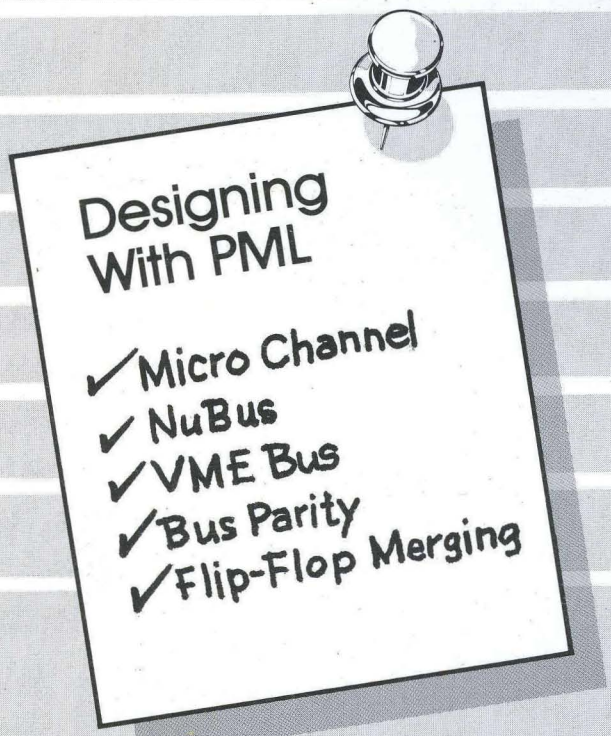
Signetics

PLHS501 Application Notes Vol. 2



PHILIPS

PHILIPS

Signetics reserves the right to make changes, without notice, in the products, including circuits, standard cells, and/or software, described or contained herein in order to improve design and/or performance. Signetics assumes no responsibility or liability for the use of any of these products, conveys no license or title under any patent, copyright, or mask work right to these products, and makes no representations or warranties that these products are free from patent, copyright, or mask work right infringement, unless otherwise specified. Applications that are described herein for any of these products are for illustrative purposes only. Signetics makes no representation or warranty that such applications will be suitable for the specified use without further testing or modification.

LIFE SUPPORT APPLICATIONS

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

Signetics registers eligible circuits under
the Semiconductor Chip Protection Act.

PLHS501

APPLICATION NOTES

Vol 2

IMPORTANT NOTICE

Signetics reserves the right to make changes without notice, in the products including circuits, standard cells, and/or software, described or contained herein in order to improve design and/or performance. Signetics assumes no responsibility or liability for the use of any of these products, conveys no license or title under any patent, copyright, or mask work right to these products or processes, and makes no representations or warranties that these products are free from patent, copyright, or mask work right infringement, unless otherwise specified. Applications that are described herein for any of these products are for illustrative purposes only. Signetics makes no representation or warranty that such applications will be suitable for the specified use without further testing or modification.

LIFE SUPPORT POLICY

SIGNETICS PRODUCTS ARE NOT FOR USE AS CRITICAL COMPONENTS IN LIFE SUPPORT DEVICES OR SYSTEMS WITHOUT EXPRESS WRITTEN APPROVAL OF AN OFFICER OF SIGNETICS CORPORATION. As used herein:

- Life support devices or systems are devices or systems which, (a) are intended for surgical implant into the body, or (b) support or sustain life, and whose failure to perform, when properly used in accordance with instructions for use provided in the labeling, can be reasonably expected to result in a significant injury to the user.

- A critical component is any component of a life support device or system whose failure to perform can be reasonably expected to cause the failure of the life support device or system, or to affect its safety or effectiveness.

TABLE OF CONTENTS

TABLE OF CONTENTS [continued]

LIST OF FIGURES

LIST OF FIGURES [continued]

LIST OF TABLES

1. Introduction

This document is written assuming the reader is familiar with Signetics PLHS501. As well, we shall assume familiarity with the predecessor document "Designing with PML" and some exposure to Signetics AMAZE software. The goal of this document (i.e., Part Two) is to expand on the original ideas and present some cookbook solutions to some useful design problems. Part Two also reflects nearly a year of experience through the multitude of design-ins achieved with the PLHS501. In fact, several of the design solutions presented here were contributions from our customers through our field applications organization. Designs we have encountered fell into a couple of interesting categories. First, many users view the part as a natural step in eliminating extraneous board "glue" (10 or more chips) or eliminating multiple programmable array logic devices (usually 3 to 5 units). Others recognized the PLHS501 capabilities of extremely wide logic functions and still others chose to invent their own solutions to standard bus interfaces. Commercially available bus interfaces often "miss the mark" and creative designers wish to implement exactly the functions they need in a concise, effective manner. To date, we have seen PLHS501 interfaces to the VME Bus II, FAST Bus, NuBus, GPIB and the IBM Micro Channel for the PS/2 system.

Before presenting these solutions however, it is appropriate to review the PML basics and expand on a number of issues which have been found to be important but which were previously treated lightly.

2. PLHS501 Review

The PLHS501 is a 52–pin, bipolar programmable logic device with a very powerful architecture. Unlike classic and/or based architectures, its basic building block is the NAND function which is configured in a foldback programming array. By cascading successive NAND functions through the array, both combinational and sequential structures may be obtained. The PLHS501 has 24 dedicated inputs, 16 outputs (with several varieties) and eight bidirectional pins. The internal NANDs may be cascaded to any depth needed, to achieve effective solutions using logic structures such as muxes, decoders and flip–flops without going off chip and wasting I/O pins to achieve cascading. To use the PLHS501 effectively, the designer should attempt to fold in function and remain within the chip as much as possible before exiting.

Figure 2–1 shows the PLHS501 architecture and illustrates several of the timing paths for internal signals to give the designer a feeling for maximum time delay within the part. These numbers are worst case maximums, regardless of switching directions, so the user may be assured that in general, the PLHS501 will be faster than these numbers.

The shorthand notation of Figure 2–1 hides something which many designers have been impressed with in the PLHS501, the wide input NAND gates. Figure 2–2 shows just how wide the internal NANDs are, from a logical viewpoint. Each NAND can accommodate up to 32 external inputs and 72 internal inputs. Hence, the part is ideal for wide decoding of 32–bit address and data busses. With 72 copies of the wide NAND, the PLHS501 is often compared against low end gate arrays. While flattering, this gives no usable method to determine the degree to which functions can be fit into the device. As a rule of thumb, the PLHS501 can accommodate three or more PLA devices and usually four to five PAL® devices.

For any particular design, the user should refer to Table 2–1 and evaluate his design incrementally, tallying against a 72 gate budget. This is a ballpark estimation against the NAND capacity of the core of the part. The clever designer will find additional function by correctly exploiting the output logic.

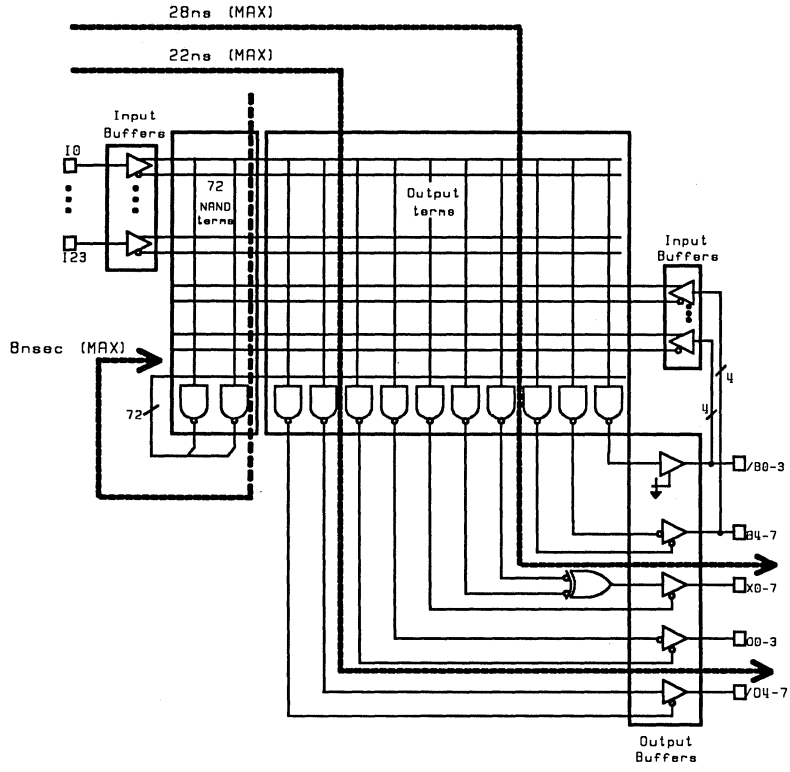


Figure 2–1 PLHS501 Logic Diagram

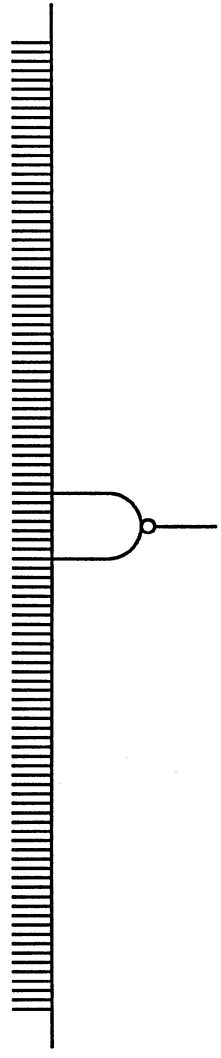


Figure 2–2 An Internal NAND Logic Equivalent

FUNCTION	PLHS501 INTERNAL NAND EQUIVALENT	COMMENTS
Gates: NANDs	1	For 1 to 32 input variables
ANDs	1	"
NORs	1	"
ORs	1	"
Decoders: 3 to 8	8	Inverted inputs available
4 to 16	16	"
5 to 32	32	Inverted inputs available (24 chip outputs only)
Encoders: 8 to 3	15	Inverted inputs, 2 logic levels
16 to 4	32	Inverted inputs, 2 logic levels
32 to 5	41	Inverted inputs, 2 logic
Multiplexors:		
4 to 1	5	Inverted inputs available
8 to 1	9	"
16 to 1	17	"
27 to 1	28	Can address only 27 externally inputs - more if internal
Flip-Flops: D-FF	6	With asynch S-R
T-FF	6	"
J-K-FF	10	"
Transparent-D Latch	4	"
S-R Latch	2	"
Adders: 8-bit	45	Full carry-lookahead (four levels of logic)
Barrel Shifters:		
8-bit	72	2 levels of logic

TABLE 2-1 PLHS501 Gate Count Equivalents

3. Flip–Flop Basics

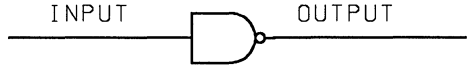
Most designers view flip–flops as black boxes with data inputs and outputs as well as additional control inputs. Some flip–flops are designed as primitive transistor structures, but in the past, gate array designers used their elementary building block, the NAND gate, to make flip–flops. Because the PLHS501 is also largely structured from NANDs, we can draw upon years of well known NAND–based flip–flop designs to readily implement flip–flops within the PLHS501.

Figure 3–1, 3–2, 3–3 and 3–4 give single sheet summaries of several flip–flop configurations. It should be noted that the transparent latch is recommended for data capturing, but not for state machines due to potential glitching. The edge triggered D–type is a convenient building block. Although external gates are saved with the J–K structure, it is at the expense of additional NANDs within the J–K flip–flop itself.

3.1 Notation

The delay of a NAND gate is most often designated as t_{PLH} or t_{PHL}, indicating that the gate output makes a high to low (t_{PHL}) or low to high (t_{PLH}) transition. For the flip–flops transition, the high–to–low ID is D0 and the low–to–high ID is D1. This also holds true for structures fully contained within the foldback core, because input and output time delays will differ and change the performance. Knowing the basic concepts, the designer can expand these structures to include I/O pins and generate flip–flops wrapped around the part – but, he must derate his parameters accordingly to reflect the slower paths.

Because it will be lengthy to explain all of the flip–flop configurations given, we will show only one in some detail. The interested reader can verify the rest by manual analysis or by digital simulation. The following table gives the typical and worst case values for an internal foldback NAND gate.



SYMBOL	PARAMETER		LIMITS		UNIT
	To (Output)	From (Input)	Min	Max	
tPHL			5.5	6.5	
tPLH		ANY	6.5	8.0	ns

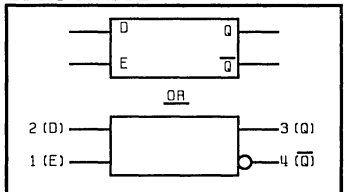
TABLE 3–1 Internal Fold Back NAND Gate

The single D–latch with enable Active–High can be described in terms of the propagation delay formula given in Figure 3–1. For instance, the first propagation is for D to Q where the Q output transitions from High–to–Low (i.e., t_{PD0}). To do this, assume Q is High so the /Q term is Low. To switch the state, /Q must be flipped first. Hence, the logic variable enters G2, then passes though G7, G4 and finally G3. This presents four transitions, two from Low–to–High (G2 & G4 outputs) and two from High–to–Low (G1 & G3 outputs). Hence, the formula reflects 2(d1&d0) which, using the previous table, gives 2(8+6.5)=29nsec.

This is a worst case value, using typicals will give a value of 2(5.5+6.5)=24nsec.

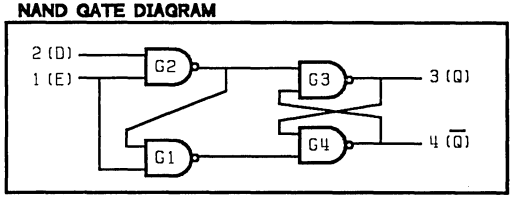
Switching in the other direction is a little different. Assuming t_{PD1} goes from Q=0 to Q=1, the /Q signal must be initially 1. Hence, G3 is armed for immediate transition. Hence, the time delay is simply traversing G2 and G3. One of them will go High–to–Low (G2) and the other Low–to–High (G3). The formula reflects the sum of the two transitions: t_{PD1} = d1+d0. From the table, this is 14.5nsec (worst case) or 12nsec (typical). The rest of the formula must be similarly analyzed, but the method is straightforward.

LOGIC SYMBOL



Note: This part is functionally similar to the 7475

NAND GATE DIAGRAM



PROPAGATION DELAY FORMULAS:

FROM	TO	FORMULAS
D	Q	tPD0 = d1+d0 x 2
		tPD1 = d1+d0
D	$\overline{Q}$	tPD0 = d1 + 2Xd0
		tPD1 = 2xd1 + d0
E	Q	tPD0 = d1 + 2Xd0
		tPD1 = d1+d0
E	$\overline{Q}$	tPD0 = d1 + 2Xd0
		tPD1 = d1+d0
SETUP TIME		= d1 + 2Xd0
HOLD TIME		= 0
ENABLE PULSE		= E TO Q tPD0

FUNCTION (TRUTH) TABLE

Operating mode	E	D	Q	$\overline{Q}$
Data enabled	1	0	0	1
	1	1	1	0
Data latched	0	X	0_{n-1}	$\overline{0}_{n-1}$

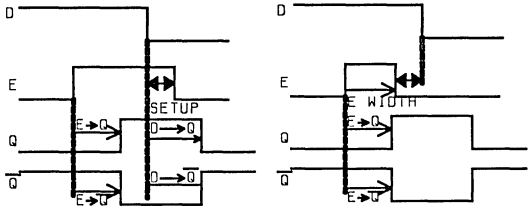
Notes:

1. When input Enable (E) is High, data enters the latch at "D" and appears at outputs "Q" and "$\overline{Q}$"; the "Q" output follows the data as long as E is High. One setup time before the High-to-Low transition of E, D is stored in the latch; the latched outputs remain stable as long as E is Low.

