline{Y_3}$ 03862A-98

CIF	M_1	M_0	OPERATION	DATA-FLOW DIAGRAM
0	X	X	CLEAR INV FLAG	
1	0 0 1	0 1 1	HOLD INV FLAG	
1	1	0	SET INV FLAG	

"0"

INV

INV

Y_3 Y_2 Y_1 Y_0 S_{OUT}

INVALID

INV
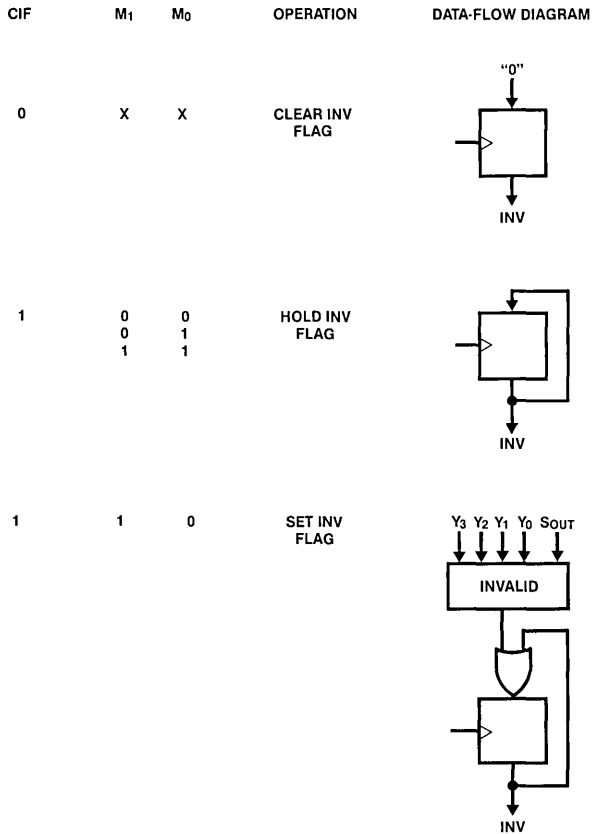
Figure 6. INV Flag Mode Definitions   03862A-99

Y_1Y_0
Y_3Y_2 00 01 11 10

	00	01	11	10
00	①	1	1	1
01	1	0	0	1
11	1	0	0	1
10	①	0	0	1

$S_{OUT} = 0$

Y_1Y_0
Y_3Y_2 00 01 11 10

	00	01	11	10
00	①	1	1	1
01	0	0	0	0
11	0	0	①	0
10	①	0	0	0

$S_{OUT} = 1$

$$INVALID = \overline{Y_3} * \overline{Y_2} \qquad +$$
$$\overline{Y_2} * \overline{Y_1} * \overline{Y_0} \qquad +$$
$$\overline{Y_0} * \overline{S_{OUT}} \qquad +$$
$$Y_3 * Y_2 * Y_1 * Y_0 * S_{OUT}$$
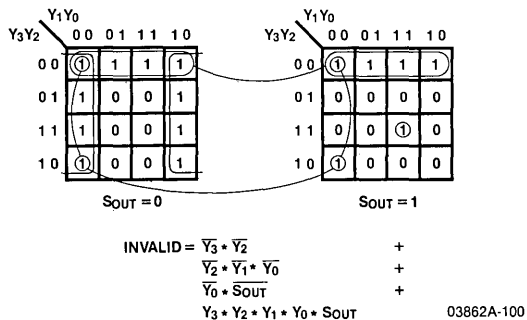
03862A-100

Figure 7. PALASM Equation for INVALID

$INV := \overline{CIF} * INV$;HOLD INV FLAG
$\overline{CIF} * M_1 * \overline{M_0} * \overline{Y_3} * \overline{Y_2}$ +   ;SET INV FLAG IF INVALID IS TRUE
$\overline{CIF} * M_1 * \overline{M_0} * \overline{Y_0} * \overline{S_{OUT}}$ +
$\overline{CIF} * M_1 * \overline{M_0} * \overline{Y_2} * \overline{Y_1} * \overline{Y_0}$ +
$\overline{CIF} * M_1 * \overline{M_0} * Y_3 * Y_2 * Y_1 * Y_0 * S_{OUT}$

03862A-101

Figure 8. PALASM Equation for INV

```
PAL16R6                                PAL DESIGN SPECIFICATION
PAT003                                 WARREN K. MILLER 2/15/82
4B-5B ENCODER/DECODER
ADVANCED MICRO DEVICES
 CK M1    M0 D3 D2 D1 D0 /EN   /CIF GND
/E  SIN /INV Y0 Y1 Y2 Y3  SOUT /H   VCC

/SOUT := EN*/SOUT                +       ;HOLD
        /EN*/M1*/M0*/SIN         +       ;SERIAL SHIFT IN
        /EN*/M1* M0*/Y0          +       ;SERIAL SHIFT OUT
        /EN* M1*/M0*/SIN         +       ;CONVERT SERIAL INPUT AND LOAD
        /EN* M1* M0* D3* D1      +       ;CONVERT PARALLEL INPUT AND LOAD
        /EN* M1* M0* D3* D0              ;

/Y0    := EN*/Y0                 +
        /EN*/M1*/M0*/SOUT        +
        /EN*/M1* M0*/Y1          +
        /EN* M1*/M0*/SOUT        +
        /EN* M1* M0* Y3* Y2*/Y0  +
        /EN* M1* M0*/D3* D1      +
        /EN* M1* M0*/D3* D2* D0

/Y1    := EN*/Y1                 +
        /EN*/M1*/M0*/Y0          +
        /EN*/M1* M0*/Y2          +
        /EN* M1*/M0*/Y0          +
        /EN* M1* M0* Y3* Y2      +
        /EN* M1* M0*/D2

/Y2    := EN*/Y2                 +
        /EN*/M1*/M0*/Y1          +
        /EN*/M1* M0*/Y3          +
        /EN* M1*/M0*/Y1          +
        /EN* M1* M0*/D3*/D1*/D0  +
        /EN* M1* M0*/D3* D2*/D1  +
        /EN* M1* M0* D3*/D1* D0

/Y3    := EN*/Y3                 +
        /EN*/M1*/M0*/Y2          +
        /EN*/M1* M0*/SOUT        +
        /EN* M1*/M0* Y3* SOUT    +
        /EN* M1* M0* Y3*/Y2      +
        /EN* M1* M0* D3*/D0      +
        /EN* M1* M0* D1*/D0

 INV := /CIF* INV                +       ;HOLD INV FLAG
        /CIF* M1*/M0*/Y3*/Y2     +       ;SET INV FLAG IF INVALID TRUE
        /CIF* M1*/M0*/Y2*/Y1*/Y0 +       ;
        /CIF* M1*/M0*/Y0*/SOUT   +
        /CIF* M1*/M0* Y3* Y2* Y1* Y0* SOUT


 H    =  Y3* Y2* Y1* Y0* SOUT
```

```
FUNCTION TABLE

CK /E /EN M1 M0 D3 D2 D1 D0 SIN /CIF   Y3 Y2 Y1 Y0 SOUT /INV /H
-----------------------------------------------------------------
;
;    HIGH Z TEST
X    H  X  X  X  X  X  X  X  X    X     Z  Z  Z  Z  Z    Z    X
;
;    INITIALIZE /INV
C    L  X  X  X  X  X  X  X  X    L     X  X  X  X  X    H    X
;
;    SERIAL SHIFT IN TEST
C    L  H  H  H  H  H  H  H  X    H     H  H  H  H  L    H    H
C    L  H  L  L  X  X  X  X  H    H     H  H  H  L  H    H    H
C    L  H  L  L  X  X  X  X  H    H     H  H  L  H  H    H    H
C    L  H  L  L  X  X  X  X  H    H     H  L  H  H  H    H    H
C    L  H  L  L  X  X  X  X  H    H     L  H  H  H  H    H    H
C    L  H  L  L  X  X  X  X  H    H     H  H  H  H  H    H    L
;
;    SYNCHRONIZATION MARK DETECT TEST
C    L  H  H  H  L  L  L  L  X    H     H  L  L  H  H    H    H
;
;    4B TO 5B CONVERSION FOLLOWED BY 5B TO 4B RECONVERSION
;    NOTE THAT DURING RECONVERSION THE BIT ORDER IS REVERSED
C    L  H  H  L  X  X  X  X  L    H     L  L  H  H  L    H    H
C    L  H  H  H  L  L  L  H  X    H     H  H  L  H  H    H    H
C    L  H  H  L  X  X  X  X  L    H     L  L  L  H  L    H    H
C    L  H  H  H  L  L  H  L  X    H     L  H  L  L  H    H    H
C    L  H  H  L  X  X  X  X  L    H     H  L  L  H  L    H    H
C    L  H  H  H  L  L  H  H  X    H     H  H  L  L  H    H    H
C    L  H  H  L  X  X  X  X  L    H     L  L  L  L  L    H    H
C    L  H  H  H  L  H  L  L  X    H     H  L  H  H  H    H    H
C    L  H  H  L  X  X  X  X  L    H     L  H  H  H  L    H    H
C    L  H  H  H  L  H  L  H  X    H     H  L  H  L  H    H    H
C    L  H  H  L  X  X  X  X  L    H     L  H  L  H  L    H    H
C    L  H  H  H  L  H  H  L  X    H     L  H  H  L  H    H    H
C    L  H  H  L  X  X  X  X  L    H     H  H  L  H  L    H    H
C    L  H  H  H  L  H  H  H  X    H     H  H  H  L  H    H    H
C    L  H  H  L  X  X  X  X  L    H     L  H  L  L  L    H    H
C    L  H  H  H  H  L  L  L  X    H     L  H  L  H  H    H    H
C    L  H  H  L  X  X  X  X  L    H     H  L  H  H  L    H    H
C    L  H  H  H  H  L  L  H  X    H     H  L  L  H  L    H    H
C    L  H  H  L  X  X  X  X  L    H     L  L  H  L  L    H    H
C    L  H  H  H  H  L  H  L  X    H     L  H  L  H  L    H    H
C    L  H  H  L  X  X  X  X  L    H     H  L  H  L  L    H    H
C    L  H  H  H  H  L  H  H  X    H     H  H  L  H  L    H    H
C    L  H  H  L  X  X  X  X  L    H     H  L  L  L  L    H    H
C    L  H  H  H  H  H  L  L  X    H     L  H  H  H  H    H    H
C    L  H  H  L  X  X  X  X  L    H     H  H  H  H  L    H    H
C    L  H  H  H  H  H  L  H  X    H     H  L  H  H  L    H    H
C    L  H  H  L  X  X  X  X  L    H     L  H  H  L  L    H    H
C    L  H  H  H  H  H  H  L  X    H     L  H  H  H  L    H    H
C    L  H  H  L  X  X  X  X  L    H     H  H  H  L  L    H    H
C    L  H  H  H  H  H  H  H  X    H     H  H  H  H  L    H    H
C    L  H  H  L  X  X  X  X  L    H     H  H  L  L  L    H    H
```

```
;
;   SERIAL SHIFT IN TEST
C   L H H H H H H X   H    H H H H L   H   H
C   L H L H X X X X X   H    L H H H H   H   H
C   L H L H X X X X X   H    H L H H H   H   H
C   L H L H X X X X X   H    H H L H H   H   H
C   L H L H X X X X X   H    H H H L H   H   H
C   L H L H X X X X X   H    H H H H L   H   H
C   L H L H X X X X X   H    L H H H H   H   H
-------------------------------------------------------------------
```

DESCRIPTION
THIS PART IMPLEMENTS A 4B 5B ENCODER/DECODER FOR TAPE DRIVES. ON
A WRITE IT ENCODES THE 4B INPUT DATA TO THE 5B FORMAT AND SERIALIZES
THE DATA. ON A READ THE 5B DATA IS SHIFTED IN, RECONVERTED TO THE
4B FORMAT, AND OUTPUT TO THE DATA BUS.

PAL16R6   PAL DESIGN SPECIFICATION
PAT003   WARREN K. MILLER 2/15/82
4B-5B ENCODER/DECODER
ADVANCED MICRO DEVICES
*D9724
FO
L0000 1111 1111 1111 1111 1111 1111 1111 1111 *
L0032 1111 1101 1101 1101 1101 1101 1111 1111 *
L0256 1111 1110 1111 1111 1111 1111 1011 1111 *
L0288 1011 1011 1111 1111 1111 1111 0111 1110 *
L0320 1011 0111 1111 1111 1111 1110 0111 1111 *
L0352 0111 1011 1111 1111 1111 1111 0111 1110 *
L0384 0111 0111 0111 1111 0111 1111 0111 1111 *
L0416 0111 0111 0111 1111 1111 0111 0111 1111 *
L0512 1111 1111 1110 1111 1111 1111 1011 1111 *
L0544 1011 1011 1111 1110 1111 1111 0111 1111 *
L0576 1011 0110 1111 1111 1111 1111 0111 1111 *
L0608 0111 1001 1101 1111 1111 1111 0111 1111 *
L0640 0111 1011 1101 1110 1111 1111 0111 1111 *
L0672 0111 0111 0111 1111 1111 1011 0111 1111 *
L0704 0111 0111 1111 1111 0111 1011 0111 1111 *
L0768 1111 1111 1111 1110 1111 1111 1011 1111 *
L0800 1011 1011 1111 1111 1110 1111 0111 1111 *
L0832 1011 0111 1110 1111 1111 1111 0111 1111 *
L0864 0111 1011 1111 1111 1110 1111 0111 1111 *
L0896 0111 0111 1011 1111 1011 1011 0111 1111 *
L0928 0111 0111 1011 0111 1011 1111 0111 1111 *
L0960 0111 0111 0111 1111 1011 0111 0111 1111 *
L1024 1111 1111 1111 1111 1110 1111 1011 1111 *
L1056 1011 1011 1111 1111 1111 1110 0111 1111 *
L1088 1011 0111 1111 1110 1111 1111 0111 1111 *
L1120 0111 1011 1111 1111 1111 1110 0111 1111 *
L1152 0111 1011 1101 1101 1111 1111 0111 1111 *
L1184 0111 0111 1111 1011 1111 1111 0111 1111 *
L1280 1111 1111 1111 1111 1111 1110 1011 1111 *
L1312 1011 1010 1111 1111 1111 1111 0111 1111 *
L1344 1011 0111 1111 1111 1110 1111 0111 1111 *
L1376 0111 1010 1111 1111 1111 1111 0111 1111 *
L1408 0111 1011 1101 1101 1111 1110 0111 1111 *
L1440 0111 0111 1011 1111 0111 1111 0111 1111 *
L1472 0111 0111 1011 0111 1111 0111 0111 1111 *
L1536 1111 1111 1111 1111 1111 1111 1110 0111 *
L1568 0111 1011 1110 1110 1111 1111 1111 0111 *
L1600 0111 1011 1111 1110 1110 1110 1111 0111 *
L1632 0111 1010 1111 1111 1111 1110 1111 0111 *
L1664 0111 1001 1101 1101 1101 1101 1111 0111 *
C8E23*
V0001 XXXXXXXXX01XZZZZZZX1 *
V0002 CXXXXXXX000XHXXXXXX1 *
V0003 C1111111100XHHHHHLH1 *
V0004 C00XXXX11001HLHHHHH1 *
V0005 C00XXXX11001HHLHHHH1 *
V0006 C00XXXX11001HHHLHHH1 *
V0007 C00XXXX11001HHHHLHH1 *
V0008 C00XXXX11001HHHHHHL1 *

```
V0009 C1100001100XHHLLHHH1 *
V0010 C10XXXX11000HHHLLLH1 *
V0011 C1100011100XHHLHHHH1 *
V0012 C10XXXX11000HHLLLLH1 *
V0013 C1100101100XHLLHLHH1 *
V0014 C10XXXX11000HHLLHLH1 *
V0015 C1100111100XHLLHHHH1 *
V0016 C10XXXX11000HLLLLLH1 *
V0017 C1101001100XHHHLHHH1 *
V0018 C10XXXX11000HHHHLLH1 *
V0019 C1101011100XHLHLHHH1 *
V0020 C10XXXX11000HHLHLLH1 *
V0021 C1101101100XHLHHLHH1 *
V0022 C10XXXX11000HHLHHLH1 *
V0023 C1101111100XHLHHHHH1 *
V0024 C10XXXX11000HLLHLLH1 *
V0025 C1110001100XHHLHLHH1 *
V0026 C10XXXX11000HHHLHLH1 *
V0027 C1110011100XHHLLHLH1 *
V0028 C10XXXX11000HLHLLLH1 *
V0029 C1110101100XHHLHLLH1 *
V0030 C10XXXX11000HLHLHLH1 *
V0031 C1110111100XHHHLHLH1 *
V0032 C10XXXX11000HLLLHLH1 *
V0033 C1111001100XHHHHLHH1 *
V0034 C10XXXX11000HHHHHLH1 *
V0035 C1111011100XHHHLHLH1 *
V0036 C10XXXX11000HLHHHLLH1 *
V0037 C1111101100XHHHHLLH1 *
V0038 C10XXXX11000HLHHHLH1 *
V0039 C1111111100XHHHHHLH1 *
V0040 C10XXXX11000HLLHHLH1 *
V0041 C1111111100XHHHHHLH1 *
V0042 C01XXXX1100XHHHHLHH1 *
V0043 C01XXXX1100XHHHLHHH1 *
V0044 C01XXXX1100XHHLHHHH1 *
V0045 C01XXXX1100XHLHHHHH1 *
V0046 C01XXXX1100XHHHHHLH1 *
V0047 C01XXXX1100XHHHHLHH1 *
CA76
```

LOGIC DIAGRAM FOR:
GCR ENCODES/DECODES USING AmPAL16R6

Columns: Inputs (0–31)
Rows: Product Terms (0–63)

= Fuse intact   ⊠ = All fuses intact   + = Fuse blown

03862A-102

.

Interfacing the 8086 (8088) to the Z-BUS

by Nick Zwick
Advanced Micro Devices

This application note describes how replacing two 8086 support chips with a Z8000 support chip and an AmPAL16R8A allows the 8086 CPU to interface directly to the Z-BUS. Since the timing of the signals used is the same for the 8088 CPU, this circuit will work equally well in those applications.

Interfacing the 8086 CPU to the Z-BUS allows 8086 users to take advantage of the very powerful Z8000 peripheral and memory support circuits that are available. The Z8000 peripheral circuits in particular offer the user higher throughput rates, simpler control software and less system overhead requirements than any previous generation peripheral family for any CPU.

DESIGN REQUIREMENTS

The 8086 CPU can operate in two different modes. In minimum mode, it generates all the bus control and timing signals for the 8086 (8085, 8088) buses directly on-chip. In maximum mode, the CPU puts out status information early in each bus cycle and relies on an external bus controller chip, the 8288, to generate timing and control signals. This implementation uses the CPU in maximum mode and replaces the 8288 with a programmable array logic element

(AmPAL16R8A) that generates the Z-BUS timing and control signals from the status signals provided by the CPU. It also makes use of the AmZ8127 clock generator to allow precise timing resolution by providing an oscillator signal at 3 times the CPU clock frequency. The AmZ8127 provides all the clock generation functions of an 8284A as well as several additional functions. Either clock chip will work in this system.

The bus controller provides the following functions:

—Generates $\overline{AS}$, $\overline{DS}$, $\overline{INTACK}$ and R/W with proper timing relative to address and data.

—Provides simultaneous assertion of $\overline{AS}$ and $\overline{DS}$ during reset.

—Automatic insertion of 1 wait state for all I/O cycles.

—Synthesizes a single Z-BUS interrupt acknowledge cycle from the 8086 IACK cycles.

Figure 1 shows the circuit interconnection diagram. The system uses a high speed AmPAL16R8A to generate $\overline{AS}$, $\overline{DS}$, R/W and $\overline{INTACK}$ of the Z-BUS, RDY for wait state generation, and three internal state variables. The registers are clocked with the 15MHz OSC signal from the 8127. The five input signals to the PAL are 5MHz CPU Clock Signal (CLK), System Reset (RESET), and the three CPU Status States ($\overline{S_0}$, $\overline{S_1}$, $\overline{S_2}$).

Figure 1. Circuit Interconnection Diagram

The CPU indicates the start of a bus cycle by bringing at least one of the status lines low from the idle high state (see Figure 2). This starts an internal timing sequence within the PAL which corresponds closely to the various T states of a bus cycle. $\overline{AS}$ is asserted during the time CLK is LOW during T_1. $\overline{DS}$ is asserted at the start of T_3. If it is an I/O cycle, then RDY would be disabled for one CLK period straddling T_2 and T_3 causing the 8127 to request 1 wait state after T_3. In either case, $\overline{DS}$ remains asserted until after the first 1/3 of T_4, which is identified by the status lines returning to the idle state during the previous cycle. R/W is generated by sampling $\overline{S_0}$ and $\overline{S_1}$ during $\overline{AS}$.

It is in the realm of interrupts where the Z8000 peripherals shine over other periperhals. Each peripheral can identify many different exception conditions during its operation. The occurrence of one or more of these conditions causes activation of a single interrupt request line. The peripheral wants the CPU to respond with a single interrupt acknowledge cycle, during which the peripheral resolves priority and provides the CPU with enough status and vector information to allow it to respond to the exception without any further interrogation of the peripheral. This allows interrupt driven systems to achieve very high data throughput rates.
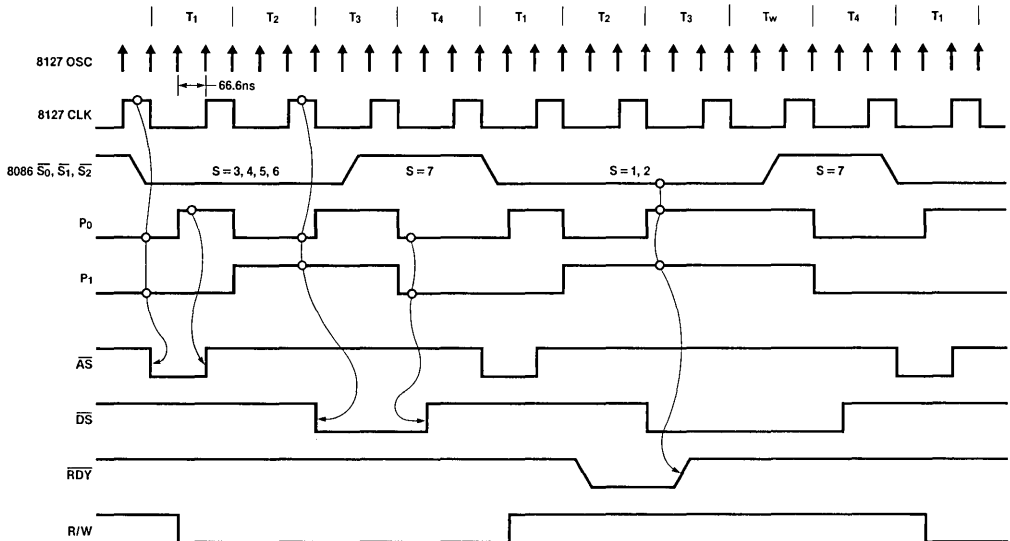
03862A-104

Figure 2. Memory and I/O Timing Diagram

The 8086 CPU responds to an interrupt request with a sequence of two interrupt acknowledge cycles, and only in the second is any data read off the bus. As stated before, the Z-BUS peripherals require only one acknowledge cycle. The timing of this has to be such that there is enough delay between $\overline{AS}$ going HIGH and $\overline{DS}$ going LOW to allow any prioritizing daisy chains to settle, and $\overline{DS}$ has to be wide enough to allow the peripheral time to place vector or status information on the bus. Figure 3 shows how these two requirements are accomplished by turning the two acknowledge cycles into one. The first cycle allows only $\overline{AS}$ and the second asserts only $\overline{DS}$ and does so for the complete cycle. This appears to the peripheral as one very long bus cycle which is identified as an interrupt acknowledge cycle by the assertion of $\overline{INTACK}$.

DESIGN APPROACH

To implement this design in a AmPAL16R8A requires recognizing the Z-BUS timing characteristics in Figures 2 and 3. The major characteristic to consider is counting the phases of a bus cycle. Internal state variables P_0 and P_1 are the result (see Figure 4a). An additional internal state variable (I_2) is necessary to count the second bus cycle of an interrupt acknowledge sequence. As shown in Figure 4b, I_2 in conjunction with $\overline{INTACK}$ allows $\overline{AS}$ to be asserted only in the first interrupt acknowledge cycle and $\overline{DS}$ only in the second. The RESET input is used to initialize the internal variables and assert $\overline{AS}$ and $\overline{DS}$. Note also that $\overline{S_0}$ and $\overline{S_1}$ are included in the $\overline{DS}$ equation to prevent $\overline{DS}$ from being asserted during a halt cycle.
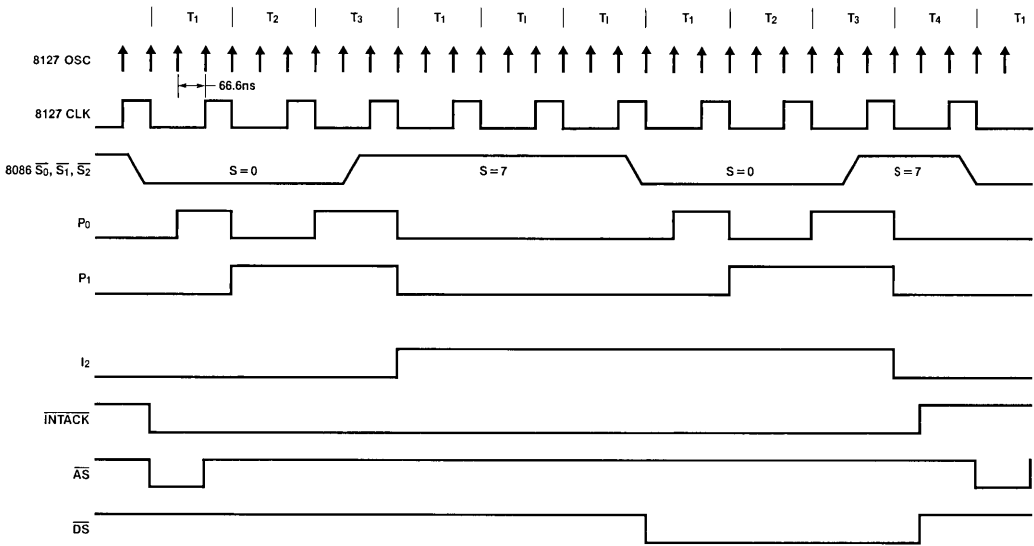
Figure 3. Interrupt Acknowledge Timing Diagram

03862A-105

The fact that AMD PALs are user programmable allows a great deal of flexibility for the designer. Minor timing changes are easily implemented by simply adding or changing a term in the logic equations and reprogramming the device. In this system, we have timing resolution to 67ns. This same configuration can be used with a 24MHz crystal for 8MHz CPU chips. The logic equations would change because the OSC period would be 42ns. The only hardware change would be the crystal.

An additional PAL could also perform chip select decoding based on both address and status signals.

CONCLUSION

We have seen how a properly programmed PAL can be used to replace a specialized bus controller chip and allow an 8086 CPU to interface directly to Z-BUS peripheral(s) and/or memory systems. This brings all the advantages of the superior Z8000 peripheral family in terms of both throughput and ease of use to 8086 users with no increase in chip count while still allowing a wide range of design flexibility. The logic diagram and PALASM equations for the AmPAL16R8A 8086 to Z-BUS interface chip are also shown.

P_1	P_0	PHASE	CPU T STATES
0	0	IDLE	T_4, T_1
0	1	$\overline{AS}$ TIME	T_1
1	0	$\overline{AS}$ TO $\overline{DS}$ DELAY	T_2
1	1	$\overline{DS}$ TIME	T_3, T_W

03862A-106

Figure 4a

I_2	$\overline{INTACK}$	$\overline{AS}$	$\overline{DS}$
NO	NO	YES	YES
NO	YES	YES	NO
YES	YES	NO	YES

03862A-107

Figure 4b

```
PAL16R8                              PAL DESIGN SPECIFICATION
PAT004                               NICK ZWICK        6/21/82
8086 TO Z-BUS INTERFACE CHIP
ADVANCED MICRO DEVICES
CK RESET CLK /S0 /S1 /S2 NC      NC   NC  GND
/E /P0   /P1 /RW /DS /AS /INTACK RDY  /I2 VCC

;INTERNAL STATE VARIABLES

P0      := /RESET* S0*/P0*/P1*/CLK +
           /RESET* S1*/P0*/P1*/CLK +
           /RESET* S2*/P0*/P1*/CLK +
           /RESET*P0*/CLK          +
           /RESET*P1*CLK* S0       +
           /RESET*P1*CLK* S1       +
           /RESET*P1*CLK* S2

P1      := /RESET*P0*/P1*CLK       +
           /RESET*P1*/CLK          +
           /RESET*P1*CLK* S0       +
           /RESET*P1*CLK* S1       +
           /RESET*P1*CLK* S2

I2      := /RESET*INTACK*/I2*CLK*P0*P1 +
           /RESET*I2*/P1           +
           /RESET*I2*/P0           +
           /RESET*I2*P0*P1*/CLK

;Z-BUS OUTPUT SIGNALS

AS      := RESET                   +
           /P0*/P1*CLK*/I2         +
           AS*/P0*/I2*/DS

DS      := RESET                   +
           /INTACK*/P0*P1*CLK*S0   +
           /INTACK*/P0*P1*CLK*S1   +
           I2* S0* S1* S2          +
           DS*P0*P1

RW      := AS*S0*/S1               +
           RW*/AS

INTACK  := /RESET* S0* S1* S2      +
           /RESET*INTACK*/I2*P1    +
           /RESET*I2

/RDY    := /RESET*S0*/S1* S2*/P0*P1 +  ;DISABLE READY ON I/O OP
           /RESET*/S0*S1* S2*/P0*P1
```

5-35

```
FUNCTION TABLE
CK RESET CLK /S2 /S1 /SO  PO P1 I2  /AS /DS /RW /INTACK RDY
-------------------------------------------------------------------
;
;INITIALIZE
C H     X   X   X   X    L L L   L   L   X   H      H
;
;MEMORY WRITE OPERATION
C L     L   H   H   H    L L L   H   H   H   H      H
C L     H   H   H   L    L L L   L   H   H   H      H
C L     L   H   H   L    H L L   L   H   L   H      H
C L     L   H   H   L    H L L   H   H   L   H      H
C L     H   H   H   L    L H L   H   H   L   H      H
C L     L   H   H   L    L H L   H   H   L   H      H
C L     L   H   H   L    L H L   H   H   L   H      H
C L     H   H   H   L    H H L   H   L   L   H      H
C L     L   H   H   L    H H L   H   L   L   H      H
C L     L   H   H   H    H H L   H   L   L   H      H
C L     H   H   H   H    L L L   H   L   L   H      H
C L     L   H   H   H    L L L   H   H   L   H      H
C L     L   H   H   H    L L L   H   H   L   H      H
;
;I/O READ OPERATION
C L     H   L   L   H    L L L   L   H   L   H      H
C L     L   L   L   H    H L L   L   H   H   H      H
C L     L   L   L   H    H L L   H   H   H   H      H
C L     H   L   L   H    L H L   H   H   H   H      H
C L     L   L   L   H    L H L   H   H   H   H      L
C L     L   L   L   H    L H L   H   H   H   H      L
C L     H   L   L   H    H H L   H   L   H   H      L
C L     L   L   L   H    H H L   H   L   H   H      H
C L     L   L   L   H    H H L   H   L   H   H      H
C L     H   L   L   H    H H L   H   L   H   H      H
C L     L   L   L   H    H H L   H   L   H   H      H
C L     L   H   H   H    H H L   H   L   H   H      H
C L     H   H   H   H    L L L   H   L   H   H      H
C L     L   H   H   H    L L L   H   H   H   H      H
;
;TWO CYCLE INTERRUPT SEQUENCE
C L     H   L   L   L    L L L   L   H   H   L      H
C L     L   L   L   L    H L L   L   H   H   L      H
C L     L   L   L   L    H L L   H   H   H   L      H
C L     H   L   L   L    L H L   H   H   H   L      H
C L     L   L   L   L    L H L   H   H   H   L      H
C L     L   L   L   L    L H L   H   H   H   L      H
C L     H   L   L   L    H H L   H   H   H   L      H
C L     L   L   L   L    H H L   H   H   H   L      H
C L     L   H   H   H    H H L   H   H   H   L      H
C L     H   H   H   H    L L H   H   H   H   L      H
C L     L   H   H   H    L L H   H   H   H   L      H
C L     L   H   H   H    L L H   H   H   H   L      H
C L     H   L   L   L    L L H   H   L   H   L      H
C L     L   L   L   L    H L H   H   L   H   L      H
C L     L   L   L   L    H L H   H   L   H   L      H
C L     H   L   L   L    L H H   H   L   H   L      H
```

```
C L    L L L L    L H H    H    L    H    L        H
C L    L L L L    L H H    H    L    H    L        H
C L    H L L L    H H H    H    L    H    L        H
C L    L L L L    H H H    H    L    H    L        H
C L    L H H H    H H H    H    L    H    L        H
C L    H H H H    L L L    H    L    H    L        H
C L    L H H H    L L L    H    H    H    H        H
```

DESCRIPTION
THIS DEVICE IS USED FOR INTERFACING THE 8086 CPU DIRECTLY TO THE Z-BUS
ALLOWING INTEGRATION OF THE VERY POWERFUL Z8000 PERIPHERAL AND MEMORY
SUPPORT CIRCUITS INTO 8086 SYSTEMS. THE DEVICE IS DESIGNED TO WORK IN
CONJUNCTION WITH THE Z8127 CLOCK GENERATOR FOR PRECISE TIMING
RESOLUTION.

```
PAL16R8                              PAL DESIGN SPECIFICATION
PAT004                               NICK ZWICK      6/21/82
8086 TO Z-BUS INTERFACE CHIP
ADVANCED MICRO DEVICES
*D9724
*F0*
L0000 1001 0111 1110 1111 1111 1111 1110 1110 *
L0032 1010 1111 1111 1111 1111 1111 1101 1111 *
L0064 1010 1111 1111 1111 1111 1111 1111 1101 *
L0096 1010 1011 1111 1111 1111 1111 1110 1110 *
L0256 1011 1111 1011 0111 1011 1111 1110 1101 *
L0288 1011 1111 0111 1011 1011 1111 1110 1101 *
L0512 1011 1111 1011 1011 1011 1111 1111 1111 *
L0544 1001 1111 1110 1111 1111 1111 1110 1111 *
L0576 1010 1111 1111 1111 1111 1111 1111 1111 *
L0768 0111 1111 1111 1111 1111 1111 1111 1111 *
L0800 1101 0111 1111 1111 1111 1111 1101 1101 *
L0832 1101 1111 1111 1110 1101 1111 1111 1101 *
L1024 0111 1111 1111 1111 1111 1111 1111 1111 *
L1056 1111 0111 1001 1111 1111 1111 1110 1101 *
L1088 1111 0111 1101 1011 1111 1111 1110 1101 *
L1120 1110 1111 1011 1011 1011 1111 1111 1111 *
L1152 1111 1111 1111 1111 1110 1111 1110 1110 *
L1280 1111 1111 1011 0110 1111 1111 1111 1111 *
L1312 1111 1111 1111 1101 1111 1110 1111 1111 *
L1536 1011 0111 1111 1111 1111 1111 1101 1110 *
L1568 1011 1011 1111 1111 1111 1111 1110 1111 *
L1600 1011 0111 1011 1111 1111 1111 1110 1111 *
L1632 1011 0111 1111 1011 1111 1111 1110 1111 *
L1664 1011 0111 1111 1111 1011 1111 1110 1111 *
L1792 1011 1011 1011 1111 1111 1111 1101 1101 *
L1824 1011 1011 1111 1011 1111 1111 1101 1101 *
L1856 1011 1011 1111 1111 1011 1111 1101 1101 *
L1888 1011 1011 1111 1111 1111 1111 1111 1110 *
L1920 1011 0111 1011 1111 1111 1111 1110 1111 *
L1952 1011 0111 1111 1011 1111 1111 1110 1111 *
L1984 1011 0111 1111 1111 1011 1111 1110 1111 *
C711D*
V0001 C1XXXXXXXOXHHXLLHHH1 *
V0002 C00111XXXOXHHHHHHHH1 *
V0003 C01011XXXOXHHHHLHHH1 *
V0004 C00011XXXOXLHLHLHHH1 *
V0005 C00011XXXOXLHLHHHHH1 *
V0006 C01011XXXOXHLLHHHHH1 *
V0007 C00011XXXOXHLLHHHHH1 *
V0008 C00011XXXOXHLLHHHHH1 *
V0009 C01011XXXOXLLLLHHHH1 *
V0010 C00011XXXOXLLLLHHHH1 *
V0011 C00111XXXOXLLLLHHHH1 *
V0012 C01111XXXOXHHLLHHHH1 *
V0013 C00111XXXOXHHLHHHHH1 *
V0014 C00111XXXOXHHLHHHHH1 *
V0015 C01100XXXOXHHLHLHHH1 *
V0016 C00100XXXOXLHHHLHHH1 *
V0017 C00100XXXOXLHHHHHHH1 *
```

```
V0018 C01100XXX0XHLHHHHHH1 *
V0019 C00100XXX0XHLHHHHLH1 *
V0020 C00100XXX0XHLHHHHLH1 *
V0021 C01100XXX0XLLHLHHLH1 *
V0022 C00100XXX0XLLHLHHHH1 *
V0023 C00100XXX0XLLHLHHHH1 *
V0024 C01100XXX0XLLHLHHHH1 *
V0025 C00100XXX0XLLHLHHHH1 *
V0026 C00111XXX0XLLHLHHHH1 *
V0027 C01111XXX0XHHHLHHHH1 *
V0028 C00111XXX0XHHHHHHHH1 *
V0029 C01000XXX0XHHHHLLHH1 *
V0030 C00000XXX0XLHHHLLHH1 *
V0031 C00000XXX0XLHHHHLHH1 *
V0032 C01000XXX0XHLHHHLHH1 *
V0033 C00000XXX0XHLHHHLHH1 *
V0034 C00000XXX0XHLHHHLHH1 *
V0035 C01000XXX0XLLHHHLHH1 *
V0036 C00000XXX0XLLHHHLHH1 *
V0037 C00111XXX0XLLHHHLHH1 *
V0038 C01111XXX0XHHHHHLHL1 *
V0039 C00111XXX0XHHHHHLHL1 *
V0040 C00111XXX0XHHHHHLHL1 *
V0041 C01000XXX0XHHHLHLHL1 *
V0042 C00000XXX0XLHHLHLHL1 *
V0043 C00000XXX0XLHHLHLHL1 *
V0044 C01000XXX0XHLHLHLHL1 *
V0045 C00000XXX0XHLHLHLHL1 *
V0046 C00000XXX0XHLHLHLHL1 *
V0047 C01000XXX0XLLHLHLHL1 *
V0048 C00000XXX0XLLHLHLHL1 *
V0049 C00111XXX0XLLHLHLHL1 *
V0050 C01111XXX0XHHHLHLHH1 *
V0051 C00111XXX0XHHHHHHHH1 *
A334
```

LOGIC DIAGRAM FOR:
8086 TO ZBUS INTERFACE USING AmPAL16R8
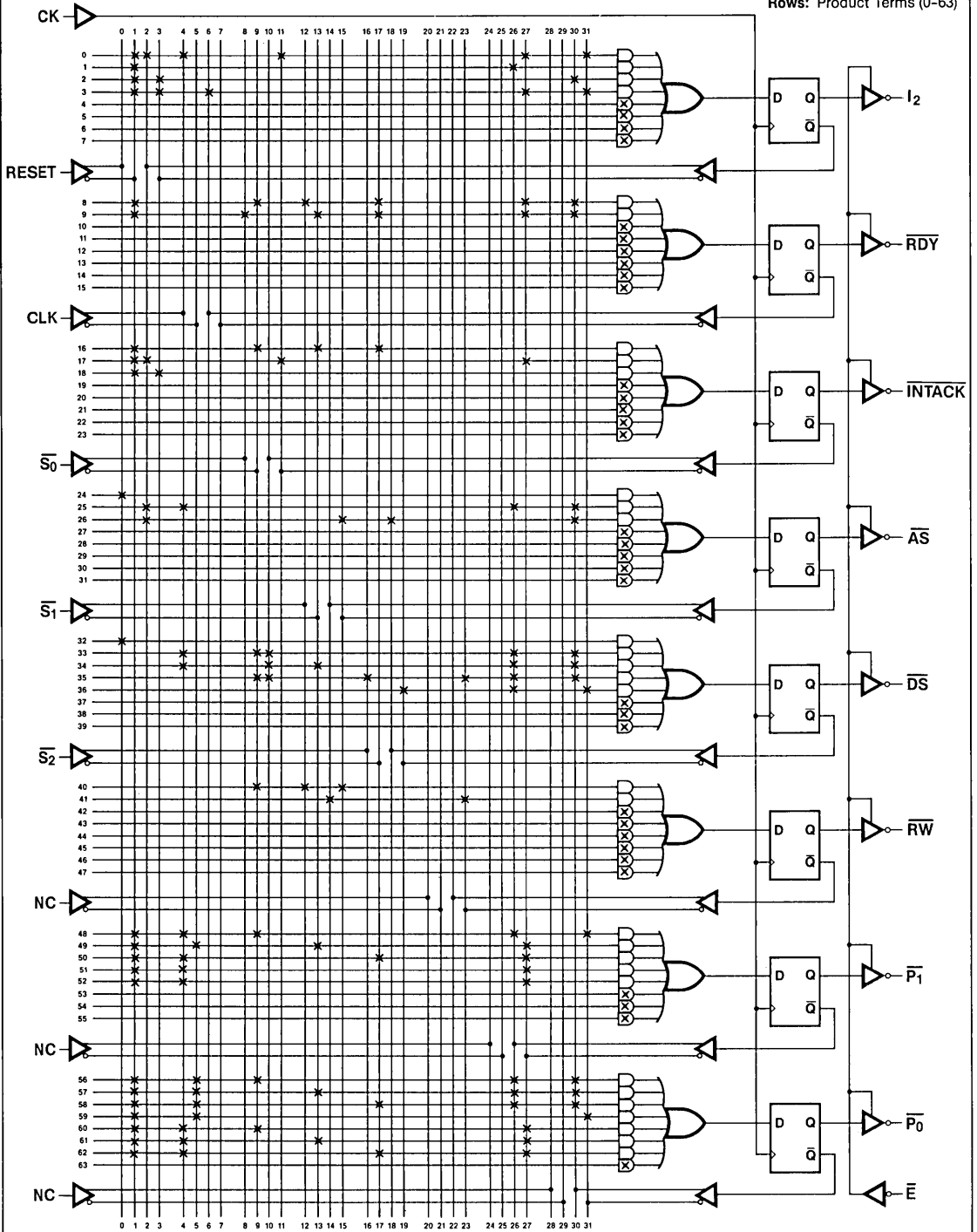
—*— = Fuse intact   —⊠— = All fuses intact   —+— = Fuse blown

03862A-108

An AMD PAL MULTIBUS Arbiter

by Mark S. Young
Advanced Micro Devices

The popularity of bus oriented systems can be traced to their low cost, flexibility, and expandability. Expansion is easy because a well defined standard enforces compatibility and simplifies interfacing. The MULTIBUS is a good example of a popular and easily used bus standard. However, all of these bus systems require some way of interacting with the other devices on the bus. This interaction is generally controlled by an arbitration unit. This application note will describe how a PAL can be used to construct a custom MULTIBUS arbiter which is higher performance and more economical than the LSI alternatives.

DESIGN REQUIREMENTS

Since the MULTIBUS can have more than one master trying to use the bus at the same time, an arbitration scheme is required to ensure correct operation of the system. Prioritization is accomplished by assigning different access priorities to the different bus masters. The two implementations available to assign priority are serial and parallel. The serial method involves a daisy chain of bus grant ins ($\overline{BPRN}$) and bus grant outs ($\overline{BPRO}$) with higher priority devices occupying the positions closer to the beginning of the chain (Figure 1a). The parallel method prioritizes the bus requests ($\overline{BREQ}$) of all the masters and generates a bus grant in ($\overline{BPRN}$) to the master of highest priority (Figure 1b). The priority decoder of Figure 1b is easily implemented in a single AmPAL16L8 (Figure 1c).

The timing requirements for the MULTIBUS are very easy to implement and are designed to handle the usual timing problems that appear in many systems. Control signals are all active LOW so that unconnected signals don't interfere with the normal operation of the bus. All bus arbitration is synchronous and all data transfers are asynchronous (see Figures 2a and 2b). The only requirement is that an address be valid 50ns before any control signals (read or write) become active. This prevents subtle timing problems caused by slow buffer/driver turn-on times.

The arbitration and grant timing (illustrated in Figure 2c) is also straightforward. Transfer requests are sent to the arbiters which then decide who gets the bus. If a master device is currently using the bus and is ordered off, it is allowed to complete its current bus transfer cycle (i.e., the current word or byte only). If the master, which was overruled, still needs the bus, it must wait until its priority is high enough to regain the bus.

A typical sequence is initiated when an external request is received ($\overline{SREQ}$) from a master device. This signal is synchronized to insure a valid bus request, minimizing the possibility of a metastable state occurring. After synchronization, the bus priority out ($\overline{BPRO}$) line is disabled to signal lower priority masters in the chain that a higher priority master wants the bus (serial method). In the parallel method, bus request and common bus request ($\overline{BREQ}$ and $\overline{CBREQ}$) are asserted to let the other MULTIBUS arbiters know a master wants the bus. Now, if the bus is not busy ($\overline{BUSY}$ is inactive), then the arbiter grants the requesting master access to the bus and asserts $\overline{BUSY}$. The address buffers are enabled at this time, one cycle ahead of the read and write signals. When the master receives a bus acknowledge ($\overline{XACK}$) from the slave device, a single transfer cycle has occurred. The bus master then releases the bus and if it needs to do more transfers, another arbitration/grant cycle must take place.
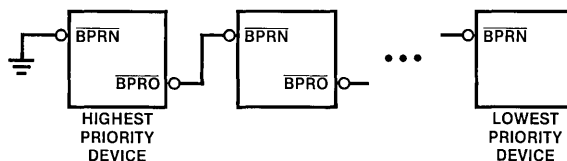
03862A-109

Figure 1a. Serial Priority Resolution
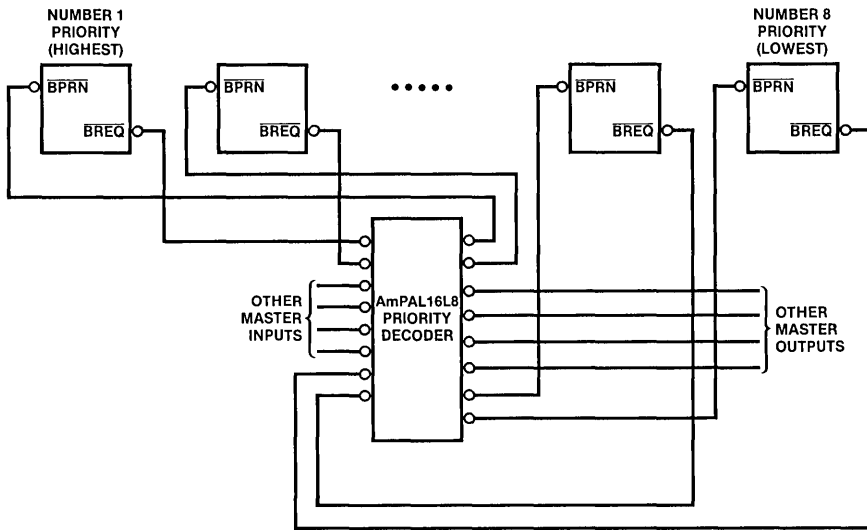
Figure 1b. Parallel Priority Technique

03862A-110
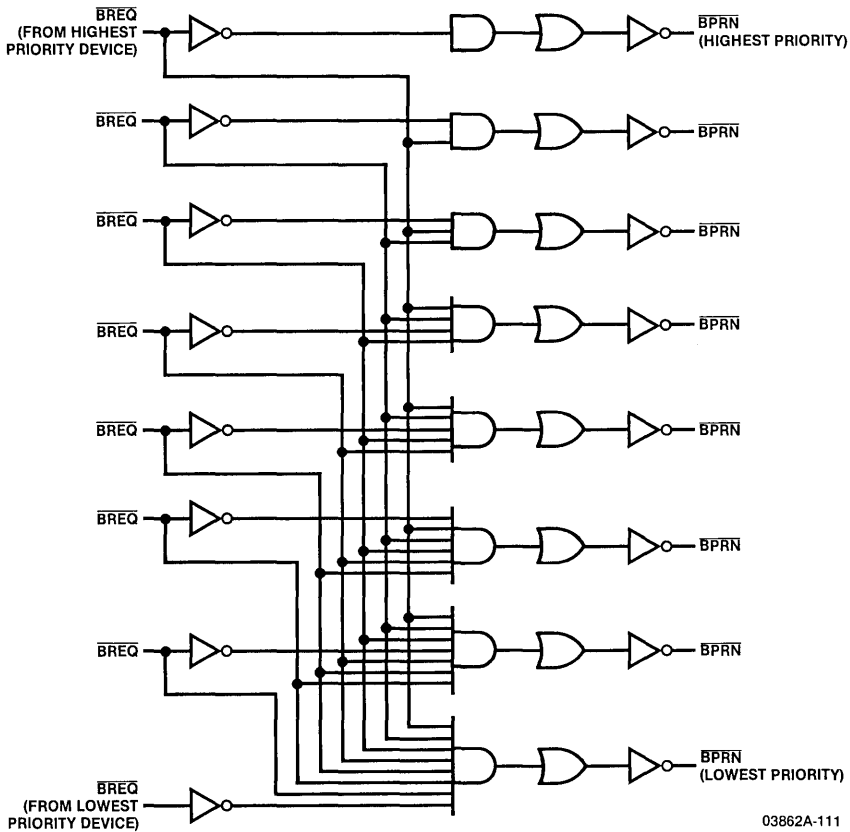
Figure 1c. PAL Implementation of Parallel Priority Resolution for MULTIBUS

03862A-111

IORC OR MRDC

STABLE ADDRESS

ADRESS LINES

MASTER TO SLAVE

XACK — PASSIVE

DATA LINES — STABLE DATA

SLAVE TO MASTER
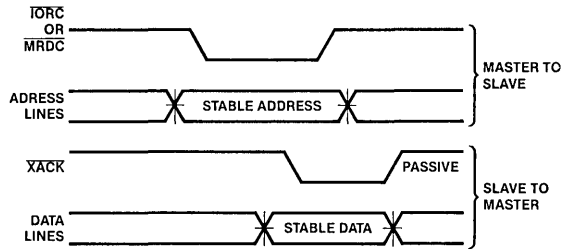
Figure 2a. Read AC Timing   03862A-112

IOWC OR MWTC

ADDRESS LINES — STABLE ADDRESS

DATA LINES — STABLE WRITE DATA

MASTER TO SLAVE

XACK — PASSIVE
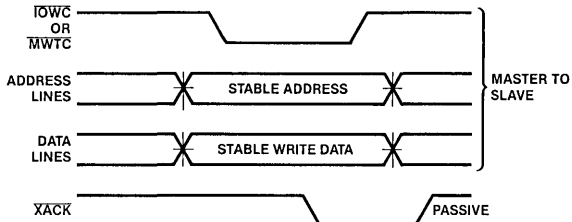
Figure 2b. Write AC Timing   03862A-113

PRIORITY RESOLUTION SHOWN HERE

MASTER A
- BCLK
- SREQ   (LOW)
- SREQ   (LOW)
- BPRN

MASTER B
- SREQ
- BREQ
- BPRN

MASTER A ON BUS   BUS EXCHANGE   MASTER B ON BUS

EXCHANGE OF BUS SHOWN HERE

MASTER A
- BUSY   HIGH IMPEDANCE STATE
- ADDRESS — ACTIVE STATE
- CONTROL — ACTIVE   HIGH IMPEDANCE
- AEN

MASTER B
- ADDRESS — HIGH IMPEDANCE   ACTIVE
- CONTROL — HIGH IMPEDANCE   ACTIVE
- AEN
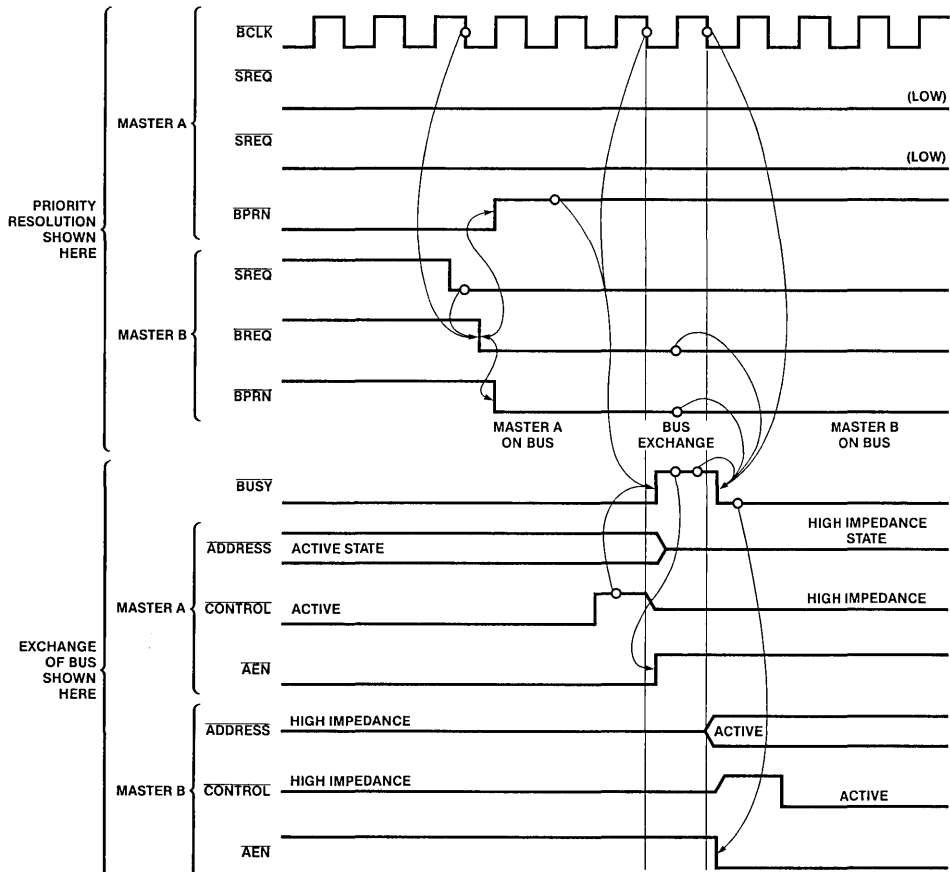
Figure 2c. Bus Control Exchange Operation   03862A-114

DESIGN APPROACH

The first step in actually designing the arbiter was to convert the arbitration/grant and control signal sequence into a simple state transition diagram (see Figure 3). The state diagram was then partitioned into its three basic components:

—request/synchronization
—grant/access logic
—control signals

The bus request logic decides when to issue a bus request using $\overline{SREQ}$ along with a qualifying read or write request ($\overline{RD}$ or $\overline{WR}$). This signal is then fed through a double synchronizer (states 1 and 2). This creates an internally stable bus request signal for the arbiter state machine (see Figure 4a). Next, this

request is fed into a bus grant flip-flop through some intervening logic. The intervening logic uses current bus status lines to determine whether to acquire the bus, give it up after the current transfer cycle is complete, or hold the bus (see grant/access logic in Figure 4b). The final major function is the bus control logic. After successful acquisition of the bus, the appropriate control signals ($\overline{MRDC}$, $\overline{MWTC}$, $\overline{IORC}$, $\overline{IOWC}$) and address buffer enables must be asserted. In this case, the address buffer enable/grant ($\overline{AEN}$) line is run through a flip-flop to become a delayed read/write control signal enable ($\overline{OEN}$). This gives the drivers enough turn-on and set-up time (100ns minimum) for the address to stabilize on the MULTIBUS. The bus transfer control signal logic is illustrated in Figure 4c.