2. 0_{n-1} = state before High-to-Low transition of E.

3. X = don't care.

TIMING WAVEFORMS:



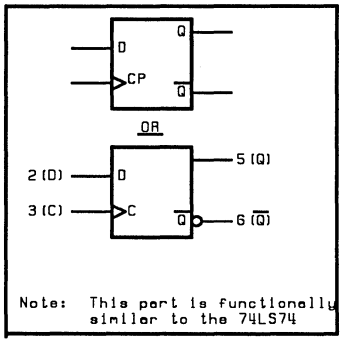
Notes:
Spikes can occur on $\overline{Q}$ during the propagation delay of E to $\overline{Q}$.

PROPAGATION DELAYS:

PARAMETER	FROM	TO	DELAY in ns
tPD0	D	Q	
	D	$\overline{Q}$	
	E	Q	
	E	$\overline{Q}$	
tPD1	D	Q	
	D	$\overline{Q}$	
	E	Q	
	E	$\overline{Q}$	
D (SETUP TIME)			
D (HOLD TIME)			
ENABLE PULSE WIDTH			

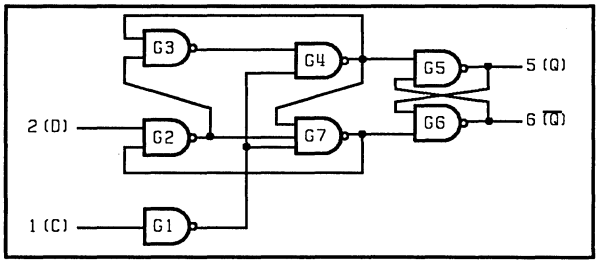
Figure 3-1 Single D-Latch (Enable Active-High)

LOGIC SYMBOL



Note: This part is functionally similar to the 74LS74

NAND GATE DIAGRAM



PROPAGATION DELAY FORMULAS:

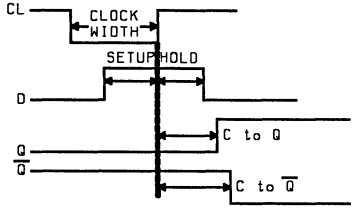
FROM	TO	FORMULAS
C	Q	tPD = 2xd1+d0
		tPD0 = 2xd1+2xd0
C	$\overline{Q}$	tPD1 = 2xd1+d0
		tPD0 = 2 (d1+d0)
D (SETUP TIME)		= ISL2 d0
D (HOLD TIME)		= ISL1 d1+d0
CLOCK PULSE WIDTH		= 2xd0+d1

FUNCTION (TRUTH) TABLE

C	D	Q_{n+1}	$\overline{Q}_{n+1}$
↓	0	0	1
↓	1	1	0

Notes:
1. Data is transferred to the outputs on the negative-going edge of the clock.
2. Qn+1 = state after High-to-Low transition of C.

TIMING WAVEFORMS:



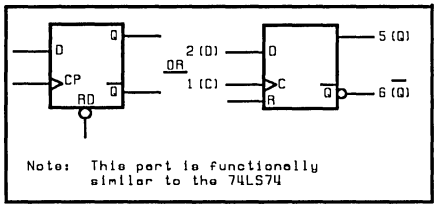
Notes:
Spikes can occur on $\overline{Q}$ during the propagation delay of E to $\overline{Q}$.

PROPAGATION DELAYS:

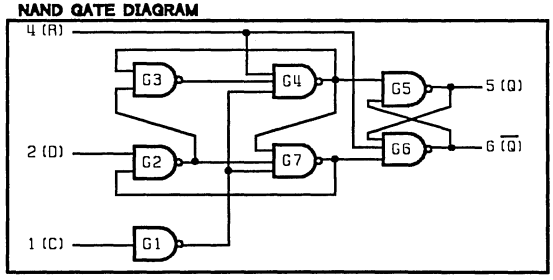
PARAMETER	FROM	TO	DELAY in ns
tPD0 ↓	C	Q	
	C	$\overline{Q}$	
tPD1 ↑	C	Q	
	C	$\overline{Q}$	
D (SETUP TIME)	ns Min		
D (HOLD TIME)	ns Min		
CLOCK WIDTH	ns Min		

Figure 3-2 D Flip-Flop (Negative Edge-Triggered)

LOGIC SYMBOL



NAND GATE DIAGRAM



PROPAGATION DELAY FORMULAS:

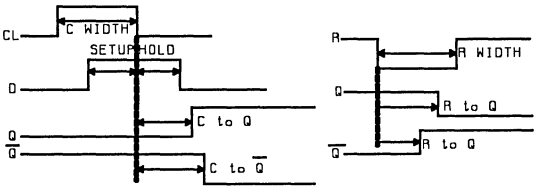
FROM	TO	FORMULAS
C	Q	tPD1 = 2d1+d0
		tPD0 = 2(d1+d0)
C	$\overline{Q}$	tPD1 = 2d1+d0
		tPD1 = 2(d1+d0)
R	Q	tPD0 = d1+d0
R	$\overline{Q}$	tPD1 = d1
D (SETUP TIME)		= d0
D (HOLD TIME)		= d1+d0
CL WIDTH (HIGH)		= d1+2d0
RESET WIDTH (LOW)		R->Q

FUNCTION (TRUTH) TABLE

C	D	Q_{n+1}	$\overline{Q}_{n+1}$
↘	0	0	1
↘	1	1	0
↘	0	0	1
↘	1	1	0

Notes:
1. Data is transferred to the outputs on the negative going edge of the clock; reset "R" (active-low) is asynchronous and independent of
2. Qn+1 = state after High-to-Low transition of C.
3. X = don't care

TIMING WAVEFORMS:

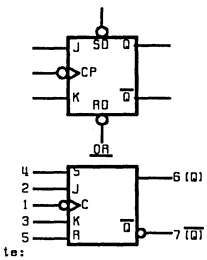


PROPAGATION DELAYS:

PARAMETER	FROM	TO	DELAY in ns
tPD1 ↗	C	Q	
	C	$\overline{Q}$	
	R	$\overline{Q}$	
tPD0 ↘	C	$\overline{Q}$	
	C	Q	
	R	Q	
D (SETUP TIME)	ns Min		
D (HOLD TIME)	ns Min		
MINIMUM CLOCK WIDTH	ns		
MINIMUM R WIDTH	ns		

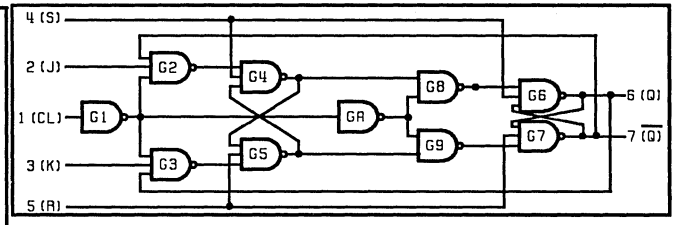
Figure 3-3 D Flip-Flop With Reset

LOGIC SYMBOL



Note:
This part is functionally similar to the 74LS109

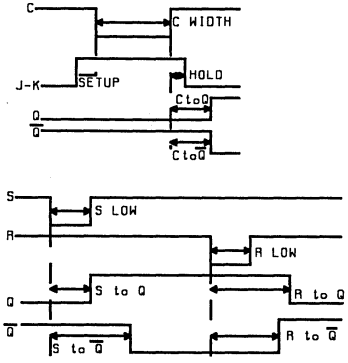
NAND GATE DIAGRAM



PROPAGATION DELAY FORMULAS:

FROM	TO	FORMULAS
C	Q, $\overline{Q}$	tPD0 = 3d0+2d1
		tPD1 = 3(d1+d0)
S	Q	tPD1 = d1
R	Q	tPD1 = d1
S	$\overline{Q}$	IfC = 0 tPD0 = d1+d0
R	Q	IfC = 1 tPD0 = 2(d1+d0
J-K SETUP TIME		0
J-K HOLD TIME		= d0
PULSE WIDTH C (MIN)		= d1+d0
S,R (MIN)		= (d1+d0)

TIMING WAVEFORMS:



FUNCTION (TRUTH) TABLE

C	J	K	S	R	Q_{n+1}	$\overline{Q}_{n+1}$
↑	0	0	1	1	Q_n	$\overline{Q}_n$
↑	1	0	1	1	1	0
↑	0	1	1	1	0	1
↑	1	1	1	1	Q	$\overline{Q}$
C	J	K	S	R	Q	$\overline{Q}$
X	X	X	1	0	0	1
X	X	X	0	1	1	0

Notes:
1. Master/slave data is pulse-triggered and enters the J-K flip-flop when the clock (C) goes Low; the entered data is transferred to the outputs on the positive going edge of "C". Set (S) and Reset (R) inputs are asynchronous, independent signals and, when either is active-low, the J-K and C inputs are overridden.
2. On+1 = next state of Q.
4. X = don't care

PROPAGATION DELAYS:

PARAMETER	FROM	TO	DELAY in ns
tPD0	C	Q&$\overline{Q}$	
	S	$\overline{Q}$	
	S	Q	
	R	Q	
	R	$\overline{Q}$	
tPD1	C	Q&$\overline{Q}$	
	S	Q	
	R	$\overline{Q}$	
J-K SETUP TIME		0	
J-K HOLD TIME		2 Min	
MINIMUM PULSE WIDTH		20	
MINIMUM PULSE WIDTH S&R (LOW)		16	

Figure 3-4 J-K Flip-Flop With Set and Reset

3.2 Flip–Flop Merging

Figure 3–5(a) shows the positive edge triggered D FF structure. By putting a 2–level AND/OR structure in front of the data input, the D FF can be steered from state to state.

Figure 3–5(b) shows such an input structure realized from a 2–level NAND gate section.

Figure 3–5(c) shows this "AND–OR" structure rolled inside of the flip–flop. The gating was merged with the flip–flop inwards to make a faster, composite function. Whereas this may appear as a trick to the uninitiated, this degree of flexibility allowed gate array designers to merge a multitude of logic into a fixed foundation. For highest efficiency, similar thinking allows the designer to break up decoders and multiplexors into their building blocks and generate only the pieces needed.

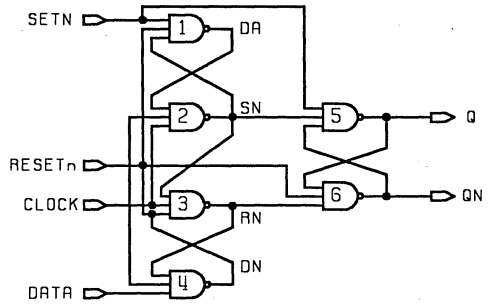


Figure 5-3 (a)
Positive Edge Triggered D-Flip-Flop with reset and set.

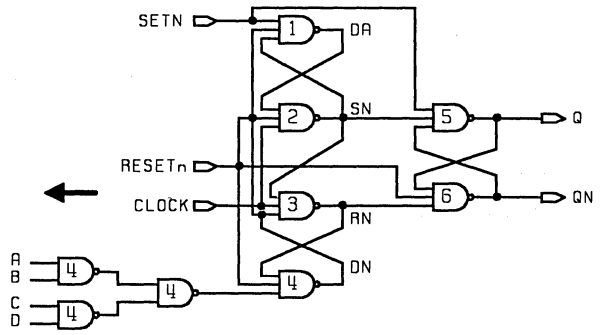


Figure 5-3 (b)
As above with input AND-OR function.

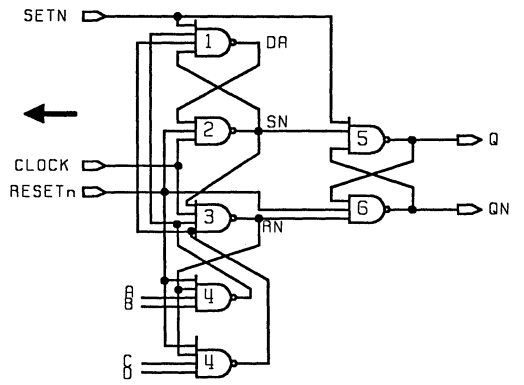


Figure 5-3 (c)
As above, with integral AND-OR input function.

Figure 3-5 Flip-Flop Merging

4. VME Bus Examples

4.1 Omnibyte VSBC20 Mailbox Interrupt Structure

One of the more popular uses for the PLHS501 is interfacing with 32-bit microprocessors. This section illustrates some of the ways the part has been used with the popular VME Bus. The Omnibyte Corporation manufactures many VME Bus products (as well as others) and was kind enough to release a portion of their VSBC20 board design as an example of using the PLHS501 in a very flexible, user configurable interrupt generation device. The VSBC20 employs two PLHS501 parts, as shown in Figure 4-1. One device is used largely as an address decoder, the other, which is the object of this Section, is the configurable interrupt generator. The target microprocessor here is a 25MHz 68020 and the application is interrupt generation. The explanation is in the words of Glenn Case, the designer:

"Following the design philosophy of giving the user as much flexibility as possible, the local interrupt structure of the VSBC20 is implemented in a PLD. It is impossible to "optimize" the assignment of the local interrupts to the interrupt levels of the processor since they are application specific. One system may want the Serial I/O and Parallel I/O to have higher levels than the Omnimodule Interrupts while yet another, using a SCSI Omnimodule, may want it to have higher level interrupts. Arbitrarily assigning and hard wiring these levels would unnecessarily constrain the use of the VSBC20 for any given application. By using the Signetics PLHS501, the entire logic to implement the interrupt structure fits into one PLD. Furthermore, the AMAZE software to program the part is available free from Signetics. The PLHS501 can be reprogrammed until the unused feedback gates are all used. So, the user can get the software free and change the interrupt levels a couple of times before having to replace the PLHS501 with a new part. This appendix describes how the PLHS501 is used and how to change the interrupt levels.

There are a total of 17 possible interrupt sources to the processor on the VSBC20. There are up to seven possible VMEbus interrupts, nine possible local interrupts, and a Front Panel Non-Maskable interrupt. The local interrupts include: ACFAIL*, SYSFAIL*, parity error, mailbox interrupt, two Omnimodule interrupts, 24 bit timer interrupt, Parallel I/O, and Serial I/O interrupts. Although ACFAIL*, SYSFAIL* and mailbox interrupts are generated by VMEbus, they are referred to as local interrupts because they are acknowledged locally. That is, no VMEbus IACK cycle takes place. The local interrupts are latched during an IACK cycle to "freeze" the state of the interrupts. This allows the correct acknowledgment of the interrupts. The ACFAIL*, SYSFAIL* and Front Panel Non-Maskable Interrupt are assigned to Level 7. The Front Panel NMI has the highest priority followed by ACFAIL*, parity error, and SYSFAIL*. The front Panel interrupt is acknowledged by an autovector while the other three generate a vector that is encoded as described in the Error Interrupt Vector CSR. The local interrupts for the mailbox interrupt, Omnimodule Interrupt 0, Omnimodule Interrupt 1, 24 bit timer interrupt, Parallel I/O Interrupt, and Serial I/O Interrupt have been assigned to levels 6-1 respectively. These interrupts are intended to be assigned by the user to the level best suited for the user's application. The mailbox interrupt uses the auto vector while the others provide interrupt acknowledge vectors. However, these may also be changed to generate autovectors. For example, if a unique Omnimodule is designed by the user and there is not enough room to provide an interrupt vector on the module, the PLD can be changed to issue an autovector instead of generating an IACK cycle to the Ominmodule. It is also possible to have two interrupts share the same level, although this seems unnecessary, since there are enough available interrupt levels.