03862A-115
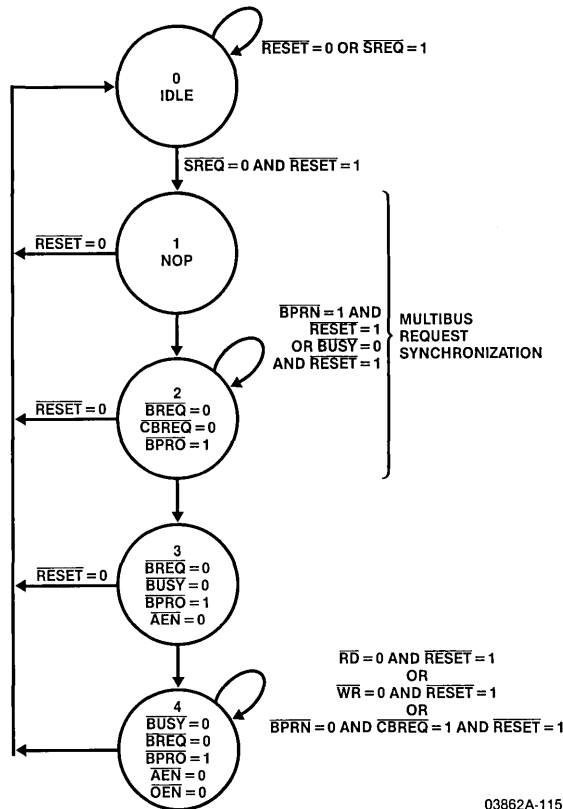
Figure 3. PAL MULTIBUS Arbiter State Transition Diagram
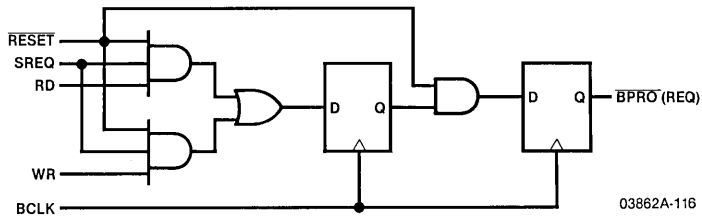
Figure 4a. Request Synchronizer
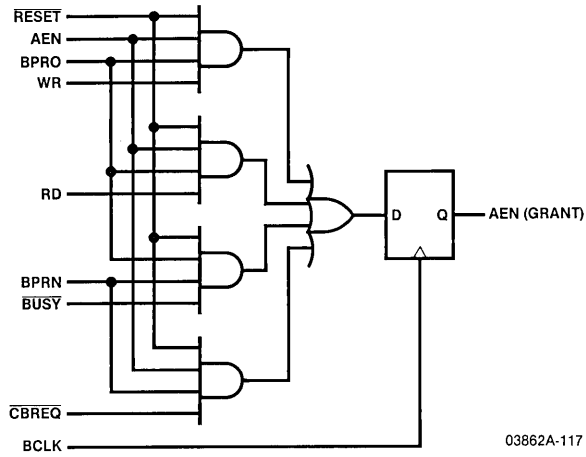
Figure 4b. Grant/Access Logic
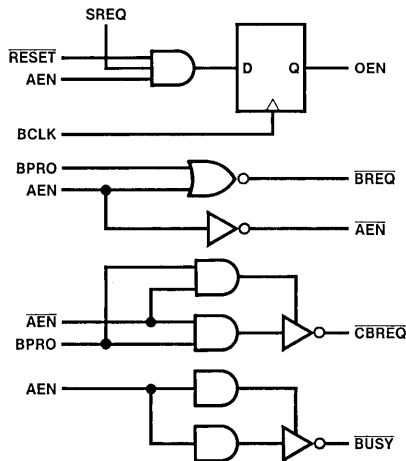
Figure 4c. Bus Transfer Control

The RESET line is run to all the registers to provide a synchronous reset for arbiter initialization. The full MULTIBUS standard requires a control signal drive capability of 32mA for all of its control lines. The PAL 24mA drive specification can drive up to 7 cards on the bus, which is more than adequate for almost all applications.

A couple of interesting features are available which show the flexibility and functionality of PALs in a custom logic design such as this. The available MULTIBUS arbiters, like most LSI devices, have been optimized for one processor family. Any microprocessor (from the Z80 to the 8086, Z8000 or 68000) can be interfaced to the MULTIBUS by tailoring the request logic of the PAL arbiter. This can result in a significant parts count reduction. The MULTIBUS uses open collector drivers for several signal lines (BUSY and CBREQ). While PALs don't have open collector drivers, that condition can easily be simulated by enabling the output drivers only when the output is active. In addition, most MULTIBUS arbiters force each master to re-arbitrate for access to the bus at the end of each data transfer. However, in the PAL arbiter design, if a master is executing multiple data transfers and if no master of higher priority is requesting the bus then the current master can retain the bus and execute more transfers. This reduces bus arbitration overhead and increases bus bandwidth. Finally, at the completion of all transfers, the current bus master normally releases the bus. In this design, if no one wants the bus, the last master to use it holds onto the bus on the assumption that it is going to be the next user. If it is, arbitration time is saved. If not, no time is lost.

Figure 5 shows a block diagram of the arbiter and bus control logic (which fits into 1/2 of a AmPAL16L8). A complete logic diagram and PALASM equations of the AmPAL16R4 are shown on the following pages.
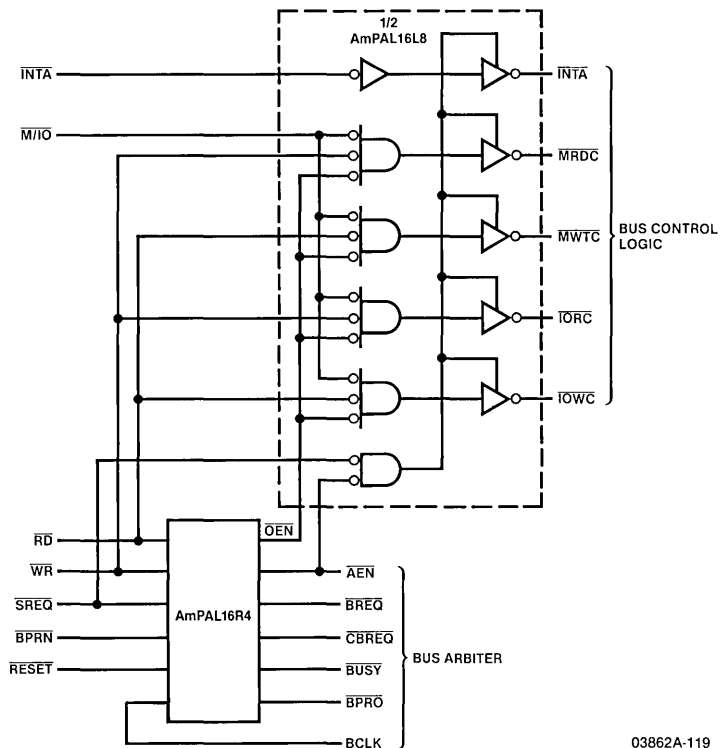
Figure 5. PAL MULTIBUS Arbiter with Bus Control Logic

```
PAL16R4                              PAL DESIGN SPECIFICATION
PAT005                               MARK S. YOUNG    6/22/82
MULTIBUS ARBITER
ADVANCED MICRO DEVICES
BCLK /RD    /WR    /SREQ /RESET /BPRN NC   NC    NC GND
/E   /CBREQ /BUSY /SYNC /BPRO  /AEN  /OEN /BREQ NC VCC

SYNC  := /RESET*SREQ*RD          +
         /RESET*SREQ*WR

BPRO  := /RESET*SYNC

AEN   := /RESET* AEN*BPRO*WR     +
         /RESET* AEN*BPRO*RD     +
         /RESET*BPRO*BPRN*/BUSY  +
         /RESET* AEN*BPRN*/CBREQ

OEN   := /RESET*SREQ*AEN

IF(BPRO*/AEN) CBREQ  = BPRO*/AEN

IF(AEN)       BUSY   = AEN

       BREQ   = BPRO              +
                AEN
```

FUNCTION TABLE

BCLK	/E	/RESET	/RD	/WR	/SREQ	/BPRN	SYNC	BPRO	AEN	OEN	CBREQ	BUSY	BREQ
;													
;INITIALIZE													
C	L	L	X	X	X	X	L	L	L	L	L	L	L
;													
;WRITE OPERATION													
C	L	H	H	H	L	L	L	L	L	L	L	L	L
C	L	H	H	L	L	L	H	L	L	L	L	L	L
C	L	H	H	L	L	L	H	H	L	L	H	L	H
C	L	H	H	L	L	L	H	H	H	L	L	X	H
C	L	H	H	L	L	L	H	H	H	H	L	H	H
C	L	H	H	L	L	L	H	H	H	H	L	H	H
C	L	H	H	L	L	L	H	H	H	H	L	H	H
C	L	H	H	H	L	L	L	H	H	H	L	H	H
C	L	H	H	H	H	L	L	L	H	L	L	H	H
;													
;REMOVE THE ARBITER FROM THE BUS													
C	L	H	H	H	H	H	L	L	L	L	L	L	L
;													
;READ OPERATION (WITH BUS CONTENTION)													
C	L	H	H	H	L	L	L	L	L	L	L	H	L
C	L	H	L	H	L	L	H	L	L	L	L	H	L
C	L	H	L	H	L	L	H	H	L	L	H	H	H
C	L	H	L	H	L	L	H	H	L	L	H	H	H
C	L	H	L	H	L	L	H	H	L	L	H	H	H
C	L	H	L	H	L	L	H	H	H	L	L	X	H
C	L	H	L	H	L	L	H	H	H	H	L	H	H
C	L	H	L	H	L	L	H	H	H	H	L	H	H
C	L	H	L	H	L	L	H	H	H	H	L	H	H
C	L	H	H	H	L	L	L	H	H	H	L	H	H
C	L	H	H	H	H	L	L	L	H	L	L	H	H

DESCRIPTION

THIS DEVICE IS A MULTIBUS ARBITER. IT SUPPORTS BOTH SERIAL AND PARALLEL
BUS ARBITRATION SCHEMES. INTERNAL SYNCHRONIZATION LOGIC MINIMIZES THE
POSSIBILITY OF METASTABLE CONDITIONS. THE GRANT/ACCESS LOGIC HAS BEEN
DESIGNED TO MINIMIZE BUS ARBITRATION OVERHEAD. THE TRANSFER CONTROL
LOGIC HAS BEEN DESIGNED TO ALLOW INTERFACING A WIDE VARIETY OF
PROCESSORS TO THE MULTIBUS.

```
PAL16R4                                    PAL DESIGN SPECIFICATION
PAT005                                     MARK S. YOUNG    6/22/82
MULTIBUS ARBITER
ADVANCED MICRO DEVICES
*D9724
*F0*
L0256 1111 1111 1111 1111 1111 1111 1111 1111 *
L0288 1111 1111 1111 1111 1110 1111 1111 1111 *
L0320 1111 1111 1111 1110 1111 1111 1111 1111 *
L0512 1111 1111 1011 0110 1111 1111 1111 1111 *
L0768 1111 1011 1111 0110 1110 1111 1111 1111 *
L0800 1011 1111 1111 0110 1110 1111 1111 1111 *
L0832 1111 1111 1111 0111 1010 1111 1101 1111 *
L0864 1111 1111 1111 0110 1011 1111 1111 1101 *
L1024 1111 1111 1111 0111 1111 1110 1111 1111 *
L1280 1011 1111 1011 0111 1111 1111 1111 1111 *
L1312 1111 1011 1011 0111 1111 1111 1111 1111 *
L1536 1111 1111 1111 1110 1111 1111 1111 1111 *
L1568 1111 1111 1111 1110 1111 1111 1111 1111 *
L1792 1111 1111 1111 1101 1110 1111 1111 1111 *
L1824 1111 1111 1111 1101 1110 1111 1111 1111 *
C3602*
V0001 CXXX0XXXX0011HHHHHX1 *
V0002 C11010XXX0011HHHHHX1 *
V0003 C10010XXX0011LHHHHX1 *
V0004 C10010XXX00L1LLHHLX1 *
V0005 C10010XXX001XLLLHLX1 *
V0006 C10010XXX001LLLLLLX1 *
V0007 C10010XXX001LLLLLLX1 *
V0008 C10010XXX001LLLLLLX1 *
V0009 C11010XXX001LHLLLLX1 *
V0010 C11110XXX001LHHLHLX1 *
V0011 C11111XXX0011HHHHHX1 *
V0012 C11010XXX0010HHHHHX1 *
V0013 C01010XXX0010LHHHHX1 *
V0014 C01010XXX00L0LLHHLX1 *
V0015 C01010XXX00L0LLHHLX1 *
V0016 C01010XXX00L0LLHHLX1 *
V0017 C01010XXX001XLLLHLX1 *
V0018 C01010XXX001LLLLLLX1 *
V0019 C01010XXX001LLLLLLX1 *
V0020 C01010XXX001LLLLLLX1 *
V0021 C01010XXX001LLLLLLX1 *
V0022 C11010XXX001LHLLLLX1 *
V0023 C11110XXX001LHHLHLX1 *
5484
```

LOGIC DIAGRAM FOR:
MULTIBUS ARBITER USING AmPAL16R4

Columns: Inputs (0–31)
Rows: Product Terms (0–63)
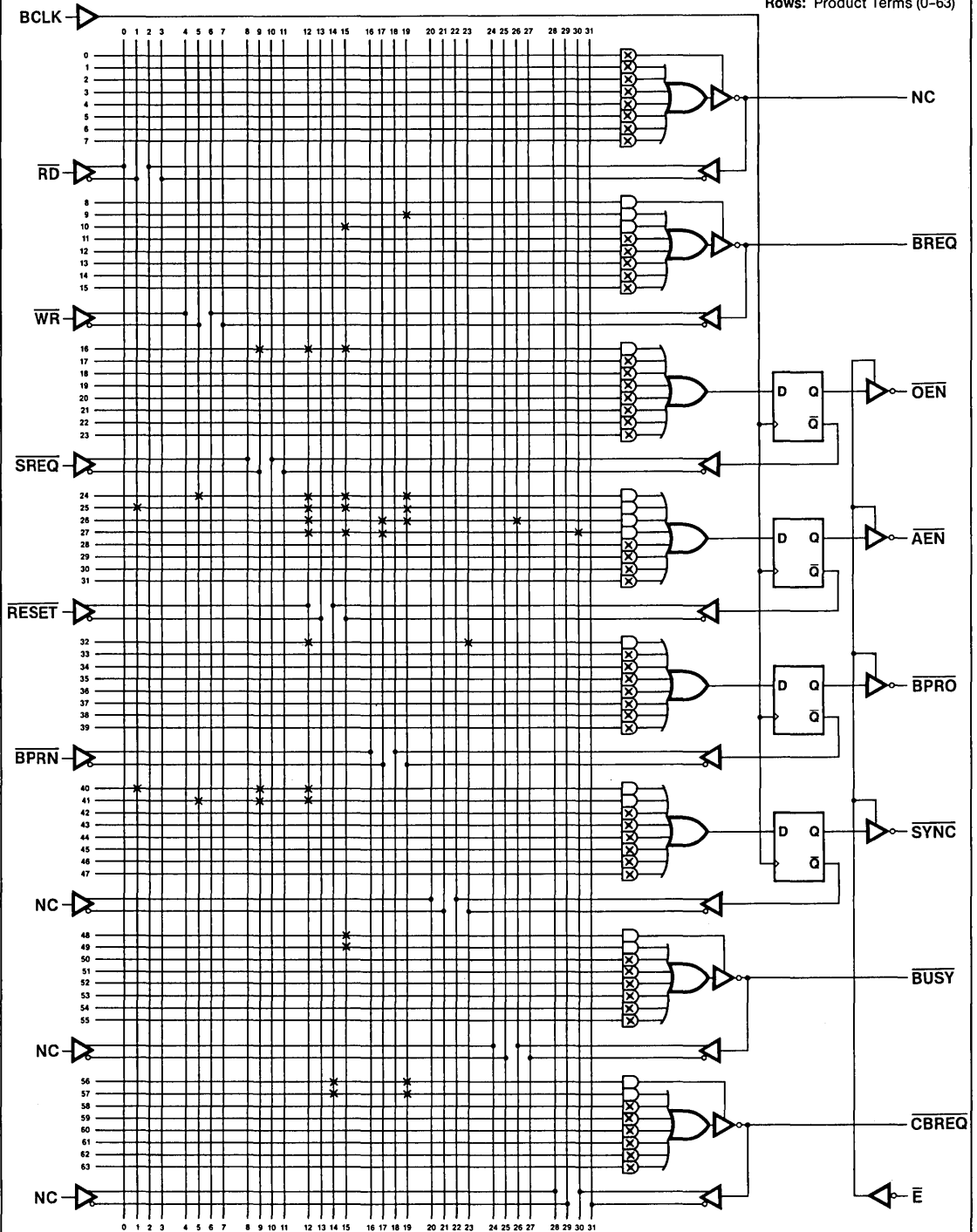
= Fuse intact   = All fuses intact   = Fuse blown

03862A-120

Am8500 to MC68000 PAL Interface

by Mark S. Young
Advanced Micro Devices

Modern 16-bit microprocessors, such as the 8086, Z8000, and 68000, are being used to form the nucleus of powerful personal/business computers and engineering workstations. However, support peripheral chips are virtually non-existent for the most recently introduced 16-bit CPUs such as the Motorola 68000. Since the modern microprocessor system depends as much on the peripheral controllers as it does on the CPU, it is important for a system designer to have a large variety of peripheral chips available. The 8500 family of peripheral chips from AMD, provides users of non-multiplexed bus microprocessors, such as the 68000, a variety of powerful peripheral chips that can be interfaced easily with a single programmable array logic (AMD PAL) device.

THE Am8500 FAMILY

The Am8500 family is a group of programmable peripheral chips which offload a variety of system functions from the main CPU. They support a variety of operating modes which are specified by writing to their control registers. The current members of the family include the Am8536 Counter/Timer and Parallel I/O Unit (CIO), the Am8038 FIFO Input/Output Interface Unit (FIO), and the Am8530 Serial Communications Controller (SCC). While the object of this article is not to discuss the capabilities of the Am8500 family, a brief overview is necessary to fully understand its interface requirements.

The Am8536 is a counter/timer chip which has available three 16-bit counters. These timer/counters have features such as duty cycle control (pulsed, one-shot, or square waved), retriggering options, and external access control lines. The CIO also provides up to 20 lines of programmable I/O ports. The Am8038 FIO is an asynchronous 128 byte buffer specially designed to be used by two CPUs or a CPU and a peripheral device as a communication or data buffer. It supports a variety of handshake interfaces on both I/O ports. Finally, the Am8530 SCC is a dual channel, multi-protocol data communications peripheral. The SCC functions as a serial to parallel, parallel to serial converter/controller. It supports a wide variety of serial communications protocols and includes extensive on-board hardware such as baud rate generators, digital phase-locked-loops, and crystal oscillators to reduce the need for external logic.

The Control and frequently used Data registers are accessed in a different manner. These registers are accessed using a single cycle or write. This scheme allows the CPU to interact efficiently with the 8500 peripherals during normal use. The slower, clumsier initialization procedure is used much less frequently and protects the user from altering the operation mode accidentally.

All the members of the Am8500 family are controlled and configured by software. The host CPU initializes the Am8500 operating modes by writing to the internal mode/options registers. The internal mode registers are not directly addressable by the CPU like the Control and some data registers. Instead, a two cycle process is needed to write to them. First, the address of the mode/options register being modified is written to the Control register; next, the data is written to the mode/options register via the Control register. The Am8500 peripheral has an internal state machine to keep track of whether address or data is being written to the Control register. Reading the value of the mode/options register is accomplished by first, writing an address to the Control register, and second, reading the mode/options data from the Control register.

DESIGN REQUIREMENTS

There are several problems associated with interfacing a general purpose peripheral device to a CPU. One major problem involves the various control signals each chip uses. Unless the two families are designed to be pin for pin compatible (e.g., the AMD/Intel 8086/8087/8089) there generally is going to be minor variations between them; the same problem exists when interfacing the 8500 peripherals to the 68000. Part of the pin incompatibility involves genuine signal differences while other pins only require name changes.

The data pins (D_0–D_7) on the 8500 parts are connected directly to the lower 8 data lines on the 68000 bus. The register select pins (A_0, A_1, A/$\overline{B}$, D/$\overline{C}$)* can be directly connected to A_1 and A_2 of the 68000 address bus. The $\overline{RD}$ and $\overline{WR}$ lines have to be generated from the 68000's R/$\overline{W}$ and $\overline{AS}$/$\overline{UDS}$/$\overline{LDS}$ signals. The 8500 clock (PCLK) is generated by dividing down the 68000 clock.

*Note: The register select/control pins have different names on each of the 8500 peripherals.

The Interrupt Request line ($\overline{\text{INT}}$) can be wire-ORed together and connected to one of the $\overline{\text{IPL}_0}$–$\overline{\text{IPL}_2}$ inputs on the 68000, giving all the peripherals a common interrupt priority level. An alternate method might be to give each of the peripherals a separate priority level (which would require priority encoding). The interrupt acknowledge line must be generated from the CPU status lines (FC_0–FC_2) by the PAL. Whenever an interrupt acknowledge cycle is started, FC_0–FC_2 equal all ones. The Interrupt daisy chain control pins (IEI and IEO on each 8500 device) are tied together in a standard priority daisy chain arrangement (see Figure 1). When implementing the daisy chain, arbitration delay down the chain must be accounted for in the PAL signal generation logic. The chip enable pins for each of the 8500 devices must come from the system memory mapping logic. The system designer must also provide an 8500 PAL enable line to select the PAL controller whenever any one of the 8500 devices has been

selected. The $\overline{\text{DTACK}}$ signal back to the CPU will be generated by the PAL logic using an internally implemented state counter to generate the correct timing. The output is implemented as a simulated open collector output so that other non-Am8500 peripherals in the system can use the $\overline{\text{DTACK}}$ line.

Another problem with interfacing general purpose peripherals to the 68000 is timing. Most peripherals run at speeds considerably slower than the 8, 10, 12, and 16MHz CPUs being produced today. This means using either a slower clock or dividing down the CPU clock. In the case of the 8500 family this generally means dividing the CPU clock in half and using a CPU operating at less than or equal to 12MHz. Aside from just speed problems, system integrators frequently have to
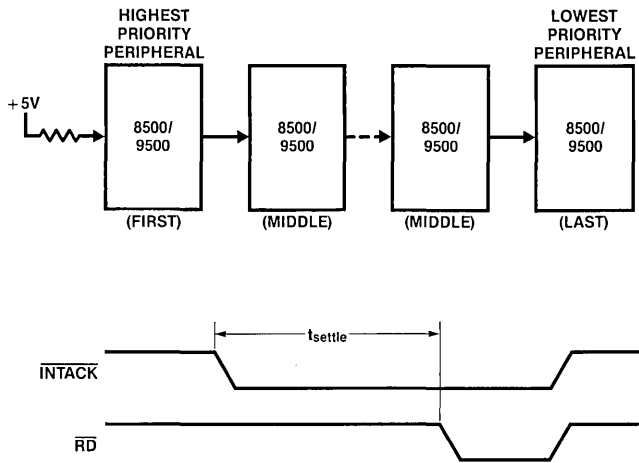
03862A-121

Figure 1

Table 1. Interrupt Daisy Chain/Propagation Delay

Chain Position (ns)				
Peripheral		First	Middle	Last
8536	CIO	350	150	100
8038	FIO	350	150	100
8530	SCC	250	120	120

Note: First position timing is INTACK to IEO.
Middle position timing is IEI to IEO.
Last position timing is IEI to data strobe set-up.

03862A-122

tackle subtle timing differences between signals or from devised signal equivalents, e.g., deriving the Am8500 $\overline{RD}$, $\overline{WR}$, and $\overline{DTACK}$ from the 68000's $\overline{LDS}$ and R/$\overline{W}$; or guaranteeing data set-up and hold times.

The 68000 has two ways of interfacing to peripherals such as the 8500 family. The first uses the special VPA (Valid Peripheral Address) input pin on the 68000. The VPA pin can be activated by the Am8500 device select logic at the start of a cycle to tell the 68000 that a peripheral is being accessed. This interface was designed to allow the slow, synchronous bus 6800 peripherals to talk to the 68000's asynchronous bus until the new 68000 peripherals could be produced. Also, the VPA interface has a slow access rate (a minimum peripheral access time of over 1000ns not including recovery time) which would slow down the CPU considerably. And, since all the 68000 peripherals are being designed to use the asynchronous method, this interface will not be discussed.

When writing data, the 68000 puts address and data onto their respective buses and uses the $\overline{DTACK}$ line as a "got data successfully" handshake from the selected device. When the $\overline{DTACK}$ line is recognized, the 68000 removes address and data one CPU clock later. This method allows the user to take advantage of the asynchronous bus of the 68000. The major difference between the 8500 family and the 68000 $\overline{DTACK}$ timing is the way data is strobed in and out of the 8500 chips. The 8500 devices sample the data on the falling edge of $\overline{WR}$. The 68000 asserts an address (when reading) onto the bus and then uses the $\overline{DTACK}$ signal from the selected peripheral (memory included) to indicate valid data and then

samples on the next falling edge of the CPU clock. The other method of interfacing the Am8500 family to the 68000 uses the Data Transfer Acknowledge ($\overline{DTACK}$) cycle.

DESIGN APPROACH

Two different methods of interfacing 8500 devices to the 68000 bus will be presented. One method allows the user to obtain fast access to all the 8500 devices. However, some minimum software requirements are imposed. The other interface slows down the access rate by the CPU but guarantees all 8500 minimum timing specifications and imposes no software overhead.

There are several timing requirements imposed by the 8500 family. The first involves read/write access to the parts. The 8500 (4MHz) peripherals have a read/write/interrupt acknowledge timing as shown in Figure 2. The minimum read/write access time is 400ns. This means the PAL interface must guarantee a valid access cycle of greater than 400ns (by forcing the 68000 to execute several wait states).

The basic read or write cycle generated by the PAL interface looks like Figure 3. The 68000 R/$\overline{W}$ and $\overline{LDS}$ lines have been converted into 8500 $\overline{RD}$ and $\overline{WR}$ control signals and with a state timing generator, produce the 68000 data valid signal $\overline{DTACK}$. While the 8500 peripherals latch the data internally on the falling edge of $\overline{WR}$, all 9500 (Intel-type) peripherals use the rising edge of $\overline{WR}$ to strobe in data. So, the timing used is designed to guarantee proper set-up and hold time for both Am8500 and Am9500 devices.
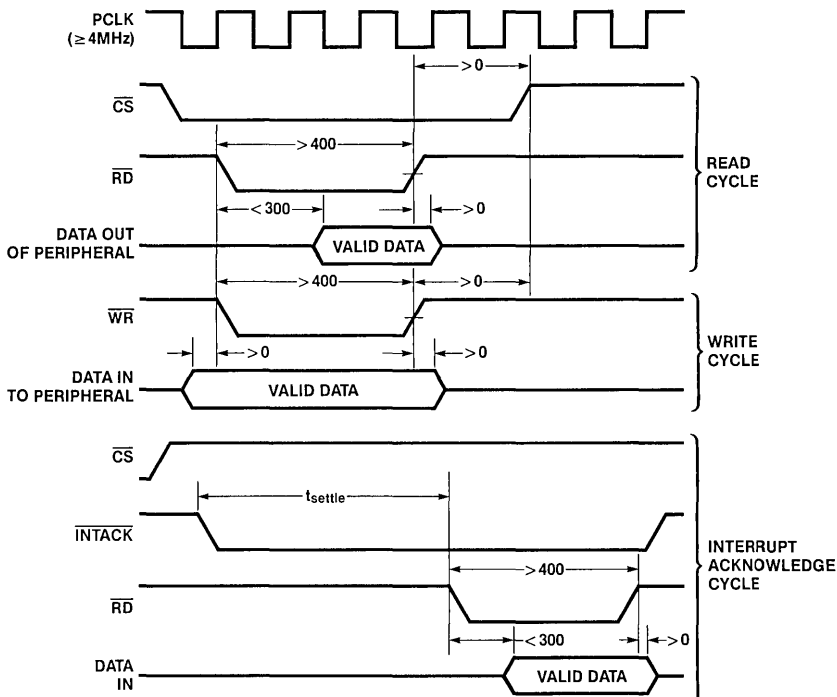
03862A-123

Figure 2. Am8500 Interface Timing (4MHz)

The $\overline{\text{DTACK}}$ control logic is the only control line which employs any sort of special timing in both of the PAL interfaces. In order to guarantee proper set-up and hold time for write operations to the 9500 parts, it was necessary to start the $\overline{\text{DTACK}}$ cycle in the middle of a PCLK cycle. Hence, it was necessary to use the CPU clock to condition the assertion of $\overline{\text{DTACK}}$. Using the C_0 (PCLK) and the C_1–C_3 inputs only, would have allowed a potential set-up time violation during a write operation under worst case conditions for an 8MHz 68000.

The interrupt acknowledge cycle is very similar to the read/write cycle; only two differences exist. First, the interrupt acknowledge cycle involves only a read operation (see

interrupt acknowledge timing in Figure 3). Secondly, the read cycle needs to be stretched out to allow time for the interrupt daisy chain to resolve priority. If a parallel priority resolution scheme is used, then only the priority decode time delay and peripheral response time is added on to the interrupt cycle. The interrupt time delay varies, based on the number of 8500 devices in the daisy chain. The time delay is based on the 8500's position in the chain: first, somewhere in the middle, and the last device in the daisy chain (see Table 1). Both of the current PAL interfaces assume there are three 8500 peripherals in the interrupt daisy chain.
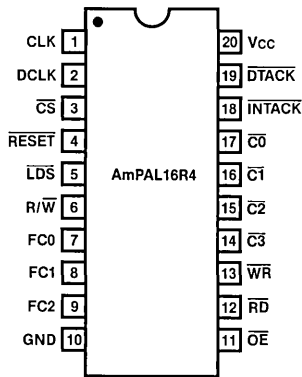
Figure 3a 03862A-124
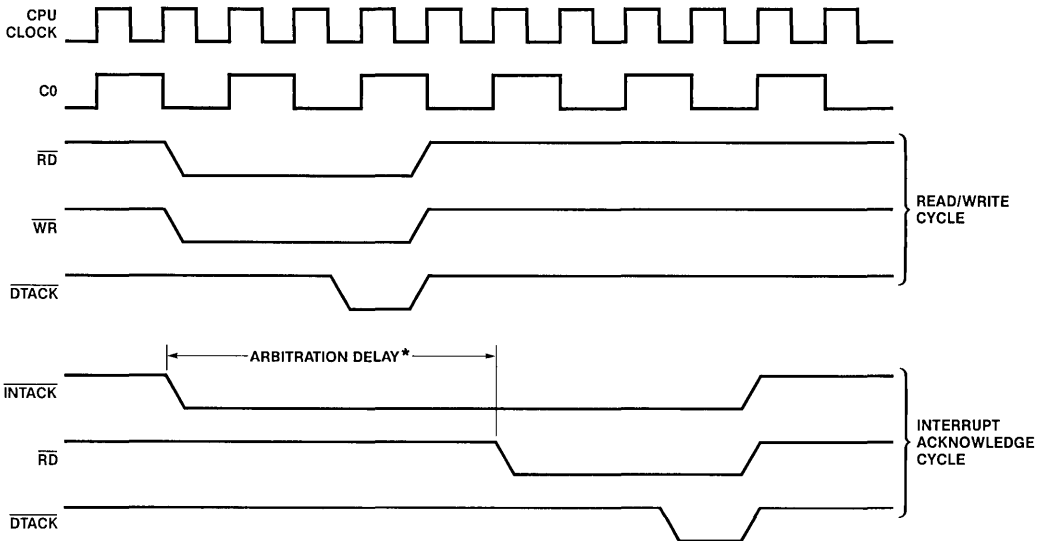
03862A-125

*Delay time assumes three Am8500 devices in the daisy chain.

Note: $\overline{\text{RD}}$ and $\overline{\text{WR}}$ may not be asserted LOW simultaneously.

Figure 3b. PAL Generated Interface Signals

The PAL interfaces offered are designed to give the system designer maximum flexibility in integrating Am8500 peripherals with 68000 based systems. The first version is designed to allow maximum access to the 8500 devices (see Figure 4a). It does this by delegating the read/write recovery time into software. All 8500 peripherals have a minimum post access recovery time, i.e., they can't be accessed for a minimum period of time after being read or written (see Table 2). Generally, this restriction manifests itself only if the CPU has to make repeated accesses to the same peripheral part rapidly. While the instruction fetch time of the CPU allows for some recovery time, it doesn't guarantee enough time (since the average recovery time is approximately 1000ns and a 68000 instruction fetch requires a minimum of 500ns@8MHz). Hence, the first design requires the user to implement minimum software recovery time.

The software recovery routines in this case generally take the form of executing 1–2 instructions (depending on execution and fetch time) in between accessing the *same* 8500 device. For most systems, these instruction executions can be used to process the data just received. Another method of insuring the minimum peripheral recovery time is to juggle the accessing of the 8500 devices in the system so the recovery time requirement is not violated.

The second design (see slow PAL timing in Figure 4b) relieves the user of all software considerations when using the Am8500 parts. The recovery time is built into the PAL design. This is done by delaying access on the read/write and then taking advantage of the 68000 next instruction fetch to guarantee that the minimum recovery time is given. Also, a minor change was required in the interrupt acknowledge timing, i.e., stretching out the INTACK timing slightly to avoid a potential glitch on the RD line after an interrupt acknowledge cycle.

The advantage of software-independent hardware is offset by longer read/write cycles to the peripherals, even for single accesses. Also, the user is denied access to another 8500 peripheral until the minimum recovery time has been met for the previous one. However, having software-independent hardware is sometimes an important feature in a system; and slowing down the peripheral access rate slightly is a small price to pay for it. Note, the interrupt acknowledge cycles for both designs are virtually the same. This occurred because the normal interrupt processing by the 68000 guarantees that another access to the 8500 parts cannot occur in time to violate their access recovery times. Hence no software delay is needed for the fast access interface.

The interrupt acknowledge delay (for the daisy chained priority resolution scheme) in this example has been chosen by using an assumption of three 8500 peripherals in the chain. Larger or smaller numbers of parts in the daisy chain would increase or decrease this result with minor changes to the PAL logic equations. The design is flexible enough to support the addition of at least 3 more peripherals in the daisy chain.

The PAL equations and logic diagram for both designs are given at the end of the article. The equations were derived directly from their respective timing diagrams (Figures 4a and 4b). Some obvious logic simplification was done on the initial equations to reduce the number of terms. The integration of the Am8500 peripherals and the PAL timing generator are shown in a sample configuration in Figure 5.

Finally, the design presented was optimized for an 8MHz 68000 system and 4MHz 8500 parts. The timing/state counter $(C_0–C_4)$ only counts as far as it is needed. Higher performance CPUs, up to 12MHz, can be used with this interface, but 6MHz 8500 parts will have to be used.

Table 2. Peripheral Access Recovery Time

Peripheral (4MHz)		Recovery Time
8530	SCC	Greater than 6 PCLK cycles + 200ns
8536	CIO	Greater than 3 PCLK cycles or 1000ns
8038	FIO	Greater than 1000ns

03862A-126

5-56
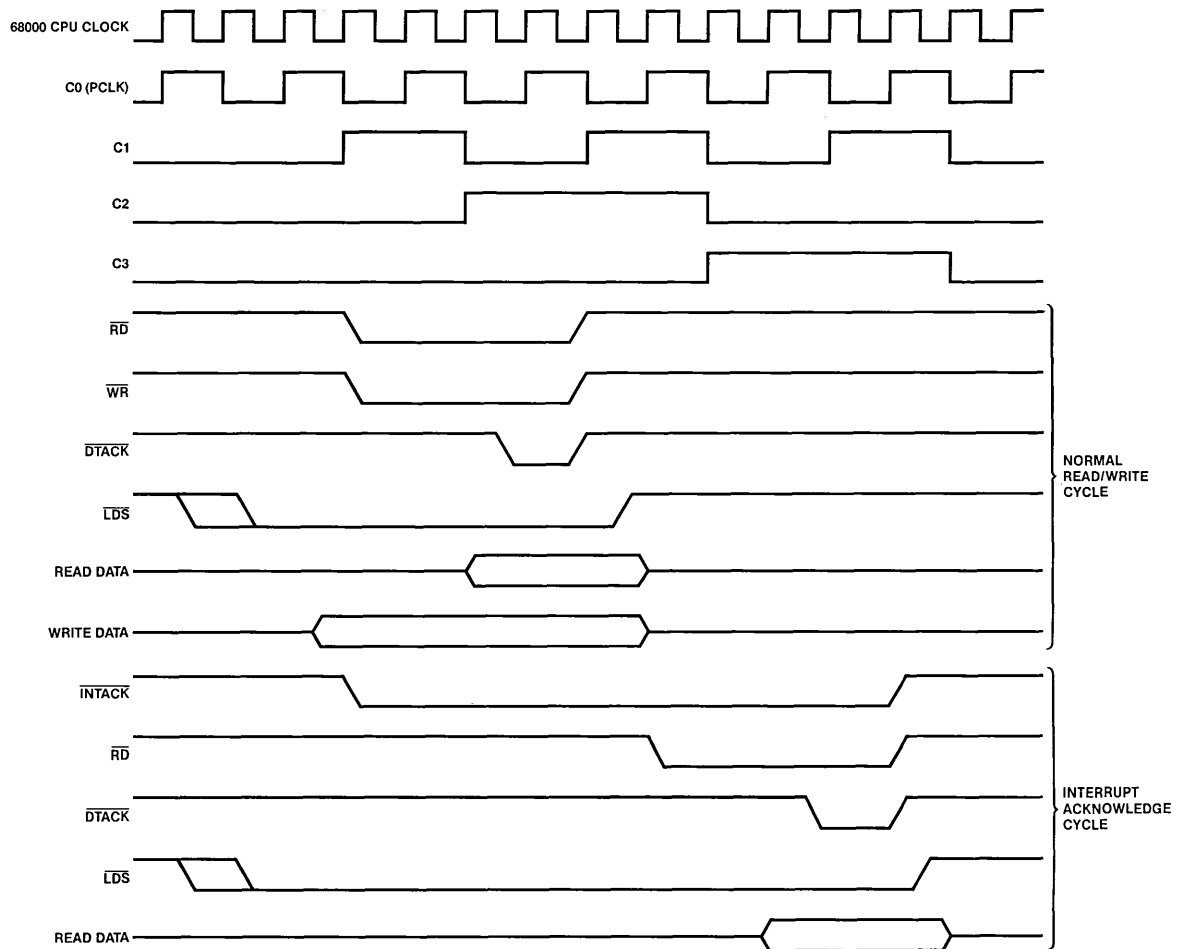
Figure 4a. "Fast" PAL Am8500 to MC68000
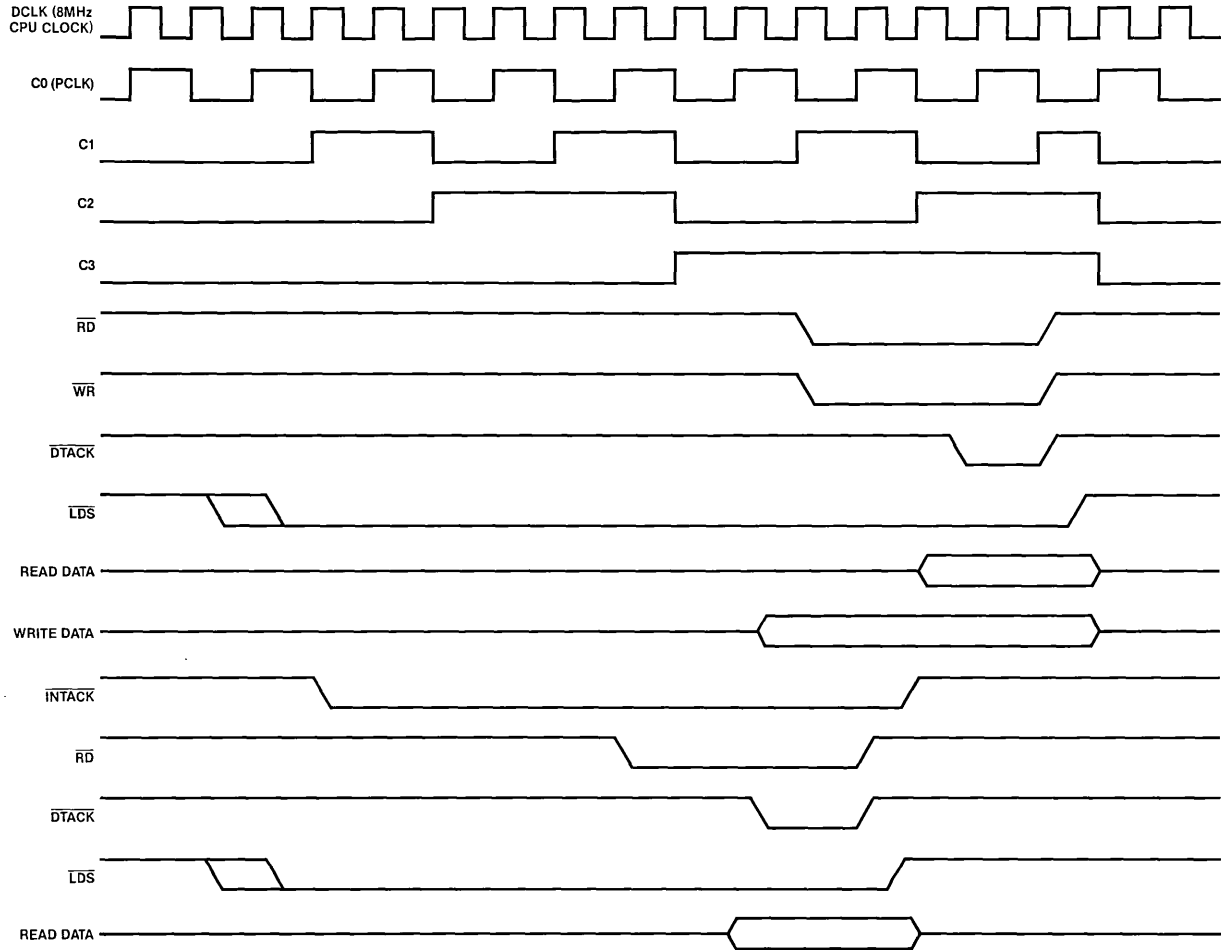
03862A-127

Figure 4b. "Slow" PAL Am8500 to MC68000

03862A-128

(FROM 68000)

A₁, A₂

CPU CLK
$\overline{CS}$
$\overline{RESET}$
$\overline{LDS}$
R/$\overline{W}$
FC0
FC1
FC2

AmPAL16R4

$\overline{DTACK}$

D₀–D₇

TLP₀

(TO 68000)

Am8530 SCC

Am8536 CIO

Am8038 FIO

+5V

TO NEXT PERIPHERAL

03862A-129

Figure 5. PAL Am8500 to MC68000 Hookup

```
PAL16R4                                      PAL DESIGN SPECIFICATION
PAT050                                       MARK YOUNG  1/21/83
FAST AM8500 TO MOTOROLA 68000 PAL
ADVANCED MICRO DEVICES
CLK DCLK /CS /RESET /LDS RW FCO FC1 FC2 GND
/OE /RD /WR /C3 /C2 /C1 /CO /INTACK /DTACK VCC
    :
    ; STATE MACHINE COUNTER CO - C3
CO := /CO                                    ; 8500 CLK

C1 := CO*/C1*CS*LDS*/RESET
   + /CO*C1*CS*LDS*/RESET

C2 := CO*C1*/C2*CS*/RESET
   + /C1*C2*CS*/RESET
   + /CO*C1*C2*CS*/RESET

C3 := CO*C1*C2*/C3*CS*/RESET
   + /C1*/C2*C3*CS*/RESET
   + /CO*C1*/C2*C3*CS*/RESET

RD = C1*/C2*/C3*RW*/INTACK*CS*/RESET     ; NORMAL READ
   + /C1*C2*/C3*RW*/INTACK*CS*/RESET     ; NORMAL READ
   + CO*C1*C2*/C3*INTACK*CS*/RESET       ; INTERRUPT ACKNOWLEDGE
   + RD*INTACK*CS*/RESET                 ; INTERRUPT ACKNOWLEDGE

WR = C1*/C2*/C3*/RW*CS*/INTACK*/RESET    ; WRITE OPERATION
   + /C1*C2*/C3*/RW*CS*/INTACK*/RESET    ; WRITE OPERATION

    ; DATA ACKNOWLEDGE
IF (CS) DTACK = /DCLK*/CO*/C1*C2*/C3*/INTACK*/RESET
      + DTACK*RD*/RESET
      + DTACK*WR*/RESET
      + /DCLK*CO*/C1*/C2*C3*INTACK*/RESET

    ; INTERRUPT ACKNOWLEDGE
INTACK = FCO*FC1*FC2*C1*/C3*LDS*/RESET
      + C2*FCO*FC1*FC2*/RESET
      + /C1*C3*FCO*FC1*FC2*/RESET
      + /CO*C1*C3*FCO*FC1*FC2*/RESET
```

FUNCTION TABLE:
```
  ;
  ;   NOTE: FOR THE SIMULATION, ALL THE SIGNALS USED ARE AT THE
  ;         PIN LEVEL (I.E. WHAT THE CHIP SEES AND PUTS OUT). THE
  ;         ONE EXCEPTION ARE THE C0 - C3 PINS. THESE ARE DEFINED
  ;         AT THE REGISTER OUTPUT LEVEL (NON-INVERTED) BECAUSE
  ;         THEY WERE DIRECTLY DERIVED FROM THE TIMING DIAGRAMS
  ;         AND THIS MAKES IT EASIER TO RELATE TO THE TIMING
  ;         DIAGRAM.
  ;
  CLK DCLK /CS /RESET /LDS RW FC0 FC1 FC2
  /OE /RD /WR C0 C1 C2 C3 /INTACK /DTACK
;                                                                  /
;                   /                                              I    /
;                   R                                              N    D
;         D         E    /                                         T    T
; C   C   /   S     L         F   F   F   /    /    /              A    A
; L   L   C   E     D    R    C   C   C   O    R    W    C   C   C  C    C
; K   K   S   T     S    W    0   1   2   E    D    R    0   1   2  3    K    K
----------------------------------------------------------------------------
  ;
  ; RESET SEQUENCE
  C   H   H   L   X   X   X   X   X   L   H   H   H   L   L   L   H   Z
  L   L   H   L   X   X   X   X   X   L   H   H   H   L   L   L   H   Z
  C   H   H   H   X   X   X   X   X   L   H   H   L   L   L   L   H   Z
  L   L   H   H   X   X   X   X   X   L   H   H   L   L   L   L   H   Z
  C   H   H   H   X   X   X   X   X   L   H   H   H   L   L   L   H   Z
  L   L   H   H   X   X   X   X   X   L   H   H   H   L   L   L   H   Z
  ;
;WRITE OPERATION (RW=L)
  C   H   H   H   X   X   X   X   X   L   H   H   L   L   L   L   H   Z
  L   L   H   H   X   X   X   X   X   L   H   H   L   L   L   L   H   Z
  C   H   H   H   X   X   X   X   X   L   H   H   H   L   L   L   H   Z
  L   L   H   H   X   X   X   X   X   L   H   H   H   L   L   L   H   Z
  C   H   H   H   X   X   X   X   X   L   H   H   L   L   L   L   H   Z
  L   L   L   H   L   L   X   X   X   L   H   H   L   L   L   L   H   H
  C   H   L   H   L   L   X   X   X   L   H   H   H   L   L   L   H   H
  L   L   L   H   L   L   X   X   X   L   H   H   H   L   L   L   H   H
  C   H   L   H   L   L   X   X   X   L   H   L   L   H   L   L   H   H
  L   L   L   H   L   L   X   X   X   L   H   L   L   H   L   L   H   H
  C   H   L   H   L   L   X   X   X   L   H   L   H   H   L   L   H   H
  L   L   L   H   L   L   X   X   X   L   H   L   H   H   L   L   H   H
  C   H   L   H   L   L   X   X   X   L   H   L   L   L   H   L   H   H
  L   L   L   H   L   L   X   X   X   L   H   L   L   L   H   L   H   L
  C   H   L   H   L   L   X   X   X   L   H   L   H   L   H   L   H   L
  L   L   L   H   L   L   X   X   X   L   H   L   H   L   H   L   H   L
  C   H   L   H   L   H   X   X   X   L   H   H   L   H   H   L   H   H
  L   L   H   H   X   X   X   X   X   L   H   H   L   H   H   L   H   Z
  C   H   H   H   X   X   X   X   X   L   H   H   H   L   L   L   H   Z
  L   L   H   H   X   X   X   X   X   L   H   H   H   L   L   L   H   Z
  ;
;INTACK CYCLE
  C   H   H   H   X   X   X   X   X   L   H   H   L   L   L   L   H   Z
  L   L   H   H   X   X   X   X   X   L   H   H   L   L   L   L   H   Z
  C   H   H   H   X   X   X   X   X   L   H   H   H   L   L   L   H   Z
```

```
L  L  H  H  X  X  X  X  X  L  H  H  H  L  L  L  H  Z
C  H  L  H  L  H  H  H  H  L  H  H  L  H  L  L  L  H
L  L  L  H  L  H  H  H  H  L  H  H  L  H  L  L  L  H
C  H  L  H  L  H  H  H  H  L  H  H  H  H  L  L  L  H
L  L  L  H  L  H  H  H  H  L  H  H  H  H  L  L  L  H
C  H  L  H  L  H  H  H  H  L  H  H  L  L  H  L  L  H
L  L  L  H  L  H  H  H  H  L  H  H  L  L  H  L  L  H
C  H  L  H  L  H  H  H  H  L  H  H  H  L  H  L  L  H
L  L  L  H  L  H  H  H  H  L  H  H  H  L  H  L  L  H
C  H  L  H  L  H  H  H  H  L  H  H  L  H  H  L  L  H
L  L  L  H  L  H  H  H  H  L  H  H  L  H  H  L  L  H
C  H  L  H  L  H  H  H  H  L  L  H  H  H  H  L  L  H
L  L  L  H  L  H  H  H  H  L  L  H  H  H  H  L  L  H
C  H  L  H  L  H  H  H  H  L  L  H  L  L  L  H  L  H
L  L  L  H  L  H  H  H  H  L  L  H  L  L  L  H  L  H
C  H  L  H  L  H  H  H  H  L  L  H  H  L  L  H  L  H
L  L  L  H  L  H  H  H  H  L  L  H  H  L  L  H  L  L
C  H  L  H  L  H  H  H  H  L  L  H  L  H  L  H  L  L
L  L  L  H  L  H  H  H  H  L  L  H  L  H  L  H  L  L
C  H  L  H  L  H  H  H  H  L  H  H  H  H  L  H  H  H
L  L  H  H  H  X  X  X  X  L  H  H  H  H  L  H  H  Z
C  H  H  H  H  X  X  X  X  L  H  H  L  L  L  L  H  Z
L  L  H  H  H  X  X  X  X  L  H  H  L  L  L  L  H  Z
-----------------------------------------------------------------------
```

DESCRIPTION:
THE FASTZTOM PAL PROVIDES INTERFACING BETWEEN THE SINGLE CHIP 8500 TO
THE 68000. THE PAL USED IS A MEANS TO PROVIDE FOR THE FASTEST POSSIBLE
INTERFACE. THE REQUIREMENTS FOR THIS MAXIMUM ACCESS UTILIZES IMPLE-
MENTED SOFTWARE FOR REPEATED ACCESSES. THE USER, HENCE, IS ABLE TO GET
MAXIMUM ACCESS WITH MINIMAL WAIT STATE INSERTION. THE INTERFACE
PROVIDES TOTAL SIGNAL COMPATIBILITY.