The following examples illustrate how easy it is to change the local interrupt levels."

Example 1:

Put the P10 interrupt in level 4 and the Omnimodule 0 interring on level 2.

```
LIRQ4 = /LIRQPIO;                                    |->
LIRQ2 = /LIRQ0OM;                                    |->  change these
IACK0OM = /(/LIRQ0M*/A3*A2*/Al*//IACK*/BAS);    <-|     equations to
IACKPIIO = /(/LIRQPIO*A3*/A2*/Al*/IACK*/BAS);   <-|
```

Example 2:

Make both the Omnimodule Interrupt Autovectored instead of bus-vectored. (LITRQ0OM uses level 4 and LIRQIOM uses level 5.

```
AUTOVECTOR = [/FPNMIRQ*A3*A2*Al*/IACK*/BAS*RESET]
           + [/LIRQMBOX*A2*A2*/Al*/IACK*/BAS*RESET]
           + [/LIRQ0OM*A3*/A2*/Al*/IACK*/BAS*RESET]  <--   ADD
           + [/LIRQIOM*A3*/A2*Al*/IACK*/BAS*RESET]   <--   ADD
           + [AUTOVECTOR*              /BAS*RESET];

IACKIOM = /(O);   <--   change
IACK0OM = /(o);   <--   equation
```

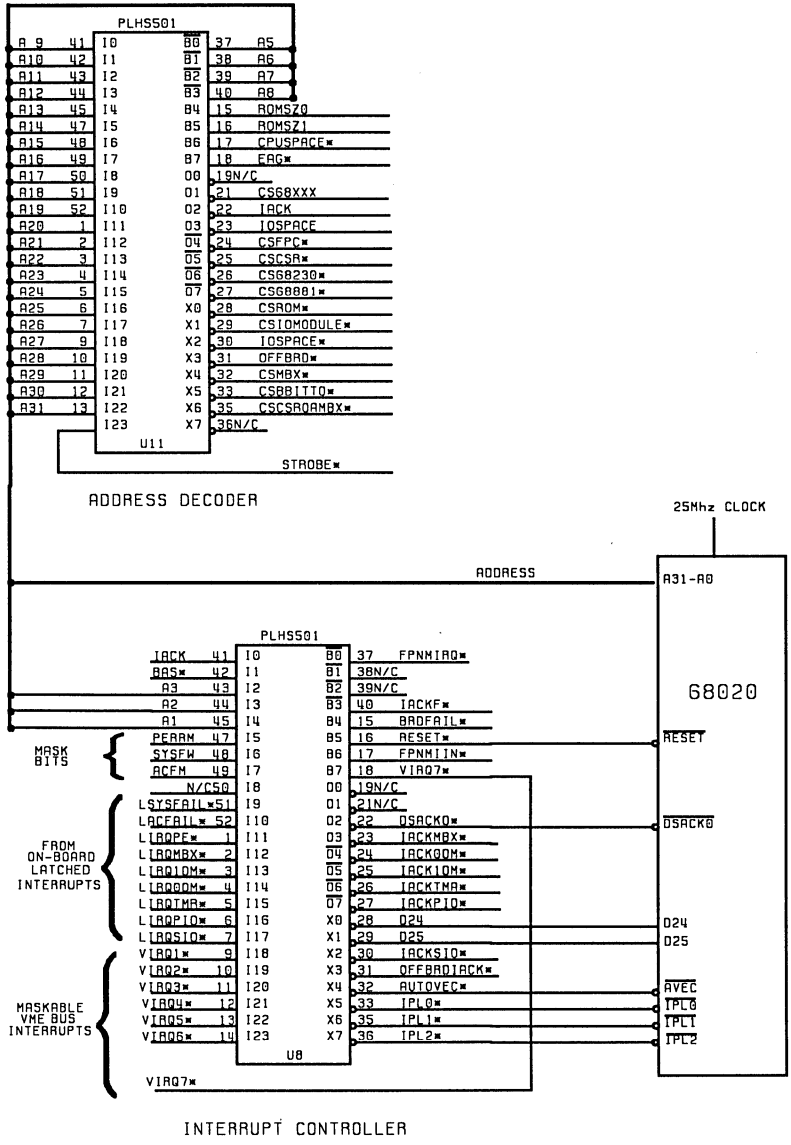


Figure 4–1 Portion of Omnibyte VSBC20 Highlighting PLHS501 Usage

```
File Name : vsbc20is
Date : 9/13/1988
Time : 9:4:2

#################### P I N   L I S T ####################

        Left                                          Right
   LABEL     ** FNC **PIN            PIN** FNC **    LABEL
VCC          ** +5V **  8-|        |-46 ** +5V **VCC
VIRQ1        ** I   **  9-|        |-45 ** I   **A1
VIRQ2        ** I   ** 10-|        |-44 ** I   **A2
VIRQ3        ** I   ** 11-|   P    |-43 ** I   **A3
VIRQ4        ** I   ** 12-|   L    |-42 ** I   **BAS
VIRQ5        ** I   ** 13-|   H    |-41 ** I   **IACK
VIRQ6        ** I   ** 14-|   S    |-40 ** /O  **IACKF
BRDFAIL      ** I   ** 15-|   5    |-39 ** /O  **N/C
RESET        ** I   ** 16-|   0    |-38 ** /O  **N/C
FPNMIIN      ** I   ** 17-|   1    |-37 ** /O  **N/C
VIRQ7        ** I   ** 18-|        |-36 ** O   **IPL2
N/C          ** O   ** 19-|        |-35 ** O   **IPL1
GND          ** 0V  ** 20-|        |-34 ** 0V  **GND


        Bottom                                        Top
   LABEL     ** FNC **PIN            PIN** FNC **    LABEL
N/C          ** O   ** 21-|        |- 7 ** I   **LIRQSIO
DSACK0       ** O   ** 22-|        |- 6 ** I   **LIRQPIO
N/C          ** O   ** 23-|        |- 5 ** I   **LIRQTMR
IACK0OM      ** /O  ** 24-|   P    |- 4 ** I   **LIRQ0OM
IACK1OM      ** /O  ** 25-|   L    |- 3 ** I   **LIRQ1OM
IACKTMR      ** /O  ** 26-|   H    |- 2 ** I   **LIRQMBX
IACKPIO      ** /O  ** 27-|   S    |- 1 ** I   **LIRQPE
D24          ** O   ** 28-|   5    |-52 ** I   **LACFAIL
D25          ** O   ** 29-|   0    |-51 ** I   **LSYSFAIL
IACKSIO      ** O   ** 30-|   1    |-50 ** I   **N/C
OFFBRDIACK   ** O   ** 31-|        |-49 ** I   **ACFM
AUTOVEC      ** O   ** 32-|        |-48 ** I   **SYSFM
IPL0         ** O   ** 33-|        |-47 ** I   **PERRM
```

Figure 4–2 PLHS501 Pinlist for VSBC20 Interrupt Structure

```
@DEVICE TYPE
PLHS501
@DRAWING   1155
@REVISION  A
@DATE                    9-9-88
@SYMBOL
@COMPANY                 OMNIBYTE CORP.
@NAME                    GLENN CASE
@DESCRIPTION      VSBC20 INTERRUPT STRUCTURE PLD
@INTERNAL NODE
LIRQ7 ALLIRQ7 AHIACKF AHFPNMIRQ AUTOVECTOR FPNMIRQ

@COMMON PRODUCT TERM
LIRQ6 = /LIRQMBX;          "LIRQ6 goes high when LIRQMBX goes low"
LIRQ5 = /LIRQ1OM;
LIRQ4 = /LIRQ0OM;
LIRQ3 = /LIRQTMR;
LIRQ2 = /LIRQPIO;
LIRQ1 = /LIRQSIO;

@I/O DIRECTION
DB4 = 0;
DB5 = 0;
DB6 = 0;
DB7 = 0;
XE0 = /IACKF;
XE1 = 1;
XE2 = 1;
XE3 = 1;
OE1 = /IACKF;
OE2 = 1;
OE3 = 1;

@I/O STEERING
@LOGIC EQUATION

LIRQ7 = [/LACFAIL * ACFM]                "LIRQ7 goes high when"
      + [/LSYSFAIL * SYSFM * BRDFAIL]
      + [/LIRQPE * PERRM];

ALLIRQ7 = /[/[/LACFAIL * ACFM]
        * /[/LSYSFAIL * SYSFM * BRDFAIL]
        * /[/LIRQPE * PERRM]];

AHFPNMIRQ = [/FPNMIIN  */IACK * RESET]
          + [AHFPNMIRQ */IACK * RESET]
```

```
          + [AHFPNMIRQ * IACK *  BAS * RESET]
          + [AHFPNMIRQ * IACK * /BAS * /A1 * RESET]
          + [AHFPNMIRQ * IACK * /BAS * /A2 * RESET]
          + [AHFPNMIRQ * IACK * /BAS * /A3 * RESET]
        + [AHFPNMIRQ * IACK * /BAS * A3 * A2 * A1 * /AUTOVECTOR * RESET];

FPNMIRQ = /(AHFPNMIRQ);

IPL0: XR1=/VIRQ7
         + LIRQ7
         +/FPNMIRQ
         +/LIRQ7 * /LIRQ6 *  VIRQ7 *  VIRQ6 *  LIRQ5
         + /LIRQ7 * /LIRQ6 *  VIRQ7 *  VIRQ6 * /VIRQ5
     + /LIRQ7 * /LIRQ6 * /LIRQ5 * /LIRQ4 * VIRQ7 * VIRQ6 * VIRQ5 *  VIRQ4

                    *  LIRQ3
     +  /LIRQ7 * /LIRQ6 * /LIRQ5 * /LIRQ4 * VIRQ7 * VIRQ6 * VIRQ5 *  VIRQ4
                    * /VIRQ3
     +  /LIRQ7 * /LIRQ6 * /LIRQ5 * /LIRQ4 * /LIRQ3 * /LIRQ2 * VIRQ7 * VIRQ6
                    *  VIRQ5 *  VIRQ4 *  VIRQ3 * VIRQ2 * LIRQ1
     +  /LIRQ7 * /LIRQ6 * /LIRQ5 * /LIRQ4 * /LIRQ3 * /LIRQ2 * VIRQ7 * VIRQ6
                *  VIRQ5 *  VIRQ4 * VIRQ3 * VIRQ2 * /VIRQ1;
XR2 = 1;

IPL1: XR1 = /VIRQ7
          +  LIRQ7
          + /FPNMIRQ
          + /LIRQ7 *  VIRQ7 * LIRQ6
          + /LIRQ7 *  VIRQ7 * /VIRQ6
      + /LIRQ7 * /LIRQ6 * /LIRQ5 * /LIRQ4 * VIRQ7 * VIRQ6 * VIRQ5 * VIRQ4
                    * LIRQ3
      + /LIRQ7 * /LIRQ6 * /LIRQ5 * /LIRQ4 * VIRQ7 * VIRQ6 * VIRQ5 * VIRQ4
                    * /VIRQ3
      + /LIRQ7 * /LIRQ6 * /LIRQ5 * /LIRQ4 */LIRQ3 * VIRQ7 * VIRQ6 *  VIRQ5
                    * VIRQ4 * VIRQ3 *  LIRQ2
      + /LIRQ7 * /LIRQ6 * /LIRQ5 * /LIRQ4 */LIRQ3 * VIRQ7 * VIRQ6 *  VIRQ5
                    * VIRQ4 * VIRQ3 * /VIRQ2;
XR2 = 1;

IPL2: XR1 = /VIRQ7
          +  LIRQ7
          + /FPNMIRQ
          + /LIRQ7 *  VIRQ7 *  LIRQ6
          + /LIRQ7 *  VIRQ7 * /VIRQ6
          + /LIRQ7 * /LIRQ6 *  VIRQ7 * VIRQ6 * LIRQ5
          + /LIRQ7 * /LIRQ6 *  VIRQ7 * VIRQ6 * /VIRQ5
          + /LIRQ7 * /LIRQ6 * /LIRQ5 * VIRQ7 * VIRQ6 * VIRQ5 * LIRQ4
```

```
      + /LIRQ7 * /LIRQ6 * /LIRQ5 * VIRQ7 * VIRQ6 * VIRQ5 * /VIRQ4;
XR2 = 1;

IACKF = /([A3 * A2 * A1 * FPNMIRQ * /LACFAIL  * IACK * /BAS]
      +   [A3 * A2 * A1 * FPNMIRQ * /LSYSFAIL * IACK * /BAS]
      +   [A3 * A2 * A1 * FPNMIRQ * /LIRQPE   * IACK * /BAS]);

OFFBRDIACK:XR1=/LIRQ7 * A3 * A2 * A1 * IACK * /BAS * FPNMIRQ * /AUTOVECTOR
               + /LIRQ6 *  A3 *  A2 * /A1 * IACK * /BAS
               + /LIRQ5 *  A3 * /A2 *  A1 * IACK * /BAS
               + /LIRQ4 *  A3 * /A2 * /A1 * IACK * /BAS
               + /LIRQ3 * /A3 *  A2 *  A1 * IACK * /BAS
               + /LIRQ2 * /A3 *  A2 * /A1 * IACK * /BAS
               + /LIRQ1 * /A3 * /A2 *  A1 * IACK * /BAS;
XR2 = 1;

AUTOVECTOR = [/FPNMIRQ  *  A3 *  A2 *  A1 * IACK * /BAS * RESET]
           + [/LIRQMBX *  A3 *  A2 * /A1 * IACK * /BAS * RESET]
           + [ AUTOVECTOR                       * /BAS * RESET];

AUTOVEC: XR1 = AUTOVECTOR;
XR2 = 1;

D24: XR1 = /LACFAIL * ACFM
         + /LIRQPE  * PERRM;
XR2 = 1;

D25: XR1 = /LACFAIL * ACFM
         +  LIRQPE  *  PERRM * /LSYSFAIL * SYSFM
         +            /PERRM * /LSYSFAIL * SYSFM;
XR2 = 1;

IACK1OM = /(/LIRQ1OM *  A3 * /A2 *  A1 * IACK * /BAS);

IACK0OM = /(/LIRQ0OM *  A3 * /A2 * /A1 * IACK * /BAS);

IACKTMR = /(/LIRQTMR * /A3 *  A2 *  A1 * IACK * /BAS);

IACKPIO = /(/LIRQPIO * /A3 *  A2 * /A1 * IACK * /BAS);

IACKSIO = /(/LIRQSIO * /A3 * /A2 *  A1 * IACK * /BAS);

DSACK0 = BAS;
```

Figure 4-3 VSCBC20IS .BEE File

4.2 VME Bus EPROM Interface

The idea for this VMEbus EPROM board came from WIRELESS WORLD CIRCUIT IDEAS, January, 1988. The implementation was done by a Philips' FAE, John McNally.

The board contains two banks of EPROMs. Each bank consists of either two 27128's or two 27256's; each of which can be enabled by comparing the address location of the board. Decoding three other address bits selects which of the banks is accessed. A 4-bit shift register combined with four jumpers provide wait states.

The circuit drawing was entered on to a PC using FutureNet Dash, a schematic capture package (Figures 4-4, 4-5 and 4-6). It was then converted to logic equations using AMAZE (Figure 4-9) and then assembled into a PLHS501.

This application, which needs eight ICs, used forty-four of the available seventy-two NAND Foldback Terms and forty of the available fifty-two pins. As the PLHS501 contains no registers, an edge triggered D type Flip-Flop was designed using NAND gates and this is used as a soft macro in order to implement the shift register function (Figure 4-6).

As suggested in the original article the circuit could be expanded to access up to eight ROM banks (Figure 4-8). This was achieved by editing the logic equation file and adding extra equations (Figure 4-9). Modifying the drawing, although fairly easy to do, was not considered necessary as the object was to design with PML and not TTL. The expanded circuit would require another three TTL IC packages, bringing the total to eleven. The number of foldback terms increased to fifty-five, with the number of pins rising to fifty. Figure 4-10 shows the pinout of both versions.

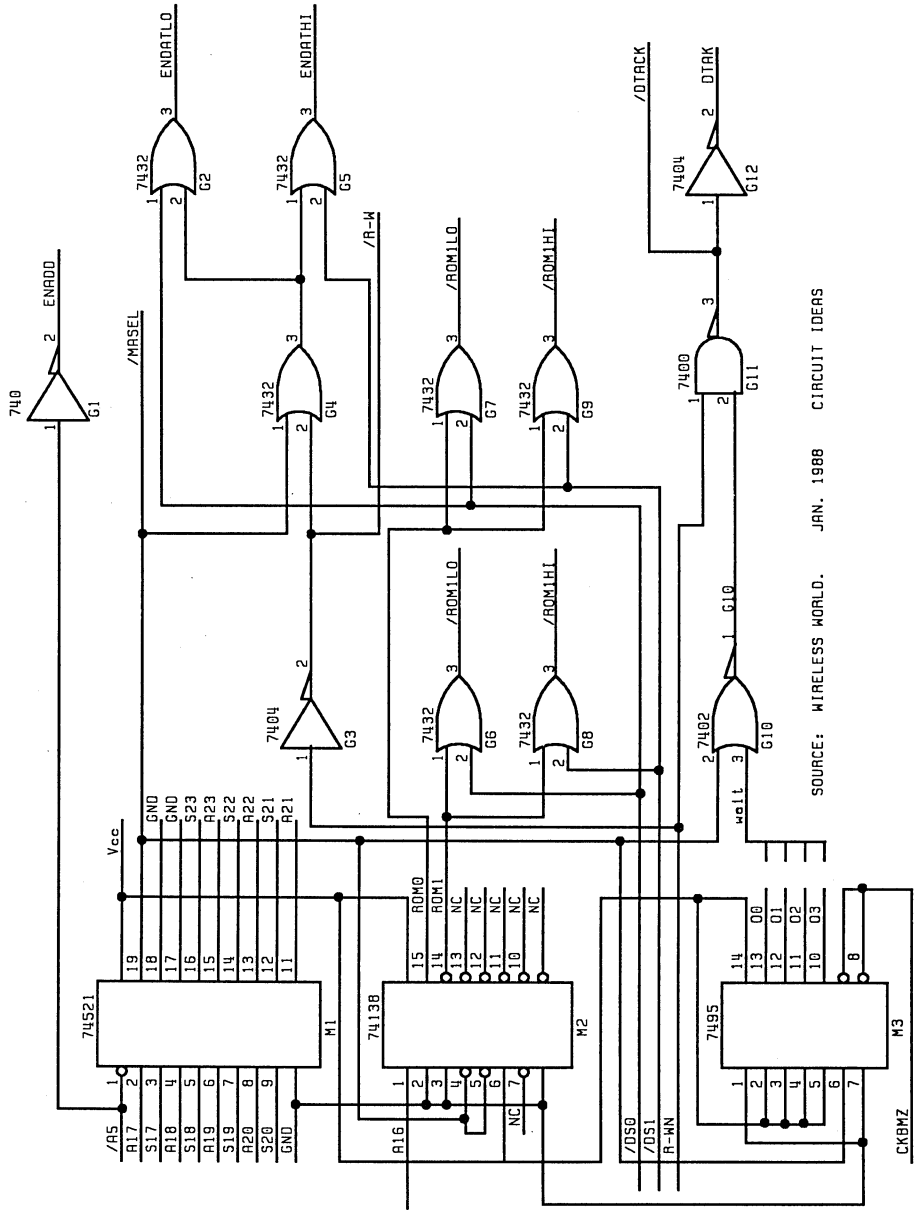


Figure 4–4 VME – EPROM Interface

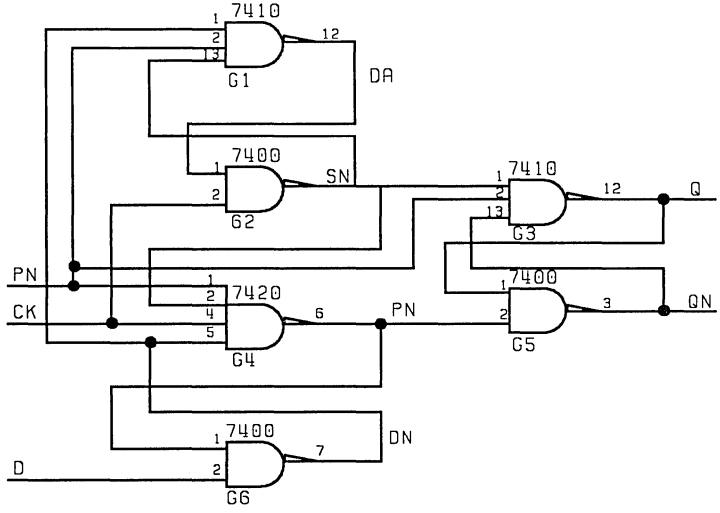


Figure 4–5 Edge–Triggered D Flip–Flop (DFFS)

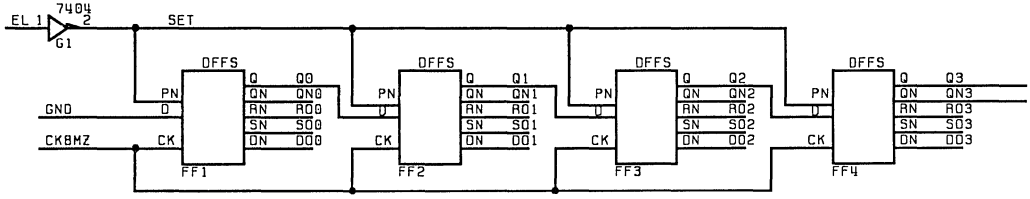
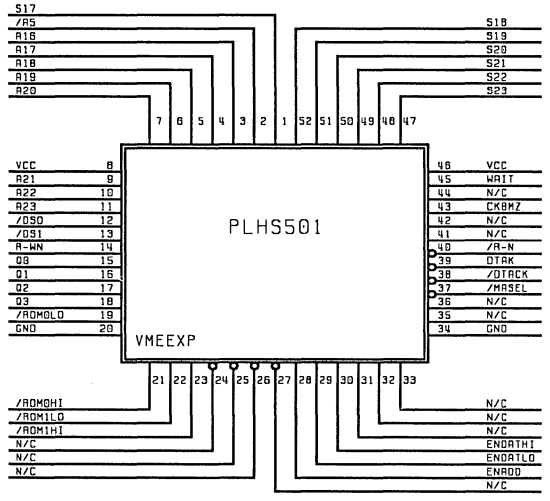


Figure 4–6 4–Bit Shifter (7495)

2 ROM BANKS AS ORIGINAL CIRCUIT.
REPLACES 8 PACKAGES - USES 46 FOLDBACK TERMS



EXPANDED TO 8 ROM BANKS
REPLACES 11 PACKAGES - USES 55 FOLDBACK TERMS

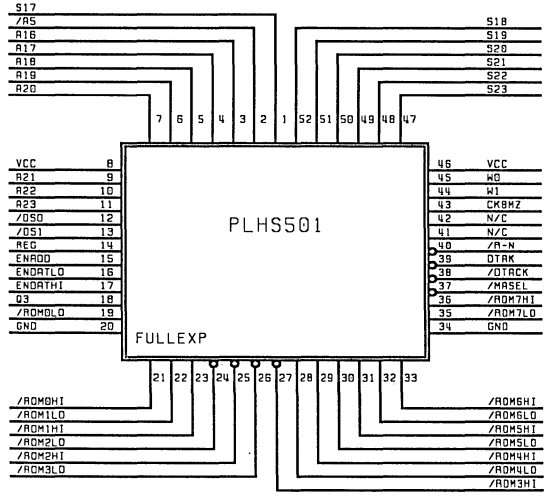


Figure 4–7 VMEEXP and FULLEXP

```
File Name : VMEEXP
Date : 2/13/1988
Time : 10:23:5

********************* P I N    L I S T *********************
        Left                                          Right
   LABEL     ** FNC **PIN                                LABEL
VCC          ** +5V **  8-|          |-46 ** +5V   **VCC
A21          ** I   **  9-|          |-45 ** I     **WAIT
A22          ** I   ** 10-|          |-44 ** I     **N/C
A23          ** I   ** 11-|     P    |-43 ** I     **CKBMZ
/DS0         ** I   ** 12-|     L    |-42 ** I     **N/C
/DS1         ** I   ** 13-|     H    |-41 ** I     **N/C
R-WN         ** I   ** 14-|     S    |-40 ** /O    **/R-W
Q0           ** O   ** 15-|     5    |-39 ** /O    **DTAK
Q1           ** 0   ** 16-|     0    |-38 ** /O    **/DTACK
Q2           ** 0   ** 17-|     1    |-37 ** /O    **/MASEL
Q3           ** 0   ** 18-|          |-36 ** 0     **N/C
/ROM0LO      ** O   ** 19-|          |-35 ** O     **N/C
GND          ** 0V  ** 20-|          |-34 ** 0V    **GND

       Bottom                                          Top
   LABEL                                               LABEL
/ROM0HI      ** 0   ** 21-|          |- 7 ** I     **A20
/ROM1LO      ** O   ** 22-|          |- 6 ** I     **A19
/ROM1HI      ** 0   ** 23-|          |- 5 ** I     **A18
N/C          ** /0  ** 24-|     P    |- 4 ** I     **A17
N/C          ** /O  ** 25-|     L    |- 3 ** I     **A16
N/C          ** /O  ** 26-|     H    |- 2 ** I     **/A5
N/C          ** /0  ** 27-|     S    |- 1 ** I     **S17
ENADD        ** O   ** 28-|     5    |-52 ** I     **S18
ENDATLO      ** O   ** 29-|     0    |-51 ** I     **S19
ENDATHI      ** 0   ** 30-|     1    |-50 ** I     **S20
N/C          ** 0   ** 31-|          |-49 ** I     **S21
N/C          ** 0   ** 32-|          |-48 ** I     **S22
N/C          ** 0   ** 33-|          |-47 ** I     **S23
```

Figure 4-8 VMEEXP PLHS501 Pinlist

```
File Name:  VMEEXP
Date:  2/13/1988
Time:  10:23:41

@DEVICE TYPE
     PLHS501
@DRAWING
     VMEEXP.DWG
@REVISION
@DATE
     2/12/1988
@SYMBOL
@COMPANY
@NAME
     VMEEXP
@DESCRIPTION
@INTERNAL NODE
     RO3 SO3 DO3 RO2 SO2
     DO2 Q0  RO1 SO1 DO1
     RO0 SO0 DO0
@I/O DIRECTION
     DB5 = 1 ;
     DB6 = 1 ;
     DB7 = 1 ;
     OE0 = 1 ;
     OE1 = 1 ;
@I/O STEERING
@LOGIC EQUATION
     RO3 = (/(((//MASEL)*SO3*CK8MZ*DO3)) ;
     SO3 = (/(CKBMZ*(/(SO3*DO3*(//MASEL))))) ;
     DO3 = (/(Q2*RO3)) ;
     RO2 = (/(((//MASEL)*SO2*CKBMZ*DO2)) ;
     SO2 = (/(CKBMZ*(/(SO2*DO2(//MASEL))))) ;
     DO2 = (/(Q1*RO2)) ;
     RO1 = (/(((//MASEL)*SO1*CKBMZ*DO1)) ;
     SO1 = (/(CKBMZ*(/(SO1*DO1*(//MASEL))))) ;
     DO1 = (/(Q0*ro1)) ;
     RO0 = (/(((//MASEL)*SO0*CKBMZ*DO0)) ;
     SO0 = (/(CKBMZ*(/(SO0*DO0*(//MASEL))))) ;
     DO0 = (/(Q*RO0)) ;
     /ROM0LO = (/DS0+(/(/A16*/0*1*//MASEL))) ;
     /ROM0HI = (/DS1+(/(/A16*/0*1*//MASEL))) ;
     /ROM1LO = (/DS0+(/(A16*/0*1*//MASEL))) ;
     /ROM1HI = (/DS1+(/(A16*/0*1*//MASEL))) ;
     Q0 = (/((/(RO0*Q0))*SO0*(//MASEL))) ;
     Q1 = (/((/(RO1*Q1))*SO1*(//MASEL))) ;
     Q2 = (/((/(RO2*Q2))*SO2*(//MASEL))) ;
```

```
Q3 = (/(((/(RO3*Q3))*SO3*(//MASEL))) ;
/MASEL =/(/([/((A17*S17+/A17*/S17)*(A18*S18+/A18*/S18)*(A19*S19+
            /A19*/S19)*(A20*S20+/A20*/S20)*(A21*S21+/A21*/S21)*
            (A22*S22+/A22*/S22)*(A23*S23+/A23*/S23)*//A5)]))  ;
/DTACK = /(((/(/MASEL+WAIT))*R-WN) ;
 DTACK = /(/DTACK) ;
/R-W = /(R-WN) ;
ENADD = (//A5) ;
ENDATLO = ((/R-W+/MASEL)+/DS0) ;
ENDATHI = (/DS1+(/R-W+/MASEL)) ;
```

Figure 4-9 VMEEXP PLHS501 .BEE File

```
File Name:  FULLEXP
Date : 2/13/1988
Time : 10:11:28

********************* P I N    L I S T *********************
        Left                                          Right
   LABEL     ** FNC **PIN                              LABEL
VCC          ** +5V **  8-|          |-46 ** +5V   **VCC
A21          ** I   **  9-|          |-45 ** I     **W0
A22          ** I   ** 10-|          |-44 ** I     **W1
A23          ** I   ** 11-|     P    |-43 ** I     **CKBMZ
/DS0         ** I   ** 12-|     L    |-42 ** I     **N/C
/DS1         ** I   ** 13-|     H    |-41 ** I     **N/C
R-WN         ** I   ** 14-|     S    |-40 ** /O    **/R-W
REG          ** O   ** 15-|     5    |-39 ** /O    **DTAK
ENADD        ** O   ** 16-|     0    |-38 ** /O    **/DTACK
ENDATLO      ** O   ** 17-|     1    |-37 ** /O    **/MASEL
ENDATHI      ** O   ** 18-|          |-36 ** O     **ROM7HI
/ROM0LO      ** O   ** 19-|          |-35 ** O     **ROM7LO
GND          ** 0V  ** 20-|          |-34 ** 0V    **GND

       Bottom                                         Top
   LABEL                                              LABEL
/ROM0HI      ** O   ** 21-|          |- 7 ** I     **A20
/ROM1LO      ** O   ** 22-|          |- 6 ** I     **A19
/ROM1HI      ** O   ** 23-|          |- 5 ** I     **A18
/ROM1LO      ** /O  ** 24-|     P    |- 4 ** I     **A17
/ROM1HI      ** /O  ** 25-|     L    |- 3 ** I     **A16
/ROM3LO      ** /O  ** 26-|     H    |- 2 ** I     **/A5
/ROM3HI      ** /O  ** 27-|     S    |- 1 ** I     **S17
/ROM4LO      ** O   ** 28-|     5    |-52 ** I     **S18
/ROM4HI      ** O   ** 29-|     0    |-51 ** I     **S19
/ROM5LO      ** O   ** 30-|     1    |-50 ** I     **S20
/ROM5HI      ** O   ** 31-|          |-49 ** I     **S21
/ROM6LO      ** O   ** 32-|          |-48 ** I     **S22
/ROM6HI      ** O   ** 33-|          |-47 ** I     **S23
```