FAST AM8500 TO MOTOROLA 68000 PAL
ADVANCED MICRO DEVICES
*D9724
F0
L0000 1111 1011 1111 1111 1111 1111 1111 1111 *
L0032 1011 1101 0101 1101 1110 1101 1111 1111 *
L0064 1110 1111 0111 1111 1111 1111 1111 1110 *
L0096 1110 1111 0111 1111 1111 1111 1110 1111 *
L0128 1011 1110 0110 1101 1101 1110 1111 1111 *
L0256 1111 1111 1111 1111 1111 1111 1111 1111 *
L0288 1111 1111 0111 1010 1111 0101 0111 0111 *
L0320 1111 1111 0111 1111 1110 0111 0111 0111 *
L0352 1111 1111 0111 1101 1111 0110 0111 0111 *
L0384 1111 1111 0101 1110 1111 0110 0111 0111 *
L0512 1111 1111 1101 1111 1111 1111 1111 1111 *
L0768 1111 1011 0110 1001 1111 1111 1111 1111 *
L0800 1111 1011 0101 1010 1111 1111 1111 1111 *
L1024 1111 1011 0110 1110 1101 1111 1111 1111 *
L1056 1111 1011 0111 1101 1110 1111 1111 1111 *
L1088 1111 1011 0101 1110 1110 1111 1111 1111 *
L1280 1111 1011 0110 1110 1110 1101 1111 1111 *
L1312 1111 1011 0111 1101 1101 1110 1111 1111 *
L1344 1111 1011 0101 1110 1101 1110 1111 1111 *
L1536 1111 1111 1111 1111 1111 1111 1111 1111 *
L1568 1111 1001 0111 1110 1001 1101 1111 1111 *
L1600 1111 1001 0111 1101 1010 1101 1111 1111 *
L1792 1111 1111 1111 1111 1111 1111 1111 1111 *
L1824 1111 1001 0111 1110 0101 1101 1111 1111 *
L1856 1111 1001 0111 1101 0110 1101 1111 1111 *
L1888 1111 1010 0110 1110 1110 1101 1111 1111 *
L1920 1111 1010 0111 1111 1111 1111 1111 1110 *
C58C0*
V0001 C110XXXXX00HHHHHLHZ1 *
V0002 0010XXXXX00HHHHHLHZ1 *
V0003 C111XXXXX00HHHHHHHZ1 *
V0004 0011XXXXX00HHHHHHHZ1 *
V0005 C111XXXXX00HHHHHLHZ1 *
V0006 0011XXXXX00HHHHHLHZ1 *
V0007 C111XXXXX00HHHHHHHZ1 *
V0008 0011XXXXX00HHHHHHHZ1 *
V0009 C111XXXXX00HHHHHLHZ1 *
V0010 0011XXXXX00HHHHHLHZ1 *
V0011 C111XXXXX00HHHHHHHZ1 *
V0012 000100XXX00HHHHHHHH1 *
V0013 C10100XXX00HHHHHLHH1 *
V0014 000100XXX00HHHHHLHH1 *
V0015 C10100XXX00HLHHLHHH1 *
V0016 000100XXX00HLHHLHHH1 *
V0017 C10100XXX00HLHHLLHH1 *
V0018 000100XXX00HLHHLLHH1 *
V0019 C10100XXX00HLHLHHHH1 *
V0020 000100XXX00HLHLHHHL1 *
V0021 C10100XXX00HLHLHLHL1 *

```
V0022 000100XXX00HLHLHLHL1 *
V0023 C10101XXX00HHHLLHHH1 *
V0024 0011XXXXX00HHHLLHHZ1 *
V0025 C111XXXXX00HHHHHLHZ1 *
V0026 0011XXXXX00HHHHHLHZ1 *
V0027 C111XXXXX00HHHHHHHZ1 *
V0028 0011XXXXX00HHHHHHHZ1 *
V0029 C111XXXXX00HHHHHLHZ1 *
V0030 0011XXXXX00HHHHHLHZ1 *
V0031 C1010111100HHHHLHLH1 *
V0032 00010111100HHHHLHLH1 *
V0033 C1010111100HHHHLLLH1 *
V0034 00010111100HHHHLLLH1 *
V0035 C1010111100HHHLHHLH1 *
V0036 00010111100HHHLHHLH1 *
V0037 C1010111100HHHLHLLH1 *
V0038 00010111100HHHLHLLH1 *
V0039 C1010111100HHHLLHLH1 *
V0040 00010111100HHHLLHLH1 *
V0041 C1010111100LHHLLLLH1 *
V0042 00010111100LHHLLLLH1 *
V0043 C1010111100LHLHHHLH1 *
V0044 00010111100LHLHHHLH1 *
V0045 C1010111100LHLHHLLH1 *
V0046 00010111100LHLHHLLL1 *
V0047 C1010111100LHLHLHLL1 *
V0048 00010111100LHLHLHLL1 *
V0049 C1010111100HHLHLLHH1 *
V0050 00111XXXX00HHLHLLHZ1 *
V0051 C1111XXXX00HHHHHHHZ1 *
V0052 00111XXXX00HHHHHHHZ1 *
7BCF
```

Logic Diagram for Fast Am8500 to MC68000 Using AmPAL16R4/AmPAL16R4A

Columns: Inputs (0–31)
Rows: Product Terms (0–63)
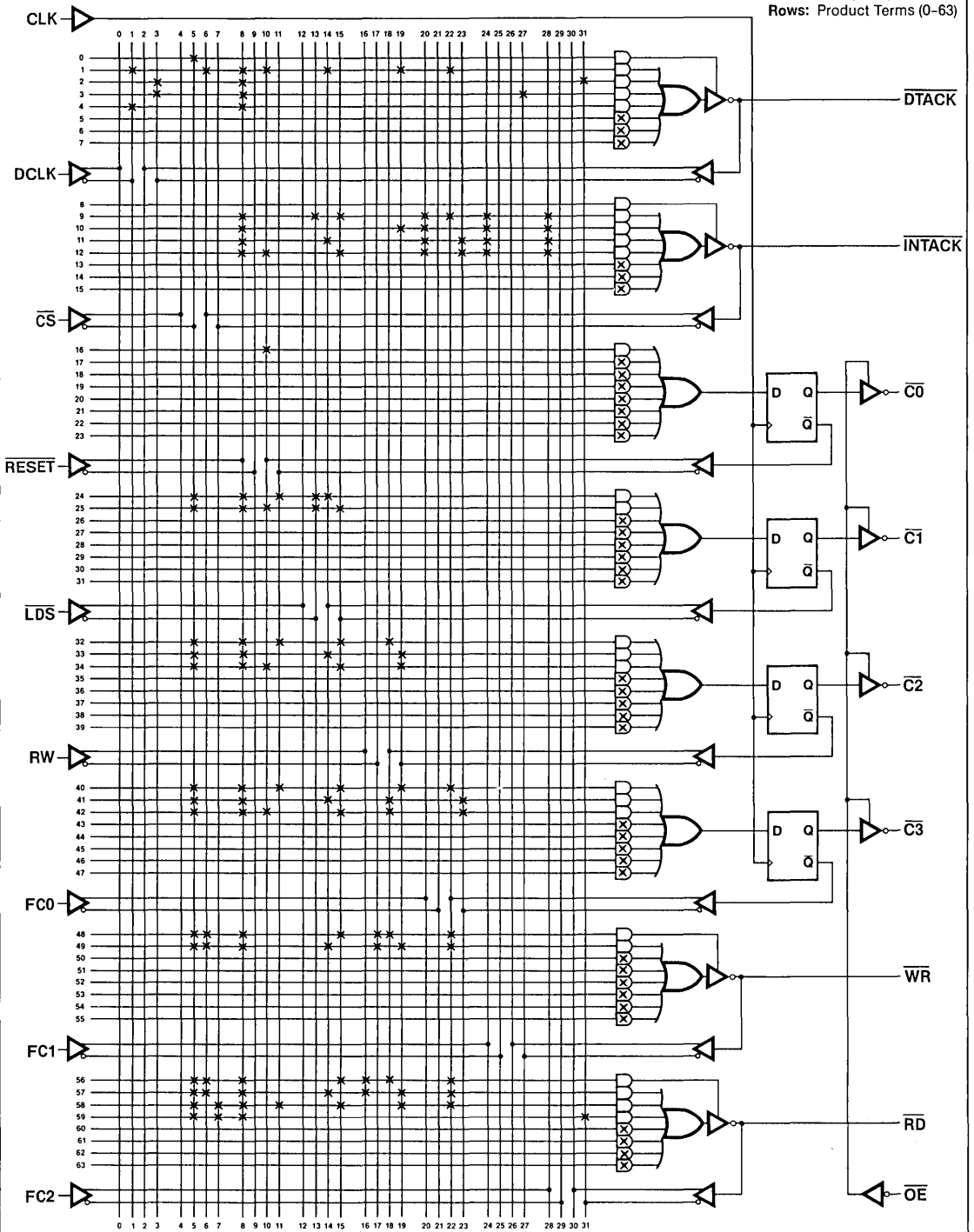
✳ = Fuse intact   ⊠ = All fuses intact   ┼ = Fuse blown

03862A-130

```
PAL16R4                                         PAL DESIGN SPECIFICATION
PAT051                                          MARK YOUNG    1/21/83
SLOW AM8500/9500 TO MOTOROLA 68000 PAL
ADVANCED MICRO DEVICES
CLK DCLK /CS /RESET /LDS RW FCO FC1 FC2 GND
/OE /RD /WR /C3 /C2 /C1 /CO /INTACK /DTACK VCC
    ;
    ; STATE MACHINE COUNTER CO - C3
    ;
CO :=  /CO                                      ; 8500 CLK

C1 :=  CO*/C1*/C2*LDS*CS*/RESET
    +  /CO*C1*/C2*LDS*CS*/RESET
    +  CO*/C1*C2*LDS*CS*/RESET
    +  /CO*C1*C2*C2*/C3*LDS*CS*/RESET

C2 :=  CO*C1*/C2*CS*/RESET
    +  /C1*C2*CS*/RESET
    +  /CO*C1*C2*/C3*CS*/RESET

C3 :=  CO*C1*C2*/C3*CS*/RESET
    +  /C2*C3*CS*/RESET
    +  /C1*C2*C3*CS*/RESET

RD =   C1*/C2*C3*RW*CS*/INTACK*/RESET    ; NORMAL READ
    +  /C1*C2*C3*RW*CS*/INTACK*/RESET    ; NORMAL READ
    +  CO*C1*C2*/C3*INTACK*/RESET        ; INTERRUPT ACKNOWLEDGE
    +  /C1*/C2*C3*INTACK*/RESET          ; INTERRUPT ACKNOWLEDGE
    +  /CO*C1*/C2*C3*INTACK*/RESET       ; INTERRUPT ACKNOWLEDGE

WR =   C1*/C2*C3*/RW*CS*/INTACK*/RESET   ; WRITE OPERATION
    +  /C1*C2*C3*/RW*CS*/INTACK*/RESET   ; WRITE OPERATION

       ; DATA ACKNOWLEDGE
IF (CS)  DTACK =  /DCLK*/CO*/C1*C2*C3*/INTACK*/DTACK*/RESET
    +  DTACK*RD*/RESET
    +  DTACK*WR*/RESET
    +  /DCLK*CO*/C1*/C2*C3*INTACK*/RESET

       ; INTERRUPT ACKNOWLEDGE
INTACK =  FCO*FC1*FC2*C1*/C3*LDS*CS*/RESET
    +  C2*/C3*FCO*FC1*FC2*CS*/RESET
    +  /C2*C3*FCO*FC1*FC2*CS*/RESET
```
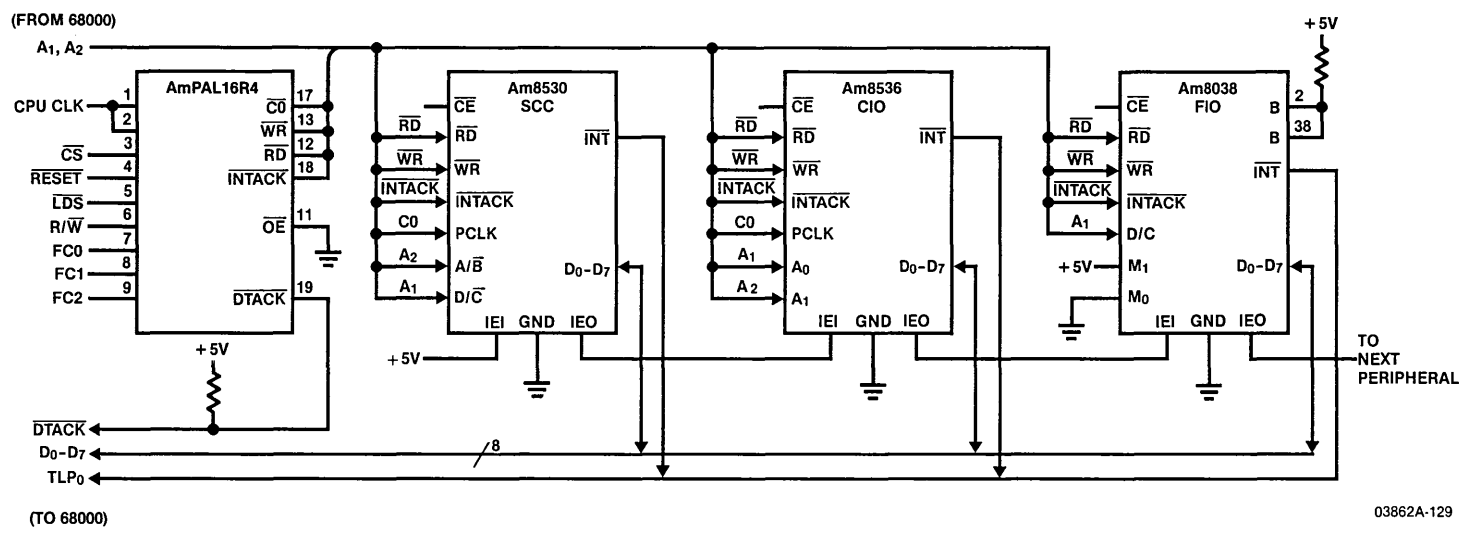
```
    FUNCTION TABLE:


    ;
    ;    NOTE: FOR THE SIMULATION, ALL THE SIGNALS USED ARE AT THE
    ;          PIN LEVEL (I.E. WHAT THE CHIP SEES AND PUTS OUT). THE
    ;          ONE EXCEPTION ARE THE C0 - C3 PINS. THESE ARE DEFINED
    ;          AT THE REGISTER OUTPUT LEVEL (NON-INVERTED) BECAUSE
    ;          THEY WERE DIRECTLY DERIVED FROM THE TIMING DIAGRAMS
    ;          AND THIS MAKES IT EASIER TO RELATE TO THE TIMING
    ;          DIAGRAM.


    CLK DCLK /CS /RESET /LDS RW FC0 FC1 FC2
    /OE /RD /WR C0 C1 C2 C3 /INTACK /DTACK

    ;
    ;                                                            /
    ;                /                                           I    /
    ;                R                                           N    D
    ;         D      E    /                                      T    T
    : C    C    /    S    L         F    F    F    /    /    /                        A    A
    ; L    L    C    E    D    R    C    C    C    O    R    W    C    C    C    C    C    C
    : K    K    S    T    S    W    0    1    2    E    D    R    0    1    2    3    K    K
    ------------------------------------------------------------------------------------------
    ;
    ;RESET SEQUENCE
      C    H    H    L    X    X    X    X    X    L    H    H    H    L    L    L    H    Z
      L    L    H    L    X    X    X    X    X    L    H    H    H    L    L    L    H    Z
      C    H    H    H    X    X    X    X    X    L    H    H    L    L    L    L    H    Z
      L    L    H    H    X    X    X    X    X    L    H    H    L    L    L    L    H    Z
      C    H    H    H    X    X    X    X    X    L    H    H    H    L    L    L    H    Z
      L    L    H    H    X    X    X    X    X    L    H    H    H    L    L    L    H    Z
    ;
    ;WRITE OPERATION (RW=L)
      C    H    H    H    X    X    X    X    X    L    H    H    L    L    L    L    H    Z
      L    L    H    H    X    X    X    X    X    L    H    H    L    L    L    L    H    Z
      C    H    H    H    X    X    X    X    X    L    H    H    H    L    L    L    H    Z
      L    L    H    H    X    X    X    X    X    L    H    H    H    L    L    L    H    Z
      C    H    H    H    X    X    X    X    X    L    H    H    L    L    L    L    H    Z
      L    H    L    H    L    L    X    X    X    L    H    H    L    L    L    L    H    H
      C    H    L    H    L    L    X    X    X    L    H    H    H    L    L    L    H    H
      L    L    L    H    L    L    X    X    X    L    H    H    H    L    L    L    H    H
      C    H    L    H    L    L    X    X    X    L    H    L    L    H    L    L    H    H
      L    L    L    H    L    L    X    X    X    L    H    L    L    H    L    L    H    H
      C    H    L    H    L    L    X    X    X    L    H    H    H    H    L    L    H    H
      L    L    L    H    L    L    X    X    X    L    H    H    H    H    L    L    H    H
      C    H    L    H    L    L    X    X    X    L    H    H    L    L    H    L    H    H
      L    L    L    H    L    L    X    X    X    L    H    H    L    L    H    L    H    H
      C    H    L    H    L    L    X    X    X    L    H    H    H    L    H    L    H    H
      L    L    L    H    L    L    X    X    X    L    H    H    H    L    H    L    H    H
      C    H    L    H    L    H    X    X    X    L    H    H    L    H    H    L    H    H
      L    L    L    H    L    L    X    X    X    L    H    H    L    H    H    L    H    H
      C    H    L    H    L    L    X    X    X    L    H    H    H    H    H    L    H    H
      L    L    L    H    L    L    X    X    X    L    H    H    H    H    H    L    H    H
      C    H    L    H    L    L    X    X    X    L    H    H    L    L    L    H    H    H
```

5-66

```
L  L  L  H  L  L  X  X  X  L  H  H  L  L  L  H  H  H
C  H  L  H  L  L  X  X  X  L  H  H  H  L  L  H  H  H
L  L  L  H  L  L  X  X  X  L  H  H  H  L  L  H  H  H
C  H  L  H  L  L  X  X  X  L  H  L  L  H  L  H  H  H
L  L  L  H  L  L  X  X  X  L  H  L  L  H  L  H  H  H
C  H  L  H  L  L  X  X  X  L  H  L  H  H  L  H  H  H
L  L  L  H  L  L  X  X  X  L  H  L  H  H  L  H  H  H
C  H  L  H  L  L  X  X  X  L  H  L  L  L  H  H  H  H
L  L  L  H  L  L  X  X  X  L  H  L  L  L  H  H  H  L
C  H  L  H  L  L  X  X  X  L  H  L  H  L  H  H  H  L
L  L  L  H  L  L  X  X  X  L  H  L  H  L  H  H  H  L
C  H  L  H  L  L  X  X  X  L  H  H  L  H  H  H  H  H
L  L  L  H  L  L  X  X  X  L  H  H  L  H  H  H  H  H
C  H  H  H  H  H  X  X  X  L  H  H  H  L  L  L  H  Z
L  L  H  H  X  X  X  X  X  L  H  H  H  L  L  L  H  Z
C  H  H  H  X  X  X  X  X  L  H  H  L  L  L  L  H  Z
L  L  H  H  X  X  X  X  X  L  H  H  L  L  L  L  H  Z
;
;INTACK CYCLE
C  H  H  H  X  X  X  X  X  L  H  H  H  L  L  L  H  Z
L  L  H  H  X  X  X  X  X  L  H  H  H  L  L  L  H  Z
C  H  L  H  L  H  H  H  H  L  H  H  L  H  L  L  L  H
L  L  L  H  L  H  H  H  H  L  H  H  L  H  L  L  L  H
C  H  L  H  L  H  H  H  H  L  H  H  H  H  L  L  L  H
L  L  L  H  L  H  H  H  H  L  H  H  H  H  L  L  L  H
C  H  L  H  L  H  H. H  H  L  H  H  L  L  H  L  L  H
L  L  L  H  L  H  H  H  H  L  H  H  L  L  H  L  L  H
C  H  L  H  L  H  H  H  H  L  H  H  H  L  H  L  L  H
L  L  L  H  L  H  H  H  H  L  H  H  H  L  H  L  L  H
C  H  L  H  L  H  H  H  H  L  H  H  L  H  H  L  L  H
L  L  L  H  L  H  H  H  H  L  H  H  L  H  H  L  L  H
C  H  L  H  L  H  H  H  H  L  L  H  H  H  H  L  L  H
L  L  L  H  L  H  H  H  H  L  L  H  H  H  H  L  L  H
C  H  L  H  L  H  H  H  H  L  L  H  L  L  L  H  L  H
L  L  L  H  L  H  H  H  H  L  L  H  L  L  L  H  L  H
C  H  L  H  L  H  H  H  H  L  L  H  H  L  L  H  L  H
L  L  L  H  L  H  H  H  H  L  L  H  H  L  L  H  L  L
C  H  L  H  L  H  H  H  H  L  L  H  L  H  L  H  L  L
L  L  L  H  L  H  H  H  H  L  L  H  L  H  L  H  L  L
C  H  L  H  L  H  H  H  H  L  H  H  H  H  L  H  L  H
L  L  H  H  H  X  X  X  X  L  H  H  H  H  L  H  H  Z
C  H  H  H  X  X  X  X  X  L  H  H  L  L  L  L  H  Z
L  L  H  H  X  X  X  X  X  L  H  H  L  L  L  L  H  Z
--------------------------------------------------------
```

DESCRIPTION:
THE SLOWZTOM PAL IS A SELF-CONTAINED 8500 TO 68000 INTERFACE. THERE IS
NO USER SOFTWARE REQUIRED FOR THIS INTERFACE, BUT THERE IS A TRADE-OFF
OF SLOWER ACCESS TIME. AGAIN, THE INTERFACE PROVIDES TOTAL SIGNAL
COMPATIBILITY.

```
PAL16R4                                      PAL DESIGN SPECIFICATION
PAT051                                       MARK YOUNG    1/21/83
SLOW AM8500/9500 TO MOTOROLA 68000 PAL
ADVANCED MICRO DEVICES
*D9724
*F0*
L0000 1111 1011 1111 1111 1111 1111 1111 1111 *
L0032 1001 1101 0101 1101 1110 1110 1111 1111 *
L0064 1110 1111 0111 1111 1111 1111 1111 1110 *
L0096 1110 1111 0111 1111 1111 1111 1110 1111 *
L0128 1011 1110 0110 1101 1101 1110 1111 1111 *
L0256 1111 1111 1111 1111 1111 1111 1111 1111 *
L0288 1111 1011 0111 1010 1111 0101 0111 0111 *
L0320 1111 1011 0111 1111 1110 0101 0111 0111 *
L0352 1111 1011 0111 1111 1101 0110 0111 0111 *
L0512 1111 1111 1101 1111 1111 1111 1111 1111 *
L0768 1111 1011 0110 1001 1101 1111 1111 1111 *
L0800 1111 1011 0101 1010 1101 1111 1111 1111 *
L0832 1111 1011 0110 1001 1110 1111 1111 1111 *
L0864 1111 1011 0101 1010 1110 1101 1111 1111 *
L1024 1111 1011 0110 1110 1101 1111 1111 1111 *
L1056 1111 1011 0111 1101 1110 1111 1111 1111 *
L1088 1111 1011 0101 1110 1110 1101 1111 1111 *
L1280 1111 1011 0110 1110 1110 1111 1111 1111 *
L1312 1111 1011 0111 1111 1101 1110 1111 1111 *
L1344 1111 1011 0111 1101 1110 1110 1111 1111 *
L1536 1111 1111 1111 1111 1111 1111 1111 1111 *
L1568 1111 1001 0111 1110 1001 1110 1111 1111 *
L1600 1111 1001 0111 1101 1010 1110 1111 1111 *
L1792 1111 1111 1111 1111 1111 1111 1111 1111 *
L1824 1111 1001 0111 1110 0101 1110 1111 1111 *
L1856 1111 1001 0111 1101 0110 1110 1111 1111 *
L1888 1111 1110 0110 1110 1110 1101 1111 1111 *
L1920 1111 1110 0111 1101 1101 1110 1111 1111 *
L1952 1111 1110 0101 1110 1101 1110 1111 1111 *
C5D43*
V0001 C110XXXXX00HHHHHLHZ1 *
V0002 0010XXXXX00HHHHHLHZ1 *
V0003 C111XXXXX00HHHHHHHZ1 *
V0004 0011XXXXX00HHHHHHHZ1 *
V0005 C111XXXXX00HHHHHLHZ1 *
V0006 0011XXXXX00HHHHHLHZ1 *
V0007 C111XXXXX00HHHHHHHZ1 *
V0008 0011XXXXX00HHHHHHHZ1 *
V0009 C111XXXXX00HHHHHLHZ1 *
V0010 0011XXXXX00HHHHHLHZ1 *
V0011 C111XXXXX00HHHHHHHZ1 *
V0012 000100XXX00HHHHHHHH1 *
V0013 C10100XXX00HHHHHLHH1 *
V0014 000100XXX00HHHHHLHH1 *
V0015 C10100XXX00HHHHLHHH1 *
V0016 000100XXX00HHHHLHHH1 *
V0017 C10100XXX00HHHHLLHH1 *
V0018 000100XXX00HHHHLLHH1 *
V0019 C10100XXX00HHHLHHHH1 *
```

```
V0020 000100XXX00HHHLHHHH1 *
V0021 C10100XXX00HHHLHLHH1 *
V0022 000100XXX00HHHLHLHH1 *
V0023 C10101XXX00HHHLLHHH1 *
V0024 000100XXX00HHHLLHHH1 *
V0025 C10100XXX00HHHLLLHH1 *
V0026 000100XXX00HHHLLLHH1 *
V0027 C10100XXX00HHLHHHHH1 *
V0028 000100XXX00HHLHHHHH1 *
V0029 C10100XXX00HHLHHLHH1 *
V0030 000100XXX00HHLHHLHH1 *
V0031 C10100XXX0HLLHLHHH1 *
V0032 000100XXX0HLLHLHHH1 *
V0033 C10100XXX0HLLHLLHH1 *
V0034 000100XXX0HLLHLLHH1 *
V0035 C10100XXX0HLLLHHHH1 *
V0036 000100XXX0HLLLHHHL1 *
V0037 C10100XXX0HLLLHLHL1 *
V0038 000100XXX0HLLLHLHL1 *
V0039 C10100XXX0HHLLLHHH1 *
V0040 000100XXX0HHLLLHHH1 *
V0041 C11111XXX00HHHHHLHZ1 *
V0042 0011XXXXX00HHHHHLHZ1 *
V0043 C111XXXXX00HHHHHHHZ1 *
V0044 0011XXXXX00HHHHHHHZ1 *
V0045 C111XXXXX00HHHHHLHZ1 *
V0046 0011XXXXX00HHHHHLHZ1 *
V0047 C1010111100HHHHLHLH1 *
V0048 00010111100HHHHLHLH1 *
V0049 C1010111100HHHHLLLH1 *
V0050 00010111100HHHHLLLH1 *
V0051 C1010111100HHHLHHLH1 *
V0052 00010111100HHHLHHLH1 *
V0053 C1010111100HHHLHLLH1 *
V0054 00010111100HHHLHLLH1 *
V0055 C1010111100HHHLLHLH1 *
V0056 00010111100HHHLLHLH1 *
V0057 C1010111100LHHLLLLH1 *
V0058 00010111100LHHLLLLH1 *
V0059 C1010111100LHLHHHLH1 *
V0060 00010111100LHLHHHLH1 *
V0061 C1010111100LHLHHLLH1 *
V0062 00010111100LHLHHLLL1 *
V0063 C1010111100LHLHLHLL1 *
V0064 00010111100LHLHLHLL1 *
V0065 C1010111100HHLHLLLH1 *
V0066 00111XXXX00HHLHLLHZ1 *
V0067 C111XXXXX00HHHHHHHZ1 *
V0068 0011XXXXX00HHHHHHHZ1 *
F7FD
```

Logic Diagram for Slow Am8500 to MC68000 Using AmPAL16R4/AmPAL16R4A

CLK

DCLK

$\overline{CS}$

$\overline{RESET}$

$\overline{LDS}$

RW

FC0

FC1

FC2

$\overline{DTACK}$

$\overline{INTACK}$

$\overline{C0}$

$\overline{C1}$

$\overline{C2}$

$\overline{C3}$

$\overline{WR}$

$\overline{RD}$

$\overline{OE}$
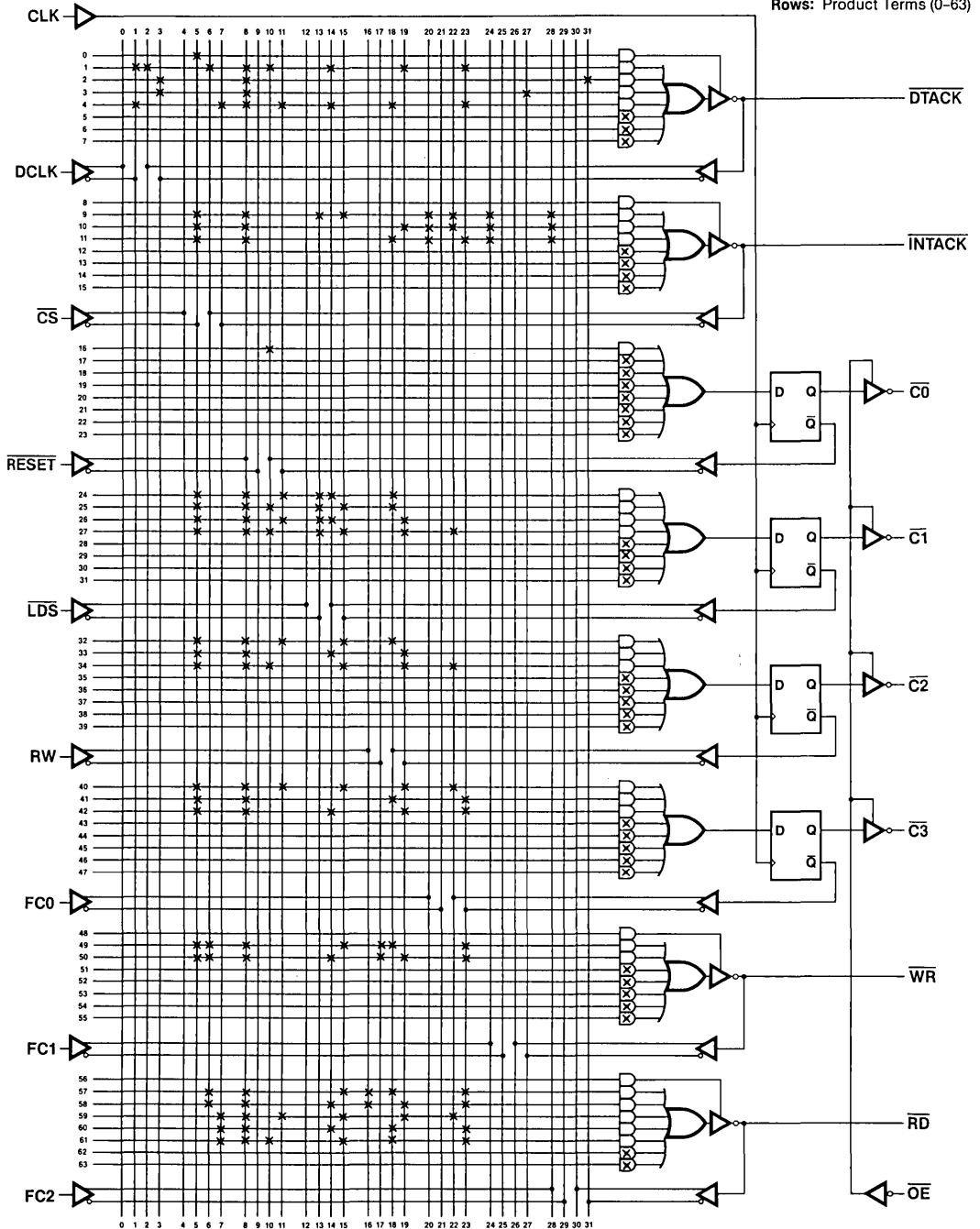
03862A-131

✳ = Fuse intact   ⊗ = All fuses intact   ┼ = Fuse blown

5-70

The Berkeley-1 Plus—
A High Performance
CPU Utilizing PALs

by Jeff Kitson and Kevin Ow-Wing
Advanced Micro Devices

INTRODUCTION

This paper illustrates the design of a high performance 16-bit CPU, called the Berkeley-1 Plus, utilizing PALs. It is intended to show where PALs fit into an overall system design, what kind of complex functions PALs can implement, and how a designer can realize the full advantages of PALs. To oversimplify, PALs fit everywhere in a design. Of the 56 chips used in this design, 34 are PALs. In the data path, they are used for data steering and data manipulation. Using PALs for data steering functions simplified the implementation of a multiple bus architecture. The data manipulation functions include a 16-bit arithmetic barrel shifter, constant generation logic and sign extension logic. PALs are also used extensively to optimize the control path. For example, control functions include instruction predecoding, double pipelining control, microprogram branch control, register file control and special instruction control.

THE BERKELEY-1 PLUS

The Berkeley-1 Plus was originally conceived of at the University of California, Berkeley as a design project for computer architecture students. The Berkeley-1 Plus is essentially a PDP-11 16-bit general purpose computer with a streamlined instruction set. The Berkeley-1 Plus was chosen as a test vehicle for PAL system design for three reasons. First, the authors had already designed the computer (in the class at Berkeley) using a limited parts list consisting mainly of fixed-function TTL SSI/MSI devices. This provided a benchmark for the new design. Second, the original computer design by the authors was done separately and the new design was a joint effort. This proved to be a true test of the flexibility of PALs because of the effect of one designer's changes on the other. Finally, the architecture was not conceived with any particular device limitations in mind. This allowed the authors to optimize the PALs to implement the architecture, instead of optimizing the PALs to fit a constrained architecture limited by available devices.

ARCHITECTURE

The architecture of the Berkeley-1 Plus, like any other computer, is defined by the instruction set and the functional blocks required to implement the instruction set. The Berkeley-1 Plus architecture is organized as a two address, register based processor to implement its simple, yet powerful instruction set. The processor interfaces to a 16-bit address bus and a 16-bit data bus. This allows the processor to address up to 64K of memory and to operate on 16-bit data and instruction words. Instruction words are orthogonal in nature allowing selection of the instruction opcode to be independent of the selection of both operand addressing

modes (for instructions with operands). The instruction set and addressing modes are shown in Appendix A. To implement the instruction set, the architecture is defined to contain an arithmetic and logic unit (ALU), an eight location register file (RF), a 16-bit program counter (PC), a 16-bit instruction register (IR), and a 4-bit processor status word (PSW). In addition, the processor also supports interrupts and illegal instruction traps.

INSTRUCTION SET

Berkeley-1 instructions have a general format that divides the instruction word into a separate 4-bit instruction opcode field (OPCODE), 6-bit source operand field (SOURCEOP) and 6-bit destination operand field (DESTOP). The general format is as follows:

```
    OPCODE          SOURCEOP          DESTOP
  15          12 11              6 5             0
 [           ][                 ][              ]
```
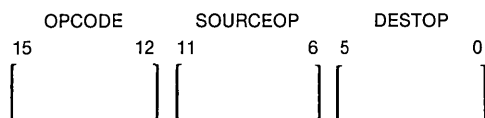
The OPCODE field encodes which of the sixteen instructions is to be executed. This is true for all Berkeley-1 instructions, even those that do not follow the general format. The SOURCEOP field encodes which of the seven addressing modes is to be used to find the source operand. The DESTOP field is the same as the SOURCEOP field except that it also defines where the reslt of an instruction is placed. For example, if the instruction is an addition, the SOURCEOP is register 1 and the DESTOP is register 2, the values inside of registers 1 and 2 would be added together and placed in register 2.

The instructions adhering to the general format include move (MOV), addition (ADD), subtraction (SUB), compare (CMP), logical and (AND), logical or (OR), exclusive-or (XOR), shift level (SHL), shift right (SHR), load multiple registers with memory (LDM), and store multiple registers to memory (STM). The unconditional jump (JMP) and unconditional jump to subroutine (CALL) instructions have the same general format except the SOURCEOP field is unused because the jump address is derived from the DESTOP field only. The unconditional return from subroutine (RET) and interrupt enable/disable (INTENB) instructions differ from the general format because they do not need any operands. The only instruction which differs from the general format significantly is the conditional branch (CBR) instruction.

The format of the CBR instruction is as follows:

```
    OPCODE        BRCOND      BROFFSET
15        12  11        8  7           0
[              ][              ][            ]
```
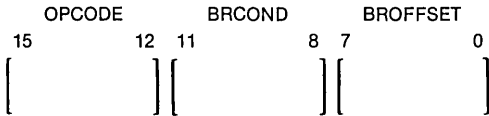
The 4-bit BRCOND field is used to select which of sixteen branch conditions are to be tested. The branch conditions are derived from the PSW. The BROFFSET field is an 8-bit signed offset that is added to the program counter, when the tested condition is true, to provide a new address. This allows position independent code, but limits the range of the CBR instruction to +127 or −128 locations relative to the program counter.

Each instruction is defined in more detail in Appendix A.

ADDRESSING MODES

As previously mentioned there are seven different addressing modes that can be used to address operands. These addressing modes are: short immediate, register, register indirect, register indexed, absolute, stack pointer auto-increment, and stack pointer auto-decrement. The short immediate addressing mode is selected when the uppermost bit of the operand field (SOURCEOP or DESTOP) is zero, leaving the lower five bits of the field as the operand. The format for short immediate is as follows:

```
    SOURCEOP VALUE           DESTOP VALUE
 5  4            0    5  4                  0
[0][          ]    [0][                ]
```
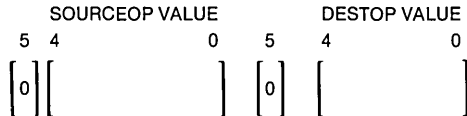
To encode the addressing modes that specify registers into the SOURCEOP and DESTOP fields, these two fields are each further divided into two smaller 3-bit fields as follows:

```
SOURCEMODE  SOURCEREG  DESTMODE  DESTREG
5        3  2        0  5       3  2      0
[        ][        ][        ][        ]
```

The SOURCEMODE and DESTMODE fields encode the addressing mode and the SOURCEREG and DESTREG fields select one of the eight registers. These modes are: register, register indirect, and register indexed. The remaining addressing modes, absolute, stack pointer auto-increment, and stack pointer auto-decrement, that do not use registers are selected by the register field when the mode field is equal to seven. Seven was chosen because the stack pointer is register seven in the register file. The format for these modes is shown below:

```
SOURCEMODE  SOURCEREG  DESTMODE  DESTREG
5        3  2        0  5       3  2      0
[1  1  1][        ][1  1  1][        ]
```
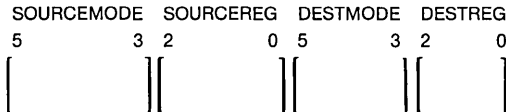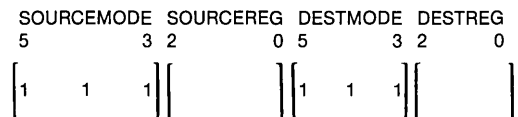
The addressing modes are shown in more detail in Appendix A.

IMPLEMENTATION

The implementation of the Berkeley-1 is obviously controlled by the architecture, but the architecture definition does not control the method of implementation. For example, the 16-bit arithmetic operations specified in the instruction set could be done serially four bits at a time in four repeatable cycles. This could result in a minimum parts count, but also in a very slow ALU. Conversely, an instruction can be executed at the same time as the next instruction is being decoded and the next one after that is being fetched. This results in a very fast machine, but parts count and the complexity of doing many things in parallel can be costly. The objective of this implementation of the Berkeley-1 Plus was to build a very high performance CPU that could fit on a single board with a very fast synchronous memory. This required a minimum parts count while implementing a complex machine operating in parallel. Thus PALs were brought to the rescue. Their ability to implement complex, custom functions at high speeds made them the ideal choice.

A block diagram of the CPU appears in Figure 1. It is characterized by multiple buses and functional blocks that have multiple input sources and multiple output destinations. The multiple buses and multiple input/output functional blocks facilitate parallel operation. The five major buses are the memory data bus (DBUS), the memory address bus (ABUS), the source operand bus (RBUS), the destination operand bus (SBUS), and the result bus (YBUS). The DBUS and ABUS are used for interface to the external world and the RBUS, SBUS and YBUS are used for internal CPU operations. The ten functional blocks are the ALU and register file, PC, IR, memory data register 1 (MDR$_1$), memory data register 2 (MDR$_2$), memory address register (MAR), barrel shifter, value register, RBUS control and SBUS control. The only other major functional block is the control sequencer.

The control sequencer block diagram appears in Figure 2. The control sequencer consists of two instruction mapping PROMs, a horizontal pipelined microprogram memory and a PAL conditional microbranch controller.

Significant performance enhancement resulted from the parallel, double-pipelined design implementation. For example, in a single cycle the Berkeley-1 can perform a barrel shift of up to 16 places while decoding the next instruction, incrementing the PC, fetching the next instruction plus one, and testing for interrupts and traps. All of this takes place in a worst case cycle time of 131ns. Effectively, due to all of these operations happening at the same time instructions are performed in a single cycle (short immediate and register mode only). This results in the Berkeley-1 Plus performing approximately 7.5 million instructions per second (7.5 MIPs). The timing and critical path analysis appears in Appendix B.

The actual design was divided up by the authors so that one was responsible for the data path and the other was responsible for the control path. The CPU portion shown in Figure 1 was designed by Jeff and the control sequencer of Figure 2 was designed by Kevin. This resulted in a more or less clean interface for interaction between the designers. Every time an optimization was contemplated in the CPU, an analysis had to take place to see if the control sequencer could support it or if the change was worth changing the sequencer, too. Likewise, every time the control sequencer needed to be changed the effects on the CPU had to be analyzed. With PALs, most of these changes were easily implemented by reprogramming a device by one designer or the other. For example, the original designs were not double-pipelined, but Jeff made changes in the data path and Kevin changed the control path to support it. Thus most of the design effort was in deciding the method of an optimized implementation (and working relationship) because of the flexibility of PALs to implement a design change easily.
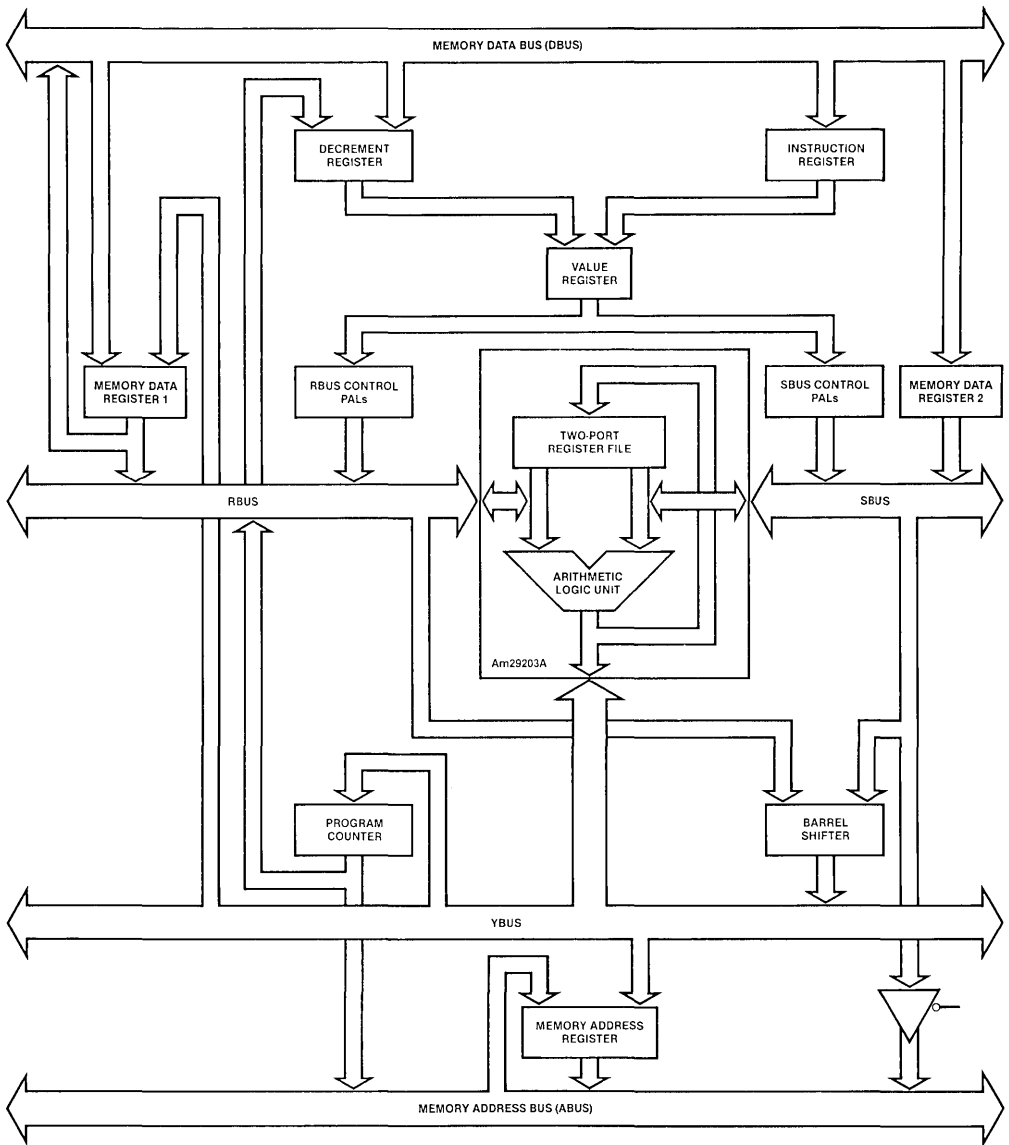
Figure 1. CPU Block Diagram

03862A-132

MEMORY DATA BUS

INSTRUCTION
REGISTER

INSTRUCTION
MAPPING
PROM 1

INSTRUCTION
MAPPING
PROM 2

EXECUTION DECODE REGISTER

MICROPROGRAM ADDRESS BUS

MICROPROGRAM PROM

PIPELINE REGISTER

| ALU AND SHIFTER CONTROL | MEMORY CONTROL | PROGRAM COUNTER CONTROL | RBUS CONTROL | SBUS CONTROL | JUMP MICROADDRESS | VECTOR MICROADDRESS |

CONDITION CODES

MICROBRANCH
CONTROL
PAL

03862A-133

Figure 2. Control Sequencer Block Diagram

A simplified state diagram of the Berkeley-1 Plus appears in Figure 3. The state diagram should allow the reader to easily understa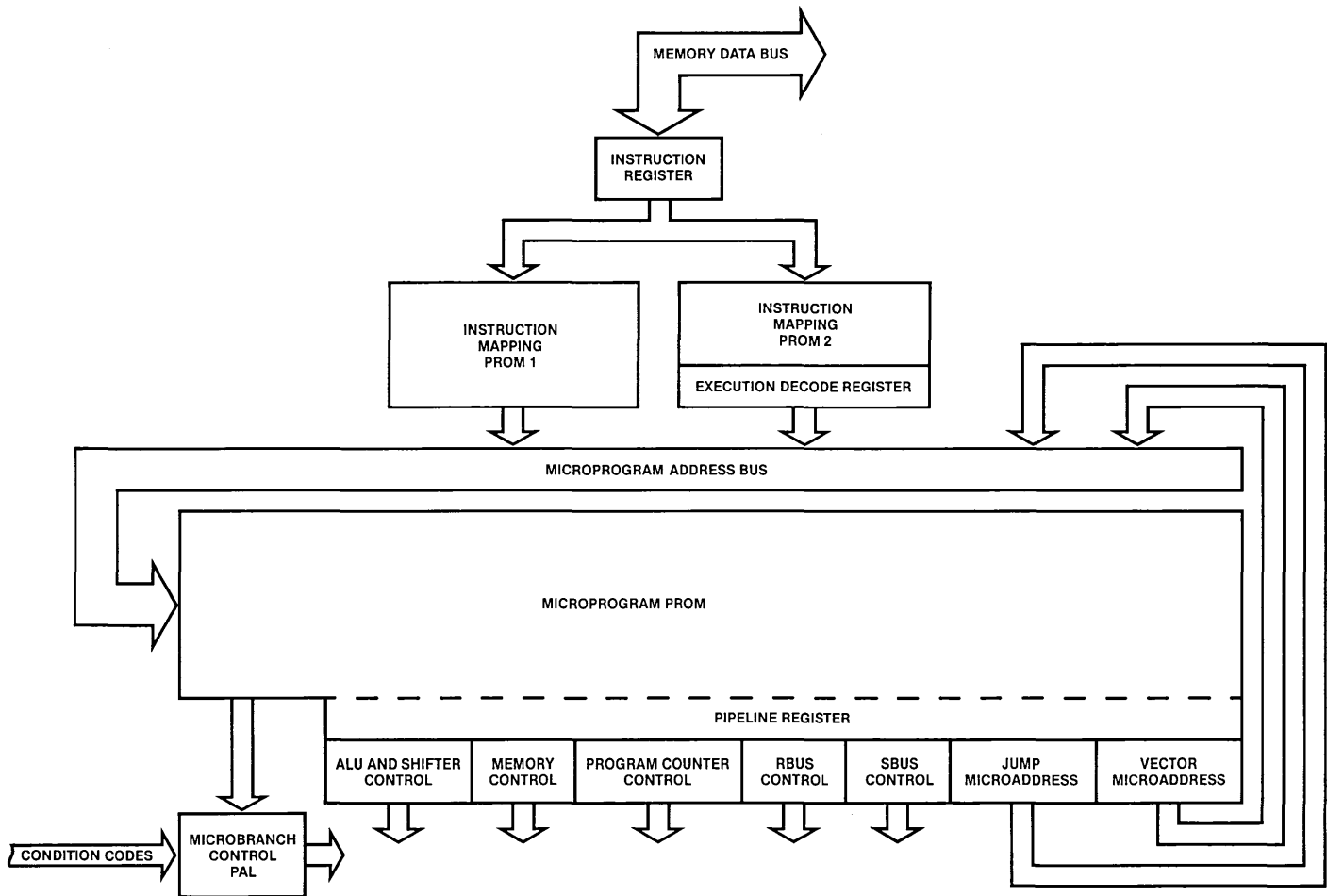nd the complexity of a double-pipelined CPU and to get a feel for how this implementation of the Berkeley-1 Plus actually performs instructions.

The various functional blocks are described in detail in the following sections. A logic diagram and PAL DESIGN SPECIFICATION for each PAL used in this design appear in Appendix C.

ALU AND REGISTER FILE

The logic diagram of the ALU and register file is shown in Figure 4. It is implemented with four 4-bit slice Am29203 bipolar microprocessors and one Am2902A high speed look-ahead carry generator. A block diagram of the Am29203 appears in Figure 5. The Am29203s were chosen because their internal three bus architecture, two-port register file, ALU, ease of control from microcode, and high speed fit ideally into the design.

The three internal buses of the Am29203 interface directly to the three internal buses of the system (see Figure 1) via I/O pins on the device. The A port of the Am29203's register file is gated onto the RBUS by the signal $\overline{OEA}$, the B port of the register file is gated onto the SBUS by $\overline{OEB}$, and the ALU output is gated onto the YBUS by $\overline{OEY}$. Each of the control signals comes directly from microcode.

The two-port register file of the Am29203 allows reading both source and destination operands from the register file and writing an operation result into the destination in a single cycle. This capability is a necessity for performing the execution cycle of an instruction in a single CPU cycle. The source and destination registers are addressed on the A and B address inputs by the value register (explained later). Writing an operation result to the destination register is controlled by the microcode signal $\overline{WR\ REG}$.

The ALU in the Am29203 is controlled directly from microcode on the instruction inputs (INST(8:0)) and by the $\overline{INSTEN}$ input. The ALU performs all arithmetic and logical operations specified in the Berkeley-1 Plus architecture except the single state barrel shift. The single state barrel shift is performed in PALs (explained later). In addition, the Am29203 has capabilities such as BCD arithmetic, multiply and divide not specified by the Berkeley-1, but which could be added to the system by simply rewriting some of the microcode.

BARREL SHIFTER

The barrel shifter is an excellent example of the trade-off between minimal implementation and performance. One easy implementation is to use the single bit shifter already present inside the Am29203. Unfortunately, a shift of fifteen bits would take fifteen CPU cycles. This is an acceptable level of performance only in machines that require a minimum amount of shift capability. Another alternative is to implement a barrel shifter which can perform a shift of any arbitrary number of bits in a single cycle. This alternative, although high in performance, can be complex to implement. An MSI solution would require the use of sixteen Am25S10 shifters. Another question is where to put the shifter; following the ALU or in parallel with the ALU. If the barrel shifter is implemented following the ALU, every operation must pass

through the ALU and the shifter resulting in a long delay. This is clearly unacceptable in a system where speed is the critical factor. In parallel, the delay is small but extra hardware is necessary to calculate the zero condition code which can no longer be calculated in the ALU. This is not a trivial task because condition codes must be calculated in parallel with the shift to maintain speed and therefore it is conditional which data bits affect the condition codes! (In MSI, this would require an additional barrel shifter for zero calculation.) In addition, hardware (two Am25LS244s) is required to gate the proper source, ALU or barrel shifter, onto the YBUS.

The Berkeley-1 uses eight AmPAL16H8As to implement the barrel shifter in parallel with the ALU for the highest performance while maintaining a reasonable parts count. In addition, the barrel shifter PALs perform condition code calculation on the data and can be gated onto the YBUS. The barrel shifter is shown in Figure 6. The implementation is in two levels; the first is the nibble shifter and the second is the bit shifter. The nibble shifter performs a shift of 0, 4, 8, or 12 bits based on the upper two bits of the four-bit shift distance (RBUS(3:2)). The nibble shifter also performs the zero condition code calculation based upon all four shift distance bits (RBUS(3:0)), the data (SBUS(15:0)), and the microcode control inputs INST(0) and SHIFTER/$\overline{ALU}$. The second level is the bit shifter which performs a right or left bit shift of 0, 1, 2, or 3 places controlled by the lower two bits of the shift distance (RBUS(1:0)) and the microcode input SHR/$\overline{SHL}$. In addition, the SHIFTER/$\overline{ALU}$ input controls the gating of the bit shifter onto the YBUS.

INSTRUCTION REGISTER

The instruction register (IR) is a good example of how PALs can be used to optimize a common function. An instruction register is generally used to receive instructions off of the data bus during memory fetch operations. The instruction is then decoded by the control sequencer and the instruction is executed. Any data in the instruction word, such as an immediate value or branch address, is operated on during the execution cycle. The Berkeley-1 instruction register performs the same function as well but in a special manner. Instruction information used during the decoding cycle is separated from information used during execution cycle. This allows the IR to predecode instructions on-the-fly from sixteen bits down to ten bits as they are received off of the data bus without losing data in the instruction word. The result is a significant increase in control sequencer speed.

The instruction register can be seen in Figure 7. The IR is implemented with two AmPAL16R8As and one Am29825 8-bit register. The microcode input CEIR when low enables instructions to be clocked into the IR. The full instruction register is 22 bits wide. This includes the 10-bit field (MAP ADR(9:0)) of predecoded instruction information and a 12-bit field (IR(11:0)) of data information.

Predecoding of the instruction from sixteen bits down to ten is accomplished by encoding the SOURCEOP and DESTOP fields down from six bits each to three bits each.

	MAP ADR(9:0)	
OPCODE	SOURCEOP	DESTOP
9 6	5 3	2 0
[ ]	[ ]	[ ]
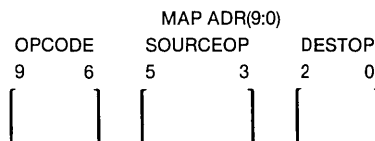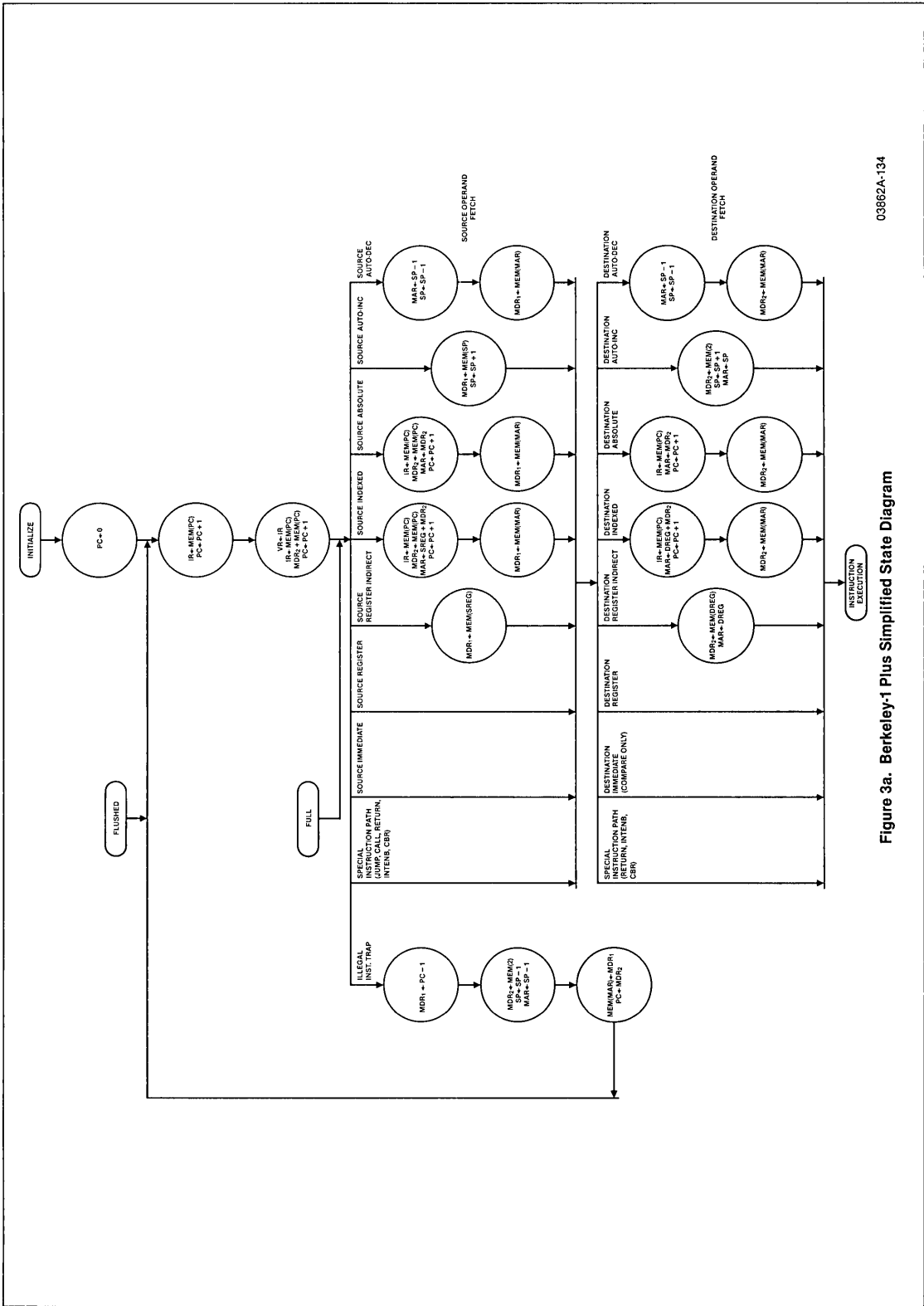
Figure 3a. Berkeley-1 Plus Simplified State Diagram
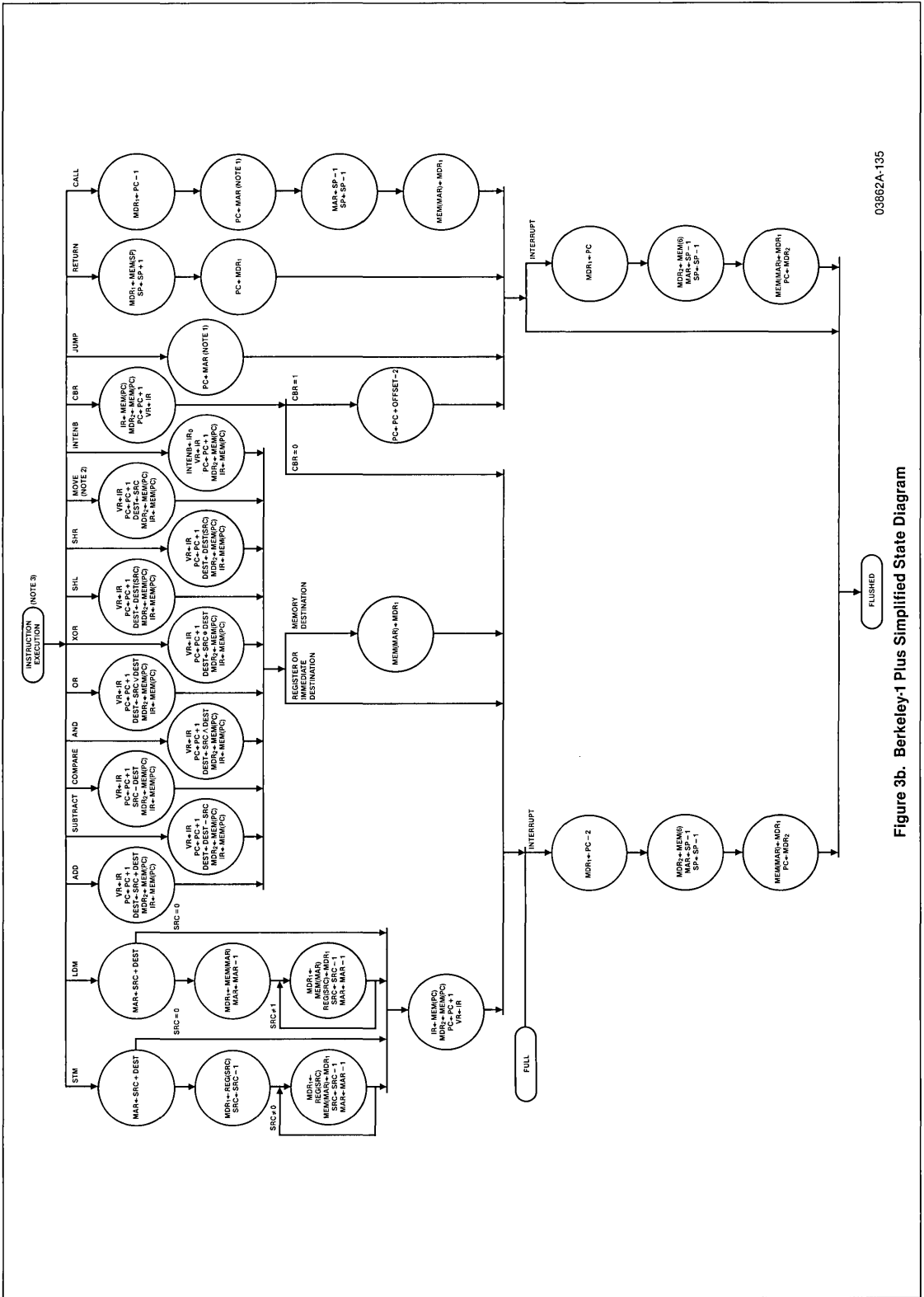
03862A-134

Figure 3b. Berkeley-1 Plus Simplified State Diagram
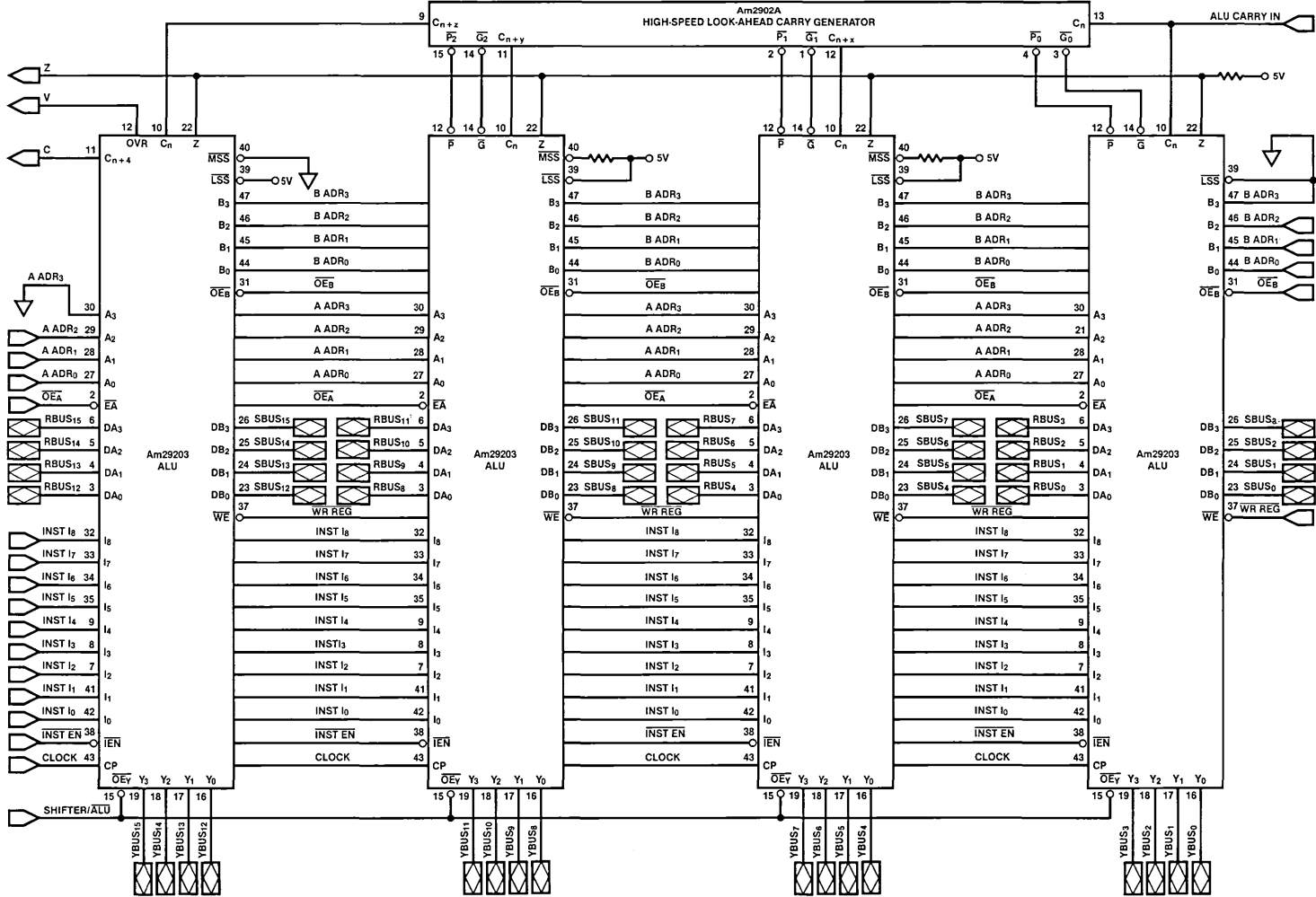
03862A-135

Figure 4. Arithmetic and Logic Unit
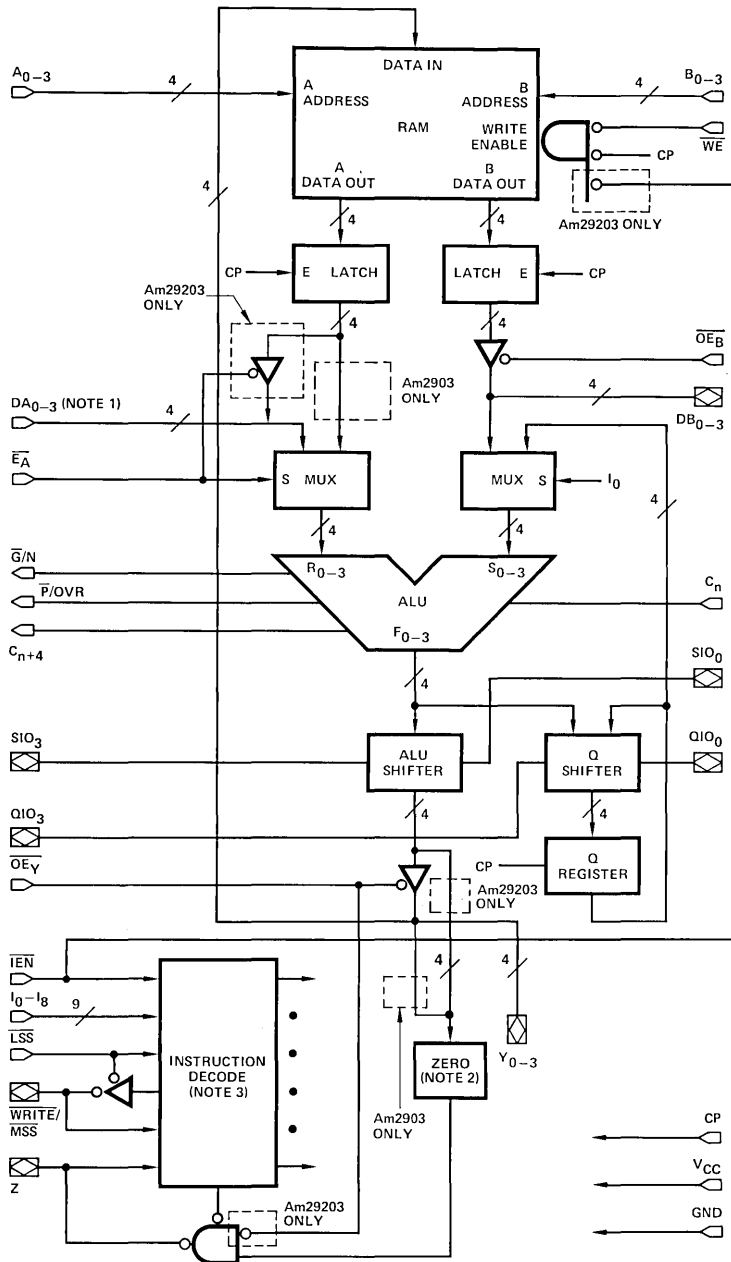
03862A-136

Notes: 1. DA_{0-3} is input only on Am2903, but is I/O port on Am29203.
2. On Am2903, zero logic is connected to Y, after the $\overline{OE}_Y$ buffer.
3. On Am2903, $\overline{IEN}$ controls $\overline{WRITE}$. On Am29203 $\overline{WRITE}$ is not affected by $\overline{IEN}$.