Figure 4-10 FULLEXP Pinlist

```
File Name :  FULLEXP
Date:  2/13/1988
Time:  10:11:30

@DEVICE
     PLHS501
@DRAWING
     VMEEXP.DWG
@REVISION
@DATE

     2/12/1988
@SYMBOL
@COMPANY
@NAME
     VMEEXP
@DESCRIPTION
@INTERNAL NODE
     RO3 SO3 DO3 RO2 SO2
     DO2 RO1 SO1 DO1 RO0
     SO0 DO0
     Q0 Q1 Q2 Q3
@I/O DIRECTION
     DB4 = 1 ;
     DB5 = 1 ;
     DB6 = 1 ;
     DB7 = 1 ;
     OE0 = 1 ;
     OE1 = 1 ;
     OE2 = 1 ;
     OE3 = 1 ;
     XE0 = 1 ;
     XE1 = 1 ;
     XE2 = 1 ;
     XE3 = 1 ;
@STEERING
     SO  = Q ;
     S1  = Q ;
     S2  = Q ;
     S3  = Q ;
@LOGIC EQUATION
     RO3 = (/(((//MASEL)*SO3*CKBMZ*DO3)) ;
     SO3 = (/(CKBMZ*(/(SO3*DO3*(//MASEL))))) ;
     DO3 = (/(Q2*RO3)) ;
     RO2 = (/(((//MASEL)*SO2*CKBMZ*DO2)) ;
     SO2 = (/(CKBMX*(/SO2*DO2*(//MASEL))))) ;
     DO2 = (/(Q1*RO2)) ;
```

```
RO1 = (/(((//MASEL)*SO1*CKBMZ*DO1)) ;
SO1 = (/(CKBMZ*(/(SO1*DO1*(//MASEL))))) ;
DO1 = (/(Q0*RO1)) ;
RO0 = (/(((//MASEL)*SO0*CKBMZ*DO0)) ;
SO0 = (/(CKBMZ*(/SO0*DO0*(//MASEL))))) ;
DO0 = (/(Q*RO0)) ;
/ROM0LO = (/DO0+/(/A16*/A17*/A18*//MASEL)) ;
/ROM0HI = (/DS1+/(/A16*/A17*/A18*//MASEL)) ;
/ROM1L0 = (/DS0+/(A16+/A17*/A18*//MASEL)) ;
/ROM1HI = (/DS1+/(A16*/A17*/A18*//MASEL)) ;
/ROM2L0 = /(/(/DS0+/(/A16*A17*/A18*//MASEL))) ;
/ROM2HI = /(/(/DS1+/(/A16*A17*/A18*//MASEL))) ;
/ROM3L0 = /(/(/DS0+/(A16*A17*/A18*//MASEL))) ;
/ROM3HI = /(/(/DS1*/(A16*A17*/A18*//MASEL))) ;
/ROM4L0 = (/DS0+/(A16*/A17*A18*//MASEL)) ;
/ROM4HI = (/DS1+/(A16*/A17*/A18*//MASEL)) ;
/ROM5L0 = (/DS0+/(A16*/A17(A18*//MASEL)) ;
/ROM5HI = (/DS1+/(A16*/A17*A18*//MASEL)) ;
/ROM6L0 = (/DS0+/(A16*A17*A18*//MASEL)) ;
/ROM6HI = (/DS1+/(A16*A17*A18*//MASEL)) ;
/ROM7LO = (/DS0+/(A16*A17*A18*//MASEL)) ;
/ROM7HI = (/DS1+/(A16*A17*A18*//MASEL)) ;
 ENADD  = (//A5) ;
ENDATLO = ((/R-W+/MASEL)+/DS0) ;
ENDATHI = (/DS1+(/R-W+/MASEL)) ;
Q0 = /((/(RO0*Q0))*SO0*(//MASEL)) ;
Q1 = /((/(RO1*Q1))*SO1*(//MASEL)) ;
Q2 = /((/(RO2*Q2))*SO2*(//MASEL)) ;
Q3 = /((/(RO3*Q3))*SO3*(//MASEL)) ;
/MASEL =/(/([/((A17*S17+/A17*/A17*/S17*(A18*S18+/A18*/S18)
              *(A19*S19+/A19+/S19*(A20*S20*/S20*(A21*S21
              +/A21*/S21)*(A22*S22+/A22*/S22)*(A23**S23
              +(A23*S23)*//A5)])) ;
/DTACK = /(((/(/MASEL+/((/Q0*W0*/W1)+(/Q1*W0*/W1)+(/Q2*/W0*W1)
              +(/Q3*W0*W1))))*R-WN) ;
 DTAK  = /(DTACK) ;
/R-W = /(R-WN) ;
REG = Q0*Q1*Q2*Q3 ;
```

Figure 4-11 FULLEXP PLHS501 .BEE File

5. Micro Channel Interface

IBM's new Micro Channel Architecture (MCA) bus implements new features not found on the XT/AT bus. One new requirement for adapter designers is that of Programmable Option Select (POS) circuitry. It allows system software to configure each adapter card upon power on, thereby eliminating option select switches or jumpers on the main logic board and on adapter cards.

Each adapter card slot has its own unique -CDSETUP signal routed to it. This allows the CPU to interrogate each card individually upon power up. By activating a card's -CDSETUP line along with appropriate address and control lines two unique 8 bit ID numbers are first read from the adapter. Based upon the ID number, the system then writes into the card's option latches configuration information that had been stored in the system's CMOS RAM. The CPU also activates POS latch address 102h bit 0, which is designated as a card enable bit.

If a new card is added to the system, an auto-configuration utility will be invoked. Each adapter card has associated with it a standardized Adapter Description File with filename of @XXXX.ADF, where XXXX is the hex ID number of the card. The configuration utility prompts the user according to the text provided in the .ADF file and updates the card's latches and the system's CMOS RAM.

IBM reserves 8 addresses for byte-wide POS latches, however, depending on the card's function, not all addresses need to be used. In addition, of those addresses that are used, only the bits used need to be latched. The first two addresses which are reserved for reading the ID bytes, and bit 0 of the third address, which is defined as a card enable bit, are mandatory. Some of the remaining bits of the third address are suggested by IBM to be used as inputs to an IO or memory address comparator to provide for alternate card addresses. Many adapter cards will not use more than these three POS locations.

The following example describes an implementation of POS circuitry realized in a PLHS501. It uses only 56 of the possible 72 internal foldback NAND gates and only a portion of the device pins, allowing additional circuitry to be added. Figure 5–1 shows a block diagram of the circuit, and Figure 5–3 and 5–4 are the AMAZE files. Pins labeled D0O–D7O must be connected externally to pins D0I–D7I. They also must be connected through a 74F245 transceiver to the Micro Channel. External transceiver direction and enable control is provided for by circuitry within the PLHS501. The external transceiver may also be used by other devices on the adapter card.

In this application, edge triggered registers are not required and therefore should not be used as transparent latches use fewer NAND gates to implement. Figure 5–2 shows the various latch circuits described by the AMAZE equations. POS byte 2 was made using four of the /B device pins and four of the B pins. Notice however, from Figure 5–2(B) that the bits on the /B pins used the complement of the input pin, thereby implementing a noninverting latch. Also, all 8 bits of this byte were brought to output pins. If some of the bits are not used by external circuitry, then the specific bit latch may not be needed or may be constructed entirely from foldback NAND gates freeing additional pins.

An external F521 may by added to provide for IO address decoding. As the MCA bus requires all 16 bits of the IO address to be decoded, 8 bits may be assigned to the F521 and 8 bits to the 501. Bit fields decoded in the 501 may be done so in conjunction with bits from POS byte 2 to provide for alternate IO addressing. Additionally, some of the available 501 outputs may be used as device enables for other devices on the card.

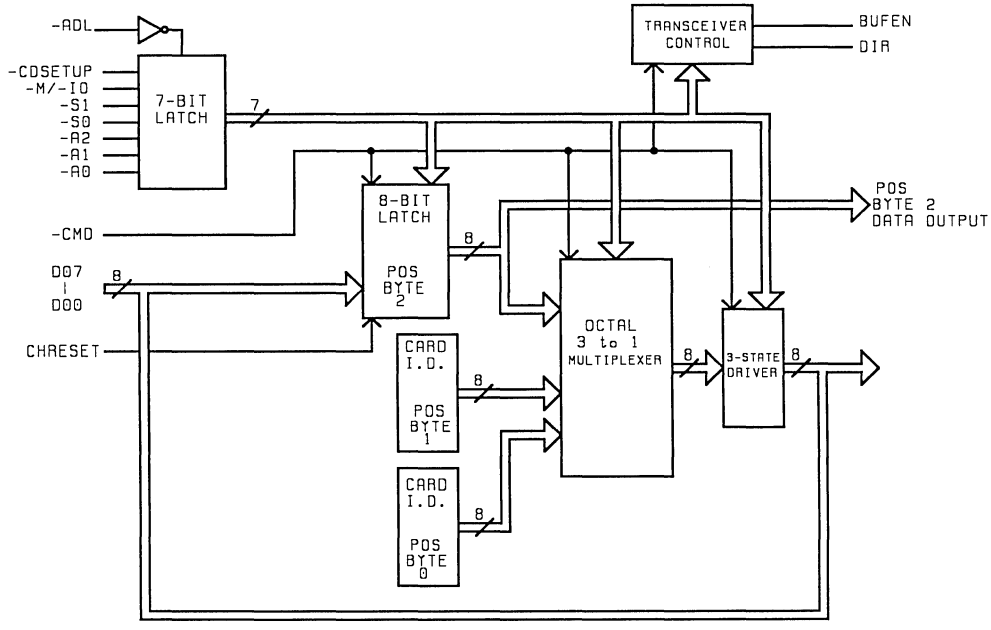


Figure 5–1 Block Diagram of Basic POS Implementation in PLHS501

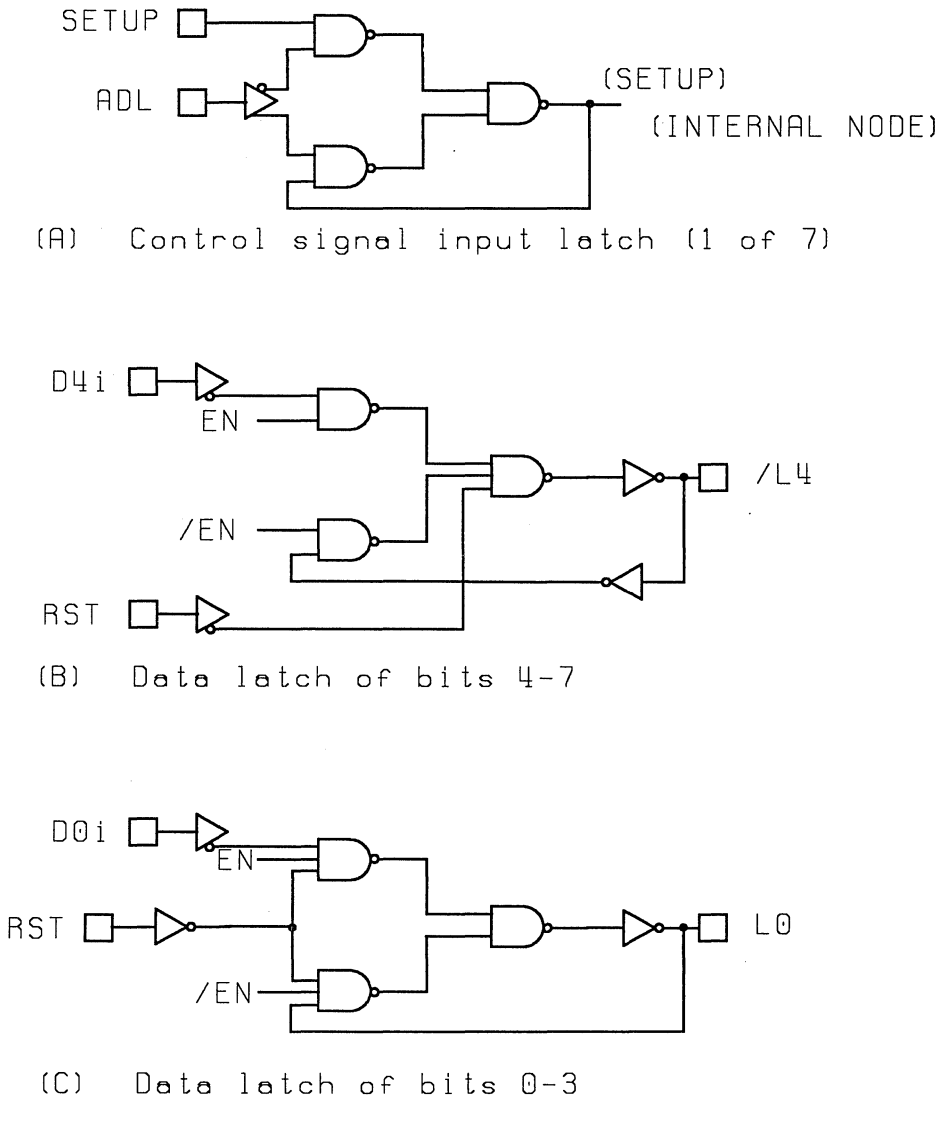


Figure 5-2 Latches used in MCA Interface

```
File Name : MCPOSREG
Date : 5/31/1988
Time : 11:50:2

 ##################### P I N   L I S T ######################

        Left                                          Right
   LABEL     ** FNC **PIN            PIN** FNC **     LABEL
VCC          ** +5V **  8-|        |-46 ** +5V **VCC
N/C          ** I   **  9-|        |-45 ** I   **D4I
N/C          ** I   ** 10-|        |-44 ** I   **D3I
N/C          ** I   ** 11-|   P    |-43 ** I   **D2I
N/C          ** I   ** 12-|   L    |-42 ** I   **D1I
N/C          ** I   ** 13-|   H    |-41 ** I   **D0I
N/C          ** I   ** 14-|   S    |-40 ** /O  **L3
/L4          ** O   ** 15-|   5    |-39 ** /O  **L2
/L5          ** O   ** 16-|   0    |-38 ** /O  **L1
/L6          ** O   ** 17-|   1    |-37 ** /O  **L0
/L7          ** O   ** 18-|        |-36 ** O   **D7O
N/C          ** O   ** 19-|        |-35 ** O   **D6O
GND          ** 0V  ** 20-|        |-34 ** 0V  **GND


        Bottom                                        Top
   LABEL     ** FNC **PIN            PIN** FNC **     LABEL
BUFEN        ** O   ** 21-|        |- 7 ** I   **SS1
N/C          ** O   ** 22-|        |- 6 ** I   **SS0
N/C          ** O   ** 23-|        |- 5 ** I   **SETUP
IOWB         ** /O  ** 24-|   P    |- 4 ** I   **CMD
N/C          ** /O  ** 25-|   L    |- 3 ** I   **A2
N/C          ** /O  ** 26-|   H    |- 2 ** I   **A1
N/C          ** /O  ** 27-|   S    |- 1 ** I   **A0
D0O          ** O   ** 28-|   5    |-52 ** I   **MIO
D1O          ** O   ** 29-|   0    |-51 ** I   **ADL
D2O          ** O   ** 30-|   1    |-50 ** I   **RST
D3O          ** O   ** 31-|        |-49 ** I   **D7I
D4O          ** O   ** 32-|        |-48 ** I   **D6I
D5O          ** O   ** 33-|        |-47 ** I   **D5I
```