Figure 5. Block Diagram
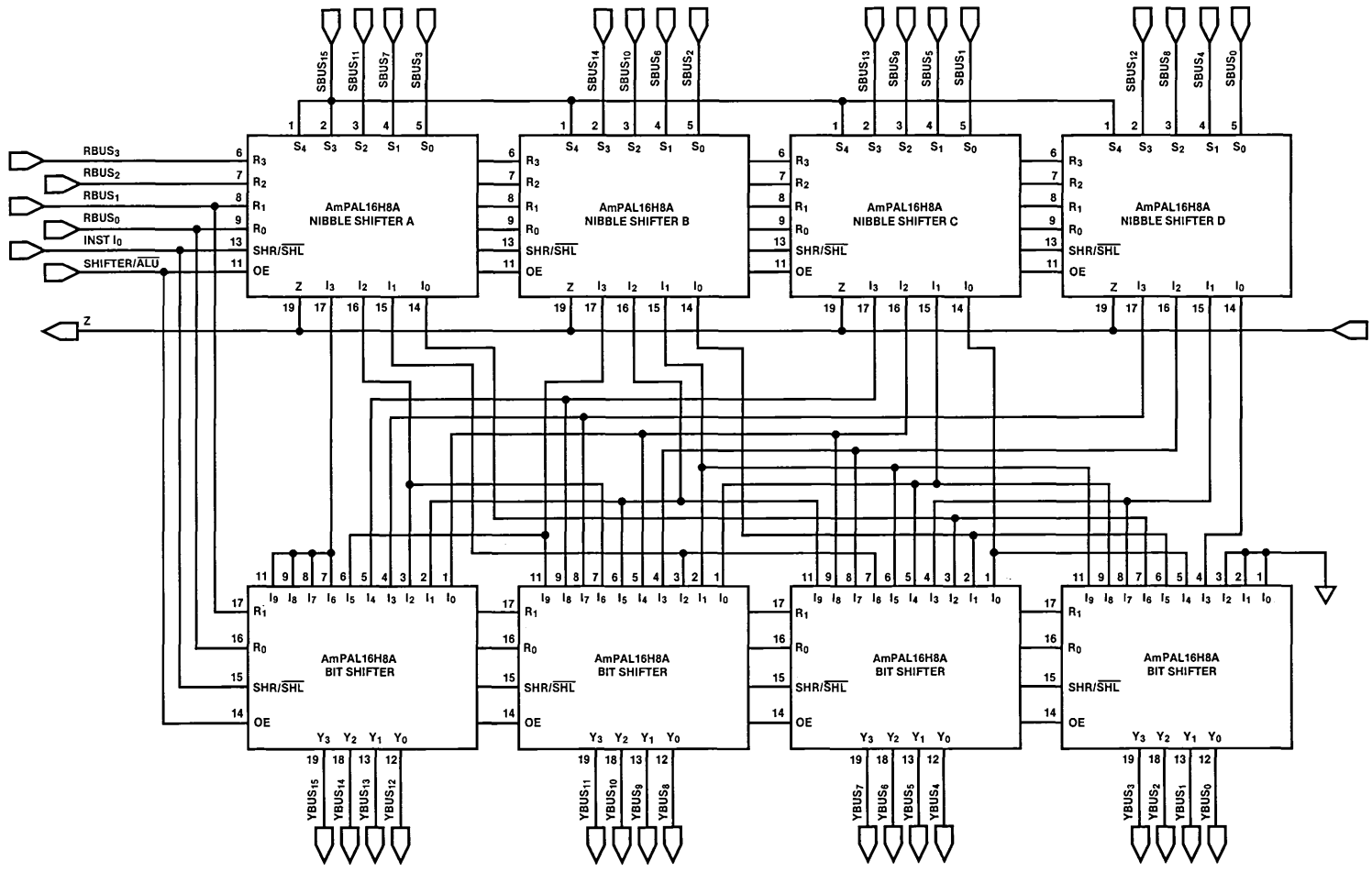
Figure 6. 16-Bit Barrel Shifter

03862A-138
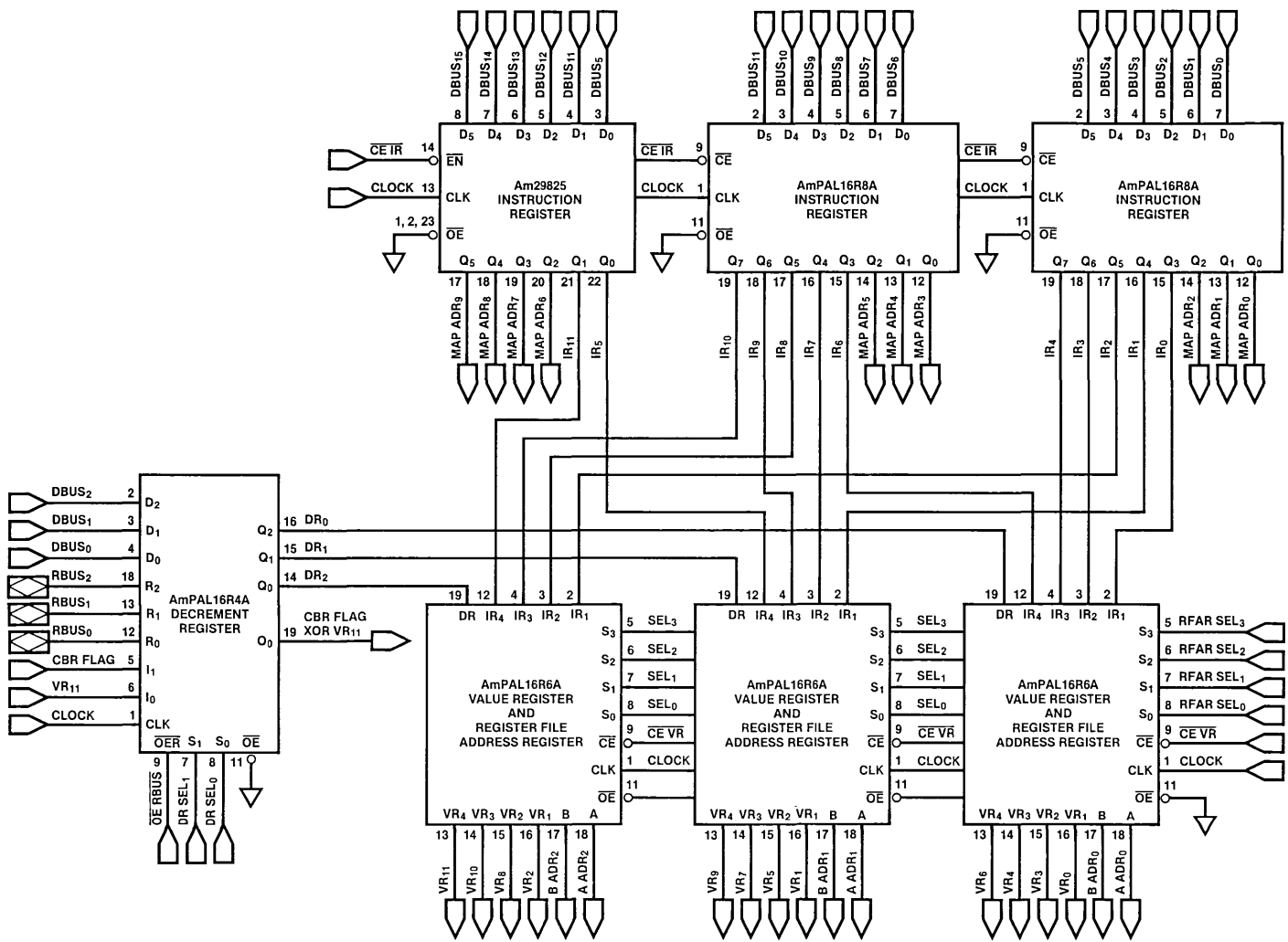
Figure 7. Instruction Register, Decrement Register and Value Register

03862A-139

Each three-bit field contains a value for one of the seven addressing modes or a value stating an illegal mode was chosen.

MODE	VALUE
Absolute	000
SP Auto-increment	001
SP Auto-decrement	010
Short Immediate	011
Register	100
Register Indirect	101
Register Indexed	110
Illegal	111

The MAP ADR(9:0) outputs are sent directly to the address inputs of the 1K mapping PROMs in the control sequencer.

The data field (IR(11:0)) is simply the lower twelve bits of the instruction word (i.e., everything but the opcode). IR(11:0) are sent directly to the value register. Notice that all 22 bits of the IR are always on and no three-state control is necessary.

VALUE REGISTER AND DECREMENT REGISTER

The value register is used to contain the instruction information, such as a conditional branch address, immediate data values, and/or register file addresses, necessary for the execution cycle. The value register is required to support the double-pipelined architecture implemented. For example, during a typical double-pipelined cycle where instruction n is being fetched from memory and instruction n − 1 is in the IR and being decoded, instruction n − 2 is being executed. Without the value register the information necessary to execute instruction n − 2 would be lost. This register is the prinicipal additional hardware required to implement double-pipelining. The value register, along with the decrement register, also controls all of the addressing required for the register file.

The value register is implemented using three identically programmed AmPAL16R6As and the decrement register is implemented using a single AmPAL16R4A. Both can be seen in Figure 7. The microcode input CEVR when low enables data to be clocked into the value register. The value register provides data information on the VR(11:0) outputs, the A address for the register file on the A ADR(2:0) outputs, and the B address on the B ADR(2:0) outputs. VR(11:0) is simply IR(11:0) delayed by one clock cycle.

Register A and B address selection is controlled by the RFAR SEL(3:0) inputs from microcode. The address selection combinations on the A address include the SOURCEREG(2:0) field (obtained from the IR or VR) and a HOLD mode. The address selection combinations on the B address include the DESTREG(2:0) field, SOURCEREG(2:0) field, decrement register, and a HOLD mode. Address selection is obtained from the IR on instructions that are to be executed in the following cycle and from the VR on instructions where multiple operand fetch and instruction execute cycles are necessary. The decrement register is used as an address source for the special LDM and STM instructions (see Appendix A) which require the loading and storing of multiple registers in sequence.

The decrement register is loaded with the address of the first register to be operated on and simply provides sequential addresses of the next registers to be operated on. The decrement register is loaded from the RBUS(2:0) when the register is selected via an immediate or register addressing mode or from the DBUS(2:0) when a memory addressing mode is selected. Loading and decrementing are controlled by the microcode inputs DR SEL(1:0). Additionally, the decrement register is gated onto the RBUS(2:0) when the microcode input OERBUS is low to provide the lower three bits so that they can be added as an offset to the memory address where data is to be stored or loaded.

RBUS CONTROL PALs

The RBUS control PAL logic consists of two AmPAL16H8As and is shown in Figure 8. Their primary function is to perform sign extension for short immediate source operands and to put constant values onto the RBUS. SOURCEOP(4) selects the sign of the short immediate operand. The function is controlled by the microcode signal RBUS SEL. Additionally, they are used to gate zeroes onto the upper thirteen bits of the RBUS when the decrement register is on the lower three bits. This is controlled by the microcode signal OERBUS(15:3). The signal OERBUS(2:0) gates the lower three bits of the PALs onto the RBUS during all other functions.

CBR AND SBUS CONTROL PALs

The CBR and SBUS control PAL logic is also shown in Figure 8. The primary function of these PALs is to perform all control necessary for execution of the conditional branch instruction (CBR), sign extension for short immediate destination operands (allowed for compare instruction (CMP) only), and constant generation. The CBR control logic is implemented in one AmPAL16R6A. The CBR logic derives a CBR flag dependent upon the true version of all eight branch conditions and which test is being selected. The branch conditions are generated from the four condition codes: zero (Z), minus (N), overflow (V), and carry (C). The condition being tested is selected by VR(10:8). The Z and N condition codes are loaded in the CBR PAL by the LOADZN microcode signal and the C and V condition codes are loaded by the LOADCV signal. The CBR flag is sent directly to the SBUS control PALs.

The SBUS control PALs are implemented using two AmPAL16H8As. The function performed by these PALs is controlled by microcode inputs SBUS SEL(1:0) and gating of data onto the SBUS is controlled by the OESBUS input. When a CBR instruction is selected, a sign extended branch offset relative to the PC is derived from VR(9:0) and gated onto the SBUS dependent upon the CBR flag and the true/false test polarity select (VR(11)). When high, the test polarity select initiates a branch when the CBR flag is also high (true), and when low it initiates a branch when the CBR flag is also low (false). Should the branch condition not be met the value − 1 is added to the PC preventing a branch.

PROGRAM COUNTER

The program counter is an excellent example of how PALs are used in this design to implement data steering. The PC is shown in Figure 9. It is implemented using four identically programmed AmPAL16R4As and one AmPAL16H8A. The PC is basically implemented as an incrementing register that can be parallel loaded from the YBUS. The inherent problem with the PC is that it must source both the ABUS for memory fetch operations and the RBUS for relative branch calculations. Unfortunately, a typical MSI device cannot drive two separate buses because they are not designed with multiple three-state outputs. This can lead to the need to add separate three-state controls for each bus (four AM25LS244s) and unnecessary added delay in the memory path. The PC PALs are designed to provide multiple outputs thus saving ICs and delay time in the critical memory path.
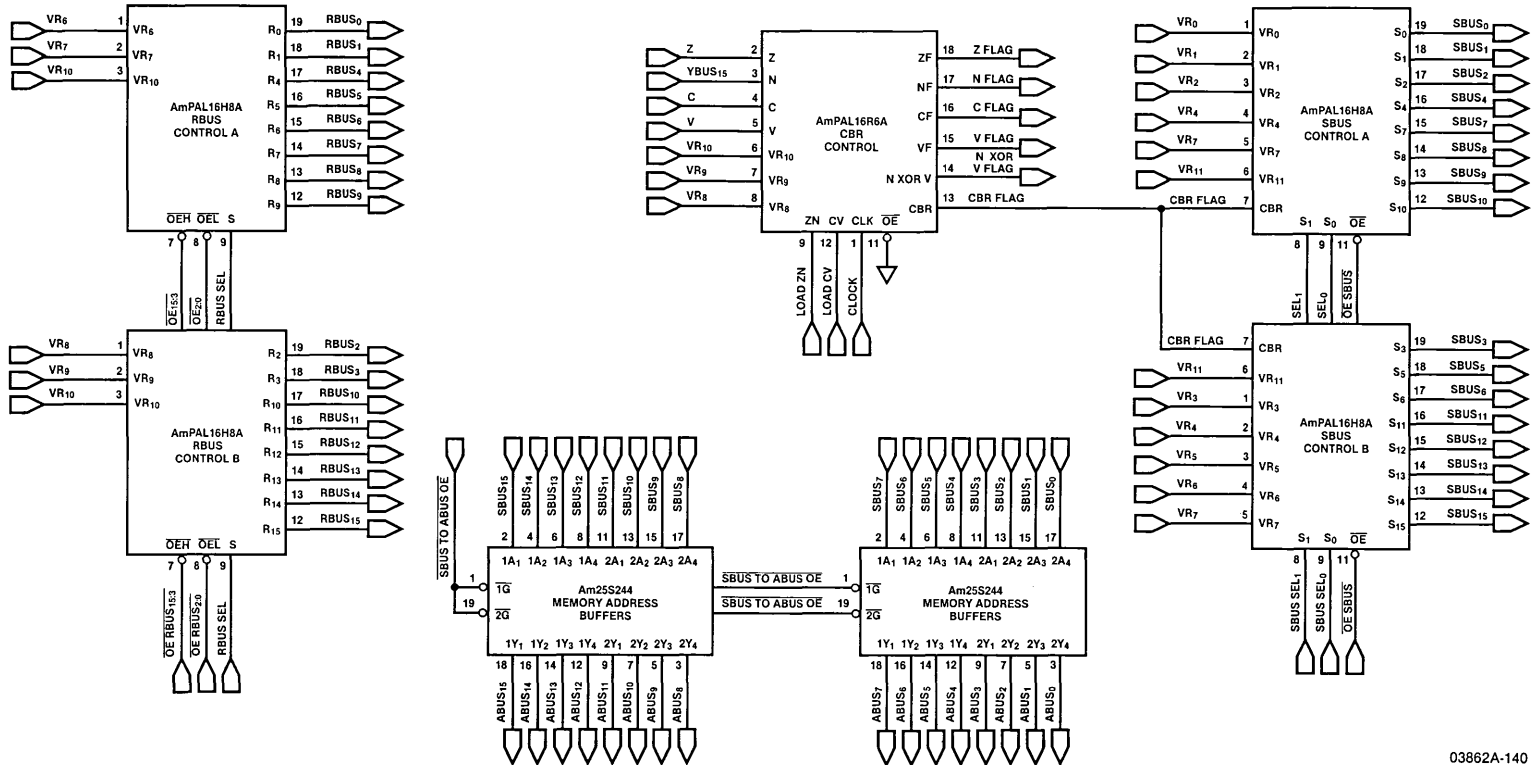
Figure 8. RBUS, SBUS and CBR Control PALs and Memory Address Buffers
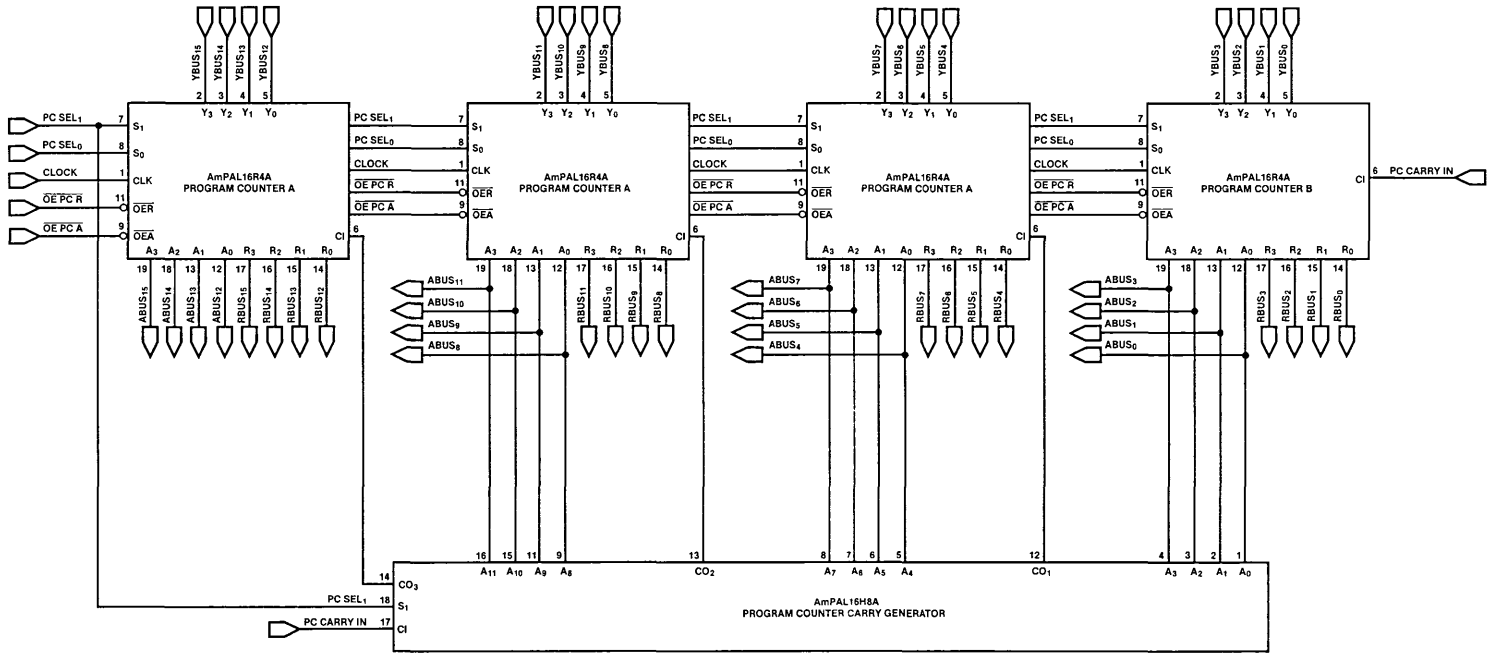
03862A-140

Figure 9. Program Counter

03862A-141

Microcode inputs PC SEL(1:0) select the function to be performed by the PC. The $\overline{OEPCR}$ and $\overline{OEPCA}$ inputs control whether the PC is to drive the RBUS or the ABUS, respectively. The PC CARRY IN signal is used to force incrementation of the PC. Additionally, the PC internally generates the vector addresses necessary for initialization, illegal instruction traps and interrupts.

MEMORY ADDRESS REGISTER

The memory address register (MAR) is primarily used for fetching of memory source and destination operands, and storing operation results in the memory destination. The MAR is implemented with four AmPAL16R4As and is shown in Figure 10. Memory address register functions are controlled by the MAR SEL(1:0) inputs. The MAR is gated onto the ABUS when the microcode input $\overline{OEMAR}$ is low. Note that the MAR also has the capability to decrement, which is necessary to support the special LDM and STM instructions.

MEMORY DATA REGISTER 1

Memory data register 1 (MDR$_1$) is another excellent example of PALs facilitating data steering functions in the Berkeley-1. MDR$_1$ is used for I/O operations between the CPU and memory requiring multiple input and output paths. It is implemented with four AmPAL16R4As and can also be seen in Figure 10. MDR$_1$ is loaded from the DBUS when the LOAD SEL input is high and the $\overline{CE\ MDR_1}$ input is low (enabled), and is loaded from the YBUS when LOAD SEL is low and $\overline{CE\ MDR_1}$ is enabled. Additionally, MDR$_1$ is gated onto the RBUS by $\overline{OE\ MDR_1\ R}$ and onto the DBUS by $\overline{OE\ MDR_1\ D}$.

MEMORY DATA REGISTER 2

Memory data register 2 (MDR$_2$) provides an additional data path from memory into the CPU. It is implemented with two Am29823 high speed 8-bit registers. It is loaded by the $\overline{CE\ MDR_2}$ input and gated onto the SBUS by the $\overline{OE\ MDR_2}$ input. MDR$_2$ is primarily used to fetch memory destination operands.

THE CONTROL SEQUENCER

The control of the CPU is performed by an exceptionally fast (105ns worst case cycle time), pipelined, microprogrammed sequencer that is comprised of only 17 chips. The operation of the sequencer is very straightforward. After initialization (PC←0), the first instruction is fetched. The instruction is decoded and the microprogram branches to a specific sequence of states to fetch the operands or begin execution (for instructions that don't require operands). The starting microprogram PROM address of this sequence of states is given by the Instruction Mapping PROMs. The pipeline register is then successively loaded with the microinstruction corresponding to each state in the sequence and the instruction is "executed." When the execution of the current instruction is complete, the next instruction is fetched (if not already done during the execution of the previous instruction), decoded, and executed. This cycle repeats for every instruction.

The sequencer and its operation will now be separated into three sections and examined in greater detail.

INSTRUCTION DECODE

The Instruction Register (IR) is composed of three chips; two AmPAL16R8As and one Am29825. The PALs are used to decode the source and destination addressing modes of the instruction prior to actually being loaded into the IR. The main advantages of this "predecode" are:

1) the number of address bits needed for the Instruction Mapping PROMs is reduced, thus reducing PROM space required.
2) sequencer cycle time is improved.

The 6-bit source and destination fields are both reduced to 3-bit fields, which is all that is necessary since there are only seven different addressing modes. These two 3-bit fields, along with the four opcode bits, form the 10-bit address to the Instruction Mapping PROMs, IMAP$_1$ and IMAP$_2$. Both IMAP$_1$ and IMAP$_2$ use these ten bits to determine the microprogram PROM address needed to begin operand fetches and instruction executions. IMAP$_1$ is used to generate the initial microprogram PROM addresses for all operand fetches and for the execution of instructions that don't require operands (e.g., CALL). IMAP$_2$ generates the initial microprogram PROM addresses for the execution of instructions that require operand fetches. Addresses generated by IMAP$_2$ are loaded into an Am29825, which is an 8-bit register with a clock enable and three-statable outputs. This step is necessary because the IR is often overwritten with the next instruction (instruction prefetching) and the address from IMAP$_2$ is not used until the operands have been fetched and execution is to begin. This is not required of IMAP$_1$ because the addresses it generates are always used immediately.

NEXT MICROPROGRAM ADDRESS SELECTION

There are four sources from which the next address for the microprogram PROM can come. These are IMAP$_1$, IMAP$_2$ and two microprogram PROMs. As previously mentioned, IMAP$_1$ contains the addresses needed to begin operand fetch cycles or instruction execution cycles, and IMAP$_2$ contains initial microprogram addresses for instruction execution cycles only. Both of the microprogram PROMs contain next state addresses for these cycles to complete.

One AmPAL16R6A is used to select which one of the four sources will provide the next microprogram PROM address. Inputs to this microbranch control PAL consist of three bits from microcode, the outputs of the decrement counter (3 bits), and two bits for interrupt control. There are four outputs, which control the three-state drivers of the four address sources.

The three bits from microcode are used to select the conditions which must be considered for microbranching (e.g., unconditional, DECREMENT COUNTER = 0?, INTERRUPT? and INTERRUPTS ENABLED?). The PAL decodes these conditions and selects the proper source for the next state address. The PAL's function table is shown in Appendix C along with the other PAL DESIGN SPECIFICATIONS.

MICROPROGRAM MEMORY

The microcode for the Berkeley-1 Plus resides in eight Am27S25 512 x 8 registered PROMs and one Am27S29 512 x 8 PROM. The latter PROM is used in conjunction with a AmPAL16R6A to generate the next microprogram address as explained above. The entire sequencer is shown in Figure 11.

CONCLUSION

The PAL implementation of the Berkeley-1 Plus results in a truly high performance 16-bit CPU. Functions such as the barrel shifter and double-pipelining control are not feasibly implemented in standard SSI/MSI devices. Specialized functions such as LDM and STM are application dependent and therefore not easily optimized in LSI implementations either. The flexibility of PALs makes these functions easy to optimize, resulting in the high performance design shown.
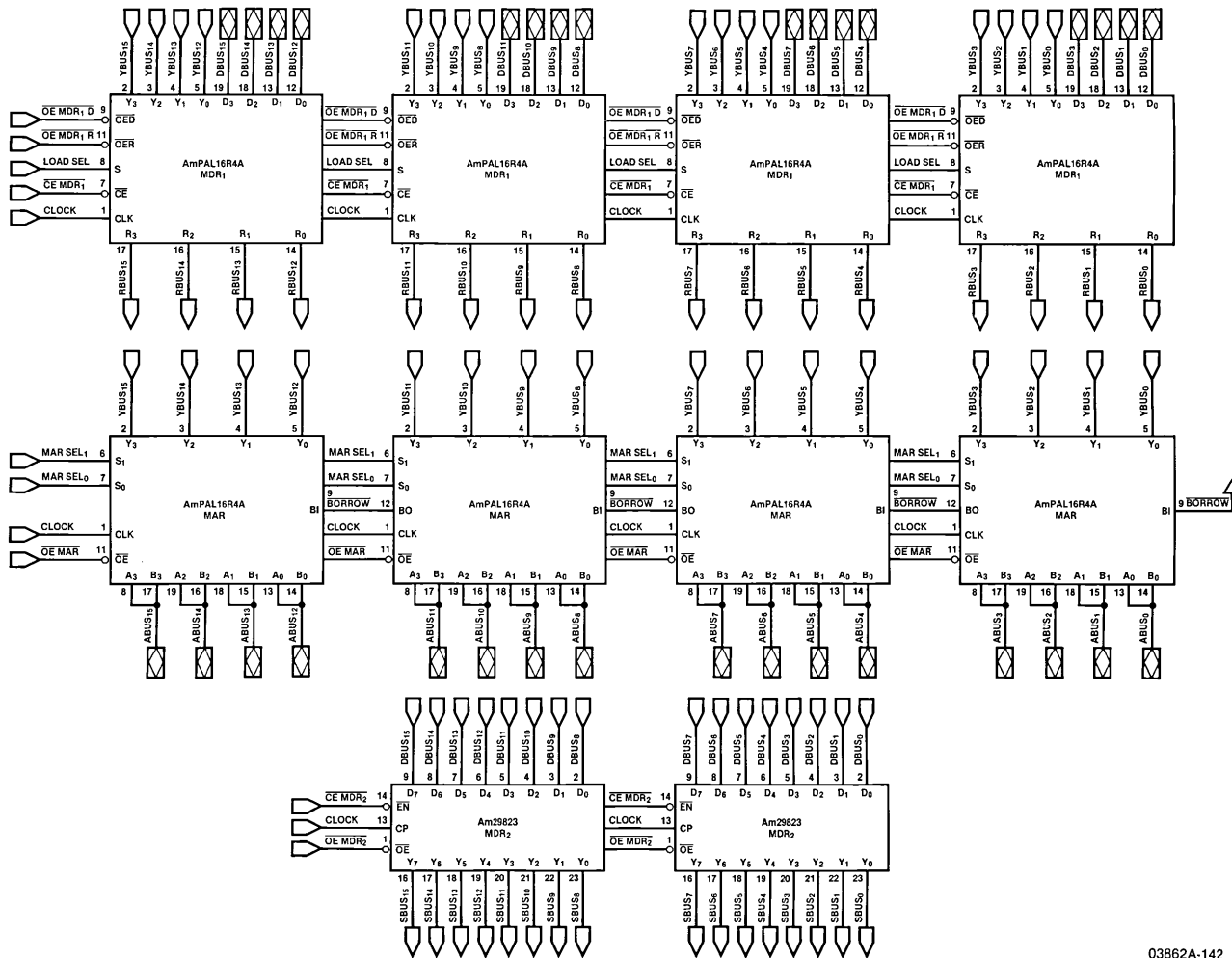
Figure 10. Memory Data Registers 1 and 2 and Memory Address Register

03862A-142

Figure 11. Control Sequencer

03862A-143

APPENDIX A
Instruction Set and Addressing Modes

INSTRUCTION SET

Mnemonic/ Instruction	OPCODE	Operation	Condition Codes
ADD Add	$00SSDD_8$	$(dst \leftarrow (src) + (dst))$	N: set if result < 0 Z: set if result $= 0$ V: set if there is arithmetic overflow as a result of the operation; that is, both operands were of the same sign and the result is of the opposite sign C: set if there is a carry from the most significant bit of the result
SUB Subtract	$01SSDD_8$	$(dst) \leftarrow (dst) - (src)$	N: set if result < 0 Z: set if result $= 0$ V: set if there is arithmetic overflow as a result of the operation, i.e., if the operands were of opposite signs and the sign of the source is the same as the sign of the result C: set if there is a borrow into the most significant bit of the result
CMP Compare	$02SSDD_8$	$(src) - (dst)$	N: set if result < 0 Z: set if result $= 0$ V: set if there is arithmetic overflow; i.e., operands of opposite signs and the sign of the destination is the same as the sign of the result C: set if there is a borrow into the most significant bit of the result
AND And	$03SSDD_8$	$(dst) \leftarrow (src) \land (dst)$	N: set if result < 0 Z: set if result $= 0$ V: unaffected C: unaffected
OR Or	$04SSDD_8$	$(dst) \leftarrow (src) \lor (dst)$	N: set if result < 0 Z: set if result $= 0$ V: unaffected C: unaffected
XOR Exclusive OR	$05SSDD_8$	$(dst) \leftarrow [(\sim src) \land (dst)] \lor$ $[(src) \land (\sim dst)]$ $= src \veebar dst$	N: set if the result < 0 Z: set if result $= 0$ V: unaffected C: unaffected
SHL Shift Left	$06SSDD_8$	$(dst_i) \leftarrow (dst_{i-src}) : i \geq src$ $(dst_i) \leftarrow 0 : i < src$	N: set if result < 0 Z: set if result $= 0$ V: unaffected C: unaffected
SHR Shift Right	$07SSDD_8$	$(dst_i) \leftarrow (dst_{i+src}) : i \leq 15 - src$ $(dst_i) \leftarrow (dst_{15}) : i > 15 - src$	N: set if result < 0 Z: set if result $= 0$ V: unaffected C: unaffected

CONDITIONAL BRANCH INSTRUCTIONS:

Mnemonic/ Instruction	OPCODE	Operation	Condition Codes
BA Branch (Unconditional)	080000_8 plus 8-bit OFFSET	$PC \leftarrow PC + OFFSET$	N: unaffected Z: unaffected V: unaffected C: unaffected
BEQ Branch if equal (to zero)	080400_8 plus 8-bit OFFSET	$PC \leftarrow PC + OFFSET$ if $Z = 1$	N: unaffected Z: unaffected V: unaffected C: unaffected
BMI Branch if minus	081000 plus 8-bit OFFSET	$PC \leftarrow PC + OFFSET$ if $N = 1$	N: unaffected Z: unaffected V: unaffected C: unaffected

INSTRUCTION SET (Continued)

Mnemonic/ Instruction	OPCode	Operation	Condition Codes
BCS Branch if carry set	08140_8 plus 8-bit OFFSET	$PC \leftarrow PC + OFFSET$ if $C = 1$	N: unaffected Z: unaffected V: unaffected C: unaffected
BVS Branch if V bit set	08200_8 plus 8-bit OFFSET	$PC \leftarrow PC + OFFSET$ if $V = 1$	N: unaffected Z: unaffected V: unaffected C: unaffected
BLT Branch if less than	08240_8 plus 8-bit OFFSET	$PC \leftarrow PC + OFFSET$ if $N \veebar V = 1$	N: unaffected Z: unaffected V: unaffected C: unaffected
BLE Branch if less than or equal to	08300_8 plus 8-bit OFFSET	$PC \leftarrow PC + OFFSET$ if $(Z) \vee (N \veebar V) = 1$	N: unaffected Z: unaffected V: unaffected C: unaffected
BLOS Branch if lower or same	08340_8 plus 8-bit OFFSET	$PC \leftarrow PC + OFFSET$ if $N \vee Z = 1$	N: unaffected Z: unaffected V: unaffected C: unaffected
NOP No Operation	08400_8 plus 8-bits of Don't Care	$PC \leftarrow PC + 1$	N: unaffected Z: unaffected V: unaffected C: unaffected
BNE Branch if not equal	08440_8 plus 8-bit OFFSET	$PC \leftarrow PC + OFFSET$ if $Z = 0$	N: unaffected Z: unaffected V: unaffected C: unaffected
BHIS Branch if higher than or same	08500_8 plus 8-bit OFFSET	$PC \leftarrow PC + OFFSET$ if $N = 0$	N: unaffected Z: unaffected V: unaffected C: unaffected
BCC Branch if carry clear	08540_8 plus 8-bit OFFSET	$PC \leftarrow PC + OFFSET$ if $C = 0$	N: unaffected Z: unaffected V: unaffected C: unaffected
BVC Branch if V bit clear	08600_8 plus 8-bit OFFSET	$PC \leftarrow PC + OFFSET$ if $V = 0$	N: unaffected Z: unaffected V: unaffected C: unaffected
BGE Branch if greater than or equal	08640_8 plus 8-bit OFFSET	$PC \leftarrow PC + OFFSET$ if $N \veebar V = 0$	N: unaffected Z: unaffected V: unaffected C: unaffected
BGT Branch if greater than	08700_8 plus 8-bit OFFSET	$PC \leftarrow PC + OFFSET$ if $(Z) \vee (N \veebar V) = 0$	N: unaffected Z: unaffected V: unaffected C: unaffected
BHI Branch if higher	08740_8 plus 8-bit OFFSET	$PC \leftarrow PC + OFFSET$ if $N \vee Z = 0$	N: unaffected Z: unaffected V: unaffected C: unaffected
MOV Move	$11SSDD_8$	$(dst) \leftarrow (src)$	N: unaffected Z: unaffected V: unaffected C: unaffected

INSTRUCTION SET (Continued)

Mnemonic/ Instruction	OPCode	Operation	Condition Codes
CALL Jump to Subroutine	$12XXDD_8$	MEM[SP − 1] ← PC PC ← dst	N: unaffected Z: unaffected V: unaffected C: unaffected
RET Return from Subroutine	$13XXXX_8$	PC ← MEM[SP] SP ← SP + 1	N: unaffected Z: unaffected V: unaffected C: unaffected
LDM Load from Memory	$14SSDD_8$	FOR i: = 0 to i = src BEGIN REG(src − i) ← MEM[dst + src − i] END	N: unaffected Z: unaffected V: unaffected C: unaffected
STM Store into Memory	$15SSDD_8$	FOR i: = 0 to i = src BEGIN MEM[dst + src − i] ← REG(src − i) END	N: unaffected Z: unaffected V: unaffected C: unaffected
INTENB Enable or Disable Interrupts	$15XXX0_8$ to disable $16XXX1_8$ to enable	INT ENB FLAG ← IR_0	N: unaffected Z: unaffected V: unaffected C: unaffected
JUMP Jump	$17XXDD_8$	PC ← dst	N: unaffected Z: unaffected V: unaffected C: unaffected

ADDRESSING MODES

Binary Code (SOURCEOP or DESTOP)	Name	Function	Binary Code (SOURCEOP or DESTOP)	Name	Function
0 I I I I I	Short Immediate	Operand is contained in the instruction. It is the lower five bits of the SOURCEOP or DESTOP field, sign extended.	111000	Absolute	The contents of the memory location pointed to by the PC is the memory address of the operand. PC ← PC + 1.
100RRR	Register	The register specified by the REG field bits RRR, contains the operand.	111001	Stack Pointer Auto-Increment	The contents of the SP (register 7) is the memory address of the operand. The SP is then automatically incremented.
101RRR	Register Indirect	The register specified by the REG field bits RRR contains the memory address of the operand.	111010	Stack Pointer Auto-Decrement	The contents of the SP is decremented and is now the memory address of the operand.
110RRR	Register Indexed	The contents of the memory location pointed to by the PC is added to the contents of the register specified by the REG field bits RRR. This forms the memory address of the operand PC ← PC + 1.	111011 1111XX	Reserved	Are currently flagged as illegal.

APPENDIX B
Critical Path Analysis

CRITICAL PATH ANALYSIS SEQUENCER

PATH 1 – – – –
PATH 2 ‹‹‹‹‹
PATH 3 – – –
PATH 4 ——
PATH 5 ·············
PATH 6 –·–·–

IR
AmPAL16R8A
AND
Am29825

INSTRUCTION
MAPPING
PROM 1
(IMAP$_1$)
Am27S281A

INSTRUCTION
MAPPING
PROM 2
(IMAP$_2$)
Am27S281A

EXECUTION
DECODE
REGISTER
Am29825

μPROGRAM
PROM
Am27S25A

μPROGRAM
PROM
Am27S25A
NEXT μADR

μPROGRAM
PROM
Am27S29A

PIPELINE
REGISTER

PIPELINE
REGISTER

μBRANCH
CONTROL
AmPAL16R6A
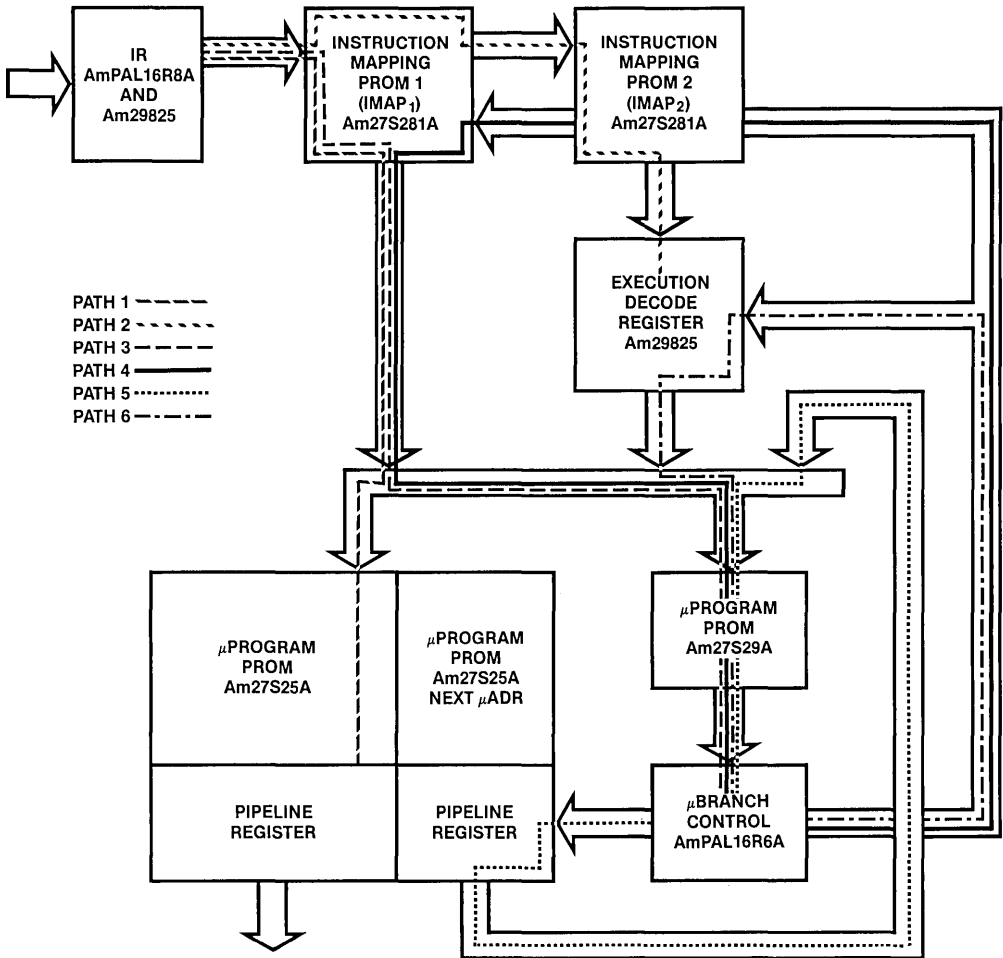
03862A-144

CRITICAL PATH TIMING ANALYSIS

CP

t_{CO}

IR
AmPAL16R8A

t_{AA}

INSTRUCTION
MAPPING PROM
Am27S281A

t_{AA}

μPROGRAM PROM
Am27S29A

t_{CO} t_S t_{CO}

μBRANCH
AmPAL16R6A

t_S t_{CO}

μPROGRAM PROM
Am27S25A

t_S t_{CO}

EXECUTION DECODE
REGISTER
Am29825

t_{PZX}

INSTRUCTION MAPPING
PROM 1
Am27S281A

t_{PZX}

μPROGRAM PROM
NEXT μADR
Am27S25A

t_{PZX}

EXECUTION DECODE
REGISTER
Am29825
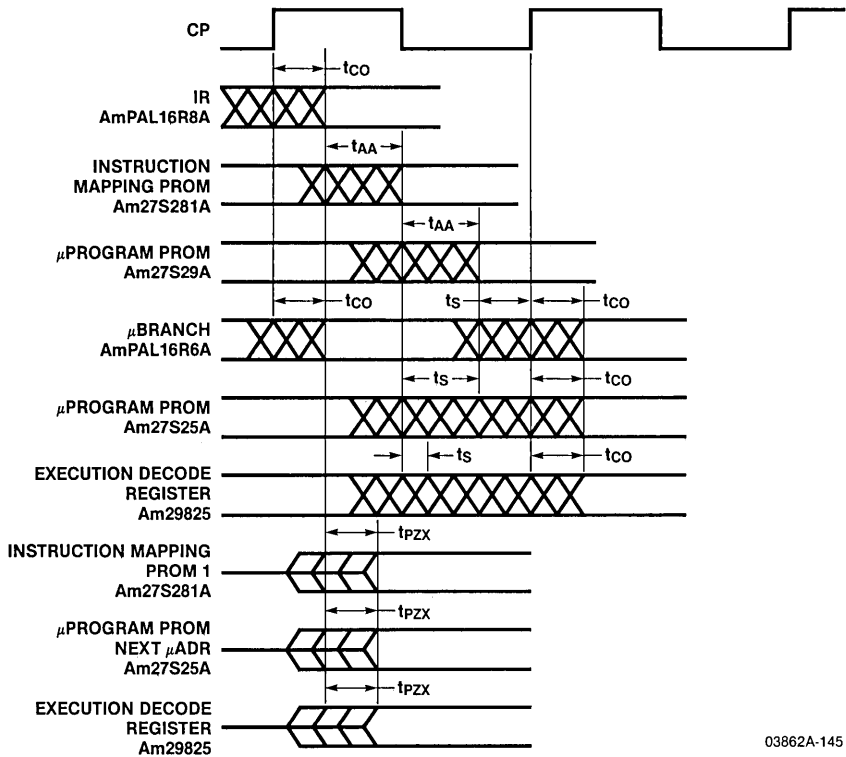
03862A-145

CRITICAL PATH ANALYSIS FOR BERKELEY-1 PLUS SEQUENCER

Path 1:

t_{CO}	IR AmPAL16R8A	15ns
t_{AA}	IMAP$_1$ Am27S281A	35ns
t_S	μProgram PROM Am27S25A	30ns
		80ns

Path 2:

t_{CO}	IR AmPAL16R8A	15ns
t_{AA}	IMAP$_2$ Am27S281A	35ns
t_S	Execution Decode REG Am29825	4ns
		54ns

Path 3:

t_{CO}	IR AmPAL16R8A	15ns
t_{AA}	IMAP$_1$ Am27S281A	35ns
t_{AA}	μProgram PROM Am27S29A	35ns
t_S	μBranch PAL AmPAL16R6A	20ns
		105ns

Path 4:

t_{CO}	μBranch PAL AmPAL16R6A	15ns
t_{PZX}	IMAP$_1$ Am27S281A	25ns
t_{AA}	μProgram PROM Am27S29A	35ns
t_S	μBranch PAL AmPAL16R6A	20ns
		95ns

Path 5:

t_{CO}	μBranch PAL AmPAL16R6A	15ns
t_{PZX}	μProgram PROM next μADDRESS Am27S25A	25ns
t_{AA}	μProgram PROM Am27S29A	35ns
t_S	μBranch PAL AmPAL16R6A	20ns
		95ns

Path 6:

t_{CO}	μBranch PAL AmPAL16R6A	15ns
t_{PZX}	Execution Decode Register Am29825	15ns
t_{AA}	μProgram PROM Am27S29A	35ns
t_S	μBranch PAL AmPAL16R6A	20ns
		85ns

Note: t_{CO} = Clock to Output Delay
t_{AA} = Access Time delay
t_S = Set-Up time
t_{PZX} = Output Enable delay

03862A-146

CRITICAL PATH ANALYSIS FOR BERKELEY-1 PLUS CPU

INSTRUCTION ADD

Path 1:

t_{CO}	VR AmPAL16R6A	15ns
ADR to $\overline{P_i}$, $\overline{G_i}$ of Am29203		52ns
$\overline{P_i}$, $\overline{G_i}$ to C_{n+z} of Am2902A		9ns
C_n to C_{n+4} of Am29203		18ns
t_S	CBR Control PAL AmPAL16R6A	20ns
		114ns

Path 2:

t_{CO}	μProgram PROM Am27S25A	25ns
I to $\overline{P_i}$, $\overline{G_i}$ of Am29203		50ns
$\overline{P_i}$, $\overline{G_i}$ to C_{n+z} of Am2902A		9ns
C_n to C_{n+4} of Am29203		18ns
t_S	CBR Control PAL AmPAL16R6A	20ns
		122ns

Path 3:

t_{CO}	VR AmPAL16R6A	15ns
ADR to Y of Am29203		68ns
t_S	Destination (PALs)	20ns
		103ns

Path 4:

t_{CO}	μProgram PROM Am27S25A	25ns
I to Y of Am29203		64ns
t_S	Destination (PALs)	20ns
		109ns

Path 5:

t_{CO}	VR AmPAL16R6A	15ns
t_{pd}	RBUS Control PALs AmPAL16H8A	25ns
DA or DB to $\overline{P_i}$, $\overline{G_i}$ of Am29203		44ns
$\overline{P_i}$, $\overline{G_i}$ to C_{n+z} of Am2902A		9ns
C_n to C_{n+4} of Am29203		18ns
t_S	CBR Control PAL AmPAL16R6A	20ns
		131ns

Path 6:

t_{CO}	VR AmPAL16R6A	15ns
t_{pd}	RBUS Control PALs AmPAL16H8A	25ns
DA or DB to Y of Am29203		59ns
t_S	Destination (PALs)	20ns
		119ns

03862A-147
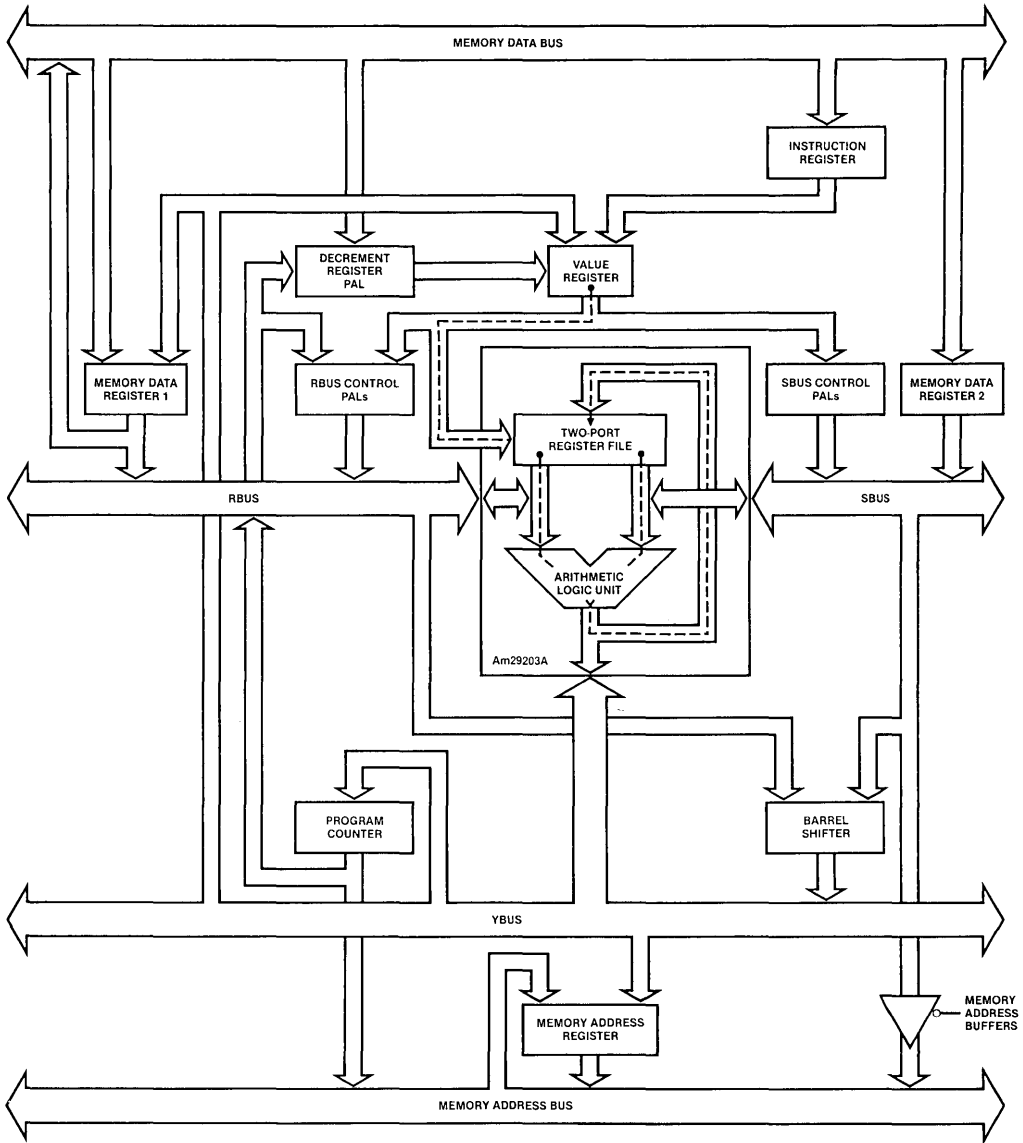
Register-to-Register ADD

03862A-148

CRITICAL PATH ANALYSIS FOR BERKELEY-1 PLUS CPU (Continued)

INSTRUCTION SHIFT:

Path 1:

t_{CO}	VR AmPAL16R6A	15ns
	A or B ADR to DA or DB of Am29203	24ns
t_{pd}	Barrel Shifter	50ns
t_s	Y to RAM of Am29203	16ns
		105ns

Path 2:

t_{CO}	VR AmPAL16R6A	15ns
t_{pd}	RBUS PALs	25ns
t_{pd}	Barrel Shifter	50ns
t_s	Y to RAM of Am29203	16ns
		106ns

Path 3:

t_{CO}	VR AmPAL16R6A	15ns
t_{pd}	RBUS PALs	25ns
t_{pd}	Barrel Shifter	50ns
t_s	MDR_1	20ns
	AmPAL16R4A	
		110ns

03862A-149

MEMORY DATA BUS

INSTRUCTION REGISTER

DECREMENT REGISTER PAL

VALUE REGISTER

MEMORY DATA REGISTER 1

RBUS CONTROL PALs

TWO-PORT REGISTER FILE

SBUS CONTROL PALs

MEMORY DATA REGISTER 2

RBUS

SBUS

ARITHMETIC LOGIC UNIT

Am29203A

PROGRAM COUNTER

BARREL SHIFTER

YBUS

MEMORY ADDRESS BUFFERS

MEMORY ADDRESS REGISTER

MEMORY ADDRESS BUS
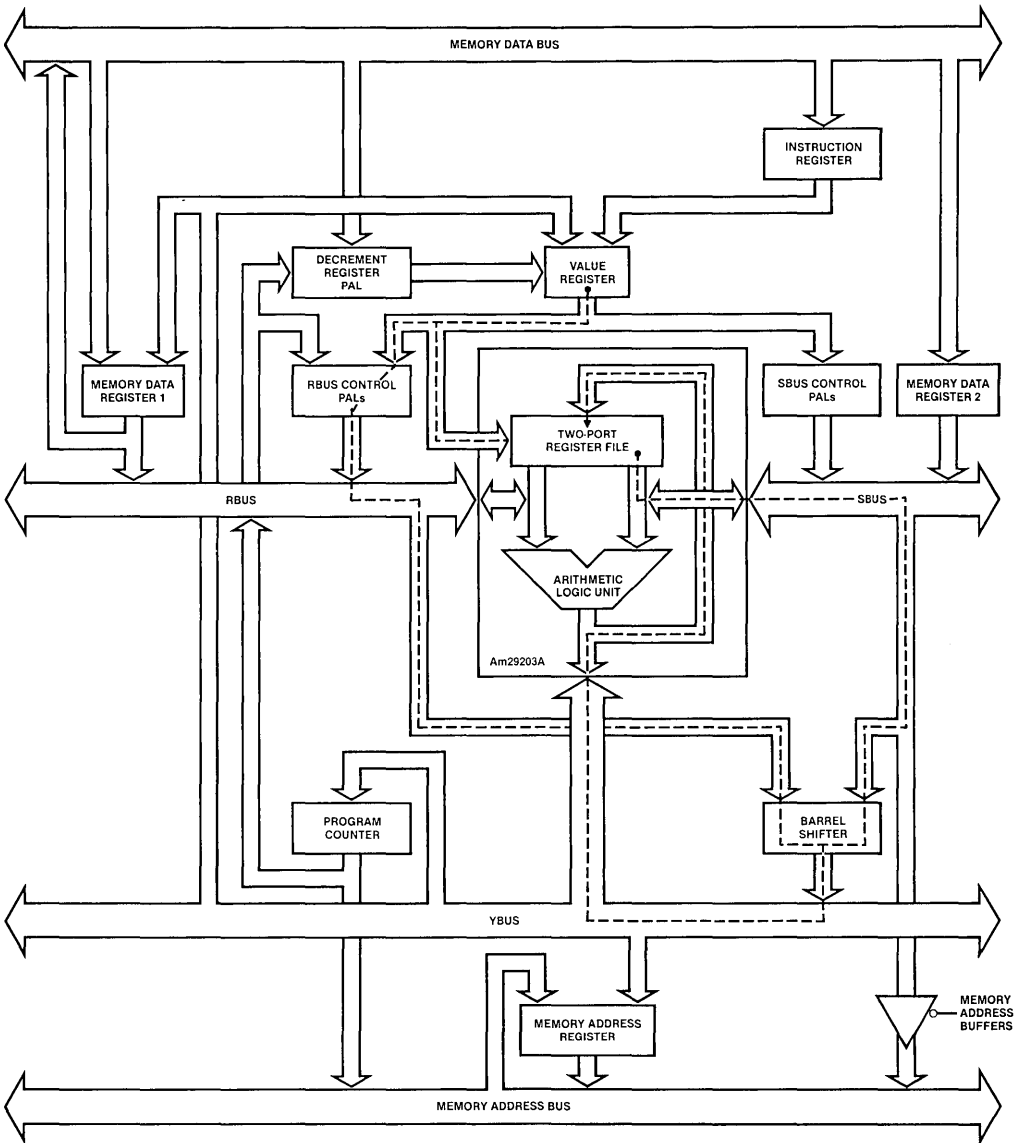
03862A-150

Immediate Source, Register Destination Shift

MEMORY ACCESS:

Indexed:

t_{CO}	VR AmPAL16R6A	15ns
ADR to Y of Am29203		68ns
t_S	MAR	20ns
	AmPAL16R4A	
		103ns

t_{CO}	μProgram PROM	25ns
	Am27S25A	
t_{PZX}	MDR_2	15ns
	Am29823	
DB to Y of Am29203		59ns
t_S	MAR	20ns
	AmPAL16R4A	
		119ns

Indirect:

t_{CO}	VR AmPAL16R6A	15ns
B ADR to DB of Am29203		24ns
t_{PD}	74S244	10.5ns
		49.5ns
	+ Memory Access Time	
	+ Set-Up Time to DEST of Data	

03862A-151

APPENDIX C
PAL Design Specifications

```
PAL16H8                                          PAL DESIGN SPECIFICATION
PAT023                                           JEFF KITSON    10/4/82
BIT SHIFTER
ADVANCED MICRO DEVICES
I0 I1 I2 I3 I4 I5 I6  I7  I8  GND
I9 Y0 Y1 OE SRNSL R0 R1 Y2 Y3 VCC

;BIT SHIFTER OUTPUT SIGNALS

IF (OE)  Y3    = /R1*/R0*I6         +
                 /R1* R0* SRNSL*I7  +
                 R1*/R0* SRNSL*I8   +
                 R1* R0* SRNSL*I9   +
                 /R1* R0*/SRNSL*I5  +
                 R1*/R0*/SRNSL*I4   +
                 R1* R0*/SRNSL*I3

IF (OE)  Y2    = /R1*/R0*I5         +
                 /R1* R0* SRNSL*I6  +
                 R1*/R0* SRNSL*I7   +
                 R1* R0* SRNSL*I8   +
                 /R1* R0*/SRNSL*I4  +
                 R1*/R0*/SRNSL*I3   +
                 R1* R0*/SRNSL*I2

IF (OE)  Y1    = /R1*/R0*I4         +
                 /R1* R0* SRNSL*I5  +
                 R1*/R0* SRNSL*I6   +
                 R1* R0* SRNSL*I7   +
                 /R1* R0*/SRNSL*I3  +
                 R1*/R0*/SRNSL*I2   +
                 R1* R0*/SRNSL*I1

IF (OE)   Y0   = /R1*/R0*I3         +
                 /R1* R0* SRNSL*I4  +
                 R1*/R0* SRNSL*I5   +
                 R1* R0* SRNSL*I6   +
                 /R1* R0*/SRNSL*I2  +
                 R1*/R0*/SRNSL*I1   +
                 R1* R0*/SRNSL*I0
```

```
FUNCTION TABLE
IO I1 I2 I3 I4 I5 I6 I7 I8 I9 OE SRNSL R1 RO      YO Y1 Y2 Y3
-----------------------------------------------------------------
;SHIFT ZERO
H  H  H  L  L  L  H  H  H  L  H   H   L  L        L  L  L  H
H  H  H  L  L  L  H  H  H  L  H   L   L  L        L  L  L  H
;SHIFT ONE
H  H  H  L  L  L  H  H  H  L  H   H   L  H        L  L  H  H
H  H  H  L  L  L  H  H  H  L  H   L   L  H        H  L  L  L
;SHIFTTWO
H  H  H  L  L  L  H  H  H  L  H   H   H  L        L  H  H  H
H  H  H  L  L  L  H  H  H  L  H   L   H  L        H  H  L  L
;SHIFT THREE
H  H  H  L  L  L  H  H  H  L  H   H   H  H        H  H  H  L
H  H  H  L  L  L  H  H  H  L  H   L   H  H        H  H  H  L
-----------------------------------------------------------------
```

DESCRIPTION

THE SECOND LEVEL OF THE BARREL SHIFTER IS THE BIT SHIFTER PERFORMS
A RIGHT OR LEFT SHIFT OF 0,1,2 OR 3 PLACES, WHICH IS CONTROLLED BY
THE LOWER TWO BITS OF THE RBUS, I.E. RBUS<1:0>. THE SHR/SHL
SIGNAL CONTROLS THE DIRECTION OF THE SHIFT. THE SHIFTER/ALU
SIGNAL CONTROLS THE GATING OF THE SHIFTER ONTO THE YBUS. THIS
FOUR INPUT AND FOUR OUTPUT PAL SLICE IS CASCADED TO IMPLEMENT A
16-BIT SHIFTER.

```
PAL16H8                                      PAL DESIGN SPECIFICATION
PAT023                                       JEFF KITSON    10/4/82
BIT SHIFTER
ADVANCED MICRO DEVICES
*D9725
*F0*
L0000 1111 1111 1111 1111 1111 1101 1111 1111 *
L0032 1111 1111 1110 1110 1111 0111 1111 1111 *
L0064 1111 1111 1110 1101 1101 1111 0111 1111 *
L0096 1111 1111 1101 1110 1101 1111 1111 0111 *
L0128 1111 1111 1101 1101 1101 1111 1111 1101 *
L0160 1111 1111 1110 1101 0110 1111 1111 1111 *
L0192 1111 1111 1101 0110 1110 1111 1111 1111 *
L0224 1111 1111 0101 1101 1110 1111 1111 1111 *
L0256 1111 1111 1111 1111 1111 1101 1111 1111 *
L0288 1111 1111 1110 1110 0111 1111 1111 1111 *
L0320 1111 1111 1110 1101 1101 0111 1111 1111 *
L0352 1111 1111 1101 1110 1101 1111 0111 1111 *
L0384 1111 1111 1101 1101 1101 1111 1111 0111 *
L0416 1111 1111 1110 0101 1110 1111 1111 1111 *
L0448 1111 1111 0101 1110 1110 1111 1111 1111 *
L0480 1111 0111 1101 1101 1110 1111 1111 1111 *
L1536 1111 1111 1111 1111 1111 1101 1111 1111 *
L1568 1111 1111 1110 0110 1111 1111 1111 1111 *
L1600 1111 1111 1110 1101 0101 1111 1111 1111 *
L1632 1111 1111 1101 1110 1101 0111 1111 1111 *
L1664 1111 1111 1101 1101 1101 1111 0111 1111 *
L1696 1111 1111 0110 1101 1110 1111 1111 1111 *
L1728 1111 0111 1101 1110 1110 1111 1111 1111 *
L1760 0111 1111 1101 1101 1110 1111 1111 1111 *
L1792 1111 1111 1111 1111 1111 1101 1111 1111 *
L1824 1111 1111 0110 1110 1111 1111 1111 1111 *
L1856 1111 1111 1110 0101 1101 1111 1111 1111 *
L1888 1111 1111 1101 1110 0101 1111 1111 1111 *
L1920 1111 1111 1101 1101 1101 0111 1111 1111 *
L1952 1111 0111 1110 1101 1110 1111 1111 1111 *
L1984 0111 1111 1101 1110 1110 1111 1111 1111 *
L2016 1101 1111 1101 1101 1110 1111 1111 1111 *
C722F*
V0001 11100011100LL1100LH1 *
V0002 11100011100LL1000LH1 *
V0003 11100011100LL1110HH1 *
V0004 11100011100HL1010LL1 *
V0005 11100011100LH1101HH1 *
V0006 11100011100HH1001LL1 *
V0007 11100011100HH1111HL1 *
V0008 11100011100HH1011HL1 *
79AF
```

LOGIC DIAGRAM FOR:
BIT SHIFTER USING AmPAL16H8

Columns: Inputs (0–31)
Rows: Product Terms (0–63)

03862A-152

─✳─ = Fuse intact   ─☒─ = All fuses intact   ─┼─ = Fuse blown

```
PAL16H8                              PAL DESIGN SPECIFICATION
PAT024                               JENNY YEE      10/4/82
NIBBLE SHIFTER A
ADVANCED MICRO DEVICES
S4 S3  S2 S1 SO  R3  R2  R1  RO GND
OE NC  SRNSL IO I1 I2 I3 ZOE Z  VCC

;NIBBLE SHIFTER A OUTPUTS

IF (OE*ZOE) Z       =  ZOE*/ZOE

          ZOE       = /SRNSL*/R1* RO* I3      +
                                         I2   +
                                         I1   +
                                         IO

          I3        = SRNSL* S2*/R3* R2        +
                       SRNSL* S1* R3* R2       +
                       SRNSL* SO* R3* R2       +
                      /SRNSL* S4*/R3* R2       +
                      /SRNSL* S4* R3*/R2       +
                      /SRNSL* S4* R3* R2       +
                               S3*/R3*/R2

          I2        = SRNSL* S1*/R3* R2        +
                       SRNSL* SO* R3*/R2       +
                      /SRNSL* S3*/R3* R2       +
                      /SRNSL* S4* R3*/R2       +
                      /SRNSL* S4* R3* R2       +
                               S2*/R3*/R2

          I1        =  SRNSL* SO*/R3* R2       +
                      /SRNSL* S2*/R3* R2       +
                      /SRNSL* S3* R3*/R2       +
                      /SRNSL* S4* R3* R2       +
                               S1*/R3*/R2

          IO        = /SRNSL* S1*/R3* R2       +
                      /SRNSL* S2* R3*/R2       +
                      /SRNSL* S3* R3* R2       +
                               SO*/R3*/R2
```

FUNCTION TABLE

S4	S3	S2	S1	S0	R3	R2	R1	R0	SRNSL	OE	I3	I2	I1	I0	ZOE	Z
;SHIFT ZERO																
L	H	H	H	L	L	L	L	L	H	H	H	H	H	L	H	L
L	H	H	H	L	L	L	L	L	L	H	H	H	H	L	H	L
;SHIFT ONE																
L	H	L	H	H	L	H	L	H	H	H	L	H	H	L	H	L
L	H	L	H	H	L	H	L	H	L	H	L	H	L	H	H	L
;SHIFT TWO																
L	H	H	L	L	H	L	H	L	H	H	L	L	L	L	L	Z
L	H	H	L	L	H	L	H	L	L	H	L	L	H	H	H	L
;SHIFT THREE																
L	H	L	H	H	H	H	H	H	H	H	H	L	L	L	L	Z
L	H	L	H	H	H	H	H	H	L	H	L	L	L	H	H	L

DESCRIPTION
THE NIBBLE SHIFTER IS IMPLEMENTED USING FOUR AMPAL16H8AS. IT PERFORMS
A SHIFT OF 0,4,8 OR 12 BITS. THE NUMBER OF SHIFTS IS DEPENDENT UPON
THE UPPER TWO BITS OF THE FOUR BIT SHIFT DISTANCE, RBUS<3:2>. ZERO
CONDITION CODE CALCULATION IS PERFORMED USING ALL FOUR SHIFT DISTANCE
BITS, RBUS<3:0>. SHIFT DIRECTION IS DETERMINED BY SHR/SHL. DATA
INPUTS ARE SBUS<15:0>. NIBBLE SHIFTER A USES INPUTS SBUS<15,11,7,3>.

```
PAL16H8                              PAL DESIGN SPECIFICATION
PAT024                               JENNY YEE      10/4/82
NIBBLE SHIFTER A
ADVANCED MICRO DEVICES
*D9725
*F0*
L0000 1111 1111 1111 1111 1111 1111 1111 1101 *
L0032 1111 1101 1111 1111 1111 1111 1111 1111 *
L0256 1111 1111 1111 1111 1111 1111 1111 1111 *
L0288 1111 1111 1101 1111 1111 1111 1010 0111 *
L0320 1111 1111 1111 1101 1111 1111 1111 1111 *
L0352 1111 1111 1111 1111 1101 1111 1111 1111 *
L0384 1111 1111 1111 1111 1111 1101 1111 1111 *
L0512 1111 1111 1111 1111 1111 1111 1111 1111 *
L0544 1111 0111 1111 1111 1011 0111 1101 1111 *
L0576 1111 1111 0111 1111 0111 0111 1101 1111 *
L0608 1111 1111 1111 0111 0111 0111 1101 1111 *
L0640 1101 1111 1111 1111 1011 0111 1110 1111 *
L0672 1101 1111 1111 1111 0111 1011 1110 1111 *
L0704 1101 1111 1111 1111 0111 0111 1110 1111 *
L0736 0111 1111 1111 1111 1011 1011 1111 1111 *
L0768 1111 1111 1111 1111 1111 1111 1111 1111 *
L0800 1111 1111 0111 1111 1011 0111 1101 1111 *
L0832 1111 1111 1111 0111 0111 1011 1101 1111 *
L0864 0111 1111 1111 1111 1011 0111 1110 1111 *
L0896 1101 1111 1111 1111 0111 1011 1110 1111 *
L0928 1101 1111 1111 1111 0111 0111 1110 1111 *
L0960 1111 0111 1111 1111 1011 1011 1111 1111 *
L1024 1111 1111 1111 1111 1111 1111 1111 1111 *
L1056 1111 1111 1111 0111 1011 0111 1101 1111 *
L1088 1111 0111 1111 1111 1011 0111 1110 1111 *
L1120 0111 1111 1111 1111 0111 1011 1110 1111 *
L1152 1101 1111 1111 1111 0111 0111 1110 1111 *
L1184 1111 1111 0111 1111 1011 1011 1111 1111 *
L1280 1111 1111 1111 1111 1111 1111 1111 1111 *
L1312 1111 1111 0111 1111 1011 0111 1110 1111 *
L1344 1111 0111 1111 1111 0111 1011 1110 1111 *
L1376 0111 1111 1111 1111 0111 0111 1110 1111 *
L1408 1111 1111 1111 0111 1011 1011 1111 1111 *
C7F31*
V0001 01110000001X1LHHHHH1 *
V0002 01110000001X0LHHHHH1 *
V0003 01011010101X1LHHLHH1 *
V0004 01011010101X0HLHLHH1 *
V0005 01100101001X1LLLLLL1 *
V0006 01100101001X0HHLLHH1 *
V0007 01011111101X1LLLLHLL1 *
V0008 01011111101X0HLLLHH1 *
8543
```
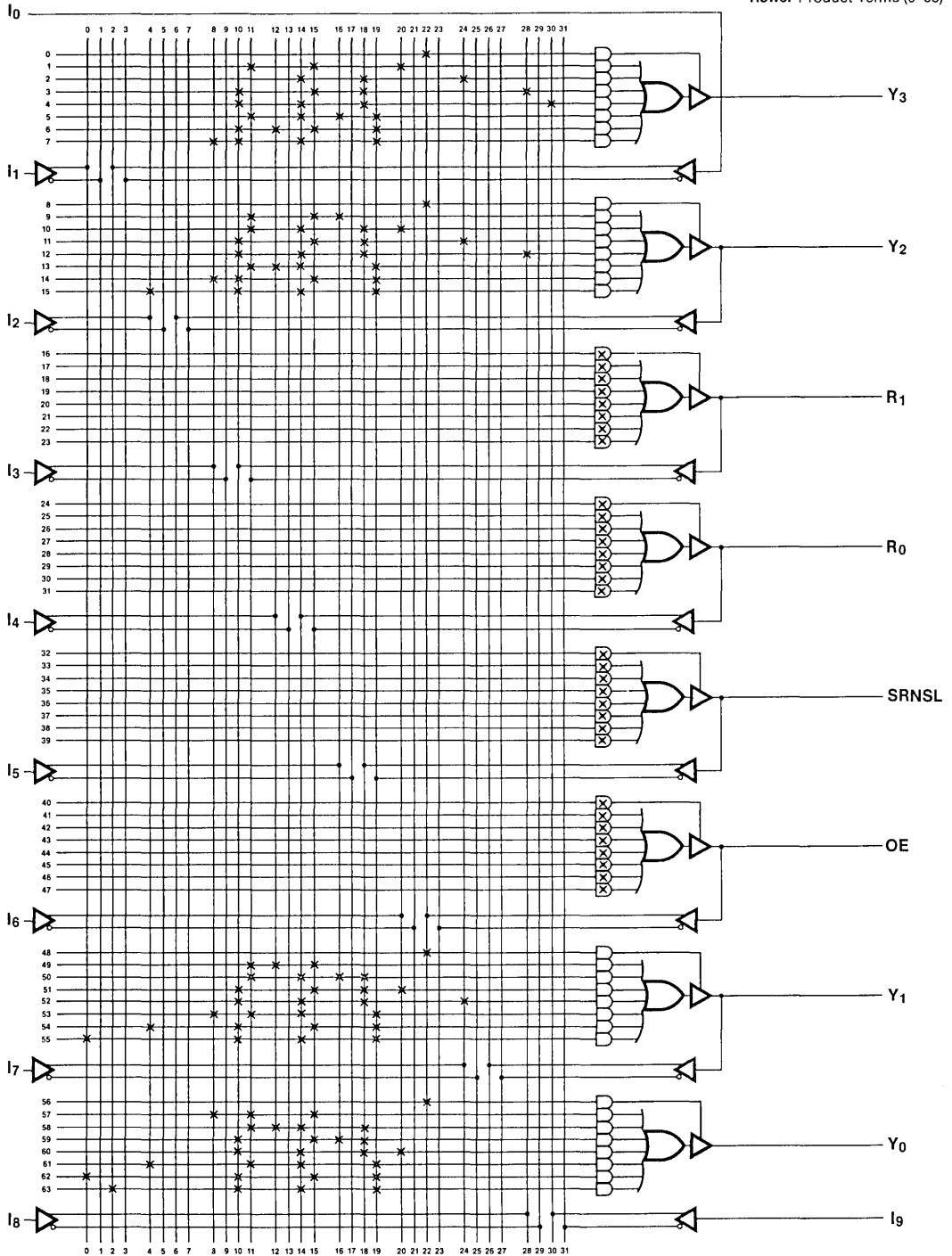
LOGIC DIAGRAM FOR:
NIBBLE SHIFTER A USING AmPAL16H8A

Columns: Inputs (0–31)
Rows: Product Terms (0–63)

—*— = Fuse intact   —⊗— = All fuses intact   —+— = Fuse blown
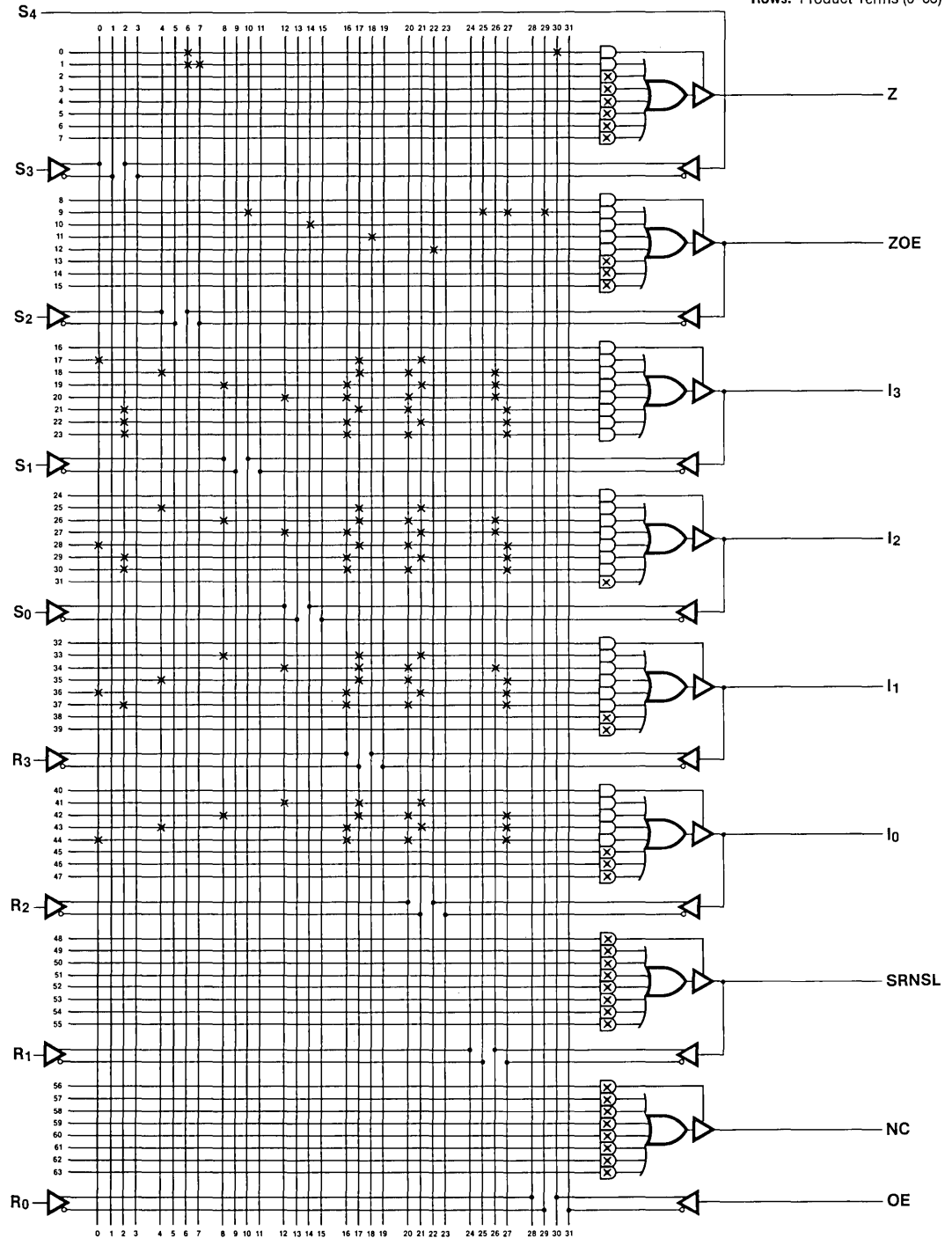
03862A-153

PAL16H8
PATO24
NIBBLE SHIFTER B
ADVANCED MICRO DEVICES
S4 S3 S2 S1 SO R3 R2 R1 RO GND
OE NC SRNSL IO I1 I2 I3 ZOE Z VCC

PAL DESIGN SPECIFICATION
JEFF KITSON   1O/4/82

; NIBBLE SHIFTER B OUTPUTS

IF (OE*ZOE) Z   = ZOE*/ZOE

         ZOE   = SRNSL*/R1* I3   +
                  SRNSL*/R1* IO   +
                  SRNSL*/RO* IO   +
                             I2   +
                             I1

         I3   = SRNSL* S2*/R3* R2   +
                  SRNSL* S1* R3*/R2   +
                  SRNSL* SO* R3* R2   +
                  /SRNSL* S4*/R3* R2   +
                  /SRNSL* S4* R3*/R2   +
                  /SRNSL* S4* R3* R2   +
                         S3*/R3*/R2

         I2   = SRNSL* S1*/R3* R2   +
                  SRNSL* SO* R3*/R2   ÷
                  /SRNSL* S3*/R3* R2   +
                  /SRNSL* S4* R3*/R2   +
                  /SRNSL* S4* R3* R2   +
                         S2*/R3*/R2

         I1   = SRNSL* SO*/R3* R2   +
                  /SRNSL* S2*/R3* R2   +
                  /SRNSL* S3* R3*/R2   +
                  /SRNSL* S4* R3* R2   +
                         S1*/R3*/R2

         IO   =/SRNSL* S1*/R3* R2   +
                  /SRNSL* S2* R3*/R2   +
                  /SRNSL* S3* R3* R2   +
                         SO*/R3*/R2

FUNCTION TABLE

S4	S3	S2	S1	S0		R3	R2	R1	R0		SRNSL	OE		I3	I2	I1	I0	ZOE	Z
;SHIFT ZERO																			
L	H	H	H	L		L	L	X	X		H	L		H	H	H	L	H	Z
L	H	H	H	L		L	L	X	X		L	L		H	H	H	L	H	Z
;SHIFT ONE																			
L	H	L	H	H		L	H	X	X		H	L		L	H	H	L	H	Z
L	H	L	H	H		L	H	X	X		L	L		L	H	L	H	H	Z
;SHIFT TWO																			
L	H	H	L	L		H	L	X	X		H	L		L	L	L	L	L	Z
L	H	H	L	L		H	L	X	X		L	L		L	L	H	H	H	Z
;SHIFT THREE																			
L	H	L	H	H		H	H	X	X		H	L		H	L	L	L	H	Z
L	H	L	H	H		H	H	X	X		L	L		L	L	L	H	L	Z

DESCRIPTION

NIBBLE SHIFTER B IS THE SECOND SLICE OF THE SERIES OF FOUR CASCADED
NIBBLE SHIFTER PALS. THE FUNCTIONS ARE THE SAME, WITH NIBBLE SHIFTER
B USING SBUS<14,10,6,2> AS DATA INPUTS.

PAL16H8   PAL DESIGN SPECIFICATION
PAT024   JEFF KITSON   10/4/82
NIBBLE SHIFTER B
ADVANCED MICRO DEVICES
*D9725
F0
L0000 1111 1111 1111 1111 1111 1111 1111 1101 *
L0032 1111 1101 1111 1111 1111 1111 1111 1111 *
L0256 1111 1111 1111 1111 1111 1111 1111 1111 *
L0288 1111 1111 1101 1111 1111 1111 1001 1111 *
L0320 1111 1111 1111 1111 1111 1101 1001 1111 *
L0352 1111 1111 1111 1111 1111 1101 1101 1011 *
L0384 1111 1111 1111 1101 1111 1111 1111 1111 *
L0416 1111 1111 1111 1111 1101 1111 1111 1111 *
L0512 1111 1111 1111 1111 1111 1111 1111 1111 *
L0544 1111 0111 1111 1111 1011 0111 1101 1111 *
L0576 1111 1111 0111 1111 0111 1011 1101 1111 *
L0608 1111 1111 1111 0111 0111 0111 1101 1111 *
L0640 1101 1111 1111 1111 1011 0111 1110 1111 *
L0672 1101 1111 1111 1111 0111 1011 1110 1111 *
L0704 1101 1111 1111 1111 0111 0111 1110 1111 *
L0736 0111 1111 1111 1111 1011 1011 1111 1111 *
L0768 1111 1111 1111 1111 1111 1111 1111 1111 *
L0800 1111 1111 0111 1111 1011 0111 1101 1111 *
L0832 1111 1111 1111 0111 0111 1011 1101 1111 *
L0864 0111 1111 1111 1111 1011 0111 1110 1111 *
L0896 1101 1111 1111 1111 0111 1011 1110 1111 *
L0928 1101 1111 1111 1111 0111 0111 1110 1111 *
L0960 1111 0111 1111 1111 1011 1011 1111 1111 *
L1024 1111 1111 1111 1111 1111 1111 1111 1111 *
L1056 1111 1111 1111 0111 1011 0111 1101 1111 *
L1088 1111 0111 1111 1111 1011 0111 1110 1111 *
L1120 0111 1111 1111 1111 0111 1011 1110 1111 *
L1152 1101 1111 1111 1111 0111 0111 1110 1111 *
L1184 1111 1111 0111 1111 1011 1011 1111 1111 *
L1280 1111 1111 1111 1111 1111 1111 1111 1111 *
L1312 1111 1111 0111 1111 1011 0111 1110 1111 *
L1344 1111 0111 1111 1111 0111 1011 1110 1111 *
L1376 0111 1111 1111 1111 0111 0111 1110 1111 *
L1408 1111 1111 1111 0111 1011 1011 1111 1111 *
C82C7*
V0001 0111000XX00X1LHHHHZ1 *
V0002 0111000XX00XOLHHHHZ1 *
V0003 0101101XX00X1LHHLHZ1 *
V0004 0101101XX00XOHLHLHZ1 *
V0005 0110010XX00X1LLLLLZ1 *
V0006 0110010XX00XOHHLLHZ1 *
V0007 0101111XX00X1LLLHHZ1 *
V0008 0101111XX00XOHLLLLZ1 *
911F

LOGIC DIAGRAM FOR:
NIBBLE SHIFTER B USING AmPAL16H8A

✗ = Fuse intact   ⊠ = All fuses intact   ╪ = Fuse blown
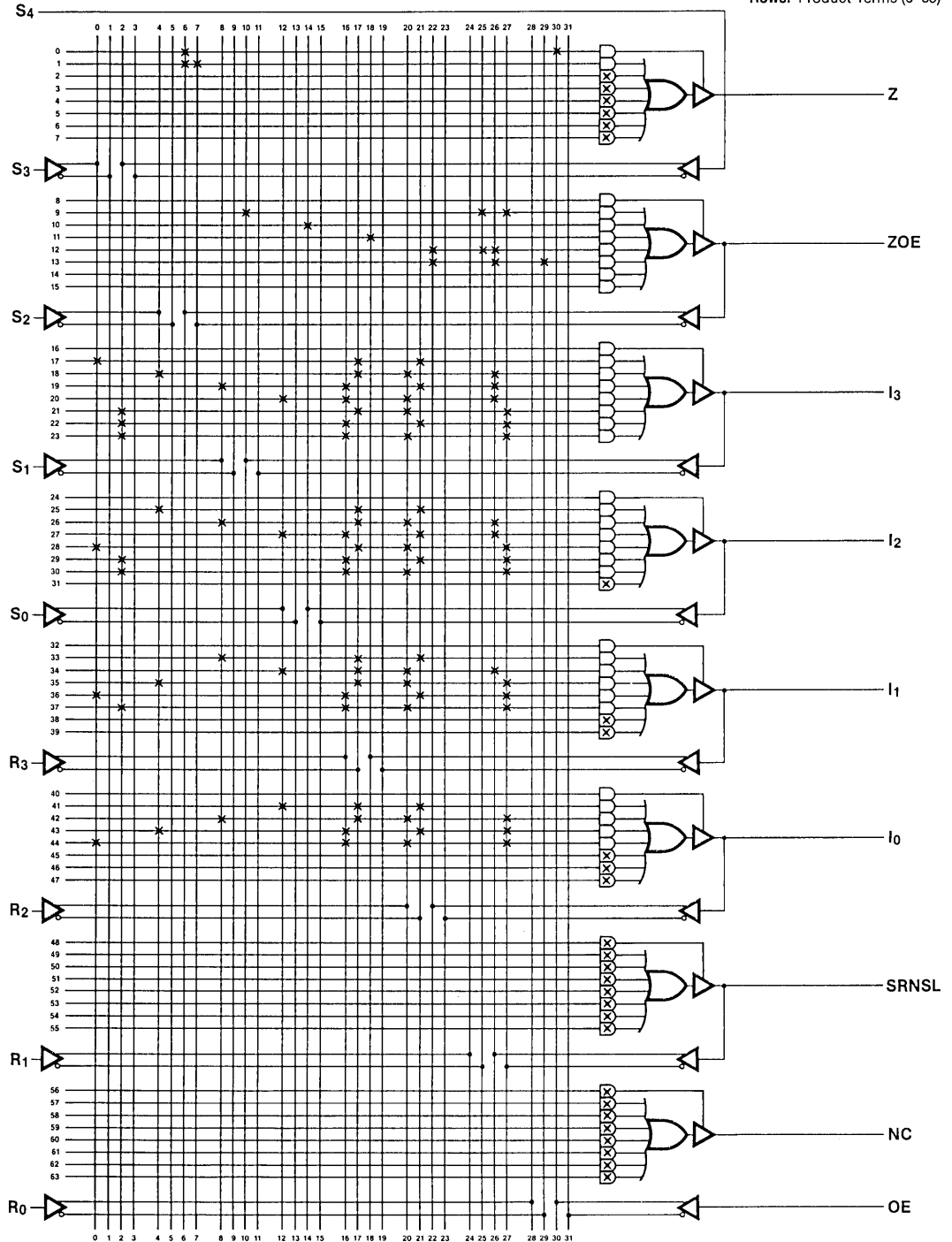
03862A-154

```
PAL16H8                                    PAL DESIGN SPECIFICATION
PAT025                                     JENNY YEE        10/4/82
NIBBLE SHIFTER C
ADVANCED MICRO DEVICES
S4 S3 S2  S1 S0  R3 R2  R1 R0 GND
OE NC SRNSL I0 I1 I2 I3 ZOE Z VCC

;NIBBLE SHIFTER C OUTPUTS

IF (OE*ZOE) Z   = ZOE*/ZOE

           ZOE  = /SRNSL*/R1* I3          +
                  /SRNSL*/R0* I3          +
                   SRNSL*/R1* I0          +
                             I2           +
                             I1

           I3   = SRNSL* S2*/R3* R2       +
                  SRNSL* S1* R3*/R2       +
                  SRNSL* S0* R3* R2       +
                  /SRNSL* S4*/R3* R2      +
                  /SRNSL* S4* R3*/R2      +
                  /SRNSL* S4* R3* R2      +
                         S3*/R3*/R2

           I2   = SRNSL* S1*/R3* R2       +
                  SRNSL* S0* R3*/R2       +
                  /SRNSL* S3*/R3* R2      +
                  /SRNSL* S4* R3*/R2      +
                  /SRNSL* S4* R3* R2      +
                         S2*/R3*/R2

           I1   = SRNSL* S0*/R3* R2       +
                  /SRNSL* S2*/R3*/R2      +
                  /SRNSL* S3* R3*/R2      +
                  /SRNSL* S4* R3* R2      +
                         S1*/R3*/R2

           I0   =/SRNSL* S1*/R3* R2       +
                  /SRNSL* S2* R3*/R2      +
                  /SRNSL* S3* R3* R2      +
                         S0*/R3*/R2
```

```
FUNCTION TABLE
S4 S3 S2 S1 S0      R3 R2 R1 R0    SRNSL OE    I3 I2 I1 I0 ZOE Z
----------------------------------------------------------------
;SHIFT ZERO
L  H  H  H  L       L  L  L  L     H    H      H  H  H  L  H   L
L  H  H  H  L       L  L  L  L     L    H      H  H  H  L  H   L
;SHIFT ONE
L  H  L  H  H       L  H  L  H     H    H      L  H  H  L  H   L
L  H  L  H  H       L  H  L  H     L    H      L  H  L  H  H   L
;SHIFT TWO
L  H  H  L  L       H  L  H  L     H    H      L  L  L  L  L   Z
L  H  H  L  L       H  L  H  L     L    H      L  L  H  H  H   L
;SHIFT THREE
L  H  L  H  H       H  H  H  H     H    H      H  L  L  L  L   Z
L  H  L  H  H       H  H  H  H     L    H      L  L  L  H  L   Z
----------------------------------------------------------------

DESCRIPTION
NIBBLE SHIFTER C IS THE THIRD SLICE OF THE FOUR CASCADED NIBBLE
SHIFTER PALS.  AGAIN, THE FUCNTIONS ARE THE SAME, WITH THIS
SLICE HAVING DATA INPUTS FROM SBUS <13,9,5,1>.
```

```
PAL16H8                                    PAL DESIGN SPECIFICATION
PAT025                                     JENNY YEE        10/4/82
NIBBLE SHIFTER C
ADVANCED MICRO DEVICES
*D9725
*F0*
L0000 1111 1111 1111 1111 1111 1111 1111 1101 *
L0032 1111 1101 1111 1111 1111 1111 1111 1111 *
L0256 1111 1111 1111 1111 1111 1111 1111 1111 *
L0288 1111 1111 1101 1111 1111 1111 1010 1111 *
L0320 1111 1111 1101 1111 1111 1111 1110 1011 *
L0352 1111 1111 1111 1111 1111 1101 1001 1111 *
L0384 1111 1111 1111 1101 1111 1111 1111 1111 *
L0416 1111 1111 1111 1111 1101 1111 1111 1111 *
L0512 1111 1111 1111 1111 1111 1111 1111 1111 *
L0544 1111 0111 1111 1111 1011 0111 1101 1111 *
L0576 1111 1111 0111 1111 0111 1011 1101 1111 *
L0608 1111 1111 1111 0111 0111 0111 1101 1111 *
L0640 1101 1111 1111 1111 1011 0111 1110 1111 *
L0672 1101 1111 1111 1111 0111 1011 1110 1111 *
L0704 1101 1111 1111 1111 0111 0111 1110 1111 *
L0736 0111 1111 1111 1111 1011 1011 1111 1111 *
L0768 1111 1111 1111 1111 1111 1111 1111 1111 *
L0800 1111 1111 0111 1111 1011 0111 1101 1111 *
L0832 1111 1111 1111 0111 0111 1011 1101 1111 *
L0864 0111 1111 1111 1111 1011 0111 1110 1111 *
L0896 1101 1111 1111 1111 0111 1011 1110 1111 *
L0928 1101 1111 1111 1111 0111 0111 1110 1111 *
L0960 1111 0111 1111 1111 1011 1011 1111 1111 *
L1024 1111 1111 1111 1111 1111 1111 1111 1111 *
L1056 1111 1111 1111 0111 1011 0111 1101 1111 *
L1088 1111 0111 1111 1111 1011 1011 1110 1111 *
L1120 0111 1111 1111 1111 0111 1011 1110 1111 *
L1152 1101 1111 1111 1111 0111 0111 1110 1111 *
L1184 1111 1111 0111 1111 1011 1011 1111 1111 *
L1280 1111 1111 1111 1111 1111 1111 1111 1111 *
L1312 1111 1111 0111 1111 1011 0111 1110 1111 *
L1344 1111 0111 1111 1111 0111 1011 1110 1111 *
L1376 0111 1111 1111 1111 0111 0111 1110 1111 *
L1408 1111 1111 1111 0111 1011 1011 1111 1111 *
C82EB*
V0001 01110000001X1LHHHHH1 *
V0002 01110000001XOLHHHHH1 *
V0003 01011010101X1LHHLHH1 *
V0004 01011010101XOHLHLHH1 *
V0005 01100101001X1LLLLLL1 *
V0006 01100101001XOHHLLHH1 *
V0007 01011111101X1LLLHLL1 *
V0008 01011111101XOHLLLLL1 *
8DF2
```

LOGIC DIAGRAM FOR:
NIBBLE SHIFTER C USING AmPAL16H8A

Columns: Inputs (0–31)
Rows: Product Terms (0–63)

= Fuse intact   = All fuses intact   = Fuse blown

03862A-155

5-117

```
PAL16H8                              PAL DESIGN SPECIFICATION
PAT026                               JEFF KITSON     10/4/82
NIBBLE SHIFTER D
ADVANCED MICRO DEVICES
S4 S3  S2  S1 SO R3 R2 R1 RO GND
OE NC SRNSL IO I1 I2 I3 ZOE Z VCC

;NIBBLE SHIFTER D OUTPUTS

IF (OE*ZOE)  Z    = ZOE*/ZOE

        ZOE  = I3                              +
               I2                              +
               I1                              +
               SRNSL* IO*/R1*/RO

       I3  =        S3*/R3*/R2        +
              SRNSL* S2*/R3* R2        +
              SRNSL* S1* R3* R2        +
              SRNSL* SO* R3* R2        +
             /SRNSL* S4*/R3* R2        +
             /SRNSL* S4* R3*/R2        +
             /SRNSL* S4* R3* R2

       I2  =        S2*/R3*/R2        +
              SRNSL* S1*/R3* R2        +
              SRNSL* SO* R3*/R2        +
             /SRNSL* S3*/R3* R2        +
             /SRNSL* S4* R3*/R2        +
             /SRNSL* S4* R3* R2

      I1 =         S1*/R3*/R2        +
             SRNSL* SO*/R3* R2        +
            /SRNSL* S2*/R3* R2        +
            /SRNSL* S3* R3*/R2        +
            /SRNSL* S4* R3* R2

      IO =         SO*/R3*/R2        +
            /SRNSL* S1*/R3* R2        +
            /SRNSL* S2* R3*/R2        +
            /SRNSL* S3* R3* R2
```

```
FUNCTION TABLE
S4 S3 S2 S1 S0     R3 R2 R1 R0    SRNSL OE    I3 I2 I1 I0 ZOE Z
---------------------------------------------------------------------
;SHIFT ZERO
L  H  H  H  L      L  L  L  L      H    L      H  H  H  L  H   Z
L  H  H  H  L      L  L  L  L      L    L      H  H  H  L  H   Z
;SHIFT ONE
L  H  L  H  H      L  H  L  H      H    L      L  H  H  L  H   Z
L  H  L  H  H      L  H  L  H      L    L      L  H  L  H  H   Z
;SHIFT TWO
L  H  H  L  L      H  L  H  L      H    L      L  L  L  L  L   Z
L  H  H  L  L      H  L  H  L      L    L      L  L  H  H  H   Z
;SHIFT THREE
L  H  L  H  H      H  H  H  H      H    L      H  L  L  L  H   Z
L  H  L  H  H      H  H  H  H      L    L      L  L  L  H  L   Z
---------------------------------------------------------------------
```

DESCRIPTION
NIBBLE SHIFTER D IS THE LAST OF THE FOUR NIBBLE SHIFTER SLICES.
IT USES DATA INPUTS SBUS <12,8,4,0>.

NIBBLE SHIFTER D
ADVANCED MICRO DEVICES
*D9725
F0
L0000 1111 1111 1111 1111 1111 1111 1111 1101 *
L0032 1111 1101 1111 1111 1111 1111 1111 1111 *
L0256 1111 1111 1111 1111 1111 1111 1111 1111 *
L0288 1111 1111 1101 1111 1111 1111 1111 1111 *
L0320 1111 1111 1111 1101 1111 1111 1111 1111 *
L0352 1111 1111 1111 1111 1101 1111 1111 1111 *
L0384 1111 1111 1111 1111 1111 1101 1001 1011 *
L0512 1111 1111 1111 1111 1111 1111 1111 1111 *
L0544 0111 1111 1111 1111 1011 1011 1111 1111 *
L0576 1111 0111 1111 1111 1011 0111 1101 1111 *
L0608 1111 1111 0111 1111 0111 0111 1101 1111 *
L0640 1111 1111 1111 0111 0111 0111 1101 1111 *
L0672 1101 1111 1111 1111 1011 0111 1110 1111 *
L0704 1101 1111 1111 1111 0111 1011 1110 1111 *
L0736 1101 1111 1111 1111 0111 0111 1110 1111 *
L0768 1111 1111 1111 1111 1111 1111 1111 1111 *
L0800 1111 0111 1111 1111 1011 1011 1111 1111 *
L0832 1111 1111 0111 1111 1011 0111 1101 1111 *
L0864 1111 1111 1111 0111 0111 1011 1101 1111 *
L0896 0111 1111 1111 1111 1011 0111 1110 1111 *
L0928 1101 1111 1111 1111 0111 1011 1110 1111 *
L0960 1101 1111 1111 1111 0111 0111 1110 1111 *
L1024 1111 1111 1111 1111 1111 1111 1111 1111 *
L1056 1111 1111 0111 1111 1011 1011 1111 1111 *
L1088 1111 1111 1111 0111 1011 0111 1101 1111 *
L1120 1111 0111 1111 1111 1011 0111 1110 1111 *
L1152 0111 1111 1111 1111 0111 1011 1110 1111 *
L1184 1101 1111 1111 1111 0111 0111 1110 1111 *
L1280 1111 1111 1111 1111 1111 1111 1111 1111 *
L1312 1111 1111 1111 0111 1011 1011 1111 1111 *
L1344 1111 1111 0111 1111 1011 0111 1110 1111 *
L1376 1111 0111 1111 1111 0111 1011 1110 1111 *
L1408 0111 1111 1111 1111 0111 0111 1110 1111 *
C7F25*
V0001 01110000000X1LHHHHZ1 *
V0002 01110000000XOLHHHHZ1 *
V0003 01011010100X1LHHLHZ1 *
V0004 01011010100XOHLHLHZ1 *
V0005 01100101000X1LLLLLZ1 *
V0006 01100101000XOHHLLHZ1 *
V0007 01011111100X1LLLHHZ1 *
V0008 01011111100XOHLLLLZ1 *
8617

LOGIC DIAGRAM FOR:
NIBBLE SHIFTER D USING AmPAL16H8A

Columns: Inputs (0–31)
Rows: Product Terms (0–63)

Z

ZOE

I_3

I_2

I_1

I_0

SRNSL

OE

03862A-156

$\times$ = Fuse intact $\boxtimes$ = All fuses intact $+$ = Fuse blown

5-121
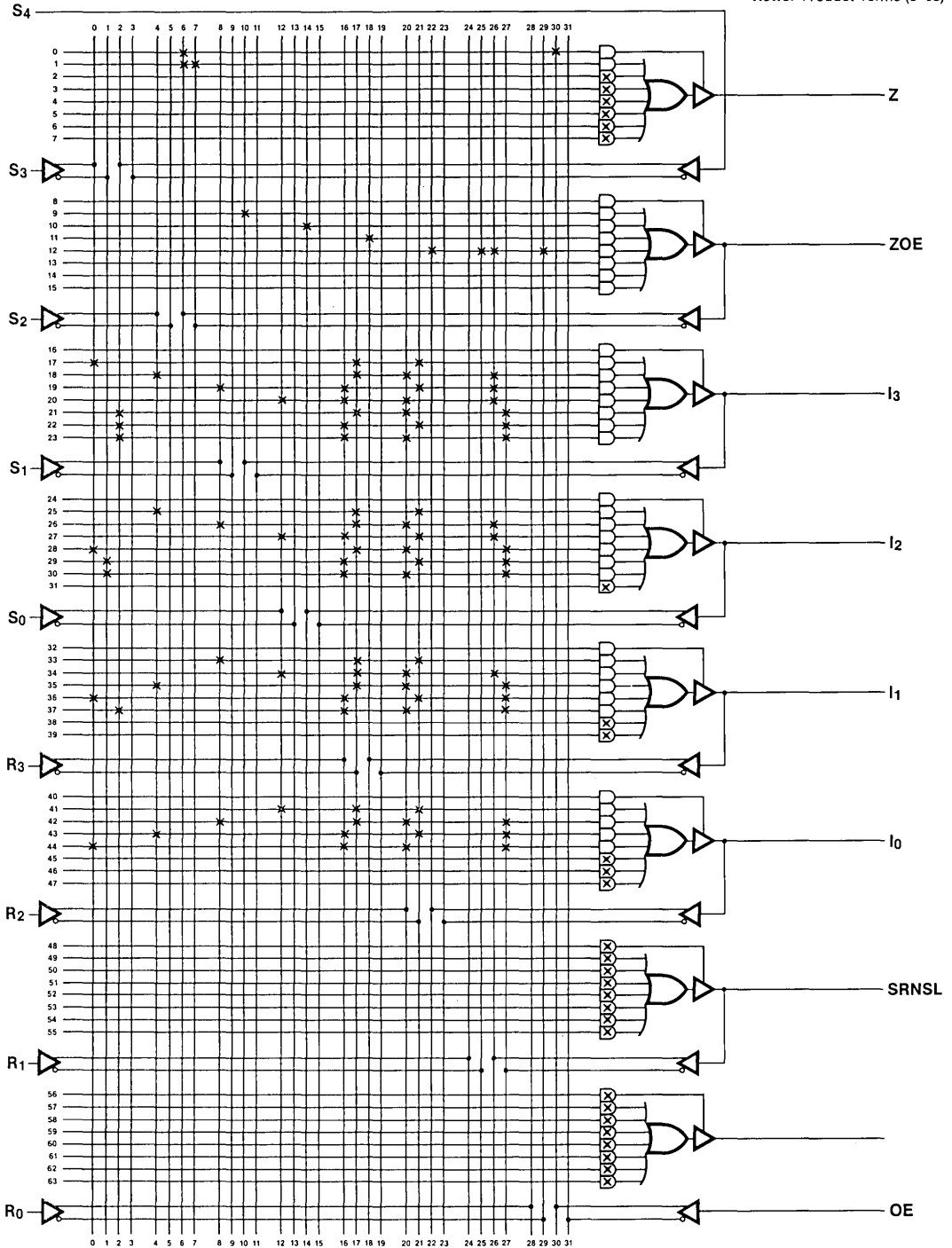
```
PAL16R8                                    PAL DESIGN SPECIFICATION
PAT017                                     JENNY YEE        10/4/82
INSTRUCTION REGISTER
ADVANCED MICRO DEVICES
 CLK D5  D4  D3  D2  D1  D0  NC /CE GND
/OE  MO  M1  M2 IR0 IR1 IR2 IR3 IR4 VCC

;INSTRUCTION REGISTER OUTPUT SIGNALS

/IR4       :=  CE*/D4                     +
               /CE*/IR4

/IR3       :=  CE*/D3                     +
               /CE*/IR3

/IR2       :=  CE*/D2                     +
               /CE*/IR2

/IR1       :=  CE*/D1                     +
               /CE*/IR1

/IR0       :=  CE*/D0                     +
               /CE*/IR0

/M2        := /CE*/M2                        +
              CE*/D5                         +
              CE* D5* D4* D3*/D2*/D1*/D0     +
              CE* D5* D4* D3*/D2*/D1* D0     +
              CE* D5* D4* D3*/D2* D1*/D0

/M1        := /CE*/M1                        +
              CE* D5*/D4*/D3                 +
              CE* D5*/D4* D3                 +
              CE* D5* D4* D3*/D2*/D1*/D0     +
              CE* D5* D4* D3*/D2*/D1* D0

/MO        := /CE*/MO                        +
              CE* D5*/D4*/D3                 +
              CE* D5* D4*/D3                 +
              CE* D5* D4* D3*/D2*/D1*/D0     +
              CE* D5* D4* D3*/D2* D1*/D0
```

FUNCTION TABLE

CLK	/OE	/CE	D5	D4	D3	D2	D1	DO	IR4	IR3	IR2	IR1	IRO	M2	M1	MO
;LOAD REGISTERS FROM THE DBUS AND OUTPUT ONTO Q																
C	L	L	L	H	L	H	L	H	H	L	H	L	H	L	H	H
C	L	L	H	L	L	L	H	L	L	L	L	H	L	H	L	L
C	L	L	H	L	H	H	L	H	L	H	H	L	H	H	L	H
C	L	L	H	H	L	L	H	L	H	L	L	H	L	H	H	L
C	L	L	H	H	H	L	L	L	H	H	L	L	L	L	L	L
C	L	L	H	H	H	L	L	H	H	H	L	L	H	L	L	H
C	L	L	H	H	H	L	H	L	H	H	L	H	L	L	H	L
C	L	L	H	H	H	L	H	H	H	H	L	H	H	H	H	H
C	L	L	H	H	H	H	L	H	H	H	H	L	H	H	H	H
;HOLD THIS VALUE AND OUTPUT ONTO Q																
C	L	H	X	X	X	X	X	X	H	H	H	L	H	H	H	H

DESCRIPTION
THE INSTRUCTION REGISTER (IR) IS IMPLEMENTED USING TWO
AMPAL16R8AS AND ONE AM29825. THE INSTRUCTION REGISTER TAKES 16-
BITS OF INPUT DATA FROM THE DBUS, PREDECODES THE OPCODE AND
ADDRESSING MODES, GENERATING A 1O-BIT FIELD FOR THE MAPPING PROM.
IN ADDITION TO THIS, IR<11:0> IS PASSED THROUGH TO THE VALUE
REGISTER, WHICH USES THIS DATA TO GENERATE CONDITIONAL BRANCH
ADDRESSES, IMMEDIATE DATA VALUES, AND REGISTER FILE ADDRESSES.

```
*D9724
*F0*
L0000 1111 1011 1111 1111 1111 1111 1111 1011 *
L0032 1110 1111 1111 1111 1111 1111 1111 0111 *
L0256 1111 1111 1011 1111 1111 1111 1111 1011 *
L0288 1111 1110 1111 1111 1111 1111 1111 0111 *
L0512 1111 1111 1111 1011 1111 1111 1111 1011 *
L0544 1111 1111 1110 1111 1111 1111 1111 0111 *
L0768 1111 1111 1111 1111 1011 1111 1111 1011 *
L0800 1111 1111 1111 1110 1111 1111 1111 0111 *
L1024 1111 1111 1111 1111 1111 1011 1111 1011 *
L1056 1111 1111 1111 1111 1110 1111 1111 0111 *
L1280 1111 1111 1111 1111 1111 1110 1111 0111 *
L1312 1011 1111 1111 1111 1111 1111 1111 1011 *
L1344 0111 0111 0111 1011 1011 1011 1111 1011 *
L1376 0111 0111 0111 1011 1011 0111 1111 1011 *
L1408 0111 0111 0111 1011 0111 1011 1111 1011 *
L1536 1111 1111 1111 1111 1111 1111 1110 0111 *
L1568 0111 1011 1011 1111 1111 1111 1111 1011 *
L1600 0111 1011 0111 1111 1111 1111 1111 1011 *
L1632 0111 0111 0111 1011 1011 1011 1111 1011 *
L1664 0111 0111 0111 1011 1011 0111 1111 1011 *
L1792 1111 1111 1111 1111 1111 1111 1111 0110 *
L1824 0111 1011 1011 1111 1111 1111 1111 1011 *
L1856 0111 0111 1011 1111 1111 1111 1111 1011 *
L1888 0111 0111 0111 1011 1011 1011 1111 1011 *
L1920 0111 0111 0111 1011 0111 1011 1111 1011 *
C5BD1*
V0001 C010101X000HHLHLHLH1 *
V0002 C100010X000LLHLHLLL1 *
V0003 C101101X000HLHHLHHL1 *
V0004 C110010X000LHHHLHLH1 *
V0005 C111000X000LLLLLLHH1 *
V0006 C111001X000HLLHLLHH1 *
V0007 C111010X000LHLLHLHH1 *
V0008 C111011X000HHHHHLHH1 *
V0009 C111101X000HHHHLHHH1 *
V0010 CXXXXXXX100HHHHLHHH1 *
5196
```