Figure 5-3 PLHS501 MCPOSREG Pinlist

```
File Name : MCPOSREG
Date : 5/31/1988
Time : 11:50:17

@DEVICE TYPE
   PLHS501
@DRAWING
@REVISION
@DATE
@SYMBOL
@COMPANY
@NAME
@DESCRIPTION
     Basic Programmable Option Select circuitry
         for a Micro Channel Adaptor card

@INTERNAL NODE
   /setupl,/miol,a0l,all,/a2l,ss0l,ssll;
   /en,outen,/iow;
@COMMON PRODUCT TERM
   read0 = (setupl*/ssll*ss0l*miol*/cmd*a2l*/all*/a0l);
   readl = (setupl*/ssll*ss0l*miol*/cmd*a2l*/all* a0l);
   read2 = (setupl*/ssll*ss0l*miol*/cmd*a2l* all*/a0l);
"
   NOTE: In the above equations, setupl, miol and a2l all should be
   preceded by a slash (/). The slash was omitted to correct for
   a mapping error in AMAZE 1.65 when using active low internal node
   definitions in common product terms.
"

   b7h = 0;  " Define high ID byte "
   b6h = 1;  "   (POS byte #1)     "
   b5h = 1;  "      7E hex         "
   b4h = 1;
   b3h = 1;
   b2h = 1;
   blh = 1;
   b0h = 0;

   b7l = 1;   " Define low ID byte "
   b6l = 1;   "    (POS byte #0)   "
   b5l = 1;   "       FF hex       "
   b4l = 1;
   b3l = 1;
   b2l = 1;
   bll = 1;
   b0l = 1;
```

```
@I/O DIRECTION
     "3-state output control of d7o-d0o"
   xe0 = (/setupl*/ssll*ss0l*/miol*/cmd*/a2l*outen);
   xel = (/setupl*/ssll*ss0l*/miol*/cmd*/a2l*outen);
   xe2 = (/setupl*/ssll*ss0l*/miol*/cmd*/a2l*outen);
   xe3 = (/setupl*/ssll*ss0l*/miol*/cmd*/a2l*outen);

@I/O STEERING
@LOGIC EQUATION

     " 7-Bit Input Latch for Control Signals "

   /setupl = /setup*/adl + /setupl*adl;
   /miol   = /mio */adl  + /miol  *adl;
   ssll    = ssl  */adl  + ssll   *adl;
   ss0l    = ss0  */adl  + ss0l   *adl;
   /a2l    = /a2  */adl  + /a2l   *adl;
   all     = al   */adl  + all    *adl;
   a0l     = a0   */adl  + a0l    *adl;

     " Option Select Octal Data Latch (POS byte #2) "
     " l0 is to be used as a card enable signal"

    /en  =  /[/setupl*/ss0l*ssll*/miol*/cmd*/a2l*all*/a0l];  "write  to
latch"

   /l7 = /[/d7i * en]  * /[l7 * /en] * [/rst];
   /l6 = /[/d6i * en]  * /[l6 * /en] * [/rst];
   /l5 = /[/d5i * en]  * /[l5 * /en] * [/rst];
   /l4 = /[/d4i * en]  * /[l4 * /en] * [/rst];
   l3 = /(/[ d3i * en * /rst] * /[l3 * /en  * /rst]);
   l2 = /(/[ d2i * en * /rst] * /[l2 * /en  * /rst]);
   l1 = /(/[ dli * en * /rst] * /[l1 * /en  * /rst]);
   l0 = /(/[ d0i * en * /rst] * /[l0 * /en  * /rst]);
```

```
  " Octal 3 to 1 Multiplexer "
  " This multiplexer selects between reading
    POS[0], POS[1] or POS[2] onto the data bus"

d7o = (b7h*read1 + b71*read0 + /17*read2);
d6o = (b6h*read1 + b61*read0 + /16*read2);
d5o = (b5h*read1 + b51*read0 + /15*read2);
d4o = (b4h*read1 + b41*read0 + /14*read2);
d3o = (b3h*read1 + b31*read0 +  13*read2);
d2o = (b2h*read1 + b21*read0 +  12*read2);
d1o = (b1h*read1 + b11*read0 +  11*read2);
d0o = (b0h*read1 + b01*read0 +  10*read2);
  "3-State output control for d7o-d0o"

outen =/[all*a01];

  "External F245 transceiver control"

iowb  = /(/a21 * /setup1 * /mio1 * ss11 * /ss01);
/iow  = /(/a21 * /setup1 * /mio1 * ss11 * /ss01);
bufen = cmd * /iow;
```

Figure 5-4 PLHS501 MCPOSREG .BEE File

6. NuBus Interface

In Apple Computer's book* "Designing Cards and Drivers for Macintosh II and Macintosh SE", an application was described for interfacing an 8-bit I/O controller to the NuBus. The controller used was a SCSI controller of the type used on the main Macintosh logic board. Seven devices (three of which were PAL architecture) were used as control circuitry interfacing the SCSI controller and two RAM chips to the bus.

This example of using the PLHS501 shows a method of interfacing the same SCSI controller and RAM chips to the NuBus using only three parts. The adapter card schematic is shown in Figure 6-2 and the AMAZE listing is in Figure 6-6. Although the AMAZE listing may seem confusing at first glance, the circuitry fused into the PLHS501 can be broken down into small blocks of latches, flip-flops, and schematically in Figure 6-4 and 6-5. Circuit timing is shown in Figure 6-3.

Referring to Figure 6-4 and Figure 6-5, the circuitry starts a transaction by first detecting a valid address in either the slot or super slot range. The detection is accomplished by two wide-input NAND gates, and controlled by the /CLK signal. Following each NAND gate is an S-R latch to hold the signal until near the end of the cycle. The two S-R latch signals are combined into one signal named ST0 such that if either NAND gate output was low, then some delay time after the rising edge of /CLK, ST0 will go low. The next rising edge of /CLK will cause signal ST1 to go low. This sets signal DE2 low, which is an input to an external flip-flop to cause ST2 to go low at the next rising /CLK edge terminating the cycle. An external flip-flop was necessary to achieve a high-speed /CLK to /IOR and /ACK transition. Also, an external FI25 buffer was added to meet the soon to be approved IEEE P1196 specification requirement of 60mA I_{OL} for signal /NMRQ and

*Designing Cards and Drivers for Macintosh II and Macintosh SE, Addison-Wesley Publish Company, Inc. 1987.

24mA I_{ol} for signals /TM0/TM1 and /ACK. Figure 6-5(b) shows an easily implemented latch which controls interrupts generated by the SCSI controller passing onto the bus. Upon /RESET the latch is put into a known state. Under software control, by writing to a decoded address, the latch may be set or reset thereby gating or blocking the interrupt signals.