LOGIC DIAGRAM FOR:
INSTRUCTION REGISTER USING AmPAL16R8A
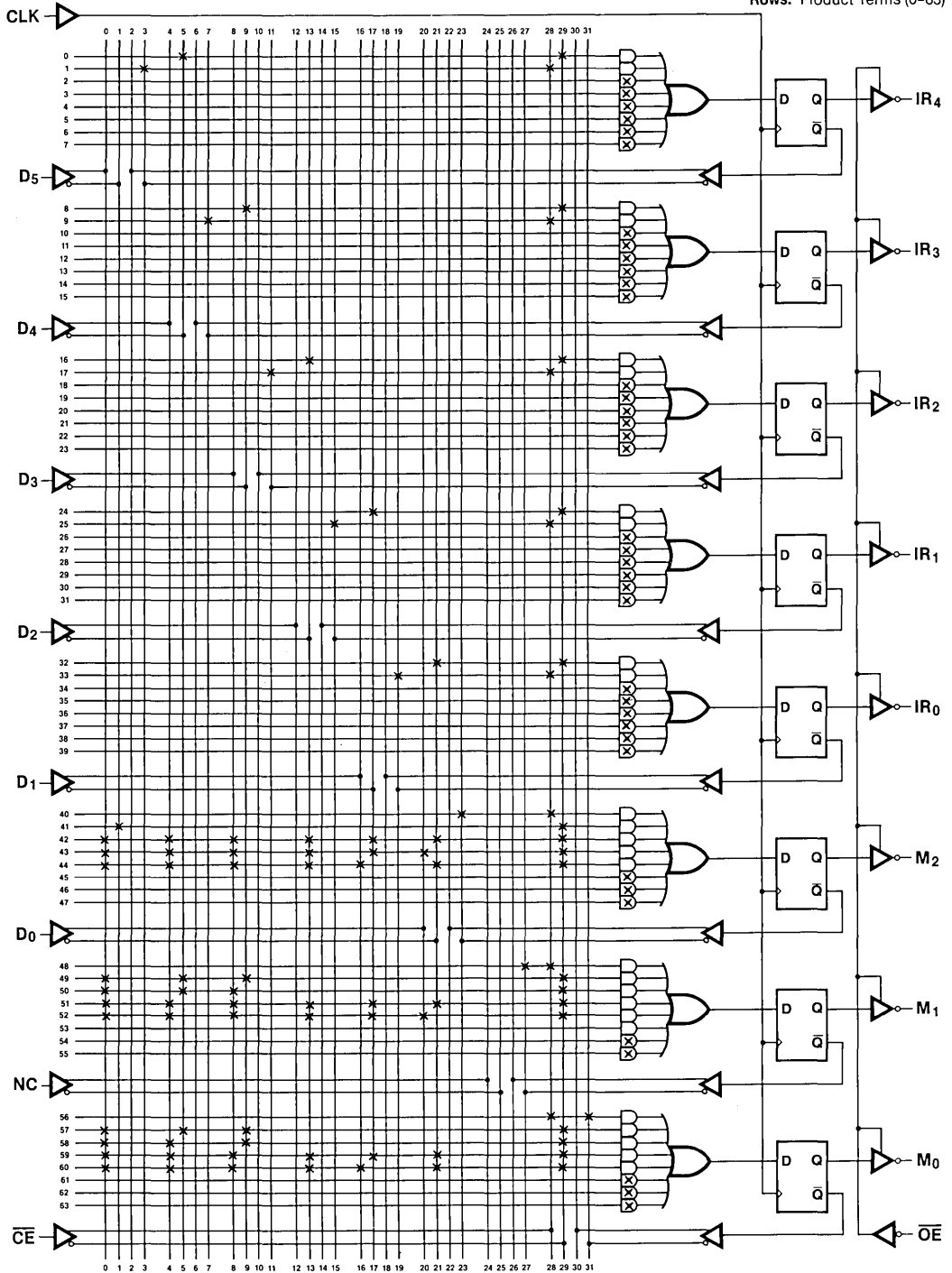
—✳— = Fuse intact   —⊗— = All fuses intact   —┼— = Fuse blown

03862A-157

```
PAL16R6                              PAL DESIGN SPECIFICATION
PATO22                               JENNY YEE        10/4/82
VALUE REGISTER AND REGISTER FILE
ADVANCED MICRO DEVICES
 CLK IR1   IR2   IR3   SL3   SL2   SL1 SL0 /CE GND
/OE  IR4   VR4   VR3   VR2   VR1   B    A    DR  VCC

;VALUE REGISTER AND REGISTER FILE OUTPUTS

/A           := /SL1*/SL0*/VR4                  +
                    SL0*/A                      +
                SL1*/SL0*/IR1

/B           := /SL3*/SL2*/B                    +
                /SL3* SL2*/SL1*/IR1             +
                /SL3* SL2* SL1*/IR4             +
                SL3*/SL2*/SL1*/VR4             +
                SL3*/SL2* SL1* SL0*/DR          +
                SL3*/SL2* SL1*/SL0*/IR1

/VR1         := /CE*/VR1                         +
                CE*/IR1

/VR2         := /CE*/VR2                   +
                CE*/IR2

/VR3         := /CE*/VR3                   +
                CE*/IR3

/VR4         := /CE*/VR4                   +
                CE*/IR4
```

FUNCTION TABLE

CLK	/OE	/CE	SL3	SL2	SL1	SL0	IR4	IR3	IR2	IR1	DR	A	B	VR4	VR3	VR2	VR1
;LOAD VR REGISTERS																	
C	L	L	X	X	X	X	L	L	H	L	X	X	X	L	L	H	L
;																	
;HOLD THIS VALUE																	
C	L	H	X	X	X	X	X	X	X	X	X	X	X	L	L	H	L
;LOADING ALL VALUES FOR A AND B																	
;A= IR1 B=IR4																	
C	L	H	L	H	H	L	L	H	L	H	L	H	L	L	L	H	L
;A=VR4 B=B																	
C	L	H	L	L	L	L	L	H	L	H	L	L	L	L	L	H	L
;A=A B=B																	
C	L	H	L	L	X	H	L	H	L	H	L	L	L	L	L	H	L
;A=IR1 B=B																	
C	L	H	L	L	H	L	L	H	L	H	L	H	L	L	L	H	L
;A=VR4 B=IR1																	
C	L	H	L	H	L	L	L	H	L	H	L	L	H	L	L	H	L
;A=A B=IR1																	
C	L	H	L	H	L	H	H	L	H	L	L	L	L	L	L	H	L
;A=A B=IR4																	
C	L	H	L	H	H	H	H	L	H	L	L	L	H	L	L	H	L
;A=VR4 B=VR4																	
C	L	H	H	L	L	L	H	L	H	L	L	L	L	L	L	H	L
;A=A B=VR4																	
C	L	H	H	L	L	H	H	L	H	L	L	L	L	L	L	H	L
;A=IR1 B=IR1																	
C	L	H	H	L	H	L	L	H	L	H	L	H	H	L	L	H	L
;A=A B=DR																	
C	L	H	H	L	H	H	L	H	L	H	L	H	L	L	L	H	L
;A=VR4 B=1																	
C	L	H	H	H	L	L	L	H	L	H	L	L	H	L	L	H	L
;A=A B=1																	
C	L	H	H	H	X	H	L	H	L	H	L	L	H	L	L	H	L
;A=IR1 B=1																	
C	L	H	H	H	H	L	L	H	L	H	L	H	H	L	L	H	L

DESCRIPTION
THE VALUE REGISTER IS NECESSARY FOR DOUBLE-PIPELINING. THIS
REGISTER, ALONG WITH THE DECREMENT REGISTER, CONTROL ALL OF THE
ADDRESSING REQUIRED FOR THE REGISTER FILE. THE VALUE REGISTER IS
IMPLEMENTED USING 3 IDENTICALLY PROGRAMMED AMPAL16R6AS. THE VALUE
REGISTER GENERATES DATA INFORMATION ON VR<11:0> (WHICH IS SIMPLY
IR<11:0> DELAYED BY ONE CLOCK CYCLE), AND THE A AND B ADDRESSES FOR
THE REGISTER FILE ON A ADR<2:0> AND B ADR<2:0>, RESPECTIVELY. THE A
AND B ADDRESS SELECTION IS CONTROLLED BY THE RFAR SEL<3:0> FROM
MICROCODE.

```
PAL16R6                                PAL DESIGN SPECIFICATION
PAT022                                 JENNY YEE      10/4/82
VALUE REGISTER AND REGISTER FILE
ADVANCED MICRO DEVICES
*D9724
*F0*
L0256 1111 1111 1111 1111 1111 1011 1010 1111 *
L0288 1111 1110 1111 1111 1111 1111 0111 1111 *
L0320 1011 1111 1111 1111 1111 0111 1011 1111 *
L0512 1111 1111 1110 1011 1011 1111 1111 1111 *
L0544 1011 1111 1111 1011 0111 1011 1111 1111 *
L0576 1111 1111 1111 1011 0111 0111 1111 1110 *
L0608 1111 1111 1111 0111 1011 1011 1110 1111 *
L0640 1110 1111 1111 0111 1011 0111 0111 1111 *
L0672 1011 1111 1111 0111 1011 0111 1011 1111 *
L0768 1111 1111 1111 1110 1111 1111 1111 0111 *
L0800 1011 1111 1111 1111 1111 1111 1111 1011 *
L1024 1111 1111 1111 1111 1110 1111 1111 0111 *
L1056 1111 1011 1111 1111 1111 1111 1111 1011 *
L1280 1111 1111 1111 1111 1111 1110 1111 0111 *
L1312 1111 1111 1011 1111 1111 1111 1111 1011 *
L1536 1111 1111 1111 1111 1111 1111 1110 0111 *
L1568 1111 1111 1111 1111 1111 1111 1111 1010 *
C3EE0*
V0001 C010XXXX0000LLHLXXX1 *
V0002 CXXXXXXX100XLLHLXXX1 *
V0003 C10101101000LLHLLH01 *
V0004 C10100001000LLHLLL01 *
V0005 C10100X11000LLHLLL01 *
V0006 C10100101000LLHLLH01 *
V0007 C10101001000LLHLHL01 *
V0008 C01001011001LLHLLL01 *
V0009 C01001111001LLHLHL01 *
V0010 C01010001001LLHLLL01 *
V0011 C01010011001LLHLLL01 *
V0012 C10110101000LLHLHH01 *
V0013 C10110111000LLHLLH01 *
V0014 C10111001000LLHLHL01 *
V0015 C10111X11000LLHLHL01 *
V0016 C10111101000LLHLHH01 *
3139
```

LOGIC DIAGRAM FOR:
VALUE REGISTER AND REGISTER FILE ADDRESS REGISTER USING AmPAL16R6A

Columns: Inputs (0–31)
Rows: Product Terms (0–63)

03862A-158

—✳— = Fuse intact   —⊗— = All fuses intact   —┼— = Fuse blown

5-129

```
PAL16R4                                    PAL DESIGN SPECIFICATION
PATO16                                     JEFF KITSON      10/4/82
DECREMENT REGISTER
ADVANCED MICRO DEVICES
CLK  D2  D1  DO  CBR VR11 S1 SO /OER GND
/OE  RO  R1  QO  Q1  Q2  NC  R2  CXV VCC

;DECREMENT REGISTER OUTPUT SIGNALS

IF (OER) /R2      =             /Q2


         /Q2    := /S1*/SO*/Q2                +
                   /S1* SO*/R2                +
                   S1*/SO*/D2                 +
                   S1* SO* Q2*/Q1*/QO         +
                   S1* SO*/Q2* QO            +
                   S1* SO*/Q2* Q1


         /Q1    := /S1*/SO*/Q1                +
                   /S1* SO*/R1                +
                   S1*/SO*/D1                 +
                   S1* SO* Q1*/QO             +
                   S1* SO*/Q1* QO


         /QO    := /S1*/SO*/QO                +
                   /S1* SO*/RO                +
                   S1*/SO*/DO                 +
                   S1* SO* QO

IF (OER) /R1      =             /Q1

IF (OER) /RO      =             /QO


         /CXV   = CBR* VR11                   +
                  /CBR*/VR11
```

FUNCTION TABLE

CLK	/OER	/OE	S1	SO	D2	D1	DO	CBR	VR11	R2	R1	RO	CXV	Q2	Q1	QO
;LOAD REGISTERS FROM RBUS AND OUTPUT ONTO Q																
C	H	L	L	H	X	X	X	X	X	L	H	L	X	L	H	L
;LOAD REGISTERS FROM DBUS AND OUTPUT ONTO Q																
C	H	L	H	L	H	L	H	X	X	Z	Z	Z	X	H	L	H
;HOLD THE VALUE AND OUTPUT ONTO Q																
C	H	L	L	L	X	X	X	X	X	Z	Z	Z	X	H	L	H
;DECREMENT VALUE AND OUTPUT ONTO Q																
C	H	L	H	H	X	X	X	X	X	Z	Z	Z	X	H	L	L
C	H	L	H	H	X	X	X	X	X	Z	Z	Z	X	L	H	H
C	H	L	H	H	X	X	X	X	X	Z	Z	Z	X	L	H	L
C	H	L	H	H	X	X	X	X	X	Z	Z	Z	X	L	L	H
C	H	L	H	H	X	X	X	X	X	Z	Z	Z	X	L	L	L
C	H	L	H	H	X	X	X	X	X	Z	Z	Z	X	H	H	H
C	H	L	H	H	X	X	X	X	X	Z	Z	Z	X	H	H	L
C	H	L	H	H	X	X	X	X	X	Z	Z	Z	X	H	L	H
;OUTPUT THE VALUE ONTO THE RBUS AND NOT ONTO Q																
X	L	H	X	X	X	X	X	X	X	H	L	H	X	Z	Z	Z
;TEST CONDITIONAL BRANCH POLARITY XOR																
L	X	X	X	X	X	X	X	L	L	X	X	X	L	X	X	X
L	X	X	X	X	X	X	X	L	H	X	X	X	H	X	X	X
L	X	X	X	X	X	X	X	H	L	X	X	X	H	X	X	X
L	X	X	X	X	X	X	X	H	H	X	X	X	L	X	X	X

DESCRIPTION
THE DECREMENT REGISTER IS IMPLEMENTED USING A SINGLE AMPAL16R4A.
IT IS LOADED FROM RBUS<2:0> WHEN THE REGISTER IS SELECTED BY
IMMEDIATE OR REGISTER ADDRESSING MODES, OR FROM DBUS<2:0> WHEN A
MEMORY ADDRESSING MODE IS USED. LOADING OR DECREMENTING IS
CONTROLLED BY DR SEL<1:0> VIA MICROCODE. THE DECREMENT REGISTER,
IN CONJUNCTION WITH THE VALUE REGISTER, PROVIDE ALL OF THE
ADDRESSING REQUIRED FOR THE REGISTER FILE. SEQUENTIAL ADDRESSING
OF THE REGITER FILE (FOR LDM AND STM INSTRUCTIONS) IS QUICK AND
EASY WITH THE DECREMENT REGISTER. IT IS ALSO USED TO PROVIDE AN
OFFSET TO THE MEMORY ADDRESS WHERE THE DATA IS TO BE STORED INTO
OR LOADED FROM.

PAL16R4   PAL DESIGN SPECIFICATION
PAT016   JEFF KITSON   10/4/82
DECREMENT REGISTER
ADVANCED MICRO DEVICES
*D9724
F0
L0000 1111 1111 1111 1111 1111 1111 1111 1111 *
L0032 1111 1111 1111 0111 0111 1111 1111 1111 *
L0064 1111 1111 1111 1011 1011 1111 1111 1111 *
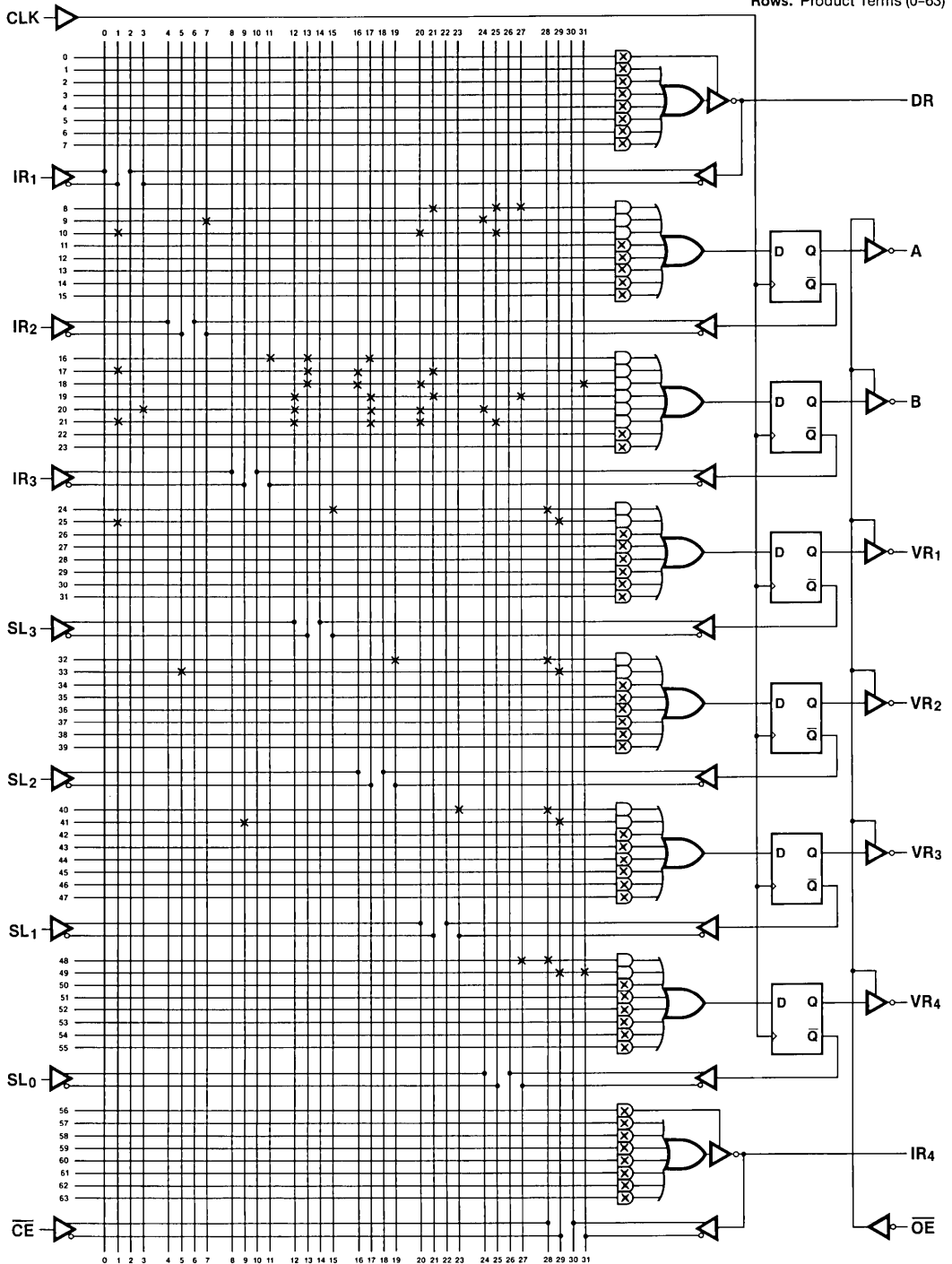L0256 1111 1111 1111 1111 1111 1111 1111 1011 *
L0288 1111 1111 1111 1110 1111 1111 1111 1111 *
L0768 1111 1111 1111 1110 1111 1011 1011 1111 *
L0800 1111 1110 1111 1111 1111 1011 0111 1111 *
L0832 1011 1111 1111 1111 1111 0111 1011 1111 *
L0864 1111 1111 1111 1101 1110 0110 0111 1111 *
L0896 1111 1111 1111 1110 1111 0101 0111 1111 *
L0928 1111 1111 1111 1110 1101 0111 0111 1111 *
L1024 1111 1111 1111 1111 1110 1011 1011 1111 *
L1056 1111 1111 1111 1111 1111 1011 0110 1111 *
L1088 1111 1011 1111 1111 1111 0111 1011 1111 *
L1120 1111 1111 1111 1111 1101 0110 0111 1111 *
L1152 1111 1111 1111 1111 1110 0101 0111 1111 *
L1280 1111 1111 1111 1111 1111 1010 1011 1111 *
L1312 1111 1111 1111 1111 1111 1011 0111 1110 *
L1344 1111 1111 1011 1111 1111 0111 1011 1111 *
L1376 1111 1111 1111 1111 1111 0101 0111 1111 *
L1536 1111 1111 1111 1111 1111 1111 1111 1011 *
L1568 1111 1111 1111 1111 1110 1111 1111 1111 *
L1792 1111 1111 1111 1111 1111 1111 1111 1011 *
L1824 1111 1111 1111 1111 1111 1110 1111 1111 *
C5754*
V0001 CXXXXX0110001LHLX0X1 *
V0002 C101XX10100ZZHLHXZX1 *
V0003 CXXXXX00100ZZHLHXZX1 *
V0004 CXXXXX11100ZZLLHXZX1 *
V0005 CXXXXX11100ZZHHLXZX1 *
V0006 CXXXXX11100ZZLHLXZX1 *
V0007 CXXXXX11100ZZHLLXZX1 *
V0008 CXXXXX11100ZZLLLXZX1 *
V0009 CXXXXX11100ZZHHHXZX1 *
V0010 CXXXXX11100ZZLHHXZX1 *
V0011 CXXXXX11100ZZHLHXZX1 *
V0012 XXXXXXXX001HLZZZXHX1 *
V0013 0XXX00XXX0XXXXXXXXL1 *
V0014 0XXX01XXX0XXXXXXXXH1 *
V0015 0XXX10XXX0XXXXXXXXH1 *
V0016 0XXX11XXX0XXXXXXXXL1 *
7DD3

LOGIC DIAGRAM FOR:
DECREMENT REGISTER USING AmPAL16R4A

Columns: Inputs (0–31)
Rows: Product Terms (0–63)
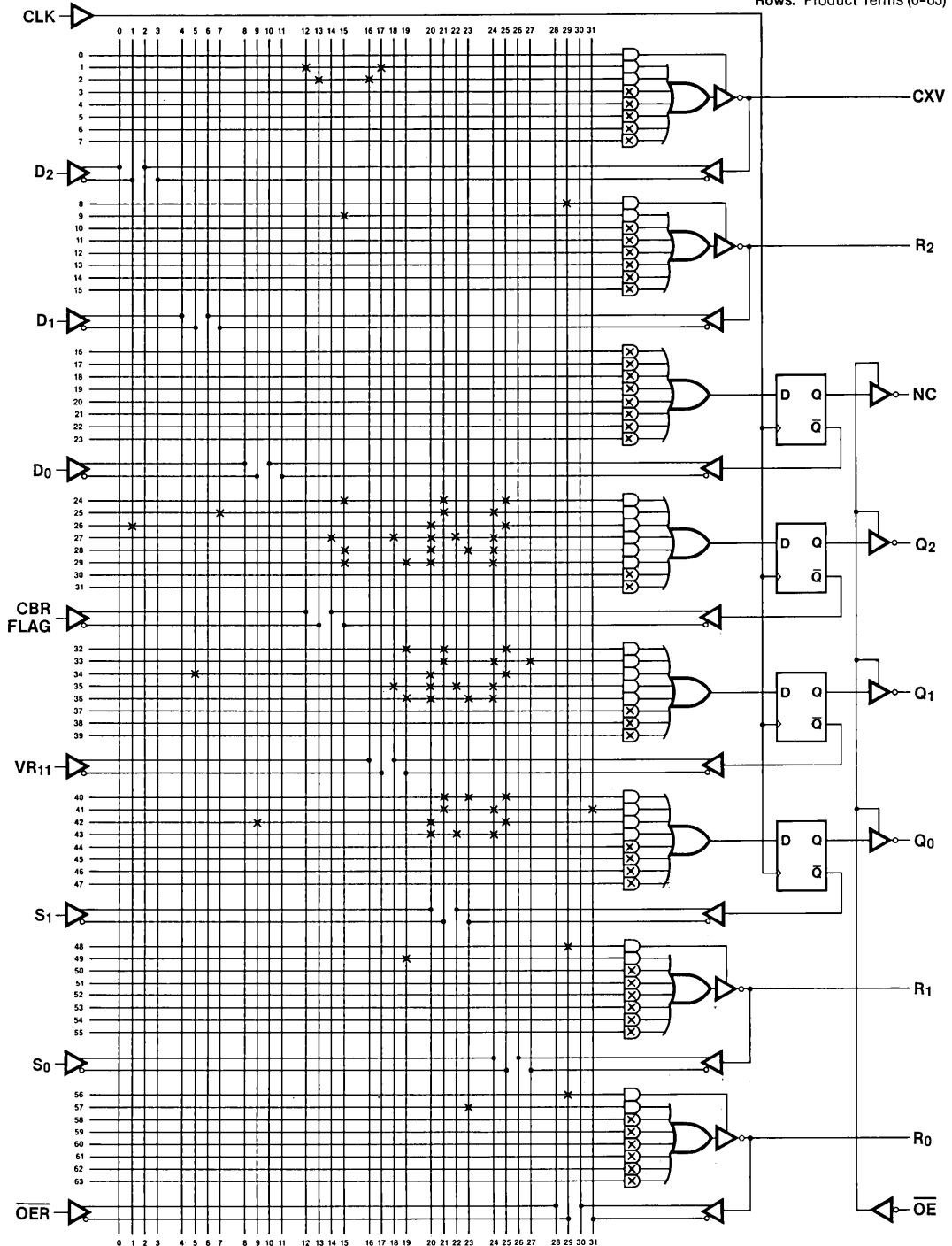
*= Fuse intact   = All fuses intact   = Fuse blown

03862A-159

```
PAL16H8                                        PAL DESIGN SPECIFICATION
PAT012                                         JENNY YEE      10/4/82
RBUS CONSTANT CONTROL A
ADVANCED MICRO DEVICES
VR6 VR7 VR10 NC NC NC  /OEH  /OEL S GND
NC  R9  R8  R7 R6 R5  R4  R1  RO  VCC

;RBUS A OUTPUT SIGNAL

IF (OEL)  RO      = VR6*S
IF (OEL)  R1      = VR7*S
IF (OEH)  R4      = VR10*S
IF (OEH)  R5      = VR10*S
IF (OEH)  R6      = VR10*S
IF (OEH)  R7      = VR10*S
IF (OEH)  R8      = VR10*S
IF (OEH)  R9      = VR10*S
```

FUNCTION TABLE

VR6	VR7	VR10	/OEH	/OEL	S		RO	R1	R4	R5	R6	R7	R8	R9
;SELECT RO AND R1														
X	X	X	H	L	L		L	L	Z	Z	Z	Z	Z	Z
L	L	X	H	L	H		L	L	Z	Z	Z	Z	Z	Z
L	H	X	H	L	H		L	H	Z	Z	Z	Z	Z	Z
H	L	X	H	L	H		H	L	Z	Z	Z	Z	Z	Z
H	H	X	H	L	H		H	H	Z	Z	Z	Z	Z	Z
;SELECT R4 TO R9														
X	X	L	L	H	H		Z	Z	L	L	L	L	L	L
X	X	H	L	H	H		Z	Z	H	H	H	H	H	H
X	X	X	L	H	L		Z	Z	L	L	L	L	L	L
;SELECT RO TO R9														
H	H	H	L	L	H		H	H	H	H	H	H	H	H
X	X	X	L	L	L		L	L	L	L	L	L	L	L

DESCRIPTION
THE RBUS CONTROL PERFORMS SIGN EXTENSION FOR SHORT IMMEDIATE
SOURCE OPERANDS AND PUTS CONSTANT VALUES ON THE RBUS. THE
FUNCTION IS CONTROLLED BY THE RBUS SELECT SIGNAL FROM
MICROCODE. ITS IMPLEMENTATION REQUIRES TWO AMPAL16H8AS.
RBUS CONTROL A MANIPULATES RBUS<9:4,1:0> USING INPUTS FROM
THE VALUE REGISTER, VR<10,7,6>.

*D9725
F0
L0000 1111 1111 1111 1111 1111 1111 1011 1111 *
L0032 1101 1111 1111 1111 1111 1111 1111 0111 *
L0256 1111 1111 1111 1111 1111 1111 1011 1111 *
L0288 0111 1111 1111 1111 1111 1111 1111 0111 *
L0512 1111 1111 1111 1111 1111 1011 1111 1111 *
L0544 1111 0111 1111 1111 1111 1111 1111 0111 *
L0768 1111 1111 1111 1111 1111 1011 1111 1111 *
L0800 1111 0111 1111 1111 1111 1111 1111 0111 *
L1024 1111 1111 1111 1111 1111 1011 1111 1111 *
L1056 1111 0111 1111 1111 1111 1111 1111 0111 *
L1280 1111 1111 1111 1111 1111 1011 1111 1111 *
L1312 1111 0111 1111 1111 1111 1111 1111 0111 *
L1536 1111 1111 1111 1111 1111 1011 1111 1111 *
L1568 1111 0111 1111 1111 1111 1111 1111 0111 *
L1792 1111 1111 1111 1111 1111 1011 1111 1111 *
L1824 1111 0111 1111 1111 1111 1111 1111 0111 *
C3E17*
V0001 XXXXXX1000XZZZZZZLL1 *
V0002 00XXXX1010XZZZZZZLL1 *
V0003 01XXXX1010XZZZZZZHL1 *
V0004 10XXXX1010XZZZZZZLH1 *
V0005 11XXXX1010XZZZZZZHH1 *
V0006 XX0XXX0110XLLLLLLZZ1 *
V0007 XX1XXX0110XHHHHHHZZ1 *
V0008 XXXXXX0100XLLLLLLZZ1 *
V0009 111XXX0010XHHHHHHHH1 *
V0010 XXXXXX0000XLLLLLLLL1 *
0C71

LOGIC DIAGRAM FOR:
RBUS CONSTANT CONTROL A USING AmPAL16H8

Columns: Inputs (0–31)
Rows: Product Terms (0–63)

VR6
VR7
VR10
NC
NC
NC
OEH
OEL
S

R0
R1
R4
R5
R6
R7
R8
R9
NC

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

03862A-160

—✱— = Fuse intact   —⊠— = All fuses intact   —┼— = Fuse blown

5-136

```
PAL16H8                                PAL DESIGN SPECIFICATION
PATO13                                 JENNY YEE      10/4/82
RBUS CONSTANT CONTROL B
ADVANCED MICRO DEVICES
VR8 VR9  VR10  NC  NC  NC  /OEH  /OEL  S  GND
NC  R15  R14  R13  R12  R11  R10  R3  R2  VCC

;RBUS B OUTPUT SIGNALS

IF (OEL)  R2    = VR8*S
IF (OEH)  R3    = VR9*S
IF (OEH)  R10   = VR10*S
IF (OEH)  R11   = VR10*S
IF (OEH)  R12   = VR10*S
IF (OEH)  R13   = VR10*S
IF (OEH)  R14   = VR10*S
IF (OEH)  R15   = VR10*S


FUNCTION TABLE

VR8 VR9 VR10 /OEH /OEL S      R2 R3 R10 R11 R12 R13 R14 R15
-----------------------------------------------------------------
;SELECT R2 ONLY
 X   X   X    H    L   L      L  Z   Z   Z   Z   Z   Z   Z
 H   X   X    H    L   H      H  Z   Z   Z   Z   Z   Z   Z
 L   X   X    H    L   H      L  Z   Z   Z   Z   Z   Z   Z
;SELECT R3 TO R15
 X   X   X    L    H   L      Z  L   L   L   L   L   L   L
 X   L   L    L    H   H      Z  L   L   L   L   L   L   L
 X   H   H    L    H   H      Z  H   H   H   H   H   H   H
 X   H   L    L    H   H      Z  H   L   L   L   L   L   L
 X   L   H    L    H   H      Z  L   H   H   H   H   H   H
;SELECT R2 TO R15
 X   X   X    L    L   L      L  L   L   L   L   L   L   L
-----------------------------------------------------------------

DESCRIPTION
THE RBUS CONTROL B HANDLES RBUS<15:10,3:2>.  ITS INPUTS ARE
PROVIDED BY VR<10:8>.
```

```
PAL16H8                                    PAL DESIGN SPECIFICATION
PAT013                                     JENNY YEE      10/4/82
RBUS CONSTANT CONTROL B
ADVANCED MICRO DEVICES
*D9725
*FO*
L0000 1111 1111 1111 1111 1111 1111 1011 1111 *
L0032 1101 1111 1111 1111 1111 1111 1111 0111 *
L0256 1111 1111 1111 1111 1111 1011 1111 1111 *
L0288 0111 1111 1111 1111 1111 1111 1111 0111 *
L0512 1111 1111 1111 1111 1111 1011 1111 1111 *
L0544 1111 0111 1111 1111 1111 1111 1111 0111 *
L0768 1111 1111 1111 1111 1111 1011 1111 1111 *
L0800 1111 0111 1111 1111 1111 1111 1111 0111 *
L1024 1111 1111 1111 1111 1111 1011 1111 1111 *
L1056 1111 0111 1111 1111 1111 1111 1111 0111 *
L1280 1111 1111 1111 1111 1111 1011 1111 1111 *
L1312 1111 0111 1111 1111 1111 1111 1111 0111 *
L1536 1111 1111 1111 1111 1111 1011 1111 1111 *
L1568 1111 0111 1111 1111 1111 1111 1111 0111 *
L1792 1111 1111 1111 1111 1111 1011 1111 1111 *
L1824 1111 0111 1111 1111 1111 1111 1111 0111 *
C3DF9*
V0001 XXXXXX1000XZZZZZZZL1 *
V0002 1XXXXX1010XZZZZZZZH1 *
V0003 0XXXXX1010XZZZZZZZL1 *
V0004 XXXXXX0100XLLLLLLLZ1 *
V0005 X00XXX0110XLLLLLLLZ1 *
V0006 X11XXX0110XHHHHHHHZ1 *
V0007 X10XXX0110XLLLLLLHZ1 *
V0008 X01XXX0110XHHHHHHLZ1 *
V0009 XXXXXX0000XLLLLLLLL1 *
052C
```
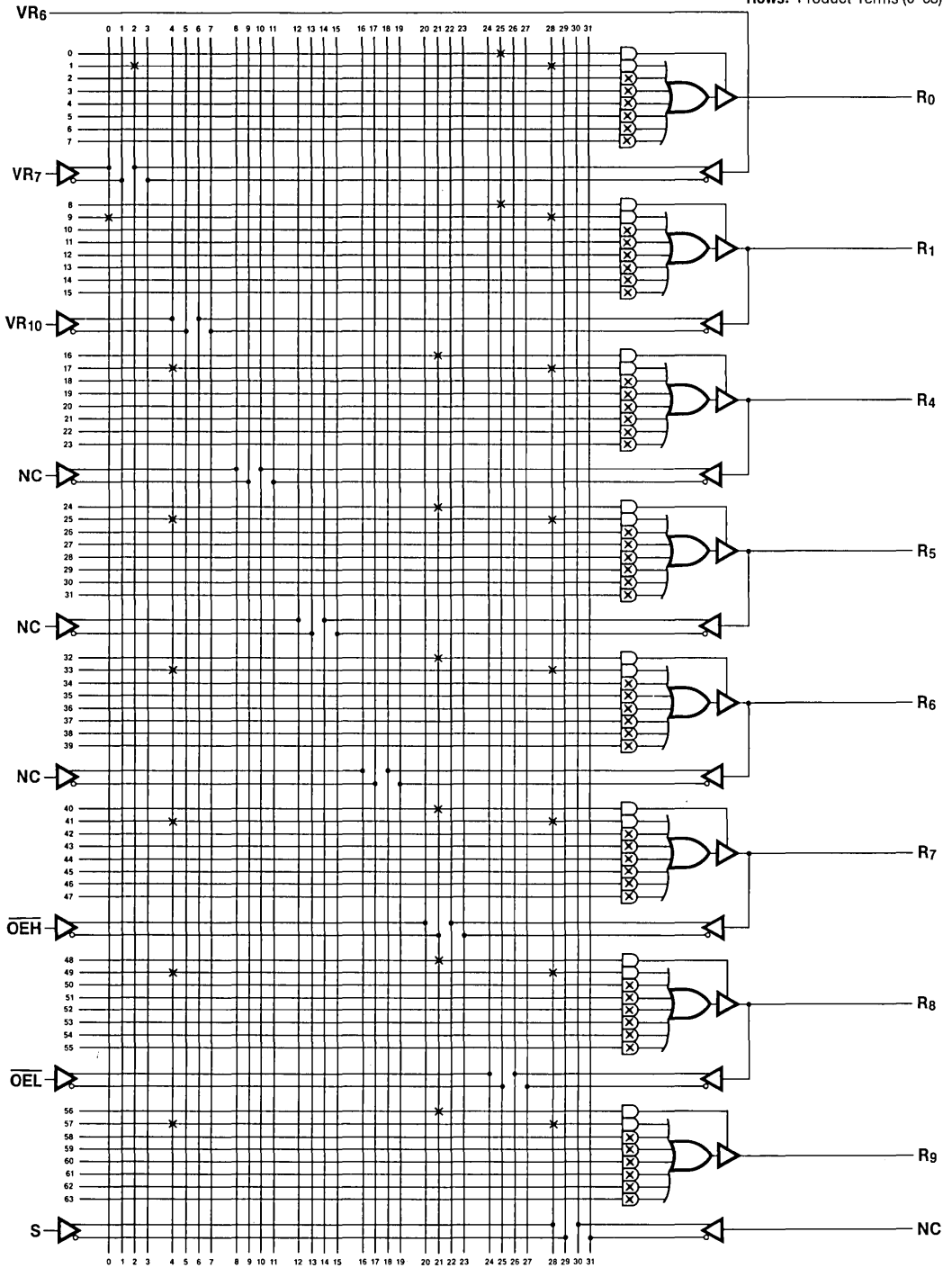
LOGIC DIAGRAM FOR:
RBUS CONSTANT CONTROL B USING AmPAL16H8

Columns: Inputs (0–31)
Rows: Product Terms (0–63)
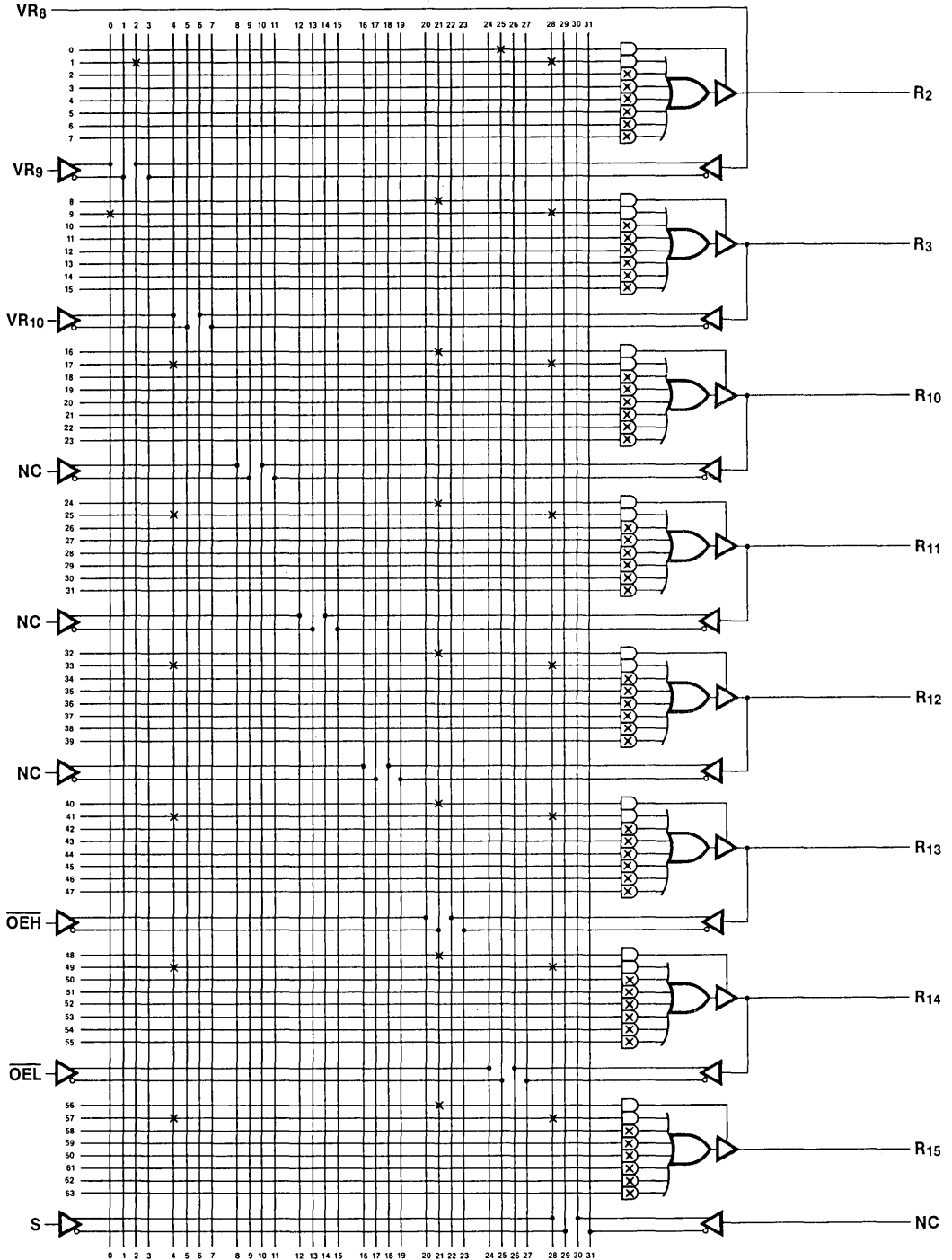
✳ = Fuse intact   ⊠ = All fuses intact   ┼ = Fuse blown

03862A-161

5-139

```
PAL16R6                                    PAL DESIGN SPECIFICATION
PAT014                                     JENNY YEE        10/4/82
CBR CONTROL
ADVANCED MICRO DEVICES
 CLK Z   N    C    V  VR10  VR9  VR8  ZN  GND
/OE  CV CBR  NXORV  VF  CF   NF   ZF  NC  VCC

 ;CBR CONTROL OUTPUT SIGNALS

/ZF          :=  ZN*/Z          +
                 /ZN*/ZF


/NF          :=  ZN*/N          +
                 /ZN*/NF


/CF          :=  CV*/C          +
                 /CV*/CF


/VF          :=  CV*/V          +
                 /CV*/VF


/NXORV       := /ZN*/CV*/NXORV   +
                /ZN* CV* NF*/V   +
                /ZN* CV*/NF*/V   +
                ZN*/CV*/N*/VF    +
                ZN*/CV* N* VF    +
                ZN* CV*/N*/V     +
                ZN* CV* N* V

/CBR         := /VR10*/VR9*/VR8*/NF          +
                /VR10*/VR9* VR8*/ZF          +
                /VR10* VR9* VR8*/CF          +
                 VR10*/VR9*/VR8*/VF          +
                 VR10*/VR9* VR8*/NXORV       +
                 VR10* VR9*/VR8*/NXORV*/ZF   +
                 VR10* VR9* VR8*/ZF*/NF
```

FUNCTION TABLE

CLK	/OE	ZN	CV	VR10	VR9	VR8	Z	N	C	V	ZF	NF	CF	VF	NXORV	CBR
;LOAD Z,N,C,AND V REGISTERS (TEST NXORV)																
C	L	H	H	X	X	X	H	L	H	L	H	L	H	L	L	X
;HOLD Z,N,C,AND V REGISTERS (TEST NXORV)																
C	L	L	L	X	X	X	X	X	X	X	H	L	H	L	L	X
;TEST NXORV UNDER OTHER CONDITIONS																
C	L	H	L	X	X	X	L	H	X	X	L	H	H	L	H	X
C	L	L	H	X	X	X	X	X	L	H	L	H	L	H	L	X
;TESTCBR FOR ALL COMBINATIONS OF VR																
;AUTOMATIC BRANCH--ALWAYS ONE																
C	L	L	L	L	L	L	X	X	X	X	L	H	L	H	L	H
;BRANCH ON ZF																
C	L	L	L	L	L	H	X	X	X	X	L	H	L	H	L	L
;BRANCH ON NF																
C	L	L	L	L	H	L	X	X	X	X	L	H	L	H	L	H
;BRANCH ON CF																
C	L	L	L	L	H	H	X	X	X	X	L	H	L	H	L	L
;BRANCH ON VF																
C	L	L	L	H	L	L	X	X	X	X	L	H	L	H	L	H
;BRANCH ON NXORV																
C	L	L	L	H	L	H	X	X	X	X	L	H	L	H	L	L
;BRANCH ON NXORV OR ZF																
C	L	L	L	H	H	L	X	X	X	X	L	H	L	H	L	L
;BRANCH ON NF OR ZF																
C	L	L	L	H	H	H	X	X	X	X	L	H	L	H	L	H

DESCRIPTION
THE CBR CONTROL PAL GENERATES ALL THE CONTROL NECESSARY FOR THE
EXECUTION OF CONDITIONAL BRANCH INSTRUCTIONS, SIGN EXTENSION FOR
SHORT IMMEDIATE DESTINATION OPERANDS, AND CONSTANT GENERATION.
THE IMPLEMENTATION REQUIRES ONE AMPAL16R6A. BRANCH CONDITIONS ARE
GENERATED FROM THE FOUR CONDITIONS CODES: ZERO(Z), MINUS(N),
OVERFLOW(V), AND CARRY(C). THE CONDITION BEING TESTED IS SELECTED
BY VR<10:8>.

CBR CONTROL
ADVANCED MICRO DEVICES
*D9724
FO
L0256 1011 1111 1111 1111 1111 1111 1111 0111 *
L0288 1111 1110 1111 1111 1111 1111 1111 1011 *
L0512 1111 1011 1111 1111 1111 1111 1111 0111 *
L0544 1111 1111 1110 1111 1111 1111 1111 1011 *
L0768 1111 1111 1011 1111 1111 1111 1111 1101 *
L0800 1111 1111 1111 1110 1111 1111 1111 1110 *
L1024 1111 1111 1111 1011 1111 1111 1111 1101 *
L1056 1111 1111 1111 1111 1110 1111 1111 1110 *
L1280 1111 1111 1111 1111 1111 1110 1111 1010 *
L1312 1111 1111 1101 0111 1111 1111 1111 1001 *
L1344 1111 1111 1110 1011 1111 1111 1111 1001 *
L1376 1111 1011 1111 1111 1110 1111 1111 0110 *
L1408 1111 0111 1111 1111 1101 1111 1111 0110 *
L1440 1111 1011 1111 1011 1111 1111 1111 0101 *
L1472 1111 0111 1111 0111 1111 1111 1111 0101 *
L1536 1111 1111 1110 1111 1011 1011 1011 1111 *
L1568 1111 1110 1111 1111 1011 1011 0111 1111 *
L1600 1111 1111 1111 1110 1011 0111 0111 1111 *
L1632 1111 1111 1111 1111 0110 1011 1011 1111 *
L1664 1111 1111 1111 1111 0111 1010 0111 1111 *
L1696 1111 1110 1111 1111 0111 0110 1011 1111 *
L1728 1111 1110 1110 1111 0111 0111 0111 1111 *
C4C20*
V0001 C1010XXX1001XLLHLHX1 *
V0002 CXXXXXXX0000XLLHLHX1 *
V0003 C01XXXXX1000XHLHHLX1 *
V0004 CXX01XXX0001XLHLHLX1 *
V0005 CXXXX0000000HLHLHLX1 *
V0006 CXXXX0010000LLHLHLX1 *
V0007 CXXXX0100000HLHLHLX1 *
V0008 CXXXX0110000LLHLHLX1 *
V0009 CXXXX1000000HLHLHLX1 *
V0010 CXXXX1010000LLHLHLX1 *
V0011 CXXXX1100000LLHLHLX1 *
V0012 CXXXX1110000HLHLHLX1 *
4800

LOGIC DIAGRAM FOR:
CBR CONTROL USING AmPAL16R6A
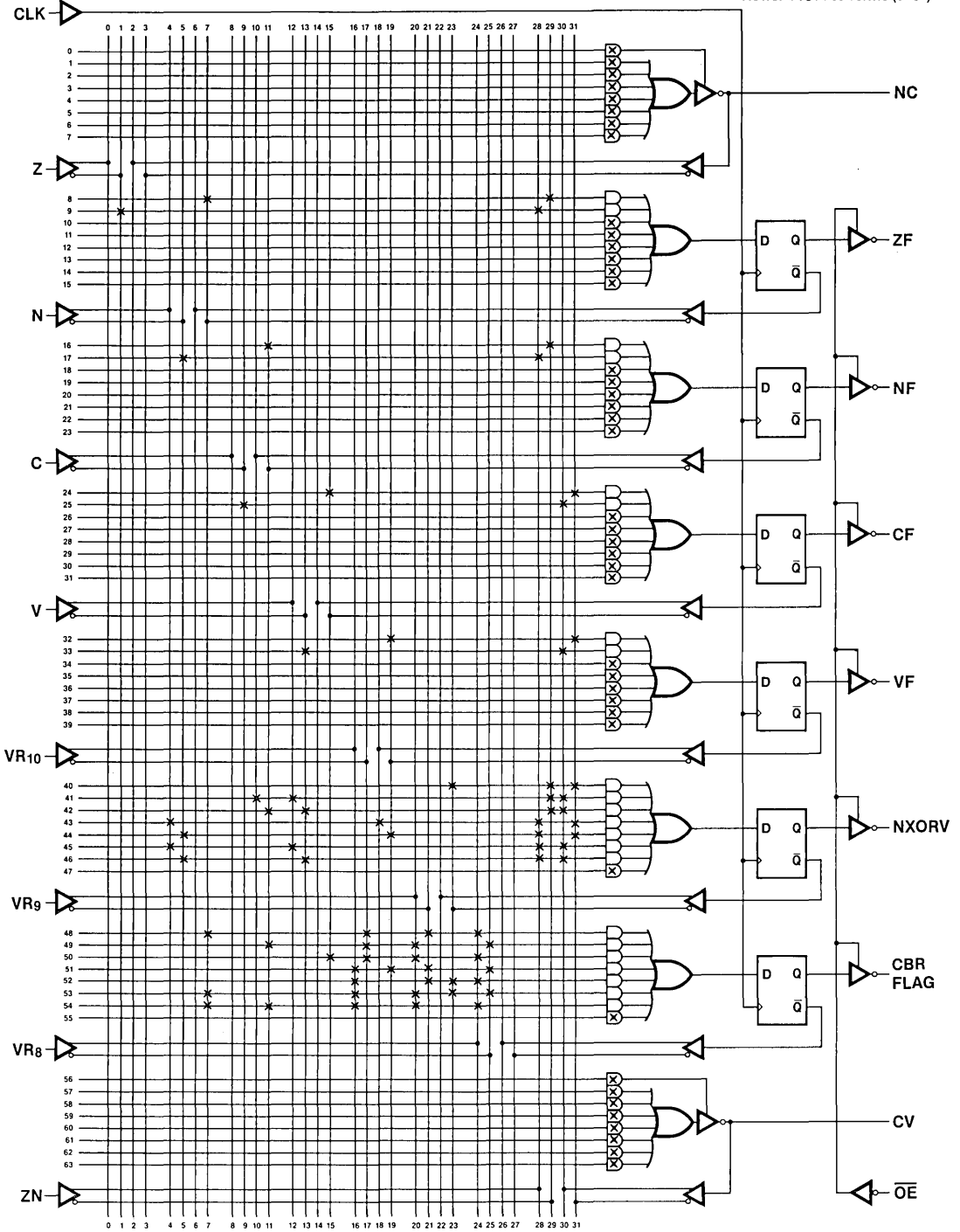
= Fuse intact   ⊠ = All fuses intact   = Fuse blown

03862A-162

```
PAL16H8                                    PAL DESIGN SPECIFICATION
PAT014                                     JEFF KITSON   10/4/82
SBUS CONTROL A
ADVANCED MICRO DEVICES
VR0  VR1  VR2  VR4 VR7 VR11 CBR  SL1 SL0 GND
/OE  S10  S9   S8  S7  S4   S2   S1  S0  VCC

;SBUS A OUTPUT SIGNAL

IF (OE)   S0       =  /SL1* SL0                  +
                      SL1*/SL0* VR0              +
                      SL1* SL0*/VR0* VR11*/CBR   +
                      SL1* SL0*/VR0*/VR11* CBR

IF (OE)   S1       =  SL1*/SL0* VR1              +
                      SL1* SL0*/VR1* VR11*/CBR   +
                      SL1* SL0*/VR1*/VR11* CBR

IF (OE)   S2       =  SL1*/SL0* VR2              +
                      SL1* SL0*/VR2* VR11*/CBR   +
                      SL1* SL0*/VR2*/VR11* CBR

IF (OE)   S4       =  SL1*/SL0* VR4              +
                      SL1* SL0*/VR4* VR11*/CBR   +
                      SL1* SL0*/VR4*/VR11* CBR

IF (OE)   S7       =  SL1*/SL0* VR4              +
                      SL1* SL0*/VR7* VR11*/CBR   +
                      SL1* SL0*/VR7*/VR11* CBR

IF (OE)   S8       =  SL1*/SL0* VR4              +
                      SL1* SL0*/VR7* VR11*/CBR   +
                      SL1* SL0*/VR7*/VR11* CBR

IF (OE)   S9       =  SL1*/SL0* VR4              +
                      SL1* SL0*/VR7* VR11*/CBR   +
                      SL1* SL0*/VR7*/VR11* CBR

IF (OE)   S10      =  SL1*/SL0* VR4              +
                      SL1* SL0*/VR7* VR11*/CBR   +
                      SL1* SL0*/VR7*/VR11* CBR
```

FUNCTION TABLE

VR0	VR1	VR2	VR4	VR7	VR11	CBR	SL1	SL0	/OE	S0	S1	S2	S4	S7	S8	S9	S10
;SELECT ZEROS
X	X	X	X	X	X	X	L	L	L	L	L	L	L	L	L	L	L
;SELECT ONES
X	X	X	X	X	X	X	L	H	L	H	L	L	L	L	L	L	L
;SELECT VR0 TO VR4
L	H	L	H	X	X	X	H	L	L	L	H	L	H	H	H	H	H
H	L	H	L	X	X	X	H	L	L	H	L	H	L	L	L	L	L
;SELECT CBR OFFSET VR(0-7) OR ZEROS
H	H	H	H	H	L	H	H	H	L	L	L	L	L	L	L	L	L
L	H	L	H	L	L	H	H	H	L	H	L	H	L	H	H	H	H
H	L	H	L	H	H	L	H	H	L	L	H	L	H	L	L	L	L
L	L	L	L	L	H	L	H	H	L	H	H	H	H	H	H	H	H

DESCRIPTION
THE IMPLEMENTATION OF THE SBUS CONTROL REQUIRES TWO AMPAL16H8AS. THE
FUNCTIONS PERFORMED BY THE PALS ARE CONTROLLED BY MICROCODE INPUTS SBUS
SEL<1:0>. FOR IMMEDIATE DESTINATION OPERANDS, THESE PALS PERFORM SIGN
EXTENSION. IF A CBR INSTRUCTION IS SELECTED, A SIGN EXTENDED BRANCH
OFFSET IS DERIVED FROM VR<9:0>. VR<11> AND THE CBR FLAG INITIATE A
BRANCH WHEN THEY ARE THE SAME POLARITY, AT WHICH TIME THE OFFSET IS
ADDED TO THE PC. IF THE BRANCH CONDITION IS NOT MET, -1 IS ADDED TO
THE PC TO RESUME PROPER INSTRUCTION EXECUTION. THE SBUS CONTROL A PAL
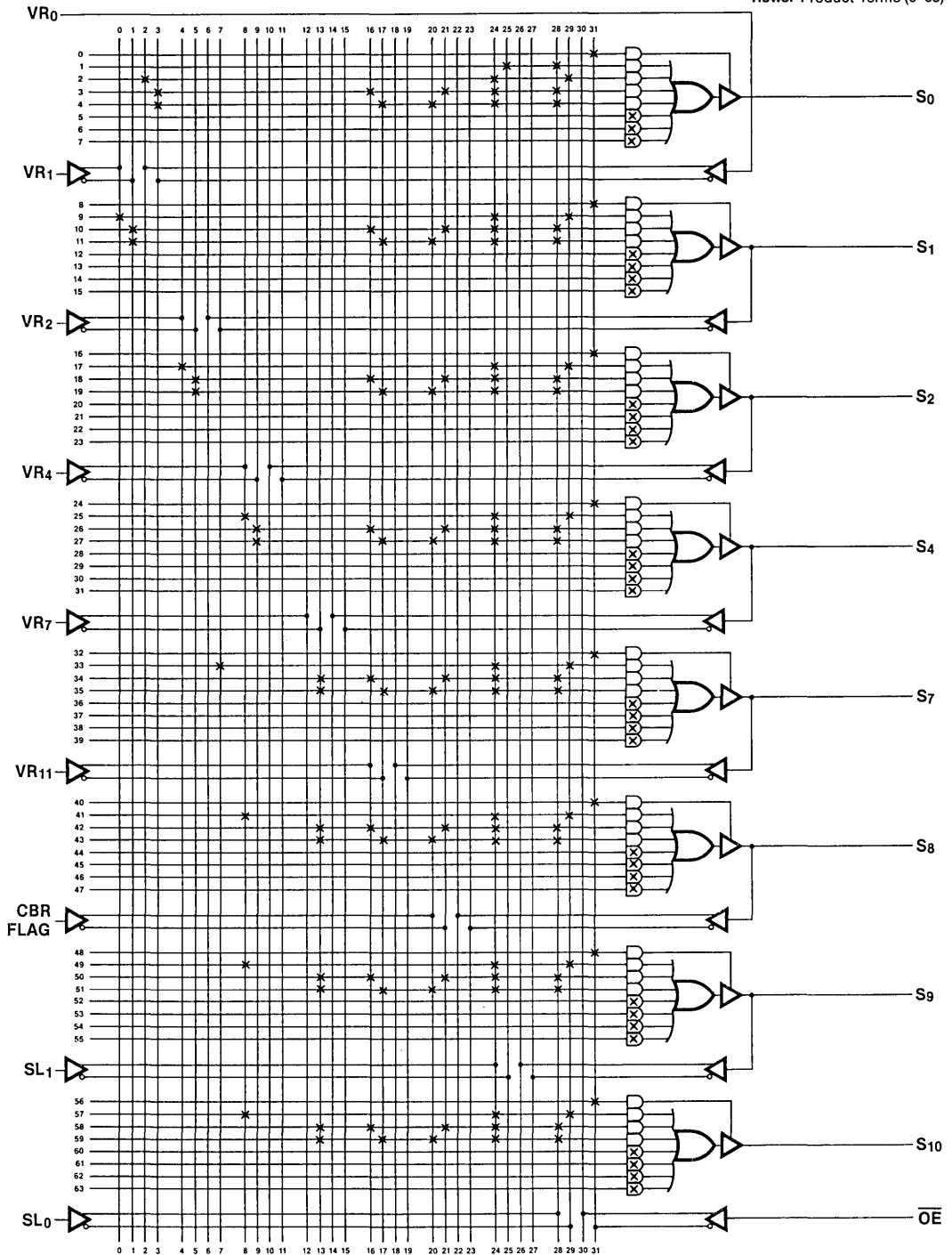CONTROLS SBUS<10:7,4,2:0>.

```
PAL16H8                                      PAL DESIGN SPECIFICATION
PAT014                                       JEFF KITSON   10/4/82
SBUS CONTROL A
ADVANCED MICRO DEVICES
*D9725
*F0*
L0000 1111 1111 1111 1111 1111 1111 1111 1110 *
L0032 1111 1111 1111 1111 1111 1111 1011 0111 *
L0064 1101 1111 1111 1111 1111 1111 0111 1011 *
L0096 1110 1111 1111 1111 0111 1011 0111 0111 *
L0128 1110 1111 1111 1111 1011 0111 0111 0111 *
L0256 1111 1111 1111 1111 1111 1111 1111 1110 *
L0288 0111 1111 1111 1111 1111 1111 0111 1011 *
L0320 1011 1111 1111 1111 0111 1011 0111 0111 *
L0352 1011 1111 1111 1111 1011 0111 0111 0111 *
L0512 1111 1111 1111 1111 1111 1111 1111 1110 *
L0544 1111 0111 1111 1111 1111 1111 0111 1011 *
L0576 1111 1011 1111 1111 0111 1011 0111 0111 *
L0608 1111 1011 1111 1111 1011 0111 0111 0111 *
L0768 1111 1111 1111 1111 1111 1111 1111 1110 *
L0800 1111 1111 0111 1111 1111 1111 0111 1011 *
L0832 1111 1111 1011 1111 0111 1011 0111 0111 *
L0864 1111 1111 1011 1111 1011 0111 0111 0111 *
L1024 1111 1111 1111 1111 1111 1111 1111 1110 *
L1056 1111 1111 0111 1111 1111 1111 0111 1011 *
L1088 1111 1111 1111 1011 0111 1011 0111 0111 *
L1120 1111 1111 1111 1011 1011 0111 0111 0111 *
L1280 1111 1111 1111 1111 1111 1111 1111 1110 *
L1312 1111 1111 0111 1111 1111 1111 0111 1011 *
L1344 1111 1111 1111 1011 0111 1011 0111 0111 *
L1376 1111 1111 1111 1011 1011 0111 0111 0111 *
L1536 1111 1111 1111 1111 1111 1111 1111 1110 *
L1568 1111 1111 0111 1111 1111 1111 0111 1011 *
L1600 1111 1111 1111 1011 0111 1011 0111 0111 *
L1632 1111 1111 1111 1011 1011 0111 0111 0111 *
L1792 1111 1111 1111 1111 1111 1111 1111 1110 *
L1824 1111 1111 0111 1111 1111 1111 0111 1011 *
L1856 1111 1111 1111 1011 0111 1011 0111 0111 *
L1888 1111 1111 1111 1011 1011 0111 0111 0111 *
C7A48*
V0001 XXXXXXX0000LLLLLLLL1 *
V0002 XXXXXXX0100LLLLLLLH1 *
V0003 0101XXX1000HHHHHLHL1 *
V0004 1010XXX1000LLLLLHLH1 *
V0005 11111011100LLLLLLLL1 *
V0006 01010011100HHHHLHLH1 *
V0007 10101101100LLLLHLHL1 *
V0008 00000101100HHHHHHHH1 *
88F1
```

LOGIC DIAGRAM FOR:
SBUS CONTROL A USING AmPAL16H8

= Fuse intact   = All fuses intact   = Fuse blown

03862A-163

```
PAL16H8                                 PAL DESIGN SPECIFICATION
PAT019                                  JEFF KITSON      10/4/82
SBUS CONTROL B
ADVANCED MICRO DEVICES
VR3 VR4  VR5  VR6  VR7  VR11 CBR S1 SO GND
/OE S15  S14  S13  S12  S11  S6  S5  S3  VCC

;SBUS CONTROL OUTPUT SIGNAL

IF (OE)  S3   = S1*/SO* VR3                    +
                S1* SO*/VR3* VR11*/CBR         +
                S1* SO*/VR3*/VR11* CBR

IF (OE)  S5   = S1*/SO* VR4                    +
                S1* SO*/VR5* VR11*/CBR         +
                S1* SO*/VR5*/VR11* CBR

IF (OE)  S6   = S1*/SO* VR4                    +
                S1* SO*/VR6* VR11*/CBR         +
                S1* SO*/VR6*/VR11* CBR

IF (OE)  S11  = S1*/SO* VR4                    +
                S1* SO*/VR7* VR11*/CBR         +
                S1* SO*/VR7*/VR11* CBR

IF (OE)  S12  = S1*/SO* VR4                    +
                S1* SO*/VR7* VR11*/CBR         +
                S1* SO*/VR7*/VR11* CBR

IF (OE)  S13  = S1*/SO* VR4                    +
                S1* SO*/VR7* VR11*/CBR         +
                S1* SO*/VR7*/VR11* CBR

IF (OE)  S14  = S1*/SO* VR4                    +
                S1* SO*/VR7* VR11*/CBR         +
                S1* SO*/VR7*/VR11* CBR

IF (OE)  S15  = S1*/SO* VR4                    +
                S1* SO*/VR7* VR11*/CBR         +
                S1* SO*/VR7*/VR11* CBR
```

FUNCTION TABLE

```
VR3 VR4 VR5 VR6 VR7 VR11 CBR S1 SO /OE S3  S5  S6  S11 S12 S13 S14 S15
--------------------------------------------------------------------------
;SELECT ZEROS
 X   X   X   X   X   X    X  L  L  L   L   L   L   L   L   L   L   L
;SELECT ONES
 X   X   X   X   X   X    X  L  H  L   L   L   L   L   L   L   L   L
;SELECT VR0 TO VR4
 L   H   L   H   X   X    X  H  L  L   L   H   H   H   H   H   H   H
 H   L   H   L   X   X    X  H  L  L   H   L   L   L   L   L   L   L
;SELECT CBR OFFSET OR ZEROS
 H   X   H   H   H   H    L  H  H  L   L   L   L   L   L   L   L   L
 L   X   H   L   H   H    L  H  H  L   H   L   H   L   L   L   L   L
 H   X   L   H   L   L    H  H  H  L   L   H   L   H   H   H   H   H
 L   X   L   L   L   L    H  H  H  L   H   H   H   H   H   H   H   H
--------------------------------------------------------------------------
```

DESCRIPTION
THE IMPLEMENTATION OF THE SBUS CONTROL REQUIRES TWO AMPAL16H8AS. THE
FUNCTIONS PERFORMED BY THE PALS ARE CONTROLLED BY MICROCODE INPUTS SBUS
SEL<1:0>. FOR IMMEDIATE DESTINATION OPERANDS, THESE PALS PERFORM SIGN
EXTENSION. IF A CBR INSTRUCTION IS SELECTED, A SIGN EXTENDED BRANCH
OFFSET IS DERIVED FROM VR<9:0>. VR<11> AND THE CBR FLAG INITIATE A
BRANCH WHEN THEY ARE THE SAME POLARITY, AT WHICH TIME THE OFFSET IS
ADDED TO THE PC. IF THE BRANCH CONDITION IS NOT MET, -1 IS ADDED TO
THE PC TO RESUME PROPER INSTRUCTION EXECUTION. THE SBUS CONTROL B PAL
CONTROLS SBUS<15:11,6:5,3>.

PAL16H8   PAL DESIGN SPECIFICATION
PAT019   JEFF KITSON   10/4/82
SBUS CONTROL B
ADVANCED MICRO DEVICES
*D9725
F0
L0000 1111 1111 1111 1111 1111 1111 1111 1110 *
L0032 1101 1111 1111 1111 1111 1111 0111 1011 *
L0064 1110 1111 1111 1111 0111 1011 0111 0111 *
L0096 1110 1111 1111 1111 1011 0111 0111 0111 *
L0256 1111 1111 1111 1111 1111 1111 1111 1110 *
L0288 0111 1111 1111 1111 1111 1111 0111 1011 *
L0320 1111 1011 1111 1111 0111 1011 0111 0111 *
L0352 1111 1011 1111 1111 1011 0111 0111 0111 *
L0512 1111 1111 1111 1111 1111 1111 1111 1110 *
L0544 0111 1111 1111 1111 1111 1111 0111 1011 *
L0576 1111 1111 1011 1111 0111 1011 0111 0111 *
L0608 1111 1111 1011 1111 1011 0111 0111 0111 *
L0768 1111 1111 1111 1111 1111 1111 1111 1110 *
L0800 0111 1111 1111 1111 1111 1111 0111 1011 *
L0832 1111 1111 1111 1011 0111 1011 0111 0111 *
L0864 1111 1111 1111 1011 1011 0111 0111 0111 *
L1024 1111 1111 1111 1111 1111 1111 1111 1110 *
L1056 0111 1111 1111 1111 1111 1111 0111 1011 *
L1088 1111 1111 1111 1011 0111 1011 0111 0111 *
L1120 1111 1111 1111 1011 1011 0111 0111 0111 *
L1280 1111 1111 1111 1111 1111 1111 1111 1110 *
L1312 0111 1111 1111 1111 1111 1111 0111 1011 *
L1344 1111 1111 1111 1011 0111 1011 0111 0111 *
L1376 1111 1111 1111 1011 1011 0111 0111 0111 *
L1536 1111 1111 1111 1111 1111 1111 1111 1110 *
L1568 0111 1111 1111 1111 1111 1111 0111 1011 *
L1600 1111 1111 1111 1011 0111 1011 0111 0111 *
L1632 1111 1111 1111 1011 1011 0111 0111 0111 *
L1792 1111 1111 1111 1111 1111 1111 1111 1110 *
L1824 0111 1111 1111 1111 1111 1111 0111 1011 *
L1856 1111 1111 1111 1011 0111 1011 0111 0111 *
L1888 1111 1111 1111 1011 1011 0111 0111 0111 *
C7631*
V0001 XXXXXXX0000LLLLLLLL1 *
V0002 XXXXXXX0100LLLLLLLL1 *
V0003 0101XXX1000HHHHHHHL1 *
V0004 1010XXX1000LLLLLLLH1 *
V0005 1X111101100LLLLLLLL1 *
V0006 0X101101100LLLLLHLH1 *
V0007 1X010011100HHHHHLHL1 *
V0008 0X000011100HHHHHHHH1 *
80F0

LOGIC DIAGRAM FOR:
SBUS CONTROL B USING AmPAL16H8

Columns: Inputs (0–31)
Rows: Product Terms (0–63)
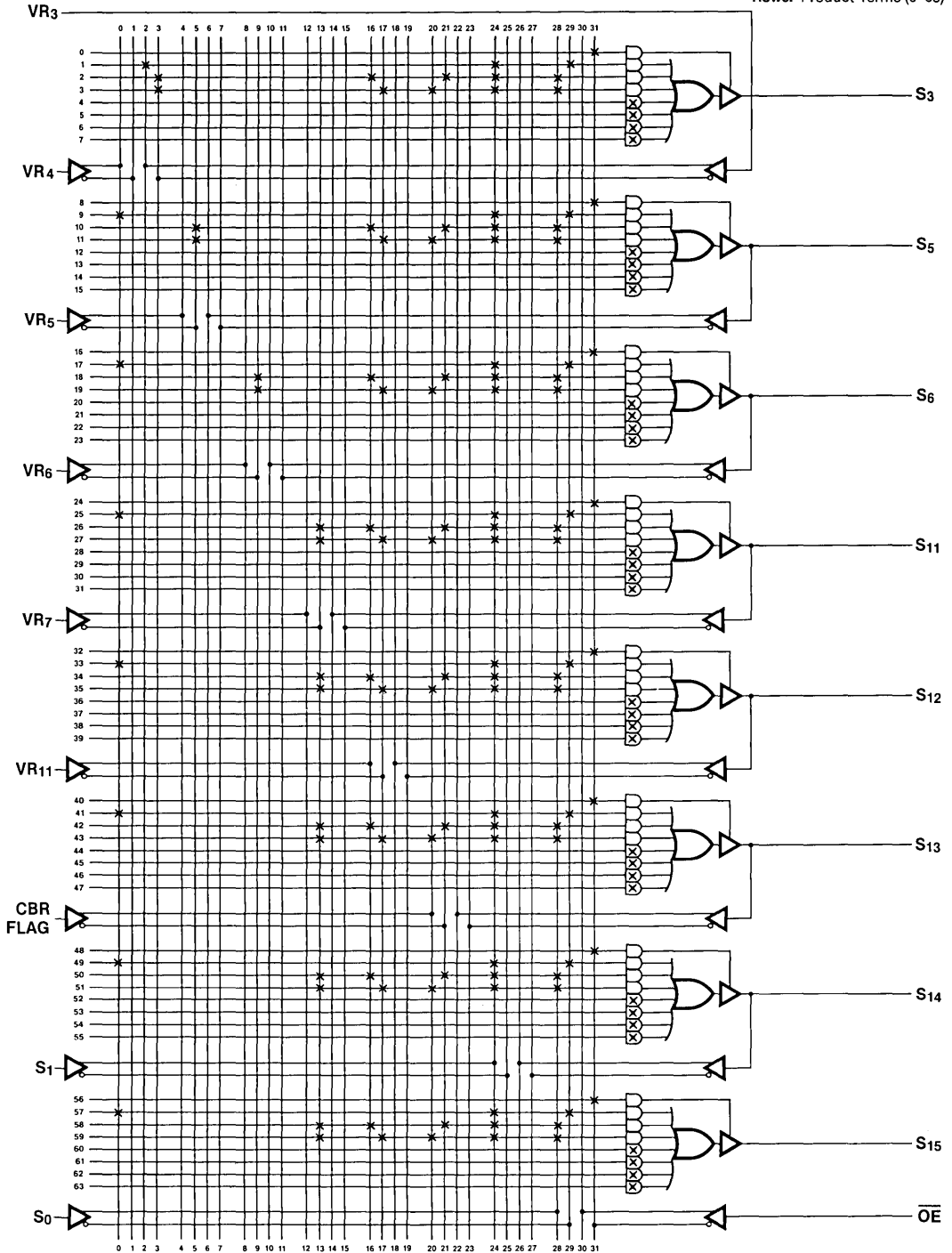
—✳— = Fuse intact   —⊠— = All fuses intact   —┼— = Fuse blown

03862A-164

```
PAL16R4                              PAL DESIGN SPECIFICATION
PAT020                               JENNY YEE       10/4/82
PROGRAM COUNTER A
ADVANCED MICRO DEVICES
 CLK  Y3  Y2  Y1  YO  CI  S1 SO /OEA GND
/OER  AO  A1  RO  R1  R2  R3  A2   A3  VCC

;PROGRAM COUNTER A OUTPUT

IF (OEA)  /A3    = /S1*/R3                        +
                   S1

IF (OEA)  /A2    = /S1*/R2                        +
                   S1

          /R3    :=  /SO*/S1*/Y3                  +
                     SO*/S1* R3* R2* R1* RO* CI   +
                     SO*/S1*/R3*/CI               +
                     SO*/S1*/R3*/R2               +
                     SO*/S1*/R3*/R1               +
                     SO*/S1*/R3*/RO               +
                        S1*/R3

          /R2    := /SO*/S1*/Y2                   +
                     SO*/S1* R2* R1* RO* CI       +
                     SO*/S1*/R2*/CI               +
                     SO*/S1*/R2*/R1               +
                     SO*/S1*/R2*/RO               +
                        S1*/R2

          /R1    := /SO*/S1*/Y1                   +
                     SO*/S1* R1* RO* CI           +
                     SO*/S1*/R1*/CI               +
                     SO*/S1*/R1*/RO               +
                        S1*/R1

          /RO    := /SO*/S1*/YO                   +
                     SO*/S1* RO* CI               +
                     SO*/S1*/RO*/CI               +
                        S1*/RO

IF (OEA)  /A1    = /S1*/R1              +
                   S1

IF (OEA)  /AO    = /S1*/RO              +
                   S1
```

FUNCTION TABLE

CLK	/OEA	/OER	S1	SO	Y3	Y2	Y1	YO	CI	A3	A2	A1	AO	R3	R2	R1	RO
;LOAD REGISTERS FROM YBUS AND OTPUT ONTO RBUS																	
C	H	L	L	L	L	H	L	H	X		Z	Z	Z	Z	L	H	L	H
;INCREMENT REGISTERS REG=REG+CI																	
C	H	L	L	H	X	X	X	X	L	Z	Z	Z	Z	L	H	L	H
C	H	L	L	H	X	X	X	X	H	Z	Z	Z	Z	L	H	H	L
C	H	L	L	H	X	X	X	X	H	Z	Z	Z	Z	L	H	H	H
C	H	L	L	H	X	X	X	X	H	Z	Z	Z	Z	H	L	L	L
C	H	L	L	H	X	X	X	X	H	Z	Z	Z	Z	H	L	L	H
C	H	L	L	H	X	X	X	X	H	Z	Z	Z	Z	H	L	H	L
C	H	L	L	H	X	X	X	X	H	Z	Z	Z	Z	H	L	H	H
C	H	L	L	H	X	X	X	X	H	Z	Z	Z	Z	H	H	L	L
C	H	L	L	H	X	X	X	X	H	Z	Z	Z	Z	H	H	L	H
C	H	L	L	H	X	X	X	X	H	Z	Z	Z	Z	H	H	H	L
C	H	L	L	H	X	X	X	X	H	Z	Z	Z	Z	H	H	H	H
C	H	L	L	H	X	X	X	X	H	Z	Z	Z	Z	L	L	L	L
C	H	L	L	H	X	X	X	X	H	Z	Z	Z	Z	L	L	L	H
C	H	L	L	H	X	X	X	X	H	Z	Z	Z	Z	L	L	H	L
C	H	L	L	H	X	X	X	X	H	Z	Z	Z	Z	L	L	H	H
C	H	L	L	H	X	X	X	X	H	Z	Z	Z	Z	L	H	L	L
C	H	L	L	H	X	X	X	X	H	Z	Z	Z	Z	L	H	L	H
;HOLD VALUE IN REGISTERS AND OUTPUT ONTO RBUS																	
C	H	L	H	X	X	X	X	X	X	Z	Z	Z	Z	L	H	L	H
;OUTPUT THE CURRENT VALUE ONTO THE ABUS AND NOT ONTO THE RBUS																	
X	L	H	L	X	X	X	X	X	X	L	H	L	H	Z	Z	Z	Z
;OUTPUT ZEROS ONTO THE ABUS																	
X	L	H	H	X	X	X	X	X	X	L	L	L	L	Z	Z	Z	Z

DESCRIPTION
THE PROGRAM COUNTER EXAMPLE USED HERE SHOWS HOW PALS CAN BE USED
IN THE DESIGN TO IMPLEMENT DATA STEERING. THE PC IS IMPLEMENTED
AS AN INCREMENTING REGISTER THAT CAN BE PARALLEL LOADED FROM THE
YBUS. THE PC IS USED TO SOURCE THE ABUS WITH MEMORY ADDRESSES
FOR INSTRUCTIONS, AND ALSO SOURCES THE RBUS FOR RELATIVE BRANCH
CALCULATIONS. PROGRAM COUNTER A GENERATES ZEROS IN THE 12 MOST
SIGNIFICANT BITS FOR TRAPS AND INTERRUPTS AS WELL. THESE 3
IDENTICALLY PROGRAMMED AMPAL16R4AS, CALLED PROGRAM COUNTER A,
ARE THE 3 MOST SIGNIFICANT SLICES OF THE 4-SLICE PROGRAM COUNTER.

*D9724
F0
L0000 1111 1111 1111 1111 1111 1111 1111 1011 *
L0032 1111 1111 1110 1111 1111 1011 1111 1111 *
L0064 1111 1111 1111 1111 1111 0111 1111 1111 *
L0256 1111 1111 1111 1111 1111 1111 1111 1011 *
L0288 1111 1111 1111 1110 1111 1011 1111 1111 *
L0320 1111 1111 1111 1111 1111 0111 1111 1111 *
L0512 1011 1111 1111 1111 1111 1011 1111 1111 *
L0544 1111 1111 1101 1101 0101 1001 0111 1111 *
L0576 1111 1111 1110 1111 1011 1011 0111 1111 *
L0608 1111 1111 1110 1110 1111 1011 0111 1111 *
L0640 1111 1111 1110 1111 1110 1011 0111 1111 *
L0672 1111 1111 1110 1111 1111 1010 0111 1111 *
L0704 1111 1111 1110 1111 1111 0111 1111 1111 *
L0768 1111 1011 1111 1111 1111 1011 1011 1111 *
L0800 1111 1111 1111 1101 0101 1001 0111 1111 *
L0832 1111 1111 1111 1110 1011 1011 0111 1111 *
L0864 1111 1111 1111 1110 1110 1011 0111 1111 *
L0896 1111 1111 1111 1110 1111 1010 0111 1111 *
L0928 1111 1111 1111 1110 1111 0111 1111 1111 *
L1024 1111 1111 1011 1111 1111 1011 1011 1111 *
L1056 1111 1111 1111 1111 0101 1001 0111 1111 *
L1088 1111 1111 1111 1111 1010 1011 0111 1111 *
L1120 1111 1111 1111 1111 1110 1010 0111 1111 *
L1152 1111 1111 1111 1111 1110 0111 1111 1111 *
L1280 1111 1111 1111 1011 1111 1011 1011 1111 *
L1312 1111 1111 1111 1111 0101 1001 0111 1111 *
L1344 1111 1111 1111 1111 1011 1010 0111 1111 *
L1376 1111 1111 1111 1111 1111 0110 1111 1111 *
L1536 1111 1111 1111 1111 1111 1111 1111 1011 *
L1568 1111 1111 1111 1111 1110 1011 1111 1111 *
L1600 1111 1111 1111 1111 1111 0111 1111 1111 *
L1792 1111 1111 1111 1111 1111 1111 1111 1011 *
L1824 1111 1111 1111 1111 1111 1010 1111 1111 *
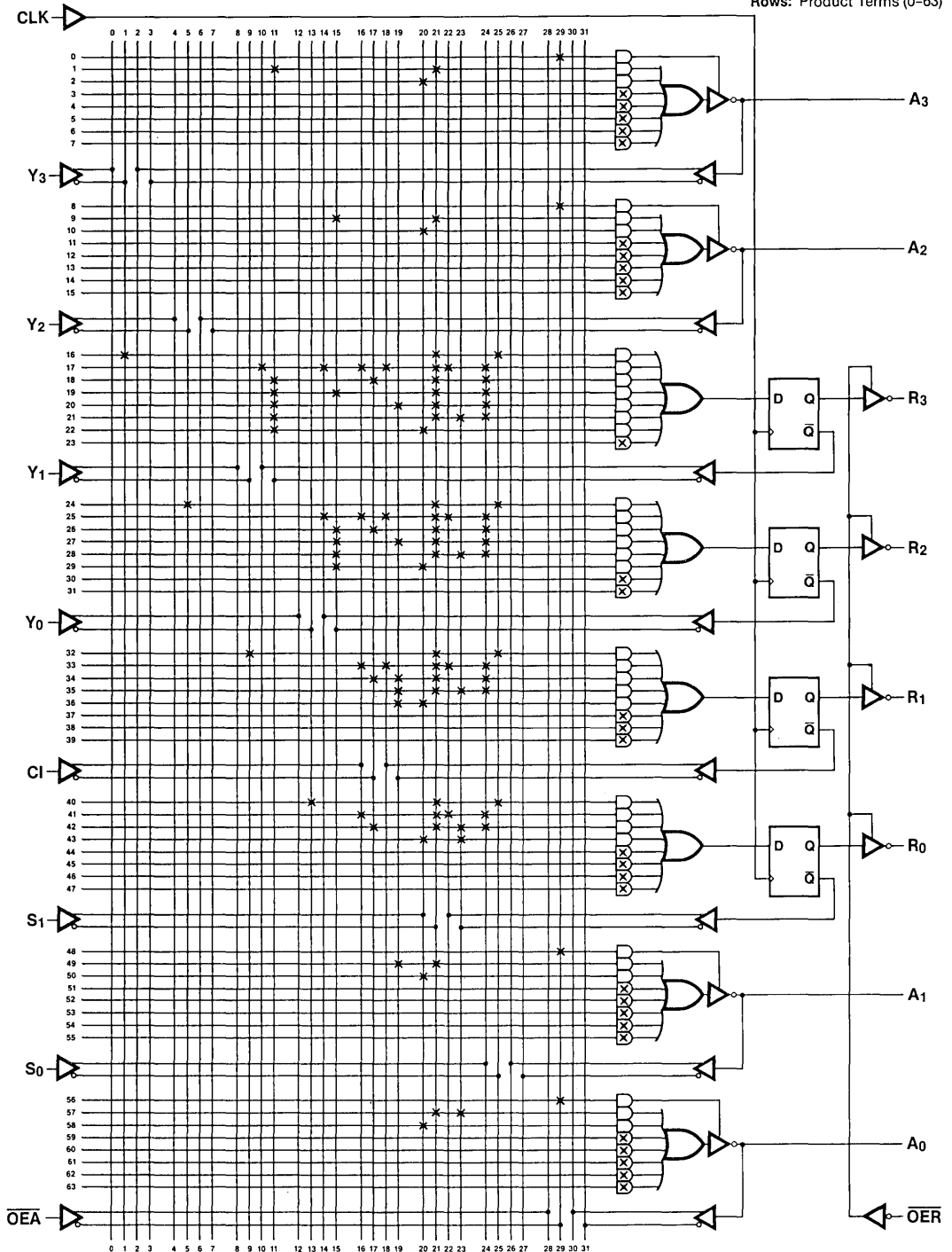L1856 1111 1111 1111 1111 1111 0111 1111 1111 *
C7B62*
V0001 C0101X00100ZZHLHLZZ1 *
V0002 CXXXX001100ZZHLHLZZ1 *
V0003 CXXXX101100ZZLHHLZZ1 *
V0004 CXXXX101100ZZHHHLZZ1 *
V0005 CXXXX101100ZZLLLHZZ1 *
V0006 CXXXX101100ZZHLLHZZ1 *
V0007 CXXXX101100ZZLHLHZZ1 *
V0008 CXXXX101100ZZHHLHZZ1 *
V0009 CXXXX101100ZZLLHHZZ1 *
V0010 CXXXX101100ZZHLHHZZ1 *
V0011 CXXXX101100ZZLHHHZZ1 *
V0012 CXXXX101100ZZHHHHZZ1 *
V0013 CXXXX101100ZZLLLLZZ1 *
V0014 CXXXX101100ZZHLLLZZ1 *
V0015 CXXXX101100ZZLHLLZZ1 *
V0016 CXXXX101100ZZHHLLZZ1 *
V0017 CXXXX101100ZZLLHLZZ1 *
V0018 CXXXX101100ZZHLHLZZ1 *
V0019 CXXXXX1X100ZZHLHLZZ1 *
V0020 XXXXXX0X001HLZZZZZHL1 *
V0021 XXXXXX1X001LLZZZZZLL1 *
F390

LOGIC DIAGRAM FOR:
PROGRAM COUNTER A USING AmPAL16R4A

Columns: Inputs (0–31)
Rows: Product Terms (0–63)

03862A-165

---✳--- = Fuse intact   ---⊗--- = All fuses intact   ---|--- = Fuse blown

```
PAL16R4                                  PAL DESIGN SPECIFICATION
PAT021                                   JEFF KITSON     10/4/82
PROGRAM COUNTER B
ADVANCED MICRO DEVICES
 CLK  Y3  Y2  Y1  Y0  CI  S1 S0 /OEA GND
/OER  A0  A1  R0  R1  R2  R3  A2   A3  VCC

; PROGRAM COUNTER B OUTPUT SIGNALS

IF (OEA)   /A3    = /S1*/R3                               +
                    S1

IF (OEA)   /A2    = /S1*/R2                               +
                    S1*/S0

           /R3    := /S0*/S1*/Y3                          +
                     S0*/S1* R3* R2* R1* R0* CI           +
                     S0*/S1*/R3*/CI                        +
                     S0*/S1*/R3*/R2                        +
                     S0*/S1*/R3*/R1                        +
                     S0*/S1*/R3*/R0                        +
                        S1*/R3

           /R2    := /S0*/S1*/Y2                          +
                     S0*/S1* R2* R1* R0* CI               +
                     S0*/S1*/R2*/CI                        +
                     S0*/S1*/R2*/R1                        +
                     S0*/S1*/R2*/R0                        +
                        S1*/R2

           /R1    := /S0*/S1*/Y1                          +
                     S0*/S1* R1* R0* CI                   +
                     S0*/S1*/R1*/CI                        +
                     S0*/S1*/R1*/R0                        +
                        S1*/R1

           /R0    := /S0*/S1*/Y0                          +
                     S0*/S1* R0* CI                       +
                     S0*/S1*/R0*/CI                        +
                        S1*/R0

IF (OEA) /A1      = /R1*/S1

IF (OEA) /A0      = /S1*/R0                    +
                    S1
```

FUNCTION TABLE

CLK	/OEA	/OER	S1	S0	Y3	Y2	Y1	Y0	CI	A3	A2	A1	A0	R3	R2	R1	R0
;LOAD REGISTERS FROM YBUS AND OUTPUT ONTO RBUS																	
C	H	L	L	L	L	H	L	H	X		Z	Z	Z	Z	L	H	L	H
;INCREMENT REGISTERS REG=REG + CI																	
C	H	L	L	H	X	X	X	X	L		Z	Z	Z	Z	L	H	L	H
C	H	L	L	H	X	X	X	X	H		Z	Z	Z	Z	L	H	H	L
C	H	L	L	H	X	X	X	X	H		Z	Z	Z	Z	L	H	H	H
C	H	L	L	H	X	X	X	X	H		Z	Z	Z	Z	H	L	L	L
C	H	L	L	H	X	X	X	X	H		Z	Z	Z	Z	H	L	L	H
C	H	L	L	H	X	X	X	X	H		Z	Z	Z	Z	H	L	H	L
C	H	L	L	H	X	X	X	X	H		Z	Z	Z	Z	H	L	H	H
C	H	L	L	H	X	X	X	X	H		Z	Z	Z	Z	H	H	L	L
C	H	L	L	H	X	X	X	X	H		Z	Z	Z	Z	H	H	L	H
C	H	L	L	H	X	X	X	X	H		Z	Z	Z	Z	H	H	H	L
C	H	L	L	H	X	X	X	X	H		Z	Z	Z	Z	H	H	H	H
C	H	L	L	H	X	X	X	X	H		Z	Z	Z	Z	L	L	L	L
C	H	L	L	H	X	X	X	X	H		Z	Z	Z	Z	L	L	L	H
C	H	L	L	H	X	X	X	X	H		Z	Z	Z	Z	L	L	H	L
C	H	L	L	H	X	X	X	X	H		Z	Z	Z	Z	L	L	H	H
C	H	L	L	H	X	X	X	X	H		Z	Z	Z	Z	L	H	L	L
C	H	L	L	H	X	X	X	X	H		Z	Z	Z	Z	L	H	L	H
;HOLD VALUE IN REGISTERS AND OTPUT ONTO RBUS																	
C	H	L	H	X	X	X	X	X	X		Z	Z	Z	Z	L	H	L	H
;OUTPUT THE CURRENT VALUE ONTO THE ABUS AND NOT ONTO THE RBUS																	
X	L	H	L	X	X	X	X	X	X		L	H	L	H	Z	Z	Z	Z
;OUTPUT TWO ONTO THE ABUS																	
X	L	H	H	L	X	X	X	X	X		L	L	H	L	Z	Z	Z	Z
;OUTPUT SIX ONTO THE ABUS																	
X	L	H	H	H	X	X	X	X	X		L	H	H	L	Z	Z	Z	Z

DESCRIPTION
PROGRAM COUNTER B IS THE LEAST SIGNIFICANT SLICE OF THE PROGRAM
COUNTER. THIS SLICE OF THE PROGRAM COUNTER IS FUNCTIONALLY THE
SAME AS PROGRAM COUNTER A. THE CONSTANTS TWO AND SIX ARE GENERATED
FOR TRAPS AND INTERRUPTS.

```
PAL16R4                                          PAL DESIGN SPECIFICATION
PAT021                                           JEFF KITSON     10/4/82
PROGRAM COUNTER B
ADVANCED MICRO DEVICES
*D9724
*F0*
L0000 1111 1111 1111 1111 1111 1111 1111 1011 *
L0032 1111 1111 1110 1111 1111 1011 1111 1111 *
L0064 1111 1111 1111 1111 1111 0111 1111 1111 *
L0256 1111 1111 1111 1111 1111 1111 1111 1011 *
L0288 1111 1111 1111 1110 1111 1011 1111 1111 *
L0320 1111 1111 1111 1111 1111 0111 1011 1111 *
L0512 1011 1111 1111 1111 1111 1011 1011 1111 *
L0544 1111 1111 1101 1101 0101 1001 0111 1111 *
L0576 1111 1111 1110 1111 1011 1011 0111 1111 *
L0608 1111 1111 1110 1110 1111 1011 0111 1111 *
L0640 1111 1111 1110 1111 1110 1011 0111 1111 *
L0672 1111 1111 1110 1111 1111 1010 0111 1111 *
L0704 1111 1111 1110 1111 1111 0111 1111 1111 *
L0768 1111 1011 1111 1111 1111 1011 1011 1111 *
L0800 1111 1111 1111 1101 0101 1001 0111 1111 *
L0832 1111 1111 1111 1110 1011 1011 0111 1111 *
L0864 1111 1111 1111 1110 1110 1011 0111 1111 *
L0896 1111 1111 1111 1110 1111 1010 0111 1111 *
L0928 1111 1111 1111 1110 1111 0111 1111 1111 *
L1024 1111 1111 1011 1111 1111 1011 1011 1111 *
L1056 1111 1111 1111 1111 0101 1001 0111 1111 *
L1088 1111 1111 1111 1111 1010 1011 0111 1111 *
L1120 1111 1111 1111 1111 1110 1010 0111 1111 *
L1152 1111 1111 1111 1111 1110 0111 1111 1111 *
L1280 1111 1111 1111 1011 1111 1011 1011 1111 *
L1312 1111 1111 1111 1111 0111 1001 0111 1111 *
L1344 1111 1111 1111 1111 1011 1010 0111 1111 *
L1376 1111 1111 1111 1111 1111 0110 1111 1111 *
L1536 1111 1111 1111 1111 1111 1111 1111 1011 *
L1568 1111 1111 1111 1111 1110 1011 1111 1111 *
L1792 1111 1111 1111 1111 1111 1111 1111 1011 *
L1824 1111 1111 1111 1111 1111 1010 1111 1111 *
L1856 1111 1111 1111 1111 1111 0111 1111 1111 *
C7774*
V0001 C0101X00100ZZHLHLZZ1 *
V0002 CXXXX001100ZZHLHLZZ1 *
V0003 CXXXX101100ZZLHHLZZ1 *
V0004 CXXXX101100ZZHHHLZZ1 *
V0005 CXXXX101100ZZLLLHZZ1 *
V0006 CXXXX101100ZZHLLHZZ1 *
V0007 CXXXX101100ZZLHLHZZ1 *
V0008 CXXXX101100ZZHHLHZZ1 *
V0009 CXXXX101100ZZLLHHZZ1 *
V0010 CXXXX101100ZZHLHHZZ1 *
V0011 CXXXX101100ZZLHHHZZ1 *
V0012 CXXXX101100ZZHHHHZZ1 *
V0013 CXXXX101100ZZLLLLZZ1 *
V0014 CXXXX101100ZZHLLLZZ1 *
V0015 CXXXX101100ZZLHLLZZ1 *
V0016 CXXXX101100ZZHHLLZZ1 *
V0017 CXXXX101100ZZLLHLZZ1 *
V0018 CXXXX101100ZZHLHLZZ1 *
V0019 CXXXXX1X100ZZHLHLZZ1 *
V0020 XXXXXX0X001HLZZZZZHL1 *
V0021 XXXXXX10001LHZZZZZLL1 *
V0022 XXXXXX11001LHZZZZZHL1 *
F275
```
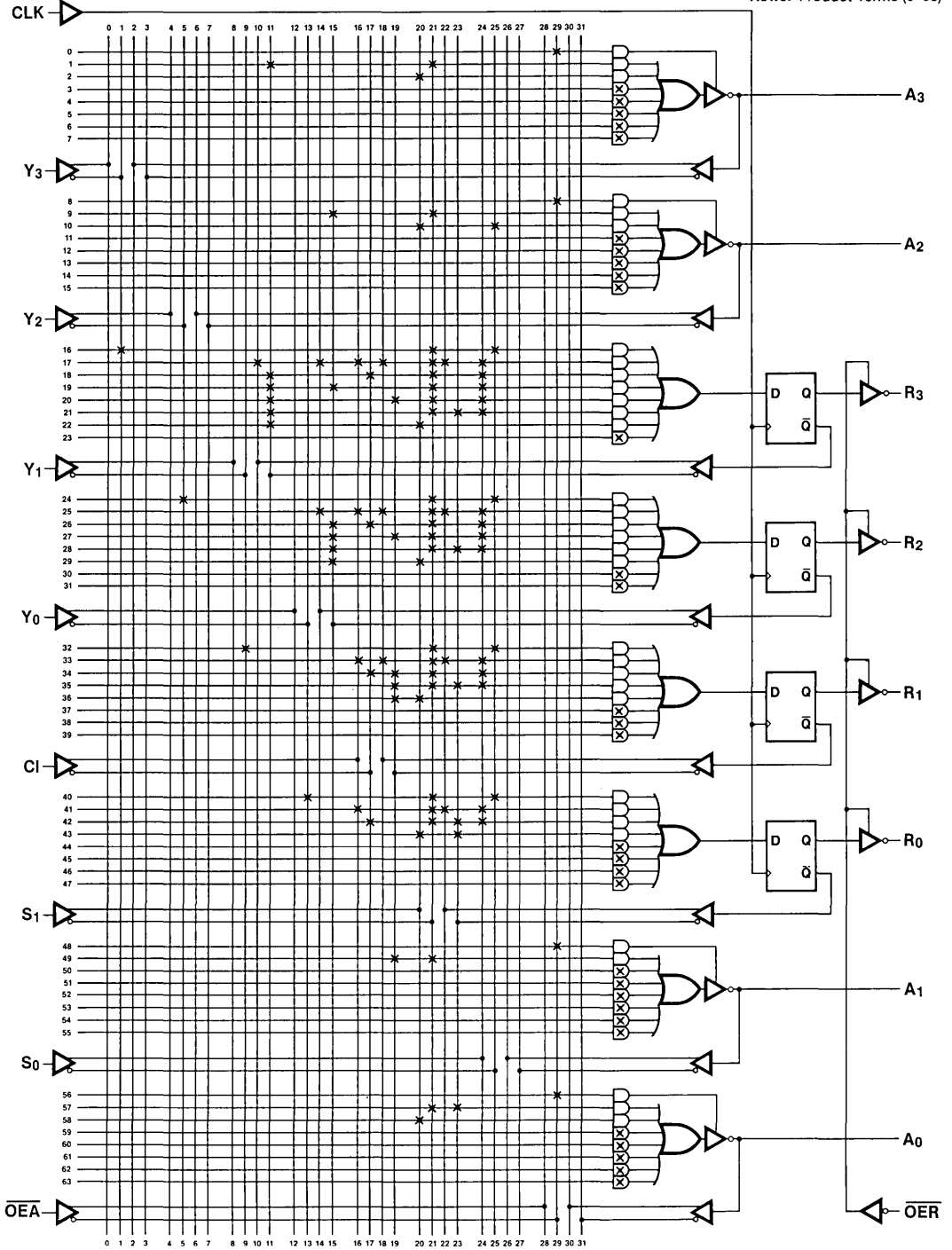
LOGIC DIAGRAM FOR:
PROGRAM COUNTER B USING AmPAL16R4A

Columns: Inputs (0–31)
Rows: Product Terms (0–63)

$-\!\!\!*\!\!\!-$ = Fuse intact $-\!\!\boxtimes\!\!-$ = All fuses intact $-\!\!+\!\!-$ = Fuse blown

03862A-166

PAT015   JEFF KITSON   10/4/82
PC CARRY GENERATOR
ADVANCED MICRO DEVICES
AO A1 A2 A3 A4 A5 A6 A7 A8 GND
A9 CO1 CO2 CO3 A10 A11 SL1 CI NC VCC

; PC CARRY GENERATOR OUTPUT SIGNALS

CO3   = SL1*CI   +
            /SL1*CI*AO*A1*A2*A3*A4*A5*A6*A7*A8*A9*A10*A11

CO2   = SL1*CI   +
            /SL1*CI*AO*A1*A2*A3*A4*A5*A6*A7

CO1   = SL1*CI   +
            /SL1*CI*AO*A1*A2*A3

FUNCTION TABLE

AO	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10	A11	CI	SL1	CO1	CO2	CO3
X	X	X	X	X	X	X	X	X	X	X	X	H	H	H	H	H
H	H	H	H	H	H	H	H	H	H	H	H	H	L	H	H	H
H	H	H	H	H	H	H	H	L	L	L	L	H	L	H	H	L
H	H	H	H	L	L	L	L	H	H	H	H	H	L	H	L	L
L	L	L	L	H	H	H	H	H	H	H	H	H	L	L	L	L

DESCRIPTION
THE 4-BIT OUTPUTS FROM THE 3 LEAST SIGNIFICANT SLICES OF THE 4-
SLICE PC ARE USED TO CREATE A 12-BIT INPUT TO THE PC CARRY
GENERATOR. THE PC CARRY GENERATOR TAKES THESE INPUTS AND
GENERATES THE CARRY-IN FOR THE 3 MOST SIGNIFICANT SLICES OF THE
PROGRAM COUNTER. THE PC CARRY GENERATOR IS CONTROLLED BY THE
PC SELECT INPUT FROM MICROCODE.

```
PAL16H8                                PAL DESIGN SPECIFICATION
PAT015                                 JEFF KITSON    10/4/82
PC CARRY GENERATOR
ADVANCED MICRO DEVICES
*D9725
*F0*
L1280 1111 1111 1111 1111 1111 1111 1111 1111 *
L1312 1111 1101 1101 1111 1111 1111 1111 1111 *
L1344 0101 0101 0110 0101 0101 0111 0111 0101 *
L1536 1111 1111 1111 1111 1111 1111 1111 1111 *
L1568 1111 1101 1101 1111 1111 1111 1111 1111 *
L1600 0101 0101 0110 0111 0111 0111 0111 1111 *
L1792 1111 1111 1111 1111 1111 1111 1111 1111 *
L1824 1111 1101 1101 1111 1111 1111 1111 1111 *
L1856 0101 0101 0110 1111 1111 1111 1111 1111 *
C211E*
V0001 XXXXXXXXXOXHHHXX11X1 *
V0002 11111111101HHH1101X1 *
V0003 11111111000HHL0001X1 *
V0004 11110000101HLL1101X1 *
V0005 00001111101LLL1101X1 *
A5CF
```
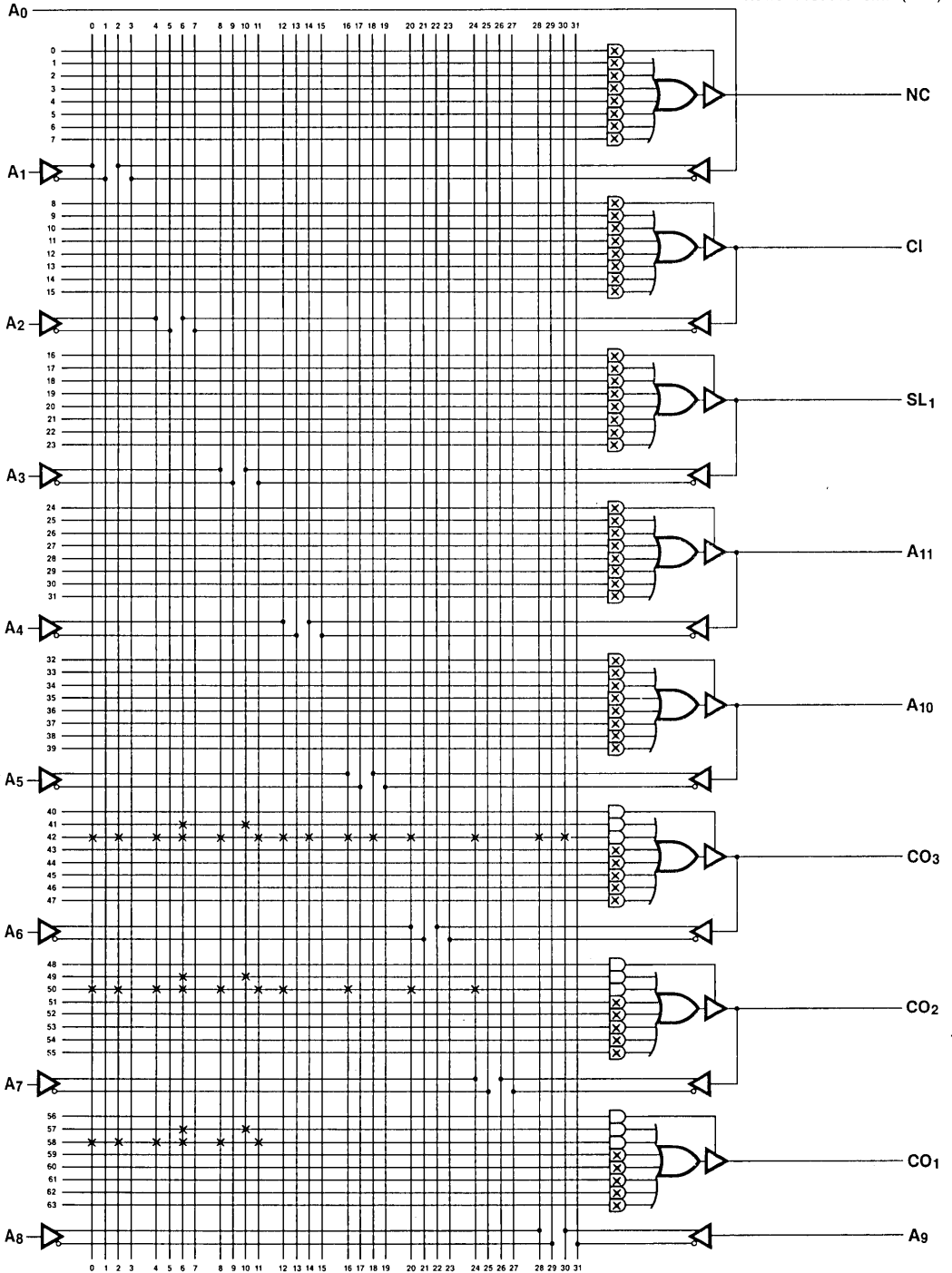
LOGIC DIAGRAM FOR:
PC CARRY GENERATOR USING AmPAL16H8

Columns: Inputs (0–31)
Rows: Product Terms (0–63)

03862A-167

= Fuse intact   = All fuses intact   = Fuse blown

5-162

```
PAL16R4                                 PAL DESIGN SPECIFICATION
PAT010                                  JENNY YEE      10/4/82
MEMORY ADDRESS REGISTER
ADVANCED MICRO DEVICES
 CLK Y3  Y2  Y1  YO  S1  SO  A3 /BI  GND
/OE /BO  AO  BO  B1  B2  B3  A1  A2  VCC

;MAR OUTPUT SIGNALS

/B3        := /S1*/SO*/B3                     +    ;HOLD
              /S1* SO*/Y3                      +    ;LOAD YBUS INTO MAR
              S1*/SO*/A3                       +    ;LOAD ABUS INTO MAR
              S1* SO* B3*/B2*/B1*/BO*BI        +    ;DECREMENT
              S1* SO*/B3* B2                   +
              S1* SO*/B3* B1                   +
              S1* SO*/B3* BO                   +
              S1* SO*/B3*/BI


/B2        := /S1*/SO*/B2                     +
              /S1* SO*/Y2                      +
              S1*/SO*/A2                       +
              S1* SO* B2*/B1*/BO*BI            +
              S1* SO*/B2* B1                   +
              S1* SO*/B2* BO                   +
              S1* SO*/B2*/BI


/B1        := /S1*/SO*/B1                     +
              /S1* SO*/Y1                      +
              S1*/SO*/A1                       +
              S1* SO* B1*/BO*BI                +
              S1* SO*/B1* BO                   +
              S1* SO*/B1*/BI


/BO        := /S1*/SO*/BO                     +
              /S1* SO*/YO                      +
              S1*/SO*/AO                       +
              S1* SO* BO* BI                   +
              S1* SO*/BO*/BI

 BO         = /B3*/B2*/B1*/BO*BI
```

FUNCTION TABLE

CLK	/OE	S1	S0	Y3	Y2	Y1	Y0	A3	A2	A1	A0	/BI		/BO	B3	B2	B1	B0
;LOAD REGISTERS FROM YBUS AND OUTPUT ONTO B																		
C	L	L	H	L	H	L	H	X	X	X	X	X		X	L	H	L	H
;LOAD REGISTERS FROM ABUS AND OUTPUT ONTO B																		
C	L	H	L	X	X	X	X	H	L	H	L	X		X	H	L	H	L
;HOLD THIS VALUE AND OUTPUT ONTO B																		
C	L	L	L	X	X	X	X	X	X	X	X	X		X	H	L	H	L
;DECREMENT REGISTERS WHERE R= R - BI																		
C	L	H	H	X	X	X	X	X	X	X	X	H		H	H	L	H	L
C	L	H	H	X	X	X	X	X	X	X	X	L		H	H	L	L	H
C	L	H	H	X	X	X	X	X	X	X	X	L		H	H	L	L	L
C	L	H	H	X	X	X	X	X	X	X	X	L		H	L	H	H	H
C	L	H	H	X	X	X	X	X	X	X	X	L		H	L	H	H	L
C	L	H	H	X	X	X	X	X	X	X	X	L		H	L	H	L	H
C	L	H	H	X	X	X	X	X	X	X	X	L		H	L	H	L	L
C	L	H	H	X	X	X	X	X	X	X	X	L		H	L	L	H	H
C	L	H	H	X	X	X	X	X	X	X	X	L		H	L	L	H	L
C	L	H	H	X	X	X	X	X	X	X	X	L		H	L	L	L	H
C	L	H	H	X	X	X	X	X	X	X	X	L		L	L	L	L	L
C	L	H	H	X	X	X	X	X	X	X	X	L		H	H	H	H	H
C	L	H	H	X	X	X	X	X	X	X	X	L		H	H	H	H	L
C	L	H	H	X	X	X	X	X	X	X	X	L		H	H	H	L	H
C	L	H	H	X	X	X	X	X	X	X	X	L		H	H	H	L	L
C	L	H	H	X	X	X	X	X	X	X	X	L		H	H	L	H	H
C	L	H	H	X	X	X	X	X	X	X	X	L		H	H	L	H	L

DESCRIPTION
THE MEMORY ADDRESS REGISTER (MAR) IS USED FOR PLACING ADDRESSES ON
THE MEMORY ADDRESS BUS WHEN FETCHING OPERANDS FROM MEMORY FOR
SOURCE AND DESTINATION. IT IS ALSO USED TO HOLD THE MEMORY
DESTINATION ADDRESS FOR STORING RESULTS FROM AN OPERATION. THE MAR
HAS THE CAPABILITY TO DECREMENT TO SUPPORT THE SPECIAL LOAD MEMORY
AND STORE MEMORY INSTRUCTIONS. THESE MAR OPERATIONS ARE CONTROLLED
BY MAR SEL<1:0>. THE MAR IS IMPLEMENTED USING FOUR AMPAL16R4AS.

```
PAL16R4                                    PAL DESIGN SPECIFICATION
PAT010                                     JENNY YEE      10/4/82
MEMORY ADDRESS REGISTER
ADVANCED MICRO DEVICES
*D9724
*F0*
L0512 1111 1111 1110 1111 1011 1011 1111 1111 *
L0544 1011 1111 1111 1111 1011 0111 1111 1111 *
L0576 1111 1111 1111 1111 0111 1011 1011 1111 *
L0608 1111 1111 1101 1110 0110 0110 1111 1011 *
L0640 1111 1111 1110 1101 0111 0111 1111 1111 *
L0672 1111 1111 1110 1111 0101 0111 1111 1111 *
L0704 1111 1111 1110 1111 0111 0101 1111 1111 *
L0736 1111 1111 1110 1111 0111 0111 1111 0111 *
L0768 1111 1111 1111 1110 1011 1011 1111 1111 *
L0800 1111 1011 1111 1111 1011 0111 1111 1111 *
L0832 1110 1111 1111 1111 0111 1011 1111 1111 *
L0864 1111 1111 1111 1101 0110 0110 1111 1011 *
L0896 1111 1111 1111 1110 0101 0111 1111 1111 *
L0928 1111 1111 1111 1110 0111 0101 1111 1111 *
L0960 1111 1111 1111 1110 0111 0111 1111 0111 *
L1024 1111 1111 1111 1111 1010 1011 1111 1111 *
L1056 1111 1111 1011 1111 1011 0111 1111 1111 *
L1088 1111 1110 1111 1111 0111 1011 1111 1111 *
L1120 1111 1111 1111 1111 0101 0110 1111 1011 *
L1152 1111 1111 1111 1111 0110 0101 1111 1111 *
L1184 1111 1111 1111 1111 0110 0111 1111 0111 *
L1280 1111 1111 1111 1111 1011 1010 1111 1111 *
L1312 1111 1111 1111 1011 1011 0111 1111 1111 *
L1344 1111 1111 1111 1111 0111 1011 1110 1111 *
L1376 1111 1111 1111 1111 0111 0101 1111 1011 *
L1408 1111 1111 1111 1111 0111 0110 1111 0111 *
L1792 1111 1111 1111 1111 1111 1111 1111 1111 *
L1824 1111 1111 1110 1110 1110 1110 1111 1011 *
C63A8*
V0001 C010101XX00XXHLHLXX1 *
V0002 CXXXX101X00X0LHLH101 *
V0003 CXXXX00XX00XXLHLHXX1 *
V0004 CXXXX11X100HXLHLHXX1 *
V0005 CXXXX11X000HXHLLHXX1 *
V0006 CXXXX11X000HXLLLHXX1 *
V0007 CXXXX11X000HXHHHLXX1 *
V0008 CXXXX11X000HXLHHLXX1 *
V0009 CXXXX11X000HXHLHLXX1 *
V0010 CXXXX11X000HXLLHLXX1 *
V0011 CXXXX11X000HXHHLLXX1 *
V0012 CXXXX11X000HXLHLLXX1 *
V0013 CXXXX11X000HXHLLLXX1 *
V0014 CXXXX11X000LXLLLLXX1 *
V0015 CXXXX11X000HXHHHHXX1 *
V0016 CXXXX11X000HXLHHHXX1 *
V0017 CXXXX11X000HXHLHHXX1 *
V0018 CXXXX11X000HXLLHHXX1 *
V0019 CXXXX11X000HXHHLHXX1 *
V0020 CXXXX11X000HXLHLHXX1 *
B9E4
```

LOGIC DIAGRAM FOR:
MAR USING AmPAL16R4A

Columns: Inputs (0–31)
Rows: Product Terms (0–63)