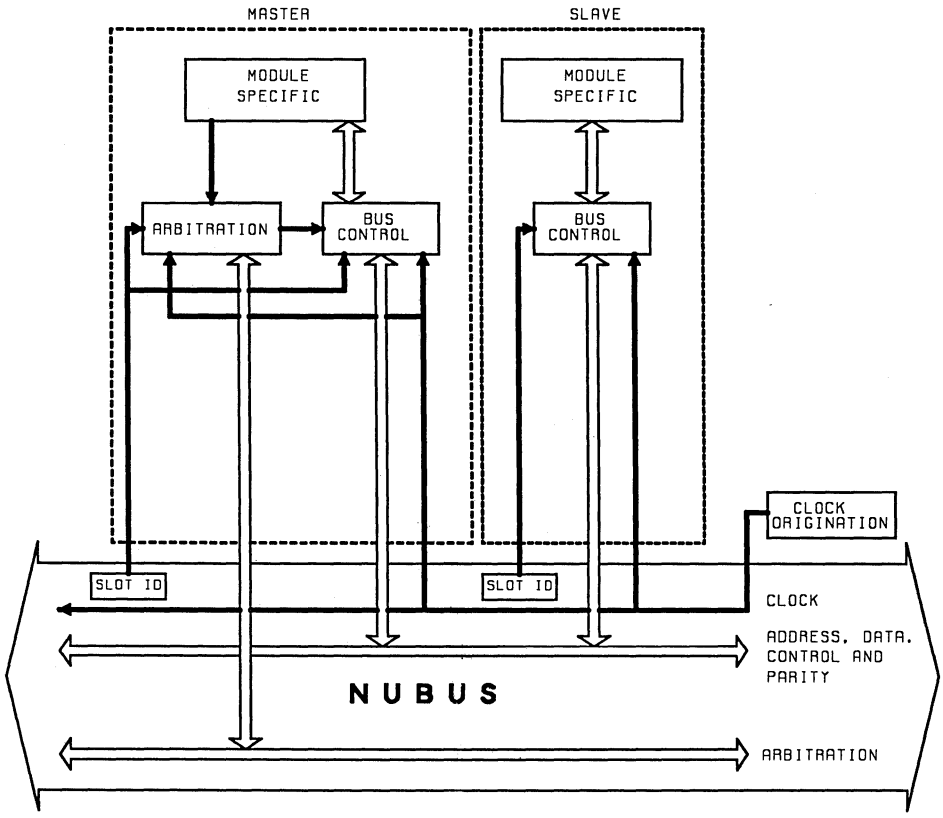


Figure 6-1 Simplified NuBus Diagram

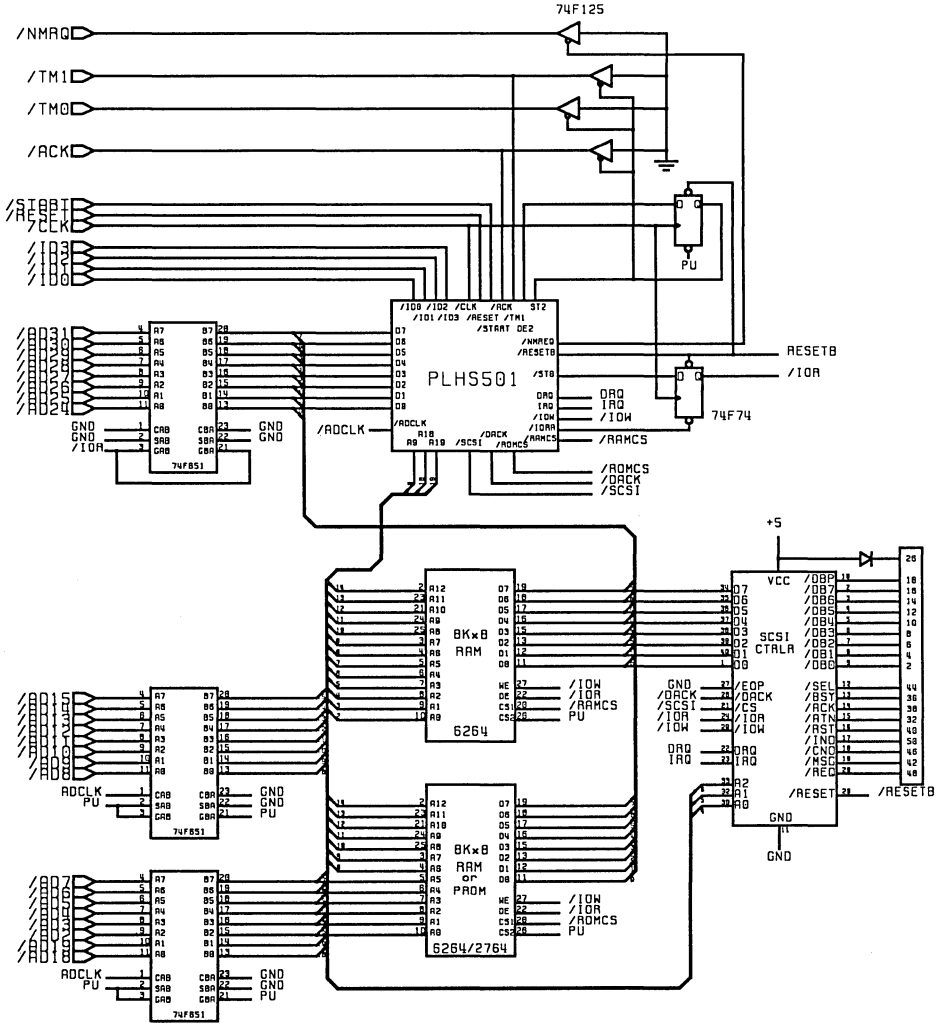


Figure 6–2 Adapter Card Schematic

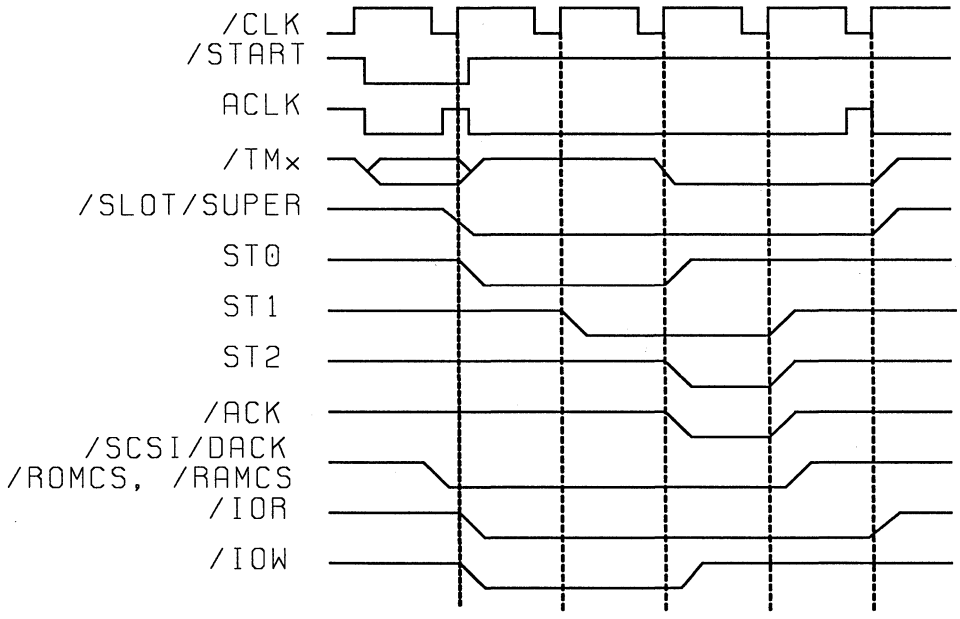


Figure 6–3 Timing Diagram

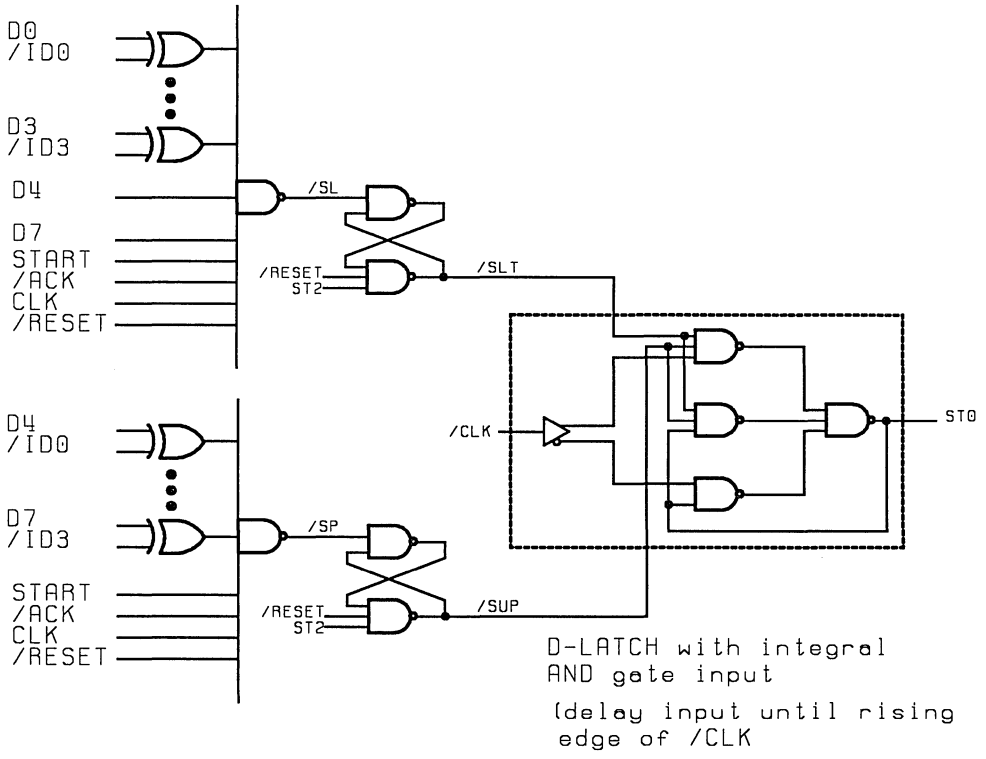


Figure 6–4 Decoding and Latch Circuitry

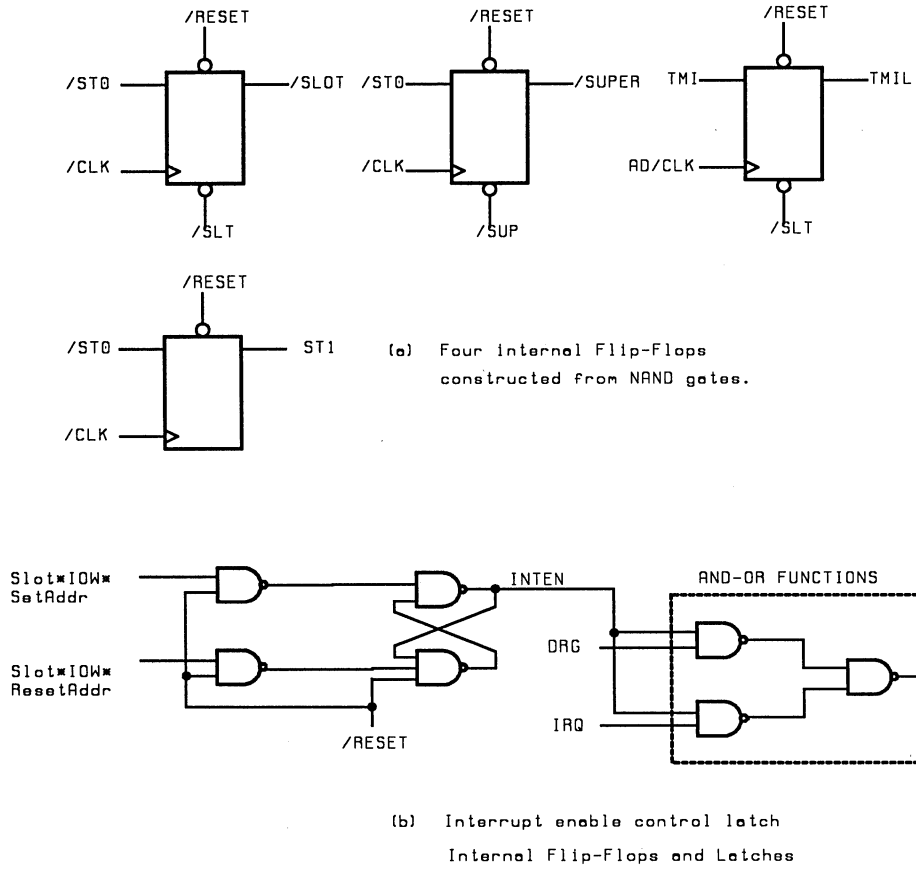


Figure 6–5 Internal Flip–Flops and Latches

```
#################### P I N   L I S T ####################

        Left                                          Right
   LABEL     ** FNC **PIN                PIN** FNC **    LABEL
VCC          ** +5V **  8-|          |-46 ** +5V **VCC
/ID2         ** I   **  9-|          |-45 ** I   **D1
/ID3         ** I   ** 10-|          |-44 ** I   **D0
DRQ          ** I   ** 11-|    P     |-43 ** I   **A19
IRQ          ** I   ** 12-|    L     |-42 ** I   **A18
ST2          ** I   ** 13-|    H     |-41 ** I   **A9
N/C          ** I   ** 14-|    S     |-40 ** /O  **ST0
N/C          ** O   ** 15-|    5     |-39 ** /O  **N/C
N/C          ** O   ** 16-|    0     |-38 ** /O  **N/C
N/C          ** O   ** 17-|    1     |-37 ** /O  **N/C
N/C          ** O   ** 18-|          |-36 ** O   **N/C
N/C          ** O   ** 19-|          |-35 ** O   **N/C
GND          ** 0V  ** 20-|          |-34 ** 0V  **GND

        Bottom                                        Top
   LABEL     ** FNC **PIN                PIN** FNC **    LABEL
N/C          ** O   ** 21-|          |- 7 ** I   **/ID1
N/C          ** O   ** 22-|          |- 6 ** I   **/ID0
ACLK         ** O   ** 23-|          |- 5 ** I   **/RESET
/ROMCS       ** /O  ** 24-|    P     |- 4 ** I   **/TM1
/RAMCS       ** /O  ** 25-|    L     |- 3 ** I   **/ACK
N/C          ** /O  ** 26-|    H     |- 2 ** I   **/START
/NMRQ        ** /O  ** 27-|    S     |- 1 ** I   **/CLK
DE2          ** O   ** 28-|    5     |-52 ** I   **D7
/RESETB      ** O   ** 29-|    0     |-51 ** I   **D6
/SCSI        ** O   ** 30-|    1     |-50 ** I   **D5
/DACK        ** O   ** 31-|          |-49 ** I   **D4
/IORR        ** O   ** 32-|          |-48 ** I   **D3
/IOW         ** O   ** 33-|          |-47 ** I   **D2
```

```
@DEVICE TYPE
   PLHS501
@DRAWING
@REVISION
@DATE
@SYMBOL
@COMPANY
@NAME
@DESCRIPTION
     SCSI-NuBus Interface
@INTERNAL NODE
     /sl,/sp,/SLOT,/SUPER;
snl,sn2,rnl,rn2;
sn3,rn3,stl;
sn4,rn4,tmll,tmlln;
CMP3a,CMP2a,CMPla,CMP0a;
CMP3b,CMP2b,CMPlb,CMP0b;
/slt,/sup,stln,adclk;
setad,rstad,inten;
slotn,supern;
@COMMON PRODUCT TERM
@I/O DIRECTION
@I/O STEERING
@LOGIC EQUATION

     "Address Decode"
cmp0a = (d0*id0+/d0*/id0);
cmpla = (dl*idl+/dl*/idl);
cmp2a = (d2*id2+/d2*/id2);
cmp3a = (d3*id3+/d3*/id3);
cmp0b = (d4*id0+/d4*/id0);
cmplb = (d5*idl+/d5*/idl);
cmp2b = (d6*id2+/d6*/id2);
cmp3b = (d7*id3+/d7*/id3);

/sl = /(d7*d6*d5*d4*cmp0a*cmpla*cmp2a*cmp3a*start*/ack*clk);
/sp = /(cmp0b*cmplb*cmp2b*cmp3b*start*/ack*clk);
     "latch slot signal"
/slt = /(/reset*st2*/[/sl*/slt]);
     "latch super signal"
/sup = /(/reset*st2*/[/sp*/sup]);
     "Let /slt or /sup through only
      until after the rising edge
      of /clk"
st0 = /(/[/slt*/sup*/clk] * /[st0*clk] * /[/slt*/sup*st0] * /reset);
```

```
     "Slot signal D-type Flip Flop"
snl = /(/clk*/slt*(/[snl*/reset*/super*(/[st0*rnl*/slt])]));
rnl = /(/clk*snl*(/[st0*rnl*/slt]));
/slot = /(/reset*/super*snl*slotn);
slotn = /(/slot*rnl*/slt);
     "Super signal D-type Flip Flop"
sn2 = /(/clk*/sup*(/[sn2*/reset*/slot*(/[st0*rn2*/sup])]));
rn2 = /(/clk*sn2*(/[st0*rn2*/sup]));
/super = /(/reset*/slot*sn2*supern);
supern = /(/super*rn2*/sup);
     "State 1 D-type Flip Flop"
sn3 = /(/clk*(/[sn3*/reset*(/[st0*rn3])]));
rn3 = /(/clk*sn3*(/[st0*rn3]));
stl = /(/reset*sn3*stln);
stln= /[stl*rn3];
     "output to external flop"
de2  = /(stln * st2);
     "address latch clock"
adclk = clk*st0*stl;
aclk  = clk*st0*stl;
     "latch tml signal for r/w info"
sn4 =  /(adclk*/reset*(/[sn4*(/[/tml*rn4*/reset])]));
rn4 =  /(adclk*sn4*(/[/tml*rn4*/reset]));
tmll = /(sn4*tmlln);
tmlln= /(rn4*/reset*tmll);
     "
     tmll  -> 1 read, 0 write
     tmlln -> 0 read, 1 write
     "
     "straight decode stuff"
/iorr = /(/st0*tmll                    * /reset);
/iow  = /(tmlln*/st0                   * /reset);
/scsi = /(slotn*/a19*/a18*/a9          * /reset);
/dack = /(slotn*/a19*/a18* a9          * /reset);
/romcs= /(slotn* a19* a18              * /reset);
/ramcs= /(supern                       * /reset);
/resetb= /reset;
     "interrupt control latch"
setad = /(tmlln*/st0*slotn* a19*/a18* a9);
rstad = /(tmlln*/st0*slotn* a19*/a18*/a9);
inten = /(setad*(/[inten*rstad*/reset]));
/nmrq = /(inten*drq+inten*irq);
```

Figure 6–6 AMAZE Listing

7. Nuggets

Much current focus for microprocessor design is on the address bus. Typically, most designers assume the processor will handle the data manipulation and the data bus is assumed to be a straight, clean path to and from the memory. Data transformations may be accomplished for specific purposes when the application requires it. For instance, a classic transformation from the early 70's was the bit reversal required to address operands for a Fast Fourier Transformation. When designers implemented bit reversal as a separate hardware process, the whole system improved. Likewise for hardware multipliers.

Also, a hidden "transformation" is the appending of parity and the calculation of E.C.C. polynominals. Clearly, when the designer recognizes that significant performance improvement can be achieved by realizing the payoff attainable with a special purpose hardware device, he should design it. For example, let's consider parity generation:

7.1 Data Bus Parity

The PLHS501 can span 32-bits of input data. It has four output EX-OR gates, and the ability to generate literally any function of the inputs. It would seem that there must be some "best" way to generate and detect parity. Recall that the PLHS501 can generate both deep logic functions (lots of levels) and wide logic functions (lots of inputs). The best solution would require the fewest gates and the fewest number of logic levels. Let's review the basics, first. Table 7-1(a) shows the parity function for two variables and Table 7-2(b) shows it for three variables. The EX-OR function generates even parity.

A	B	A ⊕ B
0	0	0
0	1	1
1	0	1
1	1	0

Table 7–1(a)

A	B	C	A⊕B⊕C
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

Table 7–1(b)

TABLE 7–1 Even Parity Functions

It is noticeable that there are precisely 50% logical 1 entries in the truth tables. This yields the famous checkerboard Karnaugh Maps. With a checkerboard K–map, no simplification of EX–OR functions is possible by Boolean simplification. The two variable EX–OR has two ones (implying 3 gates to generate), the 3 variable has four ones (implying 5 gates to generate). In general, $2^{n-1}+1$ product terms could generate EX_OR functions in two levels of NAND gates (assuming complementary input variables exist). You must have an unlimited number of gate inputs for this to hold.

The PLHS501 could do this for 7 input variables in two levels ($2^6+1=65$), but cannot support 8 ($2^7+1=129$). Hence, it is appropriate to seek a cascaded solution, hopefully taking advantage of the available output EX–OR functions. Let's solve a 16 input EX–OR function, by subpartitioning. First, consider Figure 7–1(a) where two literals are exclusive–ORed to generate an intermediate EX–OR function. This requires available complementary inputs and generates even parity in two levels. Figure 7–1(b) also does this (by factoring), requiring 3 gate levels, but does not require complementary inputs.

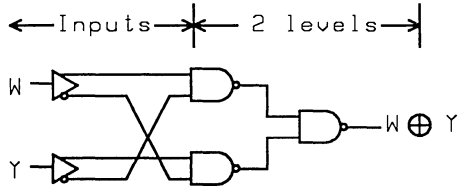


Figure 7-1 (a)

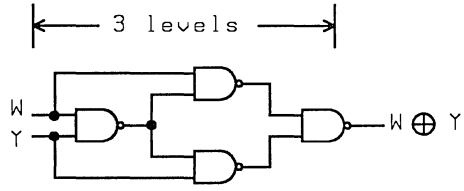


Figure 7-1 (b)

Figure 7-1 Complementary Input Levels

Assuming inputs must get into the PLHS501 through the pin receivers, it is best to generate as wide of an initial EX-OR as possible, so a structure like Figure 7-1(a) expanded is appropriate. Figure 7-1 shows a 2-level 4 input EX-OR function which may be viewed as a building block. This structure may be repeated four times, across four sets of four input bits generating partial intermediate parity values which may then be treated through two boxes similar to Figure 7-1(b). These outputs are finally combined through an output EX-OR at a PLHS501 output pin. Figure 7-3 shows the complete solution which requires 44 NANDs plus one EX-OR.