-✳- = Fuse intact   -⊠- = All fuses intact   -┼- = Fuse blown

03862A-168

5-166
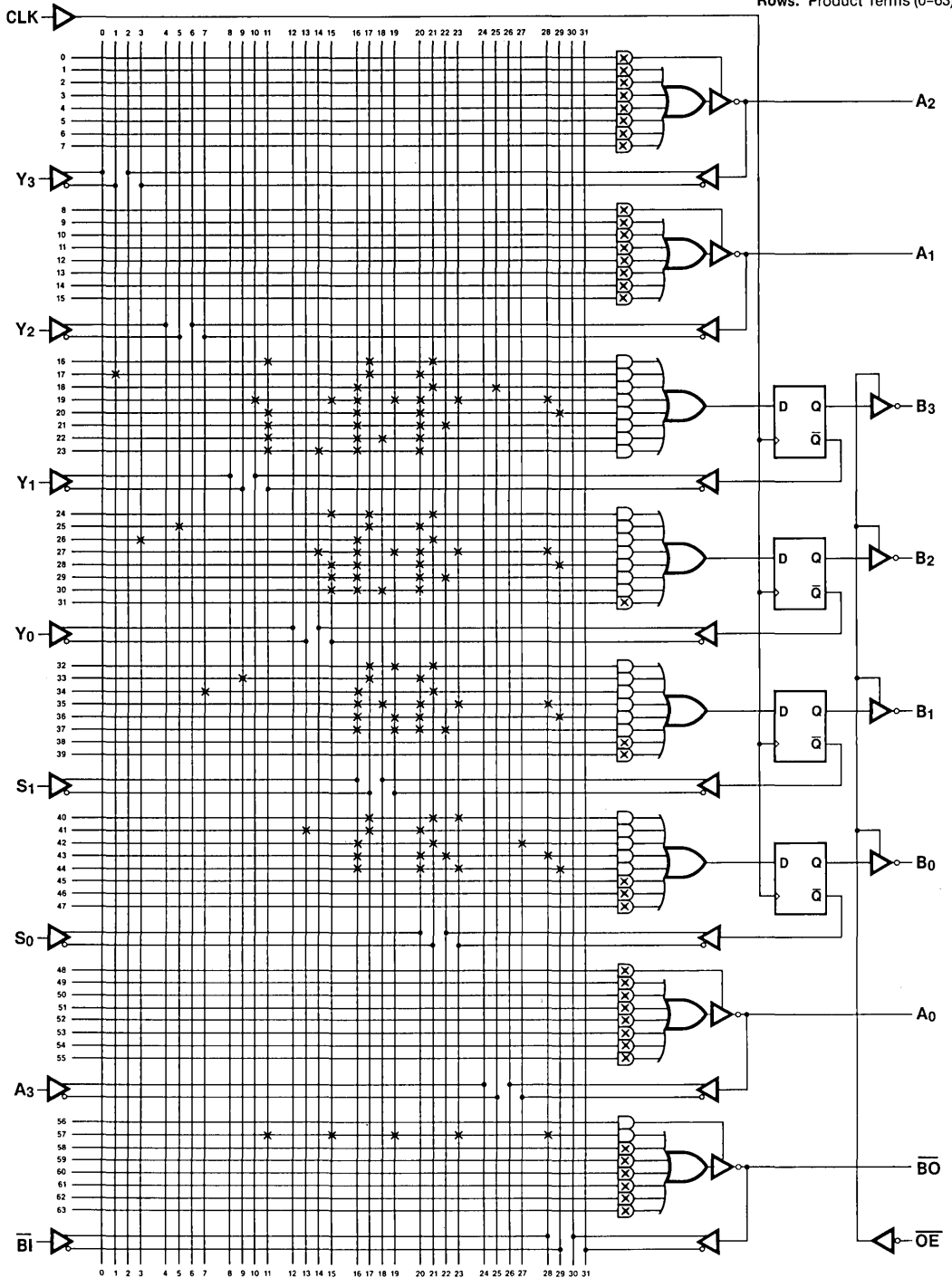
```
PAL16R4                               PAL DESIGN SPECIFICATION
PATO11                                JENNY YEE         10/4/82
MEMORY DATA REGISTER (1)
ADVANCED MICRO DEVICES
 CLK  Y3  Y2  Y1  YO  NC /CE  S  /OED GND
/OER /DO /D1 /RO /R1 /R2 /R3 /D2 /D3  VCC

;MDR OUTPUT SIGNALS

IF (OED)  D3     =  R3
IF (OED)  D2     =  R2

  R3             :=  /CE*R3       +  ;HOLD
                     CE* S* D3    +  ;LOAD FROM DBUS
                     CE*/S*/Y3       ;LOAD FROM YBUS

  R2             :=  /CE*R2       +
                     CE* S* D2    +
                     CE*/S*/Y2

  R1             :=  /CE*R1       +
                     CE* S* D1    +
                     CE*/S*/Y1

  RO             := /CE*RO        +
                     CE* S* DO    +
                     CE*/S*/YO

IF (OED)  D1     =  R1
IF (OED)  DO     =  RO
```

FUNCTION TABLE

CLK	OED	OER	S	Y3	Y2	Y1	YO	/CE	/D3	/D2	/D1	/DO	/R3	/R2	/R1	/RO
;LOAD REGISTERS FROM YBUS AND OUTPUT ONTO RBUS																
C	L	H	L	L	H	L	H	L	Z	Z	Z	Z	L	H	L	H
;																
;LOAD REGISTERS FROM DBUS AND OUTPUT ONTO RBUS																
C	L	H	H	X	X	X	X	L	H	L	H	L	H	L	H	L
;																
;HOLD VALUE AND OUTPUT ONTO RBUS																
C	L	H	X	X	X	X	X	H	Z	Z	Z	Z	H	L	H	L
;																
;OUTPUT VALUE ONTO DBUS AND NOT ONTO RBUS																
C	H	H	X	X	X	X	X	H	H	L	H	L	H	L	H	L

DESCRIPTION
THE MEMORY DATA REGISTER 1 (MDR1) IS AN I/O REGISTER WITH MULTIPLE
INPUT AND OUTPUT PATHS TO FACILITATE READING AND WRITING DATA
FROM MEMORY. IT MAY BE LOADED FROM EITHER THE YBUS (WRITE) OR
DBUS (READ) AND OUTPUT ON THE RBUS (READ) OR DBUS (WRITE). IT IS
IMPLEMENTED USING FOUR AMPAL16R4AS AND CONTROLLED FROM MICROCODE.

```
PAL16R4                              PAL DESIGN SPECIFICATION
PAT011                               JENNY YEE        10/4/82
MEMORY DATA REGISTER (1)
ADVANCED MICRO DEVICES
*D9724
*F0*
L0000 1111 1111 1111 1111 1111 1111 1111 1011 *
L0032 1111 1111 1110 1111 1111 1111 1111 1111 *
L0256 1111 1111 1111 1111 1111 1111 1111 1011 *
L0288 1111 1111 1111 1110 1111 1111 1111 1111 *
L0512 1111 1111 1110 1111 1111 0111 1111 1111 *
L0544 1110 1111 1111 1111 1111 1011 0111 1111 *
L0576 1011 1111 1111 1111 1111 1011 1011 1111 *
L0768 1111 1111 1111 1110 1111 0111 1111 1111 *
L0800 1111 1110 1111 1111 1111 1011 0111 1111 *
L0832 1111 1011 1111 1111 1111 1011 1011 1111 *
L1024 1111 1111 1111 1111 1110 0111 1111 1111 *
L1056 1111 1111 1111 1111 1111 1011 0110 1111 *
L1088 1111 1111 1011 1111 1111 1011 1011 1111 *
L1280 1111 1111 1111 1111 1111 0110 1111 1111 *
L1312 1111 1111 1111 1111 1111 1011 0111 1110 *
L1344 1111 1111 1111 1011 1111 1011 1011 1111 *
L1536 1111 1111 1111 1111 1111 1111 1111 1011 *
L1568 1111 1111 1111 1111 1110 1111 1111 1111 *
L1792 1111 1111 1111 1111 1111 1111 1111 1011 *
L1824 1111 1111 1111 1111 1111 1110 1111 1111 *
C4A70*
V0001 C0101X00100ZZHLHLZZ1 *
V0002 CXXXXX0110001LHLHO11 *
V0003 CXXXXX1X100ZZLHLHZZ1 *
V0004 CXXXXX1X000LHLHLHLH1 *
015E
```

LOGIC DIAGRAM FOR:
MDR₁ USING AmPAL16R4A

Columns: Inputs (0–31)
Rows: Product Terms (0–63)

$\overline{D}_3$
$\overline{D}_2$
$\overline{R}_3$
$\overline{R}_2$
$\overline{R}_1$
$\overline{R}_0$
$\overline{D}_1$
$\overline{D}_0$
$\overline{OER}$

CLK
Y₃
Y₂
Y₁
Y₀
NC
$\overline{CE}$
S
$\overline{OED}$
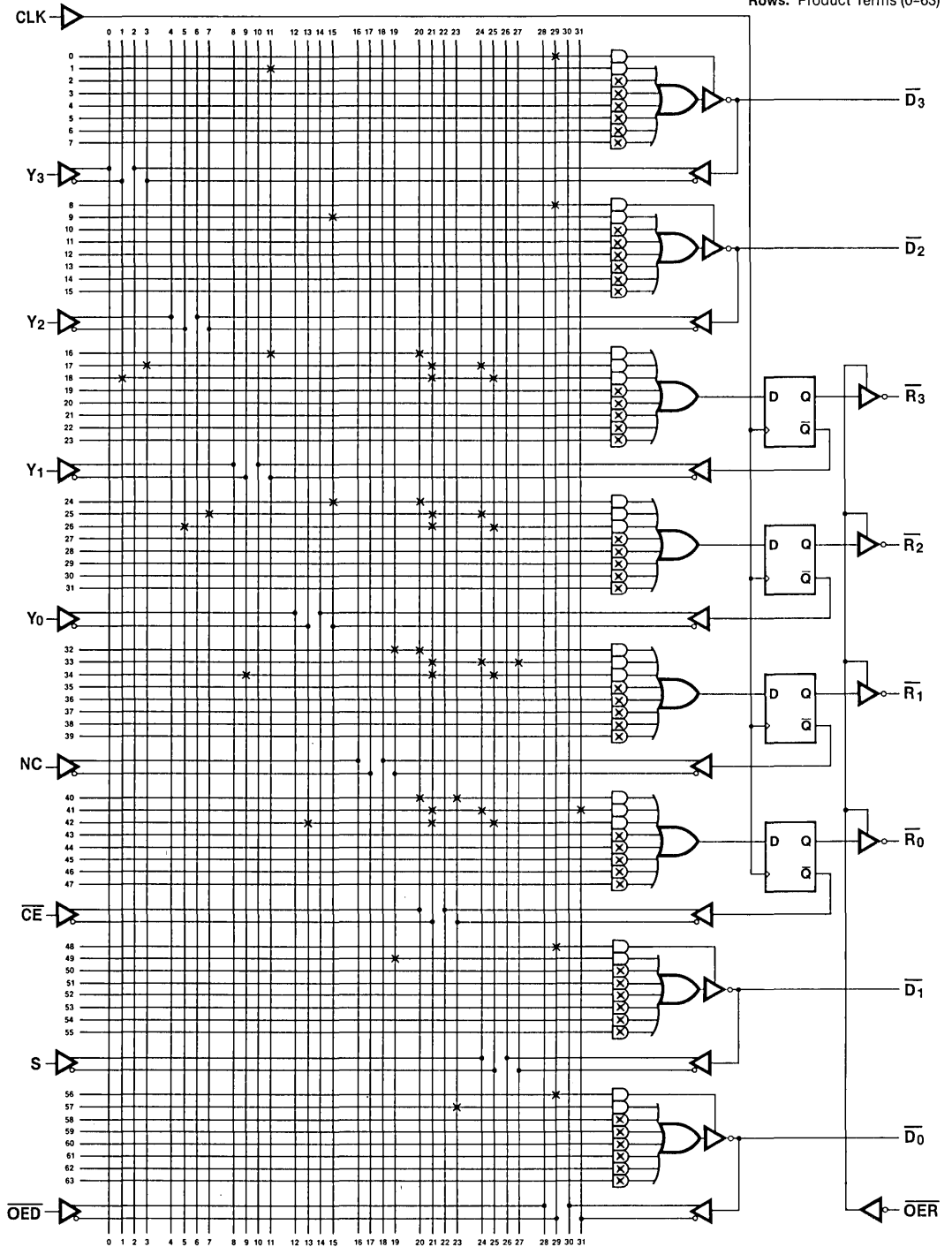
---*--- = Fuse intact   ---⊠--- = All fuses intact   ---|--- = Fuse blown

03862A-169

5-170

```
PAL16R6                                    PAL DESIGN SPECIFICATION
PAT018                                     KEVIN OW-WING   1-18-83
MICROBRANCH CONTROL
ADVANCED MICRO DEVICES

CLK  S2   S1    SO    DR0    DR1   DR2    INT   LDINT   GND
/OE VRO INTENB /OEVEC /OEJMP /OEMAP2 /OEMAP1 INTFLAG CXV VCC

;MICROBRANCH EQUATIONS

/INTFLAG        :=  /INT

 OEMAP1         :=  /S2*/S1*/SO                    +
                    /S2* S1* SO*/INTENB            +
                    /S2* S1* SO*/INTFLAG

 OEMAP2         :=  /S2*/S1* SO

 OEJMP          :=  /S2* S1*/SO                    +
                    S2*/S1*/SO* DR0                +
                    S2*/S1* DR1                    +
                    S2*/S1* DR2                    +
                    S2*/S1* SO*/DR0                +
                    S2* S1*/SO*/CXV

 OEVEC          := /S2* S1* SO* INTFLAG* INTENB    +
                    S2*/S1*/SO*/DR0*/DR1*/DR2      +
                    S2*/S1* SO* DR0*/DR1*/DR2      +
                    S2* S1*/SO* CXV

/INTENB         := LDINT*/VRO                      +
                   /LDINT*/INTENB
```

```
FUNCTION TABLE
CLK /OE  S2  S1 SO  DR2 DR1 DRO INT LDINT VRO CXV
INTFLAG    INTENB    OEMAP1    OEMAP2    OEJMP    OEVEC


;                                    I
;                                    N I O O
;                    L               T N E E O O
;                    D               F T M M E E
;C /       D D D I I V C             L E A A J V
;L O S S S R R R N N R X             A N P P M E
;K E 2 1 0 2 1 0 T T O V             G B 1 2 P C
---------------------------------------------------
;UNCONDTIONAL BRANCH
 C L L L L X X X X X X             X X H L L L
 C L L L H X X X X X X             X X L H L L
 C L L H L X X X X X X             X X L L H L
;SET INTENB AND INTFLAG
 C L X X X X X H H H X             H H X X X X
;TEST FOR INTERRUPTS
 C L L H H X X X L X X X           L H L L L H
 C L L H H X X X H H L X           H L H L L L
 C L L H H X X X X X X X           X X H L L L
;DECREMENT COUNTER = 0
 C L H L L L L L X X X X           X X L L L H
 C L H L L L L H X X X X           X X L L H L
 C L H L L X H X X X X X           X X L L H L
 C L H L L H X X X X X X           X X L L H L
;DECREMENT COUNTER = 1
 C L H L H L L H X X X X           X X L L L H
 C L H L H X X L X X X X           X X L L H L
 C L H L H X H H X X X X           X X L L H L
 C L H L H H X H X X X X           X X L L H L
;TEST FOR CONDITIONAL BRANCH
 C L H H L X X X X X X L           X X L L H L
 C L H H L X X X X X X H           X X L L L H
---------------------------------------------------


DESCRIPTION
THE MICROBRANCH PAL, AN AMPAL16R6A, IS USED TO CONTROL
BRANCHING IN THE MICROSEQUENCER.  IT OPERATES BY
SELECTING THE PROPER BRANCH CONDITION (UNCONDITIONAL,
INTERRUPT, COUNTER=0, COUNTER=1, CBR FLAG), TESTING THE
CONDITION, AND THEN ACTIVATING ONE OF FOUR POSSIBLE
ADDRESS SOURCE OUTPUT ENABLES DEPENDENT UPON THE STATUS
OF THE CONDITION TESTED.  THE SELECT INPUTS ARE FROM
MICROCODE PROM AND THE OUTPUT ENABLES GENERATED BY THE
MICROBRANCH PAL ARE ACTUALLY PART OF THE MICROCODE
PIPELINE REGISTER.
NOTE ALSO THAT INTERRUPT SYNCHRONIZATION AND CONTROL
ARE ALSO INTEGRATED INTO THE MICROBRANCH PAL.
```

```
PAL16R6                                      PAL DESIGN SPECIFICATION
PAT018                                       KEVIN OW-WING    1-18-83
MICROBRANCH CONTROL
ADVANCED MICRO DEVICES
*D9724
*F0*
L0256 1111 1111 1111 1111 1111 1111 1011 1111 *
L0512 1011 1011 1011 1111 1111 1111 1111 1111 *
L0544 1011 0111 0111 1111 1111 1111 1110 1111 *
L0576 1011 0110 0111 1111 1111 1111 1111 1111 *
L0768 1011 1011 0111 1111 1111 1111 1111 1111 *
L1024 1011 0111 1011 1111 1111 1111 1111 1111 *
L1056 0111 1011 1011 0111 1111 1111 1111 1111 *
L1088 0111 1011 1111 1111 0111 1111 1111 1111 *
L1120 0111 1011 1111 1111 1111 0111 1111 1111 *
L1152 0111 1011 0111 1011 1111 1111 1111 1111 *
L1184 0110 0111 1011 1111 1111 1111 1111 1111 *
L1280 1011 0101 0111 1111 1111 1111 1101 1111 *
L1312 0111 1011 1011 1011 1011 1011 1111 1111 *
L1344 0111 1011 0111 0111 1011 1011 1111 1111 *
L1376 0101 0111 1011 1111 1111 1111 1111 1111 *
L1536 1111 1111 1111 1111 1111 1111 1111 0110 *
L1568 1111 1111 1111 1111 1111 1111 1110 1011 *
C3FEF*
V0001 C000XXXXX00XXHHHLXX1 *
V0002 C001XXXXX00XXHHLHXX1 *
V0003 C010XXXXX00XXHLHHXX1 *
V0004 CXXXXXX11001HXXXXHX1 *
V0005 C011XXX0X00XHLHHHLX1 *
V0006 C011XXX11000LHHHLHX1 *
V0007 C011XXXXX00XXHHHLXX1 *
V0008 C100000XX00XXLHHHXX1 *
V0009 C100100XX00XXHLHHXX1 *
V0010 C100X1XXX00XXHLHHXX1 *
V0011 C100XX1XX00XXHLHHXX1 *
V0012 C101100XX00XXLHHHXX1 *
V0013 C1010XXXX00XXHLHHXX1 *
V0014 C10111XXX00XXHLHHXX1 *
V0015 C1011X1XX00XXHLHHXX1 *
V0016 C110XXXXX00XXHLHHX01 *
V0017 C110XXXXX00XXLHHHX11 *
43CA
```

LOGIC DIAGRAM FOR:
μBRANCH CONTROL USING AmPAL16R6A

Columns: Inputs (0–31)
Rows: Product Terms (0–63)

CLK

SELECT₂

SELECT₁

SELECT₀

DR₀

DR₁

DR₂

INT

LOAD
INT FF

CBR XOR
VR₁₁

INT FLAG

$\overline{OE\ MAP_1}$

$\overline{OE\ MAP_2}$

$\overline{OEJMP}$

$\overline{OE\ VEC}$

INTENB

VR₀

$\overline{OE}$ = GND

03862A-170

─✳─ = Fuse intact   ─⊠─ = All fuses intact   ─┼─ = Fuse blown

5-174
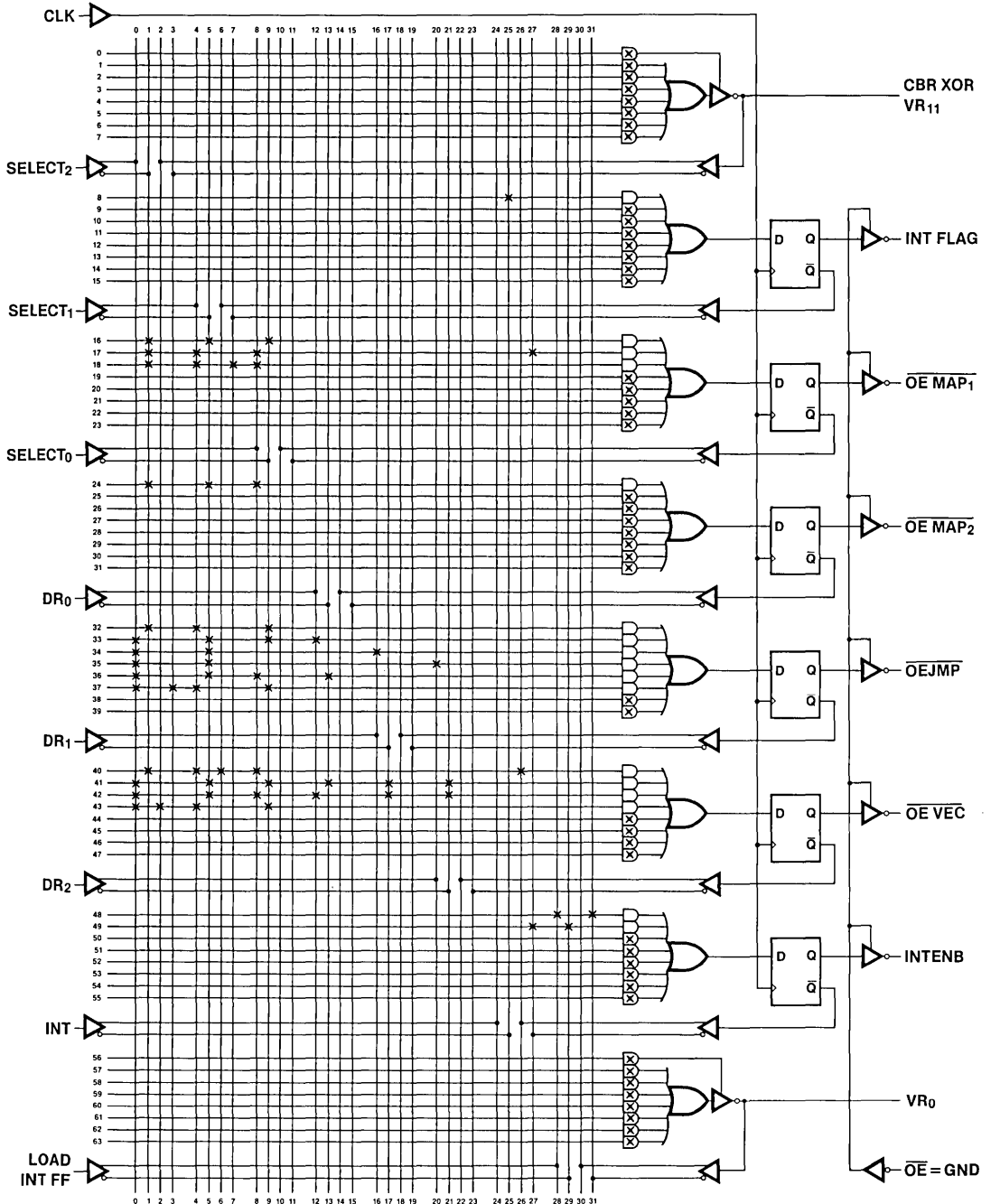
Testing, Programming, Reliability Information

Factory Testing of PALs
Logic Verification for PALs
PAL Programming
AMD Programmable Array Logic Reliability

Factory Testing of PALs

Advanced Micro Devices' PALs include special test circuitry designed to permit thorough AC and DC testing to be accomplished on an unprogrammed unit. This test circuitry is used to insure good programming yield and to verify that devices will meet all parametric and switching specifications after programming.

Programming circuitry testing includes tests to assure unique addressing of all fuses. To accomplish this, special test pads are provided which are accessible only during wafer probing. Using these, Advanced Micro Devices confirms that each driver is capable of sinking sufficient current to blow fuses and has appropriate saturation characteristics for AC performance. The ability of all circuitry in the programming path to handle the large currents and voltages necessary to blow fuses reliably is also thoroughly checked.

Each PAL has special test fuses. These test fuses are blown during factory testing and demonstrate beyond reasonable doubt that the device is capable of opening all fuses when programmed by the user. They also increase the confidence level in unique addressing.

The special probing pads and test fuses are all employed in programmability testing. This testing coupled with AMD's excellent process control gives industry leading programming yields (>98%) for all AMD PALs.

Special test circuitry, enabled by means of high voltage signals, checks functionality and DC parameters under conditions that simulate post programming operations. Most of the circuitry and levels that can be involved in operation after programming are checked under worst case conditions. For example, all input buffers are tested for functionality by switching them through a special path to a single output and all product term AND-gates are switched and sensed for uniqueness and functionality.

The fuses blown during programmability testing also permit 100% AC testing of a critical path in every device prior to shipment from the factory. These provide correlatable measures of the propagation delay times that the user can expect from his devices after he has placed his own logic in the PAL.

Because of the large percentage of die area devoted to fixed logic circuitry, all programmable devices from all manufacturers exhibit some percentage of units which fail to function to the desired truth table, even though all fuses are correctly programmed. AMD's special test circuits and extensive factory testing procedures have virtually eliminated this problem. However, to eliminate the possibility of any potential failures reaching the assembly line, the user should exercise the PAL after programming to insure that it functions correctly. This can be performed on an I/C tester, or on some PAL programmers, using user defined test vectors or by comparison against a known good unit.

Test vectors are relatively easy to generate for combinatorial designs using PALs. Sequential function testing is more difficult. AMD PALs are designed to provide the capability of loading the output registers to any desired value during testing. This feature, known as PRELOAD, simplifies functional testing of sequential devices. The following section, Logic Verification for PALs, describes PRELOAD in more detail and provides some guidelines for developing test procedures.

Logic Verification for PALs

by Brad Kitson
Advanced Micro Devices

The purpose of logic verification is to prove that a device functions correctly before it is put in a system. A completely reliable logic verification procedure should test all logic transitions of a device, through its normal inputs and outputs, and at normal TTL operating levels. This guarantees all parts of a device are functionally tested in the same way that they will be used in a system.

BENEFITS OF LOGIC VERIFICATION

The benefits of logic verification are to provide confidence that a device will function in a board. It also reduces production test time and cost because less board and system debug time is necessary. It is estimated that each succeeding level of testing during system production costs up to an order of magnitude more than the preceding level. So, should logic verification find a faulty device before it is put in a board and tested, significant test cost is saved. Should the faulty device be found before getting to system test or final on-site installation check-out, even more is saved. Programmable logic can implement functions on-chip that used to be done in multiple devices on a board. Thus some board-level testing can be done on programmable logic by logic verification, saving even more testing time and cost. Logic verification can also be beneficial during the initial design phase with programmable logic. The system designer can program a new design into a device and perform logic verification to see if the device performs the function correctly.

FUSE VERIFY FOR PROMs

With PROMs, logic verification is straightforward. A PROM is simply a read only memory with fuses as the information storage elements. Since the logic function of a PROM is to read out stored information at a given address, logic verification requires reading out and verifying correct status of the fuses at each address, after programming. Therefore, if a PROM verifies as correctly programmed, it is logically correct. Figure 1 shows the block diagram of a PROM. The fuse array is addressed by the input decode and is read from the outputs.

OUTPUTS

03862A-171
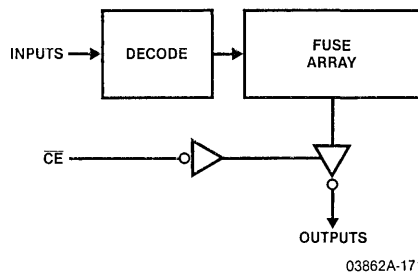
Figure 1. PROM

FUSE VERIFY FOR PROGRAMMABLE LOGIC

Fuse verification, however, doesn't perform logic verification for programmable logic. Programmable logic devices are programmed and fuse verified like PROMs, but don't logically function like PROMs. Figure 2 shows the block diagram of a registered PAL. The input decode of Figure 1 is again present as an input to the fuse array, but is enabled for programming and fuse verifying only. Instead, additional direct inputs and registered feedback inputs of the PAL serve as the logical inputs to its fuse array. The logical inputs must be disabled during fuse verification thereby losing any logical information from the direct inputs and state information from the feedback inputs. Therefore, fuse verification only helps prove correct programming. Figure 3 shows the block diagram of a combinatorial PAL. The programmable output enable and feedback buffers of this device are also untested during fuse verification. Arrows in Figures 2 and 3 denote the untested hardware blocks.

INPUT DON'T CARES

Figure 4 shows a simple design example utilizing two separate state machines. The microprogram for both machines is shown in Table 1. If both machines are implemented individually in SSI/MSI, relatively few don't care conditions result.

Table 1

FSM A

Current State	Input X	Next State
0	X	1
1	X	2
2	X	3
3	1	2
3	0	3

FSM B

Current State	Input Y	Next State
0	X	1
1	X	3
2	1	1
2	0	3
3	X	2

03862A-172
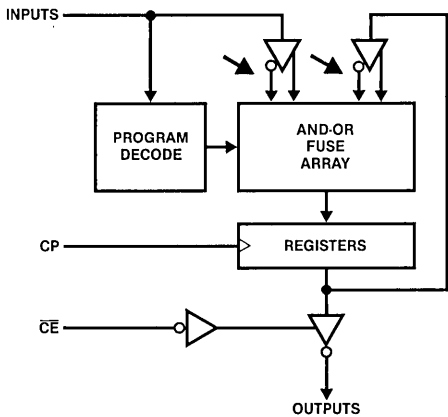
Figure 2. Registered PAL
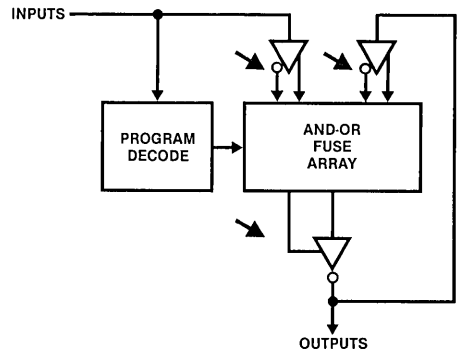
03862A-173

Figure 3. Combinatorial PAL
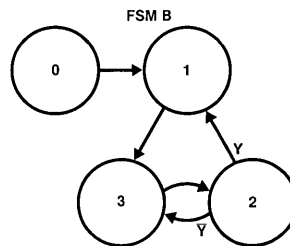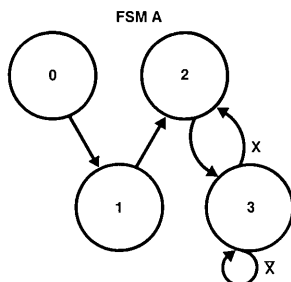
03862A-174

03862A-175

Figure 4

This is because the SSI/MSI used will only be enough to perform the function desired. In contrast, Table 2 shows the microprogram for the implementation of the design example in a single PAL. Since inputs are present for both machines and the PAL is a superset of either function, a large number of don't cares result. The point to be made here is that all don't care conditions must be tested to prove that both machines are still independent. In the example of Figure 4, to fully test FSM A, each state must be tested with every possible state of FSM B. Only in this way can one be sure that FSM A is independent of FSM B. Futhermore, every combination of states in FSM A and FSM B must be tested with all permutations of the input vectors. Table 3 shows the transitions necessary to fully test the first line in Table 2, i.e., the transition of FSM A state 0 to state 1. In general, the implementation of a logic function in a PAL, or other programmable device, will result in many don't cares since the device will be a superset of the function desired. So, one of the main requirements of logic verification is to prove that don't care conditions indeed don't care. To test all don't cares, an exhaustive list of input vectors is required. Also, in this case, a method is needed to repeatedly get into each state to test each don't care.

PRELOAD

The capability of Advanced Micro Devices' PALs to PRELOAD internal feedback registers facilitates logical verification for sequential programmable logic. Two complications exist when PRELOAD is not available. First, getting into a given state to test transitions out of that state becomes much more difficult. For example, referring to Figure 4, getting into state 3 of FSM A could require sequencing through states 0, 1, and 2. Furthermore, this would have to be repeated for each test of state 3. Moreover, to test a state in FSM A with a state in FSM B might require a lengthy and sometimes impossible sequence until the desired combination is obtained, i.e., the only way to get into FSM A state 2 and FSM B state 2 (2, 2) from state (0, 0) would be to sequence through (1, 1), (2, 3), (3, 2), and (3, 3). This is a difficult task to test just one state transition. With PRELOAD, state (2, 2) can be entered directly and tested. Second, power-up initialization becomes very difficult to test. Typically, a state machine has additional don't care states that only affect power-up. Should any of these don't care states be entered on power-up the machine must be able to exit them to start normal operations. Without PRELOAD these states can't be entered and tested to see if they can be exited. PRELOAD allows for these

states to be entered directly and tested. States 0 and 1 of FSM A and state 0 of FSM B are examples of don't care states.

Table 2

Combined State				Next State	
FSM A	FSM B	Inputs		FSM A	FSM B
		X	Y		
0	X	X	X	1	X
1	X	X	X	2	X
2	X	X	X	3	X
3	X	1	X	2	X
3	X	0	X	3	X
X	0	X	X	X	1
X	1	X	X	X	3
X	2	X	1	X	1
X	2	X	0	X	3
X	3	X	X	X	2

03862A-176

Table 3

State				Next State
FSM A	FSM B	Inputs		FSM A
		X	Y	
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	1
0	1	1	1	1
0	2	0	0	1
0	2	0	1	1
0	2	1	0	1
0	2	1	1	1
0	3	0	0	1
0	3	0	1	1
0	3	1	0	1
0	3	1	1	1

03862A-177

PAL Programming

AMD PALs are manufactured using the high performance IMOX oxide isolated process and high reliability platinum-silicide fuses. These technologies require the use of specific programming equipment which has been designed to ensure consistent programming yields in excess of 98%. To maintain these extremely high programming yields, AMD subjects all approved PAL programming equipment to a complete testing and qualification procedure which assures the user that the programmer will program AMD PALs reliably.

The fusing algorithm, which is described in detail in the reliability report, is designed to minimize tight tolerance requirements on the programming equipment. A chip enable input is used to gate the fusing current from a programming source voltage on the PAL output. The delivery of fusing current is therefore controlled by the switching speed of the internal PAL circuitry, not by the circuitry in the programmer. This should minimize the need for constant recalibration of a programmer. However, it is recommended that a user's log be maintained with each machine to collect a record of the hours of service use and programming yield of each lot. The programming equipment should be calibrated after every 50 hours of service or whenever programming yields fall below 98% with AMD PALs.

PROGRAMMER APPROVAL CRITERIA

Full details of the required programming parameters, waveforms and addressing schemes are provided on each device data sheet. All AMD PALs, standard, high speed ('A' versions) and half power ('L' versions) use the same algorithm and can be programmed on identical modules and adapters.

The minimum requirements for approval of a programmer by AMD are that it
- Programs and fuse verifies AMD PALs with the appropriate conditions described on the device data sheet and provides consistent yields in excess of 98%
- Programs the security fuse
- Can program an AMD PAL after master loading from another manufacturer's device.

A desirable feature is the ability to perform logical verification following fuse verification. This will prevent the assembly of a PAL having a logical failure, which cannot be detected from the fuse array alone, into a system.

As noted in the section on testing, all AMD PALs have additional circuitry built into the device to aid detection of logical failures at this point.

Another important feature is the ability to generate output in the JEDEC industry standard Programmable Logic Data Transfer Format. This insures that the equipment will program PALs from all suppliers without special modification or awkward copying procedures.

QUALIFIED PROGRAMMING EQUIPMENT

The list of AMD qualified PAL programmer models appears below. New programming equipment and vendors are constantly under evaluation. Contact your local AMD Field Applications Engineer or the factory to determine the approval status of any equipment not listed here.

AMD is committed to maintaining continued close working relationships with the major PAL programmer manufacturers so that new programmable logic devices will be properly supported in a timely manner.

Vendor	Programmer Model(s)	AMD PAL Personality Module	Socket Adapter
Data I/O 10525 Willows Rd. N.E. Redmond, WA 98052	Model-100, 29, 19, or 17	Logicpak 950-1942-001	715-1947-003
Digilec, Inc. 7335 E. Acoma Dr. Dept-103 Scottsdale, AZ 85260	Under Development		
Kontron Electronics, Inc. 630 Price Avenue Redwood City, CA 94063	Model-MPP-80S or EPP80	MOD-33	SA37
Stag Microsystems 528-5 Weddel Drive Sunnyvale, CA 94086	Model-PPX (or)	PPM2200	Am202S
	ZL30	On-Board	On-Board
Structured Design, Inc. 1700 Wyatt Drive Suite 3 Santa Clara, CA 95084	SD-1000	On-Board	On-Board

AMD Programmable Array Logic Reliability

by W. Sievers
Advanced Micro Devices

Advanced Micro Devices' Programmable Array Logic (PAL) devices are based on two key technologies with many years of high volume production experience behind them.

1) IMOX—the basic process technology employed is IMOX, an advanced ion-implanted, oxide isolated structure. IMOX provides very high performance devices with predictable manufacturing yields. It has accumulated many millions of hours of life test history through its application to the Am27S series of PROMs and the Am2900 family of bipolar microprocessors.
A comprehensive report on IMOX reliability titled IMOX RELIABILITY REPORT (AMD publication #03687A-MPR) is available for those interested in a detailed presentation of this subject.

2) Platinum-silicide fuses—this fuse structure was originally developed for use on Advanced Micro Devices' families of junction isolated PROMs. It quickly established a new standard of excellence for high programming yields and long term reliability. Several years ago it was applied to a new generation of ultra high performance PROMs based on the IMOX process.

This combination of IMOX and platinum-silicide fuses has an outstanding record of reliability which has been verified repeatedly through in-house life testing and by high reliability customer qualification testing and system use.

Advanced Micro Devices' PALs are fabricated with this same combined process technology. Not only is the technology for building PALs and PROMs the same, but also the programming algorithm and programming circuitry used to program the platinum-silicide fuses are the same in all characteristics of importance. The result is that the fusing conditions seen by an AMD PAL fuse are the same as those seen by an AMD PROM fuse.

Due to the common process technology, fuse design and fuse programming circuitry design, reliability and programming yield results are expected to be the same for PALs and PROMs. Data accumulated to date on PALs appears to confirm this expectation.

This report describes:
1) The characteristics of the platinum-silicide fuse and programming conditions for the fuse.
2) The dynamic and static burn-in circuits used for HTRB reliability testing.
3) Reliability results accumulated to date on IMOX PROMs and PALs.

PLATINUM-SILICIDE FUSE

Fusing Technique

Advanced Micro Devices' PAL circuits are designed to use a programming algorithm which minimizes the requirements on the programmer yet allows the circuit to fuse the platinum-silicide links quickly and reliably.

The sequence of events to program a fuse are:
1) V_{CC} power is applied to the chip.
2) The address of the fuse to be programmed is selected by TTL levels on the appropriate address pins.
3) The outputs are disabled (Pin 1 serves this purpose on PALs).
4) The programming voltage is then applied to one output.
5) A fuse enable is accomplished by raising an input to a level above normal TTL operating voltage. (Pin 11 is used for this on PALs.) This action gates the current flow through the proper fuse, resulting in an open fuse in a few microseconds.
6) The output programming voltage is lowered and then removed.
7) The device is enabled and clocked if required. The output state then indicates whether successful programming has occurred. If programming has not occurred a sequence of much longer pulses is applied until programming occurs.
8) The sequence of 2 through 7 is repeated for each bit which must be programmed.

There are several advantages to this technique relative to that used by other PAL manufacturers. First, the two high current power sources, V_{CC} and the voltage applied to the output, do not have critical timing requirements. As the fusing current is gated through the fuse actively, there is no dependence on the rise rate of the programming voltage. A fast application of fusing current is desirable for optimum fusing. Since the output programming voltage does not have to be applied rapidly, breakdown and latchback problems attributed to fast voltage rise times on the output are avoided.

This programming procedure has a second major advantage. If the fuse does not open during the first attempt to blow it, longer programming pulses are used. With the platinum-silicide fuse longer programming pulses may be safely applied with no danger of developing a reliability problem. The algorithm can therefore be designed to minimize the time required to program by using a fast first pulse to maximize the probability that any circuit will program. Then a longer pulse can be applied to the occasional fuse that does not open with the first short pulse. Most devices do fuse satisfactorily with all short pulses.

Circuit defects resistant to discovery with pre-programming testing and random marginally wide fuses make it important to have a material not adversely affected by application of near DC programming conditions. The platinum-silicide fuse is such an element.

Fuse Characteristics

When a fast (less than 500ns rise time) current pulse is applied to a fuse, the fuse voltage rises abruptly to a value approaching the level expected from calculation of the room temperature resistance. However, it quickly falls to a value of approximately 2V. This value is nearly independent of the applied curent. During the period of time the fuse is molten, the fuse current drops very abruptly to zero indicating the separation of the platinum-silicide into two distinct sections. Scanning Electron Microscope photographs of the resulting fuses (Figure 1) indicate that the typical case is a sharp clean separation in excess of a micron. This separation occurs in the center of the fuse because the "bow-tie" structure (Figure 2) concentrates the energy density in the center away from the aluminum interconnect lines. The energy density in the center of the fuse creates temperatures substantially greater than those required to melt the silicide. Melted material is then "wicked" from the center of the fuse to either side due to surface tension.
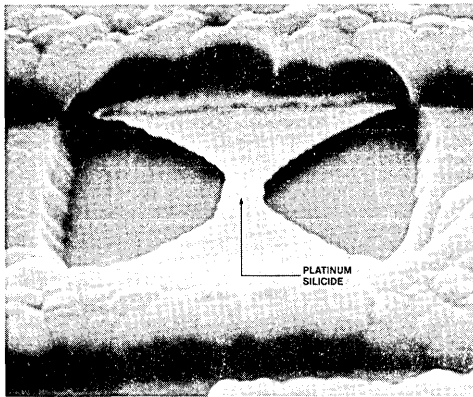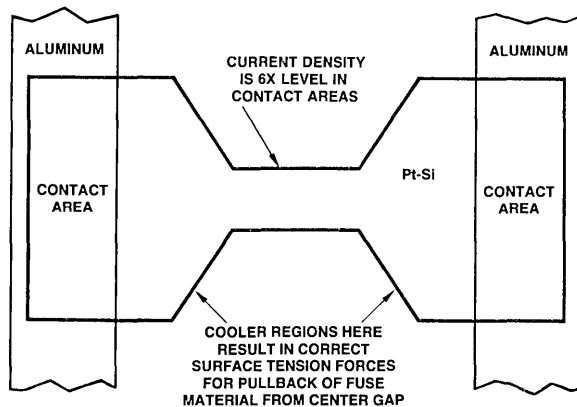

Unprogrammed Fuse
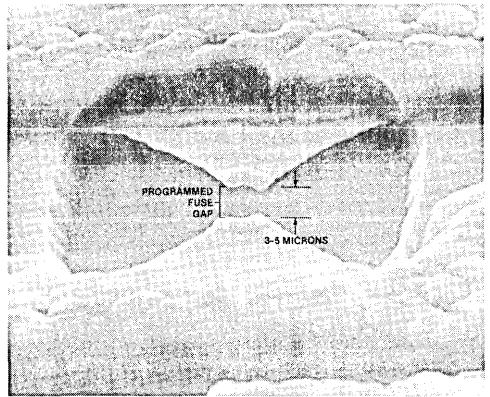
03862A-178

Programmed Fuse

03862A-179

Figure 1

Figure 2. Bow-tie Fuse Design

03862A-180

RELIABILITY OF FUSES PROGRAMMED UNDER NON-OPTIMAL CONDITIONS

The marginally opened fuse has been studied at AMD in detail even though it rarely occurs in practice. Under conditions where the fuse is purposely blown at much slower rates, it is possible for the fuse to assume a high impedance state which is sensed as an open fuse by the circuit. This occurs when the fuse cools before separation is achieved. Electrical and SEM studies of fuses blown under these conditions indicate that a small conductive path of silicon remains of sufficiently high resistance to prevent the power transfer required for complete opening on subsequent applications of power.

Under these slow-blow conditions, sufficient time exists for the heat flow to carry a significant amount of energy away from the fuse preventing the normal abrupt separation.

To investigate what might happen if a fuse were subjected to these under-blow conditions a large number of fuses were deliberately programmed this way at AMD. After over two thousand hours of life testing these devices with under-blown fuses, there have been no failures. It is clear from the study that partially blown platinum-silicide fuses are stable even though it will rarely occur in a circuit which has been programmed under normal conditions. Advanced Micro Devices believes that such fuses do not represent a reliability hazard based on this study and the results of other studies run on PROMs.

It should be noted that most manufacturers carefully specify the conditions under which their devices must be programmed in order to avoid reliability problems. Reliability data available on these devices must be assumed to have been generated using optimally programmed devices.

Advanced Micro Devices believes that the study described here and over forty billion fuse hours of data from life testing many different production lots of PROMs and PALs demonstrates the outstanding reliability record of the platinum-silicide fuse under a wide variety of conditions.

Reliability Testing Data

Data on the reliability of PAL and PROM devices with platinum-silicide fuses has been gathered over millions of device hours and more than 40 billion fuse hours of testing at 125°C. This data is shown in Table 1 to project a unit failure at 60% confidence of 0.0003% per 1000 hours at 70°C.

The life test circuits used in this work conform to MIL-STD-883 Method 1005 conditions C and D and are shown in Figure 3.

SUMMARY

In high temperature operating life tests (HTOL) to date, PALs are exhibiting the same excellent reliability results shown by other IMOX and platinum-silicide fuse products. HTOL testing is an ongoing activity with all product lines at AMD. Updates of these results are generated periodically and can be obtained through inquiry to the AMD Programmable Logic Product directorate.
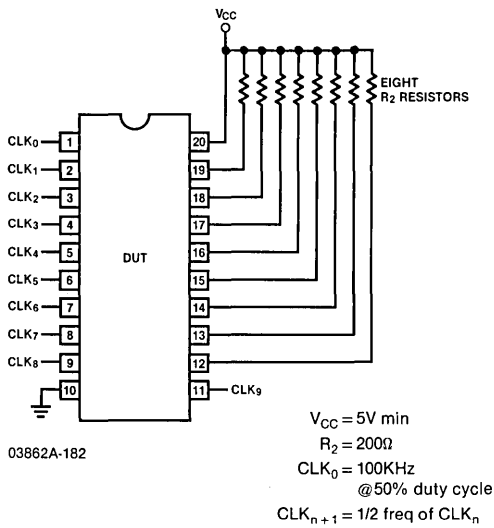
Table 1. Bipolar PROM and PAL Reliability Summary

Product	Production Lots	Units Tested	Total Unit Hours (thousands)	Total Fuse Hours (billions)	Unit Failures	Fuse Related Failures	Unit Failure Rate @ 60% Confidence %/1000 hrs at 125°C	Unit Failure Rate* @ 60% Confidence %/1000 hrs at 70°C
27S18/19 (256-bit PROM)	5	491	982	0.251B	0	0	0.10	0.0010
27S20/21 (1K bit PROM)	16	1321	2207	2.260B	2**	0	0.14	0.0013
27S12/13 (2K bit PROM)	11	571	1840	3.768B	0	0	0.05	0.0005
27S15 27S27 27S28/29 27S32/33 (4K bit PROM)	24	1870	1408	5.767B	0	0	0.07	0.0007
27S180/181 (8K bit PROM)	12	463	926	7.586B	0	0	0.11	0.0010
27S184/185 IMOX (8K bit PROM)	15	556	1112	9.109B	0	0	0.09	0.0008
27S190/191 IMOX (16K bit PROM)	2	69	795	13.025B	0	0	0.12	0.0011
20-pin IMOX PALs	10	976	700	1.434B	0	0	0.13	0.0012
Totals for PALs and PROMs	95	6317	9970	43.200B	2**	0	0.031	0.0003

03862A-181

*Assuming on activation energy of 1.0 eV.
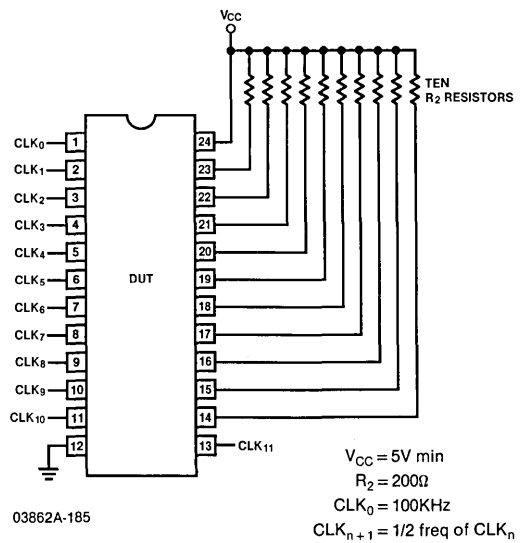**Oxide failure.
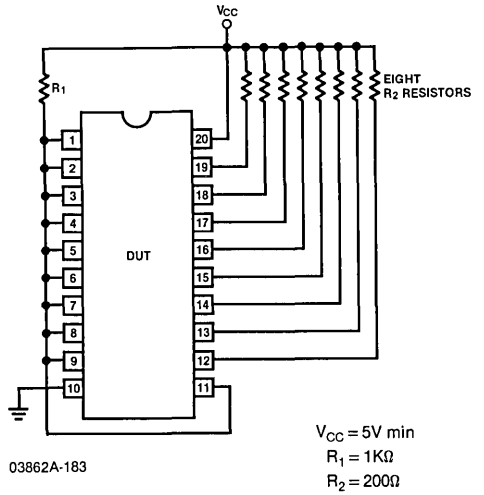
$V_{CC} = 5V$ min
$R_2 = 200\Omega$
$CLK_0 = 100KHz$
@50% duty cycle
$CLK_{n+1} = 1/2$ freq of CLK_n

03862A-182

(a) 20-Pin Dynamic Burn-In
MIL-STD-883-B Condition D

$V_{CC} = 5V$ min
$R_1 = 1K\Omega$
$R_2 = 200\Omega$

03862A-183

(b) 20-Pin Static Burn-In
MIL-STD-883-B Condition C

$V_{CC} = 5V$ min
$R_1 = 1K\Omega$

03862A-184

(c) 24-Pin Static Burn-In
MIL-STD-883-B Condition C

$V_{CC} = 5V$ min
$R_2 = 200\Omega$
$CLK_0 = 100KHz$
$CLK_{n+1} = 1/2$ freq of CLK_n

03862A-185
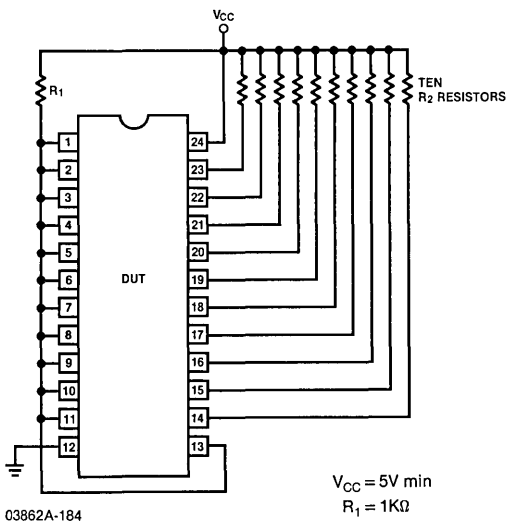
(d) 24-Pin Dynamic Burn-In
MIL-STD-883-B Condition D

Figure 3. Life Test Circuits for AMD PALs

General

Ordering Information
AMD Sales Offices

ORDERING INFORMATION

AmPAL XXXXX X XXX

Basic Device ————————
16H8
16HD8
etc.

Speed/Power Selection ————
    = Standard Speed/Standard Power
A = High Speed/Standard Power
L = Standard Speed/Half Power

Screening
* = Standard Process Flow
B = Burn-In

Operating Range
C = Commercial
M = Military

Package
P = Molded DIP (Plastic)
D = Hermetic DIP (Cerdip)
L = Chip-Pak™ (LCC)

Chip-Pak is a trademark of Advanced Micro Devices, Inc.

*A blank in this position of the ordering code indicates the part has been screened to standard process flow.

Device Listing

Ordering Part #	Package/Operating Range Combinations Available	Ordering Part #	Package/Operating Range Combinations Available
AmPAL16H8	DC, DM, LC, LM, PC	AmPAL16LD8L	DC, DM, LC, LM, PC
AmPAL16H8A	DC, DM, LC, LM, PC	AmPAL16R4	DC, DM, LC, LM, PC
AmPAL16H8L	DC, DM, LC, LM, PC	AmPAL16R4A	DC, DM, LC, LM, PC
AmPAL16HD8	DC, DM, LC, LM, PC	AmPAL16R4L	DC, DM, LC, LM, PC
AmPAL16HD8A	DC, DM, LC, LM, PC	AmPAL16R6	DC, DM, LC, LM, PC
AmPAL16HD8L	DC, DM, LC, LM, PC	AmPAL16R6A	DC, DM, LC, LM, PC
AmPAL16L8	DC, DM, LC, LM, PC	AmPAL16R6L	DC, DM, LC, LM, PC
AmPAL16L8A	DC, DM, LC, LM, PC	AmPAL16R8	DC, DM, LC, LM, PC
AmPAL16L8L	DC, DM, LC, LM, PC	AmPAL16R8A	DC, DM, LC, LM, PC
AmPAL16LD8	DC, DM, LC, LM, PC	AmPAL16R8L	DC, DM, LC, LM, PC
AmPAL16LD8A	DC, DM, LC, LM, PC	AmPAL22V10	In Development
		AmPAL22V10A	In Development

U.S. AND CANADIAN SALES OFFICES

SOUTHWEST AREA

Advanced Micro Devices
360 N. Sepulveda, Suite 2075
El Segundo, California 90245
Tel: (213) 640-3210

Advanced Micro Devices
10050 N. 25th Street
Suite 235
Phoenix, Arizona 85021
Tel: (602) 242-4400

Advanced Micro Devices
4000 MacArthur Boulevard
Suite 5000
Newport Beach, California 92660
Tel: (714) 752-6262

Advanced Micro Devices
5955 De Soto Avenue, Suite 241
Woodland Hills, California 91367
Tel: (213) 992-4155

Advanced Micro Devices
9455 Ridgehaven Court
Suite 230
San Diego, California 92123
Tel: (619) 560-7030

NORTHWEST AREA

Advanced Micro Devices
2700 Augustine Drive, Suite 109
Santa Clara, California 95051
Tel: (408) 727-1300

Advanced Micro Devices
1873 South Bellaire Street
Suite 920
Denver, Colorado 80222
Tel: (303) 691-5100

NORTHWEST AREA (Cont.)

Advanced Micro Devices
One Lincoln Center, Suite 230
10300 Southwest Greenburg Road
Portland, Oregon 97223
Tel: (503) 245-0080

Advanced Micro Devices
Honeywell Ctr., Suite 1002
600 108th Avenue N.E.
Bellevue, Washington 98004
Tel: (206) 455-3600

MID-AMERICA AREA

Advanced Micro Devices
500 Park Boulevard, Suite 940
Itasca, Illinois 60143
Tel: (312) 773-4422

Advanced Micro Devices
9900 Bren Road East, Suite 601
Minnetonka, Minnesota 55343
Tel: (612) 938-0001

Advanced Micro Devices
3592 Corporate Drive, Suite 108
Columbus, Ohio 43229
Tel: (614) 891-6455

Advanced Micro Devices
8240 MoPac Expressway
Two Park North, Suite 385
Austin, Texas 78759
Tel: (512) 346-7830

Advanced Micro Devices
6750 LBJ Freeway, Suite 1160
Dallas, Texas 75240
Tel: (214) 934-9099

MID-ATLANTIC AREA

Advanced Micro Devices
40 Crossways Park Way
Woodbury, New York 11797
Tel: (516) 364-8020

Advanced Micro Devices
290 Elwood Davis Road
Suite 316
Liverpool, New York 13088
Tel: (315) 457-5400

Advanced Micro Devices
2 Kilmer Road
Edison, New Jersey 08817
Tel: (201) 985-6800

Advanced Micro Devices
107 Lakeside Drive
Horsham, Pennsylvania 19044
Tel: (215) 441-8210
TWX: 510-665-7572

Advanced Micro Devices
205 South Avenue
Poughkeepsie, New York 12601
Tel: (914) 471-8180
TWX: 510-248-4219

NORTHEAST AREA

Advanced Micro Devices
6 New England Executive Park
Burlington, Massachusetts 01803
Tel: (617) 273-3970

Advanced Micro Devices (Canada) Ltd.
2 Sheppard Avenue East
Suite 1610
Willowdale, Ontario
Canada M2N5Y7
Tel: (416) 224-5193

SOUTHEAST AREA

Advanced Micro Devices
Parkway Center
One Parkway Drive Building
7257 Parkway Drive, Suite 204
Dorsey, Maryland 21076
Tel: (301) 796-9310

Advanced Micro Devices
7850 Ulmerton Road, Suite 1A
Largo, Florida 33541
Tel: (813) 535-9811

Advanced Micro Devices
4740 North State Road #7
Suite 102
Ft. Lauderdale, Florida 33319
Tel: (305) 484-8600

Advanced Micro Devices
6755 Peachtree Industrial Boulevard
Suite 104
Atlanta, Georgia 30360
Tel: (404) 449-7920

Advanced Micro Devices
8 Woodlawn Green, Suite 220
Woodlawn Road
Charlotte, North Carolina 28210
Tel: (704) 525-1875

INTERNATIONAL SALES OFFICES

BELGIUM
Advanced Micro Devices
Overseas Corporation
Avenue de Tervueren, 412, bte 9
B-1150 Bruxelles
Tel: (02) 771 99 93
TELEX: 61028

FRANCE
Advanced Micro Devices, S.A.
Silic 314, Immeuble Helsinki
74, rue d'Arcueil
F-94588 Rungis Cedex
Tél: (01) 687.36.66
TELEX: 202053

GERMANY
Advanced Micro Devices GmbH
Rosenheimer Str. 139
D-8000 Muenchen 80
Tel: (089) 40 19 76
TELEX: 05-23883

Advanced Micro Devices GmbH
Harthaeuser Hauptstrasse 4
D-7024 Filderstadt 3
Tel: (07158) 30 60
TELEX: 07-21211

Advanced Micro Devices GmbH
Zur Worth 6
D-3108 Winsen/Aller
Tel: (05143) 53 62
TELEX: 925287

HONG KONG
Advanced Micro Devices
1303 World Commerce Centre
Harbour City
11 Canton Road
Tsimshatsui, Kowloon
Tel: (852) 3 695377
TELEX: 50426
FAX: (852) 123 4276

ITALY
Advanced Micro Devices S.R.L.
Centro Direzionale
Palazzo Vasari, 3° Piano
I-20090 MI2 – Segrate (MI)
Tel: (02) 215 4913-4-5
TELEX: 315286

JAPAN
Advanced Micro Devices, K.K.
Dai 3 Hoya Building
8-17, Kamitakaido 1 chome
Suginami-ku, Tokyo 168
Tel: (03) 329-2751
TELEX: 2324064
FAX: (03) 326 0262

SWEDEN
Advanced Micro Devices AB
Box 7013
S-172 07 Sundbyberg
Tel: (08) 98 12 35
TELEX: 11602

UNITED KINGDOM
Advanced Micro Devices (U.K.) Ltd.
A.M.D. House,
Goldsworth Road,
Woking,
Surrey GU21 1JT
Tel: Woking (04862) 22121
TELEX: 859103

Advanced Micro Devices maintains a network of representatives and distributors in the U.S. and around the world. For a sales agent nearest you, call one of the AMD offices above.