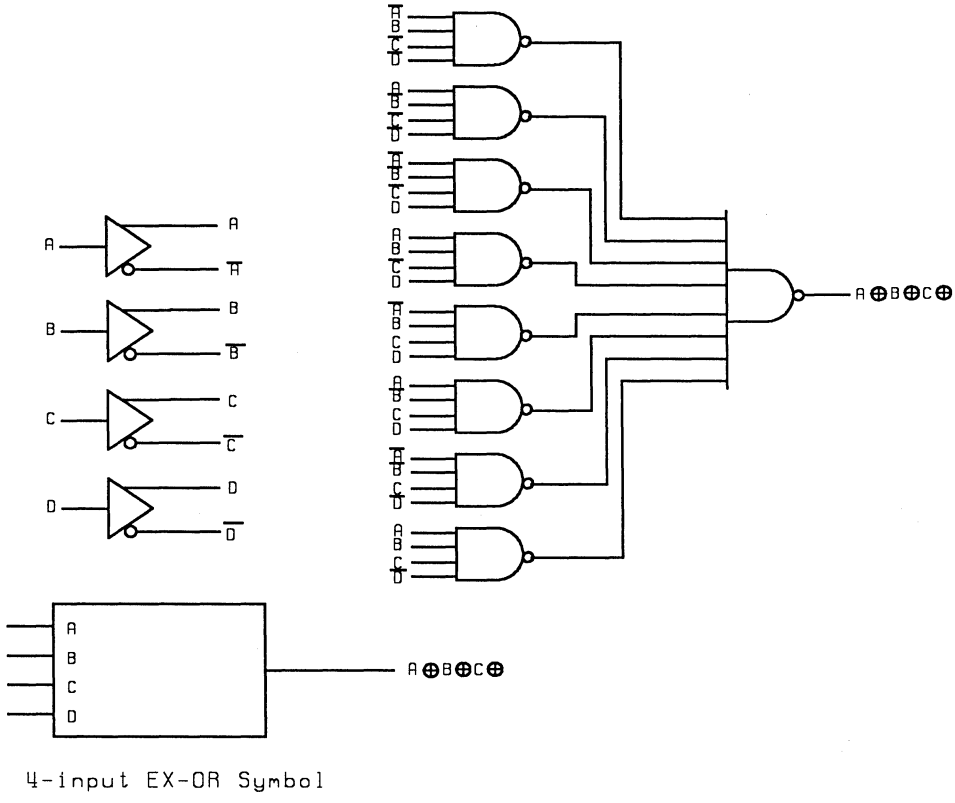


Figure 7–2 Four Variable EX–ORs

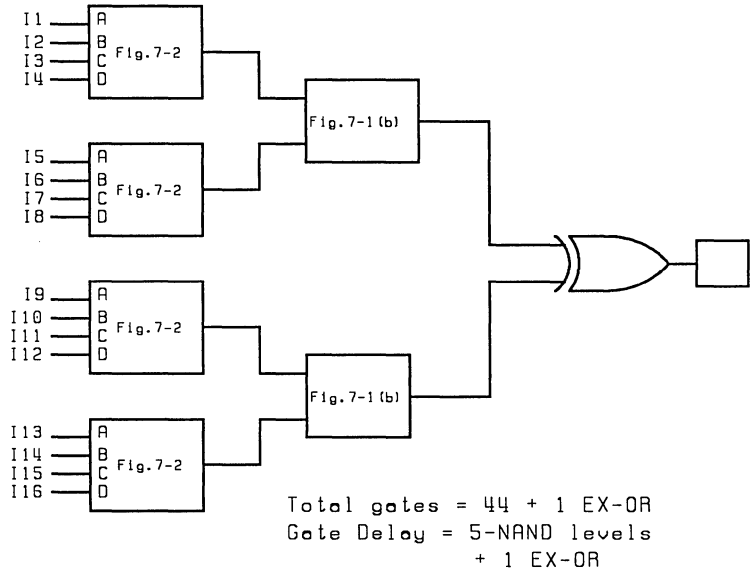


Figure 7–3 16 Input Even Parity Generation

Two examples follow which were supplied by one of our European Sales Engineers, Nils Lindgren. The first, called "paritet", calculates even and odd parity for 24 input literals. Several output options are available and the design uses a cascade with a different partitioning than just previously discussed.

The second example "compare" implements, a 16–bit comparator over 32 input bits. The design generates outputs for conditions representing the classic "EQUAL", "AGTB" (A>B) and BGTA (B>A). The long, triangularized equation for T42 suggests that Nils found a clever editing approach to accurately enter a relatively long design equation, into Signetics AMAZE.

```
File Name : PARITET
Date : 5/31/1988
Time : 10:26:22

#################### P I N   L I S T ####################

        Left                                          Right
   LABEL     ** FNC **PIN          PIN** FNC **    LABEL
VCC          ** +5V **  8-|      |-46 ** +5V **VCC
A            ** I   **  9-|      |-45 ** I   **K
B            ** I   ** 10-|      |-44 ** I   **J
C            ** I   ** 11-|   P  |-43 ** I   **I
D            ** I   ** 12-|   L  |-42 ** I   **H
E            ** I   ** 13-|   H  |-41 ** I   **G
F            ** I   ** 14-|   S  |-40 ** /O  **N/C
N/C          ** B   ** 15-|   5  |-39 ** /O  **N/C
N/C          ** B   ** 16-|   0  |-38 ** /O  **N/C
N/C          ** B   ** 17-|   1  |-37 ** I   **OEN
N/C          ** B   ** 18-|      |-36 ** O   **N/C
N/C          ** O   ** 19-|      |-35 ** O   **N/C
GND          ** 0V  ** 20-|      |-34 ** 0V  **GND

        Bottom                                        Top
   LABEL     ** FNC **PIN          PIN** FNC **    LABEL
N/C          ** O   ** 21-|      |- 7 ** I   **Y
N/C          ** O   ** 22-|      |- 6 ** I   **X
ODD_OC       ** O   ** 23-|      |- 5 ** I   **V
ODD          ** /O  ** 24-|   P  |- 4 ** I   **U
EVEN         ** /O  ** 25-|   L  |- 3 ** I   **T
EVEN_OC      ** /O  ** 26-|   H  |- 2 ** I   **S
N/C          ** /O  ** 27-|   S  |- 1 ** I   **R
N/C          ** O   ** 28-|   5  |-52 ** I   **Q
N/C          ** O   ** 29-|   0  |-51 ** I   **P
N/C          ** O   ** 30-|   1  |-50 ** I   **O
N/C          ** O   ** 31-|      |-49 ** I   **N
N/C          ** O   ** 32-|      |-48 ** I   **M
N/C          ** O   ** 33-|      |-47 ** I   **L
```

Figure 7–4 PARITET PLHS501 Pinlist

```
File Name : PARITET
@DEVICE TYPE
   PLHS501
@DRAWING
@REVISION
@DATE
   1988
@SYMBOL
@COMPANY
   Philips
@NAME
   Nils Lindgren
@DESCRIPTION
     24 bit parity circuit
@INTERNAL NODE
   J0 J1 J2 J3 J4 J5 J6 J7 J8 J9 T0 T1 T2 T3
@COMMON PRODUCT TERM
@I/O DIRECTION
   OE1=T2*T3*/OEN;
   OE2=/OEN;
   OE3=T0*T1*/OEN;
@I/O STEERING
@LOGIC EQUATION
     "FIRST LEVEL: 'EVEN' FROM GROUPS OF THREE INPUTS"
   J0=/A*/B*/C + /A*B*C + A*/B*C + A*B*/C;
   J1=/D*/E*/F + /D*E*F + D*/E*F + D*E*/F;
   J2=/G*/H*/I + /G*H*I + G*/H*I + G*H*/I;
   J3=/J*/K*/L + /J*K*L + J*/K*L + J*K*/L;
   J4=/M*/N*/O + /M*N*O + M*/N*O + M*N*/O;
   J5=/P*/Q*/R + /P*Q*R + P*/Q*R + P*Q*/R;
   J6=/S*/T*/U + /S*T*U + S*/T*U + S*T*/U;
   J7=/V*/X*/Y + /V*X*Y + V*/X*Y + V*X*/Y;
     "SECOND LEVEL: 'EVEN' FROM FOUR GROUPS AT A TIME"
   J8=/J0*/J1*/J2*/J3 + /J0*/J1*J2*J3 + J0*J1*/J2*/J3 + /J0*J1*J2*J3
      + J0*/J1*/J2*J3 + /J0*J1*/J2*J3 + J0*/J1*J2*/J3 + J0*J1*J2*J3;
   J9=/J4*/J5*/J6*/J7 + /J4*/J5*J6*J7 + J4*J5*/J6*/J7 + /J4*J5*J6*/J7
      + J4*/J5*/J6*J7 + /J4*J5*/J6*J7 + J4*/J5*J6*/J7 + J4*J5*J6*J7;
   T0=/(J8*J9);
   T1=/(/J8*/J9);
   T2=/(J8*/J9);
   T3=/(/J8*J9);
   ODD=/(T2*T3);
   EVEN=/(T0*T1);
   ODD_OC=0;
   EVEN_OC=/(1);
```

Figure 7-5 PARITET PLHS501 .BEE File

```
File Name : compare
Date : 5/31/1988
Time : 10:25:29

#################### P I N   L I S T ####################

        Left                                          Right
   LABEL     ** FNC **PIN          PIN** FNC **     LABEL
VCC          ** +5V **  8-|        |-46 ** +5V **VCC
A0           ** I   **  9-|        |-45 ** I   **B2
A1           ** I   ** 10-|        |-44 ** I   **B1
A2           ** I   ** 11-|   P    |-43 ** I   **B0
A3           ** I   ** 12-|   L    |-42 ** I   **AF
A4           ** I   ** 13-|   H    |-41 ** I   **AE
A5           ** I   ** 14-|   S    |-40 ** I   **AD
A6           ** I   ** 15-|   5    |-39 ** I   **AC
A7           ** I   ** 16-|   0    |-38 ** I   **AB
A8           ** I   ** 17-|   1    |-37 ** I   **AA
A9           ** I   ** 18-|        |-36 ** O   **N/C
N/C          ** O   ** 19-|        |-35 ** O   **N/C
GND          ** 0V  ** 20-|        |-34 ** 0V  **GND

        Bottom                                        Top
   LABEL     ** FNC **PIN          PIN** FNC **     LABEL
EQUAL        ** O   ** 21-|        |- 7 ** I   **BF
AGTB         ** O   ** 22-|        |- 6 ** I   **BE
BGTA         ** O   ** 23-|        |- 5 ** I   **BD
N/C          ** /O  ** 24-|   P    |- 4 ** I   **BC
N/C          ** /O  ** 25-|   L    |- 3 ** I   **BB
N/C          ** /O  ** 26-|   H    |- 2 ** I   **BA
N/C          ** /O  ** 27-|   S    |- 1 ** I   **B9
N/C          ** O   ** 28-|   5    |-52 ** I   **B8
N/C          ** O   ** 29-|   0    |-51 ** I   **B7
N/C          ** O   ** 30-|   1    |-50 ** I   **B6
N/C          ** O   ** 31-|        |-49 ** I   **B5
N/C          ** O   ** 32-|        |-48 ** I   **B4
N/C          ** O   ** 33-|        |-47 ** I   **B3
```

Figure 7-6 PLHS501 Pinlist for 16-Bit Comparator

```
File Name : compare
Date : 5/31/1988
Time : 10:25:43

@DEVICE TYPE
   PLHS501
@DRAWING
@REVISION
@DATE
@SYMBOL
@COMPANY
   PHILIPS
@NAME
   NILS LINDGREN
@DESCRIPTION
     16 BIT COMPARATOR WITH THREE OUTPUTS:
       EQUAL,AGTB (A>B), AND BGTA (B>A)
@INTERNAL NODE
   T1  T2  T3  T4  T5  T6  T7  T8
   T9  T10 T11 T12 T13 T14 T15 T16
   T17 T18 T19 T20 T21 T22 T23 T24
   T25 T26 T27 T28 T29 T30 T31 T32
   T41 T42
@COMMON PRODUCT TERM
@I/O DIRECTION
@I/O STEERING
@LOGIC EQUATION
   T1=/(AF*/BF);     T2=/(/AF*BF);
   T3=/(AE*/BE);     T4=/(/AE*BE);
   T5=/(AD*/BD);     T6=/(/AD*BD);
   T7=/(AC*/BC);     T8=/(/AC*BC);
   T9=/(AB*/BB);     T10=/(/AB*BB);
   T11=/(AA*/BA);    T12=/(/AA*BA);
   T13=/(A9*/B9);    T14=/(/A9*B9);
   T15=/(A8*/B8);    T16=/(/A8*B8);
   T17=/(A7*/B7);    T18=/(/A7*B7);
   T19=/(A6*/B6);    T20=/(/A6*B6);
   T21=/(A5*/B5);    T22=/(/A5*B5);
   T23=/(A4*/B4);    T24=/(/A4*B4);
   T25=/(A3*/B3);    T26=/(/A3*B3);
   T27=/(A2*/B2);    T28=/(/A2*B2);
   T29=/(A1*/B2);    T30=/(/A1*B1);
   T31=/(A0*/B0);    T32=/(/A0*B0);

   T41=T1*T2*T3*T4*T5*T6*T7*T8*T9*T10*T11*T12*T13*T14*T15*T16*T17*
      T18*T19*T20*T21*T22*T23*T24*T25*T26*T27*T28*T29*T30*T31*T32;
   T42=                                                          /T1+
```

```
                                                                    /T3*T2+
                                                                  /T5*T4*T2+
                                                                /T7*T6*T4*T2+
                                                              /T9*T8*T6*T4*T2+
                                                           /T11*T10*T8*T6*T4*T2+
                                                        /T13*T12*T10*T8*T6*T4*T2+
                                                     /T15*T14*T12*T10*T8*T6*T4*T2+
                                                  /T17*T16*T14*T12*T10*T8*T6*T4*T2+
                                               /T19*T18*T16*T14*T12*T10*T8*T6*T4*T2+
                                            /T21*T20*T18*T16*T14*T12*T10*T8*T6*T4*T2+
                                         /T23*T22*T20*T18*T16*T14*T12*T10*T8*T6*T4*T2+
                                      /T25*T24*T22*T20*T18*T16*T14*T12*T10*T8*T6*T4*T2+
                                   /T27*T26*T24*T22*T20*T18*T16*T14*T12*T10*T8*T6*T4*T2+
                               /T29*T28*T26*T24*T22*T20*T18*T16*T14*T12*T10*T8*T6*T4*T2+
                          /T31*T30*T28*T26*T24*T22*T20*T18*T16*T14*T12*T10*T8*T6*T4*T2;

EQUAL=T41;
AGTB=T42;
BGTA=/(T41+T42);
```

Figure 7–7 Compare PLHS501 .BEE File

7.2 Data Bus Operations

The following is basically an academic example, posed for the sake of illustration. Suppose some special data bus operations are desirable. For the purpose of illustration, let's label the microprocessor bus output side as ODAT0–ODAT15 and the output of our PLHS501 as D0–D15. Basically, the microprocessor will output straight data and the PLHS501 will alter it according to some plan.

We will replicate multiple identical cells, but they need not be identical in practice. Table 7–2 shows the operations to be done (just about any could be chosen, provided they meet the gate budget).

I_2	I_1	Dout
0	0	ODATI (pass)
0	1	ODATI (complement)
1	0	SWITCH
1	1	DOUBLE SHIFT

TABLE 7–2 Data Operations

The basic cell will require a structure as follows:

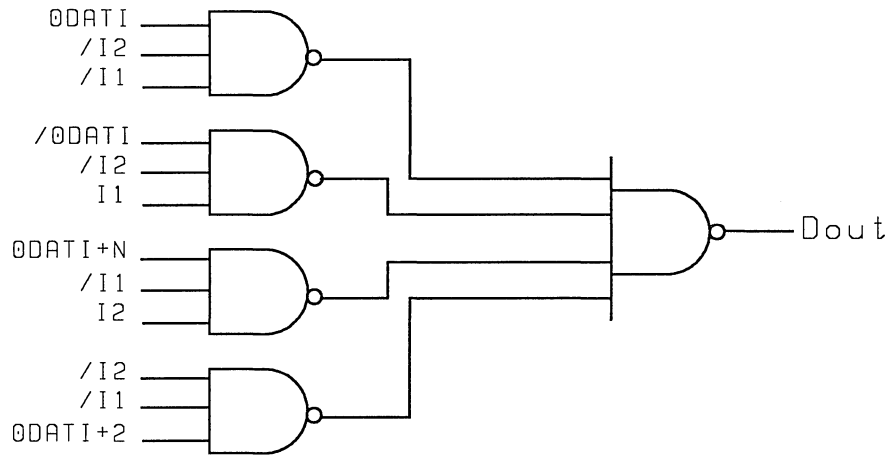


Figure 7–8 Basic Cell Structure

It may be observed that in one mode, the data passes directly, it complements in another, switches bits in another and rotates right in the last. Four input gates per bit are required to map the bits, and one output gate. Clearly, the straight PLHS501 NAND outputs can be judiciously used, but care must be taken when using other output functions. A 16–bit data bus requires 16 cell configuration where each cell is essentially identical to Figure 7–8 but, its internal structure may be altered to account for the particular output pins logic function.

Signetics

a division of North American Philips Corporation

Signetics Company
811 E. Arques Avenue
P.O. Box 3409
Sunnyvale, California 94088-3409
Telephone 408/991-2000

98–7000–620

Printed in U.S.A. 9204A/SVP/10M/CR1/0489